

Rudi Kost



ASHTON-TATE



FRAMEWORK™

Für IBM-PCs und Kompatible

Praktische Anwendungen
mit dem integrierten Softwarepaket Framework
und seinen Bereichen Textverarbeitung,
Datenbank, Tabellenkalkulation und Grafik.

Enthalten:
Alle Übungsdateien, Makros und Beispielprogramme



Framework

Rudi Kost

 ASHTON·TATE



Für IBM-PCs und Kompatible

Praktische Anwendungen mit dem integrierten
Softwarepaket Framework und seinen
Bereichen Textverarbeitung, Datenbank,
Tabellenkalkulation und Grafik.

Markt&Technik Verlag AG

CIP-Titelaufnahme der Deutschen Bibliothek

Kost, Rudi :

Framework für IBM-PCs und Kompatible : prakt. Anwendungen mit d. integrierten
Softwarepaket Framework u. seinen Bereichen
Textverarbeitung, Datenbank, Tab.-Kalkulation u. Grafik / Rudi, Kost. Ashton-Tate. –
Haar bei München : Markt-u.-Technik-Verl., 1987.
ISBN 3-89090-500-5

Die Informationen in diesem Produkt werden ohne Rücksicht auf einen eventuellen Patentschutz veröffentlicht.

Warennamen werden ohne Gewährleistung der freien Verwendbarkeit benutzt.

Bei der Zusammenstellung von Texten und Abbildungen wurde mit größter Sorgfalt vorgegangen.

Trotzdem können Fehler nicht vollständig ausgeschlossen werden.

Verlag, Herausgeber und Autoren können für fehlerhafte Angaben und deren Folgen weder eine juristische
Verantwortung noch irgendeine Haftung übernehmen.

Für Verbesserungsvorschläge und Hinweise auf Fehler sind Verlag und Herausgeber dankbar.

Alle Rechte vorbehalten, auch die der fotomechanischen Wiedergabe und der Speicherung in elektronischen Medien.

Die gewerbliche Nutzung der in diesem Produkt gezeigten Modelle und Arbeiten ist nicht zulässig.

Framework™ und FRED sind Warenzeichen von Ashton-Tate, USA
MS-DOS ist ein Warenzeichen der Microsoft Corp., USA

15 14 13 12 11 10 9 8 7 6 5 4
91 90 89

ISBN 3-89090-500-5

© 1988 by Markt&Technik Verlag Aktiengesellschaft,
Hans-Pinsel-Straße 2, D-8013 Haar bei München/West-Germany
Alle Rechte vorbehalten
Einbandgestaltung: Grafikdesign Heinz Rauner
Druck: Kösel, Kempten
Printed in Germany

Inhaltsverzeichnis

	Vorwort	7
	Framework und seine Möglichkeiten	9
1	Was ist Framework?	11
2	Was man mit Framework machen kann	14
3	Was man mit diesem Buch machen kann	16
	Mit Framework arbeiten	19
4	Wie man Framework installiert	21
5	Wie man Framework startet	28
6	Wie Framework den Speicher verwaltet	32
7	Wie Framework die Laufwerke anspricht	34
8	Wie man Framework anpaßt	36
	Framework - eine Einführung	43
9	Der Bildschirm	45
10	Die Menüs	47
11	Die Funktionstasten	49
12	Das Hilfe-System	50
13	Frames	51
14	Cursorbewegungen	64
15	Begriffe	65
16	Markieren, Kopieren und anderes	66
17	Daten- und Formeleingabe	71
18	Drucken	73
	Alltagsanwendungen	75
19	Vom Konzept zum Text	77
20	Vereinsverwaltung	92
21	Geschäftsbrieife (I)	106
22	Geschäftsbericht (I)	125
23	Verkaufsplanung	150
24	Überschuß-Berechnung (I)	155
	Die Kunst der Makros	165
25	Was ist ein Makro?	167
26	Makros erstellen	169
27	Die Makro-Bibliothek MACLIB	177

28	Eine eigene Makro-Bibliothek	183
29	Geschäftsbriefe (II)	185
30	Geschäftsbericht (II)	193
	FRED - Spaß am Programmieren	201
31	Was ist FRED?	203
32	Grundbegriffe	205
33	Die Programm-Bibliothek LIB	210
34	Überschuß-Berechnung (II)	222
	Arbeitserleichterungen	249
35	Startroutine	251
36	Tastatur-Bezeichnungen	252
37	Klangspiele	253
38	Das Demo-Makro	254
39	Sonderdienste	256
40	DOS-Zugriff	257
41	DISKUTIL	260
42	Übernahme aus Fremdprogrammen	263
43	Zeichnen mit FRED	267
	Framework - ein Überblick	269
	Bibliographie	299
	Stichwortverzeichnis	300
	Hinweise auf weitere Markt&Technik-Produkte	303
	Ausklappbare Menü-Übersicht	305

Vorwort

Unter den integrierten Software-Paketen hat Framework dank seiner Leistungsfähigkeit und Flexibilität rasch eine Vielzahl von Freunden gewonnen. Nun liegt auch eine Version vor, die dem schmaleren Geldbeutel gerecht wird – und dem Umstand, daß mittlerweile manche Software so teuer ist wie das Computersystem, auf dem sie laufen soll.

Framework ist Framework I, seit jüngstem wieder mit dem Kommunikationsteil, aber, als benutzerfreundliche Neuerung, ohne Kopierschutz. Der Leistungsumfang von Framework ist so gewaltig, daß für die Einarbeitung und die Ausnutzung der vielfältigen Möglichkeiten Hilfe geboten scheint.

Dem soll dieses Anwender-Handbuch dienen, und es verfolgt dabei zwei Zielrichtungen.

Vielleicht kennen Sie sich schon einigermaßen aus in Framework und wissen nur noch nicht so recht, wie Sie dieses Programmpaket für genau Ihre persönlichen Bedürfnisse am besten nutzen können. In diesem Buch erfahren Sie's.

Vielleicht haben Sie bisher auch nur in Framework hineingeschnuppert und, verwirrt von den vielen Funktionen und dem dicken Handbuch, resigniert Ihren PC abgeschaltet. Dieses Buch wird Ihnen der Leitfaden sein, der Sie in die Arbeit mit Framework einführt. Hernach werden Sie damit souverän umgehen können.

Für allfällige Fragen und zum Nachschlagen, damit Sie nicht immer im Handbuch blättern müssen, dienen zwei Teile dieses Buches: »Framework – eine Einführung« und »Framework – ein Überblick«. Im ersteren können Sie sich in einer Art Schnellkurs die Grundkenntnisse im Umgang mit Framework aneignen, im anderen können Sie gezielt nachschauen, was Sie tun müssen, um mit Framework etwas Bestimmtes zu erreichen.

Am meisten indessen wird interessieren, was man nun mit den grundsätzlichen Möglichkeiten von Framework konkret machen kann. Das ist der Schwerpunkt dieses Buches, orientiert an der Alltagsarbeit. Darum schien es auch gerechtfertigt, auf den Telekommunikationsteil von Framework ganz zu verzichten.

An Modellfällen werden einige Anwendungsbereiche aufgezeigt, von einer Adressenverwaltung über eine Verkaufsplanung mit einer »Was wäre, wenn ...«-Analyse bis hin zu einer kompletten Einnahme-Überschuß-Berechnung, und natürlich kommen auch Serienbriefe und die Gliederungsmöglichkeiten der Konzept-Version nicht zu kurz.

Die Beispiele sind modellhafte Lösungen – so aufgebaut, daß sie einerseits den Benutzer (den Laien wie den Fortgeschrittenen) immer tiefer in Framework hineinführen und er dabei alle Funktionen des Programmpaketes Schritt für Schritt kennenlernt. Auf der anderen Seite sind es bereits komplette, lauffähige Anwendungen, die, wie sie sind,

übernommen oder ohne großen Aufwand ausgebaut, abgeändert, variiert werden können.

Weil sämtliche Beispiele und Übungsdateien auf der beiliegenden Diskette enthalten sind, entfällt das sonst übliche, mühselige Abtippen von Tabellen, Texten oder Listings. Wer üben möchte – immer noch die beste Art, sich in ein Programm einzuarbeiten –, bekommt ausreichend Gelegenheit dazu; wer nicht, kann gleichwohl folgen.

Breiten Raum nehmen auch Makros und die Framework-Programmiersprache FRED ein, sind sie doch die eigentlichen Schlüssel für die Anpassung von Framework an die individuellen Bedürfnisse des Anwenders. Sie vereinfachen die Arbeit nicht nur, sie öffnen gleichsam einen Kosmos an Möglichkeiten, Framework zu nutzen.

Weil die Einführung in die Kunst der Makros und in FRED gleichzeitig eine Fortführung der zuvor erarbeiteten Beispiele ist, wird unmittelbar einsichtig, daß dies kein Tummelfeld für hartgesottene Computer-Freaks ist, sondern daß Makros und eigene Programme unmittelbare Bedeutung für die Alltagsarbeit haben können. Auch hier wieder: Modell-Lösungen, Handreichungen, Tips aus der Praxis. Und am Ende hoffentlich die Einsicht, daß das alles ja gar nicht so schlimm ist, wie man anfangs befürchtet hat.

Framework und seine Möglichkeiten

1 Was ist Framework?

Auch wenn's eine Binsenweisheit ist, um die jeder Käufer weiß (sonst hätte er sich nicht für dieses Programm entschieden): Framework ist ein integriertes Paket. Jedoch sollte man sich dessen stets bewußt sein und im Auge behalten, welche entscheidenden Vorteile integrierte Software mit sich bringt - und welche Nachteile.

Framework vereint vier unterschiedliche Anwendungsbereiche:

- Textverarbeitung
- Tabellenkalkulation
- Datenbank
- grafische Umsetzung von Daten

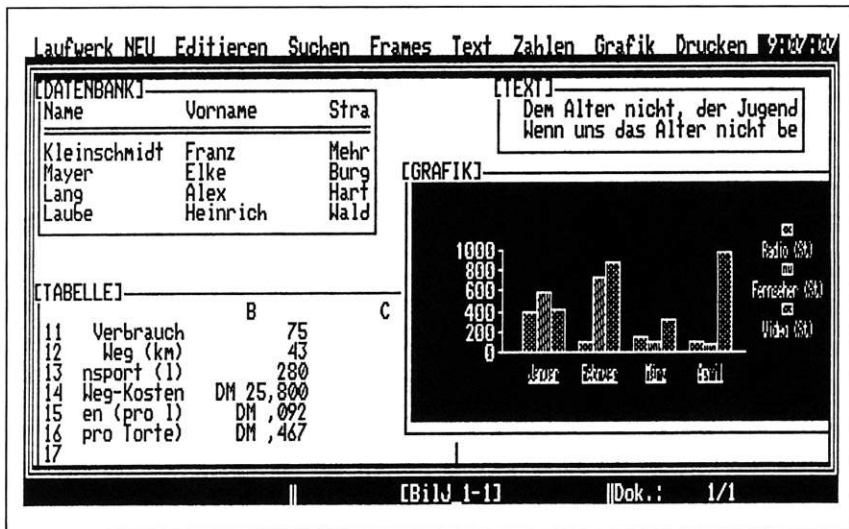


Bild 1.1: Die vier Anwendungsbereiche von Framework

Eine Menge Programme ließen sich aufzählen, die im jeweiligen Bereich mehr leisten als Framework. Es gibt genügend Textprogramme, die Frameworks Textverarbeitungsteil weit hinter sich lassen, und gleiches gilt für die Tabellenkalkulation und die Datenbank, von der Präsentationsgrafik ganz zu schweigen.

Indessen sind diese Programme nur in ihrem Bereich Meister - und versagen meist kläglich, wenn sie mit anderen Genres Daten austauschen sollen. Eine Datenbank mit einer Tabelle mischen, eine Tabelle, eine Grafik in einen Textteil übernehmen: Wo für

jeden Bereich mit einem eigenen Programm gearbeitet wird, ist das zumindest mühsam, oft sogar unmöglich.

Hier setzt Framework an. Der Sinn eines integrierten Paketes besteht darin, eben solche Zusammenführung von Daten nicht nur möglich, sondern auch leicht zu machen. Wer Bücher schreiben will, ist in der Tat mit einem ausgefuchsten Textprogramm besser beraten. Die Vorteile von Framework ausnutzen kann nur, wer mit verschiedenen Anwendungsbereichen zu tun hat.

Oder wer's bequem haben will. Da jedes Programm seine eigene Philosophie verfolgt, ist jedes auch anders zu bedienen. Wer öfters zwischen Programmen wechselt, weiß ein Lied davon zu singen. Die Tastaturbelegung ist anders, die Befehle sind unterschiedlich - und überhaupt ist schon so mancher Text auf Nimmerwiedersehen verschwunden, weil man in Gedanken noch bei einem ganz anderen Programm war.

Framework hat, wie man das so schön nennt, eine einheitliche Benutzeroberfläche. Alle Programmteile leben unterm Dach der gleichen Bedienung, und mithin ist keine Umstellung notwendig. Das macht ein integriertes Programm wie Framework auch für jene interessant, die nur gelegentlich mal eine Tabellenkalkulation brauchen. Nebenbei: Für solche Fälle reichen die Möglichkeiten der einzelnen Framework-Teile voll und ganz aus.

Framework ist, wörtlich übersetzt, Arbeit mit Rahmen. Jede Datei, die man aufruft oder bearbeitet, befindet sich in einem Frame, einem Rahmen - anderswo sagt man Fenster dazu. Solche Fenster sind nicht modisches Accessoire, sie sind gleichsam die Philosophie von Framework. Und sie sind flexibel zu handhaben:

- Frames können in nahezu beliebiger Zahl auf die Arbeitsfläche, den Bildschirm, geholt werden - nur die Kapazität des Arbeitsspeichers setzt Grenzen.
- Frames können in ihrer Größe und ihrer Position auf der Arbeitsfläche verändert werden.
- Frames können sich überlappen oder verdecken. Man kann sie auch verstecken: Obwohl man sie nicht sieht, sind sie da.
- Frames können ineinander verschachtelt werden.
- Frames können untereinander Daten austauschen: Man kann von einem Frame in den andern Daten kopieren oder verlagern.
- Frames können in Beziehung zueinander gesetzt werden: Ein Frame übergibt an den andern Daten, mit denen dann dort weitergearbeitet werden kann.

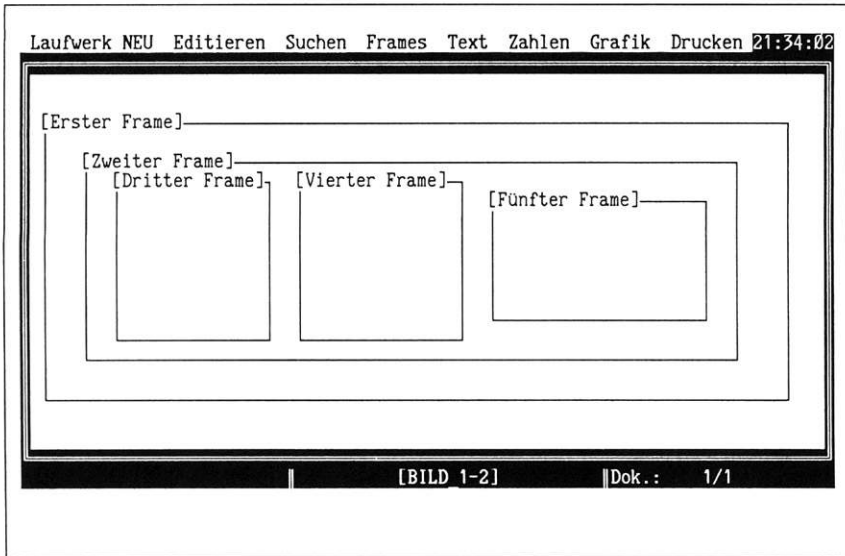


Bild 1.2: *Verschachtelte Frames*

Diese Frame-Prinzipien unterscheidet Framework grundsätzlich von Einzel-Anwendungsprogrammen und ziemlich stark von anderen Programmen, die gleichfalls mit Fenstern arbeiten oder sich auf andere Weise als integriert verstehen. Bei Framework ist die Integration tatsächlich verwirklicht. Alle Programmteile, alle Anwendungsbereiche sind stets sofort greifbar. Man muß nicht Programme wechseln, man braucht nur von einem Frame zum andern zu springen.

Was Framework von Hause aus mitbringt, ist schon mächtig genug. Wer noch tiefer einsteigen will, erarbeitet sich praktisch grenzenlose Möglichkeiten. Framework hat eine Programmiersprache integriert, mit dem amerikanischen Sinn für plastische Abkürzungen FRED genannt (was für *Frames Editor* steht), mit der man Framework gleichsam maßschneidern kann. Mit FRED kann man Arbeitsabläufe automatisieren, zum Beispiel mit Makros eine Vielzahl von Tastenanschlägen auf einen einzigen reduzieren, mit FRED kann man auch ganze Anwendungen entwickeln.

Nicht erschrecken vor dem Programmieren. So schlimm ist es gar nicht. Auch wer noch nie programmiert hat, kann lernen, mit FRED umzugehen. Zugegeben, wer FRED-Experte werden will, muß sich schon ordentlich hineinknien. Für einfache Anwendungen hingegen reicht guter Wille aus.

Und es lohnt sich. FRED ist das Bonbon, das die Alltagsarbeit versüßt, weil man sie vereinfachen kann.

2 Was man mit Framework machen kann

Da ist zum Beispiel der Sachbearbeiter Herwig Moritz. Er soll für den Chef einen Bericht über den Geschäftsverlauf schreiben. Er macht sich Frameworks Konzept-Funktion zunutze und erstellt erst eine Gliederung, die er sodann mit seinen Gedanken füllt: vom Konzept zum Text.

Der Chef will natürlich nicht nur Argumente hören, er will handfeste Zahlen sehen. Herwig Moritz hat die Umsatzzahlen in einer Tabelle erfaßt, die er, wie's dem Chef gerade einfällt, nach verschiedenen Kriterien auswertet. Das Zahlenmaterial integriert er in seinen Bericht und läßt es zudem von Fall zu Fall in Grafiken umsetzen, weil das so schön repräsentativ ist.

Die Sekretärin Katharina Kröll war es leid, von früh bis spät Briefe zu schreiben, die einander ähneln wie ein Osterei dem andern: immer die gleichen Adressen, immer die gleichen Floskeln. Deshalb hat man Framework angeschafft. Die Adressen hat sie in einer Datenbank, die Floskeln jeweils eigens abgespeichert, und ihre Briefe setzt sie nunmehr wie ein Puzzle zusammen. Daß das ihre Arbeit erleichtert, sagt sie jedem; daß sie jetzt viel mehr Zeit hat, behält sie lieber für sich. Sonst könnte der Personalchef noch auf dumme Gedanken kommen ...

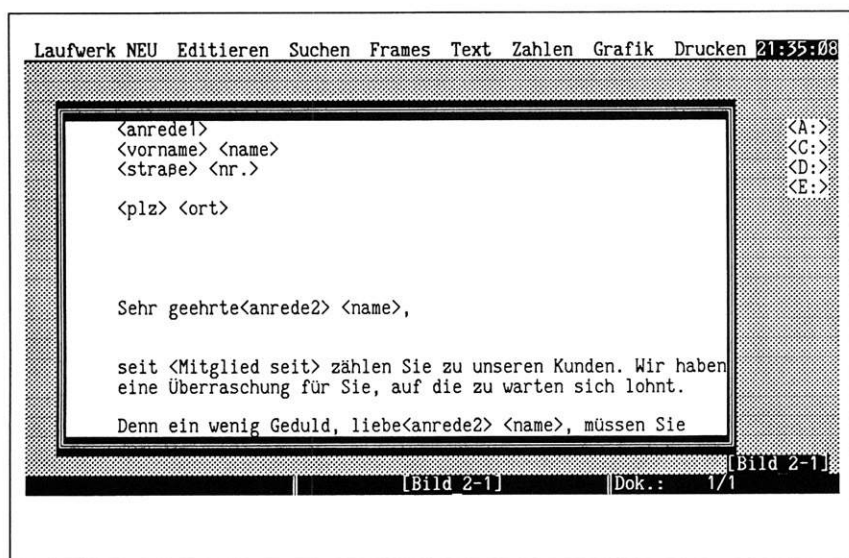


Bild 2.1: Serienbriefe mit Framework

Der Bäckermeister Werner Pfisterer, der seine Gewinnspanne für eher kümmerlich hält, macht jetzt mit der Tabellenkalkulation Nägel mit Köpfen. Wenn er sein Mehl um fünf Pfennige billiger einkauft, wie schlägt das zu Buche? Erhöht es seinen Gewinn, wenn er den Preis der Sahnetorten herabsetzt, damit sie sich besser verkaufen? Schlägt sich's in der Bilanz entscheidend nieder, wenn er ein bißchen schummelt und bei jedem Brötchen nur ein paar Gramm Teig einspart? Er macht also eine Betriebsergebnisanalyse und spielt durch, was sich insgesamt tut, wenn sich ein Posten ändert.

Der Schriftsteller Jonathan Kraft weiß, daß er Framework nicht nur zum Dichten brauchen kann. Er hat sich eine Einnahmen-Überschuß-Berechnung zusammengebastelt; zwar nur eine einfache (was kompliziert genug ist), aber sie orientiert ihn wenigstens jederzeit über die aktuelle Finanzsituation. Und seitdem hat auch der Steuererklärungs-termin viel von seinem Schrecken verloren.

Lothar Wanner, Erster Vorsitzender des ruhm- und traditionsreichen TSV Schanwald, führt seine Mitgliederliste mit Framework und hat damit auch stets im Blick, wer seinen Jahresbeitrag noch schuldig ist. Und diese peinliche Sache damals, als er doch glatt übersehen hat, daß Schreinermeister Brösele 25 Jahre Mitglied ist: Die wird ihm nicht nochmals passieren.

Das alles kann man mit Framework machen - und noch viel mehr. Der Phantasie des Anwenders sind keine, den Möglichkeiten kaum Grenzen gesetzt.

Laufwerk NEU Editieren Suchen Frames Text Zahlen Grafik Drucken 21:55:30						
Name	Vorname	Strasse	Nr.	PLZ	Ort	Geburtsjahr
Kleinschmidt	Franz	Mehringerweg	87	7307	Schanbach	1949
Mayer	Elke	Burgstraße	8	8590	Mahnstadt	1900
Lang	Alex	Hartliebstraße	1	8000	München	1951
Laube	Heinrich	Waldallee	71	3456	Zwilling	1948
Durchholz	Karl	Ariboweg	14	7000	Stuttgart	1953
Jungmann	Dorothea	Erlbachstraße	42	3619	Hohenberg	1956
Modlinger	Maria	Memminger Platz	31	8000	München	1949
Alexis	Steffen	Weidenweg	6	7000	Stuttgart	1917
Schäfer	Gerhard	Hauptstraße	22	7210	Bramme	1955
Lessing	Gotthold	Burgstraße	31	7000	Stuttgart	1928
Schlote	Reinhold	Breslauer Straße	45	7891	Wurmling	1952
Metzger	Hanns-Hermann	Hauptstraße	7	7629	Klagenfu	1952
Varner	Eduard	Anemonenweg	31	7000	Stuttgart	1959
Beck	Harald	Seestraße	27	7307	Schanbach	1946
Croner	Felix	Jägerweg	22	8000	München	1950

Strasse="Burgstraße" [Bild 2-2] Dok.: 1/1 [Bild 2-2]

Bild 2.2: Eine Datenbank mit Framework

3 Was man mit diesem Buch machen kann

Wahrscheinlich ist Ihnen Framework, wenn Sie dieses Buch in Händen halten, nicht mehr gänzlich fremd. Und wenn doch, nehmen Sie sich den Teil »Framework - eine Einführung« vor, um hernach gerüstet zu sein, mit Framework zu arbeiten.

Tip: Befassen Sie sich zunächst einmal mit dem Framework-Tutorial; das verschafft einen ersten Einblick und ist überdies recht amüsant.

Spätestens jetzt, nachdem Sie die grundsätzlichen Möglichkeiten von Framework kennen, werden Sie sich fragen, was Sie - Sie persönlich! - damit in der Praxis - in Ihrer Praxis! - anfangen können.

Das wissen, leider, nur Sie allein. Weil nur Sie wissen, wozu Sie Framework einsetzen wollen. Darum kann ich Ihnen allenfalls Anregungen geben. Und deshalb geht dieses Buch von konkreten Beispielen aus, von Beispielen aus dem Alltag.

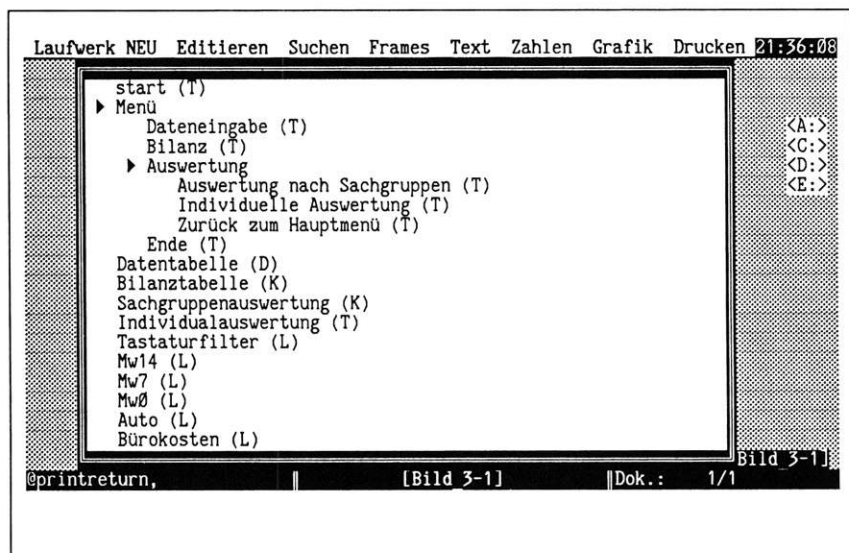


Bild 3.1: Ein Modell-Programm: Einnahmen-Überschuß-Berechnung mit Framework

Die Beispiele sind Modellfälle. Sie zeigen, was mit Framework möglich ist und weisen die Wege, mit denen man zu Lösungen gelangt. Es sind keineswegs immer die elegantesten Lösungen - weil es ums Prinzip geht; darum, wie man sich die vielfältigen Möglichkeiten von Framework zunutze machen kann. Und um der Natur von Framework gerecht zu werden, stehen die integrierten Lösungen im Vordergrund.

Die Modellfälle zeigen eine konkrete Problemlösung auf und führen gleichsam nebenbei in den Umgang mit Framework ein - oder erinnern an grundsätzliche Dinge, die man zu Anfang leicht vergißt, weil Framework gar zu umfangreich ist. Und dann beginnt das hektische Blättern: Wie schaltet man nochmals von der Koordinatendarstellung auf die Feldbezeichnungen um? An den geeigneten Stellen werden Sie daran erinnert. Mehrmals, wenn's sein muß.

Merke: An dieser Stelle finden Sie ergänzende Erläuterungen und Tips aus der Praxis.

Die vorgestellten Beispiele können teilweise direkt in die eigene Arbeit übernommen werden - wenn Sie mit den gleichen Problemen zu kämpfen haben. Sie sind in jedem Fall so ausgewählt, daß sie sich ohne Mühe übertragen lassen - vom Bäcker- auf den Metzgermeister, vom Sachbearbeiter auf den Vorstandsvorsitzenden. Und sie lassen sich erweitern und variieren. Das ist auch das große Plus von Framework: Es ist so flexibel, daß nahezu jedes Problem gelöst werden kann. Irgendwie.

Die Beispiele sollen auch Ihre Phantasie anregen: Wie kann man's noch besser, noch schneller, noch eleganter machen? Und das Fieber wird Sie packen, je tiefer Sie sich in Framework hineinarbeiten.

Deshalb werden einige der Modellfälle weiter differenziert. Auch für normale Anwendungen ist Framework schon recht mächtig. Doch wenn man sich erst einmal mit Makros und mit FRED, der Framework-Programmiersprache, befreundet hat, läßt sich das Programm noch weitaus besser nutzen. Auch hierfür finden Sie große Lösungen und nützliche Kleinigkeiten, die Sie übernehmen können - oder auf Ihren Bedarf abändern.

Sie werden auf diese Weise auch Schritt für Schritt in die Geheimnisse von Makros und FRED eingeführt. Allerdings werden wir nicht sämtliche Funktionen von FRED besprechen können. Das würde den Rahmen dieses Buches sprengen. Wir kratzen an der Oberfläche, graben hie und da tiefer, und das wird Sie in die Lage versetzen, selbständig weiterzumachen.*

* Weitere Hilfe dazu finden Sie in einem Framework-Programmierhandbuch, das beim Verlag Markt & Technik in Vorbereitung ist

Mit Framework arbeiten

4 Wie man Framework installiert

Zunächst sollten Sie sich von allen Framework-Disketten (und übrigens auch von der Diskette mit Übungsbeispielen, die diesem Buch beiliegt) Kopien anfertigen und die Originale weitab von jeglichen Gefährdungen wie magnetischen Gegenständen, neugierigen Kinderhänden, Kaffee und Vesper aufbewahren. Wenn mit Ihren Arbeitskopien etwas geschieht, können Sie jederzeit wieder neue Kopien anfertigen; sind Ihre Originale kaputt, haben Sie den größeren Schaden.

Für die ganz Unerfahrenen hier ein Hinweis, wie man Disketten kopiert:

1. Starten Sie Ihren Computer mit der Betriebssystem-Diskette (nicht mit Framework!). Bei Festplatten-Anlagen befindet sich das Betriebssystem üblicherweise auf der Festplatte – da genügt es, den Computer einzuschalten.

Tip: Versehen Sie alle Originaldisketten mit einem Schreibschutz (liegt jedem Paket neuer Disketten bei). So verhindern Sie versehentliches Überschreiben.

2. Geben Sie bei Systemen mit zwei Diskettenlaufwerken ein:

```
diskcopy a: b:
```

Bei Systemen mit nur einem Diskettenlaufwerk:

```
diskcopy
```

3. Entnehmen Sie die Betriebssystem-Diskette dem Laufwerk, legen Sie statt ihrer eine der Framework-Disketten in das linke Laufwerk A (die Quell-Diskette) und in das rechte Laufwerk B eine neue Diskette (die Ziel-Diskette), die nicht formatiert zu sein braucht. Bei nur einem Laufwerk müssen Sie die Disketten nacheinander einführen; wann Sie wechseln müssen, meldet Ihnen das Betriebssystem.

Wer mit Diskettensystemen arbeitet, braucht zudem noch Disketten, die seine Daten aufnehmen. Diese Disketten müssen aber zuvor formatiert werden mit dem Befehl

```
format a: b:
```

Die zu formatierenden Disketten werden in Laufwerk B gelegt. In Laufwerk A befindet sich die Betriebssystem-Diskette.

Es schadet nichts, gleich mehrere Disketten zu formatieren, sie haben dann immer welche zur Hand. Auch Festplatten-Besitzer benötigen sie: zur Datensicherung.

Installation auf Disketten-Systemen

Für Disketten-Systeme ist damit die Arbeit schon erledigt; Sie müssen nur noch einen Drucker installieren und gegebenenfalls den Monitor anpassen (kommt weiter unten). Sie können aber noch Weiteres tun, das Ihnen die Arbeit womöglich erleichtert.

Sie können die Systemdiskette 1 so einrichten, daß Sie mit ihr den Computer starten können (»booten«, sagt der Fachmann), ohne zuvor die Betriebssystemdiskette bemühen zu müssen.

1. Legen Sie die Betriebssystemdiskette in Laufwerk A, die Systemdiskette 1 in Laufwerk B und geben ein:

```
sys b:
```

Damit werden die versteckten Betriebssystem-Dateien, ohne die der Computer nicht gestartet werden kann, auf die Diskette in Laufwerk B kopiert.

2. Als nächstes kopieren Sie die internen Betriebssystem-Befehle, die das System beim Booten in den Arbeitsspeicher lädt:

```
copy command.com b:
```

Jetzt ist die Framework-Systemdiskette 1 proppenvoll, aber Sie können mit ihr nun »booten«.

Wahrscheinlich - das hängt von Ihrer Betriebssystem-Version ab - bekommen Sie auf diese Weise nur eine amerikanische Tastaturbelegung, und Sie müssen zuerst die deutsche installieren.

Tip: Starten Sie den Computer mit der Framework-Systemdiskette, und drücken Sie dann die Taste Y. Wird auf dem Bildschirm tatsächlich ein Y ausgegeben, ist alles in Ordnung. Bekommen Sie stattdessen ein Z, ist's die amerikanische Tastaturbelegung.

1. Dazu kopieren Sie von der Betriebssystemdiskette die Datei KEYBGR.EXE (oder KEYBGR.COM, je nach Version) auf die Framework-Systemdiskette 1 und erstellen anschließend eine AUTOEXEC-Datei, am einfachsten folgendermaßen, in allen Schritten erklärt, damit ja nichts schiefgeht:
2. Schreiben Sie

```
copy con b:autoexec.bat
```
3. Drücken Sie **RETURN**
4. Schreiben Sie, und drücken Sie nach jeder Zeile wiederum **RETURN**:

```
keybgr
fw
```
5. Drücken Sie die Funktionstaste **F6** (auf dem Bildschirm erscheint ^Z) und anschließend **RETURN**.

Mit **copy con** »kopieren« Sie eine neue Datei - in diesem Fall AUTOEXEC.BAT - auf die Diskette in Laufwerk B, auf gut deutsch: Sie erstellen diese Datei.

AUTOEXEC.BAT wird beim Systemstart automatisch gesucht, und das Betriebssystem arbeitet ab, was da drinsteht. Sie haben angewiesen, daß zunächst die deutsche Tastaturbelegung KEYBGR initialisiert werden soll und anschließend, weil Sie nun schon mal dabei waren, gleich den Start-Befehl FW für Framework gegeben. Mit der Folge, daß künftig mit der Framework-Systemdiskette 1 nicht nur der Computer gestartet, sondern gleichzeitig auch Framework aufgerufen wird.

Jetzt ist nun wirklich kein Platz mehr auf der Diskette - vielleicht war's auch vorher schon zu wenig, und Sie konnten die KEYBGR-Datei gar nicht kopieren.

In dem Fall installieren Sie zunächst den Monitor und den Drucker und löschen dann anschließend eine der folgenden Dateien (oder alle):

- DRUCKER.BAT
- INSTALL.BAT
- INSTALL.HLP
- MONITOR.BAT
- MONITOR.HLP
- PLATTE.BAT
- TUTORIAL.BAT

Tip: Sollten Sie diese Dateien später noch einmal benötigen (weil Sie vielleicht den Drucker wechseln), haben Sie ja noch das Original der Systemdiskette 1.

Auch auf der Systemdiskette 2 benötigen Sie COMMAND.COM, wenn Sie von Framework aus auf Betriebssystem-Kommandos zugreifen wollen (zum Beispiel, um Dateien zu kopieren) und sich dafür einen Diskettenwechsel ersparen möchten.

Entnehmen Sie also die Systemdiskette 1 dem Laufwerk B, legen stattdessen die Systemdiskette 2 ein und tippen nochmals den Copy-Befehl für COMMAND.COM.

Sie können noch mehr tun - müssen aber nicht. Nachdem Sie Monitor und Drucker installiert haben (nachdem!), löschen Sie auf Ihrer Arbeitskopie der Hilfsprogrammdiskette alle überflüssigen Dateien und benutzen diese dann als Ihre erste Datendiskette.

Sie löschen die Drucker- und Bildschirmtreiber sowie die Installations-Hilfsprogramme, und zwar alle auf einen Schlag, mit

```
del *.ger
del *.drv
del *.bat
```

Installation auf einer Festplatte

Seit Framework ohne Kopierschutz ausgeliefert wird, ist die Installation auf einer Festplatte kein Problem mehr. Früher war nur eine Installation möglich, diese Einschränkung ist jetzt weggefallen.

Zum Einrichten der Framework-Dateien auf einer Festplatte gibt es ein Installationsprogramm, vor dessen Start Sie jedoch ein Verzeichnis (ein Sub-Directory) einrichten müssen, dessen Namen Sie frei wählen können.

Gesetzt den Fall, er lautet FRAME, gehen Sie Schritt für Schritt so vor (wobei Sie bekanntlich nach jeder Eingabe **RETURN** drücken müssen):

1. Geben Sie ein:
`md frame`
2. Wechseln Sie über in dieses Verzeichnis mit
`cd frame`
3. Machen Sie das Laufwerk A zum aktuellen Laufwerk mit
`a:`
4. Legen Sie die Framework-Systemdiskette 1 in Laufwerk A.
5. Geben Sie ein:
`platte c:`
Hat Ihre Festplatte eine andere Laufwerks-Bezeichnung als C, müssen Sie hier diese angeben.

Nun werden die nötigen Dateien auf die Festplatte kopiert, und das Installationsprogramm sagt Ihnen, wann Sie welche Disketten wechseln müssen.

Das Installationsprogramm ist nicht ganz vollständig. Es vergißt, eine Datei zu kopieren - der »Errata«-Zettel im Handbuch macht darauf aufmerksam und sagt, wie dem abzuhelpen sei. Wenn es Ihnen zu kompliziert erscheint, die Datei PLATTE.BAT entsprechend abzuändern, erledigen Sie die Sache von Hand: mit

```
copy fwcoppy.com c:
```

kopieren Sie einfach die noch fehlende Datei von der Diskette auf die Festplatte (Framework benötigt FWCOPY.COM zur Installation des Druckers).

Nicht erfaßt vom Installationsprogramm werden die Dateien des Tutorials. Die müssen Sie eigens auf die Festplatte kopieren. Was nicht weiter schwierig ist:

1. Legen Sie die Framework-Tutorialdiskette in Laufwerk A.
2. Geben Sie ein:
`copy *.* c:`

Damit werden alle Dateien des Tutorials in Ihr Framework-Verzeichnis auf der Festplatte kopiert.

Einige Hinweise noch für diejenigen, die Framework in der kopiergeschützten Version kennen (oder haben). Im Gegensatz zu früher ist ein UNINSTAL von Framework nicht mehr notwendig. Auch die Framework-Programmdateien können jetzt, wenn das erforderlich ist, wie üblich mit den DOS-Befehlen COPY oder BACKUP gesichert werden. Das erleichtert die hin und wieder fällige Reorganisation der Festplatte, die nun kein Problem mehr darstellt, da Framework ja mehrfach installiert werden kann.

Wer mit einer kopiergeschützten Version von Framework arbeitet, muß allerdings nach wie vor zu dem Programm UNINSTAL (auf der Systemdiskette 1) greifen, wenn die Installation von Framework rückgängig gemacht werden soll. Das bloße Zurückkopieren der Programmdateien nützt nichts, und ein Backup zerstört sie!

Den Monitor installieren

Framework ist von Hause aus auf die IBM-Grafikkarte eingestellt und erkennt dabei von selber, ob es sich um einen Monochrom- oder Farbbildschirm handelt. Genauer gesagt: Framework ist das gleichgültig, es präsentiert sich ohnehin die meiste Zeit nur monochrom. Lediglich Grafiken kann man sich farbig ausgeben lassen, und wer sich tief in Frameworks Programmiersprache FRED arbeitet, kann zudem farbig zeichnen (lassen).

Alle Computersysteme, die einen normalen Grafik-Adapter verwenden, müssen daher nicht angepaßt werden - und das sind alle aus der Welt der Kompatiblen, die nicht ausdrücklich anders aufgerüstet sind.

Zur Wahl haben Sie noch

- Hercules-Karte
- AST MonoGraph Plus

Verwenden Sie diese Grafik-Adapter, müssen Sie Framework anpassen, um deren Möglichkeiten auch ausnutzen zu können:

1. Legen Sie die Systemdiskette 1 in Laufwerk A und die Hilfsprogramm-Diskette in Laufwerk B. Bei einer Festplatten-Anlage kommt die Hilfsprogramm-Diskette in Laufwerk A.
2. Tippen Sie, je nachdem, was auf Sie zutrifft:

bei Disketten-Systemen

```
monitor ibm b:
monitor hercules b:
monitor astmono b:
```

bei Festplatten-Anlagen

```
monitor ibm a:  
monitor hercules a:  
monitor astmono a:
```

Schreiben Sie nur `monitor`, sehen Sie einen kurzen Informationstext, der die eben beschriebenen Schritte erläutert: Sie müssen den Monitor-Typ angeben, gefolgt von dem Laufwerk, in welchem sich die Hilfsprogramm-Diskette befindet.

Drucker und Plotter installieren

Auch hierfür gibt es ein eigenes Hilfsprogramm, mit dem Sie, alles in allem, zwei Drucker und einen Plotter installieren können. Dazu kommt

- bei Disketten-Systemen die Systemdiskette 1 in Laufwerk A, die Hilfsprogrammdiskette in Laufwerk B
- bei einer Festplatte die Hilfsprogrammdiskette in Laufwerk A.

Sodann starten Sie das Installationsprogramm (Sie befinden sich immer noch auf Betriebssystem-Ebene!) mit

```
drucker
```

Gleichzeitig wird damit auch Framework aufgerufen, weshalb Sie bei Disketten-Systemen die Systemdiskette 1 gegen die Systemdiskette 2 tauschen müssen (wird Ihnen auf dem Bildschirm gesagt).

Das Installationsprogramm ist menügeführt, das heißt, Sie werden über die einzelnen Schritte, die Sie unternehmen müssen, informiert und können deshalb eigentlich nichts falsch machen. Und wenn doch: Beginnen Sie eben nochmals von vorne.

Framework offeriert Ihnen eine Liste von möglichen Druckern und Plottern. Natürlich wird, weil das ja immer so ist, ausgerechnet Ihrer nicht darunter sein. In diesem Fall wählen Sie einen, der zu Ihrem Drucker oder Plotter kompatibel ist; fragen Sie notfalls Ihren Händler. Gibt es keinen kompatiblen Drucker oder Plotter, was durchaus möglich ist, wählen Sie in den Menüs *Alle anderen* und probieren's mal damit.

Trotzdem werden Sie später unter Umständen feststellen, daß das nicht hinhaut, weil Ihr Drucker zum Beispiel statt Umlauten komische Zeichen schreibt; besonders bei Typenrad-Druckern mag das vorkommen. Da hat sich nämlich bei Framework ein Fehler eingeschlichen: *Alle anderen* initialisiert bei Typenrad-Druckern den Epson MX80 - der eigentlich ein Matrix-Drucker ist ...

Überlassen Sie das Problem, wenn Sie selber nicht zurechtkommen, Ihrem Händler (solcher Service muß im Kaufpreis mit eingeschlossen sein). Vielleicht reicht es, einige DIP-Schalter zu verstellen, vielleicht brauchen Sie ein kleines Basic-Programm, das den Drucker entsprechend initialisiert - der Möglichkeiten sind viele, und zwar von Drucker zu Drucker verschieden.

Tip: Funktioniert die Druckeranpassung überhaupt nicht, erstellen Sie mit Framework eine Textdatei für DOS (im Menü Laufwerk) und drucken diese dann mit dem PRINT-Befehl von MS-DOS.

Bestimmen Sie zwei Drucker, geht Framework davon aus, daß jeder an einer eigenen (parallelen) Schnittstelle angeschlossen ist: LPT1 und LPT2. Ist das nicht der Fall, hängt zum Beispiel der eine Drucker an einer parallelen, der andere an einer seriellen Schnittstelle, oder benutzen Sie dieselbe Schnittstelle für zwei Drucker, müssen Sie dies Framework in der Datei CONFIG.FW mitteilen. (Mehr dazu in Kapitel 8.)

5 Wie man Framework startet

Framework-Aufruf bei Disketten-Systemen

Bei Disketten-Systemen wird zunächst die Systemdiskette 1 benötigt. Sie kommt in Laufwerk A, sodann geben Sie ein:

fw

Es dauert ein Weilchen, bis Framework geladen ist und sich mit der Lizenzvereinbarung meldet, die Sie mit **RETURN** bestätigen müssen - nachdem Sie zuvor in Laufwerk A die Systemdiskette 1 gegen die Systemdiskette 2 getauscht haben.

Sollten Sie das vergessen haben, ist es nicht weiter schlimm. Framework moniert:

Kann Overlay-Datei nicht aktivieren. Ist Systemdiskette 2 im Laufwerk A:?

Nach dem Diskettentausch fahren Sie mit **ESC** fort, als sei nichts gewesen.

Der Aufruf mit FW erübrigt sich, wenn Sie eine AUTOEXEC-Datei erstellt haben, wie sie im Abschnitt »Installation auf Disketten-Systemen« beschrieben wurde. In ihr ist der Framework-Aufruf enthalten, mit dem Starten des Computers wird gleichzeitig auch Framework gestartet.

Framework-Aufruf bei Festplatten

Auch hier lautet der allgemeine Framework-Aufruf

fw

Zuvor müssen Sie indessen in jenes Verzeichnis überwechseln, das die Framework-Dateien enthält. Oder Sie definieren einen Pfad (vorausgesetzt, das Verzeichnis hieße FRAME):

path C:\frame

Damit weiß das System, wo die Programmdateien zu finden sind und folgt diesem Pfad. Framework kann somit aus jedem beliebigen Verzeichnis heraus aufgerufen werden.

Merke: Pfad-Definitionen, die nicht in AUTOEXEC.BAT festgeschrieben sind, werden wieder gelöscht, wenn das System abgeschaltet wird.

Einen solchen Pfad legt man am besten in der Datei AUTOEXEC.BAT ab, die beim Computerstart automatisch abgearbeitet wird. AUTOEXEC.BAT wird sich mit an Sicherheit grenzender Wahrscheinlichkeit bereits auf Ihrer Festplatte im Haupt-

verzeichnis befinden, so daß sie nurmehr um die Pfadangabe erweitert werden muß. Dies erfolgt z.B. mit dem MS-DOS-Texteditor EDLIN - oder mit Framework selbst.

AUTOEXEC.BAT wird wie jede andere Datei auf die Arbeitsfläche geholt und geöffnet, muß dann allerdings mit *Textdatei für DOS erstellen* aus dem Menü *Laufwerk* konvertiert werden (um die Framework-Steuerzeichen zu entfernen, die MS-DOS nicht versteht) und mit dem DOS-Befehl RENAME umbenannt werden (weil Framework die Endung FW anhängt, sie muß jedoch BAT lauten).

Haben Sie bereits Pfad-Definitionen, wird der Framework-Pfad einfach angehängt; die einzelnen Pfade müssen dabei durch ein Semikolon getrennt werden. Etwa so:

```
path c:\;c:\dos;c:\msword;c:\frame
```

AUTOEXEC.BAT können Sie auch benutzen, um Framework automatisch zu starten, indem Sie den Aufruf integrieren. Dann sähe das vielleicht so aus:

```
path c:\;c:\dos;c:\msword;c:\frame
fw
```

Wenn Sie beides nicht wollen, den Pfad zu Framework und den automatischen Aufruf von Framework beim Computer-Start, können Sie statt dessen auch zu einer Batch-Datei greifen, auf deutsch Stapeldatei genannt, weil sie einen Stapel von Befehlen enthält und diese gehorsam der Reihe nach ausgeführt werden, als wären sie von Ihnen einzeln eingegeben worden.

Auch eine solche Stapeldatei kann mit Framework erstellt werden, kann einen beliebigen Namen bekommen, muß als Datei-Extension jedoch BAT haben. Sie sollte sich in einem Verzeichnis befinden, zu dem Sie in AUTOEXEC.BAT einen Pfad weisen (zum Beispiel im Hauptverzeichnis mit path c:\), sonst nützt die ganze Sache nämlich gar nichts, und könnte folgendermaßen aussehen:

Zur Erinnerung: Eine mit Framework erstellte BAT-Datei muß konvertiert und umbenannt werden, ehe sie benutzt werden kann.

```
fw.bat
```

```
cd\frame
fw
```

Sie geben nur ein

```
fw
```

worauf automatisch in das entsprechende Verzeichnis (in unserem Beispiel FRAME) gewechselt und anschließend Framework aufgerufen wird.

Sie ersparen sich mit einer solchen Batch-Datei etliche Tipparbeit und landen stets in dem Verzeichnis, das Sie auch wollen (bei einer bloßen Pfadangabe bleiben Sie ja in dem Verzeichnis, von dem aus Sie Framework aufrufen). Allerdings können Sie in dieser Form eine Framework-Startroutine nicht nutzen, die den hübschen Namen

»Befehlszeilenparameter« trägt und im folgenden Abschnitt behandelt wird - eingeschlossen ein Vorschlag, diese Sache ebenfalls in eine Batch-Datei einzubinden.

Dateien automatisch laden

Was sich »Befehlszeilenparameter« nennt, ist nichts anderes als die bequeme Möglichkeit, mit dem Start von Framework gleich bestimmte Dateien auf die Arbeitsfläche zu holen.

Die Namen dieser Dateien werden dem Framework-Aufruf hintangestellt und durch jeweils ein Leerzeichen getrennt. So:

```
fw test.fw umsatz.fw torte.fw
```

Merke: In der Befehlszeile beim Framework-Aufruf müssen Dateien, die automatisch geladen werden sollen, stets mitsamt ihrer Extension angegeben werden. (Bei Framework-Dateien lautet sie FW.)

Zusätzlich kann eine dieser Dateien auch gleichzeitig ausgeführt, in der Framework-Sprachregelung: neu berechnet werden - ganz praktisch, um ein selbstgestricktes Programm zu starten oder um die Makro-Bibliothek zu aktivieren.

Solchermaßen aktiviert werden kann jedoch nur die letzte Datei in der Reihe, und sie muß mit dem Zusatz /x versehen werden. Also:

```
fw test.fw umsatz.fw torte.fw/x
```

Wenn Framework gleichzeitig mit dem Computer gestartet wird, der Framework-Aufruf also in AUTOEXEC.BAT eingebunden ist, muß AUTOEXEC.BAT entsprechend erweitert werden. In der Praxis bedeutet das tägliche Arbeit - es sei denn, Sie laden jeden Tag dieselben Dateien.

Festplatten-Besitzer greifen deshalb zu einer - gegenüber dem vorigen Abschnitt modifizierten - Framework-Start-Batch-Datei (bei Disketten-Systemen lohnt sich das nicht), die dann folgendermaßen aussehen könnte:

```
fw.bat
```

```
cd\frame  
fw %1
```

Nur einige kurze Erläuterungen dazu für Batch-Neulinge. Zunächst wird wieder in das Verzeichnis gewechselt, in welchem sich die Framework-Dateien befinden. Dem Framework-Aufruf fw folgt ein Platzhalter - an seine Stelle tritt bei der Ausführung der Batch-Datei der Dateiname, welcher bei ihrem Aufruf angegeben wird.

Gestartet wird diese Batch-Datei dann, als würden Sie Framework ganz normal aufrufen:

```
fw torte.fw/x
```

Aber nur eine Datei können Sie auf diese Weise laden. (MS-DOS erlaubt zwar acht Platzhalter, Framework jedoch nicht.) In Kapitel 35 wird eine Startroutine vorgestellt, die dieses und auch das AUTOEXEC-Problem elegant löst.

6 Wie Framework den Speicher verwaltet

Framework hält alle Daten-Dateien im Arbeitsspeicher des Computers - andere Programme haben nur einen Teil dafür reserviert und schaufeln deshalb die Daten permanent zwischen Speicher und Diskette oder Platte hin und her.

Dieser Arbeitsweise wegen ist Framework recht schnell - stößt dafür aber auch irgendwann an die Grenzen der Aufnahmefähigkeit. Haben Sie viele Dateien auf der Arbeitsfläche oder eine riesengroße Tabelle, wird Framework unter Umständen melden:

Zu wenig Speicher -- Zwischenspeicher zum Rückgängigmachen leeren (j/n)?

Damit wird ein bißchen Platz geschaffen, Sie werden freilich der Möglichkeit beraubt, die Rücknahme-Funktion in Anspruch zu nehmen.

Irgendwann schließlich lesen Sie folgende Mitteilung:

Zu wenig Speicher für Operation: Speicher freigeben durch Löschen/Abspeichern

Um weiterarbeiten zu können, müssen Sie eben dies tun: einen Teil der Dateien speichern. Oder die eine riesengroße Datei aufspalten in zwei nicht ganz so große - was freilich, wie sich denken läßt, erhebliche Schwierigkeiten bereitet, da der Arbeitsspeicher ja zum Platzen voll ist.

Deshalb ist es ratsam, ein wachsames Auge auf die anwachsende Dateigröße zu haben - sie ist im Inhaltsverzeichnis ja angegeben. Zudem läßt sich auf verschiedene Weisen abfragen, wieviel Platz im Arbeitsspeicher noch vorhanden ist:

- mit dem CHKDSK-Befehl, wozu man von Framework aus den Menü-Punkt *DOS-Kommandos* ansprechen muß

|| **Merke:** CHKDSK ist ein externer Betriebssystembefehl, für den man die DOS-Diskette braucht.

- mit dem Framework-Dienstprogramm DISKUTIL
- mit einem Makro
- mit einer Funktion aus der Framework-Programm-Bibliothek LIB.

|| **Hinweis:** Alle Möglichkeiten werden an den entsprechenden Stellen behandelt (in Kapitel 27, 33, 40 bzw. 41).

Wieviel Arbeitsspeicher läßt Framework grundsätzlich frei für seine Dateien? Bei einem Computersystem mit 512 Kbyte RAM sind es ungefähr 250 Kbyte - ungefähr, weil CHKDSK und die Makro-Abfrage unterschiedliche Werte angeben.

Warum nur so wenig? Zunächst nistet sich das Betriebssystem im Arbeitsspeicher ein, und dann beanspruchen die Framework-Programmdateien auch noch Platz. Nur ein Teil von ihnen befindet sich im Arbeitsspeicher, ansonsten wird geladen, was gerade benötigt wird. Wieviel Platz sie haben dürfen, können Sie selber bestimmen in der Datei CONFIG.FW (siehe Kapitel 8).

Sie können zum Beispiel veranlassen, daß Framework mehr von seinen Programmdateien in den Arbeitsspeicher lädt, als normalerweise vorgesehen ist - Framework arbeitet dadurch schneller, weil weniger Plattenzugriffe nötig sind, der Platz für Dateien ist dagegen knapp. Sie können den Programm-Platz auch reduzieren - das hat die umgekehrten Folgen: mehr Raum für Dateien, hingegen zahlreiche Plattenzugriffe, die Zeit kosten. Sie müssen experimentieren, was Ihren Gegebenheiten mehr entgegenkommt.

Aus all dem ergibt sich, daß bei einem Computer-System mit normaler Ausstattung eine RAM-Disk nicht unbedingt Vorteile bringt, weil sie zu Lasten jenes Platzes geht, der für die Daten-Dateien zur Verfügung steht.

7 Wie Framework die Laufwerke anspricht

Seine Programm-Dateien holt sich Framework stets von Laufwerk A bzw. von der Festplatte. Wo Ihre Daten gespeichert werden, können Sie hingegen selber festlegen.

Die Laufwerke, das wissen Sie, werden mit den Buchstaben des Alphabetes bezeichnet: Das linke Laufwerk ist A, das rechte B - soweit alles klar. Darüber hinaus wird es etwas komplizierter und hängt von der System-Konfiguration ab. Einige Beispiele:

- ein Disketten-Laufwerk, eine Festplatte: Die Festplatte ist C, Laufwerk B wird in diesem Fall übergangen
- zwei Disketten-Laufwerke, eine RAM-Disk: Hier nun ist die RAM-Disk Laufwerk C
- ein Disketten-Laufwerk, eine Festplatte, eine RAM-Disk: Die Festplatte ist C, die RAM-Disk ist D.

Und um die Sache vollends zu verwirren: Ein Computer-System kann mehr Laufwerke haben, als tatsächlich vorhanden sind; man spricht in diesem Fall von »logischen« Laufwerken im Gegensatz zu den »physikalischen«.

Wozu die »logischen«, die nur in der Theorie vorhandenen Laufwerke? Bei bestimmten Betriebssystemversionen, nämlich bei DOS Plus und bei MS-DOS von 3.0 an (also MS-DOS 3.0, MS-DOS 3.1, MS-DOS 3.2) kann einer Sub-Directory eine solche Laufwerksbezeichnung zugewiesen werden, was viel Tipparbeit erspart, etwa beim Kopieren von Dateien: Man muß nicht mehr den ganzen langen Verzeichnis-Pfad angeben, sondern lediglich das definierte logische Laufwerk.*

Dieser Exkurs hat für Framework handfeste Bedeutung. Bei den genannten Betriebssystemversionen sind von vornherein fünf Laufwerke festgeschrieben - A bis E; gleichgültig, welche davon auch physikalisch vorhanden sind.

Tip: Die Anzahl der logischen Laufwerke läßt sich auf Betriebssystemebene mit der Anweisung LASTDRIVE in CONFIG.SYS ändern.

Als Voreinstellung hat Framework aber nun festgelegt, daß Dateien auf dem letzten angegebenen Laufwerk gespeichert werden - zum Beispiel also auf Laufwerk E. Da es ein solches im Regelfalle nicht gibt, werden all Ihre mühsam erstellten Dateien ins Nichts gespeichert. Oder deutlicher gesagt: Sie gehen verloren. Kann auch sein, daß Framework sich nicht an seine eigenen Regeln hält und beim Speichern stets Laufwerk A anspricht, was für Datensicherung ja gut, ansonsten jedoch fragwürdig ist: In Lauf-

* Unter MS-DOS dient dazu der Befehl SUBST, unter DOS Plus wird CHDIR verwendet. Nähere Erläuterungen dazu finden sich in den Handbüchern zu den Betriebssystemen oder auch in: Hans C. Nieder, MS-DOS Version 3; Hans H. Gerhardt: Schneider PC - MS-DOS 3.2; Rudi Kost: Der Schneider PC (alle Verlag Markt & Technik, Haar)

werk A befindet sich notwendigerweise die Systemdiskette 2, auf der nicht mehr allzuviel Platz für Daten-Dateien ist.

Jedenfalls sollte man etwas dafür tun, daß kein Malheur passiert.

Welche Laufwerke Ihr Computer erkennt, sehen Sie, sobald Sie Framework gestartet haben: Brav werden auf der Arbeitsfläche sämtliche Laufwerke aufgeführt, die physikalischen und die logischen. Welches Laufwerk davon nun das Primär-Laufwerk sein soll, also jenes, auf dem gespeichert wird, legen Sie in der Datei CONFIG.FW fest - eine Framework-Datei, nicht zu verwechseln mit der MS-DOS-Datei CONFIG.SYS.

Sie befindet sich auf der Systemdiskette 2, und weil mit ihr noch allerhand mehr bestimmt wird, ist ihr ein eigenes, das folgende Kapitel gewidmet.

Unabhängig vom dem, was in CONFIG.FW festgeschrieben ist, läßt sich während einer Arbeitssitzung das aktuelle Laufwerk wechseln. Dazu muß man mit dem DOS-Zugriff in das Betriebssystem gehen und dort zum Beispiel A statt B oder C als Primär-Laufwerk bestimmen.

■ | **Hinweis:** Alles über den DOS-Zugriff in Kapitel 40.

Die Dateien werden dann auf Laufwerk A gespeichert, und diese Einstellung bleibt solange gültig, bis erneut gewechselt oder Framework neu gestartet wird - eine temporäre Umleitung also.

8 Wie man Framework anpaßt

Was in der Framework-Datei CONFIG.FW steht, wird beim Start von Framework ausgeführt und ist insofern der DOS-Datei CONFIG.SYS ähnlich. CONFIG.FW enthält verschiedene Parameter, mit denen Sie Framework Ihren eigenen Bedürfnissen anpassen können - maßschneidern sozusagen.

CONFIG.FW kann wie eine normale Framework-Datei geladen, editiert und gedruckt werden. Sie ist zudem freundlicherweise mit Erläuterungen versehen, so daß Änderungen wenig Probleme bereiten dürften. Wenn Sie jetzt noch nicht alles verstehen, dann später, mit wachsender Framework-Erfahrung.

Tip: Arbeiten Sie immer nur mit Kopien der Originaldisketten. Unabsichtliche Änderungen in Programmen (wie CONFIG.FW) geschehen leicht und können verheerende Folgen haben, und oft ist man dann nicht in der Lage, den Urzustand aus eigenem Wissen wiederherzustellen.

Ändern dürfen Sie nur die Werte, nicht hingegen die Kommandos (die in CONFIG.FW fett gedruckt sind). Alle Änderungen indes werden erst beim nächsten Start von Framework wirksam.

Im folgenden ist CONFIG.FW abgedruckt - ohne den Einleitungstext und ohne die Semikolons am Zeilenanfang, die Framework signalisieren, daß es sich um kommentierenden Text, nicht um eine Programmanweisung handelt.

Teile des Framework-Programmes sind auf der Systemdiskette 2 gespeichert und werden automatisch bei Bedarf in den Arbeitsspeicher geladen. Um die Zeit für diese Diskettenzugriffe zu sparen, können Sie mit dem Wert #TRUE zu dem folgenden Parameter veranlassen, daß größere Teile des Programmes ständig im Arbeitsspeicher bleiben. Voraussetzung ist jedoch, daß Ihr Rechner mehr als 320 Kbyte Arbeitsspeicher hat. Beachten Sie, daß mit #TRUE weniger Speicher für Ihre Frames zur Verfügung steht.

Standardwert = #FALSE

KEEP_OVERLAY #FALSE

Der nächste Parameter bestimmt die Größe des Pufferbereiches im Arbeitsspeicher, der für die Überlagerung von Programmteilen verwendet wird (Overlay). Falls der Parameter KEEP_OVERLAY den Wert #TRUE hat, ist die Angabe zu OVERLAY_SIZE ohne Wirkung. Der Zahlenwert bemisst sich in Kbyte, d.h. 1 steht für 1024 Byte.

Standardwert = 16. Minimum = 16. Maximum = 500.

OVERLAY_SIZE 16

Der Wert 0 zum folgenden Parameter legt fest, daß das in der alphabetischen Reihenfolge der Laufwerksbezeichnungen letzte Laufwerk als Primär-Laufwerk genommen wird: etwa B: bei einem System mit nur zwei Floppy-Laufwerken oder C: bei einem System mit zusätzlicher Hard-Disk. Wird die Zahl geändert, ist das Laufwerk Primär-Laufwerk, das bei fortlaufender Numerierung der Laufwerke durch die Zahl ausgewählt wird.

Standardwert = 0. Minimum = 0. Maximum = 26.

1 = Erstes Laufwerk

2 = Zweites Laufwerk

3 = Drittes Laufwerk . . . usw.

DEFAULT_DRIVE 0

Die nächsten beiden Parameter bestimmen die Größe des Frames zur Anzeige eines Dateiverzeichnisses, das mit dem Marker aus dem Stapel für die Laufwerke ausgewählt und mit RETURN auf der Arbeitsfläche geöffnet wird.

	Standardwert	Minimum	Maximum
Breite (WIDTH) in Zeichen:	27	15	80
Höhe (HEIGHT) in Zeilen:	9	1	30

DISK_FRAME_WIDTH 27

DISK_FRAME_HEIGHT 15

Die folgenden vier Parameter legen die Anfangsgröße eines neu auf der Arbeitsfläche zu erstellenden leeren bzw. Text-Frames (DOCUMENT, über das Menü NEU mit dem Befehl Frame (leer/Text)) bzw. eines neuen Frames für eine Grafik (GRAPH, über das Menü Grafik mit dem Befehl Neue Grafik zeichnen) fest, sofern dieser Frame nicht innerhalb eines anderen Frames liegt. Breite (WIDTH) bzw. Höhe (HEIGHT) bemessen sich in Anzahl der Zeichen bzw. Zeilen auf dem Bildschirm.

Frame-Typ	Dimension	Standardwert	Minimum	Maximum
leer/Text (DOCUMENT)	Breite	70	1	80
	Höhe	11	1	30
Grafik (GRAPH)	Breite	55	1	80
	Höhe	14	1	30

DOCUMENT_CREATE_WIDTH 70

DOCUMENT_CREATE_HEIGHT 11

GRAPH_CREATE_WIDTH 55

GRAPH_CREATE_HEIGHT 14

Der nächste Parameter bestimmt den Anfangswert für die Breite der Spalten bzw. Felder in einem neu erstellten Frame zur Tabellenkalkulation bzw. für eine Datenbank.

Standardwert = 9. Minimum = 1. Maximum = 256.

SS_COLUMN_WIDTH 9

Die folgenden beiden Parameter dienen zur Einstellung der Anfangswerte für neu erstellte Frames zur Tabellenkalkulation bzw. für Datenbanken hinsichtlich der Anzahl der Spalten bzw. Felder (SS_WIDTH), und der Anzahl der Zeilen bzw. Sätze (SS_HEIGHT). Sie erscheinen als Voreinstellungen im Menü NEU zu den Positionen Breite (#Spalten/Felder) und Länge (#Zeilen/Sätze).

Standardwert = 14. Minimum = 1. Maximum = 32000.

SS_WIDTH 14

SS_HEIGHT 14

Die nächsten vier Parameter geben die Voreinstellung zu den letzten Positionen im Menü Text.

	Standardwert	Minimum	Maximum
Linker Rand (LEFTMARGIN)	0	0	254
Rechter Rand (RIGHTMARGIN)	65	0	255
Einzug Absatz (PARAGRAPH_INDENT)	0	-30	30
Tabulatorsprung (TABSIZe)	8	0	255

WORD_LEFTMARGIN 0

WORD_RIGHTMARGIN 65

WORD_PARAGRAPH_INDENT 0

WORD_TABSIZE 8

Im Sub-Menü Ziel für Ausgabe des Menüs Drucken können Sie als Ausgabe-Ziel Erster Drucker (PRINTER1), Zweiter Drucker (PRINTER2) oder Plotter (PLOTTER) wählen. Die folgenden drei Parameter bestimmen zu diesen Geräten den Ausgabekanal in der für das Betriebssystem üblichen Bezeichnung, die in Anführungszeichen zu setzen ist.

Standardwerte:

Erster Drucker	= "LPT1"
Zweiter Drucker	= "LPT2"
Plotter	= "COM1"

Mögliche Angaben:

"AUX"	"PRN"	"NUL"	
"COM1"	"COM2"	"COM3"	"COM4"
"LPT1"	"LPT2"	"LPT3"	"LPT4"

Hinweis: Um einen Drucker oder Plotter an einem COM-Kanal zu betreiben, muß dieser Kanal mit dem MODE-Befehl des Betriebssystems initialisiert werden, bevor Framework gestartet wird. Einzelheiten hierzu finden Sie im Handbuch zum Betriebssystem.

PRINTER1"LPT1"

PRINTER2"LPT2"

PLOTTER"COM1"

Mit den folgenden zwei Parametern wird festgelegt, welche Datenmenge an einen Drucker (PRINTER1, PRINTER2) von Framework schrittweise gesendet wird, während Sie parallel dazu mit Framework weiterarbeiten (Spooling).

Falls Framework während des Druckens auf Ihre Eingaben zu langsam reagiert, sollten Sie eine kleinere Zahl nach dem Parameter des entsprechenden Druckers angeben. Arbeitet der Drucker hingegen langsamer als gewohnt, ist die Zahl zu erhöhen. Während der Ausgabe von Grafiken sind Verzögerungen bei der Reaktion auf Tastatureingaben bzw. beim Drucken/Plotten der Grafik unvermeidlich.

Standardwert = 5. Minimum = 0. Maximum = 25.

PRINTER1_SPEED 5

PRINTER2_SPEED 5

Beim Ausdrucken eines Textes, der über mehrere Seiten geht, kann es beim Wechsel auf eine neue Seite zum "Durchschneiden" eines Absatzes kommen. Der Absatz beginnt am Ende einer Seite und endet auf der nächsten Seite. Die Anzahl der Zeilen, die bei einer solchen Aufteilung eines Absatzes mindestens auf jeder der beiden aufeinanderfolgenden Seiten gedruckt werden sollen, legt der Parameter WIDOW_ORPHAN_TOLERANCE fest. Wird der dort angegebene Minimalwert unterschritten, erscheint der Absatz komplett auf der zweiten Seite.

Standardwert = 3. Minimum = 1. Maximum = 10.

WIDOW_ORPHAN_TOLERANCE 3

Die folgende Gruppe von Parametern bestimmt die Voreinstellung zu einer Reihe von Druckfunktionen. Solche Druckfunktionen werden als Formeln auf dem Rand des Frames eingeben (siehe auch Hilfe-System unter dem Stichwort "Drucken").*

Funktion	Standardwert	Minimum	Maximum
@tm (TOP_MARGIN, Unterer Rand)	6	0	255
@bm (BOTTOM_MARGIN, Oberer Rand)	6	0	255
@po (PAGE_OFFSET, Zeilenanfang)	10	0	255
@ll (LINE_LENGTH, Zeilenlänge)	65	1	255
@pl (PAGE_LENGTH, Seitenlänge)	72	1	255
@sp (SPACING, Zeilenabstand)	1	1	255

TOP_MARGIN 8

BOTTOM_MARGIN 6

PAGE_OFFSET 10

LINE_LENGTH 65

PAGE_LENGTH 72

SPACING 1

Mit dem folgenden Parameter wird die Zahl der Durchläufe bei der Ausführung einer @while-Schleife (siehe Hilfe-System zum Stichwort "Die Programmiersprache FRED") be-

* Das ist der Originaltext von CONFIG.FW. Hier sind allerdings zwei deutsche Bezeichnungen verwechselt worden: @tm meint den oberen Rand, @bm den unteren.

grenzt. Führt die bei @while angegebene Bedingung nicht vorher zum Abbruch, stoppt die Ausführung bei Erreichen der mit dem Parameter vorgegebenen Anzahl. Hiermit werden "unendliche" Schleifen vermieden.

Standardwert = 2048. Minimum = 1. Maximum = 32000.

MAX_WHILE 2048

Die nächsten drei Parameter bestimmen die bei der Anzeige von Zahlen verwendeten Symbole, wenn diese mit Befehl Währungsbetrag aus dem Menü Zahlen formatiert werden (Zelle, Datenbankfeld oder ganzer Frame).

Mit CURRENCY_SYMBOL wird eines der in Framework bereits vorhandenen einstelligen Währungssymbole als Voreinstellung ausgewählt. Die Angabe eines durch Anführungszeichen zu begrenzenden Textes bei CURRENCY_UNITS ersetzt das mit CURRENCY_SYMBOL ausgewählte Zeichen, d.h. dieser Parameterwert ist dann nicht wirksam.

Der Text zu CURRENCY-UNITS kann beliebig gewählt werden und muß nicht Geldeinheiten bezeichnen. Abhängig von der Anwendung könnte z.B. auch "m", "Tonnen" usw. sinnvoll sein. Durch Leerzeichen am Anfang bzw. Ende des Textes läßt sich der Abstand des Textes von der Zahl einstellen. Auch ein leerer Text, d.h. "", ist zugelassen und unterdrückt ebenfalls das einstellige Währungssymbol von CURRENCY_SYMBOL. Erfolgt keine Angabe zu CURRENCY_UNITS, d.h. der Parameterwert wurde einschließlich der Anführungszeichen gelöscht, gilt die Einstellung von CURRENCY_SYMBOL.

CURRENCY_POSITION dient zur Angabe, ob das Währungssymbol von CURRENCY-SYMBOL bzw. der von Ihnen selbst definierte Text bei CURRENCY-UNITS im Währungsformat vor (#BEFORE) oder nach (#AFTER) der Zahl erscheinen soll.

Mögliche Werte zu

CURRENCY_SYMBOL: @DOLLAR (Standardwert), @YEN oder @POUND (nur wirksam, wenn nach CURRENCY_UNIT alle Angaben gelöscht werden, d.h. dort kein Wert definiert ist!)

CURRENCY_UNITS: frei wählbarer Text in Anführungszeichen, z.B. "DM" (Standardwert), "Mio DM" usw.

Ebenso können wahlweise die Formate für die Zahlendarstellung zusätzlich angegeben werden. Mit Hilfe dieser Formate können Sie zwischen der Zahlenausgabe mit Dezimalkomma und Tausender-Punkte (@milli), oder dem Dezimalpunkt und Tausender-Komma-Darstellung (@thousands) definieren.

CURRENCY_POSITION: #AFTER (danach, Standardwert) oder #BEFORE (davor)

CURRENCY_SYMBOL @DOLLAR
CURRENCY_UNITS " DM"
CURRENCY_UNITS @milli
CURRENCY_POSITION #AFTER

Der folgende Parameter begrenzt die Anzahl der Zeichen, die in einem für DOS-Kommandos eingerichteten Frame gespeichert werden. Wird bei der Eingabe und Ausführen von DOS-Kommandos diese Grenze überschritten, werden die zu Beginn des Frames im Arbeitsspeicher vorhandenen Zeichen gelöscht, um Platz für weitere Zeichen am Ende des Frames zu schaffen (First in, First out)

Standardwert = 2000. Minimum = 500. Maximum = 32000.

DOS_ACCESS 2000

Der nächste Parameter setzt einen Text in der vorletzten Position des Menüs Laufwerk ein. Der frei wählbare Text ist in Anführungszeichen zu schreiben.

Standardwert = "Sonderdienste"

UTIL_NAME "Sonderdienste"

Die Zeit zum Öffnen und Schließen der Frames für Dateiverzeichnisse bzw. Dokumente auf der Arbeitsfläche läßt sich mit dem folgenden Parameter beeinflussen. Eine kleinere Zahl verkürzt diesen Vorgang. Der Wert "10" erzielt interessante Effekte.

Standardwert = 1. Minimum = 0. Maximum = 10.

ZOOMDELAY 1

Falls der nächste Parameter den Wert #TRUE hat, wird der Co-Prozessor 8087 von Framework unterstützt, sofern Ihr Computer damit ausgerüstet ist. Dieser Chip beschleunigt Berechnungen erheblich. Mit der Angabe #FALSE wird der 8087 von Framework nicht verwendet.

Standardwert = #TRUE.

USE_87 #TRUE

Der folgende Parameter regelt den Zugriff durch den Anwender auf Formeln in Framework. Wird zu SERIALNUMBER eine andere Zahl angegeben, kann die Formel nur gelesen oder verändert werden, wenn die Seriennummer des jeweiligen Framework-Programmpaketes mit der Zahl identisch ist. Auf die Ausführung der Formel sind jedoch die Angaben eines von 0 abweichenden Wertes ohne Einfluß, d.h. unter einer bestimmten Seriennummer geschriebene Formeln lassen sich auch mit jeder anderen Serien-Nummer ausführen.

Standardwert = 0. Minimum = 0. Maximum = 32000.

SERIALNUMBER 0

Framework - eine Einführung

9 Der Bildschirm

Der Framework-Bildschirm ist in fünf verschiedene Bereiche unterteilt, deren jeder einen spezifischen Namen trägt.

Ganz oben befindet sich die *Menüzeile* mit einer mitlaufenden Uhr und neun Menüs, über die man eine Vielzahl von Befehlen und Funktionen aufruft.

Der gerasterte Bereich darunter ist die *Arbeitsfläche* - hier spielt sich die eigentliche Arbeit ab, hier tummeln sich die Frames.

Die Arbeitsfläche enthält zwei *Stapel*. Der obere wird von Framework *Dateiverzeichnis* genannt - eine etwas verkrampfte Bezeichnung, denn tatsächlich werden hier die vorhandenen Laufwerke angezeigt, und zwar sowohl die physikalischen wie auch die logischen (siehe Kapitel 7).

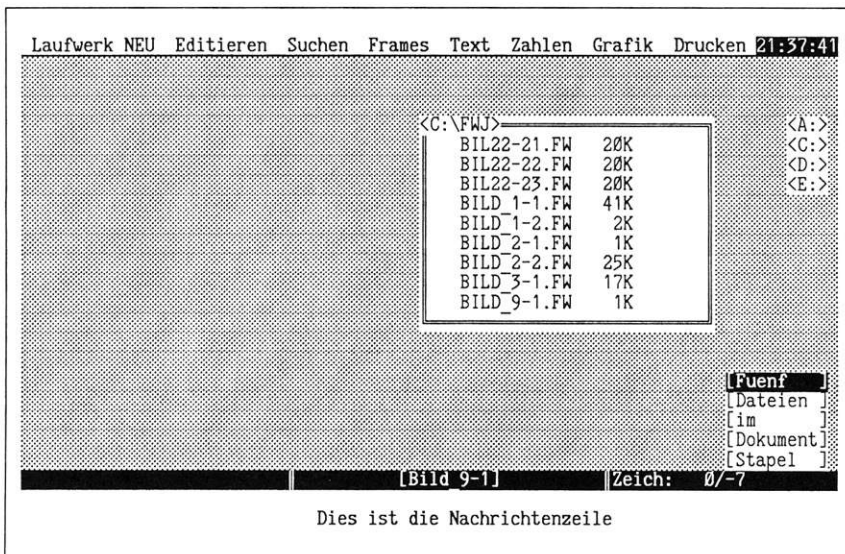


Bild 9.1: Der Framework-Bildschirm

Rechts unten auf der Arbeitsfläche ist der *Dokumentenstapel*. Er besteht aus den Dateien, die in den Arbeitsspeicher des Computers geladen worden sind und darauf warten, daß sie - auf der Arbeitsfläche - geöffnet werden, daß also mit ihnen gearbeitet wird.

Der helle Streifen unterhalb der Arbeitsfläche ist die *Statuszeile*, die ihrerseits dreifach unterteilt ist. Im linken Teil erscheinen die ersten Zeichen eines Frames oder eines Feldes in einer Tabelle oder Datenbank - Text, Zahlen oder Formeln.

Der mittlere Bereich nennt den Namen des gerade aktuellen Frames oder des Bereichs innerhalb eines Frames - etwa einer Datenzeile in einer Tabelle.

|| **Merke:** Der »aktuelle« Frame ist stets derjenige, auf welchem der Cursor steht - gleichgültig, ob er geöffnet ist oder sich noch im Dokumentenstapel befindet.

Der rechte Abschnitt der Statuszeile gibt zusätzliche Informationen, die sich je nach Art der Anwendung wandeln. Zum Beispiel:

- Befindet sich der Cursor im Dateiverzeichnis, bedeutet
Laufw: 3/5
daß der Cursor auf dem dritten von fünf vorhandenen Laufwerken steht.
- Befindet sich der Cursor im Dokumentenstapel, gilt Analoges:
Dok.: 5/16
- Befindet sich der Cursor in einer Datenbank, wird die Anzahl der Sätze angezeigt:
Sätze: 34/34
bedeutet: die Datenbank besteht aus 34 Sätzen, und derzeit werden alle 34 angezeigt. Hingegen besagt
Sätze: 4/34
daß die Datenbank selektiert worden ist und nur die vier selektierten Datensätze (aus insgesamt 34) auf dem Bildschirm zu sehen sind.

Unterhalb der Statuszeile ist das *Nachrichtenfeld*, wiederum unterteilt in zwei Zeilen.

Die obere ist die *Editierzeile*, die anzeigt, was eingegeben oder geändert wird. Darunter befindet sich die *Nachrichtenzeile* mit Framework-Meldungen über Aktionen oder Fehler, mit Hilfestellungen oder Aufforderungen.

10 Die Menüs

In den Menüs hält Framework einen Großteil seiner Funktionen abrufbereit. Es gibt zwei Möglichkeiten, die Menüs zu öffnen und ebenfalls zwei, die Funktionen zu aktivieren; beide Methoden können auch kombiniert werden.

1. Drücken Sie **[INS]**. Ein Menü öffnet sich.

■ **Merke:** Mit **[INS]** wird stets das Menü geöffnet, mit dem zuletzt gearbeitet worden ist.

2. Bewegen Sie sich mit den Cursortasten **[←]** oder **[→]** von Menü zu Menü.
3. Bewegen Sie sich mit den Cursortasten **[↓]** oder **[↑]** innerhalb eines Menüs von Funktion zu Funktion.
4. Rufen Sie die gewünschte Funktion auf mit **[RETURN]**.

Das war jetzt der umständlichere Weg. Der schnellere:

1. Drücken Sie gleichzeitig **[CTRL]** und den ersten Buchstaben des Menüs - damit wird das Menü geöffnet.
2. Drücken Sie den ersten Buchstaben der Funktion - damit wird die Funktion aktiviert.

Haben Sie ein Menü geöffnet, wollen aber keine Funktion aktivieren, verlassen Sie das Menü wieder mit **[ESC]**.

Die meisten Funktionen setzen sofort eine Aktion in Gang, andere verlangen zusätzlich weitere Angaben:

1. Aktivieren Sie die Funktion *Hole Dokument mit Namen* im Menü *Laufwerk*. Der Cursor springt in die Editierzeile.
2. Geben Sie den Namen der zu holenden Datei ein - zum Beispiel `FRAME_1.FW`, wenn sich die Beispieldiskette im Laufwerk befindet (bzw. deren Dateien auf der Platte).

■ **Zur Erinnerung:** Arbeiten Sie nur mit Kopien, nicht mit den Originaldisketten.

3. Drücken Sie **[RETURN]**.

Etliche Funktionen arbeiten auch als Schalter - zum Beispiel im Menü *Frames*. Ein *JA* vor der Funktion bedeutet, daß sie eingeschaltet, zwei Striche, daß sie ausgeschaltet ist.

Hin und wieder erscheinen die Namen von Funktionen in Kursivschrift. Das besagt, daß mit ihnen derzeit nicht zu arbeiten ist. Das hängt meist von der Cursorstellung ab.

Befindet sich der Cursor beispielsweise in einem Text-Frame, ist *Spalten/Felder entfernen* aus dem Menü *Editieren* nicht aktiv -, denn diese Funktion bezieht sich auf Tabelle oder Datenbank. Befindet sich der Cursor in einem Frame, ist *Schließen alle* im Menü *Frames* nicht zu benutzen - dazu müßte der Cursor auf dem Rand des Frames stehen.

Einige Menüs verfügen auch über Sub-Menüs, die durch ein Dreieck gekennzeichnet sind - das Menü *Drucken* etwa hat noch die Sub-Menüs *Ziel für Ausgabe* und *Optionen für Ausdruck*. Geöffnet werden diese Sub-Menüs, als wolle man eine Funktion aktivieren, also mit dem ersten Buchstaben ihres Namens oder mit **RETURN**, wenn sich der Cursor auf dem Namen befindet. Um sie zu schließen, gibt es gleichfalls zwei Möglichkeiten, mit unterschiedlichen Folgen allerdings:

- **NUM-** führt vom Sub-Menü ins Hauptmenü zurück
- **ESC** schließt Sub-Menü und Hauptmenü gemeinsam

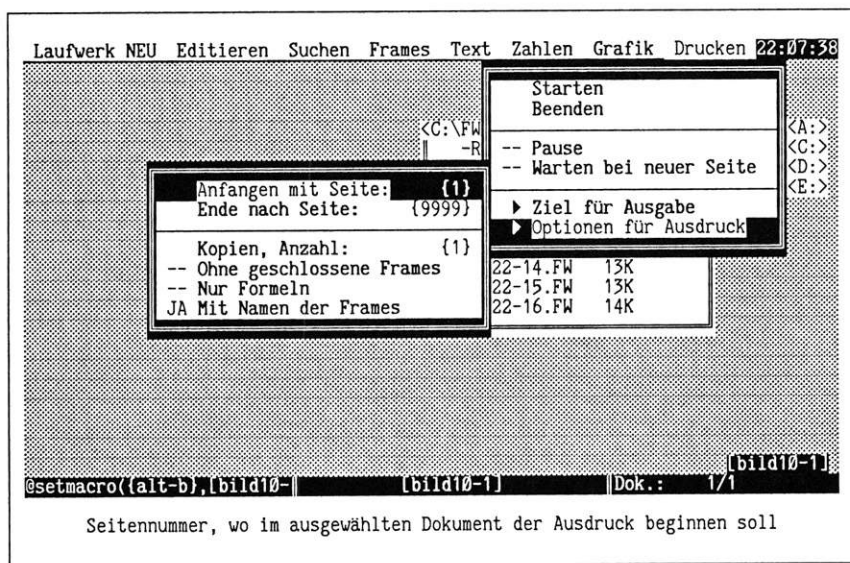


Bild 10.1: Das Menü »Drucken« mit seinem Sub-Menü »Optionen für Ausdruck«

Was die einzelnen Funktionen bewirken, tut Framework freundlicherweise, in gedrängter Form, in der Nachrichtenzeile kund. Für den Framework-Neuling ist das zu wenig - aber er wird's (unter anderem im Verlauf dieses Buches) rasch lernen. Dem etwas erfahreneren Anwender reicht's aus, um an Funktionen erinnert zu werden, die er im Prinzip zwar kennt, jedoch nicht häufig einsetzt.

Am Ende dieses Buches findet sich ein ausklappbarer Menü-Fahrplan, der hilft, sich in Frameworks anfangs doch etwas verwirrendem Menü-System zurechtzufinden.

11 Die Funktionstasten

Andere Framework-Funktionen, hauptsächlich für den Umgang mit Frames, werden mit den Funktionstasten abgerufen. Framework liegt eine Tastaturschablone mit den Tastenbezeichnungen bei, die freilich nur bei Tastaturen nach der alten IBM-Norm verwendet werden kann, bei der die Funktionstasten links in zwei Blöcken angeordnet sind.

Die Tastenbedeutungen erfolgen hier deshalb nur der Vollständigkeit zuliebe:

- | | |
|-----|------------------|
| F1 | Hilfestellung |
| F2 | Formel editieren |
| F3 | Position |
| F4 | Größe |
| F5 | Neuberechnung |
| F6 | Auswahl |
| F7 | Verlagern |
| F8 | Kopieren |
| F9 | Zoom |
| F10 | Zeigen |

Was damit jeweils ausgelöst wird, wie sie einzusetzen sind, erfahren Sie, sofern Sie's nicht schon wissen, in den nächsten Kapiteln.

12 Das Hilfe-System

Aus jeder Situation heraus lassen sich in Framework mit der Funktionstaste **[F1]** Hilfsinformationen abrufen - ein Handbuch in Kurzfassung gewissermaßen.

Die Hilfestellung geht, wo immer das möglich ist, von der aktuellen Situation aus. Haben Sie das Menü *Editieren* geöffnet, erhalten Sie Informationen zum Menü *Editieren*. Im Nachrichtenfeld erscheint ein horizontal angeordnetes Menü, zu bedienen wie die üblichen Framework-Menüs auch, mit dem innerhalb des Hilfe-Systems gewandelt werden kann.

Wichtig ist vor allem der *Stichwort-Index*, der es erlaubt, gezielt Hilfe abzurufen.

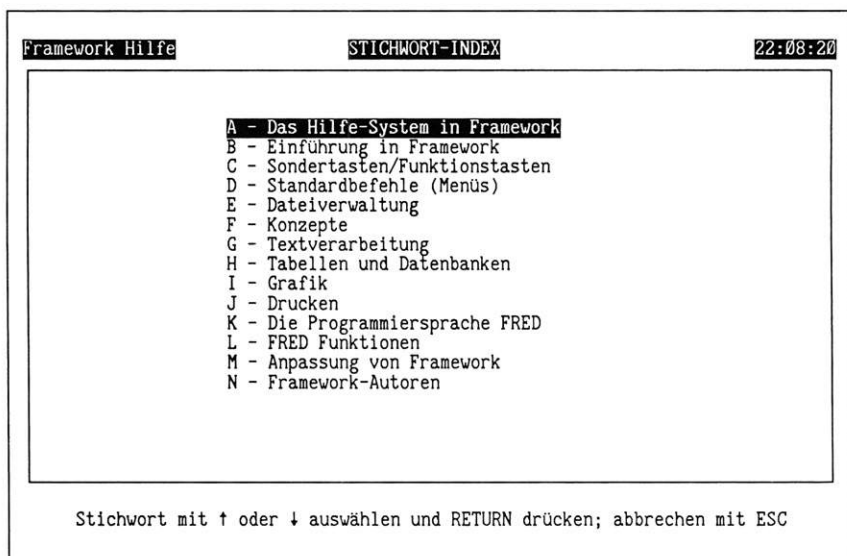


Bild 12.1: Das Hilfe-System von Framework

13 Frames

Über Frames

Framework-Anfänger haben ihre liebe Not, mit den Frames zurechtzukommen. Hilfreich ist gewiß das Framework-Tutorial - nehmen Sie sich die Muse dafür, das ist nützlich verbrachte Zeit. Ist's Ihnen immer noch nicht so ganz klar, wollen Sie noch mehr üben, wollen Sie später einmal nachschlagen? Dann greifen Sie zu diesem Kapitel, das in einem Schnelldurchgang die Grundlage schafft für die Arbeit mit Framework: den Umgang mit Frames. Möchten Sie üben? Dann befolgen Sie bitte die Anweisungen - damit Sie auch das auf dem Bildschirm haben, wovon ich rede.

Vorbereitung: Starten Sie Framework, legen Sie die Beispieldiskette dieses Buches in Laufwerk B oder kopieren Sie deren Dateien auf die Festplatte (am besten in das Framework-Verzeichnis).

Frames, nochmals gesagt, sind Rahmen - oder bezeichnen wir sie besser als Fenster, die einen Rahmen haben, der für den Umgang mit Frames von einiger Bedeutung ist.

Bei Framework spielt sich alles in Frames ab: Sie sind das »Papier«, auf dem geschrieben, gerechnet, gezeichnet wird. Frames zeigen meist nur einen Ausschnitt aus dem »Papier« - einen Ausschnitt, den man verschieben oder vergrößern kann.

Frames bestehen zunächst einmal aus dem Rahmen, der links oben, zwischen eckigen Klammern, den Namen des Frames enthält. Damit's nicht so theoretisch bleibt:

1. Bringen Sie den Cursor im Dateiverzeichnis-Stapel auf das aktuelle Laufwerk - dort ist er schon, wenn Sie Framework gestartet und sonst noch nichts unternommen haben.
2. Drücken Sie `RETURN` (im Laufwerk muß sich eine Diskette befinden!)

Dieser Frame heißt A:\ oder B:\ oder auch C:\fw - je nachdem, welches Laufwerk und welches Unterverzeichnis Sie angesprochen haben.

Merke: Auch das Dateiverzeichnis (das Inhaltsverzeichnis eines Laufwerkes) ist ein Frame, der wie jeder andere manipuliert werden kann. Seine Ausgangsgröße kann in CONFIG.FW festgelegt werden.

Derzeit befindet sich der Cursor *auf* dem Rand, dem Rahmen des Frames. Das ist sehr wichtig. Um Frames zu verschieben oder zu speichern, um ihre Größe oder Position zu verändern, muß der Cursor auf dem Rand des Frames sein.

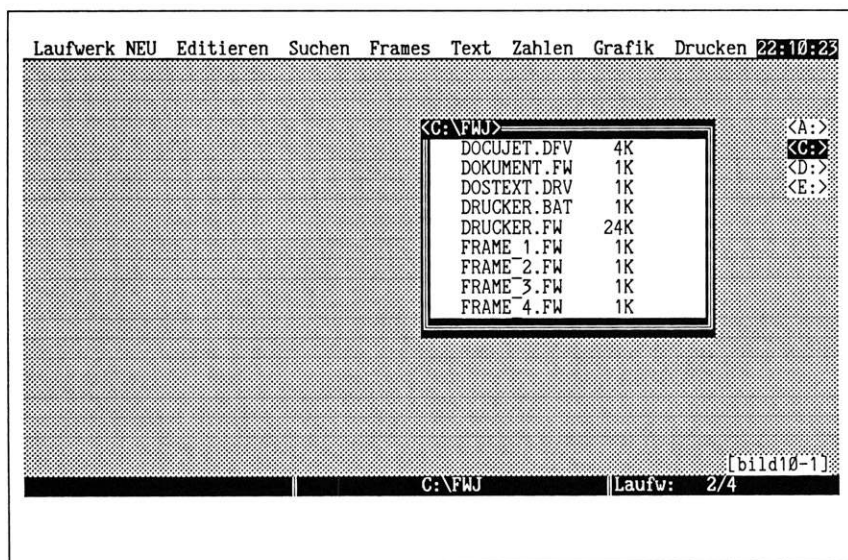


Bild 13.1: Das Dateiverzeichnis: der Inhalt eines Laufwerkes

Genauso kann er sich *im* Frame befinden - damit man mit den Dateien (Text, Tabelle usw.), die der Rahmen des Frames umschließt, arbeiten kann.

Oder man schickt den Cursor *hinter* den Frame - um Formeln einzugeben beispielsweise. Wir sehen einen Frame auf dem Bildschirm nur zweidimensional, in Wahrheit hat er auch noch eine dritte Dimension. Er ist gleichsam ein Quader, der uns seine Schokoladenseite präsentiert: eine seiner Seitenflächen.

Im Quader drinnen, also »hinter« dem Frame, kann's wüst zugehen. Dort spielt sich allerlei ab - das Wichtigste eigentlich und dasjenige, was einen Computer von der Schreibmaschine unterscheidet.

Der Bereich »hinter« dem Frame wird von Framework Formelspeicher genannt. Wie's der Name sagt: Er enthält Formeln. Sollen, simpelstes Beispiel, zwei Zahlen addiert werden, wird die Anweisung dazu im Formelspeicher abgelegt. Das Ergebnis der Addition erscheint im Frame - dieser Bereich heißt im Framework-Jargon deshalb auch Datenspeicher.

Merke: Auch jede Zelle einer Tabelle, jedes Feld einer Datenbank kann »hinter« sich Formeln aufnehmen.

Eine Formel ist für Framework, umfassend gesagt, alles, was Operationen mit Zeichen - Buchstaben, Ziffern - anweist. Das kann eine einfache Rechenformel sein, aber auch

eine Folge von Rechenformeln, die zusammengenommen schon ein kleines Rechenprogramm sind. Auch ein Programm ist für Framework demnach eine Formel.

Der Formelspeicher ist nicht direkt zugänglich. Er muß mit **[F2]** (*Formel editieren*) erst geöffnet werden - mehr dazu weiter unten.

Framework unterscheidet zwischen verschiedenen Frame-Typen, orientiert nach den Anwendungsbereichen. Es gibt Frames für Konzepte, für Textverarbeitung, für Tabellen, für Datenbank und für Grafik. Jeder Frame-Typ legt die möglichen Funktionen fest. In einem Text-Frame eine Datenbank anzulegen und die dann zu sortieren und zu selektieren - das geht nicht.

Gleichwohl kann man einen Text mit einer Datenbank verbinden, als spiele sich alles innerhalb desselben Frames ab: Das macht Framework zu einem integrierten Programmpaket.

Frames laden

Mit Frames kann man allerhand machen. Um das nachzuvollziehen, brauchen Sie einige Dateien von der Beispieldiskette - wie Sie zu ihnen kommen, ist schon der erste Teil der Lektion.

1. Sie haben bereits das Dateiverzeichnis geöffnet. Der Cursor befindet sich auf dem Rand des Frames.
2. Drücken Sie **[NUM+]** - der Cursor bewegt sich in den Frame hinein.
3. Bringen Sie den Cursor auf die Datei **FRAME_1.FW**.
4. Drücken Sie **[RETURN]**.

Die Datei wird nun in den Arbeitsspeicher des Computers geladen und auf der Framework-Arbeitsfläche im Dokumentenstapel abgelegt.

Eine Datei reicht noch nicht, wir brauchen mehrere:

1. Drücken Sie **[SCROLL LOCK]** - der Cursor springt vom Dokumentenstapel zum Dateiverzeichnis und dorthin, wo er zuletzt war, in unserem Fall also in das Dateiverzeichnis hinein.

■ **Merke:** **[SCROLL LOCK]** schaltet um zwischen Dateiverzeichnis und Dokumentenstapel.

2. Bringen Sie den Cursor auf die Datei **FRAME_2.FW**.
3. Drücken Sie **[F6]** (*Auswahl*).
4. Führen Sie den Cursor abwärts, bis er auf der Datei **FRAME_4.FW** steht.
5. Drücken Sie **[RETURN]** - damit sind drei Dateien hervorgehoben.
6. Drücken Sie **[RETURN]** - damit wird die Auswahl abgeschlossen.
7. Drücken Sie **[RETURN]** - damit werden alle drei Dateien nacheinander geladen und auf dem Dokumentenstapel abgelegt.

Dateien, die im Dokumentenstapel angezeigt werden, befinden sich im Arbeitsspeicher des Computers. Auf der Diskette sind sie deswegen nicht verschwunden: Framework lädt lediglich eine Kopie.

Frames öffnen und schließen

Um mit den Frames arbeiten zu können, müssen sie zuerst geöffnet werden:

1. Bringen Sie den Cursor auf FRAME_1 im Dokumentenstapel.
2. Drücken Sie **[RETURN]**.

Der Frame wird geöffnet. Noch befindet sich der Cursor auf dem Rahmen des Frames - was uns zupaß kommt, denn dort muß er auch sein, wenn der Frame wieder geschlossen werden soll:

1. Achten Sie darauf, daß sich der Cursor auf dem Rahmen des Frames befindet.
2. Drücken Sie **[RETURN]**.

Öffnen Sie den gleichen Frame erneut, und die anderen alle auf einen Schlag - auf die gleiche Weise, wie sie geladen worden sind:

1. Bringen Sie den Cursor auf die Datei FRAME_2.FW.
2. Drücken Sie **[F6]** (*Auswahl*).

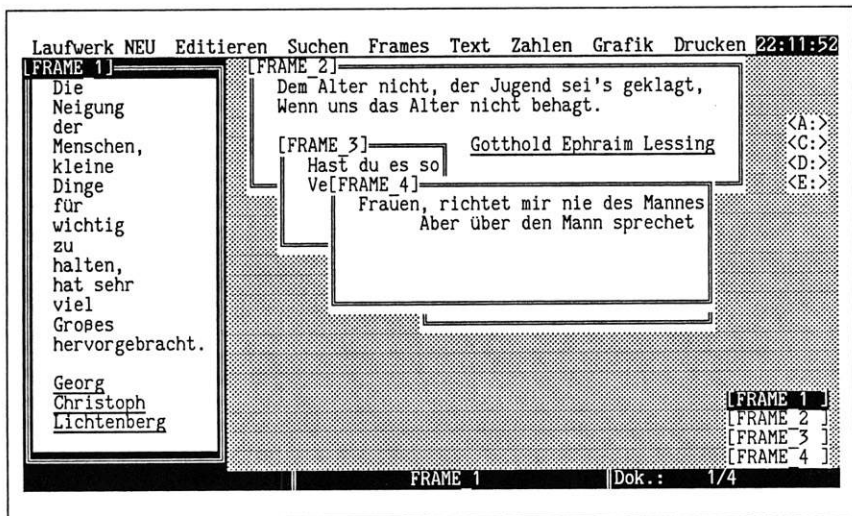


Bild 13.2: Frames unterschiedlicher Größe, sich überlappend

3. Führen Sie den Cursor abwärts, bis er auf der Datei FRAME_4.FW steht.
4. Drücken Sie **RETURN** - damit sind drei Dateien im Dokumentenstapel hervorgehoben.
5. Drücken Sie **RETURN** - damit wird die Auswahl abgeschlossen.
6. Drücken Sie **RETURN** - damit werden alle drei Dateien gemeinsam geöffnet.

Wenn man Frames gemeinsam öffnen kann, muß man sie eigentlich auch gemeinsam schließen können - und lernt gleichzeitig, wie man den Cursor von Frame zu Frame bewegt:

1. Der Cursor steht noch auf der Datei FRAME_4.FW.

Merke: Der Frame, auf dem der Cursor steht, wird der aktuelle Frame genannt; sein Rahmen und sein Name sind hell unterlegt.

2. Drücken Sie **F6** (*Auswahl*).
3. Drücken Sie die Cursortaste **↑** dreimal. Sie sehen, wie der Cursor von Frame zu Frame hüpf und gleichzeitig die Dateien im Dokumentenstapel markiert werden.
4. Drücken Sie **RETURN** - damit wird die Auswahl abgeschlossen.
5. Drücken Sie **RETURN** - damit werden alle drei Dateien gemeinsam geschlossen.

Im Dokumentenstapel bleiben die Dateien markiert, so daß es ein Leichtes ist, sie wiederum gemeinsam zu öffnen - und sie ein weiteres Mal, auf anderem Wege, zu schließen:

1. Öffnen Sie das Menü *Laufwerk*.
2. Aktivieren Sie die Funktion *Arbeitsfläche räumen*.

Wie's die Nachrichtenzeile besagt: Alle offenen Frames werden geschlossen und auf dem Dokumentenstapel abgelegt.

Frames vergrößern, verkleinern und verschieben

Sie haben nunmehr, wenn Sie die Frames ein weiteres Mal öffnen, vier Frames auf der Arbeitsfläche, die sich teilweise überlappen und deren Inhalt nicht immer zu lesen ist. Weshalb Sie nun daran gehen, die Frames zu vergrößern und zu verschieben.

1. Bringen Sie den Cursor auf FRAME_4.
2. Drücken Sie **F4** (*Größe*).
3. Drücken Sie des öfteren die Cursortasten und vergrößern Sie damit den Frame, wie es Ihnen richtig scheint.

Merke: **→** und **↓** vergrößern den Frame, **←** und **↑** verkleinern ihn. In Verbindung mit der **CTRL**-Taste geht's etwas schneller. Zweimal **F4** vergrößert auf die ganze Bildschirmgröße.

4. Drücken Sie **RETURN**.
5. Drücken Sie **F3** (*Position*).
6. Drücken Sie des öfteren die Cursortasten und verschieben Sie damit den Frame.
7. Drücken Sie **RETURN**.

Tip: Lassen sich Frames nicht verschieben, muß zuvor *Position frei wählbar* im Menü *Frames* aktiviert werden.

So können Sie sich Ihre Arbeitsfläche einrichten, wie es beliebt und wie es notwendig ist, damit Sie beispielsweise den Inhalt aller Frames lesen können.

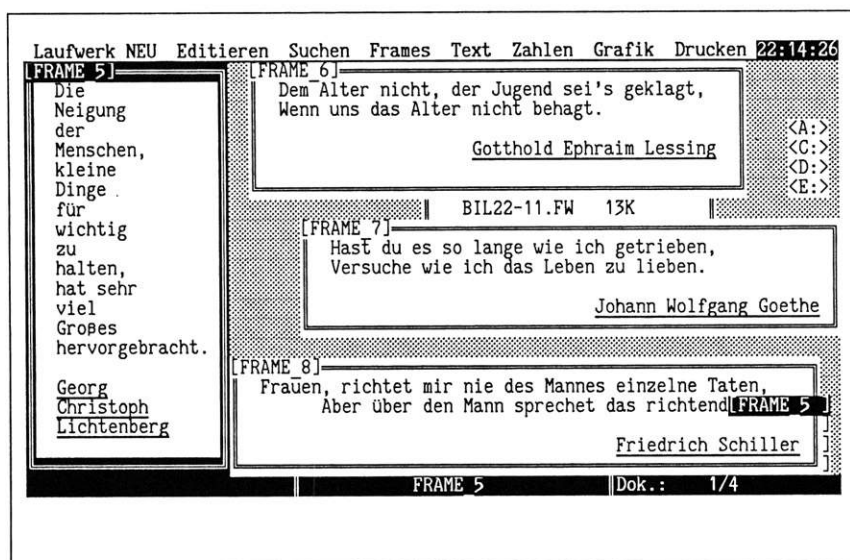


Bild 13.3: Die Frames, neu angeordnet

Frames speichern und sichern

Ein besonderes Merkmal von Framework ist, daß Frames die Größe und Position behalten, mit denen sie gespeichert oder gesichert werden.

Merke: Beim Speichern wird die Datei auf die Diskette oder Platte geschrieben und verschwindet aus dem Dokumentenstapel. Beim Sichern wird die Datei ebenfalls auf Diskette oder Platte geschrieben, verbleibt aber auf der Arbeitsfläche und im Dokumentenstapel.

Das Speichern geht nur über ein Menü:

1. Bringen Sie den Cursor auf den Rand des Frames oder markieren Sie mehrere Frames mit **[F6]** (*Auswahl*); dabei ist es gleichgültig, ob die Frames geöffnet oder geschlossen sind.
2. Aktivieren Sie *Weglegen* im Menü *Laufwerk*.

Das Sichern geht genauso, erlaubt aber noch eine andere, schnellere Methode:

1. Bringen Sie den Cursor auf den Rand des Frames oder markieren Sie mehrere Frames mit **[F6]** (*Auswahl*); wiederum ist es gleichgültig, ob die Frames geöffnet oder geschlossen sind.
2. Aktivieren Sie *Zwischendurch abspeichern* im Menü *Laufwerk* oder drücken Sie (gleichzeitig!) **[CTRL]** **[RETURN]**.

Haben Sie zwischenzeitlich die Diskette gewechselt, fragt Framework, ob trotzdem gesichert bzw. gespeichert werden soll.

Frames erstellen und benennen

Für eine der nächsten Übungen benötigen wir einen neuen, leeren Frame. Wobei wir uns zunächst entscheiden müssen, welche Art von Frame.

Framework unterscheidet zwischen fünf Frame-Typen (die jetzt nicht näher erläutert, sondern nur aufgezählt werden):

- Konzept-Frame
- Text-Frame
- Tabellen-Frame
- Datenbank-Frame
- Grafik-Frame

Ein Grafik-Frame kann nicht direkt erstellt werden, er braucht eine Tabelle oder Datenbank als Grundlage, deren Daten er umsetzt.

Die anderen Frame-Typen werden über das Menü *NEU* ausgewählt - und zwar, bevor man den Frame erstellt. Einen schon bestehenden Frame in einen anderen Frame-Typ zu verwandeln, ist nicht möglich.

1. Öffnen Sie das Menü *NEU*.
2. Aktivieren Sie die Funktion *Konzept*.
3. Öffnen Sie das Menü abermals.
4. Aktivieren Sie die Funktion *Frame (leer/Text)*.

Fahren Sie so fort, bis Sie Frames aller vier Typen erstellt haben.

Tip: Die Ausgangsgröße eines neuen Frames kann in *CONFIG.FW* bestimmt werden.

Bei Tabellen- und Datenbank-Frames kann zuvor gewählt werden, wieviel Spalten oder Zeilen sie haben sollen. Die Voreinstellung ist im Menü *NEU* abzulesen (und wird in *CONFIG.FW* festgelegt - siehe Kapitel 8): 14 Spalten, 14 Felder.

Man kann natürlich auch bestehende Tabellen und Datenbanken erweitern; doch wenn man von vornherein weiß, daß die Voreinstellungen zu niedrig sind, verfährt man so:

1. Aktivieren Sie *Breite* bzw. *Länge* im Menü *NEU*. Daraufhin öffnet sich die Editierzeile.
2. Geben Sie die neuen Werte ein.
3. Drücken Sie **[RETURN]**.
4. Erstellen Sie jetzt den Tabellen- oder Datenbank-Frame.

Um Frames identifizieren zu können, brauchen sie Namen:

1. Bringen Sie den Cursor auf den Rand eines leeren Frames.
2. Beginnen Sie zu schreiben. Der Cursor springt automatisch in die Editierzeile. Es ist gleichgültig, ob Sie Groß- oder Kleinbuchstaben verwenden.
3. Beenden Sie die Eingabe mit **[RETURN]**.

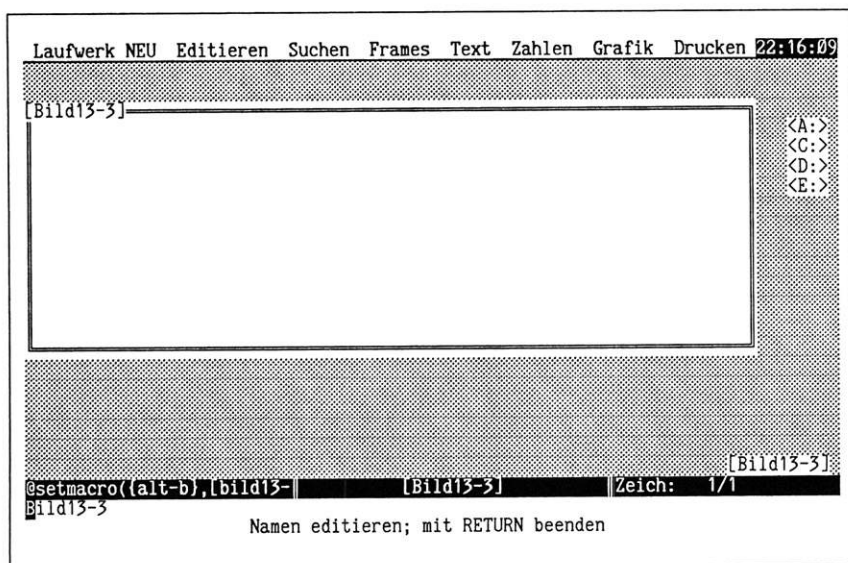


Bild 13.4: Ein Konzept-Frame mit editierter Namenszeile

Bei Framework darf der Name 32000 Zeichen lang sein - ein halber Roman. Das Betriebssystem MS-DOS erlaubt jedoch höchstens acht Zeichen - folglich werden, wenn

die Datei gespeichert wird, die Zeichen 9 bis 32000 einfach abgeschnitten. Infolgedessen müssen Sie wenigstens die ersten acht Zeichen des Frame-Namens so wählen, daß sie sich von allen anderen Namen unterscheiden. Was über acht Zeichen hinausgeht, kann man dazu nutzen, eine Datei näher zu kennzeichnen.

Tip: Auch im Dokumentenstapel werden nur die ersten acht Zeichen eines Frame-Namens angezeigt. Daran kann man überprüfen, ob der gewählte Name kennzeichnend genug ist.

Ein Frame-Name ist leichter geändert, als ein Datei-Name auf Betriebssystem-Ebene:

1. Bringen Sie den Cursor auf den Rahmen des Frames.
2. Drücken Sie die Leertaste. Der Name erscheint in der Editierzeile.
3. Ändern Sie den Namen.
4. Beenden Sie mit **RETURN**.

Erinnern Sie sich, daß alle Frames, die auf der Arbeitsfläche sind, sich im Arbeitsspeicher des Computers befinden. Der umgetaufte Frame ist demzufolge unter seinem neuen Namen ebenfalls nur im Arbeitsspeicher. Wenn Sie ihn jetzt sichern oder speichern, haben Sie auf der Diskette bzw. Platte zwei identische Dateien mit unterschiedlichen Namen.

Tip: Benutzen Sie die Namensänderung, um von einem Frame eine Kopie (unter anderem Namen) zu erstellen.

Manchmal erfordern bestimmte Operationen bestimmte Frame-Namen - unter denen man die Dateien allerdings nicht speichern kann, weil eine andere Datei gleichen Namens bereits existiert. Also ändert man rasch den Namen (es genügt ja, ihn vor den alten zu setzen) -, wenn man den Frame speichert.

Frames löschen

Das *Weglegen* aus dem Menü *Laufwerk* ist manches Mal eine umständliche Sache: dann, wenn man einen Frame nur angeschaut, aber nichts geändert hat. Der Frame läßt sich auf andere Weise schneller von der Arbeitsfläche (und aus dem Arbeitsspeicher) entfernen:

1. Bringen Sie den Cursor auf den Rahmen des Frames oder markieren Sie den Frame im Dokumentenstapel. (Sie können auch mehrere Frames markieren.)
2. Drücken Sie **DEL**.

Ist der Frame zuvor gespeichert worden, ist nichts verloren, andernfalls alles, was Sie seit dem letzten Sichern mit dem Frame unternommen haben. Nochmals: Lediglich im Arbeitsspeicher ist die Datei gelöscht worden, nicht auf der Diskette.

Tip: Wollen Sie eine Datei auf der Diskette oder Platte löschen, müssen Sie in den Dateiverzeichnis-Frame hineingehen. Weil diese Löschung endgültig ist, läßt Framework noch die Chance zum Abbruch.

Frames verschachteln und verlagern

Weil Frames auch ineinander verschachtelt werden können, ist Framework ein integriertes Paket - unter anderem deswegen.

1. Erstellen (oder benutzen) Sie einen leeren(!) Text-Frame. Er soll die anderen Frames aufnehmen und wird daher als umfassender Frame bezeichnet.
2. Drücken Sie zweimal **F4** (*Größe*) und bringen Sie damit den Frame auf volle Bildschirmgröße.
3. Bringen Sie den Cursor auf FRAME_1 im Dokumentenstapel.
4. Drücken Sie **F8** (*Kopieren*) - Sie leiten damit den Kopiervorgang ein.
5. Bringen Sie den Cursor auf den Rand des leeren Frames.
6. Bringen Sie den Cursor mit **NUM+** in den Frame hinein.

Merke: **NUM+** bewegt den Cursor vom Rahmen in den Frame hinein, **NUM-** aus dem Frame heraus auf den Rahmen.

7. Drücken Sie **RETURN**. Damit befindet sich eine Kopie von FRAME_1 im leeren Frame - noch als kleines Rähmchen.
8. Drücken Sie **F4** (*Größe*) und **RETURN** - jetzt hat er seine vorherige Größe wieder.

Kopieren Sie auf die gleiche Weise FRAME_2 hinein - nachdem Sie zuvor mit **NUM-** den Cursor auf den Rand des umfassenden Frames gebracht haben.

Führen Sie den Cursor, an dem jetzt gleichsam die Kopie von FRAME_2 klebt, wieder in den umfassenden Frame hinein, springt er auf den Rahmen von FRAME_1. Drücken Sie trotzdem **RETURN** - FRAME_2 wird nicht in FRAME_1 kopiert (weil Sie ja auf, nicht in ihm waren), sondern neben ihn. Irgendwo im umfassenden Frame. Lassen Sie ihn vorerst dort.

Die nächsten beiden Frames wollen wir nicht kopieren, sondern verlagern. Der Vorgang ist der gleiche wie beim Kopieren, Sie verwenden lediglich eine andere Funktionstaste: **F7** (*Verlagern*).

Anders ist auch der Effekt: Sie haben die beiden Frames vom Dokumentenstapel weggenommen und in dem umfassenden Frame untergebracht. Dort können Sie nun alle Frames wieder beliebig verschieben und in der Größe ändern.

Zur Erinnerung: Trotz des Verlagerns sind die beiden Frames noch auf der Diskette vorhanden.

Um Frames überhaupt verschachteln zu können, muß eine Grundregel strikt beachtet werden:

- Ein umfassender Frame kann entweder Daten aufnehmen oder einen Sub-Frame, nicht beides zusammen.

In Kapitel 22 wird über das Verschachteln von Frames noch ausführlicher zu reden sein.

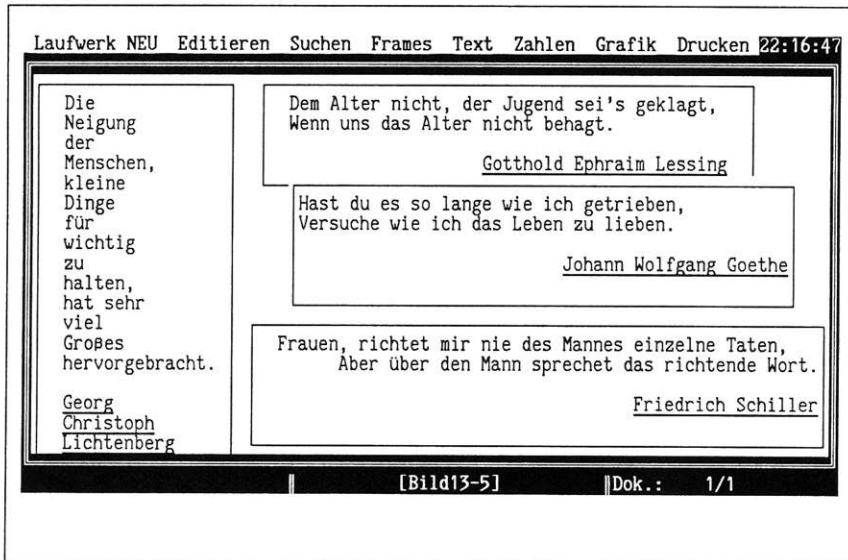


Bild 13.5: Verschachtelte Frames (in der Frame-Version)

Konzept- und Frame-Version

Verschachtelt man Frames, entsteht, was Framework ein »Konzept« nennt - auch darüber weiter unten mehr. Was uns jetzt interessiert, sind zwei unterschiedliche Darstellungsweisen eines umfassenden Frames mit seinen Sub-Frames: Konzept-Version und Frame-Version.

Was Sie jetzt auf Ihrem Bildschirm (und in der Abbildung 13.5) sehen, ist die Frame-Version: alle Sub-Frames sind geöffnet.

Im Gegensatz dazu die Konzept-Version:

1. Bringen Sie den Cursor auf den Rahmen des umfassenden Frames.
2. Drücken Sie **F10** (Zeigen).

Der umfassende Frame zeigt nunmehr allein die Namen der Sub-Frames an, nicht deren Inhalt - die Sub-Frames sind geschlossen. Geöffnet werden sie wiederum, alle-
samt, mit **[F10]** (*Zeigen*).

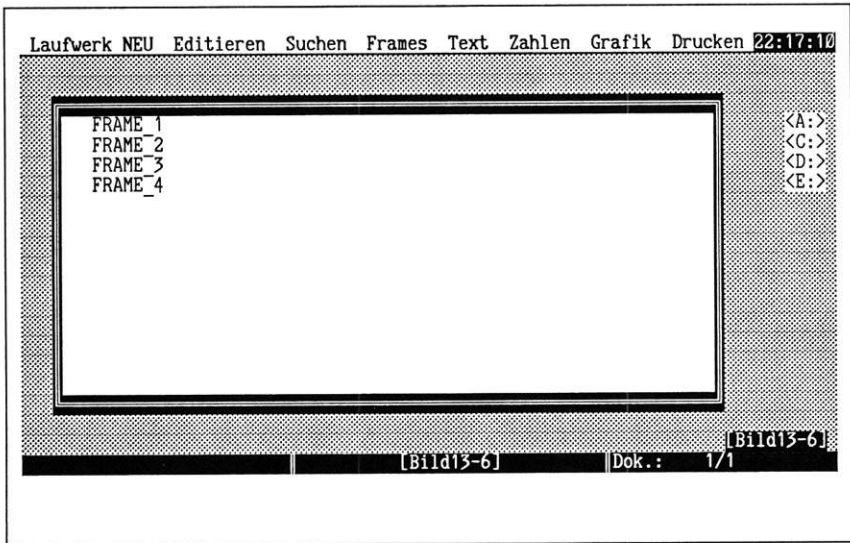


Bild 13.6: *Verschachtelte Frames in der Konzept-Version*

Frames zoomen

Bei der Arbeit mit verschachtelten Frames besteht oftmals keine Notwendigkeit, sämtliche Frames zu öffnen, um zum Beispiel in einem von ihnen komfortabel arbeiten zu können. Was heißt, daß man nicht auf einen Ausschnitt beschränkt sein möchte.

1. Bringen Sie den Cursor auf einen der Sub-Frames.
2. Drücken Sie **[F9]** (*Zoom*).

Der Frame erscheint jetzt in voller Bildschirmgröße. Genauer gesagt: sein Inhalt, sein Datenspeicher. Sein Rahmen ist verschwunden. Sichtbar machen Sie ihn wieder mit erneutem **[F9]** oder mit **[NUM-]** - beides hat den gleichen Effekt.

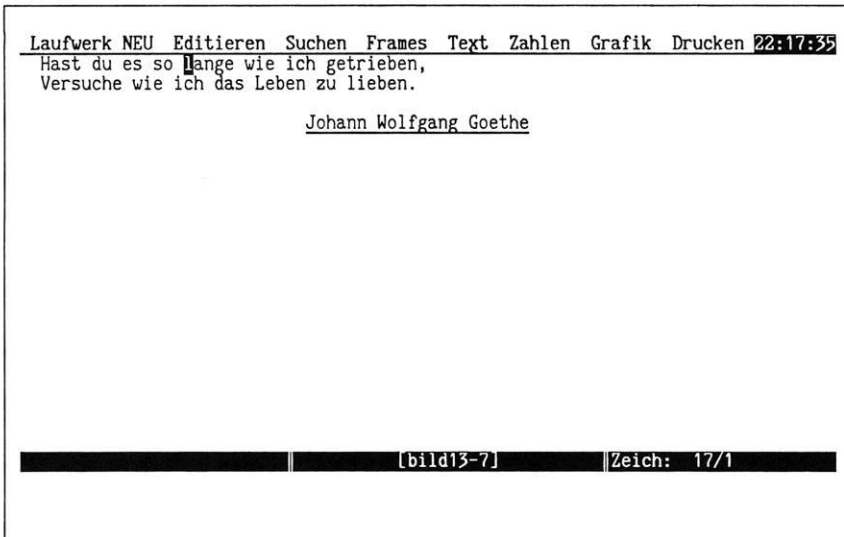


Bild 13.7: *Ein gezoomter Frame*

14 Cursorbewegungen

Als integriertes Paket arbeitet Framework mit einer Benutzeroberfläche, die in allen Programmteilen identisch ist. Dazu gehören die Frames, die Menüs und ebenso die Tastenkombinationen, mit denen man den Cursor über kürzere oder weitere Strecken bewegt. Ob Text, Tabelle oder Datenbank: die Cursorbewegungen sind gleich oder korrespondieren zumindest.

Die folgende Tabelle zeigt es.

Taste	Text	Tabelle	Datenbank
	Zeichen rechts	Spalte rechts	Feld rechts
	Zeichen links	Spalte links	Feld links
	Zeile unten	Zeile unten	Datensatz unten
	Zeile oben	Zeile oben	Datensatz oben
	Wort rechts	letzte/erste belegte Spalte	letztes belegtes Feld
	Wort links	erste/letzte belegte Spalte	erstes belegtes Feld
	nächster Satz	letzte belegte Zeile	letzter belegter Datensatz
	voriger Satz	erste belegte Zeile	erster belegter Datensatz
	Anfang Zeile	erste Zelle der Zeile	erstes Feld des Satzes
	Ende Zeile	letzte Zelle der Zeile	letztes Feld des Satzes
	Anfang Text	erste Zelle	erstes Feld
	Ende Text	letzte Zelle	letztes Feld
	vorwärts blättern	vorwärts blättern	vorwärts blättern
	zurück blättern	zurück blättern	zurück blättern
	nächster Absatz	nächste Zeile	nächster Datensatz
	voriger Absatz	vorige Zeile	voriger Datensatz
	TAB-Position	Spalte rechts	Feld rechts
		Spalte links	Feld links
	(lösche Zeichen)	Spalte links	Feld links
	(Absatzende)	Zeile abwärts	Datensatz abwärts

Daran gewöhnen muß man sich, daß Tabelle und Datenbank unterscheiden zwischen dem Gesamtbereich und dem belegten Bereich. Der Unterschied dürfte nicht allzu schwer zu fassen sein: Der belegte Bereich ist jener, der tatsächlich Daten enthält.

Befindet sich ein Datenfeld in der Editierzeile, gleich, ob Text oder Zahl, gelten dafür die Text-Cursorbewegungen, ebenso für den Formelspeicher.

15 Begriffe

Die Arbeit mit einem Computer-Programm ist oftmals wie die Reise in ein fremdes Land: Man wird mit einem Kauderwelsch konfrontiert, das zu entschlüsseln man Mühe hat.

Nicht nur computerspezifische Begriffe muß man verstehen, auch Sachbegriffe. Zum Beispiel bei Datenbanken und Tabellen.

Eine Tabelle besteht aus **Spalten**, die vertikal, und **Zeilen**, die horizontal verlaufen. Der Schnittpunkt von Spalte und Zeile ergibt eine **Zelle**.

In einer Datenbank wird die Spalte **Feld** genannt, jede Zeile ist ein **Datensatz**, eine Zelle auch ein **Feld**.

■ **Zur Erinnerung:** Um Zellen und Felder orten zu können, gibt Framework im mittleren Teil der Statuszeile die Koordinaten an.

In Tabellen werden die Spalten mit Buchstaben, die Zeilen mit Zahlen bezeichnet. Die erste Zelle einer Tabelle, links oben, hat demnach die Koordinaten A1, die Zelle darunter A2.

Gemeinhin wird man in einer Tabelle auch Spalten- und Zeilenbezeichnungen verwenden. Man kann Framework veranlassen, diese Bezeichnungen in der Statuszeile auszugeben statt der Koordinaten. Die Zelle heißt dann beispielsweise »Januar.Äpfel« statt B2. Umgeschaltet zwischen der einen und anderen Form wird mit dem Ausrufungszeichen.

Eine Datenbank arbeitet nicht mit Koordinaten. Die erste Zeile einer Datenbank ist reserviert für Feldbezeichnungen, und diese dienen zur Identifikation eines Feldes. Was in der Natur der Sache liegt. Eine Datenbank hat nur einen Sinn, wenn die gleichen Felder jedes Datensatzes gleichartige Informationen erhält.

16 Markieren, Kopieren und anderes

Auch die Methoden des Markierens, Kopierens und Verlagerns sind in allen Programmbereichen gleich, und die Prinzipien gelten nicht bloß für Daten, sondern auch für Frames - Sie haben das schon erfahren.

Wenn Sie die Erläuterungen dieses Kapitels nachvollziehen, aber selber nichts schreiben wollen, holen Sie sich von der Beispieldiskette die Dateien

- TEXT.FW
- TAB.FW
- BANK.FW

auf die Arbeitsfläche. Die drei Frames sind so eingerichtet, daß sie nebeneinander auf die Arbeitsfläche passen, so daß Sie zwischen den Frame-Typen hin- und herspringen können.

Zur Erinnerung: **NUM+** bewegt den Cursor vom Rahmen in den Frame hinein, **NUM-** aus dem Frame heraus auf den Rahmen.

Markieren

Was auf dem Bildschirm hell unterlegt ist, das ist markiert. Framework nennt darum den Cursor »Marker« - eine ebenso unglückliche Eindeutschung wie alle anderen, die hin und wieder versucht werden.

Auch der Cursor markiert: ein einzelnes Zeichen in einem Text, eine Zelle in einer Tabelle, ein Feld in einer Datenbank - die kleinsten Einheiten im jeweiligen Frame-Typ. Will man mehr markieren, muß man erweitern:

1. Bringen Sie den Cursor an den Anfang des zu markierenden Bereiches - gleich, ob das ein Zeichen, eine Zelle oder ein Feld ist.
2. Drücken Sie **F6** (*Auswahl*).
3. Bringen Sie den Cursor an das Ende des zu markierenden Bereiches.
4. Drücken Sie **RETURN**.

Der markierte Bereich (man spricht auch von einem Block) ist nunmehr hervorgehoben.

Natürlich macht man sich bei umfassenderen Markierungen die Cursorsprünge zunutze - mit **CTRL** **END** an das Ende eines Textes, einer Tabelle oder einer Datenbank zu kommen, ist allemal bequemer, als den Cursor Zeile um Zeile abwärts zu bewegen.

In einem Text-Frame ist das Markieren von Framework auf eine raffinierte Weise gelöst, die selbst manche luxuriösen Textprogramme vor Neid erblassen lassen: Die Cursorsprünge dienen gleichzeitig als Markierung. So hat man die Möglichkeit, ein Wort, einen Satz oder einen Absatz zu markieren, ohne deswegen **[F6]** (*Auswahl*) bemühen zu müssen.

Etwas Ähnliches ist in einer Tabelle, einer Datenbank und seltsamerweise bei einer Formel im Formelspeicher möglich, wenn man eine ganze Zeile bzw. Datensatz markieren will.

Das erledigen **[CTRL][PGUP]** bzw. **[CTRL][PGDN]** - ärgerlicherweise wird aber in Tabelle oder Datenbank nur die Zeile darüber oder darunter angesprochen, nicht die aktuelle Zeile. Indes, die Markierung bleibt erhalten, solange der Cursor nur auf- oder abwärts bewegt wird, so daß man sie auf die richtige Zeile verschieben kann.

■ **Zur Erinnerung:** Mit **[F9]** zoomt man einen Frame auf Bildschirmgröße.

Was markiert worden ist, das ist aus der Masse des anderen herausgehoben, zu einem bestimmten Zweck: Mit dem markierten Bereich hat man etwas Besonderes vor.

Löschen

Zum Beispiel kann man ihn löschen: mit **[DEL]**. Bei einem Text wird die Lücke sofort wieder gefüllt. Was nach dem markierten (und jetzt nicht mehr vorhandenen) Text kam, wird nach vorne geschoben.

In einer Tabelle oder Datenbank jedoch werden lediglich die Daten (und die mit ihnen verknüpften Formeln) gelöscht, die Zeilen oder Spalten bleiben erhalten - leer nunmehr. Will man auch diese entfernen, ob leer oder gefüllt, muß man dies über das Menü *Editieren* tun - und vorher nötigenfalls den Bereich markieren.

Die Löschung rückgängig machen

Was gelöscht worden ist, stellt Framework zunächst einmal in einen internen Zwischenspeicher, in der richtigen Erkenntnis, daß so manche Löschung unabsichtlich geschieht.

Mit der Funktion *Rücknahme* im Menü *Editieren* kann die Löschung (oder eine andere Aktion) wieder rückgängig gemacht werden. Aber:

- nur die letzte Aktion kann rückgängig gemacht werden
- nur Daten können zurückgeholt werden, entfernte Zeilen oder Spalten nicht
- bei manchen Operationen benötigt Framework viel Platz im Arbeitsspeicher und löscht daher den Zwischenspeicher; das wird allerdings angekündigt.

Tip: Falls die Rücknahme nicht mehr geht: löschen Sie den Frame, laden Sie ihn neu. Allerdings sind dann sämtliche Änderungen seit dem letzten Sichern verloren.

Kopieren

Beim Kopieren verbleibt der markierte Bereich an seiner Ursprungsstelle, am Zielort wird eine Kopie eingefügt.

1. Markieren Sie den gewünschten Bereich.
2. Drücken Sie **F8** (*Kopieren*).
3. Gehen Sie mit dem Cursor an den Zielort.
4. Drücken Sie **RETURN**.

Tip: Auch nach dem Kopieren bleibt die Markierung erhalten. Wenn Sie jetzt gleich **F8** drücken, können Sie den Block abermals kopieren, ohne neu markieren zu müssen.

In einem Text-Frame wird tatsächlich eingefügt, es sei denn, im Menü *Editieren* ist *Vorhandenes überschreiben* aktiviert worden.

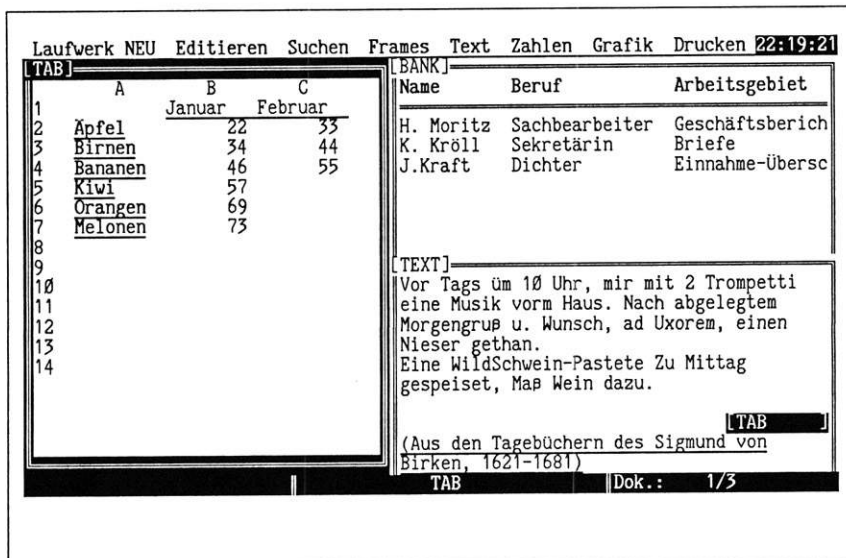


Bild 16.1: Text, Tabelle, Datenbank mit Schriftauszeichnungen

In einer Tabelle oder Datenbank hingegen wird der Inhalt der Zelle oder des Feldes stets überschrieben - das heißt, die Kopie ersetzt den vorigen Inhalt. Ist das nicht erwünscht, und meist ist es nicht, muß zuvor nötigenfalls über das Menü *Editieren* eine neue (leere) Zeile oder Spalte eingefügt werden.

Kopieren kann man auch, das haben Sie schon erlebt, ganze, geöffnete oder geschlossene Frames. Will man Frames nicht verschachteln, kann man Frameworks Frage nach dem Kopier-Ziel ignorieren und sogleich **RETURN** drücken.

Verlagern

Beim Verlagern wird der markierte Bereich verschoben: an seinem Ursprungsort entfernt, an seinem Zielort eingefügt.

Und wiederum: in Text-Frames wird eingefügt, in Tabellen und Datenbanken hingegen überschrieben.

|| **Merke:** Eingefügt, ob beim Kopieren, Verlagern oder bei der Vergrößerung einer Tabelle/Datenbank, wird stets nach der Cursorposition.

Auszeichnen

Markierten Bereichen kann auch eine andere Schriftart zugewiesen werden, und zwar in jedem Frame-Typ, egal, ob Text oder Zahlen:

1. Markieren Sie den Bereich.
2. Öffnen Sie das Menü *Text*.
3. Aktivieren Sie *Normalschrift*, *Fettdruck*, *Unterstreichen* oder *Schrägschrift*.

Fettdruck, *Unterstreichen* oder *Schrägschrift* kann beliebig kombiniert werden, allerdings nicht auf einmal: für jede Auszeichnung ist ein eigener Arbeitsgang erforderlich. Das ist etwas lästig.

Auch die Lage eines markierten Bereiches kann bestimmt werden:

- linksbündig (rechts flatternd)
- rechtsbündig (links flatternd)
- zentriert (in der Mitte)
- Blocksatz (links- und rechtsbündig).

Blocksatz, das leuchtet ein, ist nur bei Text möglich, und ohnehin sind für solche Definitionen verschiedene Menüs zuständig:

- für Text-Frames das Menü *Text*
- für Tabellen und Datenbanken das Menü *Zahlen*, wobei hier auch gewählt werden kann, ob Texte linksbündig, Zahlen hingegen rechtsbündig stehen sollen.

Derlei Auszeichnungen und Formatierungen betreffen nur den markierten Bereich - was Abwechslungen erlaubt. Soll beispielsweise in einer Tabelle alles linksbündig ausgerichtet werden, muß man erst auf den Rand des Frames gehen. Aber: Ist in der Tabelle einzelnen Zellen bereits ein anderes Format zugewiesen worden, wird das durch die generelle Formatierung nicht aufgehoben.

17 Daten- und Formeleingabe

Das Schreiben von Text ist weiter kein Problem: wie auf einer Schreibmaschine. In seiner Grundeinstellung ist Framework auf »Einfügen« geschaltet: vorhandener Text wird nicht überschrieben, das Neue wird hinzugefügt. Anders, wenn man im Menü Editieren die Funktion Vorhandenes überschreiben aktiviert.

Bei Tabellen und Datenbanken ist die Sache etwas komplizierter. Dort wird nämlich zwischen Text und numerischen Werten unterschieden, wobei auch Zahlen Text sein können.

Hinweis: Auf den Unterschied zwischen Text als Text und Zahl als Text wird in Kapitel 21 detailliert eingegangen.

Grundsätzlich gilt für die Eingabe von Text:

- Besteht der Text nur aus Buchstaben, schreiben Sie einfach drauf los. Framework schaltet automatisch um in die Editierzeile.
- Enthält der »Text« auch Zahlen oder eines dieser Zeichen:

@ + - (# \$! "

muß vor der Eingabe die Leertaste gedrückt werden, worauf Framework wiederum in die Editierzeile hüpfte; andernfalls versteht Framework die Zeichen als Teil einer Formel.

Für numerische Werte, gemeinhin als Zahlen bekannt, mit denen man rechnet, sieht das so aus:

- Besteht die Eingabe nur aus Zahlen, schreiben Sie ebenfalls einfach drauf los. Framework geht in die Editierzeile und nennt das eine »implizite Umschaltung«
- Beginnt die »Zahl« mit einem dieser Zeichen:

@ + - (# \$! "

kann gleichfalls sofort geschrieben werden.

- Alternativ dazu die »explizite Umschaltung«: vor der Dateneingabe **[F2]** (*Formel editieren*) drücken.

Analog das Editieren eines Wertes:

- bei Text: Leertaste drücken
- bei Zahlen oder Formeln: **[F2]** (*Formel editieren*) drücken

Beidesmal wird der Inhalt einer Zelle oder eines Feldes in der Editierzeile ausgegeben und kann dort verändert werden.

Werden (Rechen-)Formeln eingetragen, muß stets mit **[F2]** der Formelspeicher geöffnet und mit **[F9]** (*Zoom*) auf volle Bildschirmgröße erweitert werden, falls die Formel

für die Editierzeile zu lang ist. Damit werden wir uns im Verlauf dieses Buches noch häufiger beschäftigen; nur mal zum Üben:

1. Bringen Sie den Cursor in die Tabelle TAB und dort in die Zelle B8.
2. Öffnen Sie den Formelspeicher mit **[F2]** (*Formel editieren*).
3. Schreiben Sie:
 @sum(B2 : B8)
4. Drücken Sie **[RETURN]**.

Merke: **[RETURN]** am Ende einer Formel in der Editierzeile führt die Formel gleich aus.

In der Zelle B8 wird daraufhin die Summe der Spalte *Januar* errechnet.

18 Drucken

Framework erlaubt differenzierte Möglichkeiten des Ausdruckes, was aus der Frame-Philosophie resultiert. Grundsätzlich ist der Weg immer gleich:

1. Man bestimmt, was gedruckt werden soll.
2. Man wählt, wo gedruckt werden soll - im Menü *Drucken* das (Sub-Menü) *Ziel für die Ausgabe*.
3. Man definiert, wie gedruckt werden soll - im Menü *Drucken* die *Optionen für Ausdruck*, gleichfalls ein Sub-Menü.
4. Man beginnt den Druck mit *Starten* im Menü *Drucken*.

Was gedruckt werden soll, muß markiert werden. Zum Beispiel, mit **F6** (*Auswahl*), mehrere Frames - Framework sammelt die Druckaufträge und arbeitet sie hintereinander ab. Während dessen kann man mit Framework weiterarbeiten, wenn auch unter Umständen etwas langsamer.

Zur Erinnerung: In CONFIG.FW kann das Verhältnis zwischen Druck- und Arbeitsgeschwindigkeit eingestellt werden.

Oftmals indessen handelt es sich auch um einen verschachtelten Frame. Auch hier gilt die Regel: was markiert ist, wird gedruckt. Das kann nur ein Sub-Frame sein (dann steht der Cursor dort); das können mehrere Sub-Frames sein (dann werden sie markiert); das kann der umfassende Frame samt seinen Sub-Frames sein (dann müssen alle Frames mit **F10** (*Zeigen*) geöffnet werden). Und zusätzlich kann man im Sub-Menü *Optionen für Ausdruck* noch differenzieren.

An einem Beispiel klargemacht, um die verschiedenen Möglichkeiten zu ersehen. (Wenn Sie mitmachen wollen, dann verwenden Sie dazu BERICHT.FW von der Beispieldiskette.)

Es handelt sich um einen verschachtelten Frame, eine Konzeptversion, dessen Sub-Frames teilweise einen Inhalt haben (Text, Tabelle, was auch immer), teilweise auch nicht, also nur aus ihrem Namen bestehen, die in diesem Fall Überschriften darstellen. Mit **F10** (*Zeigen*) sind zuvor alle Frames geöffnet worden:

- *Ohne geschlossene Frames*: nur die Inhalte der Sub-Frames werden gedruckt (so vorhanden), die Überschriften (die Namen der Sub-Frames) nicht
- *Ohne geschlossene Frames*, zusätzlich *Mit Namen der Frames* - das gleiche
- *nur Mit Namen der Frames* - alles wird gedruckt, Überschriften wie Inhalte

Bei verschachtelten Frames erfolgt der Ausdruck so, wie die Sub-Frames angeordnet sind. Das erlaubt, Frames auch nebeneinander zu stellen.

Tip: Sollen Sub-Frames als fortlaufendes Dokument gedruckt werden, empfiehlt es sich, sie mit *Ausrichten in Spalte* im Menü *Frames* anzuordnen. Sie folgen dann hintereinander.

Als Ausgabe-Ziel ist einer der beiden installierten Drucker oder der Plotter auszuwählen oder *Textdatei für DOS*. Dann wird die Datei auf Diskette geschrieben und kann in ein anderes Programm übernommen oder vor dem Druck nochmals angesehen werden. Diese Textdatei hat den gleichen Namen, jedoch die Endung PRT.

Auch über das Menü *Laufwerk* kann man eine *Textdatei für DOS* erstellen, die ebenfalls auf Diskette geschrieben wird und die Endung TXT bekommt.

Was ist der Unterschied?

Zunächst einmal: Die TXT-Datei ist sauber konvertiert. Alle Umlaute sind korrekt, es schwirren keine mysteriösen Sonderzeichen umher wie bei der PRT-Datei. Eine TXT-Datei ist der Notbehelf, wenn der eigene Drucker mit keinem der in Framework vorgesehenen kompatibel ist. Diese Datei kann jederzeit auf Betriebssystem-Ebene mit PRINT ausgedruckt werden - allerdings, ohne Auszeichnungen wie Fettdruck oder Schrägschrift in Anspruch nehmen zu können.

Des weiteren wandelt eine TXT-Datei eine Datenbank in ein Format um, das die Übernahme in ein anderes Datenbank- oder Textprogramm erlaubt. Als PRT-Datei erscheint die Datenbank nach wie vor in Tabellenform, als TXT-Datei wird jeder Datensatz in eine eigene Zeile geschrieben, die einzelnen Felder sind mit Kommata getrennt und stehen zwischen Anführungszeichen:

"J. Kraft", "Dichter", "Einnahme-Überschuß"

Sodann: eine TXT-Datei enthält nur Text, Datensätze oder Tabellen - nur, was im Datenspeicher eines Frames steht. Eine PRT-Datei jedoch kann auch aus den Formeln eines Frames bestehen - die Möglichkeit, ein Programm zu dokumentieren. In diesem Fall muß im Sub-Menü *Optionen für Ausdruck* die Funktion *Nur Formeln* aktiviert werden, am besten auch *Mit Namen der Frames*, damit man hinterher weiß, welche Formel wohin gehört.

Alltags- anwendungen

19 Vom Konzept zum Text

Sie erfahren in diesem Kapitel:

- wie man eine Konzept-Version erstellt und ausbaut
- wie man mit verschiedenen Gliederungsebenen umgeht
- wie man Texten und Überschriften Fettdruck, Schrägschrift oder Unterstreichungen zuweist
- wie man ein Inhaltsverzeichnis aufbaut
- wie man die Seitennumerierung kontrolliert
- wie man Kopf- und Fußzeilen einsetzt
- wie man ein Datum in unterschiedlichen Formaten ausgeben kann
- wie man u.a. folgende Funktionen verwendet:

@hr

@fr

@np

@date

Sie benötigen:

- KONZEPT.FW von der Beispieldiskette

Fortsetzung:

- Einbinden von Tabellen und Grafiken in den Text (Kapitel 22)

Das Problem

Der Sachbearbeiter Herwig Moritz von der AGCO AG, mittlere Führungsebene, ist beauftragt, einen Bericht zu verfassen über die Geschäftsentwicklung der vergangenen Jahre, unter besonderer Berücksichtigung der allgemeinen wirtschaftlichen Lage und Prognosen, wie sich diese wohl entwickeln möge und wie seine Firma mithin zu reagieren habe, damit die Gewinnmaximierung nicht leidet. Puh!

Ein recht komplexes Thema. Natürlich braucht Herwig Moritz dazu umfangreiches Zahlenmaterial - das wird er allerdings erst in Kapitel 22 bekommen. Vorerst macht ihm das Ausufernde des gesamten Stoffes mehr zu schaffen. Allerhand Ideen schwirren durch seinen Kopf, die er in seinem Bericht unterzubringen gedenkt - wirre Gedanken vorerst, ungeordnet und widersprüchlich. Wie das so ist, wenn man sich einem Thema erst nähert.

Wie bekommt Herwig Moritz das in Griff? Bei einem Kollegen hat er mal was gesehen. Der hat sich seine Einfälle auf kleine Zettel notiert und die sodann wie ein Puzzle geordnet. Herwig Moritz jedoch sind solche Spielereien zu fummelig, und überdies

befürchtet er, daß er damit den Überblick erst recht verliert: so viele Zettelchen, und ein so kleiner Schreibtisch!

Die Lösung

Framework besitzt eine Funktion, die derlei Zettelwirtschaft überflüssig macht. Bei Framework heißt sie Konzept, anderswo Gliederung oder Umriss-Editor, doch die Aufgaben sind immer die gleichen: Gedanken zu ordnen, zu gliedern, ein Konzept in Umrissen zu entwerfen - gleichsam ein Inhaltsverzeichnis zu erstellen, das zunächst nur aus Kapitelüberschriften besteht, die erst später um den dazugehörigen Text ergänzt werden.

Die Konzept-Version von Framework ist eine besondere Form der Darstellung, nicht zu vergleichen mit einem normalen Frame und muß daher eigens ins Leben gerufen werden: im Menü *NEU* mit der Funktion *Konzept*.



Bild 19.1: Ein Konzept-Frame mit verschiedenen Gliederungsebenen

Herwig Moritz hat das bereits getan - die Übungsdatei KONZEPT.FW entspricht in Aufbau und Umfang einem leeren Konzept-Frame, in den lediglich schon einige Kapitel-Überschriften eingetragen wurden; ansonsten ist an ihm nichts verändert.

Um mit einem Konzept-Frame arbeiten zu können, muß Herwig Moritz verstehen, wie er aufgebaut ist. Für den Anfang ist das nicht einfach, denn ein Konzept-Frame bündelt gleichsam eine der wesentlichsten Eigenschaften von Framework: die beliebige Verschachtelung von Frames.

|| **Merke:** Ein Konzept-Frame besteht aus anderen Frames, die in mehreren Ebenen verschachtelt sind.

KONZEPT.FW ist ein umfassender Frame, der andere Frames enthält. Je nach Tiefe der Gliederung stehen diese anderen Frames auf gleicher Ebene oder sind selber wieder umfassende Frames, die ihrerseits andere Frames enthalten.

Die Numerierung gibt Auskunft über die Ebene und über die Tiefe der Verschachtelung: »1 Die Wirtschaftslage« und »2 Die AGCO AG« stehen auf derselben Ebene und sind gleichzeitig umfassende Frames. Der Frame mit der Nummer 1 enthält drei Frames mit den Nummern 1.1, 1.2 und 1.3, die sich ihrerseits wieder auf der gleichen Ebene befinden.

|| **Merke:** Ein Dreieck links im Konzept-Frame zeigt auf einen umfassenden Frame, der Sub-Frames enthält.

Dank der Numerierung und den Einrückungen, die Framework in der Konzept-Version automatisch vornimmt, sind die Gliederungsebenen relativ einfach zu durchschauen. Sie folgen einem klaren Schema:

- 1 Erste Ebene
 - 1.1 Zweite Ebene
 - 1.2 Zweite Ebene
 - 1.2.1 Dritte Ebene
 - 1.2.2 Dritte Ebene
 - 1.2.3 Dritte Ebene
 - 1.2.3.1 Vierte Ebene
 - 1.3 Zweite Ebene
 - 1.3.1 Dritte Ebene
 - 1.4 Zweite Ebene
- 2 Erste Ebene
 - 2.1 Zweite Ebene

Man charakterisiert die Gliederungsebenen auch mit Verwandtschaftsverhältnissen: 1.1 und 1.2 sind die Kinder von 1 und demzufolge Geschwister, 1.2.1 usw. die Enkel von 1 und die Kinder von 1.2 - je größer die Familie wird, umso komplizierter wird das allerdings.

Die Kapitelüberschriften in der Gliederung von Herwig Moritz sind nichts anderes als die Namen der verschachtelten Frames und zu behandeln wie jeder andere Frame-Name auch:

1. Herwig Moritz bringt den Cursor auf den Punkt 3.3 - ein zwar schon vorhandener Frame, der jedoch noch nicht benannt worden ist.
2. Er trägt eine Kapitelüberschrift, also einen Frame-Namen ein, indem er einfach zu tippen beginnt, zum Beispiel »Zusammenfassung«.

Will er eine bestehende Kapitelüberschrift ändern, muß er sie mit der Leertaste editieren.

Tip: Wird die Eingabe einer Kapitelüberschrift (eines Frame-Namens) mit **TAB** abgeschlossen, springt der Cursor gleich zum nächsten Gliederungspunkt.

Weil ein Konzept-Frame aus verschachtelten Frames besteht, die Sub-Frames unter Umständen Sub-Sub-Frames enthalten, kann bestimmt werden, welche Gliederungsebene angezeigt werden soll: Frames werden geschlossen oder geöffnet.

1. Herwig Moritz bringt den Cursor in die erste Zeile, auf »Die Wirtschaftslage« - er bringt den Cursor damit auf den Rand eines umfassenden Frames, gekennzeichnet durch ein Dreieck.
2. Mit **RETURN** versteckt er dessen Sub-Frames - oder holt sie mit nochmaligem **RETURN** wieder hervor.

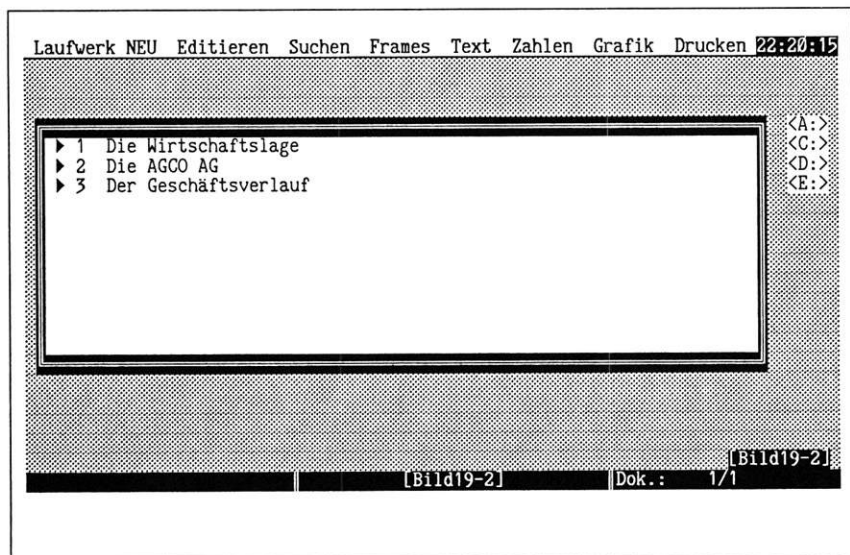


Bild 19.2: Ein Konzept-Frame, der nur die erste Gliederungsebene anzeigt

Das ist der Vorteil der Konzept-Version, den Herwig Moritz zu schätzen lernt: Er hat den Überblick über die grobe Gliederung seines Konzeptes und kann nach Wunsch umschalten in die Feingliederung.

Die Unterkapitel kleben gleichsam an ihrem umfassenden Frame - nicht weiter verwunderlich, sie sind ja der Inhalt des umfassenden Frames. Das führt Herwig Moritz zum nächsten Vorteil der Konzept-Funktion: Er kann ganze Komplexe verschieben, löschen oder kopieren. Macht er das mit einem umfassenden Frame, wirkt sich das zugleich auf dessen Sub-Frames aus.

1. Er bringt den Cursor auf den dritten Gliederungspunkt, »Der Geschäftsverlauf«, wobei es gleichgültig ist, ob die Unterpunkte sichtbar oder versteckt sind.
2. Mit **[F7]** (*Verlagern*) leitet er eine Verschiebe-Aktion ein.
3. Er bringt den Cursor auf den ersten Gliederungspunkt, »Die Wirtschaftslage«.
4. Mit **[RETURN]** verlagert er.

■ **Merke:** Eingefügt wird stets nach dem Cursor.

Framework hat dabei auch automatisch die Numerierung geändert: Der Komplex »Geschäftsverlauf« war vorher der dritte Gliederungspunkt und ist jetzt der zweite.

Auf die gleiche Weise läßt sich ein Komplex mit **[F8]** (*Kopieren*) auch duplizieren oder mit **[DEL]** löschen (was dann mit *Rücknahme* im Menü *Frames* wieder rückgängig gemacht werden kann) - stets bezieht eine solche Operation die Gliederungsebene(n) darunter mit ein, ohne daß diese eigens markiert werden müßten. Weshalb man bei einer tief verschachtelten Gliederung aufpassen muß, daß sich der Cursor auf der richtigen Ebene befindet.

■ **Tip:** Reduzieren Sie beim Verlagern oder Löschen die Gliederungsebenen. Dann gehen Sie sicher, daß Sie auch den richtigen Komplex erwischen.

Ein Konzept läßt sich jederzeit und (fast) immer und auf allen Ebenen erweitern und verändern, und Herwig Moritz beginnt nun, seine Gliederung auszubauen.

Zum Beispiel will er seinem Bericht eine Einleitung voranstellen, er muß also vor den Gliederungspunkt 1 einen neuen Frame einfügen. Dazu ist ein kleiner Umweg nötig, denn eingefügt wird ja stets nach dem Cursor.

1. Er bringt den Cursor auf den ersten Gliederungspunkt, »Die Wirtschaftslage« - und zwar auf diese (die erste) Zeile, denn er will einen Frame einfügen, der sich ebenfalls auf dieser Gliederungsebene befindet.
2. Mit *Frame (leer/Text)* im Menü *NEU* fügt er einen neuen, leeren Text-Frame ein - er bekommt automatisch die Nummer 2.
3. Er bringt den Cursor erneut auf den ersten Gliederungspunkt, »Die Wirtschaftslage«.
4. Er drückt **[F7]** (*Verlagern*), um diesen Komplex verschieben zu können.

5. Er bringt den Cursor auf den zweiten, noch nicht benannten Gliederungspunkt, der seine Einleitung werden soll.
6. Mit **[RETURN]** verlagert er den Komplex »Wirtschaftslage«, der bisherige Gliederungspunkt 2 rückt nach vorne und wird zum Punkt 1.
7. Er benennt diesen Frame mit »Einleitung«.

Tip: Achten Sie beim Einfügen von Gliederungspunkten darauf, daß Sie sich auf der richtigen Ebene befinden.



Bild 19.3: Ein umgestelltes, erweitertes Konzept

Auf diese Weise lassen sich andere Gliederungspunkte einfügen, egal, auf welcher Ebene - aber immer nur auf der gleichen Ebene.

Will Herwig Moritz hingegen sein Konzept weiter differenzieren, also neue Unter-ebenen erstellen, muß er zu einem anderen Verfahren greifen.

1. Er bringt den Cursor auf den Gliederungspunkt 4.1, »Die Stellung im Weltmarkt«.
2. Er drückt **[CTRL]** **[NUM+]**.

Merke: Neue Gliederungspunkte auf der gleichen Ebene werden mit einem Frame über das Menü *NEU* eingefügt. Eine neue, tieferliegende Gliederungsebene wird mit **[CTRL]** **[NUM+]** erreicht.

Damit hat er die neue Gliederungsebene 4.1.1 geschaffen, kann sie benennen, kann sie um den Punkt 4.1.2 ergänzen (indem er einen neuen Frame einfügt), kann nach Belieben weitere Ebenen tiefer gehen.

Framework numeriert brav und unerbittlich; ob für den Leser ein Gliederungspunkt 4.1.2.3.3.2.2.1 noch irgend einen Orientierungswert hat, ist eine andere Frage. Im Menü *Frames* läßt sich deshalb auch die *Durchnumerierung der Namen* ab- (und wieder an-) schalten. Wobei Herwig Moritz aufpassen muß: Auch diese Funktion wirkt sich auf alle Ebenen unterhalb der Cursorposition aus, aber nicht auf die darüberliegenden. Er muß deshalb entweder auf die gewünschte Ebene hochwandern oder mit **[F6]** (*Auswahl*) markieren. Auch ohne Numerierung ist indessen an den Dreiecken und an der typographischen Anordnung, an den immer weiter eingerückten Kapitelüberschriften, die Gliederung und ihr Zusammenhang ersichtlich.

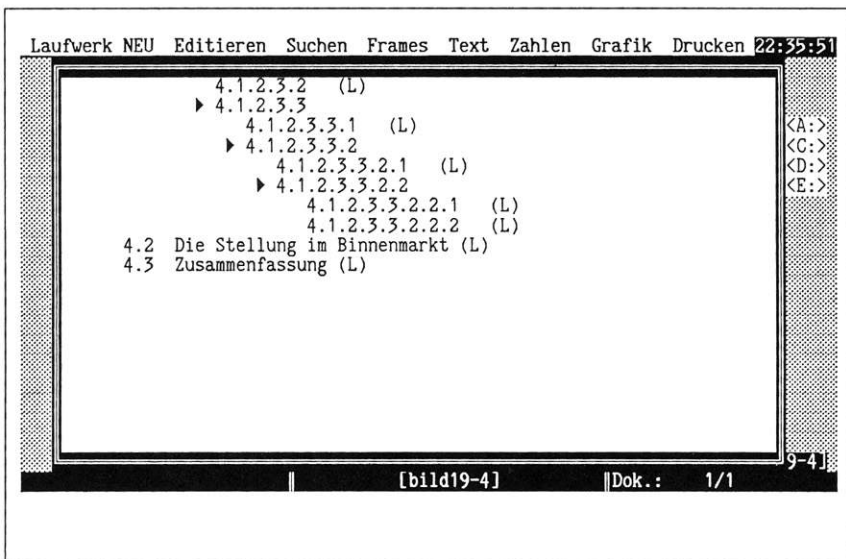


Bild 19.4: Eine tief verschachtelte Gliederung

Beim Blick auf seine Gliederung kommt Herwig Moritz eine Idee. Viel besser dünkt ihm, aus den zwei Unterabschnitten des Kapitels »AGCO AG« jeweils eigene Kapitel zu machen: Er muß sie also höherstufen.

Wäre dies ein normaler Text, könnte Herwig Moritz die Kapitelüberschrift löschen und müßte dann nur die Unterabschnitte neu numerieren. Bei einer Konzept-Version hätte das fatale Folgen: Mit dem Kapitel werden zugleich auch dessen Unterabschnitte gelöscht - das Kapitel ist ja ein umfassender Frame.

Zur Erinnerung: Mit *Rücknahme* im Menü *Frames* läßt sich die letzte Operation rückgängig machen - zum Beispiel eine unabsichtliche Löschung.

Herwig Moritz muß den Unterabschnitt verlagern und nur darauf achten, daß er sich das richtige Ziel aussucht:

1. Er bringt den Cursor in die Zeile »4.1 Die Stellung im Weltmarkt«.
2. Mit **[F7]** (*Verlagern*) leitet er die Verlagerung ein.
3. Er bringt den Cursor in die Zeile »3 Der Geschäftsverlauf«.
4. Mit **[RETURN]** löst er die Verlagerung aus.

Damit ist dieser Abschnitt eine Gliederungsebene höher gerückt. Hätte Herwig Moritz als Ziel den Cursor auf einen der Unterabschnitte des Punktes 3 gebracht, zum Beispiel in die Zeile »3.2 Die Entwicklung von 1970 bis 1986«, wäre der Abschnitt nur von einem ins andere Kapitel verschoben worden.

Etwas komplizierter wird es, einen Abschnitt eine Ebene tiefer zu rücken. Herwig Moritz probiert's einmal übungshalber, wenn's für sein Konzept auch keinen Sinn ergibt:

1. Mit **[CTRL][NUM+]** fügt er dem Abschnitt »3.3 Zusammenfassung« einen Unterabschnitt an.
2. Mit **[F7]** (*Verlagern*) fügt er dort den Abschnitt »5.2 Die Stellung im Binnenmarkt« ein - er wird zum Gliederungspunkt 3.3.2.
3. Er löscht den Gliederungspunkt 3.3.1 wieder, der für ihn nur ein Hilfskonstrukt war.

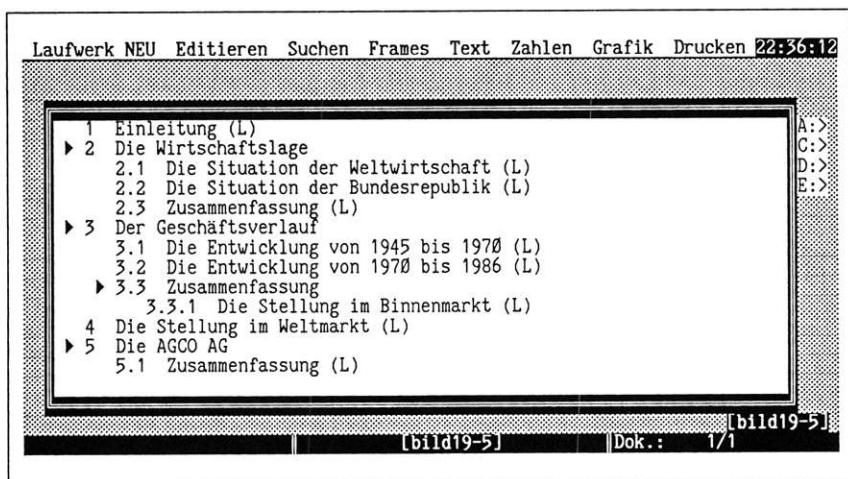


Bild 19.5: Wie man Gliederungspunkte eine Ebene höher oder tiefer rückt

Bisher hat Herwig Moritz getan, was man beim Entwurf eines Konzeptes gemeinhin tut: Er hat sich nur Gedanken um seine Gliederung gemacht. Und keine darüber, mit welchen Frame-Typen er operiert.

Das indessen ist gar nicht unerheblich. Der Frame-Typ ist in Klammern jeweils hinter den Gliederungspunkten angegeben (was sich mit *Typbezeichnungen anfügen* im Menü *Frames* ab- und anschalten läßt).

Merke:

- L leerer Frame
- T Text-Frame
- D Datenbank-Frame
- K (Tabellen-) Kalkulations-Frame
- G Grafik-Frame

Meist ist L angegeben - das steht für einen leeren Frame. Aber wie er so auf seinen Bildschirm blickt: Nicht immer kann er ein L entdecken. Es taucht nur auf, wenn dem Abschnitt keine weitere Ebene darunter folgt.

Um das zu verstehen, vergegenwärtigt sich Herwig Moritz, daß er es bei einer Konzept-Version mit verschachtelten Frames zu tun hat - wenn er mit **[F10]** (*Zeigen*) von der Konzept- in die Frame-Version umschaltet, hat er das auch bildlich vor Augen.

Merke: Ein Frame kann entweder einen anderen Frame enthalten oder Text, nicht beides zusammen.

Nur auf der jeweils untersten Ebene kann er deshalb in dem Frame schreiben:

1. In der Konzept-Version, mit dem Cursor auf einer Überschrift, also auf dem Rahmen eines Sub-Frames, öffnet er den Frame mit **[F9]** (*Zoom*).
2. In der Frame-Version muß er mit **[NUM+]** in den Frame hineingehen (und kann dann natürlich auch hier mit **[F9]** (*Zoom*) auf Bildschirmgröße erweitern).

Zur Erinnerung: Die Konzept-Version zeigt nur die Frame-Namen, die Frame-Version hingegen die Sub-Frames samt Rahmen.

Die Typbezeichnung ändert sich beidesmal in T - der Hinweis darauf, daß aus dem leeren Frame nunmehr ein Text-Frame geworden und es somit unmöglich ist, hier noch eine Gliederungsebene tiefer zu gehen.

Nun möchte Herwig Moritz zwischen Kapitelüberschrift und Unterabschnitt aber auch Text haben, gewissermaßen einleitende Worte zu jedem Kapitel - zwischen »Geschäftsverlauf« und »Die Entwicklung von 1945 bis 1980« beispielshalber.

Einen neuen Frame dort einzuschieben, ist nicht weiter schwer, wenngleich etwas umständlich (weil er ja bekanntlich erst verlagert werden muß). Weil dort kein weiterer

Unterabschnitt folgt, kann er auch mit Text gefüllt werden. Jedoch: Framework hat brav die Numerierung geändert - zum Verdruß für Herwig Moritz, für den dies kein Unterabschnitt gleich den anderen sein soll.

Herwig Moritz besinnt sich darauf, daß man die Numerierung im Menü *Frames* ja auch abschalten kann.

1. Er bringt den Cursor auf den leeren Frame.
2. Er schaltet die Numerierung ab.

Und was macht Framework zu seiner großen Freude? Die Nummer wird nicht bloß versteckt, der Abschnitt wird gleichsam aus dem gesamten Numerierungssystem herausgelöst, die Numerierung beginnt erst im folgenden Abschnitt. Und fügt er nun dem unnummerierten Abschnitt (auf derselben Ebene) einen weiteren Frame an, fällt der ebenfalls in das Numerierungsloch.

Mit einem kleinen Trick schafft Herwig Moritz also doch noch, was eigentlich unmöglich ist. Er wird sich das gut merken müssen, denn dessen wird er sich noch oft bedienen. Zum Beispiel, wenn er in Kapitel 22 in sein Konzept eine Tabelle und eine Grafik einfügt.

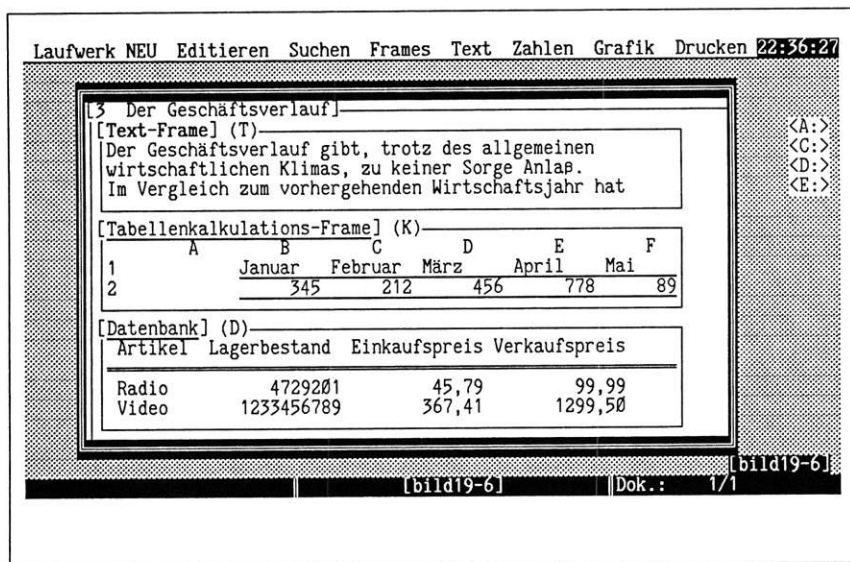


Bild 19.6: Verschiedene Frame-Typen in einem Konzept

Weiß Herwig Moritz bereits gewiß, wann er wo eine Tabelle oder eine Datenbank in seinem Konzept haben will, bestimmt er gleich den richtigen Frame-Typ. Er hat dann zum Beispiel eine leere Datenbank, die er nur noch zu füllen braucht. Auch hinterher ist es freilich kein größeres Problem: Er kann ja auch einen leeren Text-Frame erstellen und in diesen eine Datenbank schachteln.

Tip: Wenn Ihnen vornherein klar ist, daß Sie Frame-Typen mischen, zum Beispiel eine Tabelle in einen Text einfügen wollen, dann definieren Sie in der Konzept-Version gleich leere Frames. Überflüssige sind hinterher leicht gelöscht.

Das Konzept ist das Gerüst eines Textes: Es sind die Kapitelüberschriften, und die möchte Herwig Moritz zum einen in ihrer Erscheinung abheben vom normalen Text, zum andern will er sie zugleich auch als Inhaltsverzeichnis benutzen.

Die Schriftauszeichnung erfolgt über das Menü *Text*: Fettdruck, Unterstreichen, Schrägschrift. Herwig Moritz kann trennen zwischen der Auszeichnung des normalen Textes und den Kapitelüberschriften, der Frame-Namen also.

Zur Erinnerung: Fettdruck, Unterstreichen und Schrägschrift können miteinander kombiniert werden, allerdings nicht in einem Arbeitsgang.

Es gibt einen grundsätzlichen Unterschied in der Art, wie man Texte auszeichnet: bevor sie geschrieben werden oder danach.

Wird eine Schriftauszeichnung aktiviert, zum Beispiel *Fettdruck*, wirkt sich das auf allen nachfolgend geschriebenen Text aus: Alles wird fett. Für Schriftauszeichnungen im nachhinein muß der Text zunächst mit **F6** (*Auswahl*) markiert werden. Und andererseits: Steht der Cursor auf dem Rand eines Frames, gelten alle Anweisungen für seinen gesamten Inhalt - auch Schriftauszeichnungen.

Diese Basisregeln und ihre Folgen muß Herwig Moritz beachten. Steht der Cursor auf einer Abschnittsüberschrift, zum Beispiel »Die Situation der Weltwirtschaft«, und Herwig Moritz aktiviert *Unterstreichen*, wirkt sich das auf den gesamten - schon vorhandenen oder noch zu schreibenden - Text dieses Abschnittes aus. Nicht jedoch auf die Überschrift - sie ist ja der Name des Frames und muß deshalb zunächst mit der Leertaste editiert und sodann markiert werden.

Merke: Um das Schriftbild eines Gliederungspunktes, also eines Frame-Namens, zu ändern, muß er zunächst editiert und markiert werden.

Um die Gliederung gleichzeitig auch als Inhaltsverzeichnis verwenden zu können, stellt Framework im Menü *Frames* eine eigene Funktion bereit: *Ergänzen um Seitennummer*. Gleichzeitig werden damit die Leerstellen zwischen Text und Seitennummer mit Punkten gefüllt.

Da Herwig Moritz bislang noch keinen Text geschrieben und noch keinerlei Druckanweisungen gegeben hat, setzt Framework jeweils die Seitennummer 1.

Laufwerk NEU Editieren Suchen Frames Text Zahlen Grafik Drucken 22:36:47		
1	Einleitung (L)	1
▶ 2	Die Wirtschaftslage	2
2.1	Die Situation der Weltwirtschaft (T)	2
2.2	Die Situation der Bundesrepublik (L)	2
2.3	Zusammenfassung (L)	2
▶ 3	Der Geschäftsverlauf	3
	Text-Frame (T)	3
	Tabellenkalkulations-Frame (K)	3
	Datenbank (D)	3
3.1	Die Entwicklung von 1945 bis 1970 (T)	4
3.2	Die Entwicklung von 1970 bis 1986 (L)	5
▶ 3.3	Zusammenfassung	6
3.3.1	Die Stellung im Binnenmarkt (L)	6
4	Die Stellung im Weltmarkt (L)	7
▶ 5	Die AGCO AG	8
5.1	Zusammenfassung (L)	8

[bild19-7] [bild19-7]

[bild19-7] Dok.: 1/1

Bild 19.7: Eine Gliederung als Inhaltsverzeichnis

Herwig Moritz ändert das sogleich. Zwar sind die Seitenzahlen derzeit gleichgültig, doch fügt Herwig Moritz jetzt schon Druckformeln ein, die sich dann auch auf die Seitennumerierung auswirken.

Zum Beispiel möchte Herwig Moritz, daß jedes Kapitel sowie jeder Unterabschnitt eine Gliederungsebene darunter auf einer neuen Seite beginnt. Er muß das eigens anweisen, Framework würde sonst den gesamten Text fortlaufend drucken.

■ Zur Erinnerung: Formeln stehen »hinter« dem Frame.

Druckanweisungen dieser und anderer Art gehören zum großen Bereich der Framework-Formeln. Sie sind den Punktbefehlen ähnlich, die andere Programme verwenden, werden im Gegensatz zu diesen jedoch nicht in den Text eingefügt, sondern in den Formelbereich »hinter« den Frames. Framework trennt strikt zwischen den eigentlichen Daten (Texte, Tabellen) und den Anweisungen, wie mit denen zu verfahren sei.

1. Herwig Moritz bringt den Cursor auf eine Kapitelüberschrift, zum Beispiel »Die Wirtschaftslage« - er bringt also, das ist wichtig zum Verständnis, den Cursor auf den Rand eines (Sub-) Frames.

2. Mit **[F2]** (*Formel editieren*) öffnet er den Formelbereich - der Cursor springt in die Editierzeile.
3. Er gibt ein:
@np

Merke: Framework-Funktionen beginnen stets mit @ oder, alternativ, mit \$. (Der Klammeraffe @ hat den ASCII-Wert 64.)

4. Mit **[RETURN]** schließt er den Formelbereich wieder.

Dieser Befehl weist Framework an, den folgenden Text beim Ausdruck auf einer neuen Seite beginnen zu lassen. Die Folgen sind unmittelbar zu sehen, wenn Herwig Moritz nochmals die Seitennumerierungsfunktion im Menü *Frames* aktiviert. (Er muß sie dazu zunächst ab- und dann wieder anschalten, damit Framework die Änderungen anzeigen kann.)

Tip: Wenn Sie Mühe haben, sich Framework-Druckformeln zu merken, dann versuchen Sie, den gewünschten Befehl ins Englische zu übersetzen: neue Seite - new page - np.

Andere Druckanweisungen kann Herwig Moritz auch gleich einfügen, zum Beispiel Kopf- und Fußzeilen - er geht davon aus, daß sich sein Konzept zwar noch erweitern, doch nicht mehr ändern wird.

Kopf- und Fußzeilen stehen, wie der Name sagt, am Kopf bzw. Fuß einer Seite und können beliebigen Text und/oder Seitenzahlen aufnehmen. Ihr entscheidender Vorteil ist, daß sie nur einmal geschrieben werden müssen und dann auf jeder Seite gedruckt werden.

Für Kopf- und Fußzeilen muß Herwig Moritz wiederum auf den Rand eines Sub-Frames gehen, den Cursor also auf eine Kapitelüberschrift bringen und den Formelbereich öffnen.

Daß in ihm bereits eine andere Formel (die np-Anweisung) vorhanden ist, stört nicht - der Formelspeicher eines Frames kann bis zu 32000 Zeichen aufnehmen, und das ist dann schon ein ausgewachsenes Programm.

Merke: Formeln können aneinandergereiht werden, müssen dann aber durch Kommata getrennt werden.

Der allgemeine Befehl für Kopf- oder Fußzeilen lautet:

@h	Kopfzeile
@f	Fußzeile

Dann folgt, zwingend, ein weiterer Buchstabe, der die Position der Kopf- oder Fußzeile definiert:

@hl	Kopfzeile linksbündig
@hc	Kopfzeile zentriert
@hr	Kopfzeile rechtsbündig
@hp	Kopfzeile Position (in Zeilen von oben)

Analoges gilt für Fußzeilen:

@fl	Fußzeile linksbündig
@fc	Fußzeile zentriert
@fr	Fußzeile rechtsbündig
@fp	Fußzeile Position (in Zeilen von unten)

Ein Befehl indessen lautet bei Kopf- und Fußzeilen komischerweise gleich:

@hf	Kopf-/Fußzeilenbeginn (von welcher Seite an die Kopf/Fußzeilen gedruckt werden)
------------	---

Und schließlich gibt es noch eine Reihe weiterer Differenzierungen, die Kopf- und Fußzeilen gemeinsam verwenden:

@pn	Seitenzahl
#even	Druck nur auf geraden Seiten
#odd	Druck nur auf ungeraden Seiten

Und das alles läßt sich miteinander kombinieren, was ganz schön wild aussehen kann. Herwig Moritz trägt jetzt in den Formelbereich des Frames »Die Wirtschaftslage« ein:

```
@hr("Die Wirtschaftslage"),@hp(5),@hf(2),  
@fc("Seite" & @pn),@fp(2)
```

Tip: Schreiben Sie logisch zusammengehörige Formeln in jeweils eine eigene Zeile. Das erleichtert den Überblick.

Als Kopfzeile

- erscheint der Text »Die Wirtschaftslage«
- rechtsbündig - **@hr**
- auf der fünften Zeile von oben - **@hp(5)**
- und erst von Seite 2 an - **@hf(2)**

Die Fußzeile

- steht zentriert - **@fc**
- in der zweiten Zeile von unten - **@fp(2)**
- und sie enthält den Text »Seite« sowie die aktuelle Seitennummer

Merke: Die Texte von Kopf- und Fußzeilen müssen in der Druckformel von Anführungszeichen umschlossen sein.

Das läßt sich noch weiter ausbauen:

```
@fc("Bericht vom" & @date4(@today) ", Seite" &
@pn,#odd),@fp(2),@np(13)
```

Die Fußzeile

- wird jetzt nur auf Seiten mit ungerader Seitenzahl gedruckt - #odd
- die Seitennumerierung beginnt mit 13 - @np(13)
- der Fußzeilentext nimmt das aktuelle Tagesdatum auf - @date4(@today)

Er lautet also beispielsweise:

Bericht vom 13. Februar 1987, Seite 13

Herwig Moritz muß teuflisch aufpassen, daß er im Fußzeilentext die Anführungszeichen richtig setzt: Alles, was als Text erscheinen soll, braucht die Anführungszeichen - also auch das trennende Komma zwischen Datum und Seitenzahl.

Die Form des Datums kann Herwig Moritz durch verschiedene Anweisungen differenzieren:

@date1	13. Febr. 1987 (Tag, abgekürzter Monatsname, vierstellige Jahreszahl)
@date2	Febr. 1987 (abgekürzter Monatsname, vierstellige Jahreszahl)
@date3	13. Febr. (Tag, abgekürzter Monatsname)
@date4	13. Februar 1987 (Tag, Monatsname, vierstellige Jahreszahl)

■ | **Hinweis:** In Kapitel 33 lernen Sie noch andere Datumsformate kennen.

Das soll an Druckanweisungen vorerst genügen - die Sekretärin Katharina Kröll wird in Kapitel 21 noch mehr mit ihnen zu tun haben, und Herwig Moritz kann vorerst nach Lust und Laune darangehen, sein Konzept zu erweitern und zu verfeinern. Später dann wird er den ganzen Bericht ausdrucken (Kapitel 22).

Na, soll er doch gleich drucken. Wenigstens seine Gliederung, noch ohne dessen Inhalt.

1. Er schaltet mit F10 (Zeigen) wieder zurück in die Konzept-Version, falls er sich in der Frame-Version befand.
2. Er bringt den Cursor auf den Rand des Konzept-Frames.
3. Er schaltet den Drucker ein.
4. Er aktiviert im Sub-Menü *Optionen für Ausdruck* des Menüs *Drucken* die Funktionen *Ohne geschlossene Frames* und *Mit Namen der Frames*.
5. Er startet den Druck mit *Starten* im Menü *Drucken*.

■ | **Fortsetzung:** Kapitel 22 - wie man Tabellen, Grafiken und Text miteinander mischt.

20 Vereinsverwaltung

oder wie der Vorsitzende Lothar Wanner nie mehr eine Ehrung vergißt

Sie erfahren in diesem Kapitel:

- wie man eine Datenbank aufbaut
- wie man eine Datenbank sortiert
- wie man eine Datenbank selektiert
- wie man die Größe eines Feldes verändert
- wie man Felder versteckt
- wie man zwischen Tabellendarstellung und Maskendarstellung schaltet
- wie man in einer Datenbank (oder einer Tabelle oder einem Text) gezielt sucht
- wie man folgende logischen Operatoren verwendet:
 - @and**
 - @or**
 - @not**

Sie benötigen:

- die Übungsdatei DATENBANK.FW.

Das Problem

Als Vorsitzendem des TSV Schanwald obliegen Lothar Wanner vielfältige Aufgaben. Die stete Kontrolle über die säumigen Beitragszahler zum Beispiel. Es geht doch nicht, daß da einer Leistungen in Anspruch nimmt, ohne dafür zu bezahlen.

Genauso wichtig ist ihm indes die Pflege guter Beziehungen zu seinen Mitgliedern. Aber einmal hat er doch tatsächlich das Jubiläum eines Vereinsmitgliedes verschwitzt, eines recht bedeutsamen dazu - ein überaus peinlicher Vorfall. Der ihm nicht wieder unterkommen soll. Und auch mit den runden Geburtstagen ist das immer so eine Sache - ein Vereinsvorsitzender kann schließlich nicht alles im Kopf haben.

Seine Mitgliederliste war bisher ein alter Karteikasten. An sich ja eine praktische Sache, die freilich in Arbeit ausartet, wenn einer aus dem Vereinsvorstand wirklich einmal eine Liste aller Mitglieder braucht - vollständig bitte, und schnell. In solchen Momenten fragt sich Frau Wanner manchmal, ob sie nicht besser seinerzeit den Klaus ...

Auch der umfangreichste Karteikasten ist bei Jubiläen und Geburtstagen nicht sonderlich hilfreich - die Daten schlummern friedlich darin, und einer blättert immer. Lothar Wanner hat einmal sogar schon angefangen, alle Geburtstage in seinen Kalender einzutragen. Was sich ganz gut anließ. Bis er dann eines Tages einer Witwe zum Geburts-

tag ihres Mannes gratulierte ... Hatte er doch vergessen, in aus seinem Geburtstagskalender zu streichen. Daher kommt das Wort von den Karteileichen, dachte er düster.

Lothar Wanner braucht einen elektronischen Karteikasten: eine Datenbank, die beliebig zu erweitern, zu überarbeiten und nach differenzierten Kriterien abzufragen ist.

Die Lösung

Lothar Wanner hat sich die Datenbank DATENBANK.FW angelegt - sie heißt so, weil sie im nächsten Kapitel nochmals verwendet wird, und dort wird dann dieser Dateiname obligatorisch sein. Glücklicherweise enthält DATENBANK.FW schon 34 Datensätze (abzulesen im rechten Teil der Statuszeile), und das lästige Eintippen von Adressen entfällt in diesem Kapitel.

■ **Hinweis:** Über die Prinzipien (und Fallen) der Dateneingabe bei Tabellen und Datenbanken wird in den Kapiteln 21 und 22 gesprochen.

Die Struktur der Datenbank hat sich Lothar Wanner zuvor reiflich überlegt. Der Anfang aller Überlegungen war, nach welchen Kriterien er sie später auszuwerten gedenkt.

Für die Erfassung der Daten ist beispielsweise unerheblich, ob das Geburtsdatum zusammenhängend erfaßt oder auf drei Felder verteilt wird: Tag, Monat, Jahr. Für die Auswertung hingegen kann es von entscheidender Bedeutung werden, wie noch zu sehen sein wird.

■ **Tip:** Planen Sie erst eine Datenbank, ehe Sie Daten eingeben.

Im Gegensatz zu vielen anderen Datenbankprogrammen läßt sich bei Framework die Struktur einer Datenbank leicht ändern. Mit den Funktionen *Zeilen einfügen* und *Spalten einfügen* im Menü *NEU* kann eine bestehende Datenbank erweitert werden. Trotzdem sind einige Vorüberlegungen ratsam, denn Feldinhalte später zusammenzufassen bereitet einige Mühe; im nächsten Kapitel kommen wir darauf zurück.

■ **Merke:** Die erste Zeile der Datenbank nimmt die Feldbezeichnungen auf. Jede Zeile ist ein Datensatz, der sich aus Feldern zusammensetzt.

Lothar Wanner hat sich, zur Vorsicht, dafür entschieden, in möglichst viele Felder aufzuteilen: jeder Information ihr eigenes Feld. Also etwa die Postleitzahl getrennt vom Ort. Das hat den Nachteil, daß die Datenbank aufgebläht wird, unter Umständen weit über den Bildschirm hinausreicht und deshalb seitwärts gerollt werden muß.

■ **Tip:** Entfernen Sie überflüssige Spalten aus einer Datenbank. Dann kommen Sie mit **END** schnell in das letzte Feld.

Das indessen ist in Kauf zu nehmen um des Vorteils willen, daß sehr differenziert abgefragt werden kann.

Was Lothar Wanner nun macht. Denn dazu hat er schließlich die Framework-Mitgliederliste angelegt.

Zunächst bringt er Ordnung in seinen Datenwust und sortiert ihn. Denn erfaßt werden können die Daten in willkürlicher Reihenfolge, Framework sortiert sie nach Wunsch aufsteigend oder absteigend.

Sortiert werden kann nach jedem Feld. Allerdings muß Framework zuvor mitgeteilt werden, welches dies ist.

1. Lothar Wanner bringt den Cursor in die Spalte *Namen* - irgendwohin, es muß nicht die Feldbezeichnung sein.
2. Mit **[CTRL]-S** öffnet er das Menü *Suchen*.
3. Er aktiviert die Funktion *Vorwärts sortieren*.

■ | **Merke:** Beim Sortieren wird der gesamte Datenbestand neu geordnet.

Laufwerk NEU Editieren Suchen Frames Text Zahlen Grafik Drucken 22:38:32						
Name	Vorname	Straße	Nr.	PLZ	Ort	Geburts
Alexis	Steffen	Weidenweg	6	7000	Stuttgart	
Bader	Heinz	Hohenzollernplatz	12	2345	Apfelhausen	
Bansch	Getrud	Cäcilienweg	43	7300	Esslingen	
Beck	Harald	Seestraße	27	7307	Schanbach	
Birkenstock	Peter	Sumpfdotterstraße	11	8000	München	
Croner	Felix	Jägerweg	22	8000	München	
Doll	Albert	Gyslingstraße	25	1234	Rodelstetten	
Durchholz	Karl	Ariboweg	14	7000	Stuttgart	
Fleißner	Renate	Burgstraße	12	7000	Stuttgart	
Frech	Michael	Goetheplatz	32	7000	Stuttgart	
Hauptmann	Carl	Welfenstraße	69	7000	Stuttgart	
Jungmann	Dorothea	Erlbachstraße	42	3619	Hohenberg	
Kleinschmidt	Franz	Mehringeweg	87	7307	Schanbach	
Kolb	Irmgard	Morgensternstraße	54	5983	St. Peter	
Lang	Alex	Harliebstraße	1	8000	München	
Laube	Heinrich	Waldallee	71	3456	Zwillingen	

[bild20-1].Name Sätze: 34/34

AUSWAHL: mit Cursor-Tasten Auswahl ändern; mit RETURN beenden

Bild 20.1: Sortiert nach Namen

Framework reagiert sehr schnell - zumindest bei Datenbanken solch kleinen Umfangs. Und weil das fasziniert und weil zudem jederzeit eine neue Ordnung hergestellt werden

kann, sortiert Lothar Wanner gleich nochmals, und zwar zunächst nach Postleitzahlen und dann nach Geburtsjahren.

Laufwerk NEU Editieren Suchen Frames Text Zahlen Grafik Drucken 22:39:16						
Straße	Nr.	PLZ	Ort	Geburtsjahr	Geburtsmonat	Geburt
Gyslingstraße	25	1234	Rodelstetten	1960		9
Hohenzollernplatz	12	2345	Apfelhausen	1953		2
Waldallee	71	3456	Zwillingen	1948		7
Erlbachstraße	42	3619	Hohenberg	1956		8
Hauptstraße	18	4567	Gundelfingen	1945		12
Siegestor	17	4678	Dornstetten	1954		4
Poitstraße	17	5678	Pfahlingen	1956		5
Morgensternstraße	54	5983	St. Peter	1944		5
Weidenweg	6	7000	Stuttgart	1917		7
Ariboweg	14	7000	Stuttgart	1953		10
Burgstraße	12	7000	Stuttgart	1949		9
Goetheplatz	32	7000	Stuttgart	1948		10
Welfenstraße	69	7000	Stuttgart	1955		4
Burgstraße	31	7000	Stuttgart	1928		3
Burgstraße	2	7000	Stuttgart	1949		10
Laienweg	10	7000	Stuttgart	1948		10

[bild20-2].PLZ Sätze: 34/34

AUSWAHL: mit Cursor-Tasten Auswahl ändern; mit RETURN beenden

Bild 20.2: Sortiert nach Postleitzahlen

Tip: Um die aktuelle Sortierfolge zu erhalten, muß die Datenbank gespeichert werden. Mit **CTRL RETURN** geht das am schnellsten.

Bisher hat Lothar Wanner nur nach einem Kriterium, also nach jeweils einem Feld sortiert. Mehrere Felder lassen sich indessen auch verknüpfen. Zum Beispiel möchte Lothar Wanner eine alphabetische Liste seiner Mitglieder, die aber zugleich auch nach den Postleitzahlen, dem Geburtsjahr und der Dauer der Mitgliedschaft angeordnet ist.

Das läßt sich nicht in einem Arbeitsgang erledigen, es braucht dazu mehrfaches Sortieren. Und die Entscheidung, welches Feld das wichtigste sei. Die Namen, sagt sich Lothar Wanner - und deshalb steht dieses Feld in der Sortierrangfolge an letzter Stelle. Das unwichtigste Kriterium kommt zuerst. Also:

1. Er sortiert zuerst nach *Mitgliedschaft seit*.
2. Dann sortiert er nach *Geburtsjahr*.
3. Dann sortiert er nach *PLZ*.
4. Dann erst sortiert er nach *Namen*.

Merke: Framework sortiert (vorwärts) in dieser Reihenfolge:

- Großbuchstaben
- Kleinbuchstaben
- Zahlen

Über das Menü *Suchen* kann man die *Großschreibung ignorieren*.

Das Sortieren schafft Überblicke unter wechselnden Gesichtspunkten - und damit auch Einblicke. Lothar Wanner hat beispielsweise noch die Altersstruktur seiner Mitglieder so deutlich vor Augen gehabt.

Laufwerk NEU Editieren Suchen Frames Text Zahlen Grafik Drucken 22:40:00							
Nr. PLZ Ort			Geburtsjahr	Geburtsmonat	Geburtstag	Mitglied seit	
	8	8590	Mahnstadt	1900	6	27	1972
	6	7000	Stuttgart	1917	7	27	1974
	31	7000	Stuttgart	1928	3	27	1975
	11	8000	München	1933	1	12	1972
raße	666	8765	Albstadt	1936	12	24	1977
raße	54	5983	St. Peter	1944	5	6	1970
pe	18	4567	Gundelfingen	1945	12	16	1975
	27	7307	Schanbach	1946	1	27	1972
	34	8000	München	1947	4	18	1977
	71	3456	Zwillingen	1948	7	22	1973
	32	7000	Stuttgart	1948	10	5	1976
	10	7000	Stuttgart	1948	10	27	1971
	12	7000	Stuttgart	1949	9	14	1978
	2	7000	Stuttgart	1949	10	31	1976
	9	7000	Stuttgart	1949	8	27	1981
	87	7307	Schanbach	1949	3	2	1978
[bild20-3].Geburtsjahr				Sätze: 34/34			
AUSWAHL: mit Cursor-Tasten Auswahl ändern; mit RETURN beenden							

Bild 20.3: Sortiert nach dem Geburtsjahr

Die viel bedeutsamere Funktion einer Datenbank und ihr Spezifikum indessen ist das Selektieren, mit dem gezielt gleichgeartete Daten aus dem gesamten Bestand herausgezogen werden.

Zum Beispiel, nach einem bedenklichen Blick in die Vereinskasse: Wer hat seinen Beitrag noch nicht bezahlt?

Merke: Beim Selektieren wird aus dem gesamten Datenbestand eine Auswahl getroffen.

Lothar Wanner hat dafür ein eigenes Feld vorgesehen und trägt dort *j* (für Ja) und *n* (für Nein) ein - der Bequemlichkeit halber in Kleinbuchstaben.

Also muß er diejenigen Mitglieder herausfiltern, bei denen das Feld *Beitrag* ein *n* aufweist.

Die Formel dafür, also die Selektier-Anweisung für Framework, auch Filter genannt, muß »hinter« den Datenbank-Frame eingetragen werden.

Zur Erinnerung: Jeder Frame kann im Hintergrund eine beliebig lange Formel aufnehmen.

1. Er bringt den Cursor mit **[NUM-]** auf den Rand des Frames.
2. Er öffnet mit **[F2]** (*Formel editieren*) den Formelspeicher, worauf der Cursor in die Editierzeile springt.
3. Er trägt die Filter-Formel ein:

Beitrag="n"

4. Er löst mit **[RETURN]** die Selektion aus.

Er muß dabei auf die Syntax achten, also auf die Form, in welcher die Filter-Formel geschrieben wird. Framework verlangt die Einhaltung genauer Vorschriften:

- Zunächst muß die Spalte bestimmt werden, nach der selektiert werden soll - »Beitrag« in diesem Fall. Die Schreibweise muß exakt der Feldbezeichnung in der Datenbank entsprechen, wobei es auf Groß- oder Kleinschreibung jedoch nicht ankommt.
- Dann muß die Bedingung angegeben werden, die erfüllt sein soll - Lothar Wanner muß festlegen, was er filtern will. Er bedient sich dazu eines Vergleichsoperators, in diesem Fall des Gleichheitszeichens: Die Bedingung ist dann erfüllt, wenn in der Spalte »Beitrag« der Eintrag gleich »n« ist.

Merke:

=	gleich
>	größer als
<	kleiner als
>=	größer als oder gleich
<=	kleiner als oder gleich
<>	ungleich

Framework hat damit die Datenbank selektiert - alle Sätze herausgezogen, welche die Bedingung erfüllen: alle Mitglieder also, die ihren Beitrag nicht bezahlt haben.

Die Auswahl ist zum aktuellen Inhalt des Frames geworden, alle folgenden Operationen beziehen sich nur auf diesen Teil der Datenbank. Das läßt sich dazu nutzen, die Auswahl-Daten noch weiter einzuengen: sie nochmals zu selektieren.

Wer von den beitrags säumigen Mitgliedern, überlegt Lothar Wanner, wohnt in Stuttgart?

Wiederum gibt er mit **[F2]** (*Formel editieren*) ein neues Selektionskriterium vor (wozu er zunächst die alte Filter-Formel löschen muß, sie bleibt erhalten):

plz=7000

Merke: Texte in Selektions-Bedingungen müssen, im Gegensatz zu Zahlen, von Anführungszeichen umschlossen sein. Sie können auch Leerzeichen enthalten.

Name	Vorname	Straße	Nr.	PLZ	Ort	Geburtsja
Alexis	Steffen	Weidenweg		6 7000	Stuttgart	19
Maier	Ludger	Burgstraße		2 7000	Stuttgart	19
Unkrodt	Heinz	Bahnhofplatz		13 7000	Stuttgart	19

Bild 20.4: Selektion - ein Auszug aus der Datenbank

Damit hat sich der Auszug aus der gesamten Mitgliederliste auf drei Datensätze eingengt - jene, die in Stuttgart wohnen und ihren Beitrag nicht bezahlt haben.

Will Lothar Wanner wieder mit seinem gesamten Datenbestand arbeiten, muß er

- entweder die Formel löschen und mit **[F5]** (*Neuberechnung*) die Datenbank neu berechnen

Tip: **[CTRL]** **[PGDN]** markiert die gesamte Editierzeile.

- oder, einfacher, im Menü *Frames* die Funktion *Öffnen alle* aktivieren, nachdem er zuvor den Cursor auf den Rand des Frames gebracht hat.

Warum *Öffnen alle*? Weil eine Datenbank in Wahrheit aus einer Vielzahl von Sub-Frames besteht. Jedes Feld in jedem Datensatz wird als eigener Frame behandelt, was sichtbar gemacht werden kann, wenn man mit **[F10]** (*Zeigen*) von der Tabellendarstellung auf die Maskendarstellung umschaltet (wobei sich der Cursor in der Datenbank befinden muß, nicht auf dem Rand des Frames).

The screenshot shows a software interface with a menu bar at the top: 'Laufwerk NEU', 'Editieren', 'Suchen', 'Frames', 'Text', 'Zahlen', 'Grafik', 'Drucken', and a clock '22:41:57'. Below the menu is a large rectangular frame containing several smaller input fields, each with a label in brackets. The fields are filled with the following data:

[Vorname]	[Name]	
Ludger	Maier	
[Straße]	[Nr.]	
Burgstraße	2	
[PLZ]	[Ort]	
70000	Stuttgart	
[Geburstag]	[Geburtsmonat]	[Geburtsjahr]
31	10	1949
[Mitglied seit]	[Beitrag]	
1976	n	

At the bottom of the frame, there is a status bar with the text '|| [bild20-5] || Dok.: 1/1'.

Bild 20.5: Ein Datensatz in Maskendarstellung, mit Frame-Name und mit Rahmen

Nunmehr ist ersichtlich, daß ein Datensatz aus einzelnen Frames besteht, die man behandeln kann, wie jeden anderen Frame auch:

1. Mit **[F4]** (*Größe*) ändert Lothar Wanner die Größe.
2. Mit **[F3]** (*Position*) bestimmt Lothar Wanner den Standort innerhalb des umfassenden Frames.

Tip: Lassen sich Sub-Frames nicht verschieben, muß zuerst *Position frei wählbar* im Menü *Frames* aktiviert werden.

3. Über das Menü *Frames* entscheidet Lothar Wanner, ob der Rahmen und/oder der Name der Sub-Frames gezeigt werden soll. Er kann sich dabei einzelne heraus-picken oder mehrere mit **[F6]** (*Auswahl*) markieren.

Merke: Größen- und Positionsveränderungen in der Maskendarstellung haben keine Auswirkung auf die Tabellendarstellung.

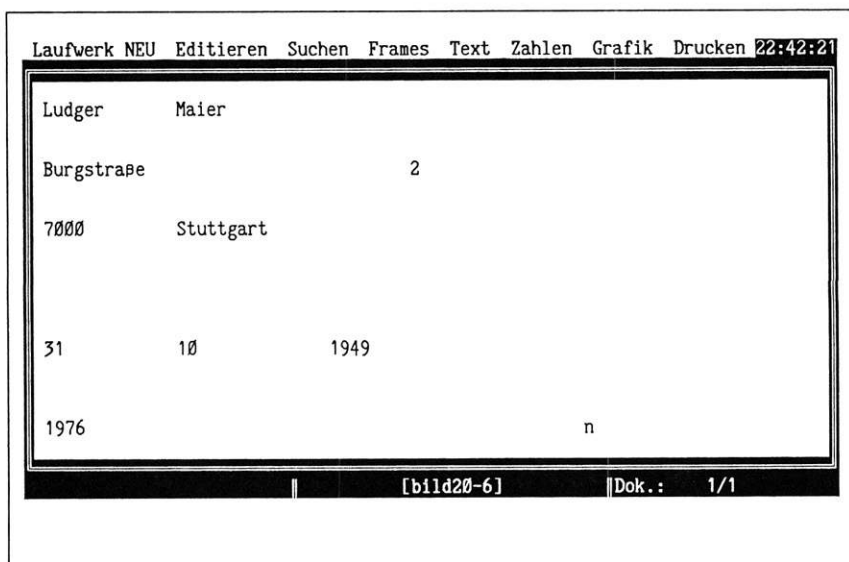


Bild 20.6: Ein Datensatz in Maskendarstellung, ohne Frame-Name und ohne Rahmen

Von Feld zu Feld, von Satz zu Satz bewegt man sich wie gewohnt mit den Cursor-Steuertasten; ungewohnt daran ist allenfalls, daß lediglich ein Datensatz zu sehen ist.

Wozu die Maskendarstellung? Zum Beispiel kann man auf diese Weise das Aussehen von Adreßetiketten gestalten - damit wird die Sekretärin Katharina Kröll im nächsten Kapitel zu tun haben. Zum Beispiel kann Lothar Wanner, um seine Datenbank in ihrer Gesamtheit übersehen zu können, in der Tabellendarstellung die Felder recht schmal (oder sogar unsichtbar) machen und hat in der Masken-Version dann den Blick aufs Ganze:

1. Er bringt den Cursor in ein Feld eines beliebigen Datensatzes (in der Tabellendarstellung).
2. Er drückt **F4** (Größe).
3. Mit den Cursortasten bestimmt er die Größe und kann sie im rechten Teil der Statuszeile ablesen.

Ein verstecktes Feld ist zwar nicht mehr zu sehen, gleichwohl noch vorhanden und kann demzufolge auch Daten beeinhalten - wie ja, grundsätzlich, die Feldgröße nichts über die Größe der aufzunehmenden Daten sagt.

Lothar Wanner ist vorerst noch und auch weiterhin mit dem Selektieren beschäftigt - schließlich will er ausnutzen, daß er nicht mehr einen Karteikasten von Hand durchwühlen muß. Und: daß er mehrere Selektionskriterien auch miteinander verknüpfen kann; er kommt sich vor wie ein Schleppnetzfahrer. Oder wie ein Volkszähler auf Abwegen.

Zum Beispiel: Welche Mitglieder, die in München wohnen, feiern im Jahre 1987 ihr zehnjähriges Vereinsjubiläum? Folgende Filter-Formel sagt es:

@and(Plz=8000,Mitglied seit=1977)

Um Selektionsbedingungen zu koppeln, bedarf es logischer Operatoren. Sie stellen mitunter wirklich harte Anforderungen an die Logik, wie bei den nächsten Beispielen noch zu sehen sein wird - und Sie sollten versuchen, zunächst selber auf die Lösung zu kommen, ehe Sie auf die Formel schauen.

Merke:

@and	und
@or	oder
@not	nicht

Zur Verdeutlichung des Unterschiedes zwischen @and und @or die folgende Bedingung: Lothar Wanner möchte alle Mitglieder, die 1949 geboren wurden und außerdem alle Mitglieder, die ihren Beitrag nicht bezahlt haben.

@or(Geburtsjahr=1949,Beitrag="n")

Spaßeshalber ersetzt er (nachdem er zuvor wieder auf den gesamten Datenbestand umgeschaltet hat) das @or durch @and und beobachtet, was passiert. Und dann ist die Sache klar.

Bei @and müssen alle Bedingungen gleichzeitig zutreffen - sowohl das Geburtsjahr 1949 wie auch die Tatsache, daß der Beitrag nicht bezahlt wurde.

Hingegen definiert @or Bedingungen, die gleichsam unabhängig voneinander existieren: Zum einen ist hier der Topf mit dem Jahrgang 1949, zum anderen der Topf mit den Beitragsschuldnern - in der gefilterten Datenbank werden beide dann zusammengeworfen.

Bei den logischen Operatoren ist Vorsicht angebracht. Die Umgangssprache unterscheidet mit ihrem »und« oftmals nicht zwischen @and und @or - Framework hingegen schon.

Bedingungen können beliebig miteinander verknüpft werden, was die Sache spannend und überaus kompliziert macht. Einige Beispiele:

- Suche alle Mitglieder, die in Stuttgart wohnen, nach 1950 geboren wurden und mehr als zehn Jahre dem Verein angehören.

```
@and(Plz=7000,Geburtsjahr>1950,Mitglied seit<1977)
```

Hier wird von den anderen Vergleichsoperatoren Gebrauch gemacht, wobei man aufpassen und entscheiden muß, was mit eingeschlossen werden soll. In dieser Formel werden nur die Geburtsjahrgänge von 1951 an erfaßt. Soll auch 1950 eingeschlossen werden, muß der Vergleichsoperator $\geq$ verwendet werden - größer als 1950 oder gleich 1950.

- Suche alle Mitglieder, die in Stuttgart wohnen, nach 1950 geboren wurden, mehr als zehn Jahre dem Verein angehören und ihren Beitrag nicht bezahlt haben.

```
@and(Plz=7000,Geburtsjahr>1950,Mitglied  
seit<1977,@not(Beitrag="j"))
```

Die Bedingungen, die durch die logischen Operatoren eingengt werden, müssen stets eindeutig auf den Operator bezogen sein. Deshalb werden sie zwischen Klammern gesetzt, und so häufen sich die Klammern mitunter, und deshalb muß genau überlegt werden, wie man die Bedingung formuliert. In diesem Fall wäre es natürlich einfacher, auf das **@not** zu verzichten und statt dessen, anders formuliert, `Beitrag="n"` anzuhängen.

Tip: Wenn Sie nicht sicher sind, wie verknüpfte Bedingungen zu formulieren sind, selektieren Sie sicherheitshalber schrittweise.

- Suche alle, die im Juli, August, September und Oktober Geburtstag haben.

```
@and(Geburtsmonat>=7,Geburtsmonat<=10)
```

Das ist relativ einfach, lediglich bei den Vergleichsoperatoren muß Lothar Wanner Obacht geben. Meist läßt sich eine Bedingung auch anders formulieren, zum Beispiel:

```
@and(Geburtsmonat>6,Geburtsmonat<11)
```

- Suche alle, die im Juli, August, September und Oktober Geburtstag haben, aber nicht diejenigen, die in Stuttgart wohnen, und auch nicht diejenigen, die 1978 beigetreten sind.

```
@and(Geburtsmonat>=7,Geburtsmonat<=10,@not(@or(Ort=  
"Stuttgart",Mitglied seit=1978)))
```

Ein Gewirr von Klammern, denn alle drei logischen Operatoren verlangen ja einen eindeutigen Bezug.

Nicht so eindeutig ist indessen oftmals ein Feldinhalt zu bestimmen. Framework sucht nur nach dem, was in der Filter-Formel angegeben ist - exakt nach dem. Schon ein (unabsichtliches) Leerzeichen am Ende des Feldeintrages bringt Framework zwar nicht durcheinander, aber Lothar Wanner. Denn diesen Satz ignoriert Framework - das

Leerzeichen war ja in der Formel nicht angegeben. Andererseits: Schließt man zur Vorsicht stets ein Leerzeichen ein, werden alle Datensätze übergangen, die keines haben.

Framework erlaubt deshalb in Filter-Formeln auch die Joker ? und *, wie man sie von den Befehlen auf Betriebssystem-Ebene kennt, mit den gleichen Bedeutungen.

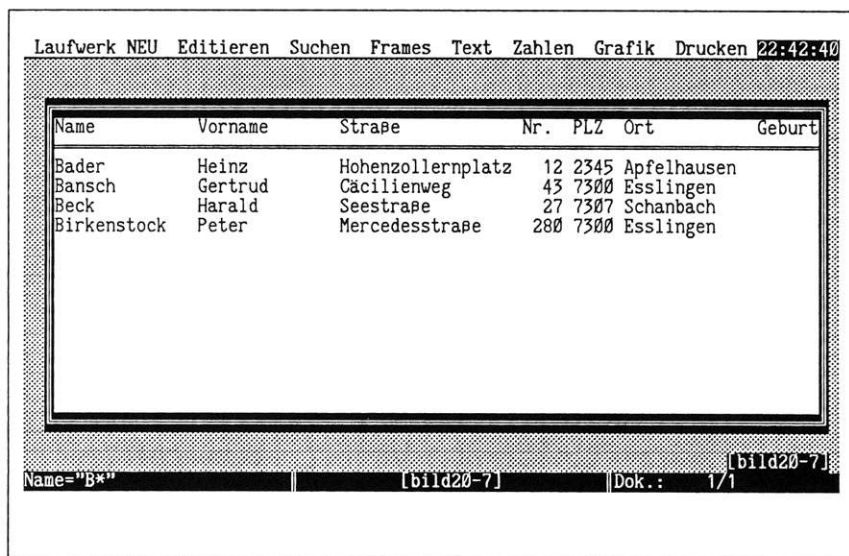
■ **Merke:** ? ersetzt ein Zeichen, * ersetzt mehrere Zeichen.

Damit lassen sich, zum Beispiel, sämtliche Meiers, Mayers, Meyers, Maiers selektieren, wie immer sie sich auch schreiben:

Name="M??er"

Oder alle Straßen, die mit B beginnen:

Straße="B*"



Name	Vorname	Straße	Nr.	PLZ	Ort	Geburt
Bader	Heinz	Hohenzollernplatz	12	2345	Apfelhausen	
Bansch	Gertrud	Cäcilienweg	43	7300	Esslingen	
Beck	Harald	Seestraße	27	7307	Schanbach	
Birkenstock	Peter	Mercedesstraße	280	7300	Esslingen	

Name="B*" [bild20-7] Dok.: 1/1

Bild 20.7: Selektieren mit Joker

■ **Merke:** Die Joker ? und * dürfen nicht am Anfang stehen und haben bei Zahlen keine Wirkung.

Seine solchermaßen selektierten Mitglieder kann Lothar Wanner nunmehr behandeln wie jeden anderen Datensatz auch. Er kann sie

- löschen
- kopieren in einen anderen Frame
- verlagern in einen anderen Frame

Nun geht es Lothar Wanner jedoch nicht immer darum, Gruppen von Mitgliedern, die ein gemeinsames Kennzeichen haben - gleicher Wohnort, gleiches Geburtsjahr - herauszufiltern; er sucht mitunter auch jemanden bestimmten.

Also läßt er suchen:

1. Er öffnet das Menü *Suchen*.
2. Er aktiviert *Suchen nach*. Der Cursor springt in die Editierzeile.
3. Er gibt seinen Suchbegriff ein, zum Beispiel
Birkenstock

Im Gegensatz zur Selektion sind hierbei keine Anführungszeichen erforderlich.

4. Er löst die Suche aus mit `RETURN`.

Und dann durchsucht Framework die Datenbank, bis Herr (oder Frau) Birkenstock gefunden ist. Dann hat Lothar Wanner die Wahl:

- ☐ Framework sucht nach dem nächsten Fundort
- ☐ Framework sucht nach dem vorhergehenden Fundort

Vielleicht sucht Framework auch vergeblich und findet nichts. Eine mögliche Ursache ist, daß Lothar Wanner den zu durchsuchenden Bereich, absichtlich oder unversehens, zu eng begrenzt hat.

Merke: Framework durchsucht nur das Feld, in welchem der Cursor steht, oder die Felder, die mit `F6` (Auswahl) markiert sind. Befindet sich der Cursor auf dem Rand des Frames, wenn die Suche gestartet wird, sind alle Felder einbezogen.

Möglicherweise ist auch ein schlichter Schreibfehler schuld - weiß Lothar Wanner immer so genau, wie die Namen seiner Mitglieder buchstabiert werden? Aber dafür gibt es ja die Joker, die auch bei der Suche eingesetzt werden können.

Er versucht's mal, bringt den Cursor vorsichtshalber auf den Rand des Frames und gibt ein:

B*

Framework findet sehr, sehr viel - jedes Wort, in dem der Buchstabe B vorkommt. Weil nämlich üblicherweise im Menü *Suchen* die Option *Großschreibung ignorieren* aktiv ist. Schaltet Lothar Wanner sie aus, läßt sich die Sache schon besser an. Dann werden nämlich nur Wörter gefunden, die mit dem Großbuchstaben B beginnen, und das sind beträchtlich weniger.

Hinweis: Die gleichen Prinzipien gelten auch für das Suchen in anderen Frametypen (Text, Tabelle) und für das Ersetzen, bei dem eine Zeichenfolge durch eine andere getauscht wird.

Damit kann Lothar Wanner seinen alten Karteikasten auf den Sperrmüll geben. Nie wieder wird er einen Geburtstag vergessen, nie wieder ein Jubiläum. Und die berühmten Karteileichen sind rasch beerdigt. Und wenn er gerade Zeit hat, spielt er Big Brother. So nach und nach hat er nämlich noch einige Felder mehr eingebaut in seine Datenbank und füllt sie mit gewissen Persönlichkeitsmerkmalen: Geschlecht, Familienstand, Religionszugehörigkeit, Erwerbstätigkeit, Schulbildung ... Und dann wollen wir mal sehen, denkt er, ob wir aufgrund all dieser Angaben nicht ein bestimmtes Mitglied herausfiltern oder wenigstens einengen können, ohne dessen Name zu wissen ...

21 Geschäftsbriefe (I)

oder wie die Sekretärin Katharina Kröll ihre Korrespondenz von Framework erledigen läßt

Sie erfahren in diesem Kapitel:

- wie man eine Datenbank erweitert
- wie man eine Datenbank für Briefadressen nutzt
- wie man Serienbriefe schreibt
- wie man Etiketten druckt
- wie man in einer Datenbank Formeln eingibt
- wie man Zahlen als Text-Strings schreibt und wie man Text-Strings in Zahlen verwandelt
- wie man Absätze einrückt
- wie man Seiten formatiert
- wie man u.a. folgende Funktionen verwendet:
 - @integer
 - @pl
 - @sp

Sie benötigen:

- DATENBANK.FW und BRIEF.FW (für die Briefe) von der Beispieldiskette
- ADRESSEN.FW und SCHABLONEN.FW (für die Etiketten) von der Beispieldiskette
- MAILUTIL.FW von der Framework-Hilfsprogrammdiskette

Fortsetzung:

- Makros für Textbausteine benutzen (Kapitel 29)

Das Problem

Katharina Kröll ist Sekretärin bei Troll & Co., Import und Export (Werbeslogan: »Sie brauchen etwas? Wir besorgen es!«), und zu ihrer ungeliebten Hauptbeschäftigung gehört die Korrespondenz. Weil Troll & Co. über einen großen Kunden- und Lieferantenstamm verfügen, ist auch der Briefausstoß gewaltig und entspricht in seinem Ausmaß so ziemlich genau dem Ärger der Katharina Kröll.

Denn natürlich ist das zeitraubend und enervierend, wie jede ihrer Berufskolleginnen bestätigen kann: Adresse heraussuchen, zweimal tippen - einmal auf den Umschlag, dann auf den Brief -, den Brief tippen. Und den womöglich hunderte Mal. Immer der gleiche Brief, nur mit jeweils anderer Adresse und anderen Kundendaten.

Formbrief nennt man das: weil in einen feststehenden Text lediglich individuelle Daten einzutragen sind; und Serienbrief: weil der gleiche (oder ähnliche) Brief an eine Vielzahl von Empfängern geht.

Wie schön wäre es doch, könnte man die stupide Tipparbeit auf ein Minimum reduzieren ...

Die Lösung

Framework hat ein Dienstprogramm integriert, das sich MAILUTIL.FW nennt und solche Serienbriefe im Handumdrehen und selbsttätig erledigt. Erläuterungen zu diesem Programm befinden sich im Frame MAILDOC.FW, den man sich anschauen und ausdrucken lassen kann. Wie man MAILUTIL handhabt, erkundet jetzt unsere Sekretärin.

Tip: Arbeiten Sie mit einem Diskettensystem, erstellen Sie sich für Briefe am besten eine eigene Datendiskette. Sie enthält MAILUTIL.FW sowie die Dateien für die Adressen und den Brieftext.

Katharina Kröll hat sich zunächst einmal eine Datenbank angelegt, welche die Adressen der Kunden und deren spezifische Daten aufnimmt - praktischerweise benutzen wir dazu DATENBANK.FW, die Vereinsmitgliederliste aus dem vorigen Kapitel. Denn natürlich blickt der Vereinsvorsitzende Lothar Wanner Katharina Kröll insgeheim über die Schulter, schließlich muß er ab und an auch Briefe an seine Mitglieder verschicken ...

MAILUTIL benötigt eine Datenbank, die DATENBANK heißt - und wenn sie anders heißt, funktioniert die Sache nicht. Es wäre kein größeres Problem, das Programm so abzuändern, daß jede Datenbank akzeptiert würde. Doch wozu der Aufwand? Benutzt Katharina Kröll eine Datenbank anderen Namens, ändert sie rasch den Frame-Namen um in DATENBANK und hernach wieder zurück.

MAILUTIL braucht zudem den Brief, der gedruckt werden soll. Dessen Name ist beliebig. Er muß sich nur zusammen mit DATENBANK und MAILUTIL auf der Arbeitsfläche befinden.

Jedes Feld der Datenbank kann an beliebiger Stelle des Briefes eingesetzt werden. Als Statthalter dient der Feldname, von spitzen Klammern umschlossen, also etwa so:

```
<vorname> <name>
<straße> <nr.>
<plz> <ort>
```

Beim Druck ersetzt dann MAILUTIL bei jedem Datensatz die Feldnamen durch den Feldinhalt.

Eine ganz einfache Sache, bei der Katharina Kröll schlichtweg nichts falsch machen kann. Das erste Problem jedoch taucht bei der Anrede auf:

Sehr geehrter Herr <name>

Nun gut. Unglücklicherweise arbeitet Katharina Kröll nicht bei einem Herrenclub - zum Adressenstamm von Troll & Co. gehören auch Frauen. Und da wird es dann mit einer genormten Anrede, in der MAILUTIL nur noch den Namen einzusetzen braucht, etwas schwierig. Katharina Kröll behilft sich mit neuen Feldern in ihrer Datenbank - Feldern für die individuelle Anrede.

Hinweis: Wenn Sie das jetzt nicht mitmachen wollen: Die Übungsdatei ADRESSEN enthält eine Datenbank mit den Feldern, die neu hinzukommen.

Anrede	Anrede2	Name	Vorname	Straße
Herrn	r Herr	Kleinschmidt	Franz	Mehringeweg
Frau	Frau	Mayer	Elke	Burgstraße
Herrn	r Herr	Lang	Alex	Hartliebstraße
Herrn	r Herr	Laube	Heinrich	Waldallee
Herrn	r Herr	Durchholz	Karl	Ariboweg
Fräulein	s Fräulein	Jungmann	Dorothea	Erlbachstraße
Frau	Frau	Modlinger	Maria	Memminger Platz
		Alexis	Steffen	Weidenweg
		Schäfer	Gerhard	Hauptstraße
		Lessing	Gotthold	Burgstraße
		Schlote	Reinhold	Breslauer Straße
		Metzger	Hanns-Hermann	Hauptstraße
		Varner	Eduard	Anemonenweg

Bild 21.1: Felder für die individuelle Anrede

Dazu erweitert sie die Datenbank um zwei Spalten:

1. Sie öffnet mit **[CTRL]-L** das Menü *Laufwerk*.
2. Sie aktiviert die Funktion *Spalten einfügen*. Der Cursor springt in die Editierzeile.
3. Sie gibt die Zahl der einzufügenden Spalten an.
4. Sie drückt **[RETURN]**.

|| Zur Erinnerung: Framework fügt Spalten oder Zeilen stets nach dem Cursor ein.

An welcher Stelle der Datenbank die neuen Spalten eingefügt werden, ist MAILUTIL gleichgültig. Die ganze Datenbank wird solange durchsucht, bis die Felder, die in den Variablen des Briefes angegeben sind, gefunden worden sind.

Die beiden neuen Felder nennt Katharina Kröll »Anrede« und »Anrede2«. Das erste Feld dient für den Adressen-Kopf, es wird also »Herr«, »Frau« oder »Fräulein« eingetragen. Das zweite Feld ist für die Anrede gedacht.

Gleich welchen Geschlechts oder Familienstandes der Adressat ist: »Sehr geehrte« braucht es in jedem Falle. Dann differiert es, und das trägt Katharina Kröll in das Feld »Anrede2« ein. Sowohl im Brief wie in der Datenbank muß sie jedoch scharf aufpassen.

Im Brief muß sie den Platzhalter <anrede2> unmittelbar an den Text anschließen, damit es im Druck später nicht so aussieht:

Sehr geehrte r Herr

In der Datenbank auf der anderen Seite ist vor »Frau« ein Leerzeichen unbedingt vonnöten, um folgendes zu vermeiden:

Sehr geehrteFrau

Tip: Halten Sie sich vor Augen, daß an die Stelle der Platzhalter später die Feldinhalte kommen und bedenken Sie die richtigen Wortabstände.

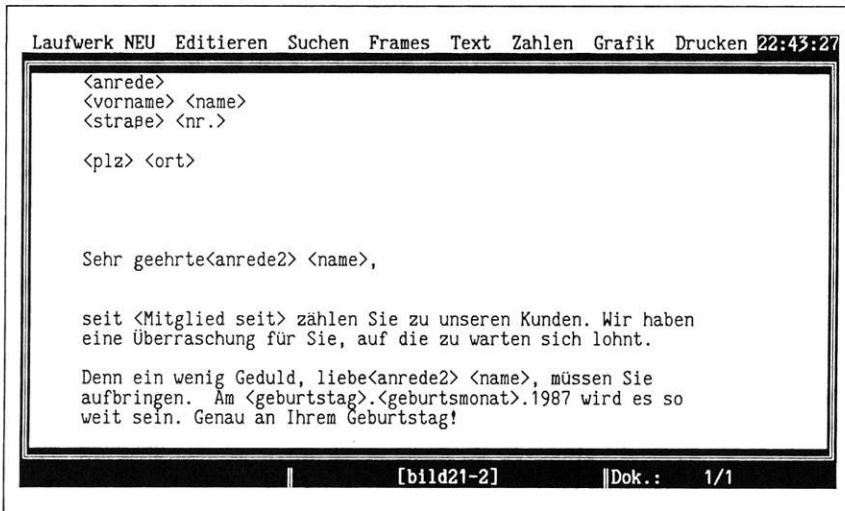


Bild 21.2: Ein Brief mit Platzhaltern

Damit kann die erste Serie von Briefen schon in Druck gehen - die Übungsdatei BRIEF verwendet in ihrem Text noch mehrere Felder von DATENBANK, um deren variablen Einsatz zu demonstrieren.

Katharina Kröll geht in streng reglementierten Schritten vor:

■ **Merke:** MAILUTIL, DATENBANK und der Frame mit dem Brieftext müssen auf der Arbeitsfläche sein.

1. Sie selektiert aus der Datenbank diejenigen Kunden, die einen Brief erhalten sollen.
2. Sie bringt den Cursor auf den Frame MAILUTIL und drückt **[F5]** (*Neuberechnung*); dazu braucht sie MAILUTIL nicht zu öffnen, der Frame kann im Stapel verbleiben.

In der Editierzeile erscheint eine Nachricht, die wegen ihrer Kürze vielleicht mißverständlich sein kann. Sie kümmert sich nicht weiter darum, denn was ihr hier mitgeteilt wird, weiß sie nach dem zweiten Mal ohnehin: welche Tastenkombinationen sie für Serienbriefe und für Etiketten drücken muß.

3. Sie vergewissert sich, daß im Menü *Suchen* die Funktion *Inhalt berücksichtigen* eingeschaltet ist.
4. Sie bringt den Cursor auf den Rand des Frames BRIEF oder in den Frame hinein - das bleibt sich gleich, und so kann sie, gewissermaßen in letzter Minute, auch noch Änderungen im Text vornehmen. Zum Beispiel das Tagesdatum eintragen.
5. Sie drückt **[ALT]-M**.

Sodann wird sie gefragt, ob sie bei den folgenden Operationen zuschauen möchte. Zumindest am Anfang möchte sie, weil sie dabei beobachten kann, was MAILUTIL eigentlich macht.

■ **Tip:** Aber nicht zu lange überlegen! Nach elf Sekunden legt MAILUTIL von allein los!

Grob gesagt folgendes: MAILUTIL erstellt sich aus DATENBANK eine Arbeitsdatenbank, durchsucht den Brief nach dem Namen des ersten Feldes und ersetzt den Platzhalter durch den Inhalt des Feldes. Und so geht es weiter, Feld für Feld, Satz für Satz.

Daraus ist schon ersichtlich, daß Katharina Kröll etliche Tassen Kaffee trinken kann, wenn ihre Datenbank viele Felder umfaßt, von denen indessen nur wenige für den Brief verwendet werden. Denn MAILUTIL durchsucht den Brief ja nach jedem Feldnamen.

In einem solchen Falle ist ein kleiner Kniff zeitsparend, wenn er zunächst auch etwas aufwendig scheint, weil man dazu die Struktur der Datenbank ändern muß.

MAILUTIL hat eine Funktion eingebaut, die wie ein Stoppschild wirkt. Stößt MAILUTIL auf ein Feld ohne Namen, mißachtet es die nachfolgenden Felder, kehrt

um und fährt beim nächsten Datensatz fort. Dort trifft es dann nach einiger Zeit wieder auf das namenlose Feld, mißachtet die folgenden usw.

Merke: Befindet sich in einer Datenbank ein Feld ohne Namen, werden alle Felder vor dem namenlosen Feld berücksichtigt und alle Felder danach ignoriert.

Anfangs hat Katharina Kröll bei der Sache ja einen Schrecken bekommen. Was passiert, wenn ein Feld leer ist? Macht dann MAILUTIL nicht weiter? Macht es schon. Der *Feldinhalt* ist gleichgültig, auf den *Feldnamen* kommt es an, wenn man MAILUTIL davon abhalten will, auch jene Felder zu durchsuchen, die im Brief garantiert nicht verwendet werden.

Deshalb ordnet Katharina Kröll ihre Datenbank ruhigen Gewissens so an, daß alle verwendeten Felder hintereinander kommen (die Reihenfolge ist gleichgültig) und fügt dann eine leere Spalte ein.

Natürlich ist der Feldinhalt ansonsten von einiger Bedeutung - zumal dann, wenn es keinen gibt. Wenn zum Beispiel Firmen angeschrieben werden sollen, mal mit Ansprechpartner, mal ohne und dementsprechend in der Datenbank Felder des öfteren leer sind.

MAILUTIL kümmert's nicht. Stur wird der Platzhalter durch den Feldinhalt ersetzt. Durch nichts also. Und im Adressenfeld des Briefes gähnt ein Loch.

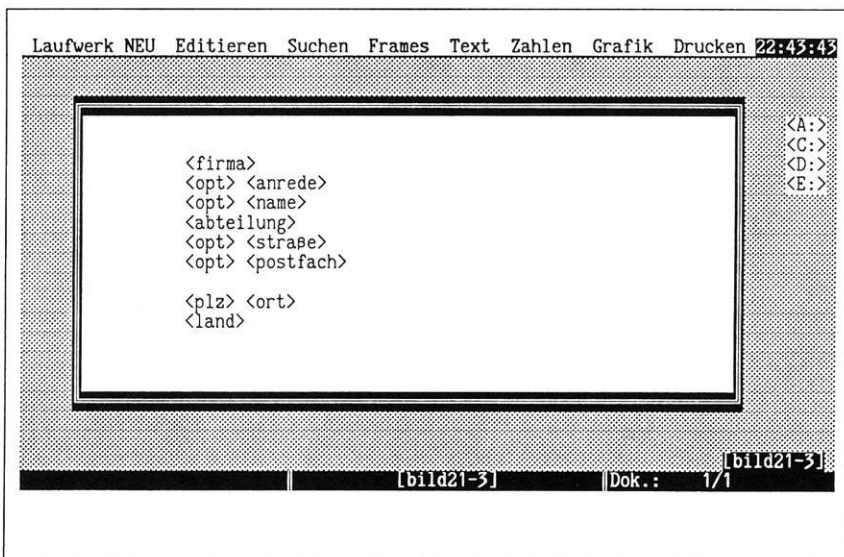


Bild 21.3: Ein Briefkopf mit dem MAILUTIL-Indikator <opt>

Aber auch zur Abhilfe dessen hat MAILUTIL eine eigene Funktion. Jeder Zeile, die nicht jedesmal gefüllt wird, muß der Indikator **<opt>** vorangehen. So etwa (entsprechende Felder in der Datenbank vorausgesetzt):

```
<firma>
<opt> <herrn>
<opt> <vorname> <name>
<opt> <abteilung>
<straße> <nr.>
```

MAILUTIL löscht dann die mit **<opt>** gekennzeichneten Zeilen, sofern sie Leerzeilen sind.

Leider geht das nur mit Zeilen, die ausschließlich aus Platzhaltern bestehen - in der Anschrift etwa. Bei zusätzlichem Text funktioniert es nicht. Und das ärgert Katharina Kröll gewaltig. Sie hat nämlich einige Kunden, denen sich der Chef besonders verbunden fühlt und die er deshalb nicht mit dem formellen »Sehr geehrter Herr ...« anredet wissen will, sondern mit »Lieber Herr ...«.

Da bleibt wohl Katharina Kröll nichts anderes übrig, als in ihre Datenbank im Feld »Anrede2« die gesamte Anrede-Formel einzutragen und ein zusätzliches Feld einzufügen, das den »lieben« Kunden vorbehalten bleibt. Dann kann sie mit dem Indikator **<opt>** differenzieren - weil die gesamte Zeile ein Platzhalter ist. Dafür muß sie in der Datenbank tippen und tippen ... und sie hat gedacht, diese Zeiten wären endlich vorbei.

Bevor die Depression sie überfällt, wartet sie lieber bis Kapitel 29, in dem weitere Arbeitserleichterungen ihrer harren, gerade für solche stereotypen Formeln, die ja auch in normalen, nicht in Serie geschriebenen Briefen häufig vorkommen.

Derweil kümmert sie sich um die Umschläge zu ihren Briefen, die ja auch noch die Anschrift benötigen - nicht immer sind Briefumschläge mit Fenster erwünscht.

Auch für den Druck von Etiketten bringt Framework ein eigenes Programm bereits mit, das in MAILUTIL schon integriert ist. Es gibt dazu auch einen Frame mit Erläuterungen: LABELDOC.FW, in dem indessen nichts anderes steht als in den entsprechenden Abschnitten von MAILDOC.FW auch.

Katharina Kröll holt sich den Frame ADRESSEN.FW von der Beispieldiskette auf die Arbeitsfläche - ein verschachtelter Frame mit mehreren Sub-Frames, die teils notwendig sind, teils nicht.

Nicht notwendig für den Etikettendruck ist, daß sich auch die Datenbank innerhalb dieses Frames befindet. Das habe ich nur darum gemacht, daß Fräulein Kröll alles bei- einander hat, wenn sie den Frame lädt. Diese Datenbank entspricht im wesentlichen der bereits bekannten DATENBANK.FW - diese könnte genauso gut für den Etikettendruck verwendet werden.

Ein wenig anders ist die Datenbank des Frames ADRESSEN schon. Einige Felder wurden eliminiert, die zwar für den Vereinsvorsitzenden Lothar Wanner und seine Mit-

gliederliste notwendig waren, jedoch für Etiketten ohne Belang sind, und überdies wurde sie um etliche Datensätze gekürzt - um auf der Beispieldiskette Bytes zu sparen und beim Probeausdruck Papier.

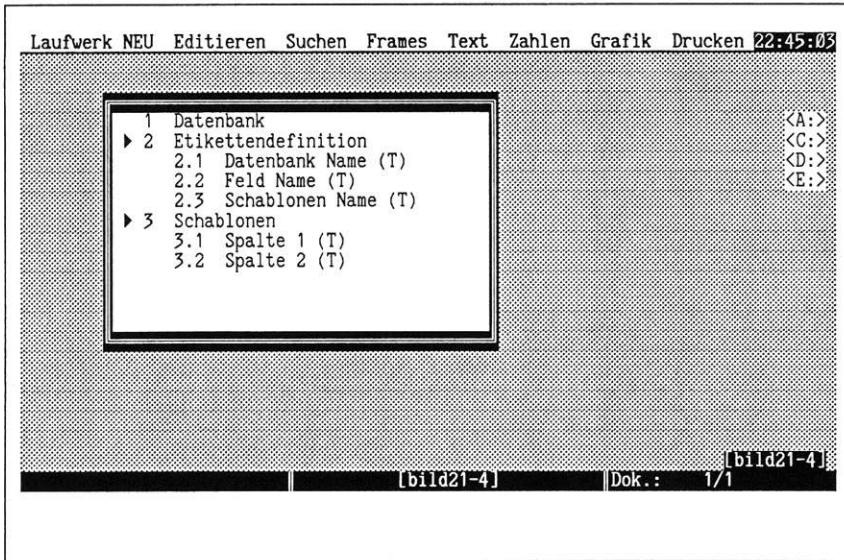


Bild 21.4: *Der Frame ADRESSEN in der Konzept-Version*

Nochmals: Jede andere Datenbank mit Adressen kann genausogut verwendet werden - sie muß sich nur auf der Arbeitsfläche befinden und DATENBANK heißen. Und sie *muß* ein spezielles Feld für den Etikettendruck enthalten, das praktischerweise »Etiketten« heißt (weil MAILUTIL diesen Feldnamen nun mal als feste Bezeichnung einprogrammiert hat, wie den Frame-Namen DATENBANK), und sie *sollte* ein Feld mit dem Namen »Komplettnamen« enthalten, wenn zu befürchten ist, daß in den Adreßzeilen der Etiketten auch leere Felder verwendet werden (bei Firmenanschriften etwa).

Dieses Feld enthält eine Formel, die das gleiche bewirkt wie der Indikator <opt> in Briefanschriften: Leerzeilen als Folge von leeren Feldern werden gelöscht.

Um die Formel sichtbar zu machen, muß sie von Katharina Kröll editiert werden.

1. Sie bringt den Cursor auf den Feldnamen. Daß sich hinter ihm eine Formel verbirgt, ist im linken Teil der Statuszeile zu erkennen - dort taucht sie auf; soweit der Platz reicht.

2. Sie drückt **[F2]** (*Formel editieren*), worauf die Formel in der Editierzeile erscheint. Jetzt kann sie auch verändert werden, was manchmal auch unabsichtlich geschieht, und als noch wenig Erfahrener weiß man hinterher nicht mehr, wie der Schaden zu beheben ist. Allerdings ist die Formel zu lang für die Editierzeile; deshalb:

Tip: Arbeiten Sie nur mit Kopien von Beispieldateien, zumal, wenn diese Formeln enthalten. Sie können dann im Falle eines Falles auf die Originale zurückgreifen.

3. Sie drückt **[F9]** (*Zoom*) und hat damit für die Formel den gesamten Bildschirm zur Verfügung.

Die Formel lautet:

```
Komplettname := @MailUtil.Begrenzung(Anrede," ") &
@MailUtil.Begrenzung(Vorname," ") & Name
```

Was das genau bedeutet, weiß Katharina Kröll nicht so recht (und wird es erst verstehen lernen, wenn sie sich in den späteren Kapiteln in die Framework-Programmiersprache FRED eingearbeitet hat). Deshalb nur soviel zur Erläuterung: Diese Formel bezieht sich auf ein in MAILUTIL hinterlegtes Programm namens BEGRENZUNG, welches die Leerzeilen entfernt, wenn die Felder »Anrede« und »Vorname« leer sind.

Laufwerk NEU Editieren Suchen Frames Text Zahlen Grafik Drucken 22:45:52				
Komplettname	Anrede	Anrede2	Name	Vorname
Herrn Franz Kleinschmidt	Herrn	r Herr	Kleinschmidt	Franz
Frau Elke Mayer	Frau	Frau	Mayer	Elke
	Herrn	r Herr	Lang	Alex
	Herrn	r Herr	Laube	Heinrich
	Herrn	r Herr	Durchholz	Karl
	Fräulein	s Fräulein	Jungmann	Dorothea
	Frau	Frau	Modlinger	Maria
	Herrn	r Herr	Alexis	Steffen
	Herrn	r Herr	Schäfer	Gerhard
	Herrn	r Herr	Lessing	Gotthold
	Herrn	r Herr	Schlote	Reinhold
	Herrn	r Herr	Metzger	Hanns-Herma
	Herrn	r Herr	Varner	Eduard
	Herrn	r Herr	Beck	Harald
	Herrn	r Herr	Groner	Felix

[bild21-5] [Dok.: 1/1]

Bild 21.5: Die Datenbank für den Etikettendruck mit dem Feld »Komplettname«

Auch ohne nähere Programmierkenntnisse ist die Formel indessen leicht abzuändern, wenn auf andere Felder Bezug genommen werden soll. Gesetzt der Fall, eine Adreß-Datenbank enthielte auch das - nicht immer besetzte - Feld »Firma«, müßte die Formel heißen:

```
Komplettname := @MailUtil.Begrenzung(Firma," ") & @MailUtil.Begrenzung(Anrede," ") & @MailUtil.Begrenzung(Vorname," ") & Name
```

Was die Formel in ihrer Datenbank bewirkt, bemerkt Katharina Kröll sogleich, wenn sie einen neuen Datensatz ausfüllt (und wenn sich MAILUTIL auf der Arbeitsfläche befindet, sonst bekommt sie eine Fehlermeldung): Füllt sie die Felder »Anrede«, »Vorname« und »Name« aus, wird deren Inhalt automatisch in »Komplettname« übernommen - wobei dann wiederum »Komplettname« auch als Platzhalter in einer Briefanschrift verwendet werden kann.

Die Formel ist »hinter« dem Feldnamen eingetragen, damit sie sich auf sämtliche Datensätze dieser Spalte auswirkt und wird deshalb auch eine globale Formel genannt, im Gegensatz zu einer lokalen Formel »hinter« einem Feld, die Anweisungen nur für dieses eine Feld in diesem einen Datensatz gibt. Von solchen Formeln in Datenbanken wird in Kapitel 24 eingehend die Rede sein, wenn der Schriftsteller Jonathan Kraft an seiner Überschuß-Berechnung arbeitet.

Merke: Globale Formeln in einer Datenbank stehen in der ersten Zeile hinter den Feldbezeichnungen und beziehen sich auf alle Sätze.

Lokale Formeln stehen hinter einem Feld und beziehen sich nur auf dieses Feld in diesem Satz.

Weil eine solche globale Formel auf die gesamte Spalte wirkt, kann Katharina Kröll auch die noch leeren »Komplettname«-Felder in der Übungs-DATENBANK mit Leichtigkeit füllen lassen: Sie muß lediglich den Cursor in das Feld bringen und **F5** (*Neuberechnung*) drücken, dann wird dieses Feld, wie die Tastenbezeichnung schon sagt, neu berechnet. Die Formel holt sich ihre Werte aus den definierten Feldern.

Um das nicht einzeln, Feld für Feld, machen zu müssen, kann Katharina Kröll auch einen Bereich markieren:

1. Sie bringt den Cursor in das erste leere Feld der Spalte »Komplettname«.
2. Sie drückt **F6** (*Auswahl*) und markiert damit den Beginn des Bereiches.
3. Sie bringt den Cursor in das letzte leere Feld (exakter formuliert: in den letzten Datensatz), am besten mit **CTRL** **↓** (es macht nichts, wenn dabei auch schon besetzte Felder eingeschlossen werden; sie werden dann einfach nochmals berechnet).
4. Sie müßte jetzt eigentlich **RETURN** drücken, um die Auswahl abzuschließen. Framework-Profis ersparen sich das - es funktioniert auch ohne.
5. Sie drückt **F5** (*Neuberechnung*).

Der gesamte markierte Bereich wird daraufhin abgearbeitet und durchgerechnet, Datensatz für Datensatz.

Bild 21.6: Das Feld »Etiketten« in der Maskendarstellung

Auf die gleiche Weise läßt sich auch die Spalte »Etiketten« füllen - das wichtigste Feld für den Etikettendruck. Hier wird festgelegt, welche Felder der Datenbank in die Anschrift übernommen und wie sie plziert werden.

Zu diesem Zweck ist auch »hinter« dem Feldnamen »Etiketten« eine Formel:

```
Etiketten :=
Anrede & "
" & Vorname & " " & Name & "
" & Strasse & " " & @integer(Nr) & "
" & @integer(PLZ) & " " & Ort
```

Dem Feld »Etiketten« werden mit := Werte zugewiesen, nämlich die Werte der im folgenden definierten Felder.

In der Datenbank ist davon lediglich die Anrede zu sehen - die Tabellendarstellung kann nur eine Zeile zeigen, die Etiketten-Anschrift indessen besteht aus mehreren Zeilen. Schaltet Katharina Kröll mit **[F10]** (Zeigen) um in die Maskendarstellung, ist der gesamte Inhalt des Feldes zu sehen.

Die Formel wurde aus der von Framework mitgelieferten Beispiel-Datenbank (in den Frames MAILDOC.FW und LABELDOC.FW) übernommen, allerdings mit einer wesentlichen Änderung. Versucht Katharina Kröll nämlich, diese Framework-Beispiel-Datenbank um einen Datensatz zu ergänzen, wird sie feststellen müssen, daß das Etiketten-Feld Hausnummer und Postleitzahl unterschlägt.

Das Geheimnis ist leicht zu lösen. In der Framework-Beispiel-Datenbank sind die Zahlen als - Text eingegeben. Das ist möglich und manches Mal auch notwendig. Dann etwa, wenn eine Zahl solche Zeichen enthält, die Framework auch zum Rechnen benutzt.

Eine Telefonnummer beispielsweise: 0711/364636. Der Schrägstrich ist die Framework-Anweisung für Division. Wird die Telefonnummer ganz normal als Zahl eingegeben, indem man einfach die Zahl zu schreiben beginnt und hernach mit **RETURN** in das Feld übernimmt - interpretiert Framework sie als den Befehl, 0711 durch 364636 zu teilen. Und das Ergebnis dieser Rechenoperation steht dann im Feld. Statt der Telefonnummer.

Framework unterscheidet also genau zwischen Zahlen als Text (Fachjargon: String) und Zahlen als numerischen Werten. Zahlen als numerische Werte sind dann notwendig und sinnvoll, wenn mit ihnen gerechnet werden soll. Bei Hausnummern, Postleitzahlen oder Telefonnummern ist das nicht der Fall - deshalb gibt man solche Zahlen sicherheitshalber als Text ein. Sonst erreicht der Brief mit der Hausnummer 30/3 nie seinen Empfänger - weil er, 30 geteilt durch 3, an die Hausnummer 10 geht.

Daß eine Zahl als Text, nicht als numerischer Wert behandelt werden soll, signalisiert man Framework mit der Leertaste. Ein Feld füllt Katharina Kröll demnach folgendermaßen:

1. Sie bringt den Cursor in das (leere) Feld.
2. Sie drückt die Leertaste. Framework öffnet daraufhin automatisch die Editierzeile - gerade so, als sei ein Zeichen normal eingegeben worden.
3. Sie schreibt die Zahl und bringt sie mit **RETURN** oder **TAB** in das Feld.

Merke: Zahlen müssen als Text eingegeben werden, also beginnend mit der Leertaste, wenn sie einen der folgenden arithmetischen Operatoren enthalten:

-
+
*
/

Auch bei Texten muß man die Leertasteneingabe anwenden, wenn sie mit bestimmten Zeichen beginnen.

Merke: Bei der Eingabe von Texten muß zuvor die Leertaste gedrückt werden, wenn sie mit einer Zahl oder einem der folgenden Zeichen beginnen:

@

.

+

-

(

#

\$

"

Das ist eine Sache hartnäckigen Übens und steter Konzentration. Selbige hat offensichtlich der Vereinsvorsitzende Lothar Wanner in Kapitel 20 nicht aufgebracht, als er seine Mitglieder-Datenbank erstellt hat, die ja von der Sekretärin Katharina Kröll übernommen wurde. Und die Sache ist bisher auch nicht weiter aufgefallen, weil es, zum Beispiel, keine Schrägstrich-Hausnummern gab. Die originale Formel der Framework-Beispiel-Datenbank jedoch erwartet einen Text, keinen numerischen Wert und ignoriert deshalb einen solchen.

Aus diesem Grund wurde die Etiketten-Formel um eine FRED-Funktion erweitert:

`@integer(PLZ)`

Diese Funktion bewirkt, daß ein numerischer Wert, in diesem Fall die Postleitzahl, in einen String umgewandelt wird.

String ist die Bezeichnung für einen Text - wobei dieser »Text« ja auch, wie wir gesehen haben, aus einer Zahl bestehen kann: Was ein Text ist und was ein numerischer Wert (den wir landläufig eine Zahl nennen), ist eine Sache der Definition. Bei der Eingabe eines Wertes in ein Feld, definiert die vorangehende Leertaste die folgenden Zeichen als Text.

Die Unterscheidung zwischen Text (String) und Zahl ist nicht nur für Datenbanken und Tabellen wichtig, sondern auch für die Framework-Programmierung mit FRED.

Hinweis: Alle FRED-Funktionen sind im Framework-Referenz-Handbuch aufgelistet und erläutert.

Framework verfügt neben `@integer` noch über andere Funktionen, die eine Zahl in einen (Text-)String verwandeln, zum Beispiel `@business` oder `@currency`. Alle Funktionen jedoch bewirken ein jeweils unterschiedliches Zahlenformat:

- `@business` trennt die Tausenderstellen mit einem Punkt - bei einer Postleitzahl ist das nicht erwünscht.
- `@currency` wiederum ergänzt die Zahl um ein Währungssymbol - ebenfalls nicht brauchbar.

- **@integer** jedoch rundet die Zahl lediglich auf und wandelt sie dann in einen String um - was in der Praxis bedeutet, daß sie bloß umgewandelt wird, da die Postleitzahlen und Hausnummern ja keine Nachkommastellen haben.

Weiter im Programm - das im übrigen, um das nochmals klarzustellen, dem in Framework mitgelieferten Etiketten-Druck-Programm entspricht, nur zu Zwecken der Verdeutlichung und der Übung leicht abgewandelt wurde. (Und vom Benutzer so, notfalls variiert, übernommen werden kann.)

Der nächste Punkt dieses verschachtelten Frames heißt »Etikettendefinition« und enthält seinerseits drei Sub-Frames, deren Aufgabe einzig darin besteht, die Namen der Datenbank, des Feldes (mit den zu druckenden Daten) und der Schablone zu definieren.

Die Schablone - Sub-Frame SCHABLONEN - positioniert die Adressen. In dem Beispiel sind zwei Spalten angegeben - Katharina Kröll verwendet zweibahnige Etiketten. Die Beispieldatei SCHABLONEN.FW hingegen hat sich, nur um mal zu zeigen, wie das aussieht, auf sechs Adressenfelder festgelegt - drei nebeneinander, in zwei Reihen untereinander.

Wie macht Katharina Kröll sie sichtbar? Mit **[F10]** (Zeigen) - auf dem Rand des Frames SCHABLONEN.FW.

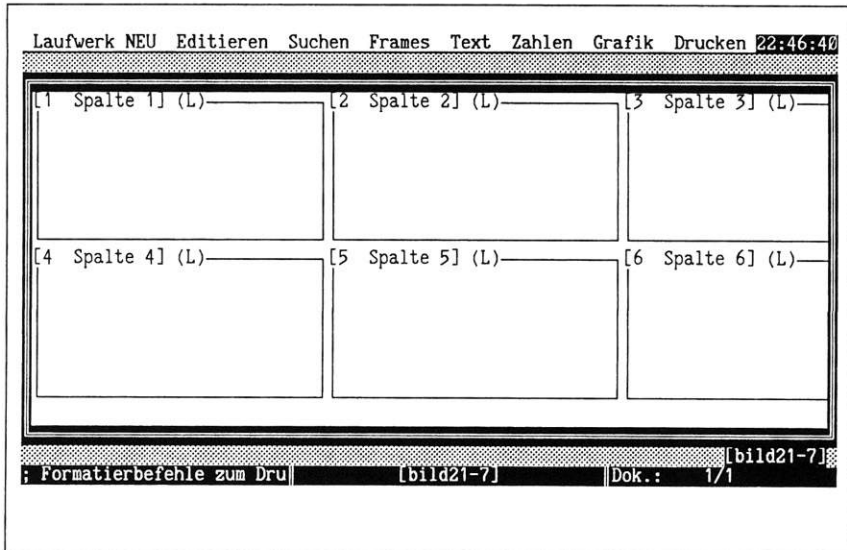


Bild 21.7: Schablonen-Frame für sechs Etiketten

Nun ist schön zu sehen, daß es sich bei einer Schablonendefinition um einen Konzept-Frame handelt: ein verschachtelter Frame, dessen Sub-Frames die gleiche Größe haben. Haben müssen, wie ja leicht einsichtig ist. Sonst gäbe es ein schönes Durcheinander auf den Etiketten.

Merke: Schablonen-Frames müssen die gleiche Größe haben. Drückt man **F4**, kann man in der Statuszeile rechts außen die Größe in Spalten und Zeilen ablesen.

Diese Schablonen-Frames entsprechend der Etikettenbahnen anzuordnen, die verwendet werden, ist ja nicht weiter schwer:

1. Katharina Kröll legt den umfassenden Frame SCHABLONEN als Konzept-Frame an.
2. Sie löscht alle Gliederungspunkte bis auf den ersten - und auch dessen sämtliche Unterabschnitte.
3. Sie bewegt den Cursor dorthin, also auf den Rand des Sub-Frames 1.
4. Mit **F10** (*Zeigen*) schaltet sie um von der Konzept- in die Frame-Version.
5. Mit **F4** (*Größe*) bestimmt sie die Schablonen-Größe.
6. Mit **F8** (*Kopieren*) kopiert sie den Frame, so oft sie ihn benötigt.
7. Mit **F3** (*Position*) ordnet sie die Sub-Frames richtig an.

Tip: Lassen sich Sub-Frames nicht verschieben, müssen Sie im Menü *Frames* zuerst *Position frei wählbar* aktivieren.

Dann müssen aber auch die Formatierbefehle für den Druck entsprechend angepaßt werden. Sie befinden sich hinter dem Sub-Frame »Schablonen« und lauten im ADRESSEN-Beispielframe mit seinen zwei Schablonen-Spalten:

```
@tm(0),
@bm(0),
@po(0),
@ll(132),
@pl(6),
@hp(0),
@fp(0),
@sp(2)
```

Was die Kurzformeln im einzelnen bedeuten, weiß Katharina Kröll nicht auf Anhieb; sie muß erst hektisch im Handbuch blättern. Vielleicht hilft's, wenn sie sich die englischen Übersetzungen merkt:

@tm	oberer Rand (<i>top margin</i>)
@bm	unterer Rand (<i>bottom margin</i>)
@po	linker Rand (<i>page offset</i>)
@ll	Zeilenlänge (<i>line length</i>)
@pl	Seitenlänge (<i>paper length</i>)
@hp	Position der Kopfzeile (<i>head position</i>)

@fp	Position der Fußzeile (<i>foot position</i>)
@sp	Zeilenabstand (<i>spacing</i>)

In diesem besonderen Fall ist die Seitenlänge auf 6 Zeilen mit doppeltem Zeilenabstand eingestellt - das hängt von der Größe der Etiketten ab und dem, was darauf untergebracht werden soll.

Diese Formatierbefehle werden auch auf jeden Text angewendet - auf Katharina Krölls Brief beispielsweise. Framework verwendet Voreinstellungen:

oberer Rand	6
unterer Rand	6
linker Rand	10
Zeilenlänge	65
Seitenlänge	72
Zeilenabstand	1

Zur Erinnerung: Die Framework-Voreinstellungen sind in der Datei CONFIG.FW festgeschrieben. Änderungen werden erst beim nächsten Start von Framework wirksam.

Diese Voreinstellungen gelten für sämtliche Framework-Dateien. Formatierbefehle, auf dem Rand eines Frames als Formel eingegeben wie bei Katharina Krölls Etiketten-Druck, heben diese Voreinstellungen auf - für diesen einzelnen Frame. Aber: Sie gelten für den gesamten Frame. Und dessen Sub-Frames, so vorhanden und sofern sie nicht andere Druckformeln enthalten oder geschlossen sind.

Merke: In der Konzept-Version gelten Druckformeln eines umfassenden Frames auch für dessen Sub-Frames. Anderslautende Druckformeln der Sub-Frames werden ignoriert. Damit auch sie berücksichtigt werden, muß in die Frame-Version umgeschaltet werden.

Auch im Menü *Text* tauchen bestimmte Grundeinstellungen auf (die ebenfalls in CONFIG.FW definiert werden) für

- linken Rand
- rechten Rand
- Einzug Absatz
- Tabulatorsprung

Die wiederum gelten, auf dem Rand des Frames ausgelöst, für den gesamten Frame, innerhalb des Frames nur für einen Absatz - womit sich Grundeinstellungen für einen gesamten Text gezielt Absatz für Absatz abwandeln lassen.

Merke: Druckformeln gelten für den gesamten Frame. Mit den Einstellungen im Menü *Text* lassen sie sich absatzweise ändern.

Was Katharina Kröll nun mit ihrem BRIEF tut:

1. Sie bringt den Cursor in den ersten (Text-)Absatz - gleichgültig, wohin.
2. Sie öffnet das Menü *Text*.
3. Sie aktiviert mit **RETURN** *Einzug Absatz*.
4. Sie definiert meinetwegen einen Einzug von 5 Zeichen.
5. Sie drückt **RETURN**.

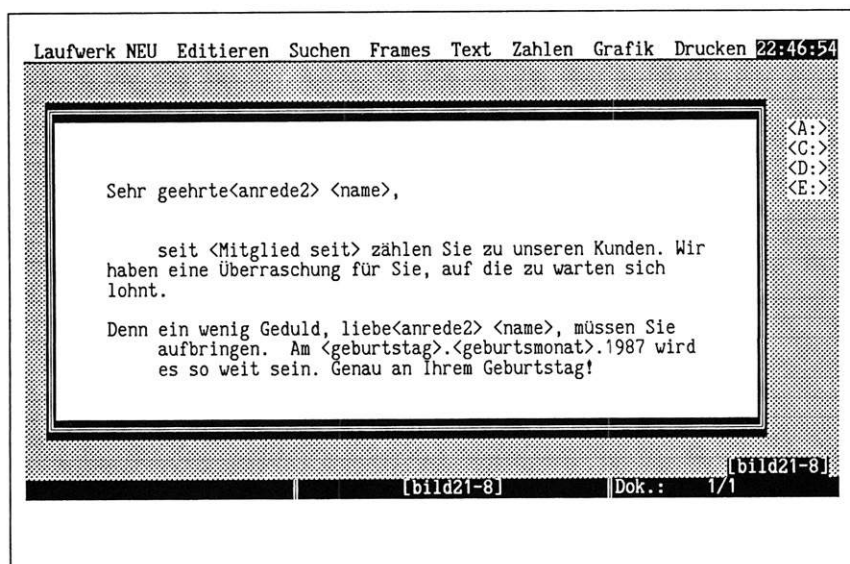


Bild 21.8: Absätze mit positivem und negativem Einzug

Die erste Zeile dieses Absatzes ist damit um 5 Zeichen eingerückt. Auch ein negativer Einzug ist möglich, bei dem die erste Zeile länger ist als die folgenden Zeilen. In dem Fall allerdings sollte zuerst der linke Rand dieses Absatzes um die entsprechende Zeichenzahl nach rechts verschoben werden, sonst hängt die erste Zeile über den linken Textrand hinaus.

Durch entsprechende Änderung der Werte für linken und rechten Rand läßt sich ein Absatz auch auf beiden Seiten einrücken und dann womöglich, ebenfalls über das Menü *Text*, als Blocksatz darstellen. Der allerdings in diesem Brief-Text dann etwas fürchterlich aussieht, weil ja Variablen-Namen enthalten sind, die erst während des Druckes durch die richtigen Werte ersetzt werden, wodurch sich dann auch der Zeilenfall wieder ändert.

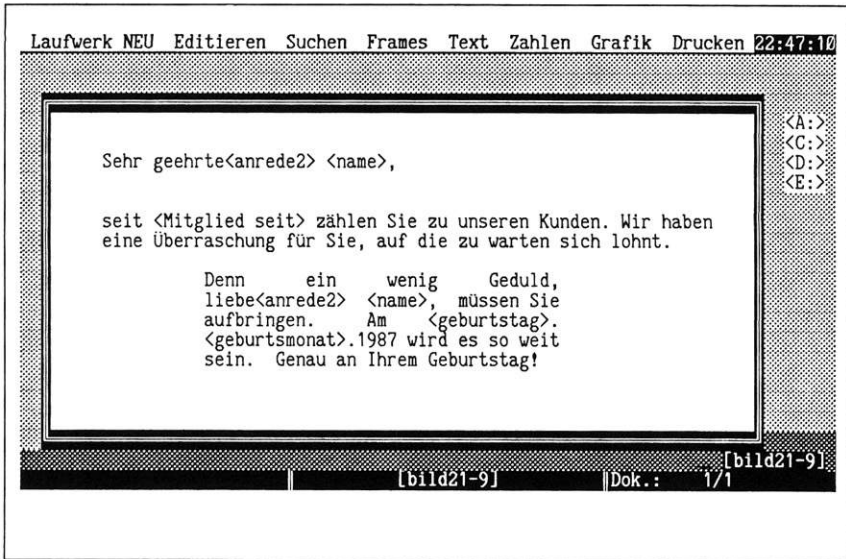


Bild 21.9: Absatz links und rechts eingerückt im Blocksatz

Dann kann es ja an den Ausdruck gehen - zunächst des Briefes:

1. Katharina Kröll selektiert aus der Datenbank diejenigen Adressen, an die der Brief gehen soll - die Auswahl ist vermutlich noch vom Druck des eigentlichen Briefes her erhalten.
2. Sie bringt den Cursor auf den Frame MAILUTIL und aktiviert das Programm mit **[F5]** (*Neuberechnung*); der Frame muß dazu nicht geöffnet werden.

In der Nachrichtenzeile wird sie abermals über die Tastenkombinationen informiert, mit denen der Druck der Briefe oder der Etiketten gestartet wird. **[ESC]** übrigens löscht die Nachricht, ohne deswegen das Programm zu beeinflussen.

Sodann ist, wie beim Briefdruck, die richtige Reihenfolge wichtig. Katharina Kröll muß zunächst den Schablonen-Frame so herrichten, daß nicht auch die Namen der Sub-Frames mit ausgedruckt werden:

3. Sie markiert den übergeordneten Frame und deaktiviert *Namen anzeigen* im Menü *Frames*. Statt dessen könnte sie auch die Sub-Frames mit **[F6]** (*Auswahl*) markieren - aber so ist's bequemer. Das muß sie nur einmal machen: nachdem sie sich ihren Schablonen-Frame hergerichtet hat und so speichert. Dann gilt's für alle Ewigkeit.

■ **Tip:** Als Alternative lassen sich die Namen im Frame erhalten, beim Druck jedoch unterdrücken, wenn im Sub-Menü *Optionen für Ausdruck* des Menüs *Drucken* die Funktion *Mit Namen der Frames* entsprechend eingestellt wird.

4. Sie bringt den Cursor auf den Rand des Frames.
5. Sie drückt **ALT**-E, um den Druck zu starten.

■ **Merke:** Der Cursor muß sich auf oder in dem Frame befinden, wenn MAILUTIL gestartet wird.

Damit hat Katharina Kröll der umfangreichen Korrespondenz ihrer Firma einiges an Schrecken und Zeitaufwand genommen. Allerdings wird sie lernen, das Briefeschreiben noch weiter zu vereinfachen.

■ **Fortsetzung:** Kapitel 29 - Textbausteine als Makros für Brieffloskeln.

22 Geschäftsbericht (I)

oder wie der Sachbearbeiter Herwig Moritz Text, Tabellen, Grafik und eine Datenbank mischt

Sie erfahren in diesem Kapitel:

- wie man eine Tabelle erstellt und fortschreibt
- wie man Rechenformeln eingibt
- wie man relative und absolute Bezüge herstellt
- wie man für Bezüge die Cursormethode benutzt
- wie man die Spaltenbreite ändert
- wie man die erste Spalte/Zeile fixiert
- wie man in Tabellen Schriftauszeichnungen verwendet
- wie man eine Tabelle in eine Grafik umsetzt
- wie man eine Grafik bearbeitet
- wie man eine Grafik mit Hilfe von Formeln zeichnet
- wie man Text, Tabelle und Grafik mischt und zu einem Dokument zusammenbindet
- wie man u.a folgende Funktionen verwendet

@sum

@avg

@drawgraph

Sie benötigen:

- die Übungsdatei UMSATZ.FW
- die Übungsdatei TABELLE.FW
- die Übungsdatei BERICHT.FW

Fortsetzung:

- die Tabelle mit einem Makro auswerten (Kapitel 30)

Das Problem

Der Sachbearbeiter Herwig Moritz ist unterdessen mit seinem Konzept aus Kapitel 19 so weit gediehen, daß er sich nun an die eigentliche Ausarbeitung seines Geschäftsberichtes machen kann. Er will, er muß ja Zahlenmaterial mit einbeziehen - im Grunde ist sein Bericht nur eine Umrahmung und Erläuterung der Zahlen.

Zudem will er nicht nur mit eindrucksvollen Umsatzzahlen glänzen, sondern sie auch optisch ansprechend präsentieren - als Grafik. Und nicht nur mit einer, sondern mit mehreren Grafiken: Er will das Zahlenmaterial unter verschiedenen Gesichtspunkten analysieren.

Herwig Moritz hat mehrere Arbeitsschritte vor sich. Er muß zunächst die Zahlen in einer Tabelle erfassen und dort die nötigen Berechnungen durchführen.

Sodann sind die Ergebnisse in Grafiken umzusetzen, wobei zu entscheiden sein wird, welcher Grafik-Typ die optimale Aussagekraft besitzt. Das ist nicht generell zu beantworten, sondern hängt davon, was gezeigt werden soll.

Schließlich sind Tabelle, Grafik und Text in das Konzept einzubinden.

Die Lösung

Als erstes widmet sich Herwig Moritz der Tabelle - der Übungstabelle UMSATZ.FW. Der besseren Übersichtlichkeit wegen ist sie vorerst nur so breit, daß sie mit **[F9]** (Zoom) vollständig auf dem Bildschirm zu sehen ist.

	A	B	C	D	E	F	G	H
	Januar	Februar	März	April	Summe	Durchschnitt		
1								
2								
3	Umsatz							
4	Radio (St)	387	85					
5	Umsatz							
6	Fernseher	596	745					
7	Umsatz							
8	Video (St)	423	899					
9	Umsatz							
10	Summe Umsatz							
11								
12	Kosten							
13	Lohn	810270	975289					
14	Material	28762	92567					
15	Sonstiges	54370	65968					
16	Summe Kosten							
17								
18	Gewinn (DM)							

Bild 22.1: Die UMSATZ-Tabelle

Schrittweise beginnt Herwig Moritz, die Tabelle zu überarbeiten und zu vervollständigen.

Zunächst sorgt er für ein gefälligeres und übersichtlicheres Aussehen und macht sich die Möglichkeit zunutze, daß auch in einer Tabelle (wie in einer Datenbank) die Schriftauszeichnungen des Menüs *Text* wirksam sind.

1. Er markiert die Rubriken »Umsatz«, »Kosten« sowie die beiden Gewinn-Felder und macht sie fett.

Merke: Mit **[F6]** (*Auswahl*) kann nicht übersprungen, sondern es können nur zusammenhängende Zellen, Spalten oder Zeilen markiert werden.

2. Auf die gleiche Weise setzt er »Summe Umsatz« und »Summe Kosten« kursiv.
3. Die Spaltenbezeichnungen läßt er unterstreichen und macht sich's dazu bequem: Mit **[CTRL][PGUP]** wird die gesamte Zeile oberhalb des Cursors markiert. Allerdings: eben die gesamte. Was zur Folge hat, daß das leere Feld A1 ebenfalls unterstrichen wird.

Laufwerk NEU Editieren Suchen Frames Text Zahlen Grafik Drucken 22:48:26								
	A	B	C	D	E	F	G	H
1		Januar	Februar	März	April	Summe	Durchschnitt	
2								
3	Umsatz							
4	Radio (St)	387	85					
5	Umsatz							
6	Fernseher	596	745					
7	Umsatz							
8	Video (St)	423	899					
9	Umsatz							
10	<u>Summe Umsatz</u>							
11								
12	Kosten							
13	Lohn	810270	975289					
14	Material	28762	92567					
15	Sonstiges	54370	65968					
16	<u>Summe Kosten</u>							
17								
18	Gewinn (DM)							

[bild22-2] [Dok.: 1/1]

Bild 22.2: Die Tabelle mit Schriftauszeichnungen

Die Breite einer Tabellenspalte ist von Framework auf neun Stellen programmiert. Was darüber hinausgeht, ist zwar nicht verloren, doch auf dem Bildschirm nicht sichtbar. Deshalb verändert Herwig Moritz die Spalte A:

Zur Erinnerung: Die Spaltenbreite einer neuen Tabelle läßt sich in CONFIG.FW festlegen.

1. Er markiert ein beliebiges Feld dieser Spalte.
2. Mit **[F4]** (*Größe*) und der Cursortaste **[+]** vergrößert er die Spalte auf etwa 15 Zeichen; in der Statuszeile rechts außen kann er die Veränderung ablesen.
3. Mit **[RETURN]** fixiert er die neue Spaltenbreite.

Noch fehlen die Formeln. Herwig Moritz macht sich zunächst an die Rubrik »Umsatz«.

Ein Radio kostet 99 Mark - zur Vereinfachung unserer Übungsdatei hat sich die Firma AGCO AG entschlossen, nur einen Radiotyp zu verbreiten.

1. Er bringt den Cursor in das Feld B5.
2. Mit **[F2]** (*Formel editieren*) wechselt er den Modus und gibt ein
99*B4

Mit B4 wird der Bezug hergestellt zu dem Feld, aus welchem der Umsatz zu ersehen ist. Und Framework errechnet sofort und automatisch den Umsatz.

Eine simple Formel, ein simpler Bezug: ein relativer Bezug oder, mit einem anderen Begriff, eine relative Adressierung. Sie hat den Vorteil, daß sie in andere Felder kopiert werden kann und sich dort den Koordinaten anpaßt:

3. Mit **[F8]** (*Kopieren*) markiert Herwig Moritz das Feld B5, springt dann nach C5, und mit **[RETURN]** wandelt sich die Formel in
99*C4

Aber, überlegt Herwig Moritz: Wenn sich nun der Preis ändert, muß er in jedem Feld die Formel ebenfalls anpassen, denn sie rechnet ja stets mit dem expliziten Wert 99 - die relative Adressierung bezieht sich nur auf den zweiten Teil der Rechenformel, auf das Feld mit den Absatzzahlen.

|| Hinweis: Falls Sie jetzt nicht mitmachen möchten: Bei der Beispieldatei *TABELLE* handelt es sich um die erweiterte UMSATZ-Tabelle.

Darum ändert er die Tabelle, wie aus Bild 22.3 ersichtlich:

4. Er fügt über das Menü *Neu* eine Spalte und mehrere Zeilen ein, in denen der aktuelle Verkaufspreis der Geräte festgehalten wird.

|| Zur Erinnerung: Eingefügt wird stets nach der aktuellen Cursor-Position.

Die Zahl zwischen den geschweiften Klammern im Menü ist nicht bindend; sie erscheint nach **[RETURN]** in der Editierzeile, wo sie geändert werden kann.

5. Zu den entsprechenden Feldern stellt er einen absoluten Bezug her, etwa für die Januar-Radios mit
\$B\$4*C3

Laufwerk NEU Editieren Suchen Frames Text Zahlen Grafik Drucken 22:49:16								
	A	B	C	D	E	F	G	H
			Januar	Februar	März	April	Summe	Durchsc
1								
2	Umsatz							
3	Radio (St)		387	85				
4	Stückpreis	99						
5	Umsatz		38313	8415				
6	Fernseher (St)		596	745				
7	Stückpreis	599						
8	Umsatz		357004	446255				
9	Video (St)		423	899				
10	Stückpreis	1295						
11	Umsatz		547785	1164205				
12	<u>Summe Umsatz</u>		<u>943102</u>	<u>1618875</u>				
13								
14	Kosten							
15	Lohn		810270	975289				
16	Material		28762	92567				
17	Sonstiges		54370	65968				
18	<u>Summe Kosten</u>		<u>893402</u>	<u>1133824</u>				

[bild22-3] [Dok.: 1/1]

Bild 22.3: Die erweiterte UMSATZ-Tabelle

Ein absoluter Bezug entnimmt seinen Wert stets der mit \$ definierten Zeile und/oder Spalte. In diesem Fall sind sowohl Zeile wie Spalte absolut adressiert. Ändert sich der Wert im Feld B4, wird die gesamte Tabelle neu berechnet. Weil der zweite Parameter einen relativen Bezug hat, kann Herwig Moritz die Formel nach D5 kopieren, wo die alte überschrieben und die neue angepaßt wird.

Merke: Ein absoluter Bezug in einer Formel wird Framework mit dem Zeichen \$ mitgeteilt. Relative und absolute Adressierung können gemischt werden.

Mit anderen Formeln - jeweils mit **[F2]** (*Formel editieren*) eingeleitet, sodann mit **[F8]** (*Kopieren*) in die Spalte D kopiert - werden die weiteren Berechnungen veranlaßt.

Im Feld C12 wird mit

$C5+C8+C11$

die Umsatzsumme ermittelt - eine relative Adressierung, weil sie ja für die weiteren Spalten ebenfalls verwendet werden soll.

Im Feld C18 kann sich Herwig Moritz die Einzelangaben in der Formel ersparen, weil die zu addierenden Posten untereinander stehen, er also einen ganzen Bereich als Bezug vorgeben kann.

Er bedient sich dabei der Cursormethode, was ihm etliche Tipparbeit erspart und die Fehlerquote durch falsche Koordinatenangaben reduziert: Man schaut leicht mal schief und gibt dann die falsche Spalte oder Zeile als Bezugspunkt an.

In einzelnen Schritten:

1. Mit **F2** (*Formel editieren*) schaltet er um in die Editierzeile.
2. Er gibt ein:

@sum(

@sum ist die Framework-Formel zur Summenbildung, in Klammern dahinter folgt der Bereich, der addiert werden soll.

3. **↑** schaltet um auf die Cursormethode.
4. **↑ ↑ ↑** bringt den Cursor an den Anfang des Bereiches für die Summenbildung (das Feld C15).
5. **F6** (*Auswahl*) aktiviert die Blockfunktion.
6. **↓ ↓ ↓** bringt den Cursor zum Ende des Bereiches (das Feld C17).
7. **RETURN** schließt die Auswahl ab.
8. **)** schließt die Formel ab (der Cursor springt bei Eingabe der Klammer automatisch wieder in die Editierzeile). Die Formel lautet jetzt vollständig:

@sum(C5:C17)

9. **RETURN** aktiviert die Formel.

Tip: Verwenden Sie nach Möglichkeit die Cursormethode. Sie ersparen sich Eingabefehler bei der Bestimmung der Bereichskoordinaten.

Für die Berechnung des Gewinnes indessen versagt die Cursormethode. Sie funktioniert nur, wenn sich als Bezüge geschlossene Bereiche ergeben, die in der Formelsprache also mit einem Doppelpunkt getrennt werden.

An Formeln sind nötig:

- in Feld C20 Umsatz minus Kosten:
C12-C18
- in Feld C21 Gewinn geteilt durch Umsatz mal 100:
C20/C12*100

Bei der prozentualen Berechnung des Gewinnes in dieser Form ergibt sich der absolute Wert. Statt dessen kann man ihn über das Menü *Zahlen* auch mit dem Prozentzeichen versehen lassen. In diesem Fall ist in der Formel der Divisor 100 überflüssig.

Weil Herwig Moritz auch eine bilanzierte Übersicht haben will, hat er die Rubriken »Summe« und »Durchschnitt« bereits angelegt. Weil die Tabelle indes übers Jahr hinweg fortgeschrieben werden soll, ergeben sich für die Formeln einige Schwierigkeiten. Für eine Bilanz im Februar sind die Bezugspunkte nur die Spalten C und D, im April aber die Spalten C bis F.

Damit er nicht jeden Monat die Formeln ändern muß, erweitert Herwig Moritz seine Tabelle um acht Spalten bis in den Dezember. Vorsorglich fixiert er auch gleich im Menü *Editieren* die erste Spalte und Zeile. Wenn er jetzt aufs Jahresende zumarschiert, schiebt sich die Tabelle also links über den Bildschirmrand hinaus, kann er gleichwohl seine Zeilenbezeichnungen ablesen und weiß, wo er die Daten eintragen soll.

Zur Erinnerung: Spalten und Zeilen werden über das Menü *NEU* eingefügt - und stets nach der Cursorposition.

Eine Tabelle gleich vollständig auszubauen, ist auch aus einem anderen Grunde sinnvoll. Auf recht einfache Weise kann dann das gesamte Formelgerüst kopiert werden.

1. Mit **F6** (*Auswahl*) markiert Herwig Moritz die Spalte D und kopiert sie nach E.
2. Er löscht mit **DEL** sämtliche Daten in den Feldern, muß dabei aber höllisch aufpassen (und blinzelt deshalb mit dem einen Auge ständig in die Nachrichtenzeile), daß er nicht aus Versehen auch ein Feld löscht, das eine Formel im Hintergrund hat - E5.

Denn der Sinn des Unternehmens ist ja, daß er die Formeln behält und sie dann, wiederum komplett spaltenweise, weiter kopieren kann.

Zwar erhält er spätestens, wenn eine neue Berechnung ansteht, in diesen Feldern die Meldung

#VALUE!

Aus der Sicht von Framework ist das völlig verständlich, denn tatsächlich fehlen ja in den Feldern der Bezüge die Werte.

Merke: Im Menü *Zahlen* und dessen Untermenü *Optionen für Neuberechnung* läßt sich die automatische Berechnung abschalten; dann muß man allerdings für jede Rechnung **F5** (*Neuberechnung*) drücken.

Nunmehr sind auch in den beiden letzten Spalten die Formeln einfach:

- in O3 (*»Summe«*)

@sum(C3:N3)

Das ist ein relativer Bezug, daher in die Zeilen darunter kopierbar.

- in P3 (*»Durchschnitt«*)

@avg(C3:N3)

Dies ist die Framework-Funktion für den Mittelwert, ebenfalls als relativer Bezug und darum ebenfalls bequem kopierbar.

Damit werden sämtliche Spalten ausgewertet, auch die leeren. Weil die jedoch den Wert 0 haben, ändert sich am Ergebnis nichts.

Der Mittelwert indessen wird, mathematisch durchaus logisch, in Exponentialdarstellung ausgegeben. Über das Menü *Zahlen* kann in den Zeilen, wo es wünschenswert ist, umgeschaltet werden, zum Beispiel auf ganze Zahlen. Denn auch für das penibel rechnende Unternehmen AGCO AG sind Viertelradios und Achtfelfernseher beim durchschnittlichen monatlichen Absatz ohne Belang.

Und nun kann Herwig Moritz die neuesten Absatzzahlen eingeben und verfolgen, wie sich Umsatz und Gewinn sofort neu berechnen.

Zum Üben ist solche Dateneingabe ja ganz nett, aber auf die Dauer wird's dann doch langweilig, zumal, wenn man mit Phantasiezahlen arbeitet. Deshalb gibt es die Übungsdatei TABELLE.FW - sie ist schon etwas gefüllter, wenn sie auch nur bis zum Monat April reicht.

Laufwerk NEU Editieren Suchen Frames Text Zahlen Grafik Drucken 22:49:43								
[bild22-4]								
	A	B	C	D	E	F	G	H
			Januar	Februar	März	April	Summe	Durchsc
1								
2	Umsatz							
3	Radio (St)		387	85	129	77	678	17
4	Stückpreis	99						
5	Umsatz		38313	8415	12771	7623	67122	1678
6	Fernseher (St)		596	745	89	56	1486	37
7	Stückpreis	599						
8	Umsatz		357004	446255	53311	33544	890114	22252
9	Video (St)		423	899	325	988	2635	65
10	Stückpreis	1295						
11	Umsatz		547785	1164205	420875	1279460	3412325	85308
12	Summe Umsatz		943102	1618875	486957	1320627	4369561	109239
13								
14	Kosten							
15	Lohn		810270	975289	401190	787987	2974736	74368
16	Material		28762	92567	39876	70485	231690	5792
17	Sonstiges		54370	65968	37345	49763	207446	5186
18	Summe Kosten		893402	1133824	478411	908235	3413872	85346
[bild22-4] Dok.: 1/1								

Bild 22.4: Die ausgefüllte TABELLE

Aber das genügt für den nächsten Arbeitsschritt: das Umwandeln einer Tabelle in eine Grafik.

Bevor das vonstatten gehen kann, muß Herwig Moritz drei Vorentscheidungen treffen:

1. Er muß in der Tabelle die Daten auswählen, die in eine Grafik umgesetzt werden sollen.
2. Er muß festlegen, was die X-Achse und was die Y-Achse werden soll.
3. Er muß den Grafik-Typ bestimmen.

■ **Zur Erinnerung:** Die X-Achse verläuft waagrecht, die Y-Achse senkrecht.

Zunächst die Datenauswahl, zu bewerkstelligen mit **[F6]** (*Auswahl*). Woraus sich auch schon das erste Problem ergibt: Damit können nur benachbarte Zellen markiert, indes- sen keine übersprungen werden. Herwig Moritz will aber, erst einmal, den Absatz von Radio-, Fernseh- und Videoapparaten grafisch darstellen lassen.

Dazu muß er die Tabelle manipulieren: ihre Struktur so verändern, daß die entspre- chenden Daten nebeneinander liegen.

■ **Tip:** Arbeiten Sie zur Sicherheit mit der Kopie einer Tabelle, wenn Sie deren Struktur für eine Grafik ändern. Verlagern Sie die Spalten oder Zeilen auch nicht, sondern kopieren Sie sie.

Nun ist es ja kein Problem, eine ganze Spalte oder Zeile zu verlagern oder zu kopieren. Bloß wird am Zielort nicht eingefügt, sondern erbarmungslos überschrieben - das unterscheidet Tabelle und Datenbank grundsätzlich von Text.

Das bedeutet also: Vor der Umstrukturierung muß Platz geschaffen werden durch Einfügen von Zeilen und/oder Spalten über das Menü *NEU*. Am besten nicht mitten in der Tabelle, sondern an deren Ende. Und gleich großzügig (soweit der Arbeitsspeicher reicht): vielleicht jeweils 10.

Laufwerk NEU Editieren Suchen Frames Text Zahlen Grafik Drucken 22:50:16									
	A	B	C	D	E	F	G		
16	Material		28762	92567	39876	70485	231690	5	>
17	Sonstiges		54370	65968	37345	49763	207446	5	>
18	<u>Summe Kosten</u>		<u>893402</u>	<u>1133824</u>	<u>478411</u>	<u>908235</u>	<u>3413872</u>	85	>
19									>
20	Gewinn (DM)		49700	485051	8546	412392	955689	23	>
21	Gewinn (%)		5,27	29,96	1,75	31,23	21,87		
22									
23									
24									
25	Radio (St)		387	85	129	77	678		
26	Fernseher (St)		596	745	89	56	1486		
27	Video (St)		423	899	325	988	2635		
28									
29									
30									
31									

[bild22-5] [bild22-5]
Dok.: 1/1

Bild 22.5: Ausgewählte Daten für die Grafik

Und nun kann's losgehen, ohne Gefahr, daß die Tabelle zerstört wird:

1. Mit **CTRL** **PGUP** bzw. **CTRL** **PGDN** markiert Herwig Moritz die Zeile 3 mit dem Absatz von Radios in Stück.
2. Mit **F8** (*Kopieren*) kopiert er sie in Zeile 25 (oder so).
3. Mit den Zeilen 6 (Fernseher) und 9 (Video) verfährt er genauso und kopiert sie unter die Radio-Zeile.
4. Mit **F6** (*Auswahl*) markiert er in diesen drei (kopierten) Zeilen gemeinsam die Spalten C bis F - also nur die reinen Absatzzahlen; die Beschriftungen werden von der Grafik automatisch übernommen.

Jetzt weiß Framework, welche Daten für die Grafik herangezogen werden sollen.

Als nächsten Schritt muß Herwig Moritz die X- und Y-Achsen definieren - nicht bloß eine ästhetische Entscheidung, sondern davon bestimmt, was mit der Grafik ausgesagt werden soll.

Die Abbildungen 22.6 und 22.7 zeigen die Unterschiede am Beispiel eines Balkendiagramms. Wird die Zeile als X-Achse genommen, läßt sich besser ablesen, wie die Artikel der Vergnügungsindustrie, Monat für Monat, im Vergleich zueinander abgeschnitten haben. Ist hingegen die Spalte die X-Achse, ergibt sich daraus die Verkaufsentwicklung der einzelnen Geräte übers Jahr hinweg.

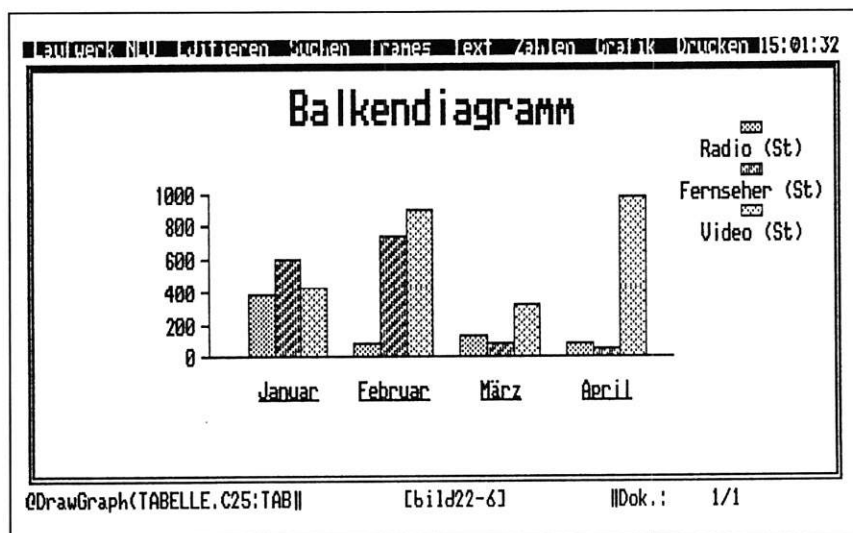


Bild 22.6: Zeile als X-Achse (Balkendiagramm)

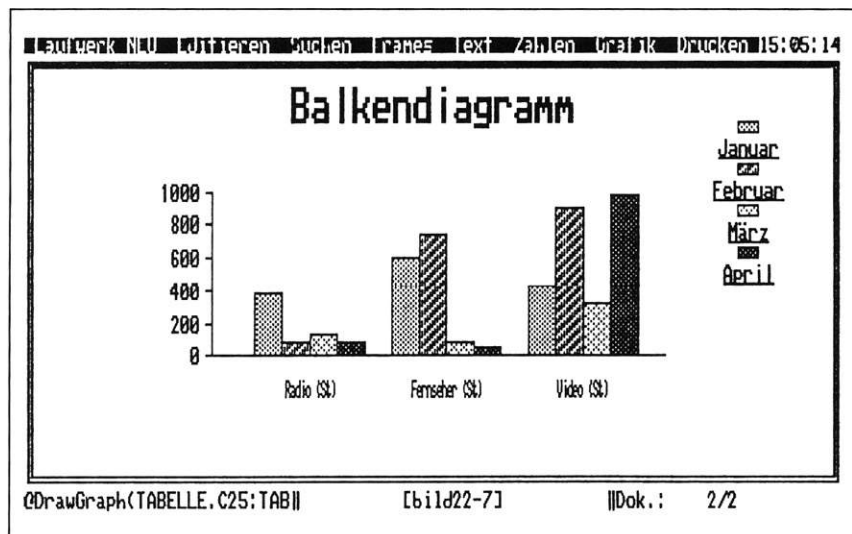


Bild 22.7: Spalte als X-Achse (Balkendiagramm)

Auch die Art der Grafik muß davon abhängig gemacht werden, was mit ihr gezeigt werden soll. Oder deutlicher: wie das, was ausgesagt werden soll, am besten zur Geltung kommt.

Framework hat sechs Grafik-Typen (die alle, mit demselben Zahlenmaterial, auf diesen Seiten abgebildet sind):

- Balken nebeneinander
- Abschnitts-Balken (Turmendiagramm)
- Kreis
- Linie
- Punkte, markiert
- X-Y

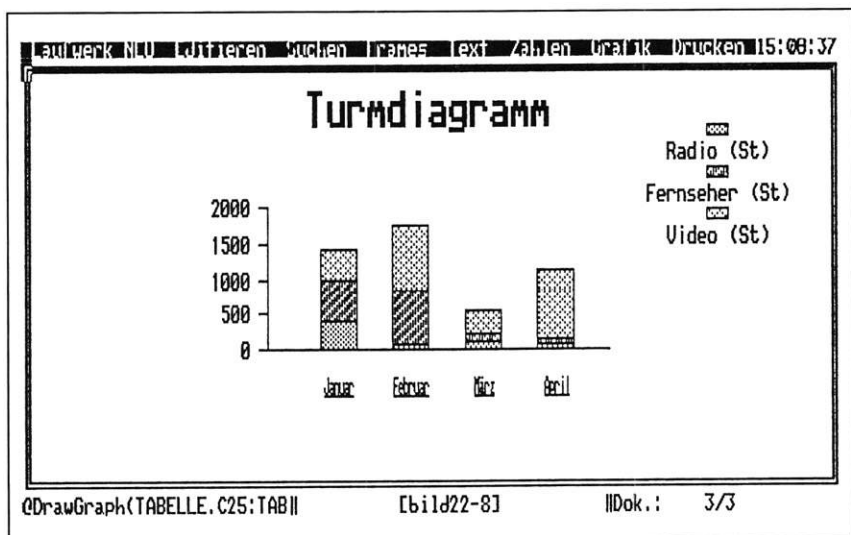


Bild 22.8: Turndiagramm (Abschnitts-Balken), Zeile als X-Achse

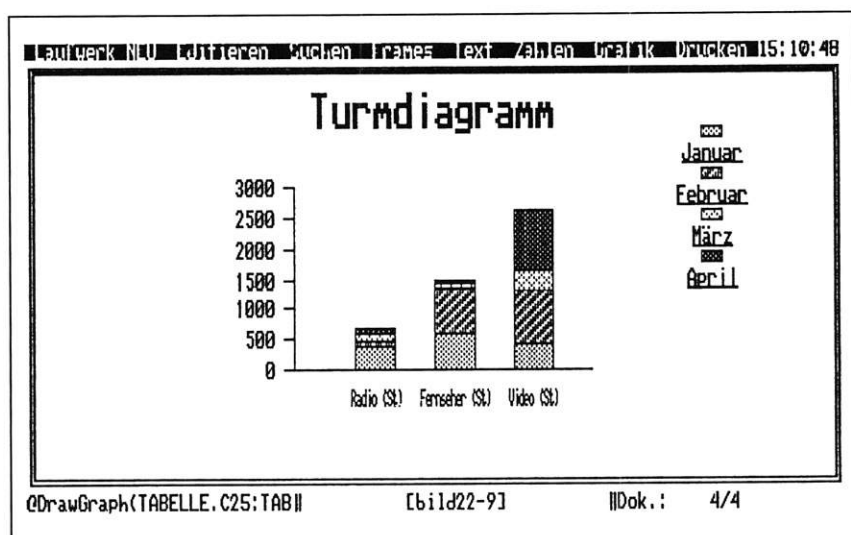


Bild 22.9: Turndiagramm (Abschnitts-Balken), Spalte als Y-Achse

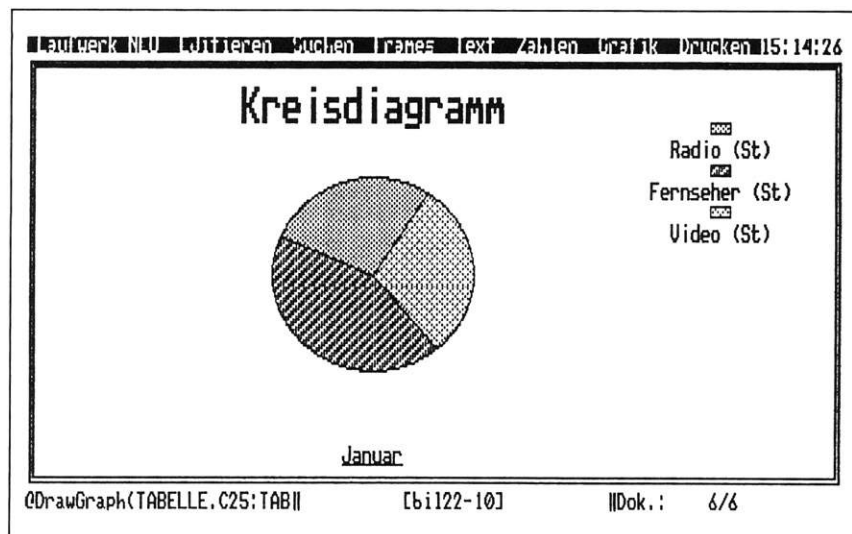


Bild 22.10: Kreisdigramm, Spalte als X-Achse

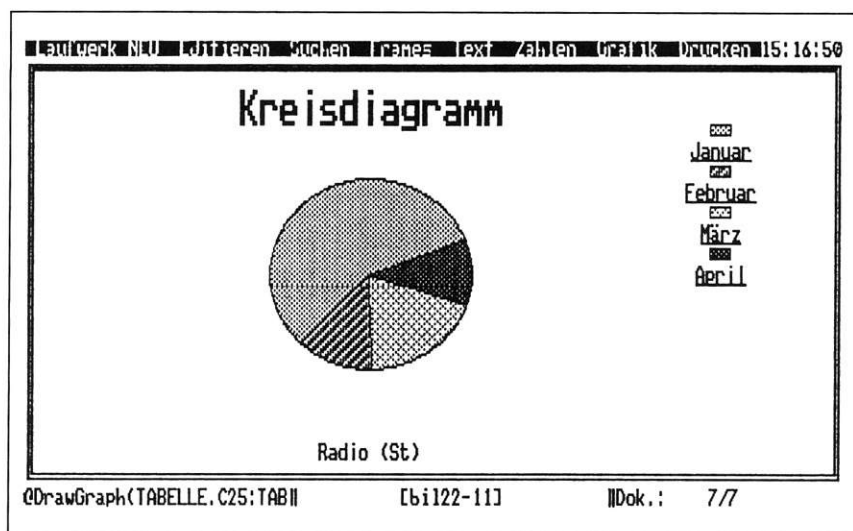


Bild 22.11: Kreisdigramm, Zeile als X-Achse

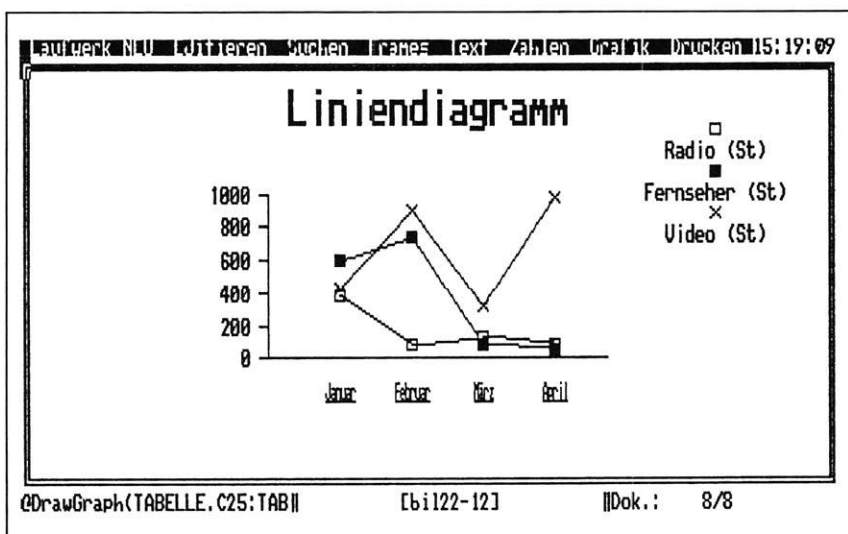


Bild 22.12: Linie, Zeile als X-Achse

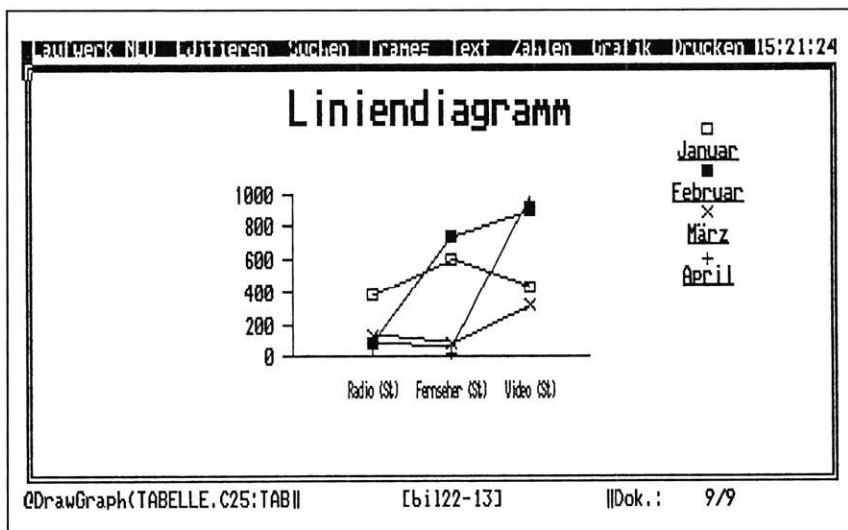


Bild 22.13: Linie, Spalte als X-Achse

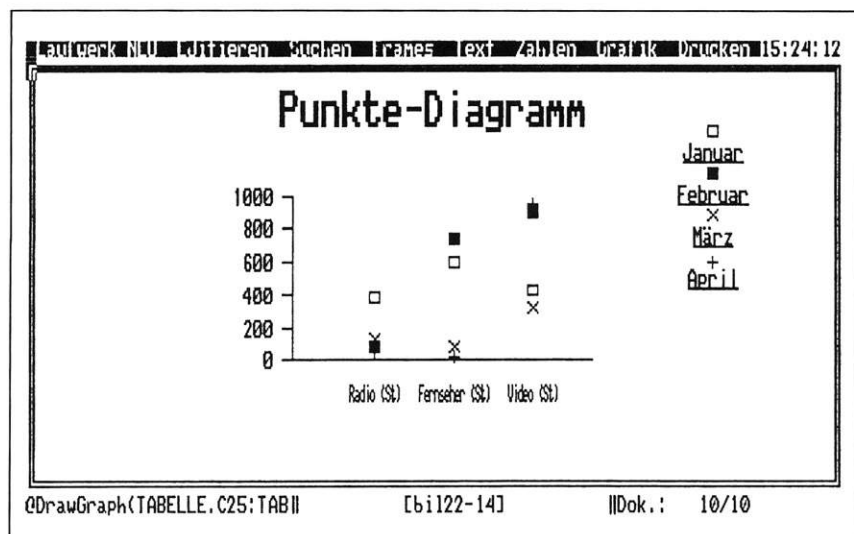


Bild 22.14: Punkte, Spalte als X-Achse

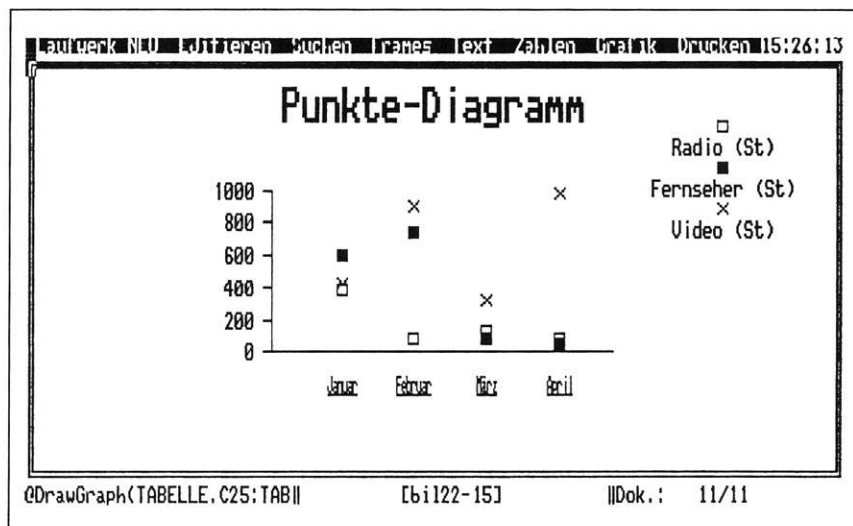


Bild 22.15: Punkte, Zeile als X-Achse

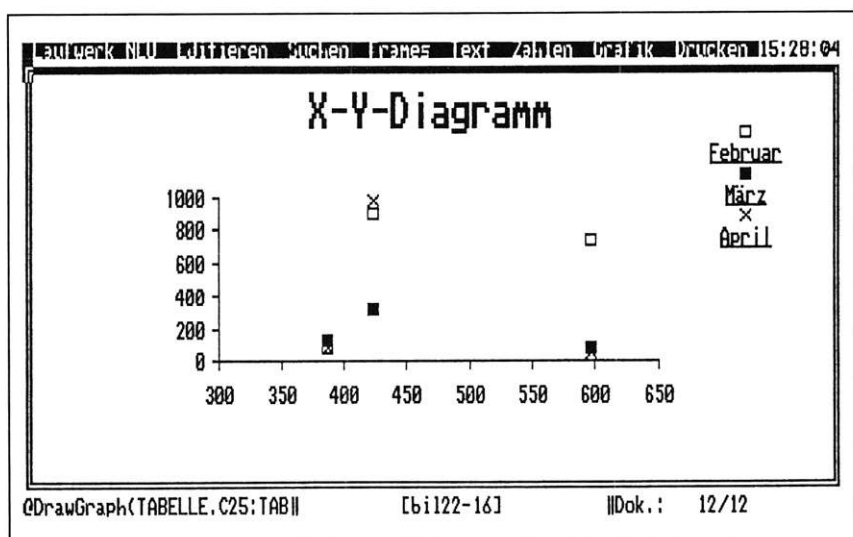


Bild 22.16: X-Y, Spalte als X-Achse

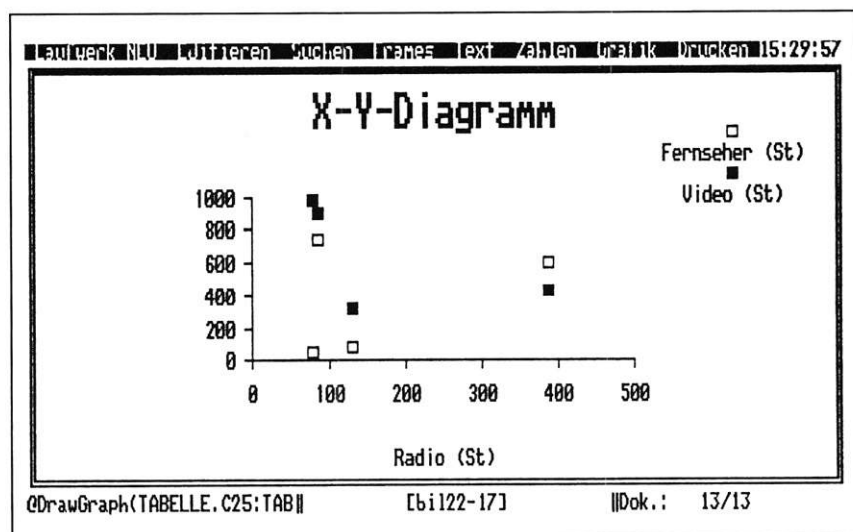


Bild 22.17: X-Y, Zeile als X-Achse

Fürs erste entscheidet sich Herwig Moritz für das Balkendiagramm - mit den Balken nebeneinander:

1. Er öffnet das Menü *Grafik*.
2. Er wählt nötigenfalls den Grafik-Typ und die Achsenorientierung (dabei schaltet das Menü nicht zurück!).
3. Er aktiviert *Neue Grafik zeichnen*.

Merke: Damit eine Grafik gezeichnet werden kann, muß sich der Cursor im Tabellen- oder Datenbank-Frame befinden und die Daten müssen markiert sein.

Framework fragt nach:

RETURN - Grafik hier zeichnen oder anderen Frame wählen und dann
RETURN

Der Cursor steht dabei auf dem Rand des Tabellen-Frames. Doch was Herwig Moritz befürchtet, tritt nicht ein, wenn er RETURN drückt: Die Grafik wird nicht in den Tabellen-Frame gezeichnet, sondern Framework öffnet hierfür einen neuen Frame.

Die Abfrage mit ihrer Wahlmöglichkeit ist für den Fall gedacht, daß Herwig Moritz eine bereits vorhandene Grafik neu zeichnen lassen möchte - etwa, weil er X- und Y-Achse tauscht oder einen anderen Grafik-Typ wählt. Denn beides kann er jederzeit machen. Er muß dazu nur wieder den Cursor zurück in den Tabellen-Frame bringen, im Menü *Grafik* die entsprechenden Vorgaben machen und sodann die Grafik neu zeichnen lassen. So kann er sämtliche Möglichkeiten durchspielen, wenn er für die Grafik nicht jedesmal einen neuen Frame erstellt haben will.

Auf diese Weise kann man die optimale Präsentationsform herausfinden. Und kann auch direkt vergleichen. Denn Grafik-Frames sind Frames wie alle andere, und demzufolge kann man mit ihnen alles machen, was man mit Frames gemeinhin macht.

Merke: Eine Grafik kann auch unabhängig von der Tabelle aufgerufen, gedruckt und geplottet, hingegen nicht bearbeitet werden.

Zum Beispiel kann Herwig Moritz die Grafik mit F9 (*Zoom*) auf volle Bildschirmgröße bringen. Hat er einen Colormonitor (und ihn als solchen auch installiert), wird sie ihm dann farbig präsentiert - das einzige Mal, daß Framework die Farbmöglichkeiten ausschöpft. (Der FRED-kundige allerdings kann sich farbige Zeichnungen erstellen.)

Natürlich kann man auch Größe und Position des Grafik-Frames ändern - und sich die Arbeitsfläche so organisieren, daß man mehrere Grafiken nebeneinander betrachten kann.

Die Größe zu verändern, ist ganz spaßig. Im Gegensatz zu Text-, Tabellen- oder Datenbank-Frames wird dabei nicht der Ausschnitt verschoben, sondern Framework

zeichnet die Grafik neu und paßt sie der Frame-Größe an. Mithin läßt sich auf diese Weise eine Grafik dehnen oder quetschen. Was natürlich nicht nur Spielerei ist, sondern erhebliche Auswirkungen für den Druck hat: Frame-Größe ist gleich Druck-Größe.

Und schließlich kann Herwig Moritz eine Grafik auch editieren. Framework macht's dabei etwas umständlich. Ohnehin sind die Editiermöglichkeiten gering und wenig flexibel, und zudem gelten sie (mit einer Ausnahme) erst für die nächste Zeichnung. Will Herwig Moritz seine aktuelle Grafik überarbeiten, muß er sie deshalb nochmals zeichnen lassen.

Tip: Wollen Sie die Überschrift einer Grafik nicht auch als Frame-Namen haben (etwa weil es eine Datei dieses Namens bereits gibt), dann ändern Sie zum Speichern den Frame-Namen. Als Überschrift wird er nur mit **F5** (*Neuberechnung*) übernommen oder wenn die Grafik neu gezeichnet wird.

Die Ausnahme ist die Überschrift der Grafik. Sie ist der Name des Grafik-Frames - ganz einfach. Sie wird in die Mitte der Grafik gestellt, die Schriftgröße ist vorgegeben und nicht veränderlich, Herwig Moritz kann lediglich entscheiden, ob er sie fett, kursiv oder unterstrichen haben will.

Zur Erinnerung: Um die Schriftauszeichnung eines Frame-Namens zu ändern, muß er mit der Leertaste zunächst editiert werden.

Alle anderen Bearbeitungsmöglichkeiten gehen über das Untermenü *Optionen* des Menüs *Grafik* und sind tunlichst vor dem Zeichnen der Grafik zu überlegen:

- die Beschriftung der X-Achse
- die Beschriftung der Y-Achse
- die manuelle Skalierung der Y-Achse, wobei deren größter und kleinster Wert sowie die Schrittweite bestimmt werden kann
- bei einem Tortendiagramm, welches Segment herausgezogen werden soll

Wie gesagt: Will Herwig Moritz das seiner bereits bestehenden Grafik hinzufügen bzw. in ihr ändern, muß er sie nochmals neu zeichnen lassen.

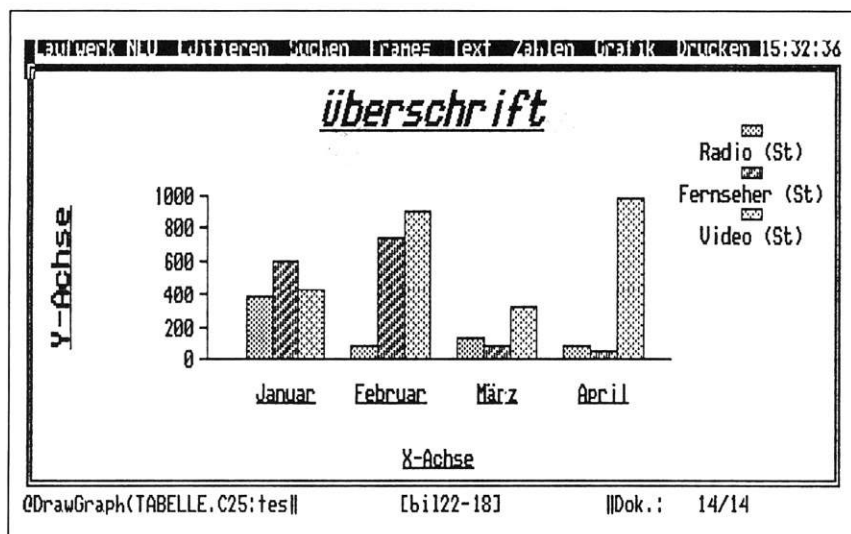


Bild 22.18: Eine überarbeitete Grafik mit beschrifteten Achsen und Titel

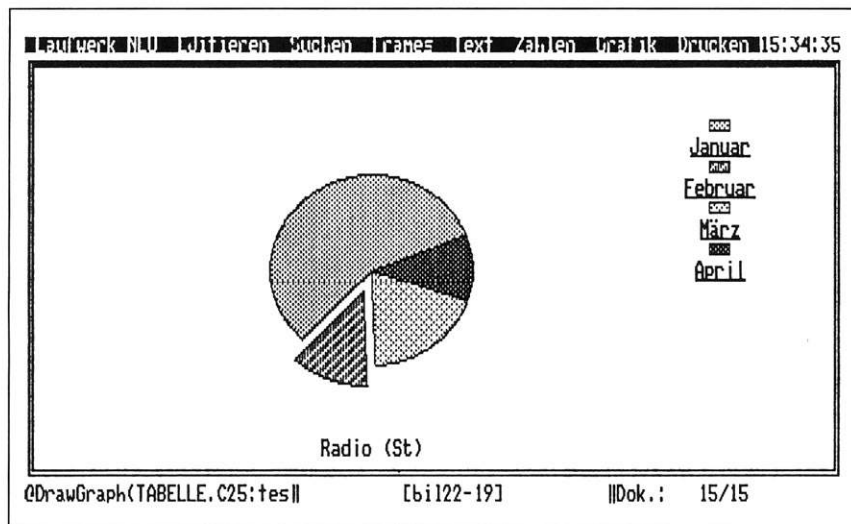


Bild 22.19: Tortendiagramm mit herausgezogenem Segment

Es sei denn, er bestimmt seine Änderungen nicht über das Menü, sondern in Handarbeit:

1. Mit **[F2]** (*Formel editieren*) macht er die Formel sichtbar, die sich hinter dem Grafik-Frame verbirgt.
2. Mit **[F9]** (*Zoom*) bringt er sie auf volle Bildschirmgröße, damit sie leichter bearbeitet werden kann.

Diese Formel ist gar nicht so schlimm, wie sie aussieht. Und folgendermaßen sieht sie aus, wenn beispielsweise all ihre Parameter benutzt werden:

```
@DrawGraph(TABELLE.C25:TABELLE.F27, #ROW, #BAR, "Überschrift", "X-Achse", "Y-Achse", 0, 1000, 200)
```

Das bedeutet:

@DrawGraph	die FRED-Funktion für das Zeichnen der Grafik
TABELLE.C25:TABELLE.F27	der Bereich in der Tabelle
#ROW	Zeile als X-Achse (Spalte wäre #COLUMN)
#BAR	der Grafik-Typ
"Überschrift"	die Überschrift der Grafik
"X-Achse"	der Titel der X-Achse
"Y-Achse"	der Titel der Y-Achse
0	kleinster Y-Wert
1000	größter Y-Wert
200	Schrittweite der Y-Achse

Merke: Die Namen der Grafik-Typen sind:

#BAR	<i>Balken</i>
#STACKEDBAR	<i>Turn</i>
#PIE	<i>Kreis</i>
#LINE	<i>Linie</i>
#MARKEDPOINTS	<i>Punkt</i>
#XY	<i>XY</i>

Bei dieser Grafik-Formel müssen nur wenige Dinge beachtet werden:

- alle Parameter sind von Klammern umschlossen - und zwar alle gemeinsam
- alle Parameter werden durch Kommata getrennt
- alle Parameter müssen angegeben werden; werden für sie keine Werte gesetzt (etwa, wenn man die Achsen nicht beschriftet), steht nur das Komma
- die Titel (Überschrift, Achsenbeschriftung) müssen zwischen Anführungszeichen stehen.

Künftig wird Herwig Moritz gewiß die Formel editieren, wenn er eine Grafik überarbeitet oder neu zeichnen lassen will. Das geht allemal schneller, als der Weg übers Menü. Zudem kann er auf diese Weise auch die Achsenbeschriftungen nach Wunsch

fett, kursiv oder unterstrichen auszeichnen: Eine Formel kann ja editiert werden wie ein normaler Text.

Merke: Soll eine Grafik bearbeitet oder neu berechnet werden, muß die zugrundeliegende Tabelle auf der Arbeitsfläche sein, weil Framework sonst nicht weiß, woher die Zahlen zu nehmen sind.

Eine vorhandene Grafik kann auch von einer anderen überlagert werden, allerdings nur unter eingeschränkten Bedingungen:

- Die vorhandene Grafik darf kein Kreis- oder XY-Diagramm sein.
- Die überlagernde Grafik muß ein Linien- oder Punkt-Diagramm sein.

Zwei Grafiken in einer - das erfordert von Herwig Moritz ein paar Arbeitsschritte mehr:

1. Er markiert in der Tabelle einen Bereich.
2. Er zeichnet eine neue Grafik - etwa ein Balkendiagramm.
3. Er wechselt wieder zurück in die Tabelle und markiert dort einen anderen Bereich.
4. Er wählt einen anderen Grafik-Typ (zum Beispiel *Linie*) und dann *Überlagern vorhandene Grafik*.
5. Er zeichnet die zweite Grafik in den ersten Grafik-Frame.

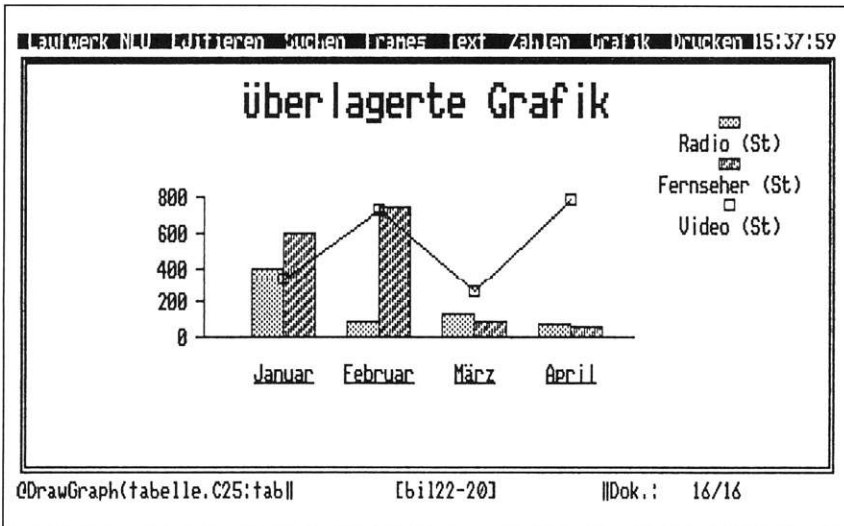


Bild 22.20: Ein Balkendiagramm, von einer Linien-Grafik überlagert

Nun also hat Herwig Moritz alles beisammen für seinen Bericht: eine Tabelle, eine Grafik, ein Konzept (aus Kapitel 19), welches sich unterdessen auf wundersame Weise auch mit etwas Text gefüllt hat - die Übungsdatei heißt `BERICHT.FW`.

Der Text befindet sich unter dem Punkt »3.2 Die Entwicklung von 1970 bis 1986«. Herwig Moritz hat ihn einfach nur so geschrieben, in einer Minute, die von guten Einfällen gekrönt war. Es war ihm zwar da schon klar, an welchen Stellen er Tabelle und Grafik einblenden möchte, doch hat er keine Rücksicht darauf genommen, wie das technisch vonstatten gehen soll.

Zur Erinnerung: Ein Frame kann entweder Text (oder Tabelle oder Datenbank) enthalten oder einen Frame, aber nicht beides zusammen.

Er kann ja nicht einfach die Tabelle in den Text hineinsetzen, er muß Frames verschachteln - und hat das ja gelernt und geübt in der Konzept-Version (Kapitel 19).

Er muß dazu seinen schon geschriebenen Text in verschiedene Frames aufteilen und sodann die Tabelle und die Grafik dazwischenschieben.

Der Text ist bisher im Gliederungspunkt 3.2 zu finden - was bedeutet, daß keine Unterabschnitte mehr gebildet werden können, weil der Frame schon Text enthält. Herwig Moritz hat deshalb einige Arbeit vor sich:

1. Er kopiert den kompletten Punkt »3.2 Die Entwicklung ...« zweimal und macht ihn zu den Punkten 3.3 und 3.4.
2. Punkt 3.3 nennt er »Text 1«. Er löscht in diesem Frame den letzten Absatz.
3. Punkt 3.4 nennt er »Text 2«. Er löscht in diesem Frame die ersten beiden Absätze.
4. Er löscht den gesamten Text des Punktes 3.2 - er schafft damit wieder einen leeren Frame, damit er ihn untergliedern kann:
5. Mit `CTRL` `NUM+` erzeugt er einen Unterabschnitt 3.2.1.
6. Er markiert mit `F6` (*Auswahl*) die Punkte 3.3 und 3.4.
Er verschiebt sie mit `F7` (*Verlagern*) nach Punkt 3.2.1, macht aus ihnen also die Unterabschnitte 3.2.2 und 3.2.3.
7. Er löscht den Unterabschnitt 3.2.1 - den hat er ja nur benötigt, um überhaupt Unterabschnitte einfügen zu können.
8. Er kopiert die Tabelle nach »Text 1«.
9. Er kopiert die Grafik nach »Text 2«.



Bild 22.21: Die neue Struktur des Konzeptes

10. Er markiert alle Unterpunkte und schaltet im Menü *Frames* das *Durchnumerieren der Namen* ab.
11. Er schaltet mit **F10** (*Zeigen*) um auf die Frame-Version.
12. Er verändert bei den Frames »Text 1« und »Grafik« Position und Größe so, daß sie nebeneinander zu stehen kommen - die Frame-Position ist Druckposition.

Zur Erinnerung: Wenn sich Frames nicht verschieben lassen, muß zuerst *Position frei wählbar* im Menü *Frames* eingeschaltet werden.

13. Er unterdrückt im Menü *Frames* bei diesen Sub-Frames die Namen-Anzeige - so hat er den Eindruck, es handle sich um ein fortlaufendes Dokument.

Laufwerk NEU Editieren Suchen Frames Text Zahlen Grafik Drucken 17:52:16						
[3 Der Geschäftsverlauf]						
[3.2 Die Entwicklung von 1970 bis 1986]						
der Umsatzentwicklung zu beobachten? Ist also der Amüsiertrieb so mächtig, daß er sich auch im Kaufverhalten niederschlägt? Ein Run auf die Unterhaltungselektronik? Das Käuferinteresse hat sich in den ersten vier Monaten dieses Jahres sehr unterschiedlich entwickelt, wie folgende Tabelle zeigt:						
	A	B	C	D	E	F
1			Januar	Februar	März	April
2	Umsatz					
3	Radio (St)		387	85	129	77
4	Stückpreis	99				
5	Umsatz		38313	8415	12771	7623
6	Fernseher (St)		596	745	89	56
[bil22-22] Dok.: 2/2						

Bild 22.22: Text und Tabelle

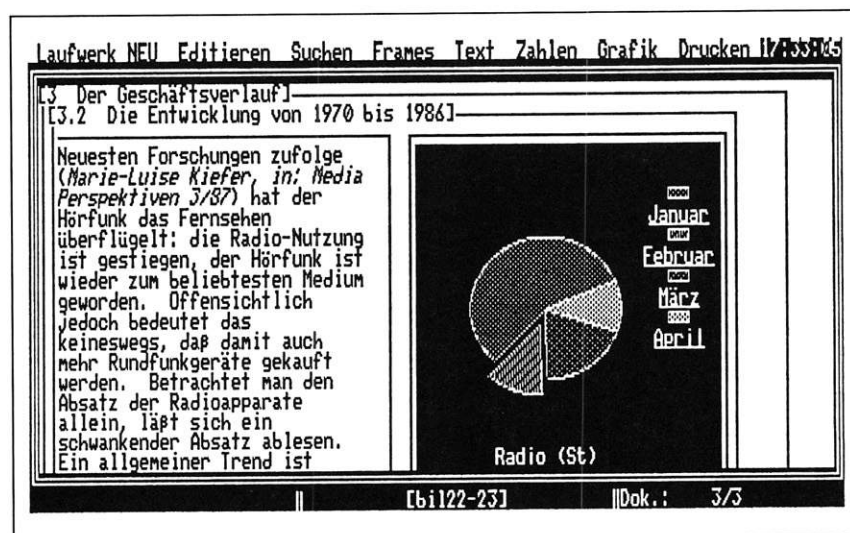


Bild 22.23: Text neben Grafik

Weil er diese Frames allesamt zu Sub-Frames auf der gleichen Ebene gemacht hat, kann er nach Belieben seinen Bericht erweitern und Text, Tabellen und Grafiken mischen: Framework, das integrierte Paket. Wenn er diesen Bericht jetzt seinem Chef einreicht, legt er gewiß Ehre ein.

■| **Fortsetzung:** Kapitel 30 - eine Tabelle mit einem Makro auswerten.

23 Verkaufsplanung

oder wie der Bäckermeister Werner Pfisterer zwischen Sahnetorte und Brötchen schwankt

Sie erfahren in diesem Kapitel:

- wie man eine Tabelle in Abhängigkeit verschiedener Variablen neu berechnet
- wie man eine »Was wäre, wenn ...«-Analyse durchführt
- wie man in einer Tabelle Zellen schützt
- wie man Zahlenformate gestaltet

Sie benötigen:

- die Übungsdatei TORTE.FW

Das Problem

Soweit ist der Bäckermeister Werner Pfisterer ganz zufrieden. Die Geschäfte laufen, und der Gewinn ist ordentlich. Aber vielleicht, überlegt er, könnte er noch ein bißchen mehr verdienen? Und tatsächlich, er hat da einige Hersteller ausfindig gemacht, die ihm das Mehl und die Sahne für die Sahnetorten billiger verkaufen als sein bisheriger Lieferant.

Der Pferdefuß: Er muß sich seine Sachen selber abholen. Die Hersteller stellen nur her und liefern nicht aus. Deshalb sind sie ja so billig. Wie dieses unmögliche Möbelhaus aus ...

Und weil das nicht gerade um die Ecke ist, hebt sich der Preisvorteil vielleicht wieder auf. Denn Werner Pfisterer verbraucht Benzin und frißt Kilometer, was wiederum den Wert seines Wagens drückt.

Fährt er mit seinem alten, aber teureren Lieferanten doch günstiger?

Mit Taschenrechner und Bleistift hätte Werner Pfisterer einige ausgefüllte Abende vor sich. Denn er steht vor einer Rechnung mit mehreren, von einander abhängigen Variablen. Billigeres Mehl drückt seine Kosten, aber er muß die zusätzlichen Aufwendungen für die Fahrt berücksichtigen. Je mehr Mehl er auf einmal transportiert, desto weniger fallen die Fahrtkosten ins Gewicht. Lohnt sich unter Umständen gar die Anschaffung eines größeren Wagens, obschon dann jeder gefahrene Kilometer teurer wird? Lohnt sich der etwas weitere Weg zu einem noch billigeren Anbieter?

Weil Werner Pfisterer schon mal dabei ist, seine gesamte Verkaufskalkulation zu berechnen, denkt er über weitere Einsparungsmaßnahmen nach.

Werner Pfisterer führt eine »Was wäre, wenn ...«-Analyse durch. Wenn sich ein Posten seiner Kalkulation ändert, zieht das Änderungen bei vielen anderen Posten nach sich.

Die Lösung

Die Tabelle, die sich Werner Pfisterer unter dem Namen TORTE.FW erstellt hat, berücksichtigt der Einfachheit halber lediglich Sahnetorten und Brötchen und verzichtet auf sämtliche sonstigen Zutaten, die aus Mehl und Sahne erst etwas Leckeres machen, vom Backpulver bis zur Belegkirsche. Und die Preis- und Verbrauchsangaben mögen mir alle davon betroffenen Branchen verzeihen - es geht hier ums Prinzip.

Und für das muß Werner Pfisterer zunächst einiges an Gehirnschmalz aufbringen. Die Tabelle soll hinreichend differenziert sein, aber noch genügend übersichtlich, um aussagekräftig zu bleiben.

Tip: Planen Sie den Aufbau einer Tabelle erst mit Bleistift und Papier. Nachher sind grundlegende Änderungen mühselig.

Laufwerk NEU Editieren Suchen Frames Text Zahlen Grafik Drucken 22:56:20			
	A	B	C
1		Sahnetorten	Brötchen
2	Lohn		
3	Arbeitszeit (min)	30	,500
4	Stundenlohn	DM 48,000	DM 48,000
5	Lohn pro Stück	DM 24,000	DM ,400
6			
7	Material		
8	Betriebskosten pro km	DM ,600	
9	Sahne		
10	Einkaufspreis (l)	DM ,530	
11	Verbrauch	75	
12	Weg (km)	43	
13	Transport (l)	280	
14	Weg-Kosten	DM 25,800	
15	Weg-Kosten (pro l)	DM ,092	
16	Sahne-Kosten (pro Torte)	DM ,467	
17			
18	Mehl		

[bild23-1] Dok.: 1/1

Bild 23.1: Die Übungsdatei TORTE

Wenn Werner Pfisterer einen neuen, leeren Tabellen-Frame erstellt und sich über den grundsätzlichen Aufbau im klaren ist, entfernt er zunächst alle überflüssigen Zeilen und

Spalten - in der Beispieldatei ist eine leere Spalte belassen worden, wenn Sie's üben wollen - über das Menü *Editieren* (weil er dann mit dem Cursor besser springen kann) und ruft sich dabei stets vor Augen, daß Zeilen von links nach rechts und Spalten von oben nach unten gehen.

■ **Merke:** Gelöschte Zeilen oder Spalten können nicht mehr zurückgeholt werden, sind also, samt ihren Daten und Formeln, für immer verloren.

Werner Pfisterer wähnt sich ganz schlau und denkt, er könne sämtliche Zeilen und Spalten, die erhalten bleiben sollen, schützen:

1. Er markiert sie mit **F6** (*Auswahl*).
2. Er schützt sie über das Menü *Editieren*.

Nun kann nichts mehr passieren. Meint er. Er kann in der Tat die geschützten Zellen mit **DEL** nicht mehr löschen. Er kann, indem er einfach losschreibt, keine andere Zahl eingeben. Aber er kann mit **F2** (*Formel editieren*) den Formelspeicher öffnen - und auf diese Weise dann doch die Zahlen ändern. Und er kann die Spalte nach wie vor entfernen. Der Schutz funktioniert also nur teilweise und ist mehr ein Schutz gegen Unachtsamkeiten. Es sei denn, er schützt den gesamten Frame (mit dem Cursor auf dessen Rand). Dann lassen sich Zeilen und Spalten erst entfernen, wenn der Schutz wieder aufgehoben ist.

Der Übersichtlichkeit dienlich sind auch die vielfältigen Gestaltungsmöglichkeiten von Framework. Fett, kursiv, unterstrichen: Damit bringt man Ordnung in die Zeilen und Spalten, hebt Wichtiges hervor.

■ **Zur Erinnerung:** Schriftauszeichnungen, auf dem Rand des Frames angewiesen, gelten für den gesamten Frame, ansonsten nur für die markierten Bereiche.

Zudem läßt sich jedes Feld einzeln formatieren. Normalerweise ordnet Framework bei Tabellen den Text linksbündig und die Zahlen rechtsbündig an; so ist die Voreinstellung im Menü *Zahlen*. Genauso indessen läßt sich ein Text (oder eine Zahl) rechtsbündig oder zentriert darstellen.

Auch das Zahlenformat kann nach Wunsch dargestellt werden:

Standardformat	7654,32
Ganze Zahl	7654
Festpunktzahl	7654,32
Währungsbetrag	DM 7.654,32
Unterteilung in Tausender	7.654,32
Prozent	765432,00%
Exponentendarstellung	7,65E+3

Für alle Formate (abgesehen von der Ganzzahl) kann Werner Pfisterer überdies die Zahl der Dezimalstellen definieren:

1. Er aktiviert die Funktion *Dezimalstellen* im Menü *Zahlen*.
2. In der Statuszeile erscheint die gegenwärtig geltende Zahl der Dezimalstellen (Voreinstellung ist 2).
3. Werner Pfisterer ändert, zum Beispiel in 4, was sich für manche Werte seiner Tabelle anbietet.
4. Mit **RETURN** hat er dem markierten Feld das neue Format zugewiesen.

Natürlich kann Werner Pfisterer auch die gesamte Tabelle global formatieren, indem er nicht einzelne Felder, sondern den gesamten Frame markiert und dazu mit **NUM-** den Cursor auf den Rand des Frames bringt. Allerdings: Hat er zuvor schon einzelne Felder anders formatiert, werden diese durch eine globale Formatierung nicht verändert. Das gilt für das Zahlenformat und die Dezimalstellen sowie für die Ausrichtung der Daten.

Merke: Individuell formatierte Felder werden von einer globalen Formatierung des Frames nicht berührt.

Als letztes schließlich kann Werner Pfisterer entscheiden, ob die Zahlen Dezimalpunkt oder Dezimalkomma haben und ob die Währungsbezeichnung vor oder hinter die Zahl gestellt wird. Dazu allerdings muß er die entsprechenden Werte in der Datei CONFIG.FW ändern und Framework nochmals neu starten - denn CONFIG.FW wird ja nur beim Start des Programmes wirksam (siehe Kapitel 8).

Als Grundeinstellung hat Framework die Währungsbezeichnung nach dem Betrag; für eine Tabelle indes bietet sich eine Umstellung an, damit die Zahlen untereinander zu stehen kommen. Ohnehin hat Werner Pfisterer lange geschwankt, ob er überhaupt dieses *DM* in die Datenfelder aufnehmen soll und hat sich dann doch dafür entschieden: um die Geldbeträge deutlich von den anderen Werten zu unterscheiden.

Merke: Geänderte Zahlendarstellungen in CONFIG.FW ändern individuell formatierte Felder nicht automatisch.

Das wäre das äußere Erscheinungsbild der Tabelle. Viel mehr Kopfzerbrechen hat Werner Pfisterer der innere Aufbau bereitet. Vor allem, welche Felder er in welchen Bezug zueinander setzt.

Wie sich die Werte errechnen, welche Formeln also einzugeben sind, ist nicht weiter schwierig, das ergibt sich aus der Natur der Sache. Doch Werner Pfisterer möchte ja nicht nur zusammenzählen und multiplizieren, er möchte wissen, wie sich die Änderung eines Wertes, zum Beispiel des Einkaufspreises für Mehl, auf die gesamte Kalkulation auswirkt.

Deshalb hat er für diese Dreh- und Angelpunkte seiner Berechnung eigene Felder vorgesehen - für den Stundenlohn seiner Angestellten, für die Wegekosten pro Kilometer, für die Menge an Mehl und Sahne, die er auf einmal transportieren kann. Und statt diese Werte in die Berechnungsformeln einzugeben, hat er sie jeweils absolut adressiert.

|| Zur Erinnerung: Ein \$-Zeichen vor einem Zeilen- und/oder Spaltennamen schafft einen absoluten Bezug.

Auf diese Weise hat er die Garantie, daß sich alle nötigen Daten neu errechnen, sobald er die Grundwerte ändert.

Welche Formeln wohin gehören, bereitet weiter kein Kopfzerbrechen, und sie sind auch nicht so kompliziert, daß sie diskutiert werden müßten. Es geht ja hier auch um ganz was anderes. Mit dem Zahlenmaterial will Werner Pfisterer ergründen, wie er rentabler arbeiten kann. Weil die Daten alle aufeinander bezogen sind, werden sie neu berechnet, sobald ein Wert sich ändert.

|| Hinweis: Die Übungstabelle ist vollständig ausgefüllt.

So kann Werner Pfisterer also spekulieren. Was wäre,

- wenn er das Mehl um drei Pfennige billiger einkauft, dafür aber 15 Kilometer weiter fahren muß?
- wenn er sich einen größeren Wagen anschafft, was zwar die Wegekosten auf 98 Pfennig pro Kilometer erhöht, ihm aber ermöglicht, 600 Kilogramm Mehl auf einmal zu transportieren?
- wenn er den Boden seiner Torte ein wenig dünner macht, also nur noch 280 Gramm Mehl braucht, was den Kunden bestimmt nicht auffällt, aber ihm vielleicht größeren Gewinn verschafft?
- wenn er den Stundenlohn um zwei Mark erhöht, in der Hoffnung, das möge für seinen Gesellen Anreiz genug sein, eine Sahnetorte in nur noch 28 Minuten herzustellen?
- wenn er für seine Brötchen einen Kampfspreis von 20 Pfennig macht, wodurch sich der Absatz verdoppelt?

Werner Pfisterer kann mit einer solchen "Was wäre, wenn ..." - Analyse herausfinden, welche Maßnahme erfolgversprechend ist. Etwas mißlich ist allenfalls, daß er seine Tabelle stets auf- und abrollen muß, um zu sehen, wie sich die Werte wandeln.

Ein grundsätzliches Problem - leider ist der Bildschirm halt so klein. Vielleicht schaut Werner Pfisterer einmal ins nächste Kapitel hinein. Der Dichter Jonathan Kraft hat eine (eine!) Möglichkeit gefunden, diese Unannehmlichkeit zu bewältigen.

24 Überschuß-Berechnung (I)

oder wie der Dichter Jonathan Kraft seine roten Zahlen verwaltet

Sie erfahren in diesem Kapitel:

- wie man in einer Datenbank mit globalen Formeln rechnet
- wie man Tabelle und Datenbank aufeinander bezieht
- wie man mehrere Frames übersichtlich anordnet

Sie benötigen:

- EINNAHME.FW, AUSGABEN.FW und GEWINN.FW von der Beispieldiskette
- BILANZ.FW von der Beispieldiskette

Fortsetzung:

- leichtere Eingabe und Auswertung mit einem selbst entwickelten FRED-Programm (Kapitel 34)

Das Problem

Trotz unermüdlichen Fleißes ist das Genie des Dichters Jonathan Kraft noch nicht erkannt, und so nährt er sich, neben der wahren, doch brotlosen Kunst, redlich mit allerlei kleinen Gelegenheitsarbeiten. Was ihm alljährlich peinvolle Tage beschert, wenn er für seine Steuererklärung Stöße von Belegen sortieren und zusammenrechnen soll.

Wegen der Vielzahl der Kleinstbeträge - hier mal ein paar Mark Honorar für eine Buchbesprechung, dort ein paar Pfennige für einen Bleistift - kann er seine Finanzlage schlecht einschätzen und ist über die roten Zahlen auf seinem Konto stets baß erstaunt. Verdient er so wenig? Verschlingen seine berufsnotwendigen Ausgaben so viel Geld? Oder lebt er einfach zu gut?

Zudem nervt ihn seine Steuerberaterin. Sie hat ihm das letzte Mal ein dickes Heft in die Hand gedrückt, »Kassenbuch« betitelt, und darin soll er säuberlich seine Einnahmen und Ausgaben eintragen. Nebst Mehrwertsteuer. Ist er nun Dichter oder Buchhalter?

Mit Junior Framework müßte das doch etwas einfacher gehen. Jonathan Kraft braucht Tabellen, in denen er lediglich seine Einnahmen und Ausgaben eintragen muß. Die weitere Verarbeitung der Daten soll das Programm übernehmen. Insbesondere sollen folgende Forderungen erfüllt werden:

- die Mehrwert- bzw. Umsatzsteuer soll vom Programm errechnet und eigens ausgewiesen werden
- er will jederzeit eine aktuelle Bilanz präsentiert bekommen.

Die Lösung

Als Demonstrationsobjekt hat Jonathan Kraft zwei unschätzbare Vorteile. Zum einen muß er für seine künstlerische Arbeit lediglich sieben Prozent Umsatzsteuer abführen, aber für alle sonstigen Dienstleistungen, die er erbringt (z.B. macht er gelegentlich Lektoratsarbeit oder liest Korrektur) 14 Prozent, und ebenso unterliegen seine berufsnotwendigen Ausgaben unterschiedlichen Mehrwertsteuer-Sätzen. Die Tabellen müssen also mehrere Mehrwertsteuer-Variablen berücksichtigen - das kompliziert die Sache.

Auf der anderen Seite wird von Jonathan Kraft keine Buchführung im üblichen Sinne verlangt, nur eine Einnahmen-Überschuß-Berechnung; zudem ist, seiner schwankenden Einkünfte wegen, lediglich ein Jahressaldo notwendig, hingegen eine monatliche oder quartalsweise Übersicht entbehrlich. Das vereinfacht das Problem.

Es genügte also, die Einnahmen und Ausgaben gegeneinander aufzurechnen. Doch Jonathan Kraft will sich nicht allein das manuelle Rechnen ersparen, er möchte auch seine Finanzlage analysieren können und herausfinden, womit er am meisten verdient, und wo er nötigenfalls einsparen kann.

Jonathan Kraft benutzt drei verschiedene Frames: EINNAHME, AUSGABEN und GEWINN. Er unterteilt also nach unterschiedlichen Sparten, in der Überlegung, daß eine Tabelle allein wohl doch zu unübersichtlich wäre. Er hat sogar mit dem Gedanken gespielt, noch weiter zu unterteilen und seine Einnahmen und Ausgaben zu Sachgruppen zusammenzufassen: eine Datei für das Porto, eine für die Parkgebühren, eine fürs Auto ...

Monat	Gegenstand	Brutto 14	Brutto 7	Brutto 0	Netto 14	Netto
1	Porto			2,20		
3	Porto			14,40		
1	Farbbänder	48				
1	Abo Xtra		60			
3	Parkgebühr	3,50				
2	Benzin	12,50				
2	Zeitschrift		5,50			
1	Disketten	488,15				
4	Bürostuhl	495				

Bild 24.1: Der AUSGABEN-Frame

Befürchtet hat er indes, er verlöre dadurch den Überblick. Und ohnehin fühlt er sich von solcherlei buchhalterischen Arbeiten belästigt und versucht darum, möglichst wenig Zeit darauf zu verschwenden. Jede Rechnung, jeder Beleg wird sofort erfaßt.

Seine Einnahmen und Ausgaben erfaßt er seltsamerweise nicht in einer Tabelle, sondern in einer Datenbank. Warum?

Weil er nicht monats- oder quartalsweise Buchhaltung macht, sondern täglich fort-schreibt, bleibt seine Tabelle stets unabgeschlossen: Jeden Tag kommen neue Posten hinzu.

Das aber macht es ihm schwierig, die Beträge zu addieren, denn er kann ja keine Bezugspunkte herstellen. Oder er müßte jedesmal eine neue Rechenformel eingeben. Das ist ihm zu kompliziert. Er will schließlich jederzeit und sofort die aktuelle Bilanz sehen, sonst könnte er gleich den Taschenrechner nehmen.

Auch mit den notwendigen Formeln für die Mehrwertsteuer-Berechnung hat er so seine Probleme. Er müßte vorab eine leere Tabelle mit Formeln füllen.

■ **Merke:** Eine Tabelle, die lediglich ein Formelgerüst enthält, wird Maske genannt - nicht zu verwechseln mit der Masken-Version einer Datenbank.

Mit F8 (*Kopieren*) ist das weiter nicht schwierig. Allerdings, wenn er genügend vor-arbeitet, schleppt er dauernd eine riesige, speicherplatzfressende Tabelle mit sich herum - es kommt ja einiges zusammen im Laufe eines Jahres.

Genausogut könnte er für jeden Eintrag die vorherige Zeile kopieren, samt ihren Formeln, die sich dann - relatives Adressieren - automatisch an die neuen Bezüge anpassen. Auch das ist ihm jedoch zu viel Arbeit.

Eine Datenbank bringt die Lösung. Auch in einer Datenbank läßt sich rechnen, mit den gleichen Formeln wie in einer Tabelle, jedoch anders gearteten Bezügen.

Zunächst natürlich muß die Frage geklärt werden, wie die Datenbank aufgebaut werden soll. Der unterschiedlichen Mehrwertsteuersätze wegen legt Jonathan Kraft für jeden ein eigenes Feld an und trennt zudem noch zwischen Brutto- und Nettobeträgen sowie der Mehrwertsteuer. Die soll er extra ausweisen, hat seine Steuerberaterin gesagt. Naja, wenn sie meint ...

Die Übungsdateien EINNAHME und AUSGABEN enthalten zwar einige Daten, hingegen noch keine Formeln; Jonathan Kraft trägt sie jetzt erst ein. Als Künstler hat er ein gestörtes Verhältnis zu Zahlen und lange gebraucht, bis er die richtige Rechen-formel ausgetüfelt hat.

■ **Hinweis:** Falls Sie sich das Eingeben der Formeln ersparen möchten: in der Übungsdatei BILANZ, auf die später eingegangen wird, sind sie bereits enthalten.

Er schreibt sie als globale Formeln hinter die Feldbezeichnungen, in die erste Zeile also, und nimmt sich zuerst das Feld »Netto 14« vor. Mit diesem Standort gewährleistet er, daß sich alle Sätze der Datenbank in diesem Feld diese Formel hier holen: eine Formel für eine ganze Spalte.

■ **Merke:** Globale Formeln in einer Datenbank stehen in der ersten Zeile hinter den Feldbezeichnungen und beziehen sich auf alle Sätze.

Lokale Formeln stehen hinter einem Feld und beziehen sich nur auf dieses Feld in diesem Satz.

Und zwar soll sie den Netto-Wert jener Brutto-Beträge errechnen, auf die 14 Prozent Mehrwertsteuer anfallen, während »Mwst. 14« den Mehrwertsteuer-Betrag ausweist. Für sieben Prozent Mehrwertsteuer stehen ebenfalls die entsprechenden Felder bereit; zudem gibt es noch eine Spalte »Brutto 0« - für Porto beispielsweise. Die Einnahmen-Datenbank ist ähnlich aufgebaut.

Jonathan Kraft will also lediglich die Brutto-Beträge eingeben müssen, alles andere soll sich automatisch errechnen.

1. Er bringt den Cursor in das Feld »Netto 14«.
2. Mit **[F2]** (*Formel editieren*) öffnet er die Editierzeile für die Formeleingabe.
3. Er gibt ein:

```
@set(netto 14,(brutto 14/114)*100)
```

@set ist eine Zuweisungsfunktion von Framework, die zwei Parameter benötigt - also Werte, auf welche diese Funktion angewendet werden soll. Ihre allgemeine Form lautet:

```
@set(Bezug, Ausdruck)
```

@set weist **Bezug** den **Ausdruck** zu - was immer dieser sein mag. Zum Beispiel eine Rechenformel. Der Bezug ist in diesem Fall der Name eines Feldes: »Netto 14«. Auf dieses Feld soll der Ausdruck, die Rechenformel, angewendet werden.

■ **Merke:** Parameter bei Funktionen werden mit einem Komma getrennt.

Für eine solche Zuweisung ist auch noch eine andere Schreibweise möglich, die das gleiche Ergebnis bringt wie **@set**:

```
netto 14:=(brutto 14/114)*100)
```

Bezug und Ausdruck stehen hier in der gleichen Reihenfolge, werden indessen mit **:=** getrennt. Diese Schreibweise ist etwas geläufiger als die **@set**-Formel.

Die Rechenformel, über der Jonathan Kraft so lange gebrütet hat, muß auf die Tatsache Rücksicht nehmen, daß ja der Netto-Wert 100 Prozent entspricht, dem dann 14 Prozent Mehrwertsteuer zugeschlagen werden.

Entsprechend lautet die Formel für das Mehrwertsteuer-Feld:

```
@set([mwst. 14],(brutto 14/114)*14)
```

Oder, in anderer Schreibweise:

```
[mwst. 14]:= (brutto 14/114)*100)
```

Hier muß der Feldname von eckigen Klammern eingeschlossen werden, weil er sowohl Text als auch eine Zahl und dazu einen Punkt enthält - insofern eine ungeschickte Spaltenbezeichnung, die man demzufolge nach Möglichkeit vermeiden sollte.

Bei der Rechenformel wiederum sind die runden Klammern nicht vonnöten und dienen lediglich der Klarheit. Ansonsten jedoch schreiben manche Rechenoperationen Klammern vor, weil die Rechenarten eine bestimmte Hierarchie haben, die Jonathan Kraft dank seiner fundierten Schulbildung eigentlich wissen mußte, aber natürlich längst vergessen hat.

Merke: Die Hierarchie der Rechenoperationen:

- * Multiplikation
- / Division
- + Addition
- Subtraktion

Entsprechend erstellt sich Jonathan Kraft die restlichen Formeln, wobei er sie, um Tipparbeit zu sparen, mit **F8** (*Kopieren*) kopiert, auch in die andere Datenbank und dann lediglich die Bezüge und Werte ändert, wo es vonnöten ist.

Bei der Neuberechnung der Frames erhält Jonathan Kraft allerdings stete Fehlermeldungen von Framework:

```
#VALUE!
```

Das braucht ihn jedoch nicht zu kümmern, ist allenfalls unschön. Framework meldet diesen Fehler immer in den leeren Feldern. Klar: Das Programm will rechnen, findet aber keine Werte dazu. Wenn's ihm nicht zu umständlich wäre, könnte Jonathan Kraft jeweils 0 eintragen.

Aber die Fehlermeldungen übersieht er hoheitsvoll. Die beiden Datenbanken sind ja lediglich sein Gerüst. Sie sammeln die Einnahmen und die Ausgaben; worauf's ihm weitaus mehr ankommt, ist GEWINN.FW. Denn hier wird zusammengezählt, hier erfährt Jonathan Kraft, wie's um seinen Geldbeutel steht.

GEWINN.FW ist eine Tabelle, keine Datenbank. Die Bezüge, die für die hier geforderten Berechnungen notwendig sind, lassen sich in einer Tabelle leichter herstellen als in einer Datenbank.

Die Anordnung der Spalten und Zeilen ist vielleicht etwas ungewöhnlich, aber erklärlich. Jonathan Kraft möchte sich seinen Bildschirm so einrichten, daß er alle drei Frames seiner selbstgestrickten Buchhaltung auf einmal im Blick hat. (In der

Übungsdatei BILANZ.FW wird das vollbracht sein, und in ihr sind auch die Formeln schon eingetragen.)

	A	B	C	D
1		Ausgaben		
2	Brutto			
3	Netto			
4				
5		Einnahmen		
6				
7	Brutto			
8	Netto			
9				
10		Bilanz		
11				
12	Brutto			
13	Netto			
14	Gewinn			
15	Brutto			
16	Mwst.			

Bild 24.2: Die GEWINN-Tabelle

Aber der Reihe nach. Zuerst muß Jonathan Kraft das Formelgerüst in GEWINN erstellen.

Die Aufgabe ist klar: Framework soll die verschiedenen Spalten der beiden Datenbanken addieren und mit diesen Ergebnissen dann weiterrechnen.

Merke: Verbindet man zwei Tabellen oder Datenbanken, brauchen Bereichsangaben unbedingt auch die Frame-Namen.

In einer Tabelle verlangt die Additionsformel eindeutige Koordinaten: B2 zum Beispiel - die Zelle B in der Zeile 2. Und sie braucht zwei Koordinaten: B2 soll zu C5 addiert werden.

In einer Datenbank hingegen kann man ein gesamtes Feld, eine gesamte Spalte also ansprechen. Mit seinen Mehrwertsteuer-Berechnungsformeln hat Jonathan Kraft dies bereits getan, und auf gleiche Weise kann er auch alles addieren, was in einem Feld steht:

```
@sum(brutto 14)
```

Überaus praktisch, besonders bei dieser Art Daten, die sich ja ständig vermehren, so daß das Ende des Bezuges jedesmal neu bestimmt werden müßte.

Nun muß Jonathan Kraft die Datenbanken mit der Tabelle verbinden, wozu er sich der Cursormethode bedient, die am komfortabelsten ist und deshalb nochmals ausführlich erklärt wird:

1. Er bringt in GEWINN den Cursor in das Feld »Ausgaben.Netto«.

Zur Erinnerung: Mit ! schaltet man um von den Koordinatenangaben zu den Feldbezeichnungen.

2. Er öffnet mit **[F2]** (*Formel editieren*) die Editierzeile.
3. Er gibt den Anfang der Formel ein:
@sum(
4. Mit **[↑]** schaltet er um auf die Cursormethode.
5. Mit **[NUM-]** verläßt er den Frame GEWINN.FW.
6. Mit den Cursortasten wechselt er über zum Frame AUSGABEN.
7. Mit **[NUM+]** springt er in den Frame.
8. Er markiert das Feld »Brutto 14«. Die Feldbezeichnung erscheint automatisch in der Editierzeile.

Tip: Verwenden Sie nach Möglichkeit die Cursormethode. Framework sorgt dann selber für die richtigen Bezugsangaben in den Formeln.

9. Er gibt die schließende Klammer der Rechenformel ein und gleich deren Fortgang, weil er ja alle Brutto-Spalten addieren will:
@sum(Ausgaben.Brutto 14)+@sum(
10. Mit **[↑]** schaltet er erneut um auf die Cursormethode.
11. Er markiert das Feld »Brutto 7«, schließt diesen Bezug wiederum mit der Klammer und schreibt gleich weiter:
@sum(Ausgaben.Brutto 14)+@sum(Ausgaben.Brutto 7)+@sum(
12. Er fügt wieder mit der Cursormethode das Feld »Brutto 0« hinzu.
13. Mit **[RETURN]** schließt er die gesamte Formel ab.

So kann er nun alle Felder in GEWINN mit den Formeln belegen. Er übernimmt auf diese Weise die Daten aus EINNAHME und AUSGABEN und rechnet mit ihnen in GEWINN weiter - zum Beispiel, welchen Überschuß er erzielt hat, und was er an Mehrwert- bzw. Umsatzsteuer an das Finanzamt abführen muß.

Merke: Die Frames, aus denen Werte entnommen werden, müssen sich auf der Arbeitsfläche befinden.

Allein, das befriedigt ihn noch nicht. Er muß jedesmal drei Dateien laden. Praktischer ist es, die drei Frames zusammenzupacken. In BILANZ ist das Ergebnis zu sehen, und Jonathan Kraft ist folgendermaßen dazu gekommen:

Zur Erinnerung: Frames behalten die Größe und Position, mit der sie gespeichert wurden.

1. Über das Menü *NEU* richtet er einen leeren Text-Frame ein.
2. Er markiert den Frame »Ausgaben«, das heißt, er bringt den Cursor auf dessen Rand.
3. Er drückt **[F7]** (*Verlagern*).
4. Er bringt den Cursor in(!) den leeren Frame.
5. Er drückt **[RETURN]** und schließt die Operation damit ab und holt sich auf die gleiche Weise die Frames »Einnahme« und »Gewinn«.

Laufwerk NEU Editieren Suchen Frames Text Zahlen Grafik Drucken 23:03:48

[Einnahmen]					[Gewinn]	
Monat	Gegenstand	Brutto 14	Brutto 7	Netto 14		B
2	dpa		150	#VALUE!		1 Ausgaben
3	DLF		1500	#VALUE!	1	2 1205,25
2	Schanbacher Zeitu		70	#VALUE!		3 1046,43
1	Esslinger Zeitung		77,04	#VALUE!		4
4	Korrektorat	850		745,61	#VAL	5
1	Verlag A.		301,90	#VALUE!		6 Einnahme
						7 20895,56
						8 19114,04
						9
						10
						11 Bilanz
						12
						13 18067,60
						14
						15 19690,31
						16 1622,71

[Ausgaben]					
Monat	Gegenstand	Brutto 14	Brutto 7	Brutto 0	N
1	Farbbänder	48			
1	Abo Xtra		60		#VAL
1	Porto			2,20	#VAL
3	Parkgebühr	3,50			
3	Porto			14,40	#VAL

|| [bild24-3] ||Dok.: 1/1

Bild 24.3: Alles beisammen: der Frame BILANZ

Jetzt braucht er sie mit **[F3]** (*Position*) und **[F4]** (*Größe*) nur noch geschickt innerhalb des Konzept-Frames anzuordnen, wofür er die Sub-Frames mit **[F10]** (*Zeigen*) zunächst einmal öffnet und hat nun alle drei Frames im Blick. Deshalb wurde GEWINN so eigenartig aufgebaut: damit der Frame als schmaler Streifen rechts abgelegt werden kann.

Somit hat Jonathan Kraft alles beisammen und braucht nur noch einen Frame aufzurufen. Sobald er einen neuen Einnahme- oder Ausgabe-Posten erfaßt hat, wechselt er zum Sub-Frame GEWINN über, drückt mit Wollust **F5** (*Neuberechnung*) und hat einen stets aktuellen Überblick über seine Finanzlage.

Er kann auch seine Einnahmen und Ausgaben kontrollieren. Weil sie in Datenbanken erfaßt sind, kann er selektieren, also einen Auszug aus dem gesamten Datenwust erstellen (siehe Kapitel 20). Was hat er bisher bei der »Schanbacher Zeitung« verdient? Und wieviel im März? Und was hat er im Februar an Porto ausgegeben?

Ein wenig hat damit die Buchführung ihren Schrecken verloren. Er hat zumindest alles erfaßt und bilanziert, und was die Steuerberaterin damit anfängt, ist ihre Sache.

Gleichwohl, wenn er so darüber nachdenkt, reicht ihm dieses Modell noch nicht aus. Er kann zwar die einzelnen Honorarüberweisungen der »Schanbacher Zeitung« selektieren, aber er möchte auch den Gesamtbetrag wissen. Und zwar sofort. Automatisch. Und überdies, sinnt er, müßte sich die Datenerfassung noch vereinfachen lassen. Wenn man zum Beispiel ...

Fortsetzung: Kapitel 34 - ein FRED-Programm erleichtert die Dateneingabe und die Auswertung

Die Kunst der Makros

25 Was ist ein Makro?

Ein Makro ist ein Abruf per Tastendruck. Sie drücken zwei Tasten gleichzeitig und damit wird abgerufen, was gleichsam »auf« dieser Taste gespeichert ist. Das kann ein Text sein, den Sie einmal schreiben - wenn Sie das Makro erstellen - und dann, wann immer Sie ihn benötigen, auf Tastendruck in Ihr Dokument einlesen: »Mit freundlichen Grüßen« beispielsweise in einen Brief. Ein Makro kann auch aus einem umfangreichen Programm bestehen, das dann, auf Tastendruck, abgearbeitet wird.

Hinweis: Sie benötigen für dieses Kapitel die Datei MAKRO.FW von der Beispieldiskette.

Ein Beispiel zu Demonstrationszwecken:

1. Holen Sie von der Übungsdiskette, die diesem Buch beiliegt, MAKRO.FW auf die Arbeitsfläche.
2. Bringen Sie den Cursor auf MAKRO.FW (Sie brauchen den Frame nicht zu öffnen!) und drücken Sie **F5** (*Neuberechnung*).
3. Halten Sie die **ALT**-Taste und die Taste **D** gleichzeitig gedrückt.

Tip: Arbeiten Sie stets nur mit einer Kopie von MAKRO. Kleine Änderungen an Programmen geschehen leicht und unabsichtlich, können aber fatale Folgen haben.

Dieses Makro öffnet einen neuen Frame und schreibt in ihn den Text, den Sie auf dem Bildschirm gesehen haben. Im Jargon von Framework heißt das: »Die Betätigung der Makro-Taste veranlaßt Framework, die zu dieser Taste definierte Makro-Formel auszuführen.«

Die Makro-Taste ist

- eine der 26 Buchstaben-Tasten (A - Z)
- eine der zehn Ziffern-Tasten oberhalb der numerischen Tastatur (0 - 9)
- eine der zehn Funktionstasten (**F1** - **F10**).

Insgesamt stehen also 46 Makro-Tasten zur Verfügung. Jede dieser 46 Tasten kann mit einer anderen Makro-Formel belegt werden.

Abgerufen werden die Makros, wie Sie gesehen haben, mit der Kombination aus der **ALT**-Taste und der betreffenden Makro-Taste. **ALT** sagt Framework, daß mit der Taste D nicht das Zeichen D geschrieben werden soll, sondern die Makro-Formel ausgeführt wird, die »auf« der Taste D gespeichert ist.

Abgelegt ist das kleine Demo-Makro in einer Makro-Bibliothek. Sie können beliebig viele Makro-Bibliotheken aufbauen und sich somit eine unendliche Zahl von Makros

erstellen. Eine Makro-Bibliothek hat Framework dabei: MACLIB.FW; eine weitere befindet sich auf der Übungsdiskette dieses Buches: MAKRO.FW - mit ihr haben Sie eben gearbeitet.

Beide Makro-Bibliotheken enthalten Makros - aber nicht die gleichen. MACLIB hat die zehn Funktionstasten sowie die Zifferntasten 1 bis 6 belegt. Diese Makros befinden sich nicht in MAKRO (aus Urheberrechtsgründen), indessen sind hier diese Makro-Tasten frei gelassen. Sie können demnach beide Makro-Bibliotheken zusammenlegen - wie, wird noch erklärt.

Damit ein Makro aus einer Bibliothek abgerufen werden kann, muß sich die Bibliothek, zum Beispiel MACLIB, auf der Arbeitsfläche befinden. Und sie muß zunächst mit F5 (*Neuberechnung*) aktiviert werden. Dann lauern die Makros abrufbereit.

Merke: Befinden sich mehrere Makro-Bibliotheken auf der Arbeitsfläche, ist diejenige benutzbar, die zuletzt neu berechnet worden ist.

Man muß Makros aber nicht in Bibliotheken sammeln. Sie können ebenso einzeln erstellt werden. Man kann dann ein ganzes Programm als Makro abrufen, ohne deswegen eine Makro-Bibliothek aktivieren zu müssen - in Kapitel 34 wird ein Beispiel dafür vorgestellt. Der Speicherkapazität wegen ist es ratsam, umfangreiche Programme nicht in die Makro-Bibliothek einzubinden.

26 Makros erstellen

Es gibt zwei Wege, Makros zu erstellen. Im einen Fall wird die Makro-Formel als Programm von Hand eingetragen. Im andern läßt man das von einem Framework-Hilfsprogramm erledigen.

Hinweis: Für dieses Kapitel benötigen Sie MACLIB.FW von der Framework-Hilfsprogrammdiskette.

Dieses Hilfsprogramm ist selber ein Makro, befindet sich in MACLIB und ist auf der Funktionstaste **[F2]** abgelegt. Dieses Makro, einmal aktiviert, speichert sämtliche Tastenanschläge und ordnet sie einer von Ihnen zu bestimmenden Taste zu.

1. Entfernen Sie MAKRO.FW mit *Weglegen* aus dem Menü *Laufwerk* von der Arbeitsfläche.
2. Laden Sie MACLIB.FW.
3. Aktivieren Sie MACLIB.FW mit **[F5]** (*Neuberechnung*).

Wir wollen zunächst ein Makro erstellen, das den Fettdruck einschaltet.

1. Drücken Sie **[ALT][F2]**.

Damit lösen Sie das Makro aus, das auf dieser Taste gespeichert ist und der Erstellung eines neuen Makros dient. Framework meldet in der Nachrichtenzeile:

[ALT] und Buchstabentaste drücken (Makrotaste für Makro-Aufruf)

Sie müssen Framework mitteilen, welcher Taste Ihr neu zu erstellendes Makro zugeteilt werden soll. Wir nehmen die Taste F dazu - für Fettdruck:

Tip: Wählen Sie immer eine Taste, mit der Sie das auszulösende Makro leicht identifizieren können.

2. Drücken Sie **[ALT]-F**.

Alle Tastenanschläge werden nunmehr gespeichert:

3. Öffnen Sie das Menü *Text* mit **[CTRL]-T**.
4. Aktivieren Sie den Fettdruck mit F.
5. Beenden Sie die Makro-Definition mit **[ALT]-F** - also nochmals mit der Tastenkombination, mit welcher das Makro aufgerufen wird.

Hinweis: Alle hier erstellten Makros befinden sich auch in der Makro-Bibliothek MAKRO auf der Übungsdiskette.

Probieren Sie's aus, ob's auch funktioniert:

1. Öffnen Sie einen (leeren) Text-Frame.
2. Bewegen Sie den Cursor in den Frame.
3. Drücken Sie **[ALT]-F** und schreiben Sie irgendetwas. Es müßte fett erscheinen.

Um dieses Makro in der Praxis nutzen zu können, brauchen Sie noch eines, das die Normalschrift wieder anschaltet. Das wollen wir indessen nicht auf die gleiche, bequeme Weise erstellen wie eben, sondern wir gehen den anderen Weg: Handarbeit. Wir schreiben dieses Makro direkt in die Makro-Bibliothek MACLIB.

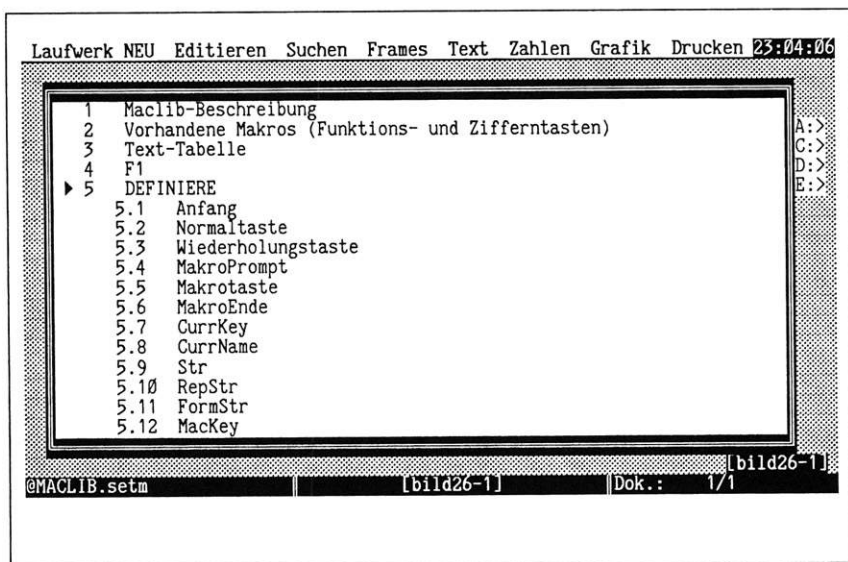


Bild 26.1: Die Makro-Bibliothek MACLIB

1. Öffnen Sie MACLIB. Eine Makro-Bibliothek ist ein Konzept-Frame (nähere Erläuterungen in Kapitel 28), und Sie sollten nötigenfalls mit **[F10]** (*Zeigen*) in die Konzept-Version umschalten.
2. Bringen Sie den Cursor in MACLIB auf die Zeile »38 N«. Das Makro für Normalschrift soll auf der Taste N abgelegt werden.
3. Öffnen Sie den Formelspeicher dieses Sub-Frames mit **[F2]** (*Formel editieren*) und bringen Sie ihn mit **[F9]** (*Zoom*) auf volle Bildschirmgröße - dann schreibt sich's leichter.

Merke: Makros sind Formeln und werden daher in den Formelspeicher »hinter« einen Frame geschrieben.

4. Schreiben Sie:

```
@performkeys("{CTRL-T}n")
```

Um die geschweifte Klammer links ({) einzugeben, halten Sie die **[ALT]**-Taste fest und drücken auf dem Ziffernblock die Tasten 1, 2 und 3. Die geschweifte Klammer rechts (}) erhalten Sie mit **[ALT]** und 1, 2 und 5.

5. Schließen Sie den Formelspeicher wieder mit **[ESC]**.

Tip: Sichern Sie die Makro-Bibliothek zwischendurch möglichst oft. Programme schreiben ist mühsam, und wenn das System abstürzt oder der Strom ausfällt, haben Sie sich die Mühe umsonst gemacht.

Sie haben jetzt das geschrieben, was sonst Framework schreibt, wenn Sie ein Makro über **[ALT][F2]** definieren.

Was Sie geschrieben haben, bedarf einiger Erläuterungen.

@performkeys ist ein FRED-Kommando und bedeutet, daß alle folgenden Tastaturangaben ausgeführt werden, als ob die entsprechenden Tasten gedrückt würden. In der spröden Sprache der Programmierer heißt das: Alle Zeichen werden als Tastatureingabe an Framework übergeben. (Man muß den Programmierer-Slang nicht sprechen, aber wenigstens so weit beherrschen, daß man versteht, was damit gemeint ist. Darum zitiere ich ihn hin und wieder.)

Diese Tastaturangaben müssen allesamt in runde Klammern eingeschlossen sein. Die von **@performkeys** zu schreibenden Zeichen wiederum müssen zwischen Anführungszeichen gesetzt werden. Ein Makro, das den Text »Mit freundlichen Grüßen« schreiben soll, sieht dann so aus:

```
@performkeys("Mit freundlichen Grüßen")
```

Ein Makro kann jedoch nicht nur schreibbaren Text beinhalten, jede Taste kann mit ihm ausgelöst werden. Das macht unser Makro ja: Es öffnet mit der Tastenkombination **[CTRL]-T** das Menü *Text*.

Diese Tastenkombination indessen soll von Framework nicht als die Zeichenfolge **[CTRL]-T** betrachtet werden, sondern als Funktionsaufruf. Damit Framework das auch merkt, müssen alle Bezeichnungen von Cursor-, Funktions- und sonstigen Sondertasten zwei Regeln folgen:

- sie stehen zwischen den geschweiften Klammern { und }
- für sie müssen bestimmte, von Framework verbindlich vorgeschriebene Bezeichnungen verwendet werden

Hinweis: Die Tastaturbezeichnungen sind im Framework-Referenz-Handbuch auf den Seiten 10-10 bis 10-13 aufgelistet. In Kapitel 36 wird ein Makro vorgestellt, das sich selbsttätig schreibt.

Einige Beispiele, welche Umschreibung Framework verlangt:

CONTROL-Taste	{CTRL}
Cursortaste links	{LEFTARROW}
END-Taste	{END}
Abwärts-Taste	{DNLEVEL}

Groß- oder Kleinschreibung (auch gemischt) ist unerheblich, ansonsten jedoch muß die vorgegebene Schreibweise eingehalten werden.

Damit ist die Syntax unseres Makros vollends leicht zu entschlüsseln: {CTRL-T} öffnet das Menü *Text*, dann wird mit **n** die Normalschrift aktiviert. Weil das ja keine Sonder-taste ist, sondern ein ganz normales Zeichen (auf welches das Menü anspricht, da es so programmiert ist), sind hier die geschweiften Klammern nicht notwendig.

Tip: Verwenden Sie in einem Makro immer die *CTRL*-Kommandos, wenn Sie ein Menü öffnen wollen. Sonst landen Sie womöglich im falschen.

Mit einiger Wahrscheinlichkeit werden Sie den »Klammeraffen« @ und mit ziemlicher Sicherheit die geschweiften Klammern auf Ihrer Tastatur vergeblich suchen. Man kann sie trotzdem schreiben: indem man die [ALT]-Taste gedrückt hält und mit dem Zehnerblock den ASCII-Code des Zeichens eingibt. Die Codes sind:

```
@    64
{    123
}    125
```

Tip: Statt des Klammeraffen @ erlaubt Framework auch das Paragraph-Zeichen §.

Indessen, das muß man sich ja merken. Besser, man benutzt ein Makro dazu - wenigstens für die geschweiften Klammern:

1. Gehen Sie mit dem Cursor in einen Text-Frame.
2. Drücken Sie [ALT][F2].
3. Drücken Sie [ALT]-K - das Makro wird auf der Taste K (für Klammer) abgelegt.
4. Schreiben Sie:
{ }
5. Bewegen Sie dann den Cursor mit der Steuertaste [←] um ein Zeichen nach links, so daß er auf der schließenden Klammer steht.
6. Beenden Sie die Makro-Definition mit [ALT]-K.

Den Rückschritt um ein Zeichen müßten Sie später, wenn Sie die Klammern benutzen, ohnehin machen, weil die Tastenbezeichnung ja zwischen die Klammern kommt - so erledigt ihn das Makro gleich mit.

Auch die Funktion **@performkeys** wird man häufig benutzen, und deshalb ist auch dafür ein Makro geschickt. Dies ist gleichzeitig ein Beispiel dafür, daß man (schon bestehende) Makros auch bei der Definition eines neuen Makros verwenden kann: Der Name der Funktion soll fett werden. Programmierer machen das gerne, um die Befehle von den Parametern zu trennen:

1. Gehen Sie mit dem Cursor in einen Text-Frame - es kann derselbe sein wie vorhin; Sie brauchen nur irgendetwas, wo Sie schreiben können.
2. Drücken Sie **[ALT] F2**.
3. Drücken Sie **[ALT]-P** - dieses Makro wird auf der Taste P gespeichert.
4. Schreiben Sie bzw. drücken Sie die angegebenen Tasten

```
[ALT]-F
@performkeys
[ALT]-N
( " " )
←
←
```

5. Beenden Sie die Makro-Definition mit **[ALT]-P**.

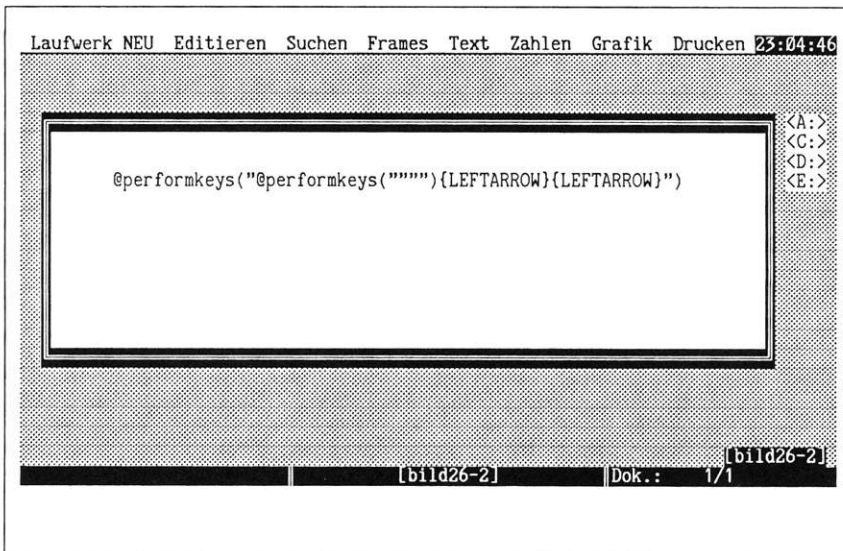


Bild 26.2: So sieht die Formel des **@performkeys**-Makros dann aus

Die runden Klammern und die Anführungszeichen werden mit aufgenommen, denn notwendigerweise sind sie in jeder **@performkeys**-Anweisung vorhanden. Der zweimalige Rückschritt bewirkt wiederum, daß der Cursor an der richtigen Schreibstelle steht.

Tip: Versehen Sie Programme (auch Makros sind Programme!) in der ersten Zeile mit einem Kommentar, der in Erinnerung ruft, was dieses Programm bewirkt. Solche Kommentare beginnen in der ersten Spalte mit einem Semikolon. Jede Kommentar-Zeile braucht das Semikolon.

@performkeys ist die häufigste Funktion, die in einem Makro verwendet wird, und wenn man mit ihr viele Tastenanschläge speichert, wird sie ziemlich verwirrend: eine endlos lange Zeile - im Höchstfall 32000 Zeichen lang (wenn im Arbeitsspeicher so viel Platz ist).

Der Übersichtlichkeit wegen ist es daher besser, die **@performkeys**-Formel auf mehrere Zeilen aufzuteilen, in Portionen, die logisch zusammengehören. Das sieht dann so aus:

```
@performkeys("{ALT-1}DEMOBRIEF{RETURN}"
& "{F4}{F4}{RETURN}{DNLEVEL}"
& "{CTRL-T}15{RETURN}"
& "{CTRL-T}r60{RETURN}"
& "{RETURN}{RETURN}{RETURN}{RETURN}{RETURN}{RETURN}"
& "{RETURN}{RETURN}{RETURN}{RETURN}{RETURN}{RETURN}"
& "<anrede>{RETURN}"
& "<vorname> <name>{RETURN}"
& "<straße> <nr.>{RETURN}{RETURN}"
& "<plz> <ort>{RETURN}"
& "{RETURN}{RETURN}{RETURN}{ALT-2}{RETURN}"
& "{RETURN}{RETURN}{RETURN}{RETURN}")
```

Hinweis: In Kapitel 38 wird besprochen, wie man ein Makro mit vielen gleichlautenden Tastaturbezeichnungen vereinfachen kann.

Das ist ein Makro aus dem Kapitel 29, dessen Inhalt jetzt nicht interessieren soll (vielleicht können Sie's schon selber entschlüsseln?), sondern allein dessen Schreibweise.

Wie Sie sehen, sind die einzelnen Teile - Zeilen in diesem Falle - durch das Zeichen **&** verbunden und stehen jeweils zwischen Anführungszeichen, weil es sich um Tastenbezeichnungen bzw. zu schreibende Zeichen handelt. (Genausogut können innerhalb einer **@performkeys**-Anweisung andere Funktionen aufgerufen werden, die dann ohne Anführungszeichen stehen.) Und weil das alles zusammen eine einzige Funktion ist, braucht - und darf - am Ende jeder Zeile kein Komma stehen.

Bisher sind wir davon ausgegangen, daß die Makros in einer Makro-Bibliothek abgelegt und dort abgerufen werden. Das ist jedoch keineswegs zwingend. Makros können im Formelspeicher jedes Frames oder jeder Zelle in einer Tabelle oder Datenbank sein.

Eine Demonstration am praktischen Beispiel:

1. Erstellen Sie einen Text-Frame.
2. Nennen Sie ihn TEST.
3. Schreiben Sie irgendetwas, und sei's nur ein Zeichen.

Zur Erinnerung: Soll ein Text-Frame einen Text (im Frame) und eine Formel (im Formelspeicher) enthalten, muß zuerst Text vorhanden sein (ein Zeichen genügt), ehe die Formel eingetragen wird. Andersrum geht's nicht.

4. Gehen Sie auf den Rand des Frames.
5. Öffnen Sie den Formelspeicher mit **[F2]** (*Formel editieren*).
6. Bringen Sie ihn auf volle Bildschirmgröße mit **[F9]** (*Zoom*).
7. Schreiben Sie (und benutzen Sie dazu die schon erstellten Makros):
`@setmacro({ALT-A},TEST),`
`@performkeys("Guten Tag!")`
8. Schließen Sie den Formelspeicher mit **[ESC]**.
9. Aktivieren Sie das Makro mit **[F5]** (*Neuberechnung*).
10. Gehen Sie in den Frame.
11. Drücken Sie **[ALT]-A**.

An der Cursorposition wird der Text »Guten Tag!« geschrieben.

Mit der Funktion **@setmacro** wird zunächst der Taste **[ALT]-A** ein Makro zugewiesen, welches, zweiter Parameter, im Frame TEST abgelegt ist. Die zweite Zeile ist dieses Makro: Es schreibt den Text. Und zwar nicht nur in diesen Frame, sondern in jeden - solange sich der Frame TEST auf der Arbeitsfläche befindet.

Andersrum: legt man ihn weg, funktioniert auch das Makro nicht mehr - eine Möglichkeit, es aufzuheben. Das ist auf diese Weise nicht immer erwünscht, deshalb geht's auch auf andere:

1. Öffnen Sie den Formelspeicher von TEST.
2. Fügen Sie der Formel in der zweiten Zeile ein Komma hinzu und sodann als dritte Zeile:
`@setmacro({ALT-A},)`
3. Drücken Sie **[F5]** (*Neuberechnung*).

Zur Erinnerung: Die einzelnen Funktionen in einer Formel werden durch Kommata getrennt.

Durch die bloße Angabe der Makro-Taste, ohne den Fundort der Makro-Formel, wird dieses Makro deaktiviert. Und wieder aktiviert, wenn diese dritte Zeile entfernt wird. Oder, eleganter, außer Kraft gesetzt wird:

1. Löschen Sie das Komma am Ende der zweiten Zeile.
2. Ersetzen Sie in der dritten Zeile den »Klammeraffen« @ durch ein Semikolon.

|| **Merke:** Eine Zeile, die mit einem Semikolon beginnt, wird von Framework als Kommentar betrachtet, nicht als Kommando.

So läßt sich ohne viel Tippen ein Makro aktivieren und deaktivieren.

Trotz dieses neuen, im Frame TEST definierten Makros sind alle anderen Makros, zum Beispiel in MACLIB, weiterhin aufzurufen - mit Ausnahme jenes, das die gleiche Tasten verwendet. Das paßt oft nicht - also setzt man an den Anfang des TEST-Makros einen Deaktivierungsbefehl (am Ende das Komma nicht vergessen!), der sämtliche vorhandenen Makros matt setzt:

```
@setmacro({ALL}),
```

27 Die Makro-Bibliothek MACLIB

Diese Makro-Bibliothek hat, wie schon erwähnt, einige Makro-Tasten vorbelegt mit Formeln, die nützliche Dienste leisten. Und zwar sind dies (in der ersten Zeile die Tasten, mit der sie abgerufen werden, und die Frame-Namen, unter denen sie gespeichert sind):

Tips: Schauen Sie sich die Formeln dieser Makros an und lernen Sie daraus. Wandeln Sie die Makros nach Ihrem Bedarf ab.

ALT **F1**

MACLIB-Hilfe

Zeigt eine Liste der vorhandenen Makros - also dieser hier. Ein normaler Text-Frame, der um die eigenen Makros erweitert werden kann.

ALT **F2**

DEFINIERE

Bekannt; damit erstellt man ein neues Makro.

ALT **F3**

GOTO

Sprung zu einem Frame, dessen Name in der Nachrichtenzeile erfragt wird. Muß sich auf der Arbeitsfläche befinden!

ALT **F4**

GOBACK

Nach GOTO der Rücksprung zum Ausgangsframe.

ALT **F5**

MERKEN

Ein markierter Text wird in einen Zwischenspeicher gestellt und kann dort mit **ALT** **F6** wieder abgeholt werden.

Merke: Der Zwischenspeicher ist ein Frame, der Text geht also nicht verloren, wenn der Computer abgeschaltet wird.

ALT **F6**

EINSCHIEBEN

Der mit **ALT** **F5** zwischengespeicherte Text wird an der Cursorposition eingelesen.

ALT **F7**

FREIER SPEICHER

Gibt in Bytes an, wieviel Platz noch im Arbeitsspeicher ist.

ALT F8**DOS-ZUGRIFF**

Öffnet die Tür zum Betriebssystem: Alle DOS-Kommandos können aufgerufen werden. Mit »exit« kehrt man zu Framework zurück.

ALT F9**TEXT-FRAME TEILEN**

Der Text von der Cursorposition an wird in einen neuen Frame geschrieben und im ursprünglichen Frame gelöscht.

ALT F10**MEMO/BRIEF**

Ein kleines, menügeführtes Programm, das ein Memo oder einen Brief nach Muster anlegt. Wird weiter unten näher erläutert.

ALT -1

Neuen Frame erstellen

Erspart den Weg über das Menü *Laufwerk*.

ALT -2

Tagesdatum eintragen

Trägt das Datum an der Cursorposition ein.

ALT -3

Uhrzeit eintragen

Übernimmt von der Systemuhr die Zeit und trägt sie an der Cursorposition ein.

ALT -4

Formel löschen

Löscht die Formel - bei einem Text-Frame im Formelspeicher des Frames, ansonsten an der Cursorposition.

ALT -5

Formel @np eintragen

Schreibt die Druckanweisung für »neue Seite« bzw. »Seitennummer« in den Formelspeicher des Frames, auch wenn sich der Cursor im Frame befindet.

ALT -6

Kopf-/Fußzeile eintragen

Schreibt in den Formelspeicher des Frames folgende Zeilen:

```
@hl("Kopf text links")
```

```
@hc("Kopf text zentriert")
```

```
@hr("Kopf text rechts")
```

```
@fl("Fuß text links")
```

```
@fc("Fuß text zentriert")
```

```
@fr("Fuß text rechts")
```

Tip: Drucken Sie sich die Liste der Makros im MACLIB-Sub-Frame »Vorhandene Makros« aus.



Bild 27.1: Das Hilfe-Makro **[ALT] [F1]**

Diese Makros sind schöne Beispiele dafür

- wie man oft gebrauchte Tastenfolgen (Frame erstellen, DOS-Zugriff) oder umständlich zu schreibende Anweisungen (freier Speicher, Tagesdatum) auf eine Taste reduzieren kann
- wie man Operationen definieren kann, für die Framework eigentlich keine Funktionen vorgesehen hat (Frame teilen, Merken/Einschieben)
- wie man recht komplexe Programme schreiben kann (Memo/Brief).

Sie sollten sich mal ein Stündchen Zeit nehmen und diese Makros ausprobieren. Insbesondere das Makro **[ALT] [F10]**, welches das Schreiben von Memos, Geschäfts- und Privatbriefen erleichtert.

Rufen Sie das Makro auf, erscheint in der Nachrichtenzeile ein Menü, mit dem Sie wählen - entweder mit den Cursortasten und anschließendem **[RETURN]** oder mit dem Anfangsbuchstaben des Menüpunktes.

Hinweis: In Kapitel 30 wird gezeigt, wie man ein vorhandenes Makro individuell abändert.

Jedesmal erstellt das Makro einen neuen Frame und füllt ihn mit vorbereitetem Text - welchen, ersehen Sie aus den Abbildungen 27.2 bis 27.3. Das Memo enthält zudem eine

zentrierte Fußzeile mit der Seitennummer. Der Cursor stoppt an der ersten Stelle, an der Sie eigenen Text eingeben.

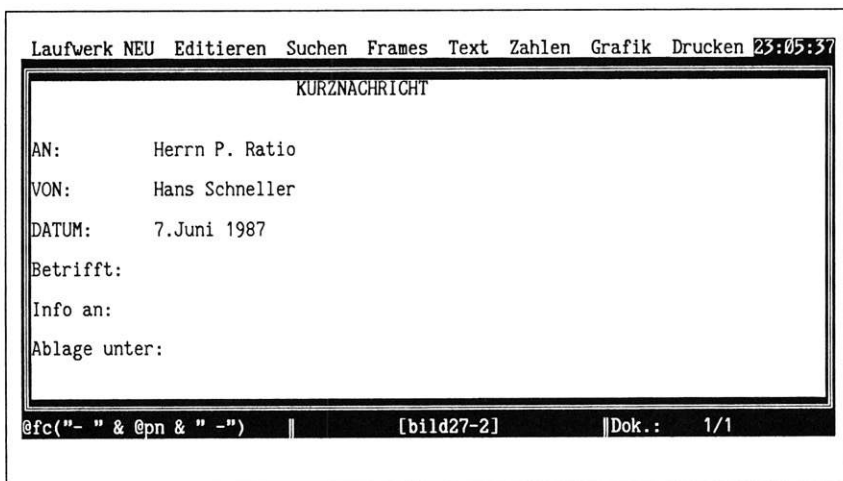


Bild 27.2: *Das Memo*

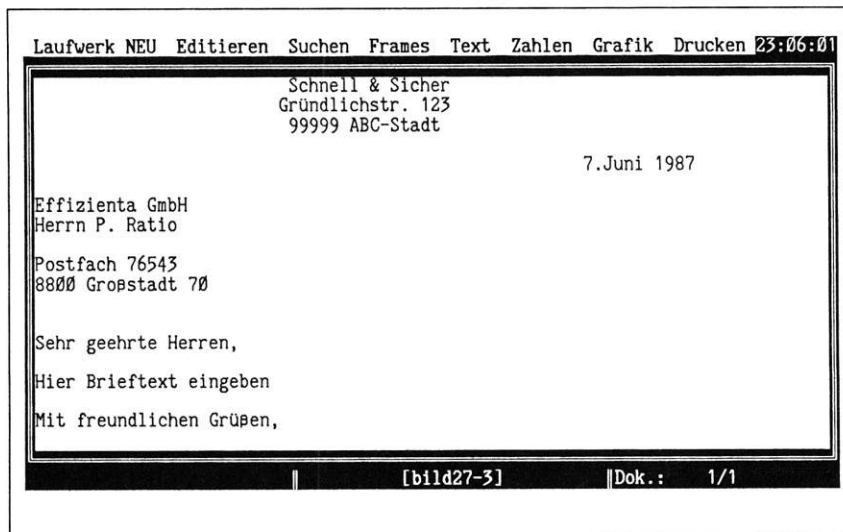


Bild 27.3: *Der Geschäftsbrief*

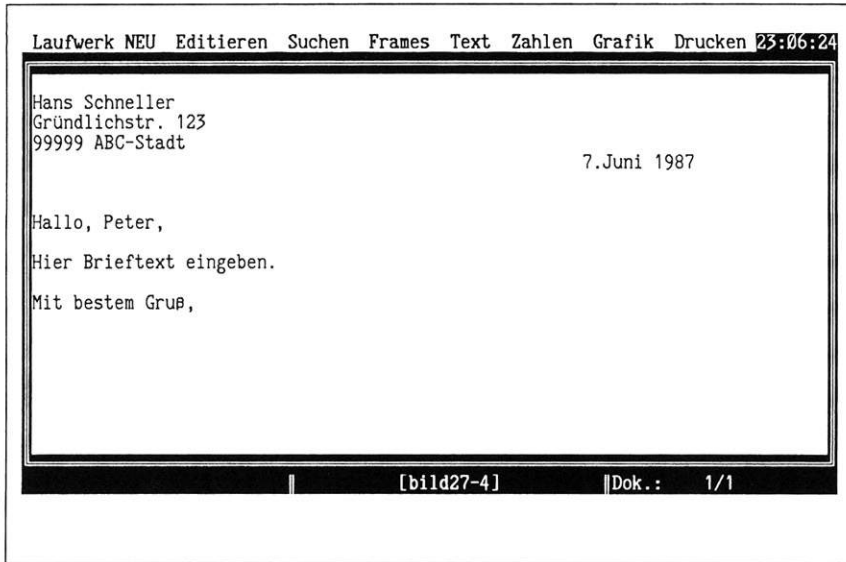


Bild 27.4: *Der Privatbrief*

Die Namen und Adressen holt sich das Makro aus dem MACLIB-Sub-Frame »Text-Tabelle« - ein Text-Frame, der sich editieren läßt und in den Sie Ihre Namen eintragen können. Die Mustertexte müssen Sie, so Sie dies wünschen, in den Makro-Formeln ändern:

Tip: Arbeiten Sie nur mit Kopien! Sie können dann jederzeit auf das originale Makro zurückgreifen.

1. Bringen Sie in MACLIB den Cursor auf den Frame »Memo«, »Geschäftsbrief« oder »Privatbrief«.
2. Drücken Sie **F2** (*Formel editieren*).
3. Bringen Sie mit **F9** (*Zoom*) den Formelbereich auf volle Bildschirmgröße.

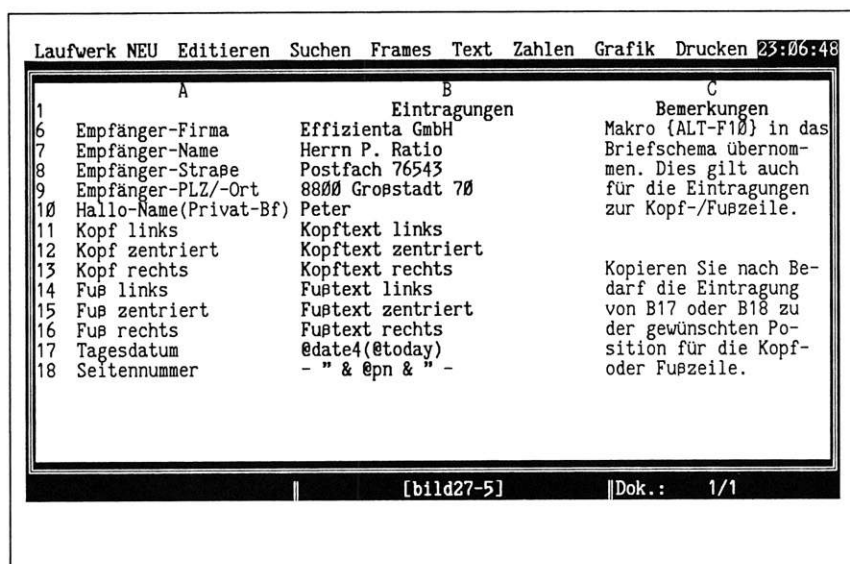


Bild 27.5: Der Sub-Frame »Text-Tabelle«, aus dem die Briefe ihre Daten holen

28 Eine eigene Makro-Bibliothek

Eine Makro-Bibliothek ist ein Konzept-Frame, der (maximal) 47 Sub-Frames enthält. 46 davon sind mit den Tastenbezeichnungen benannt: von **F1** bis **F10**, von 0 bis 9, von A bis Z. Sie enthalten in ihrem Hintergrund die Makro-Formeln. Der 47. und letzte Sub-Frame organisiert die Zuordnung der Makro-Formeln zu den Makro-Tasten.

In ihm, oder genauer: »hinter« ihm sind die notwendigen Funktionen einzutragen. Sie lauten beispielsweise:

```
@setmacro({ALT-F1}, MAKRO.F1)
```

Sie wissen es bereits, trotzdem nochmals: **@setmacro** ist das Framework-Kommando zur Makro-Zuweisung. Seine Parameter regeln das Nähere: die Formel der Makro-Taste **ALT F1** ist in dem Frame **MAKRO.F1** zu finden - wobei, in diesem Fall, **MAKRO** der Name des umfassenden Frames ist und **F1** der des Sub-Frames von **MAKRO**: eine Pfadangabe.

Der Name des umfassenden Frames ist beliebig - er kann **MACLIB** oder **MAKRO** sein oder ein ganz anderer. Die Sub-Frames können gleichfalls nach Wunsch benannt werden - aber praktischerweise wird man die Tastaturbezeichnungen verwenden. Und schließlich: Auch der Name des 47. Frames mit den **@setmacro**-Formeln ist frei wählbar. In **MACLIB** und **MAKRO** heißt er jeweils **SETM** - leicht zu entschlüsseln, warum?.

Tip: Kopieren Sie für eine neue Makro-Bibliothek eine bestehende, löschen Sie die Formeln und tauschen Sie mit der Ersetzen-Funktion im **@setmacro**-Frame den Namen des umfassenden Frames. Das erspart viel Tipparbeit.

Für jeden Sub-Frame, also für jede Makro-Taste muß im Zuweisungs-Frame ein entsprechender Eintrag erfolgen. Und außerdem braucht der umfassende Frame noch eine Formel, die den Zuweisungs-Frame aktiviert:

```
@MAKRO.SETM
```

MAKRO ist der Name des umfassenden Frames, **SETM** der des Zuweisungs-Frames, des 47. Sub-Frames also.

Wer Framework intensiv benutzt, wird sich zur Arbeitserleichterung bald viele Makros erstellen - mehr, als eine Makro-Bibliothek aufnehmen kann. Eine neue aufzubauen, ist kein Problem. Aber in ihr hätte man schon gern auch einige der alten Makros - zum Beispiel jene, mit denen man einen neuen Frame erstellt oder das Tagesdatum einliest.

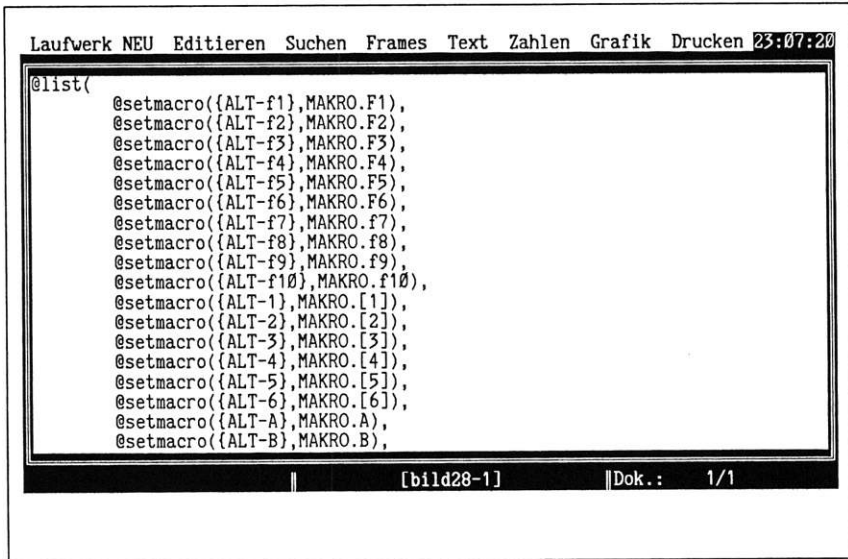


Bild 28.1: Der Makro-Zuweisungsframe der Makro-Bibliothek MAKRO

Tip: Sollen nur Teile einer Makro-Formel übernommen werden, sind die Makro-Funktionen »Merken« (**ALT** **F5**) und »Einschieben« (**ALT** **F6**) praktisch.

Sie zu übernehmen, ist einfach. Angenommen, Sie wollten die Speicherabfrage aus MACLIB auch in MAKRO benutzen:

1. Bringen Sie in MACLIB den Cursor auf den Sub-Frame »10 F7«.
2. Drücken Sie **F8** (*Kopieren*).
3. Bringen Sie den Cursor in MAKRO auf den Sub-Frame »7 F7«.
4. Drücken Sie **RETURN**.
5. Löschen Sie mit **DEL** den leeren F7-Frame.

Auf diese Weise können Sie auch die beiden Makro-Bibliotheken MACLIB und MAKRO zusammenlegen.

Tip: Legen Sie MACLIB und MAKRO zusammen. Sie haben dann Zugriff auf die Makros beider Bibliotheken.

29 Geschäftsbrieife (II)

oder wie sich die Brieife der Sekretärin Katharina Kröll alleine schreiben

Sie erfahren in diesem Kapitel:

- wie man Makros in der praktischen Arbeit verwendet
- wie man Makros editiert und überarbeitet
- wie man Makros verkettet
- wie man einen Brief aus Bausteinen zusammensetzt
- wie man ein einfaches Makro-Menü erstellt
- wie man u.a. folgende Funktionen einsetzt:
@performkeys
@menu

Sie benötigen:

- MAKRO.FW von der Beispieldiskette
- MACLIB.FW von der Framework-Dienstdiskette

Das Problem

Katharina Kröll hat in Kapitel 21 die Korrespondenz mit Framework schon ziemlich vereinfacht: Sie entnimmt Adressen und Namen aus einer Datenbank. Oder besser gesagt: Mit dem Hilfsprogramm MAILUTIL läßt sie das Framework machen.

Trotzdem macht die tägliche Korrespondenz in einem Büro noch mächtig viel Arbeit. Und lästige Arbeit zudem. Denn die meisten Brieife, die Katharina Kröll, die Sekretärin, zu schreiben hat, sind eine Art Formbrief. Sie bestehen weitgehend aus den gleichen Floskeln.

Dazu zählt nicht allein die Anrede. Im Geschäftsleben tauchen vielfach die gleichen Formulierungen auf: Da wird an eine unbezahlte Rechnung gemahnt, ein Auftrag wird bestätigt, eine Bestellung wird aufgegeben usw.

Wie schön wäre es, träumt Katharina Kröll manchmal, während der Kaffee aus der Maschine blubbert, wenn dies der Computer machte. Ganz alleine. Von sich aus. Vollautomatisch. Auf bloßen Gedankenwunsch sozusagen.

So weit ist's noch nicht. (Glücklicherweise. Weil sonst ja auch Katharina Kröll überflüssig wäre. Als Sekretärin zumindest.) Aber beinahe.

Die Lösung

Natürlich macht sich Katharina Kröll Frameworks Makro-Funktion zunutze. Was im Grunde den gleichen Effekt hat wie die Adressenübernahme aus einer Datenbank: Was häufig benötigt wird, muß nur einmal erfaßt werden und wird dann abgerufen.

Zur Erinnerung: Um eine Makro-Bibliothek wie MACLIB benutzen zu können, muß sie sich auf der Arbeitsfläche befinden und mit **[F5]** (Neuberechnung) aktiviert werden.

Wie so ein Makro gebastelt wird, hat Katharina Kröll im Prinzip zwar kapiert. Zur Sicherheit indessen übt sie es nochmals, Schritt für Schritt.

Hinweis: Wenn Sie nicht mitmachen wollen: Das Makro findet sich auch auf der Beispieldiskette in der Makro-Bibliothek MAKRO.

Was in jedem Brief benötigt wird, ist das Anschriftenfeld, das Datum, die Anrede und dergleichen. Man kann das alles zusammen in ein Makro packen - wie das zum Beispiel in dem MACLIB-Makro **[ALT][F10]** getan wird, wo zugleich eine Adresse (eine!) und eine Anrede (eine!) aus einer kleinen Tabelle entnommen wird.

Für die Bedürfnisse von Katharina Kröll reicht das nicht aus. Sie muß verschiedene Arten der Anrede berücksichtigen beispielsweise, und darum beschließt sie von vornherein, Adressenfeld und Anrede zu trennen: Sie erstellt sich Bausteine für jeden nur erdenklichen Zweck. Wenigstens ein Stück weit wollen wir ihr dabei über die Schulter schauen.

Das erste Makro füllt den Briefkopf: Adressenfeld mit den MAILUTIL-Variablen, Datum und dergleichen; überdies wird der Brief gleich ins rechte Seitenformat gebracht. Und überhaupt läßt sie alles, soweit's nur geht, von Framework machen und benutzt dazu auch bereits vorhandene Makros.

Schritt für Schritt, jeder einzelne Tastendruck (weil ja jeder gespeichert wird), sieht das so aus (nachdem Katharina Kröll MACLIB aktiviert hat):

1. Mit **[ALT][F2]** beginnt Katharina Kröll die Definition des neuen Makros.
2. Mit **[ALT]-A** teilt sie Framework mit, daß sie dieses Makro auf dieser Taste abulegen gedenkt.
3. Mit **[ALT]-1** erstellt Katharina Kröll einen neuen Frame. Soll er auch gleich einen Namen bekommen? Wahrscheinlich ist das in der Praxis nicht ratsam, denn Katharina Kröll wird ihn wohl jeweils individuell benennen, weil sie ihn nach dem Druck speichern und aufheben wird - falls Nachfragen kommen. Zu Übungszwecken macht's Katharina Kröll gleichwohl:
4. Sie tippt (womit sie automatisch in den Editiermodus wechselt):

DEMOBRIEF

5. Sie beschließt die Namenseingabe mit **[RETURN]**.
6. Sie bringt den Frame mit zweimal **[F4]** auf volle Bildschirmgröße und schließt das mit **[RETURN]** ab.
7. Sie bewegt den Cursor in den Frame mit **[NUM+]**.
8. Sie definiert den linken Rand: **[CTRL]-T-L 5 [RETURN]**.
9. Sie definiert den rechten Rand: **[CTRL]-T-R 60 [RETURN]**.

Tip: Korrigieren Sie Tipp- und sonstige Fehler während einer Makro-Definition unbesorgt. Das wird im Makro zwar auch gespeichert (kann man für interessante Effekte benutzen!), läßt sich aber später leicht wieder eliminieren.

10. Sie drückt zwölfmal **[RETURN]** und bewegt damit den Cursor um 12 Zeilen abwärts - hier sei die richtige Position für die Anschrift.
11. Sie schreibt:


```
<anrede>
<vorname> <name>
<straße> <nr.>

<plz> <ort>
```
12. Sie bewegt den Cursor um drei Zeilen nach unten - also dreimal **[RETURN]**- und bringt ihn etwa in Spalte 50, drückt also entsprechend oft die Leertaste.

Tip: Halten Sie bei einer Makro-Definition die Leertaste nicht gedrückt. Framework reagiert sehr träge, speichert aber gleichwohl jeden Tastendruck und schießt dann weit übers Ziel hinaus.

13. Mit **[ALT]-2** liest sie an dieser Stelle das Tagesdatum ein.
14. Sie bewegt den Cursor mit viermal **[RETURN]** um vier Zeilen nach unten - hierher kommt später die Anrede.
15. Sie beschließt die Makro-Definition mit erneutem **[ALT]-A**.

Natürlich probiert Katharina Kröll ihr neues Makro sogleich aus - ob's auch so funktioniert, wie sie sich das vorgestellt hat.

Hinweis: Wenn Sie das Makro nicht selber erstellt haben, sondern es der Beispiel-Bibliothek MAKRO entnehmen: Es funktioniert nur, wenn die Makros **[ALT]-1** und **[ALT]-2** aus MACLIB zur Verfügung stehen. Sie müssen also beide Bibliotheken zusammenlegen.

Und dann ändert sie's. Vielleicht hat sie sich bei der Definition des Makros verschrieben und mußte korrigieren. Vielleicht ändert sie die Angaben im Adressenfeld. Vielleicht hätte sie gern ein zweites Makro mit anderem Adressenfeld. Vielleicht fügt sie an der passenden Stelle noch »Unser Zeichen: ruk/kk« oder dergleichen ein. In all diesen Fällen ist es einfacher, das bestehende Makro zu editieren (oder zu kopieren) und zu

überarbeiten, als es nochmal neu zu erstellen. Wobei sich von Fall zu Fall andere Vorgehensweisen anbieten.

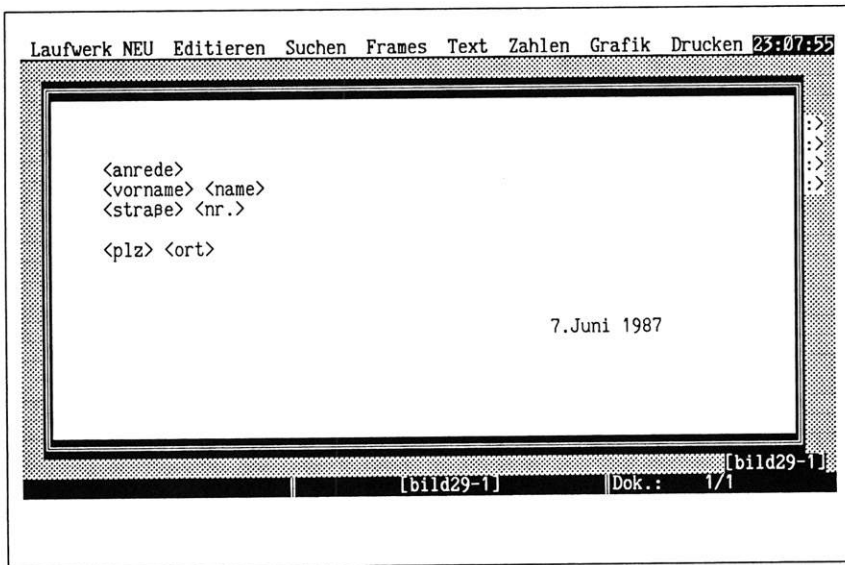


Bild 29.1: Dies löst das Makro **[ALT]-A** aus

Tip: Makros, die viele Cursorbewegungen oder Menü-Zugriffe enthalten, erstellt man leichter mit dem »Definiere«-Makro **[ALT][F2]** aus MACLIB. Sonst ist es arg viel (und arg viel verwirrende) Schreibaarbeit.

In den Fällen 1 und 2, also bei der Korrektur eines Makros, macht Katharina Kröll folgendes:

1. Sie öffnet die Makro-Bibliothek.
2. Sie bringt den Cursor auf den Sub-Frame, der das Makro enthält, in unserem Beispiel also auf A.
3. Sie geht in den Formelbereich mit **[F2]** (*Formel editieren*) und bringt ihn mit **[F9]** (*Zoom*) auf volle Bildschirmgröße.

Dann kann sie ihre Änderungen vornehmen - das Makro sieht zwar etwas verwirrend aus, aber daran gewöhnt man sich.

Zur Erinnerung: Wie man ein solches Makro entwirrt, wurde bereits in Kapitel 26 besprochen.

Zu einem zweiten Makro, im Prinzip gleich, doch mit geändertem Text (Fall 3), kommt man am schnellsten mit einer Kopie:

1. Katharina Kröll markiert das erste Makro.
2. Sie drückt **[F8]** (*Kopieren*).
3. Sie bringt den Cursor auf den Sub-Frame, den sie als Makro-Taste für den zweiten Frame ausersuchen hat.
4. Sie drückt **[RETURN]**.
5. Sie gibt dieser Kopie den richtigen Frame-Namen. Er ist jetzt zweimal vorhanden, also muß sie den leeren Frame löschen.
6. Sie editiert ihre Makro-Kopie und überarbeitet sie.

Für den vierten Fall - in das erste Makro wird etwas eingefügt - hat sich Katharina Kröll etwas Raffiniertes ausgedacht. Weil's ziemlich mühsam ist, viele Tastaturbezeichnungen zu schreiben, weil's schwer abzuschätzen ist, wieviele Leerzeichen und Leerzeilen man braucht, und weil Katharina Kröll zu faul ist, nochmal zu schreiben, was sie schon geschrieben hat.

|| Hinweis: In Kapitel 36 wird ein Makro vorgestellt, das die Tastaturbezeichnungen schreibt.

Dieses jetzt neu entstehende Makro ist übrigens nicht auf der Beispieldiskette enthalten; ich beschreibe es trotzdem recht ausführlich, weil man die Methode vielfach anwenden kann und weil hierbei auch einige der in Framework mitgelieferten Makros benutzt werden.

Für die Operation braucht Katharina Kröll insgesamt drei Makros. Zum besseren Verständnis: das Original-Makro sei **[ALT]-A**, das neue Makro **[ALT]-B**, sodann braucht Katharina Kröll noch ein Makro als Zwischenablage, das sei **[ALT]-X**.

1. Katharina Kröll editiert ihr Makro **[ALT]-A** und markiert in ihm mit **[F6]** (*Auswahl*) bis zu der Stelle, zu der sie es übernehmen will.
2. Mit dem MACLIB-Makro **[ALT][F5]** kopiert sie diesen Teil in einen Puffer.
3. Sie editiert das neue, noch leere Makro **[ALT]-B** (volle Bildschirmgröße!)

|| Tip: Halten Sie sich stets eine Makro-Taste frei als Zwischenablage.

4. Sie liest mit dem MACLIB-Makro **[ALT][F6]** den zwischengespeicherten Textteil ein, wird vermutlich die Abführung und die schließende Klammer noch tippen müssen, die **@performkeys** verlangt, und schließt den Formelbereich wieder - und am besten auch MACLIB.
5. Jetzt beginnt sie mit der Definition des Zwischenablage-Makros **[ALT]-X** und ruft dabei als erstes das Makro **[ALT]-B** auf. Dieses wird ausgeführt, der Brief also erstellt und gefüllt - bis zu jener Stelle eben, die Katharina Kröll aus dem ersten Makro kopiert hat.

6. Sie gibt nun ihre Ergänzungen und Änderungen ein und beendet dann die Makro-Definition.
7. Sie öffnet MACLIB und editiert das Zwischenablage-Makro **[ALT]-X** und weiß jetzt, was diese ganze umständlich erscheinende Operation soll: der aus **[ALT]-A** in **[ALT]-B** übernommene Teil steht dort nicht Taste für Taste, sondern als **{ALT-B}**.
8. Sie kann nun exakt die Stelle bestimmen, an welcher der zweite Teil des Makros beginnt, diesen markieren, in den Puffer kopieren und dann in **[ALT]-B** einlesen und dieses Makro somit vervollständigen.

Bei einem so kleinen Makro mutet das in der Tat verzwickelt an, hilft bei umfangreicheren Definitionen indessen Tipparbeit sparen.

Das will sich Katharina Kröll ja ohnehin mit den Makros - und erstellt sich demzufolge, ganz nach Bedarf, eine Vielzahl von Makros mit jenen Floskeln, die sie in ihren Briefen häufig benötigt.

Tip: Sammeln Sie Makros aus einem Arbeitsbereich in jeweils einer eigenen Makro-Bibliothek.

Aber freilich: Sie muß sich ja alle Makro-Tasten merken - oder, wenn sie die Liste in MACLIB vervollständigt, wenigstens immer nachschauen.

Man soll sich die Arbeit erleichtern, wo's nur geht; und Framework unterstützt dies berechnete Begehren und ermöglicht, eigene Menüs aufzubauen.

Ein solches schreibt jetzt Katharina Kröll aber nicht, sondern sie bedient sich des Makros **[ALT]-B** (für »Brieffloskeln«) in der Bibliothek MAKRO auf der Beispieldiskette (das sie, wenn sie will, ja auch in MACLIB kopieren kann).

1. Sie bringt den Cursor in einen Text-Frame.
2. Sie ruft das Makro **[ALT]-B** auf.
3. Sie sieht in der Nachrichtenzeile eine Menüleiste.
4. Mit den Cursortasten wählt sie einen Menüpunkt aus und drückt anschließend **[RETURN]** und harrt der Dinge, die da kommen werden.

Die Menüpunkte sind auch, und in der Praxis wohl bequemer, mit ihren Anfangsbuchstaben aufzurufen - und insofern ist dieses Menü ziemlich ungeschickt, weil zwei Menüpunkte mit dem gleichen Buchstaben beginnen.

Solch ein Menü ist leicht zusammenzustellen. Es besteht aus einem umfassenden Frame und dessen Sub-Frames. Der umfassende Frame ist sozusagen der Mantel, die Sub-Frames sind die Menüpunkte. Die Namen der Frames sind die Namen der Menüpunkte.

Jeder Frame enthält natürlich noch eine Formel, die ausgeführt wird, sobald Katharina Kröll den entsprechenden Menüpunkt aktiviert.

Zur Erinnerung: Formeln werden stets im Formelspeicher »hinter« dem Frame eingetragen, wenn der Cursor sich also auf dem Rand des Frames befindet, nicht im Frame selbst.

Die Formel des umfassenden Frames MACLIB.B ist ganz kurz:

```
@menu(B)
```

Das ist die Framework-Funktion, mit der ein Menü ins Leben gerufen wird; in Klammern steht als Parameter der Name des Frames, dessen Sub-Frames das Menü bilden. In diesem Fall verweist der Frame auf sich selber - trotzdem kann der Frame-Name nicht weggelassen werden.

Die Formeln der Sub-Frames haben jeweils den gleichen Aufbau, nur wechselnden Text:

```
Maclib.B.MFG:
```

```
@performkeys("Mit freundlichen  
Grüßen{return}{return}{return}Nikolaus Niembsch von  
Strehlenau{return}Vorstandsvorsitzender"),  
@quitmenu
```

```
Maclib.B.SgH:
```

```
@performkeys("Sehr geehrte Herren,{return}{return}{return}"),  
@quitmenu
```

```
Maclib.B.[SgD+H]:
```

```
@performkeys("Sehr geehrte Damen und Her-  
ren,{return}{return}{return}"),  
@quitmenu
```

@performkeys ist bekannt: in Klammern der Text, der geschrieben wird, wenn der Menüpunkt aktiviert wird.

@quitmenu veranlaßt Framework, das Menü zu verlassen, sobald der betreffende Menüpunkt ausgeführt ist.

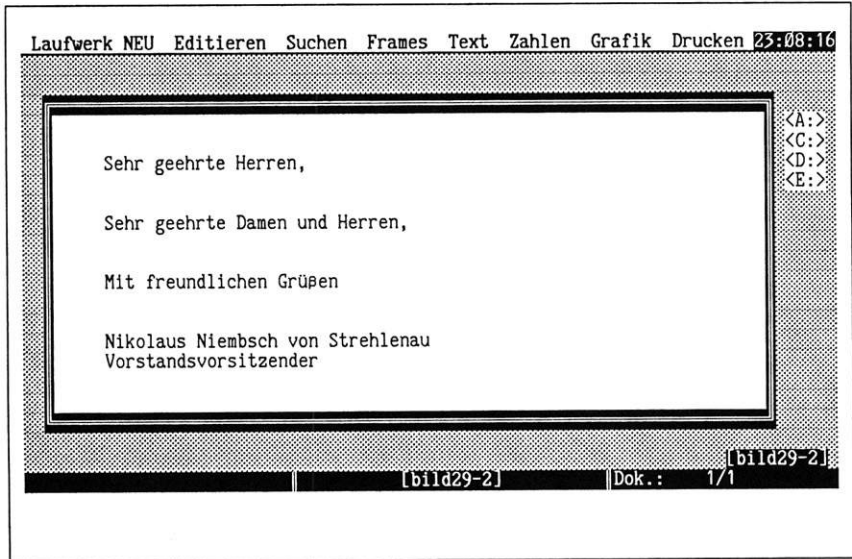


Bild 29.2: Beispiele für ein Briefloskel-Makro

Auch hier übrigens kann man, wie in den normalen Framework-Menüs, mit **[ESC]** jederzeit vorzeitig aussteigen. Das ist ja auch ein normales Framework-Menü, nur selbst programmiert. Und es ist horizontal angeordnet statt vertikal.

Das indessen kann Katharina Kröll selber bestimmen:

- Ist der umfassende Frame numeriert, erscheint das Menü horizontal in der Nachrichtenzeile.
- Wird die Numerierung unterdrückt (geht ja über das Menü *Frames*, ist ja ein Konzept-Frame), erscheint das Menü vertikal und bildschirmfüllend.

Für die Zwecke der Katharina Kröll ist ohne Zweifel die horizontale Anordnung geeigneter. (Die andere Form wird den Dichter Jonathan Kraft in Kapitel 34 beschäftigen.)

Somit kann Katharina Kröll ihr Menü um weitere, häufig gebrauchte Brieffloskeln erweitern - soweit der Platz in der Nachrichtenzeile eben reicht. Und weil sie sich auch für jeden erdenklichen Zweck Makros bastelt, schreiben sich ihre Briefe tatsächlich fast alleine. Dem Himmel sei Dank, daß auch der Computer wenigstens noch einen Menschen braucht, der das richtige Knöpfchen drückt.

30 Geschäftsbericht (II)

Sie erfahren in diesem Kapitel:

- wie man Arbeitsabläufe mit einem Makro automatisiert
- wie man mit Hilfe eines Makros eine Kurznachricht generiert
- wie man mit einem Makro eine Tabelle auswertet
- wie man u.a folgende Funktionen einsetzt:

@prompt

@nextkey

@vlookup

@local

@currency

@decimal

@max

@min

Sie benötigen:

- TABELLE.FW und TAB2.FW von der Beispieldiskette
- MAKRO.FW von der Beispieldiskette

Das Problem

Sachbearbeiter Herwig Moritz, aus den Kapiteln 19 und 22 bestens bekannt, hat, unter der Hand sei's verraten, mit seinem Geschäftsbericht Eindruck gemacht. Und das hat sich ausgezahlt. Der anerkennende Händedruck des Chefs gibt zu der berechtigten Hoffnung Anlaß, daß es Herwig Moritz noch weit bringen wird.

Aber offensichtlich hat er den Chef auf den Geschmack gebracht. Läßt er doch ausrichten, Herwig Moritz möge ihm sofort (und sofort heißt: jetzt, auf der Stelle) die neueste Umsatzbilanz herüberreichen.

Sachbearbeiter Herwig Moritz hat ja in seiner Tabelle - der kluge Mann baut vor - schon die Rubriken »Summe« und »Durchschnitt« eingetragen und mit den entsprechenden Formeln versehen. Er braucht also nur die Daten herausuchen und in die Chefvorlage übertragen.

Dank der Frame-Technik kann sich Herwig Moritz die Sache einfach machen: Er erstellt einen Text-Frame, holt sich zusätzlich die Umsatz-Tabelle auf den Bildschirm, und kann nun die Zahlen in sein Memo an den Chef kopieren.

Das jedoch, überlegt Herwig Moritz, auch auf den Geschmack gekommen, müßte sich doch automatisieren lassen. Was ebenso praktisch wie vorausdenkend wäre. Denn derartige Anfragen des Chefs werden wohl in Zukunft häufig kommen.

Die Lösung

Herwig Moritz bedient sich eines Makros - der schnellste Weg für häufig benötigte Befehlsformeln. Er hat es auf dem Buchstaben M abgelegt - M für Memo -, braucht nur **[ALT]-M** zu drücken und kann sich sodann genüßlich zurücklegen.

Zur Erinnerung: Um die Makros zu aktivieren, muß die Makro-Bibliothek erst neu berechnet werden.

Dieses Makro sieht, ausgeschrieben, so aus:

```
@prompt("Diskette mit TABELLE und TAB2 im Laufwerk? - Beliebige
Taste setzt fort"),
@nextkey,
@eraseprompt,
@echo(#OFF),
@performkeys("{ctrl-L}hTABELLE.FW{return}{return}"),
@performkeys("{ctrl-L}hTAB2.fw{return}{dnlevel}"
    & "{ctrl-home}{F6}{ctrl-dnarrow}{return}{ctrl-s}v{uplevel}"),
@performkeys("{ctrl-n}f{f4}{f4}{return}Memo{return}{dnlevel}"),
@performkeys("{ctrl-T}l5{return}
{ctrl-T}r55{return}{return}{return}"),
@echo(#ON),
@performkeys("{ctrl-t}f{ctrl-t}z"
    & "Saldo zum " & @date4(@today) & "{return}{return}{return}"
    & "{ctrl-t}a{ctrl-t}n"),
@performkeys(
    "Umsatz: " & @currency(TABELLE.Summe.Summe Umsatz,0) &
    "{return}{return}" &
    "Gewinn: " & @currency(TABELLE.Summe.[Gewinn (DM)],0) &
    "{return}{return}" &
    " " & @decimal(TABELLE.Summe.[Gewinn (%)])) &
    "%" & "{return}{return}{return}{return}"),
@performkeys(
    "schlechtester Monat: " &
    @vlookup(@min(TABELLE.c20:TABELLE.f20),TAB2.a1:TAB2.b4,1)
    & " (" & @currency(@min(TABELLE.c20:TABELLE.f20),0)
    & "){return}{return}" &
    "bester Monat: " &
    @vlookup(@max(TABELLE.c20:TABELLE.f20),TAB2.a1:TAB2.b4,1)
    & " (" & @currency(@max(TABELLE.c20:TABELLE.f20),0)
    & "){return}{return}"),
@performkeys("{return}{return}" &
    "Sachbearbeiter: Herwig Moritz" & "{return}")
```


■ **Zur Erinnerung:** Tastenanschläge werden in geschweifte Klammern eingeschlossen.

Das ist schon ein richtiges kleines Programm - und benutzt übrigens Teile von Frameworks *Memo*-Makro aus MACLIB: Herwig Moritz hat es sich kopiert und dann abgeändert. Allerdings beträchtlich.

■ **Tip:** Nehmen Sie, soweit es geht, schon vorhandene Makros als Grundlage. Sie ersparen sich Tipparbeit und vermeiden Fehler.

Als erstes gibt das Makro mit **@prompt** eine Meldung in der Nachrichtenzeile aus - eine kleine Erinnerung, daß zwei benötigte Dateien auf Diskette vorhanden sein müssen, wenngleich nicht auf der Arbeitsfläche: Das Makro lädt sie.

Löscht aber zuvor die Nachrichtenzeile mit **@eraseprompt** wieder. **@nextkey**, damit verbunden, wartet, wie in der Meldung angekündigt, auf einen beliebigen Tastendruck und führt dann die folgende Funktion aus - eben das Löschen der Nachricht.

■ **Merke:** **@nextkey** mit einem Sekunden-Wert in Klammern wartet entsprechend lang auf einen Tastendruck und fährt andernfalls im Programm fort. Ohne Wert ist die Wartezeit unbegrenzt.

Sodann werden die beiden Frames **TABELLE** und **TAB2** geladen, **TABELLE** wird gleichzeitig geschlossen und auf dem Dokumentenstapel abgelegt - Herwig Moritz geht davon aus, daß er sie stets geöffnet speichert.

TAB2 ist eine kleine Hilfstabelle. Sie übernimmt aus **TABELLE** die monatlichen Gewinn-Werte - warum sie vonnöten ist, wird gleich erläutert. Vorerst einmal bewegt Herwig Moritz den Cursor mit **{dnlevel}** in diesen Frame hinein und läßt ihn mit **{ctrl-home}** in die erste Zelle links oben springen - eine Vorsichtsmaßnahme, denn er markiert die gesamte erste Spalte und weiß nicht, wo der Cursor zuvor stand.

Er markiert die Spalte, um sie anschließend aufsteigend sortieren zu können - **{ctrl-s}** öffnet das Menü *Suchen* und aktiviert die Funktion *Vorwärts (aufsteig.) sortieren*. Sortiert werden kann in einer Tabelle genauso gut wie in einer Datenbank, mit den gleichen Effekten: Die Zellen in der markierten Spalte nehmen gleichsam ihre Zeilen huckepack und ordnen sie neu an.

Wichtig ist, daß nun der Cursor mit **{uplevel}** wieder aus dem Frame **TAB2** herausbewegt wird, denn jetzt richtet Herwig Moritz einen neuen Text-Frame ein mit **{ctrl-n}f** - was nicht ginge, wenn der Cursor irgendwo innerhalb eines Frames wäre (es sei denn, innerhalb eines umfassenden Frames).

Mit **{f4}{f4}** wird zweimal die Funktionstaste **F4** (*Größe*) angesprochen, der Frame damit auf Bildschirmgröße gebracht. **{return}** bringt den Cursor in die Editierzeile, wo der Name des Frames, »Memo«, eingetragen wird. Dann wird mit **{ctrl-T}** das Menü *Text* angesprochen und der linke Rand auf 5, der rechte auf 55 gesetzt.

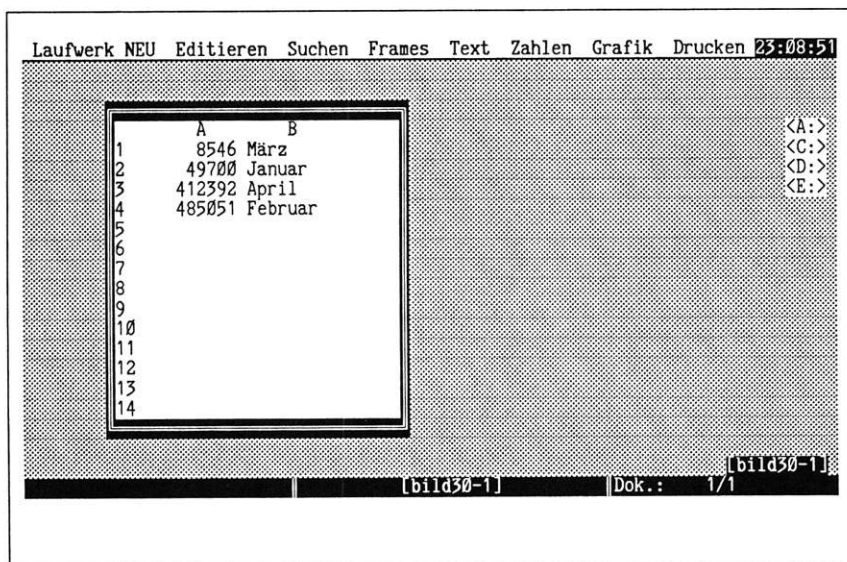


Bild 30.1: Die (Hilfs-)Tabelle TAB2

Von all dem hat Herwig Moritz noch nichts zu sehen bekommen, dank der Anweisung `@echo(#OFF)` in Zeile 4 des Programmes. Sie schaltet bis zur jetzt folgenden `@echo(#ON)`-Formel die Bildschirmdarstellung ab: Die Operationen verlaufen unsichtbar im Hintergrund.

Tip: Fügen Sie `@echo(#OFF)` zum Schluß in ein Programm ein. Beim Testen sehen Sie andernfalls mögliche Fehler nicht, die sonst sofort ins Auge springen.

Das Folgende hingegen möchte Herwig Moritz unbedingt mitverfolgen: Er findet's immer so schön, wenn Framework schreibt, ohne daß er die Tasten berührt.

Geschrieben wird eine Überschrift, und zwar gleich fett und zentriert: »Saldo zum« - zum heutigen Tag.

Herwig Moritz hat die Auswahl unter verschiedenen Formen der Datumsangabe. `@date4` ist die ausführliche Form: Tag, Monat, Jahr jeweils ausgeschrieben. Die Funktion `@today` sorgt dafür, daß das Datum von der Systemuhr übernommen wird. Hernach setzt Herwig Moritz die Ausrichtung auf linksbündig und zurück auf die Normal-schrift.

Tip: Das Tagesdatum läßt sich auch mit dem Makro `[ALT]-2` aus MACLIB einsetzen.

Dann erstellt sich Herwig Moritz die nötigen vorgestanzten Rubriken für Umsatz und Gewinn. Er trägt auch mit Leerzeichen den Abstand zu den Daten ein, so daß das Memo sich übersichtlich präsentiert.

Jetzt muß er nur noch die Daten aus der Tabelle holen und gibt dazu den Namen der Tabelle und die Koordinaten der Felder an. Er hat die Koordinaten ausgeschrieben, aus einem einfachen Grund: Es könnte sein, daß die Tabelle erweitert wird, so daß die bloße Angabe der Spalte und Zeile dann das falsche Feld ergibt; die Spaltenbezeichnung »Summe« und die Zeilenbezeichnung »Summe Umsatz« hingegen wird bleiben - darauf achtet Herwig Moritz schon.

Zur Erinnerung: In @performkeys-Formeln stehen Tastaturbezeichnungen und Texte (auch Klammern und dergleichen!) zwischen Anführungszeichen, Funktionen hingegen nicht.

Den der Tabelle zu entnehmenden Daten hat Herwig Moritz Formatieranweisungen vorangestellt:

- @currency versteht die Zahl mit dem Währungssymbol, DM also; der Parameter dazu legt die Anzahl der Nachkommastellen fest - weil er hier 0 lautet, gibt es keine
- @decimal gibt eine Dezimalzahl aus, definiert wiederum durch den Parameter mit diesmal zwei Nachkommastellen

Laufwerk NEU Editieren Suchen Frames Text Zahlen Grafik Drucken 23:09:33

Saldo zum 8.Juni 1987

Umsatz: 4.369.561 DM

Gewinn: 955.689 DM

= 21,87 %

schlechterster Monat: März (8.546 DM)

bester Monat: Februar (485.051 DM)

Sachbearbeiter: Herwig Moritz

[bild30-2] ||Dok.: 1/1

Bild 30.2: Das fertige Memo

Das war ja noch relativ einfach, und auch nicht eben schwierig ist es, in der Tabelle den niedrigsten und höchsten Wert ausfindig zu machen:

```
@min(TABELLE.c20:TABELLE.f20)  
@max(TABELLE.c20:TABELLE.f20)
```

Beide Funktionen brauchen natürlich den Bereich in der Tabelle (zur Abwechslung mal mit den Koordinaten angegeben, dessen minimalen und maximalen Wert sie ermitteln sollen.

Allein, was nützt der Wert als solcher, wenn nicht zugleich der Monat mitgeteilt wird, welcher derart aus der Reihe tanzt? Dafür greift Herwig Moritz zu einer jener Framework-Funktionen, die eine Tabelle nach einem bestimmten Wert durchsuchen. Und in die Trickkiste.

Die Funktion lautet in ihrer allgemeinen Form:

```
@vlookup(wert,bereich,abstand)
```

@vlookup sucht in der ersten Spalte des definierten Bereiches nach dem angegebenen Wert. Ist er gefunden worden, wandert **@vlookup** in der gleichen Zeile um soviel Spalten nach rechts, wie als Abstand festgelegt wird, ermittelt den Wert dieser Zelle und gibt ihn aus - in diesem Fall mit der **@performkeys**-Anweisung in Herwig Moritz' Memo-Frame. (Eine analoge Funktion dazu, nur auf Zeilen statt Spalten bezogen, ist **@hlookup**.)

Merke: Der definierte Bereich in **@vlookup** muß auch die Spalte umfassen, in die gewandert werden soll.

Allerdings funktionieren **@vlookup** und **@hlookup** nur, wenn die Werte in der Ausgangsspalte bzw. Ausgangszeile aufsteigend angeordnet sind - was in TABELLE nicht der Fall ist. Unter Umständen kann man eine Tabelle von vornherein so anlegen, daß diese Forderung berücksichtigt wird. Oder wendet den Trick von Herwig Moritz' Hilfstabelle TAB2 an. Weil sie lediglich die monatlichen Gewinne enthält, kann er sie sortieren, ohne daß die gesamte Tabelle durcheinander gerät, und hat er sie sortieren lassen, sind die Werte in der richtigen Reihenfolge.

Das Makro sucht also in der ersten Spalte von TAB2 nach dem niedrigsten bzw. höchsten Wert, wandert dann eine Spalte nach rechts, findet dort den Monatsnamen und gibt den zurück. Womit Herwig Moritz sein Memo vervollständigt hat. So schnell hat noch kein Sachbearbeiter eine Forderung des Chefs erfüllen können.

Oder doch? Zumindest hätte sich Herwig Moritz bei der Definition seines Makros etliche Schreibarbeit ersparen können - solche Formeln mit ihren Bezügen, bei denen man so höllisch aufpassen muß, nehmen ja doch einigen Umfang an und wiederholen sich gelegentlich.

Das kann man umgehen, indem man lokale Variable definiert - lokal deshalb, weil sie nur für dieses eine Programm bzw. Makro gelten. Dann sähe der zweite Teil des Makros so aus:

```
@local(wert1,wert2),  
wert1:=@min(TABELLE.c20:TABELLE.f20),  
wert2:=@max(TABELLE.c20:TABELLE.f20),  
@performkeys(  
"schlechtester Monat:      " & @vlookup(wert1,TAB2.a1:TAB2.b4,1)  
& " (" & @currency(wert1,0) & "){return}{return}" &  
"bester Monat:            " & @vlookup(wert2,TAB2.a1:TAB2.b4,1)  
& " (" & @currency(wert2,0) & "){return}{return}"),
```

Mit **@local** werden die Variablen definiert (deren Namen beliebig sein können), dann muß noch vereinbart werden, was sie bedeuten sollen. Dafür gibt es den Zuweisungsoperator **:=**. Wann immer Herwig Moritz den so bestimmten Wert benötigt, braucht er lediglich den Namen der Variablen zu schreiben - ohne Anführungszeichen, denn nicht diese Zeichenfolge soll ausgegeben werden, sondern der Wert.

|| **Zur Erinnerung:** Statt **:=** kann auch die Funktion **@set** verwendet werden - siehe Kapitel 24.

FRED - Spaß am Program- mieren

31 Was ist FRED?

FRED heißt ausgeschrieben *F*Rames *E*ditor und ist die Programmiersprache von Framework. Angst vorm Programmieren? Braucht niemand zu haben. Und vor allem Sie nicht, wenn Sie in diesem Buch bis hierher gekommen sind. Sie haben schon programmiert - ohne sich dessen wahrscheinlich bewußt zu sein.

Als unterdessen erfahrener Framework-Anwender wissen Sie, wie man in einer Tabelle mehrere Zellen addiert. Zum Beispiel mit dieser Formel:

```
@sum(A1:A16)
```

@sum ist eine FRED-Funktion. Jede Formel besteht aus FRED-Funktionen. Jede Formel ist demnach ein FRED-Programm. Also haben Sie bereits programmiert. Oft.

Auch ein Makro ist nichts anderes als ein FRED-Programm - ein schon etwas komplexeres, weil es mehrere FRED-Funktionen benutzt und verknüpft. Und so ein Makro ist ja nun wirklich nicht die Welt. Also ist es mit dem Programmieren im Grunde gar nicht so schlimm.

FRED wurde erfunden, damit jeder Benutzer Framework seinen Bedürfnissen anpassen kann. Framework ist an sich schon ein recht mächtiges Programm - FRED eröffnet geradezu grenzenlose Möglichkeiten für alle nur denkbaren Anwendungen.

Framework und FRED zusammen können selbst (teure) Spezialprogramme ersetzen. Warum sollten Sie sich eine Finanzbuchhaltung, eine Lagerverwaltung oder dergleichen anschaffen? Mit FRED können Sie sich das selber programmieren - und zwar genauso, wie Sie es brauchen. Ihrer individuellen Situation gemäß, der kein Standard-Anwendungsprogramm gerecht werden kann. Wenn Sie den Programmieraufwand nicht scheuen. Und, zugegeben, der ist in solchen Fällen dann schon erheblich.

Aber so weit muß man ja nicht gehen. Am Anfang zumindest noch nicht. Die folgenden Kapitel geben einen Überblick, was man mit FRED machen kann - und wie man's macht. Sie sind hernach kein fertiger FRED-Programmierer, dafür müßte dieses Buch ungefähr viermal so dick sein. Doch Sie haben einen ersten Einblick gewonnen und werden dann in der Lage sein, selber weiterzumachen. Sie wissen dann, und das ist die Hauptsache, um die Struktur von FRED.

Im Gegensatz zu einer Programmiersprache wie Basic besteht ein FRED-Programm nicht aus einem ellenlangen Listing. Ein FRED-Programm teilt sich auf in verschiedene kleine Module - strukturiertes Programmieren nennt man dies. Das erleichtert nicht nur den Überblick, es erleichtert auch das Programmieren selber.

Ein Programm dient der Lösung eines bestimmten Problems. Das Problem selber aber besteht wiederum aus vielen Teilproblemen. Ein strukturiertes Programm widmet sich,

vereinfacht gesagt, zunächst einmal jedem Teilproblem für sich und packt die Teillösungen dann zur Gesamtlösung zusammen.

■ **Tip:** Auch wenn's selbstverständlich klingen mag: Analysieren Sie zunächst das Problem, ehe Sie darüber nachdenken, wie es zu lösen sei.

Die folgenden Kapitel geben zunächst eine grobe Einführung in FRED. Damit's nicht gar zu theoretisch bleibt, wird sodann ein komplexes Programm vorgestellt, eine Einnahmen-Überschuß-Berechnung, und ausführlich analysiert und diskutiert. Hier sehen Sie, wie FRED in der Praxis angewendet werden kann. Das Programm ist, wie es sich auf der Beispieldiskette befindet, einsetzbar - aber gleichzeitig so aufgebaut, daß es leicht zu erweitern, zu differenzieren, zu variieren ist.

Natürlich enthält dieses Programm nicht sämtliche FRED-Funktionen - es sind über 140 an der Zahl. Hat man aber erst einmal die Struktur von FRED durchschaut, ist es nicht weiter schwer, die für ein Teilproblem benötigte Funktion herauszufinden und einzubauen.

■ **Hinweis:** Im Framework-Referenzhandbuch sind alle FRED-Funktionen aufgeführt und erläutert.

Wenn Sie auch, wie gesagt, nach diesen Kapiteln noch kein perfekter FRED-Programmierer sein werden: Sie werden zumindest die Angst vor dem Programmieren verloren haben und Lust bekommen, selber zu experimentieren.

32 Grundbegriffe

Ein Programm ist nichts anderes als eine Fülle von Anweisungen, die Framework zu befolgen hat, um ein bestimmtes Ergebnis zu erzielen. Damit Framework die Anweisungen versteht, müssen sie bestimmten Regeln folgen - das ist die Syntax, die Grammatik von FRED. Framework ist da ein bißchen rigider in seinen Anforderungen als wir Menschen. Wer des Deutschen auch nur halbwegs mächtig ist, versteht einen Satz wie »Wein du möchte« trotz seiner haarsträubenden Syntax. Framework würde nur bedauernd den Kopf schütteln. Es besteht auf absolut korrekter Grammatik.

|| **Hinweis:** Alle hier besprochenen Beispiele befinden sich auf der Diskette, die diesem Buch beiliegt.

Funktion, Argument, Parameter

Eine solche Anweisung besteht stets aus dem Namen einer Funktion und deren Argument, wobei dieses sich wiederum aus verschiedenen Parametern zusammensetzen kann.

Ein kleines Programmbeispiel hierfür:

```
; DEMO1
@rept("Hallo! ",5)
```

|| **Hinweis:** Wenn Sie die Programmbeispiele ablaufen lassen wollen, müssen Sie die Frames laden, öffnen und mit **[F5]** neu berechnen.

Dieses Programm schreibt fünfmal den Text »Hallo!«.

- **@rept** ist die *Funktion* - eine sogenannte »eingebaute« Funktion im Gegensatz zu einer benutzerdefinierten (die kommt später). Die Schreibweise der FRED-Funktion ist verbindlich.
- **("Hallo! ",5)** ist das *Argument* der Funktion.
- **"Hallo!"** und **5** sind die *Parameter* des Arguments.

Die Funktion ist die allgemeine Anweisung - wiederhole. Das Argument mit seinen Parametern sagt, was die Anweisung in diesem Fall zu tun hat: wiederhole den Text »Hallo!« fünfmal.

Jede Funktion braucht notwendigerweise ein Argument (von wenigen Ausnahmen abgesehen). Manchmal besteht ein Argument nur aus einem Parameter, manchmal aus mehreren - das hängt von der Funktion ab.

Die Parameter werden stets mit einem Komma voneinander getrennt (ohne Leerzeichen dazwischen), das gesamte Argument steht in Klammern. Ein Argument kann selbst wieder aus Funktionen bestehen, die ihrerseits Argumente haben. Da häufen sich dann die Klammern:

```
; DEM02
@rept(@integer(@sum(4,2)),5)
```

Diesmal wird die Summe aus 4 und 2 gebildet und fünfmal ausgegeben.

- 4,2 ist das Argument von @sum
- @sum(4,2) ist das Argument von @integer
- @integer(@sum(4,2)),5 ist das Argument von @rept

Die Funktion @integer wird in diesem Fall benötigt, um das Ergebnis von @sum, das eine Zahl ist, in einen String umzuwandeln.

|| Hinweis: Der Unterschied zwischen einer Zahl und einem String wurde in Kapitel 21 bereits erläutert.

Um mit den Klammern nicht durcheinander zu kommen (oder um auch sonst den Überblick zu bewahren), lässt sich eine Formel auch auf mehrere Zeilen verteilen und einrücken. Zum Beispiel so:

```
@rept(@integer
      (@sum(4,2)
      )
      ,5)
```

|| Tip: Gewöhnen Sie sich an, Programme übersichtlich zu schreiben. Benutzen Sie für jede Funktion eine eigene Zeile, brechen Sie die Zeilen, wenn sie über die Bildschirmbreite hinausreichen, machen Sie von Einrückungen Gebrauch.

Schreibweisen

Die sonstigen Grundregeln für die Schreibweise von Formeln sind mittlerweile Binsenweisheiten für Sie, weil Sie nach ihnen bisher schon verfahren sind:

- Funktionen beginnen stets mit dem »Klammeraffen« @ oder, alternativ, mit dem Paragraph-Zeichen §.
- Programme (Formeln) stehen »hinter« einem Frame, einem Feld (in einer Datenbank) oder einer Zelle (in einer Tabelle)
- Programme werden in der Regel ausgelöst mit **[F5]** (Neuberechnung) oder mit einer Makro-Taste
- mehrere Funktionen hintereinander werden von einem Komma getrennt
- Kommentare in einem Programm beginnen mit einem Semikolon in der ersten Spalte jeder(!) Zeile.

Werte, Bezüge, Variablen

Eine Formel arbeitet gemeinhin mit Werten: Zahlen, Texten, den Ergebnissen aus Berechnungen. Was die Formel liefert, ist ebenfalls ein Wert. Zum Beispiel das Ergebnis aus $4+2$ oder der Text »Hallo!«.

Werte werden in der Formel entweder als Parameter der Funktion angegeben oder sie werden irgendwoher geholt - etwa aus einer Tabelle. Dann muß in der Formel ein Bezug hergestellt werden zu dieser Tabelle - das kennen Sie alles bereits: Auch Formeln in einer Tabelle oder Datenbank verlangen Angaben darüber, wo die zu berechnenden Werte zu finden sind.

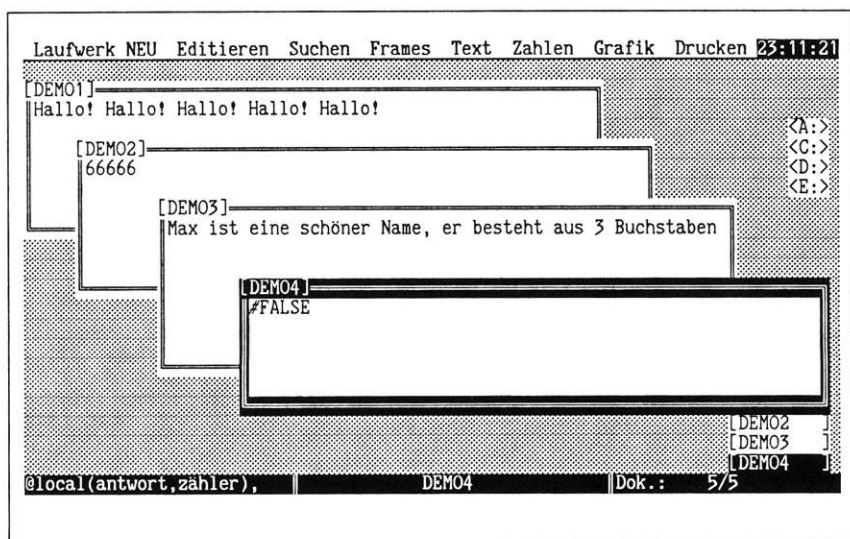


Bild 32.1: Die vier Programmbeispiele [F5]

Der Wert, den ein Programm als Ergebnis liefert, muß irgendwohin ausgegeben werden - er soll ja nicht in den Tiefen des Computers verschwinden, er soll lesbar und weiterzuverwenden sein. Der Ausgabeort ist üblicherweise dort, wo die Formel zu finden ist. Befindet sie sich »hinter« der Zelle einer Tabelle, wird das Ergebnis dort ausgegeben. Steht sie »hinter« einem Frame, wie in den beiden DEMO-Programmen, wird das Ergebnis in den Frame geschrieben.

Solange nichts anderes bestimmt wird. Das Ergebnis kann auch unsichtbar zwischengespeichert und irgendwo anders ausgegeben werden. Das Programm DEMO3 ist dafür ein (beliebtes) Beispiel:

```
; DEMO3
@local(name, zeichen),
name:=@inputline("Wie heißen Sie?"),
zeichen:=@len(name),
name & "ist eine schöner Name, er besteht aus" &
@integer(zeichen) & " Buchstaben")
```

Werte werden zum Beispiel in lokalen Variablen (**name**, **zeichen**) abgelegt, die Variablen werden nach Bedarf in andere Funktionen eingebunden. **@inputline** erwartet vom Benutzer eine Eingabe in der Editierzeile und gibt sie umgehend im Datenspeicher des Programm-Frames aus, hier also im Frame DEMO3 - nachdem sie zuvor in der Variablen **name** zwischengespeichert wurde.

Hinweis: Die Funktion **@local** wurde bereits in Kapitel 30 eingesetzt, **@inputline** kommt in Kapitel 34 ausführlicher zur Sprache.

Die Ausgabe läßt sich auch lenken:

1. Erstellen Sie einen leeren Text-Frame und nennen Sie ihn TEST.
2. Öffnen Sie den Formelspeicher von DEMO3 und ändern Sie die letzte Zeile folgendermaßen ab:

```
test:=name & "ist eine schöner Name, er besteht aus" &
@integer(zeichen) & " Buchstaben")
```
3. Legen Sie DEMO3 auf dem Dokumentenstapel ab und berechnen Sie dann diesen Frame neu mit **[F5]**.

Der Text wird nunmehr im Frame TEST ausgegeben: Mit der Angabe seines Namens haben Sie in der Formel von DEMO3 einen Bezug zu ihm hergestellt.

Bedingungen und Schleifen

Bedingungen und Schleifen sind eine der wichtigsten Funktionen einer Programmiersprache. Bedingungen (**@if**) filtern gleichsam ein Programm: Eine Formel wird nur dann ausgeführt, wenn die Bedingung erfüllt ist. Schleifen (**@while**) ermöglichen Wiederholungen: Eine Formel wird solange ausgeführt, wie die Bedingung zutrifft.

Eine Schleife hat also eine Bedingung bereits integriert - gleichwohl kann man **@if** und **@while** miteinander kombinieren, wie das Beispielprogramm DEMO4 zeigt:

```
; DEMO4:
@local(antwort, zähler),
antwort:=@inputline("Um fortzufahren, bitte J drücken"),
@if(@or(antwort="j", antwort="J"),
    @while(zähler<50,
        demo4:=zähler,
        zähler:=zähler+1
    )
)
```

Das Programm zählt von 1 an hoch - doch nur bis 49. Diese Bedingung ist in der Anweisung **@while(zähler<50** formuliert: Sie dient als Ausstieg aus der Schleife, die sich sonst endlos wiederholen würde.

Zur Erinnerung: In der Framework-Datei CONFIG.FW kann festgelegt werden, nach wieviel Durchgängen eine **@while**-Schleife automatisch abgebrochen wird (unabhängig von der Bedingung). Die Voreinstellung ist 2048.

Das Programm selber indessen wird nur gestartet, wenn auf die Frage in der Nachrichtenzeile mit der Taste J geantwortet wird - das definiert die Bedingung **@if**, wobei hier sowohl der Groß- als auch der Kleinbuchstabe akzeptiert wird. (Mehr über Bedingungen im Kapitel 34.)

Noch ein kleines Beispiel für eine Schleife, mit der man durch ein Dateiverzeichnis blättert:

Hinweis: Das Programm befindet sich als Makro **[ALT]-S** in der Makro-Bibliothek **MAKRO** auf der Beispieldiskette.

```
@while(@nextkey<>{esc},
  @performkeys("{del}{scroll-lock}{dnarrow}{return}"))
```

Solange der nächste Tastenanschlag nicht **[ESC]** ist, wird die folgende **@performkeys**-Anweisung ausgeführt. Das heißt: **[ESC]** stoppt das Programm, jede andere Taste führt es aus. Nämlich: Ser aktuelle Frame wird gelöscht, die nächste Datei geladen. Das Programm setzt voraus, daß, zum einen, der Cursor sich zuvor im Dateiverzeichnis-Frame befunden hat und daß er, zum andern, auf dem Rand eines Frames steht, wenn das Programm gestartet wird.

33 Die Programm-Bibliothek LIB

FRED unterscheidet zwischen »eingebauten« und »benutzerdefinierten« Funktionen. Die eingebauten Funktionen sind jene etwa 140, die im Referenz-Handbuch aufgeführt sind und mit denen Sie zum Teil schon gearbeitet haben.

Benutzerdefinierte Funktionen sind, wie der Name schon sagt, nach Wunsch und Wille des Anwenders entstanden: Sie erweitern FRED, ganz nach Belieben. Sie bestehen aus den eingebauten FRED-Funktionen oder selber aus benutzerdefinierten. Sie sind Programme, die mit einem selbstgewählten Namen aufgerufen werden.

Machen Sie einen Test:

1. Holen Sie das Beispielprogramm DEMO1 auf die Arbeitsfläche, legen Sie es aber im Dokumentenstapel ab.
2. Erstellen Sie einen neuen Frame.
3. Schreiben Sie in dessen Formelspeicher:
@demo1
4. Beenden Sie mit RETURN.

Sie aktivieren damit die benutzerdefinierte Funktion @demo1.

Daraus ersehen Sie bereits, wie man zu eigenen Funktionen kommt: Es sind Programme, die »hinter« einem Frame abgelegt sind; dessen Name ist der Name der Funktion.

Es ist leicht auszumalen, welch unschätzbaren Vorteil solche selbst definierten Funktionen haben: Sie unterstützen das strukturierte Programmieren (weil sie die Lösung eines Teil-Problems enthalten), und sie sind immer abrufbar, mehrfach benutzbar, ohne daß deswegen im aktuellen Programmteil dieses (unter Umständen lange) Programm nochmals geschrieben werden muß.

Freilich kann die Funktion nur eingesetzt werden, wenn Framework weiß, wo sie zu finden ist. In unserem Fall war der Frame DEMO1, also die Funktion @demo1, auf der Arbeitsfläche - kein Problem für Framework.

In der Regel wird man jedoch eigene Funktionen nicht isoliert in einem Frame speichern, sondern sie in umfangreichere Programme einbinden. In diesem Fall ist der Funktions-Frame ein Sub-Frame, zu ihm muß mit einer Pfadangabe ein Bezug hergestellt werden.

■ | **Hinweis:** In Kapitel 34 gibt es dafür Beispiele.

Vielfach aber wird eine benutzerdefinierte Funktion in mehreren Programmen eingesetzt - es wäre unwirtschaftlicher Umgang mit dem Speicherplatz, wollte man sie jedes-

mal integrieren. Deshalb bietet Framework die Möglichkeit einer Programm-Bibliothek, ähnlich den schon bekannten Makro-Bibliotheken.

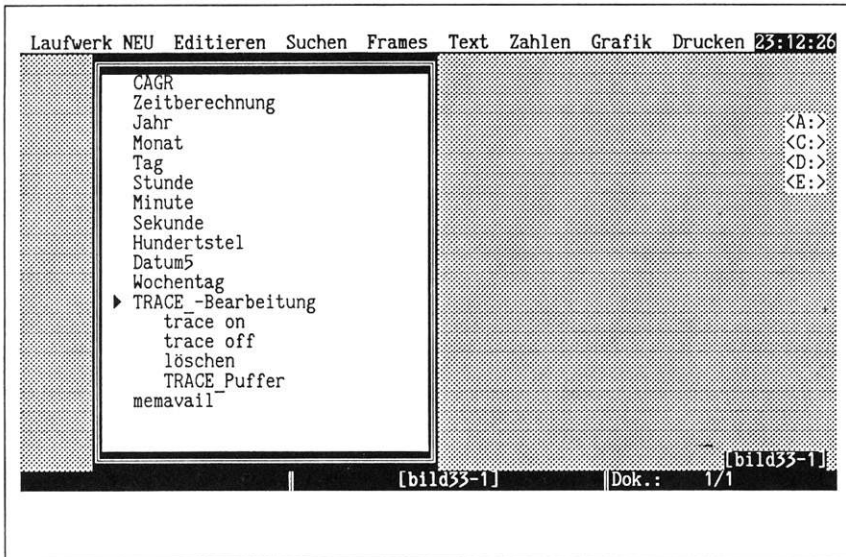


Bild 33.1: Die Framework-Programm-Bibliothek LIB

Diese Programm-Bibliothek heißt LIB.FW - der Name ist vorgeschrieben; indessen können Sie sich beliebig viele solcher Programm-Bibliotheken erstellen. Was freilich Probleme mit sich bringt, wenn sie auf derselben Diskette oder Platte gespeichert werden sollen: Mehrere Dateien gleichen Namens erlaubt MS-DOS nicht.

Merke: Sind mehrere Frames mit dem Namen LIB auf der Arbeitsfläche, wird von Framework nur der erste berücksichtigt.

Da hilft nur ein Trick: Erweitern Sie den Namen zum Beispiel mit Nummern - LIB1, LIB2 usw. - und ändern Sie das wieder um zu LIB, wenn Sie die Bibliothek benutzen. Vorsicht: Beim Speichern die erneute Namensänderung nicht vergessen!

Wenn Framework beim Abarbeiten des Programmes auf eine Funktion stößt, die nirgendwo unterzubringen ist, wird im Frame LIB gesucht - vorausgesetzt, er befindet sich auf der Arbeitsfläche. Darum stehen die LIB-Funktionen jederzeit und in jedem Programm zur Verfügung.

LIB ist ein ganz normaler Konzept-Frame, dessen Sub-Frames die Funktionen enthalten und ihnen zugleich die Namen geben.

Framework bringt in seiner LIB-Bibliothek bereits einige Funktionen mit. Sie sind nachstehend aufgeführt - samt den Programm listings (kann man viel lernen draus!); einige Erläuterungen dazu folgen hernach. Der besseren Lesbarkeit wegen sind die den Programmen vorangestellten Kommentare kursiv gesetzt, auf die in Framework-Programmen notwendigen Semikolon wurde verzichtet; ansonsten wurden sie wie im Original belassen.

CAGR:

CAGR (Compound Annual Growth Rate) - Gesamte Jährliche Wachstumsrate - (Zukunftswert, Gegenwartswert, Anzahl Jahre)

Diese benutzerdefinierte Funktion verlangt drei Parameter

EINGANGSWERTE:

ZUKUNFTSWERT: Zukünftiger Geldbetrag in "n Jahren"

GEGENWARTSWERT: Wert des Startkapitals

N JAHRE: Anzahl der Jahre, über die sich das Startkapital zusammensetzt

AUSGANGSWERT:

CAGR erbringt als Ergebnis den Zinswert, welchen das Startkapital nach n Jahren erreicht haben wird.

```
(10^((@log(@item1)-@log(@item2))/(@item3)))-1
```

Zeitberechnung:

```
@Zeitberechnung(@item1,@item2) => @item1-@item2
```

Die Funktion Zeitberechnung benötigt zwei Parameter (@item), beide müssen alphanumerische Zeitzeichenketten der folgenden Form sein: hh:mm:ss.

Als Ergebnis erscheint diese Differenz-Zeit in der folgenden Form: hh:mm:ss.

```
@local(hr,min,sec,delta),
@set(hr,@value(@mid(@item1,1,2))-@value(@mid(@item2,1,2))),
@set(min,@value(@mid(@item1,4,2))-@value(@mid(@item2,4,2))),
@set(sec,@value(@mid(@item1,7,5))-@value(@mid(@item2,7,5))),
@if(sec<0, @list(@set(min,min-1),@set(sec,sec+60))),
@if(min<0, @list(@set(hr,hr-1),@set(min,min+60))),
@if(hr<0,@set(hr,hr+24)),
@set(delta,
  @if(hr<10,"0","")&@integer(hr)&
  @if(min<10,":0","")&@integer(min)&
  @if(sec = 0, ":00.00",
    @if(sec<1,":00",
      @if(sec<10,":0","")&@decimal(sec,2))),
@return(delta)
```

Jahr:

```
@Jahr(@item1) => Jahr
```

Die Funktion Jahr benutzt einen Parameter, der den Wert des Tagesdatums beinhalten muß.

Als Ergebnis wird die Jahreszahl angezeigt.

```
@local(datestr),
@set(datestr,@scientific(@item1)),
@local(i,j),
@while(i <> "/",
    @set(j,j+1),
    @set(i,@mid(datestr,j,1))
),      ; found first /
@set(i,0),
@while(i <> "/",
    @set(j,j+1),
    @set(i,@mid(datestr,j,1))
),      ; found second /
@set(j,j+1),      ; für Jahr-Feld
@value(@mid(datestr,j,4)) ; konvertiert die Jahr-Zeichenkette
in einen numerischen Wert.
```

Monat:

```
@Monat(@item1) => Monat
```

Die Funktion Monat (@Monat) benutzt einen Parameter, welcher das Tagesdatum beinhaltet.

Als Ergebnis wird der Monatswert angezeigt.

```
@local(datestr),
@set(datestr,@scientific(@item1)),
@local(i,j),
@while(i <> "/",
    @set(j,j+1),
    @set(i,@mid(datestr,j,1))
),      ; found first /
@set(j,j-1),      ; für die Länge des Monat-Feldes.
@value(@mid(datestr,1,j)) ; konvertiert die Monat-Zeichenkette
in einen numerischen Wert.
```

Tag:

```
@Tag(@item1) => Tag
```

Die Funktion Tag (@Tag) benutzt einen Parameter, welcher das Tagesdatum beinhalten muß.

Erbringt als Ergebnis das Tagesdatum.

```
@local(datestr),
@set(datestr,@scientific(@item1)),
@local(i,j,k),
@while(i <> "/",
    @set(j,j+1),
    @set(i,@mid(datestr,j,1))
),      ; found first /
@set(k,j+1),
@set(i,0),
@while(i <> "/",
    @set(j,j+1),
    @set(i,@mid(datestr,j,1))
),      ; found second /
@set(j,j-k),      ; für die Länge des Tag-Feldes
@value(@mid(datestr,k,j)) ; konvertiert die Tag-Zeichenkette
in einen numerischen Wert.
```

Stunde:

```
@Stunde(@item1) => Stunde
```

Die Funktion Stunde benutzt einen Parameter, welcher das Tagesdatum beinhalten muß.

Als Ergebnis wird die Stunde angezeigt.

```
@return(@value(@mid(@time3(@item1),1,2)))
```

Minute:

```
@Minute(@item1) => Minute
```

Die Funktion Minute (@Minute) benutzt einen Parameter, welcher das Tagesdatum beinhaltet.

Als Ergebnis werden die Minuten angezeigt.

```
@return(@value(@mid(@time3(@item1),4,2)))
```

Sekunde:

```
@Sekunde(@item1) => Sekunde
```

Die Funktion Sekunde (@Sekunde) benutzt einen Parameter, welcher den Tageswert beinhalten muß.

Als Ergebnis werden die Sekunden angezeigt.

```
@return(@value(@mid(@time3(@item1),7,5)))
```

Hundertstel:

```
@Hundertstel(@item1) => Hundertstel
```

Die Funktion `Hundertstel (@Hunderstel)` benutzt einen Parameter, welcher den Tageswert beinhalten muß.

Als Ergebnis werden die Sekunden angezeigt.

```
@local(sec),
@set(sec,@sekunde(@item1)),
@return(sec-@int(sec))
```

Datum5:

```
@Datum5(@item1) => Wochentag Monat Tag, Jahreszahl
```

Diese Funktion (`@Datum5`) benötigt einen Parameter, welcher das Tagesdatum enthält.

Erbringt als Ergebnis eine Datums-Zeichenkette.

```
@return(@Wochentag(@item1) & " " & @date4(@item1))
```

Wochentag:

```
@Wochentag[(@item1,@item2,@item3)] => Angabe des Wochentages
```

Diese Funktion (`@Wochentag`) wird als Ergebnis das aktuelle Datum erbringen, wenn keine anderen Parameter benutzt werden. Andernfalls benutzt `@Wochentag` einen Parameter, der das Tagesdatum enthält und ergibt als Resultat den Wochentag.

```
@local (checkdate, diffdate),
checkdate := @if (@itemcount <> 0, @item1, @today),
diffdate := @mod(@diffdate(checkdate, @date(1984,1,1)),7),
@if (diffdate < 0, diffdate := diffdate + 7),
@select (diffdate + 1,
"Sonntag", "Montag", "Dienstag", "Mittwoch", "Donnerstag",
"Freitag", "Samstag")
```

[TRACE_-Bearbeitung]:

Positionieren und Größenänderung des »Trace_Puffers« in Dokumentform

```
{f5} 'lib.TRACE_-Bearbeitung.trace'zum Starten von 'Trace'.
{f5} 'on lib.TRACE_-Bearbeitung.trace off'zum Stoppen von
'Trace'.
{f5} 'on lib.TRACE_-Bearbeitung.clean'zum Löschen des 'Trace-
Puffers'.
Der 'Trace-Puffer'ist im lib.TRACE_Bearbeitung.TRACE_Puffer
angelegt.
```

[TRACE_-Bearbeitung].trace on:

```
@trace(TRACE_Puffer,#on)
```

[TRACE_-Bearbeitung].trace off:

```
@trace(TRACE_Puffer,#off)
```

[TRACE_-Bearbeitung].löschen:

```
TRACE_Puffer := " "
```

memavail:

Bitte herauskopieren und mittels POSITION- und GRÖSSE-Taste der Frame in die obere rechte Ecke auf der Arbeitsfläche positionieren.

Durch Drücken der {f5} NEUBERECHNEN-Taste wird die aktuell noch verfügbare Speicherkapazität angezeigt.

ACHTUNG: Der Rücknahme-Befehl aus dem Editier-Menü ist danach unwirksam. Das heißt, die als gelöscht markierten Speicherbereiche werden für weitere Bearbeitung endgültig gelöscht.

```
@memavail
```

Einige Erläuterungen zu diesen Funktionen.

@zeitberechnung:

Die beiden Werte müssen zuvor definiert werden, am besten mit Variablen, weil das am übersichtlichsten ist:

```
@local(a,b),  
a:=@time3(@time(12,12,12),  
b:=@time3(@time(11,11,11),  
@zeitberechnung(a,b)
```

Wer keine Angst vor Klammern hat, kann auf Variablen verzichten:

```
@zeitberechnung((@time3(@time(12,12,12))),(@time3  
(@time(11,11,11))))
```

@time3 ist die FRED-Funktion, welche das in **@zeitberechnung** geforderte Zeitformat ergibt.

@jahr, @monat, @tag, @stunde, @minute, @sekunde, @hundertstel:

Das sind Ergänzungen zu Frameworks sonstigen Datums- und Zeitfunktionen, die nur einen Ausschnitt aus den sonst üblichen Angaben liefern. **@monat** und **@tag** gehen allerdings vom amerikanischen Datumsformat aus, bei dem der Monat vor dem Tag angegeben wird, und deshalb liefert **@monat** den Tag, **@tag** den Monat. Wie dem am

einfachsten abzuhefen ist, ergibt die Logik: Wer den Monat wissen will, benutzt eben **@tag** - und umgekehrt.

@datum5:

Ebenfalls eine Ergänzung zu den sonstigen Datumsfunktionen. Ausgegeben wird die Schreibweise

Sonntag 31. Mai 1987

Ihnen fehlt ein Komma zwischen Wochentag und Datum? Das ist leicht einzubringen. Ändern Sie die Funktion folgendermaßen ab:

```
@return(@Wochentag(@item1) & ", den " & @date4(@item1))
```

Als Ergebnis sehen Sie dann:

Sonntag, den 31. Mai 1987

@wochentag:

Diese Funktion liefert ohne jeglichen Parameter den aktuellen Wochentag. Ansonsten kann man damit blitzschnell den Wochentag eines beliebigen Datums ermitteln:

```
@wochentag(@date(1946,12,6))
```

Framework meldet, der 6. Dezember 1946 sei ein Freitag gewesen. Beachten Sie das Datumsformat: erst das Jahr (vier Ziffern!), dann der Monat, dann der Tag.

@memavail:

Eigentlich ist das eine ganz normale FRED-Funktion. Das Besondere an der LIB-Version ist lediglich, daß der Sub-Frame ein kleinstmöglicher Frame ist, den man sich irgendwo in die Ecke stellt. Dort bleibt er - das ist der Unterschied zu der Speicherplatz-Abfrage mit dem Makro **[ALT][F7]**, deren Wert in der Nachrichtenzeile ausgegeben und dort beim nächsten Tastenanschlag wieder gelöscht wird.

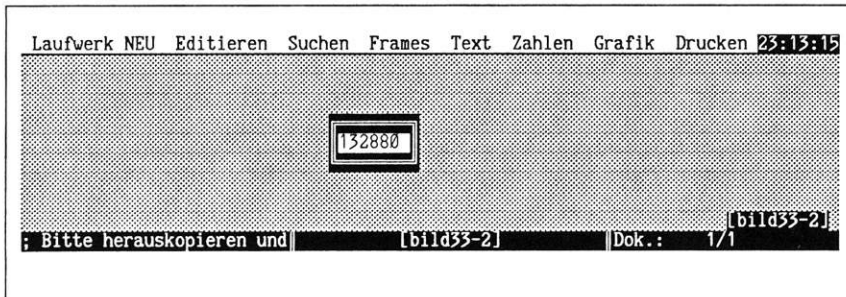


Bild 33.2: MEMAVAIL, der Speicherabfrage-Mini-Frame aus LIB

Ein anderer Unterschied wird das große Geheimnis von Framework bleiben müssen: Beide geben verschiedene Werte aus. Aber nicht so gravierend, daß es beunruhigen müßte.

@trace_bearbeitung:

Auch **@trace** ist eine normale FRED-Funktion, die einen Programmablauf dokumentiert und zur Fehlersuche eingesetzt wird - zum Debugging, wie die Profis sagen.

Diese Ablaufdokumentation wird normalerweise so in Gang gesetzt:

```
@trace(Frame, #ON)
```

»Frame« steht hier für den Bezug auf einen Frame, in welchen die Dokumentation ausgegeben wird.

Die **@trace**-Funktion in LIB erleichtert den Gebrauch, weil nicht ein eigener Ausgabe-Frame angelegt und angegeben werden muß; das Ergebnis wird in den Sub-Frame *TRACE_Puffer* geschrieben.

Tip: Sie können den gesamten *Trace*-Komplex aus LIB herauskopieren und ihn als eigenständigen Frame benutzen, wie es im Kommentar empfohlen wird. Das ist für den Programmablauf nicht notwendig, erleichtert jedoch die Handhabung.

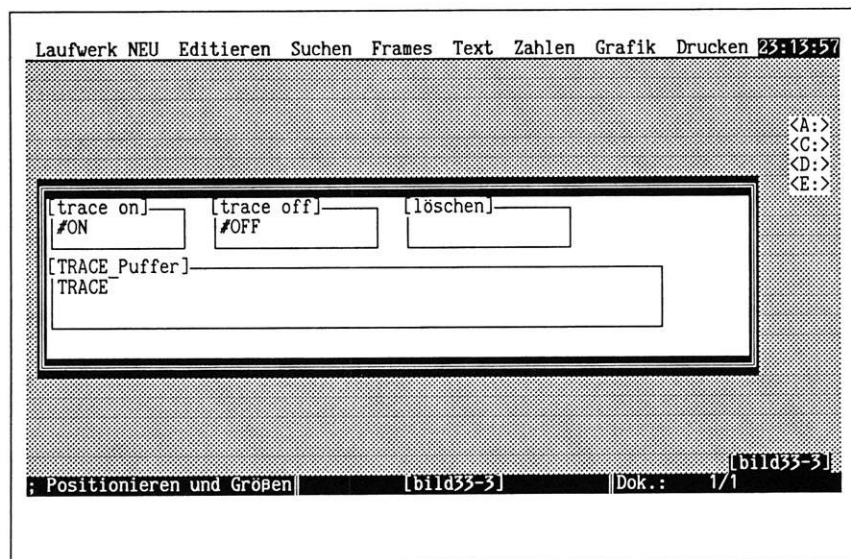


Bild 33.3: Das Dokumentationsprogramm *TRACE*

Gehandhabt wird das folgendermaßen:

1. Berechnen Sie den Sub-Frame *trace on* neu mit **[F5]** - das schaltet die Funktion an.
2. Bringen Sie den Cursor auf den Frame mit Ihrem Programm und lassen Sie es ablaufen.
3. Berechnen Sie den Sub-Frame *trace off* neu mit **[F5]** - das schaltet die Funktion wieder ab.
4. Öffnen Sie den Sub-Frame *TRACE_Puffer* mit **[F9]** (*Zoom*) und schauen Sie sich die Dokumentation an.
5. Berechnen Sie den Sub-Frame *löschen* neu mit **[F5]** - das löscht den Puffer.

Wie erkennt man in einer solchen Dokumentation die Programmfehler? Gar nicht so einfach. Schauen wir zunächst ein fehlerloses Programm an, nämlich DEMO3. Zur Erinnerung nochmals das Listing:

```
;DEMO3
@local(name,zeichen),
name:=@inputline("Wie heißen Sie?"),
zeichen:=@len(name),
name & "ist eine schöner Name, er besteht aus" &
@integer(zeichen) & " Buchstaben")

@trace zeichnet den Programmablauf so auf:

TRACE Result: #ON
LOCAL
LOCAL Result: 0
SET
    INPUTLINE
    INPUTLINE Result: "Max"
SET Result: "Max"
SET
    LEN
    LEN Result: 3
SET Result: 3
INTEGER
INTEGER Result: "3"
TRACE
```

Das Ergebnis 0 zu Beginn zeigt an, daß die Variablen definiert sind und vorerst noch den Wert 0 haben - der richtige wird ihnen ja erst im Verlauf des Programms zugewiesen.

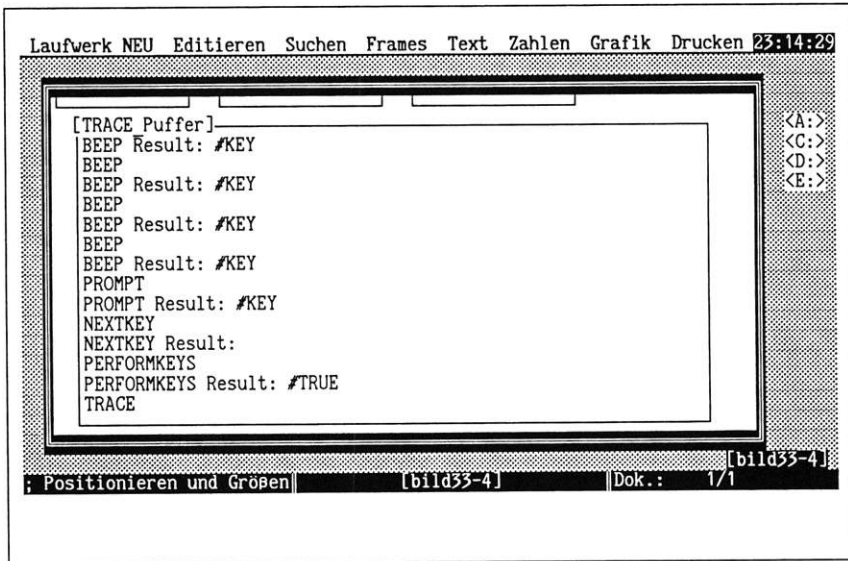


Bild 33.4: Eine Programmdokumentation mit TRACE

Jetzt ändern wir das Programm um eine Winzigkeit und lassen in der dritten Zeile im Zuweisungszeichen := den Doppelpunkt weg, so daß es nunmehr so aussieht:

```
@local(name,zeichen),
name:=@inputline("Wie heißen Sie?"),
zeichen=@len(name),
name & "ist eine schöner Name, er besteht aus" &
@integer(zeichen) & " Buchstaben")
```

Das Programm läuft nach wie vor, errechnet aber als Anzahl der Buchstaben 0.

In der Dokumentation schlägt sich das folgendermaßen nieder:

```
TRACE Result: #ON
LOCAL
LOCAL Result: 0
SET
  INPUTLINE
  INPUTLINE Result: "Max"
SET Result: "Max"
LEN
LEN Result: 3
INTEGER
INTEGER Result: "0"
TRACE
```

Der Unterschied fällt auf Anhieb wahrscheinlich nicht ins Auge: Nach der Zeile

```
SET Result: "Max"
```

fehlt im zweiten Beispiel die Zeile

```
SET
```

Das ist der Hinweis darauf, daß mit der Zuweisung zu einer Variablen etwas nicht in Ordnung ist - der fehlende Doppelpunkt, der verhindert, daß der Variablen **zeichen** im Programmverlauf ein Wert zugewiesen wird.

Am Anfang ist nicht so leicht auszumachen, wo der Fehler zu orten ist, obschon er getreulich protokolliert wird. Aber auch hier hilft die Erfahrung weiter.

34 Überschuß-Berechnung (II)

oder wie der Dichter Jonathan Kraft den Überblick über seine Finanzlage behält

Sie erfahren in diesem Kapitel:

- wie man ein komplexes Programm aufbaut
- wie man mit einem Menü die Eingabe erleichtert
- wie FRED selbständig Eingaben generiert
- wie man mit FRED Daten auswertet
- wie man u.a. folgende Funktionen einsetzt:

@printreturn

@menu

@keyfilter

@execute

@inputline

@local

Sie benötigen:

- VERSTART.FW von der Beispieldiskette
- VERLUST.FW von der Beispieldiskette

Das Problem

Jonathan Kraft, der Dichter, hat sich ja nun schon (in Kapitel 24) eine Möglichkeitersonnen, seine ungeliebte Buchhaltung halbwegs in Griff zu bekommen. Halbwegs aber nur - und selbst das ist noch der pure Euphemismus. (Als Dichter muß Jonathan Kraft solche Fremdwörter drauf haben, das gehört zum Image: Ein Euphemismus ist eine Übertreibung.)

Denn sonderlich bequem ist seine Überschuß-Berechnung nicht zu handhaben. Er muß, wenn er sich in einer stillen Stunde schon mal daran macht, seine ganzen Belege zu erfassen, die sich meist schon beängstigend türmen - er muß ständig hin- und herspringen zwischen Einnahmen und Ausgaben.

Die Datenbanken für seine Einnahmen und Ausgaben sind überdies Monstren, schwer zu überblicken und nur mit einem Höchstmaß an Konzentration auszufüllen: Welcher Betrag kommt in welche Spalte? Und außerdem stören ihn Frameworks stete Fehlermeldungen, wenn eine Spalte keinen Wert aufweist - was leider in der Natur der Sache liegt und deswegen nicht zu umgehen ist.

Es sei denn, man macht das alles ganz anders. Einfacher. Praktischer. Komfortabler. Brauchbarer. Jonathan Kraft ist nur halb gedient, wenn einer, Framework nämlich, seine Einzelposten zusammenrechnet. Gut, das ist schon was. Aber Jonathan Kraft

möchte dazuhin seine Finanzlage auch auswerten können: Was hat er im Januar verdient? Wieviel hat er für Büromaterial ausgegeben?

Das geht auch mit seiner bisherigen Überschuß-Berechnung, gewiß. Aber es sollte doch möglich sein, wünscht sich Jonathan Kraft, das alles gewissermaßen per Knopfdruck zu erledigen (lassen).

Die Lösung

Es ist möglich. Dazu allerdings hat Jonathan Kraft ein veritables Programm schreiben müssen - das er VERLUST.FW genannt hat. Und das sogar noch ein kleines Anhängsel hat: VERSTART.FW.

Hinweis: Beide Frames müssen auf der Arbeitsfläche vorhanden sein, damit das Programm funktioniert.

Das Anhängsel ist eigentlich, um in der Begrifflichkeit von Jonathan Krafts Dichter-Sphäre zu bleiben, ein Epilog, welcher nur aus einer kleinen Formel besteht:

```
@setmacro({alt-v},Verlust.Start)
```

Damit wird also **ALT-V** als eine Makro-Taste definiert, das Makro selber ist im Frame VERLUST.START zu finden: Das gesamte Überschuß-Berechnungsprogramm ist ein Makro.

Jonathan Kraft hat dies aus wohlervogenen Gründen so gemacht. Framework erlaubt zwei unendlich wichtige FRED-Funktionen nämlich nur innerhalb eines Makros:

- @performkeys
- @setselection

Daß dies überflüssigen Umstand bereitet, haben die Framework-Entwickler auch mitbekommen und in Framework II, der neuesten, erweiterten (und ziemlich teuren) Fassung, diese Beschränkung aufgehoben. Jonathan Kraft muß mit ihr noch zurechtkommen, was ihn, wie gesagt, dazu zwingt, sein Programm als Makro zu definieren. Das ist mehr eine Formsache und nichts, worüber man graue Haare bekommen könnte.

Tip: Obschon es möglich ist: Legen Sie solche umfangreichen Makros nicht in einer Makro-Bibliothek ab - zugunsten des Speicherplatzes.

Daß dieses Makro im VERSTART-Epilog angelegt wird, ist ebenfalls mit Bedacht geschehen. Um mit einem Makro arbeiten zu können, das weiß Jonathan Kraft, muß es mit **F5** (*Neuberechnung*) erst einmal aktiviert werden. Wenn Framework aber sein gesamtes VERLUST-Programm neu durchrechnet, nur weil eben das Makro erst aktiviert sein will, ist das Zeitverschwendung - und kann überdies das Programm auch

durcheinanderbringen, wenn zum Beispiel Tastatureingaben erwartet werden, die dann nicht kommen.

Dann lieber, zur Sicherheit, ein Start-Frame. Um sein Programm VERLUST aufzurufen, muß Jonathan Kraft folgendes unternehmen:

1. Er bringt den Cursor auf VERSTART.
2. Mit **[F5]** (*Neuberechnung*) aktiviert er das Makro.
3. Mit **[ALT]-V** startet er das eigentliche Programm VERLUST.

Und befindet sich sofort in einem Menü, das ihm verschiedene Auswahlmöglichkeiten läßt. Verfolgen wir einmal, was Jonathan Kraft nun macht - um zu sehen, wie dieses Programm überhaupt funktioniert.

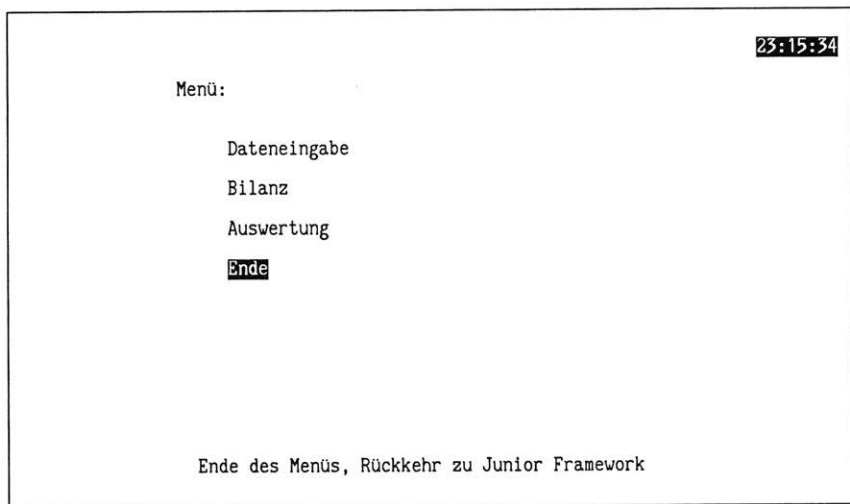


Bild 34.1: Das Hauptmenü des VERLUST-Programmes **[F5]**

Er wählt im Menü zunächst *Dateneingabe* - wie gehabt mit den Cursortasten und anschließendem **[RETURN]** oder mit dem ersten Buchstaben des Menüpunktes.

Er landet in einer Datenbank und erfaßt dort sogleich, ohne den Cursor bewegen zu müssen, weil er bereits in der ersten freien Zeile steht, eine Rechnung, die auf seinem Schreibtisch liegt, und tippt also (es sei März):

1. 3 **[TAB]**
2. Disketten **[TAB]**
3. 39.25 **[TAB]**
4. **[CTRL]-B**

Laufwerk NEU Editieren Suchen Frames Text Zahlen Grafik Drucken 23:16:03						
Monat	Gegenstand	Brutto	Kategorie	Netto	Mwst	Gesamtwert
3	ABC Verlag	41,84	V	39,10	2,74	
1	Ultimo Verlag	5350	V	5000	350	
2	PC Magazin	150	Z	140,19	9,81	
2	Aichwalder Nachrich	63,45	Z	59,30	4,15	
1	Schanbacher Zeitung	192,60	Z	180	12,60	
1	VG Wort	1270,87	S	1187,73	83,14	
2	SR	500	R	467,29	32,71	
1	DLF	610,50	R	570,56	39,94	
1	SRG	228,40	R	228,40	0	
2	Reparatur	570,34	A	500,29	70,04	712,92
1	Autosteuer	150,40	A	150,40	0	188
1	Parkgebühren	206,10	X	180,79	25,31	
2	Telefon	822,51	D	822,51	0	1370,85
3	Strom	151,47	H	132,87	18,60	
3	Heizöl	622,85	H	546,36	76,49	2491,41
2	Zeit	143	M	133,64	9,36	
1	Briefumschläge	5,45	B	4,78	0,67	
3	Fachbücher	25,20	F	23,55	1,65	
@echo(#OFF),		Verlust.Datentabelle.Monat		Sätze: 38/38		

Bild 34.2: Der Frame zur Dateneingabe

Auf wundersam scheinende Weise rechnet das Programm sofort den Netto-Betrag und die Mehrwertsteuer aus und bringt den Cursor sodann an den Anfang der nächsten Zeile, bereit für die als nächstens zu erfassenden Daten.

Jonathan Kraft begnügt sich damit, verläßt mit **[ALT]-V** die Dateingabe, landet wieder im Menü und wählt nun dort *Bilanz*.

■ **Merke:** In diesem Programm kommt man mit **[ALT]-V** stets zum Hauptmenü zurück.

Framework beginnt zu rechnen und präsentiert kurz darauf eine Tabelle mit der aktuellen Bilanz.

Laufwerk NEU		Editieren	Suchen	Frames	Text	Zahlen	Grafik	Drucken	23:16:39
		A		B					
1				Ausgaben					
2	Brutto			4172,06					
3	Netto			3880,14					
4									
5									
6				Einnahmen					
7	Brutto			9548,29					
8	Netto			8938,58					
9									
10									
11				Bilanz					
12									
13	Brutto/Netto			5058,44					
14	Gewinn brutto			5376,23					
15	Mwst			317,79					

@setselection("Verlust.Bil	Verlust.Bilanztabelle.B3	Zelle: 2/3
----------------------------	--------------------------	------------

Bild 34.3: Die allgemeine Bilanz

Wieder zurück im Menü, entscheidet sich Jonathan Kraft für *Auswertung* - was ihm ein erneutes Menü beschert. Fürs erste nimmt er *Auswertung nach Sachgruppen*. Framework rechnet wieder - und zeigt ihm eine Bilanz, nach Sachgruppen untergliedert. Und er weiß nunmehr, was er im März für Bürobedarf ausgegeben und was er im Januar bei Zeitungen eingenommen hat.

Wie Sie an dieser Stelle bemerken, ist das Programm VERLUST zwar lauffähig, doch noch nicht vollständig. Zum Beispiel geht diese Auswertung nur bis März, und es sind auch nicht alle möglichen Sachgruppen eingetragen. Dies, um auf der Beispieldiskette Platz zu sparen, und um es Ihnen leichter zu machen, das Programm Ihren Bedürfnissen anzupassen. Denn daß Sie die gleichen Sachgruppen und dergleichen benötigen wie Jonathan Kraft, ist eher unwahrscheinlich. Aber keine Sorge: das Programm so zu variieren, daß es überall und jedem paßt, ist kein Problem.

Laufwerk NEU Editieren Suchen Frames Text Zahlen Grafik Drucken 23:17:20						
	A	B	C	D	E	
1	Einnahmen (brutto)					
2						
3		Januar	Februar	März	Gesamt	
4						
5	Zeitungen	192,60	546,45	807,63	1546,68	
6	Verlage	5350		0	41,84	5391,84
7	Rundfunk					
8	Sonstiges					
9						
10						
ust.Sachgruppenauswertung.E8 Zelle: 5/8						

Bild 34.4: Die Auswertung nach Sachgruppen "ENDE"

Doch vorerst weiter im Programm. Jonathan Kraft ist wieder ins Hauptmenü zurückgekehrt und will sich nunmehr im Untermenü *Auswertung* eine *Individuelle Auswertung* erstellen lassen.

Das Programm fragt ihn, welche Sachgruppe er denn ausgewertet haben möchte. Er gibt »Disketten« an - und schon wird ihm gesagt, was er bis zu diesem Zeitpunkt für Disketten ausgegeben hat. Offensichtlich ist er ein Vielschreiber.

Laufwerk NEU Editieren Suchen Frames Text Zahlen Grafik Drucken 23:19:01						
Disketten: 528,50 DM >						
Porto: 743,50 DM >						
EZ: 1.140,63 DM >						
Benzin: 33,09 DM >						
Autowäsche: 9,60 DM						
Bürobedarf: 160,05 DM						
[bild34-5] Dok.: 1/1						

Bild 34.5: Die Sachgruppenauswertung

Damit sei der Rundgang im Programm VERLUST abgeschlossen. Jonathan Kraft wählt im Menü *Ende*, das Programm samt seinen Tabellen wird gespeichert und ge-

schlossen, Jonathan Kraft befindet sich wieder in Framework und kann sich die Einzelheiten seines Programmes nun anschauen.

Es besteht, zunächst einmal, aus dem umfassenden Frame VERLUST und etlichen Sub-Frames. Die Struktur:

1. Start (T)
2. Menü
 - 2.1 Dateneingabe (T)
 - 2.2 Bilanz (T)
 - 2.3 Auswertung
 - 2.3.1 Auswertung nach Sachgruppen (T)
 - 2.3.2 Individuelle Auswertung (T)
 - 2.3.3 Zurück zum Hauptmenü (T)
 - 2.4 Ende (T)
3. Datentabelle (D)
4. Bilanztabelle (K)
5. Sachgruppenauswertung (K)
6. Individualauswertung (T)
7. Tastaturfilter (T)
8. Mw14 (T)
9. Mw7 (T)
10. Mw0 (T)
11. Auto (L)
12. Bürokosten (L)

Ein Konzept-Frame also - verschachtelte Frames. Die Numerierung indessen erfolgt hier nur zur Verdeutlichung der Struktur. In seinem Programm hat sie Jonathan Kraft unterlassen, damit er vertikal angeordnete Menüs bekommt.

|| Zur Erinnerung: Ein Konzept-Frame mit Numerierung erbringt ein horizontal angeordnetes Menü, ohne Numerierung ein vertikal angeordnetes.

Hinter den Frames verbirgt sich eine stolze Anzahl von Formeln. Es folgt hier zunächst einmal das gesamte Listing (allerdings ohne die Formeln in den Tabellen und der Datenbank):

Verlust:

```
@printreturn,  
@setmacro({alt-v},Verlust.start),  
@verlust.start
```

Verlust.start:

```
@keyfilter({all}),  
@menu(Menü)
```

Verlust.Menü.Dateneingabe:

```
@echo(#OFF),
@execute(Verlust.Datentabelle,#TRUE),
@setselection("Verlust.Datentabelle"),
@performkeys("{f9}"),
@performkeys("{ctrl-home}{ctrl-dnarrow}{dnarrow}"),
@echo(#ON),
@keyfilter({ctrl-z},Tastaturfilter),
@quitmenu
```

Verlust.Menü.Bilanz:

```
@setselection("Verlust.Bilanztabelle"),
@performkeys("{f5}"),
@performkeys("{f9}"),
@quitmenu
```

Verlust.Menü.Auswertung.Auswertung nach Sachgruppen:

```
@setselection("Verlust.Sachgruppenauswertung"),
@performkeys("{f5}"),
@performkeys("{f9}"),
@quitmenu
```

Verlust.Menü.Auswertung.Individuelle Auswertung:

```
@local(Kriterium,Ergebnis),
Kriterium:=@inputline("Welche Sachgruppe soll ausgewertet werden?"),
@echo(#OFF),
@execute(Verlust.Datentabelle,#TRUE),
@setselection("Verlust.Datentabelle"),
@performkeys("{f2}{home}{f6}{end}{del}"),
@performkeys("Gegenstand=" & " " & Kriterium & " " & "{return}"),
Ergebnis:=@currency(@sum(Datentabelle.Brutto)),
@setselection("Individualauswertung"),
@performkeys("{f9}{ctrl-end}" & Kriterium & ":" & "{tab}" & Ergebnis & "{return}"),
@echo(#ON),
@quitmenu
```

Verlust.Menü.Auswertung.Zurück zum Hauptmenü:

```
@Verlust.start,
@quitmenu
```

Verlust.Menü.Ende:

```
@keyfilter({all}),
@setselection("Verlust"),
@performkeys("{f10}"),
@performkeys("{ctrl-return}"),
@performkeys("{return}"),
@quitmenu
```

Verlust.Tastaturfilter:

```
;Honorar Zeitungen (7%)
@if(@key={ctrl-z},
```

```
@list(@performkeys("Z"),@mw7)
),

;Honorar Verlage (7%)
@if(@key={ctrl-v},
    @list(@performkeys("V"),@mw7)
),

;Honorar Rundfunk (7%)
@if(@key={ctrl-r},
    @list(@performkeys("R"),@mw7)
),

;Honorar Sonstige (7%)
@if(@key={ctrl-s},
    @list(@performkeys("S"),@mw7)
),

;Ausgaben Bürobedarf (14%)
@if(@key={ctrl-b},
    @list(@performkeys("B"),@mw14)
),

;Ausgaben Porto
@if(@key={ctrl-p},
    @list(@performkeys("P"),@mw0)
),

;Ausgaben Diverses (14%)
@if(@key={ctrl-d},
    @list(@performkeys("D"),@mw14)
),

;Ausgaben Magazine/Zeitschriften (7%)
@if(@key={ctrl-m},
    @list(@performkeys("M"),@mw7)
),

;Ausgaben Fachbücher (7%)
@if(@key={ctrl-f},
    @list(@performkeys("F"),@mw7)
),

;Ausgaben Auto (14%); anteilig 80%
@if(@key={ctrl-a},
    @list(@performkeys("A"),@auto)
),

;Ausgaben Bürokosten (14%); anteilig 25 %
@if(@key={ctrl-h},
    @list(@performkeys("H"),@Bürokosten)
),

;Ausgaben Parkgebühren (14%)
@if(@key={ctrl-x},
    @list(@performkeys("X"),@mw14)
)
```

Verlust.Mw14:

```
@performkeys("{tab}"),
@performkeys("{f2}" & "Brutto/114*100" & "{return}"),
@performkeys("{tab}"),
@performkeys("{f2}" & "Brutto/114*14" & "{return}"),
@performkeys("{tab}{return}{home}")
```

Verlust.Mw7:

```
@performkeys("{tab}"),
@performkeys("{f2}" & "Brutto/107*100" & "{return}"),
@performkeys("{tab}"),
@performkeys("{f2}" & "Brutto/107*7" & "{return}"),
@performkeys("{tab}{return}{home}")
```

Verlust.Mw0:

```
@performkeys("{tab}"),
@performkeys("{f2}" & "Brutto*1" & "{return}"),
@performkeys("{tab}"),
@performkeys(""),
@performkeys("{tab}{return}{home}")
```

Verlust.Auto:

```
@performkeys("{tab}{leftarrow}{leftarrow}{f8}{tab}{tab}{tab}{tab}{tab}{return}"),
@performkeys("{leftarrow}"),
@performkeys("{f2}" & "Brutto*1" & "{return}"),
@performkeys("{leftarrow}"),
@performkeys("{f2}" & "(Anteilig/114*14)/100*80" & "{return}"),
@performkeys("{leftarrow}"),
@performkeys("{f2}" & "(Anteilig/114*100)/100*80" & "{return}"),
@performkeys("{leftarrow}{leftarrow}{del}"),
@performkeys("{f2}" & "Anteilig/100*80" & "{return}"),
@performkeys("{dnarrow}{home}")
```

Verlust.Bürokosten:

```
@performkeys("{tab}{leftarrow}{leftarrow}{f8}{tab}{tab}{tab}{tab}{tab}{return}"),
@performkeys("{leftarrow}"),
@performkeys("{f2}" & "Brutto*1" & "{return}"),
@performkeys("{leftarrow}"),
@performkeys("{f2}" & "(Anteilig/114*14)/100*25" & "{return}"),
@performkeys("{leftarrow}"),
@performkeys("{f2}" & "(Anteilig/114*100)/100*25" & "{return}"),
@performkeys("{leftarrow}{leftarrow}{del}"),
@performkeys("{f2}" & "Anteilig/100*25" & "{return}"),
@performkeys("{dnarrow}{home}")
```

Betrachten wir die Formeln im einzelnen, wobei wir logisch vorgehen, nicht streng chronologisch. Die Formel des umfassenden Frames VERLUST ist nur kurz:

```
@printreturn,
@setmacro({alt-v},Verlust.start),
@verlust.start
```

An den Anfang eines Programmes setzt man üblicherweise das Kommando **@printreturn** - eigentlich eine Funktion, die zum Abbruch eines Druckvorganges benutzt wird und hier dem Gegenteil dient: Die Anweisung ermöglicht den Ausdruck des Frames und seiner Formeln, ohne daß die Formeln während des Druckes neu berechnet, also ausgeführt werden. Ohne **@printreturn** wird beim Druck plötzlich das Menü aktiviert.

Die nächste Zeile initialisiert noch einmal das Makro, damit Jonathan Kraft innerhalb des Programmes mit **[ALT]-V** jederzeit wieder zu seinem Hauptmenü zurückkommt. So dann wird die Formel **@verlust.start** ausgeführt.

Die gibt es im FRED-Befehlsspektrum natürlich nicht, es ist eine sogenannte benutzerdefinierte Funktion: Das Programm wird angewiesen, die Formel abzuarbeiten, die im Frame **VERLUST.START** hinterlegt ist. Nämlich:

```
@keyfilter({all}),
@menu(Menü)
```

Ein Teil des Gesamtprogrammes verwendet einen Tastaturfilter. Die Anweisung **@keyfilter({all})** setzt diesen Tastaturfilter (und alle anderen eventuell bestehenden) zurück - auf diese Sache wird noch einzugehen sein.

Ein kleines Start-Programm wie dieses ist ebenfalls empfehlenswert. Darenin kann man alles packen, was vor dem Aufruf des eigentlichen Programmes notwendig oder nützlich ist. Zum Beispiel läßt sich damit der Bildschirm bereinigen (mit *Arbeitsfläche räumen* aus dem Menü *Laufwerk*), lassen sich eventuell benötigte Frames automatisch laden - ehe dann mit **@menu(Menü)** das Menü aufgerufen wird.

Tip: Werden Programme von mehreren Anwendern benutzt, ist ein Info-Frame mit Angaben über den Programmaufbau hilfreich, den man am besten in das Start-Programm einbaut.

Das FRED-Kommando **@menu** braucht als Parameter lediglich den Namen desjenigen Frames, der die Schachtel für die Menü-Punkte bildet - **MENÜ** in diesem Fall - und der selber keine Formeln enthält: Seine Daseinsberechtigung erschöpft sich darin, ein umfassender (Menü-) Frame zu sein.

Bei der Form des vertikal angeordneten Menüs, wie es Jonathan Kraft verwendet, können die Menü-Punkte (die ja die Namen der Sub-Frames sind) mit zusätzlichen Erläuterungen versehen werden - wiederum insbesondere dann nützlich, wenn ein Programm von mehreren Anwendern benutzt wird.

Zur Erinnerung: Soll ein Frame sowohl Text (im Frame) wie Formeln (hinter dem Frame) aufnehmen, muß zuerst der Text eingegeben werden.

Diese Zusatzinformationen werden als normaler Text eingegeben - nachdem Jonathan Kraft den Frame mit **[F9]** (*Zoom*) zuvor geöffnet hat. Mehr als eine Zeile indessen ver-

kraftet Framework nicht, wobei diese dann später, wenn das Menü aufgerufen wird, automatisch zentriert, der Text also auf Mitte gestellt wird.

Tip: Wählen Sie die Menü-Punkte (also die Namen der Menü-Sub-Frames) so, daß sie auch mit ihrem ersten Buchstaben aufgerufen werden können.

Wie ein Framework-Menü kann auch ein selber programmiertes Menü aus Unter-Menüs bestehen - und dieses sich wiederum teilen in Unter-Unter-Menüs. Jonathan Kraft hätte dazu nicht greifen müssen, seine insgesamt fünf Menü-Punkte wären ohne weiteres auch in einem einzigen Menü unterzubringen gewesen.

Das Unter-Menü dient hier vornehmlich Demonstrationszwecken, folgt indessen auch einer grundsätzlichen Überlegung. Daß die Eingabe der Daten vom Hauptmenü aus zu erreichen sein soll, ist klar. Ebenso die Gesamtbilanz: Die Aufrechnung der Einnahmen und Ausgaben zum jeweils aktuellen Stand wird sich Jonathan Kraft des öfteren anschauen.

Zur eingehenderen Analyse jedoch, die sich hinter dem Punkt *Auswertung* verbirgt, wird er seltener greifen - darum wurde sie in einem Unter-Menü versteckt. Zudem sind noch andere Analysemöglichkeiten denkbar, als sie hier jetzt vorgestellt werden - Jonathan Kraft kann also sein Unter-Menü *Auswertung* noch beliebig erweitern und belastet dadurch sein Hauptmenü nicht. Denn irgendwann wird es unübersichtlich und stößt dann auch an die Aufnahmefähigkeit des Bildschirmes.

Tip: Fassen Sie logische Arbeiten in Sub-Menüs zusammen.

Ein Punkt ist in jeder Art von Menü unabdinglich: Irgendwie muß man ja wieder 'raus aus dem Menü. Aus dem Unter-Menü *Auswertung* führt der Weg sinnvollerweise zurück zum Hauptmenü:

```
@Verlust.start,  
@quitmenu
```

@**Verlust.start** verweist wieder auf die Start-Routine, die ihrerseits das Hauptmenü aktiviert. @**quitmenu** ist die übliche (und notwendige) Anweisung zum Verlassen eines Menüs.

Sie taucht darum auch in der Formel im Menü-Punkt *Ende* auf, mit dem man das gesamte Programm verläßt und wieder zu Framework zurückkehrt:

```
@keyfilter({all}),  
@setselection("Verlust"),  
@performkeys("{f10}"),  
@performkeys("{ctrl-return}"),  
@performkeys("{return}"),  
@quitmenu
```

Zur Sicherheit werden mit `@keyfilter({all})` alle Tastaturfilter außer Kraft gesetzt. Sodann benutzt die Formel eine Anweisung, die in Programmen häufig benutzt wird und derentwegen Jonathan Kraft sein Programm VERLUST als Makro definieren mußte: `@setselection("Verlust")`.

`@setselection` wählt den in Klammern angegebenen Frame aus, macht ihn also zum aktuellen Frame: Der Cursor springt gleichsam auf dessen Rand - mit diesem Frame kann nunmehr gearbeitet werden. Jonathan Kraft macht folgendes: Er schaltet zunächst um auf die Konzept-Version (was entbehrlich wäre), sichert dann mit `@performkeys("{ctrl-return}")` seine Arbeit (was ratsam ist) und schließt dann den Frame - er wird im Dokumentenstapel abgelegt, die Arbeit ist beendet, der Bildschirm bereinigt.

■ | **Zur Erinnerung:** `@setselection` kann nur innerhalb eines Makros verwendet werden.

Der Ausstieg aus dem Programm ließe sich noch weiter ausbauen. Zum Beispiel könnte Jonathan Kraft mit

```
@performkeys("{ctrl-l}w")
```

den Frame gleich aus dem Dokumentenstapel verschwinden lassen (wodurch dann die vorherige Speicher-Anweisung überflüssig wäre). Den gleichen Effekt erreicht er auch mit

```
@performkeys("{del}")
```

Das allerdings ist eine gefährliche Sache. Wehe, der Frame ist nicht gesichert! Dann ist die ganze Arbeit umsonst gewesen - alles futsch.

■ | **Tip:** Lassen Sie ein Programm sich selber nur löschen, wenn es keine (veränderbaren) Daten beinhaltet.

Daß Jonathan Kraft weder auf die eine oder andere Weise seine Überschuß-Berechnung aus dem Dokumentenstapel wegräumt, mag mit seiner notorischen Faulheit zu tun haben. Vielleicht, denkt er, braucht er das Programm gleich nochmals (wenn die Post den dicken Scheck seines Verlegers bringt ...) und muß es dann nicht erneut laden. Daß er damit einen Großteil seines Arbeitsspeichers blockiert, kommt ihm freilich nicht in den Kopf.

Schreiten wir zur Dateneingabe. Alle notwendigen Tabellen, Datenbanken und dergleichen sind Teil des umfassenden Frames VERLUST - klar, so ist alles beisammen und braucht nicht eigens aufgerufen werden. Sie sind allerdings nicht in das Menü selber integriert, aus ebenfalls einleuchtenden Gründen: Das Menü setzt Aktionen in Gang - zum Beispiel, um in die Datenbank Daten einzugeben.

Die Datenbank ist im Vergleich zur ersten Fassung von Jonathan Krafts Überschuß-Berechnung beträchtlich reduziert worden. Sie besteht nur noch aus Spalten für den Monat, für den zu erfassenden Gegenstand, für Brutto- und Netto-Beträge sowie die

Mehrwertsteuer, und außerdem aus den vorerst noch ominösen Feldern *Kategorie* und *Gesamtwert*.

Laufwerk NEU Editieren Suchen Frames Text Zahlen Grafik Drucken 23:19:20

Monat	Gegenstand	Brutto	Kategorie	Netto	Mwst	Gesamtwert
2	Bürobedarf	160,05	B	140,39	19,66	
2	Autowäsche	9,60	A	8,42	1,18	
3	Benzin	20,48	A	17,96	2,52	25,60
3	Benzin	12,61	A	11,06	1,55	15,76
3	EZ	807,63	Z	754,79	52,84	
2	EZ	333	Z	311,21	21,79	
1	Porto	723,50	P	723,50	0	
1	Porto	20	P	20	0	
2	Disketten	80,94	B	71	9,94	
3	Disketten	447,56	B	392,60	54,96	
3	ABC Verlag	41,84	V	39,10	2,74	
1	Ultimo Verlag	5350	V	5000	350	
2	PC Magazin	150	Z	140,19	9,81	
2	Aichwalder Nachricht	63,45	Z	59,30	4,15	
1	Schanbacher Zeitung	192,60	Z	180	12,60	

[bild34-6] Dok.: 1/1

Bild 34.6: Die Datentabelle

Wenden wir uns aber zuerst der Formel zu, die hinter dem Menü-Punkt *Dateneingabe* (nicht hinter der Datenbank!) steckt:

```
@echo(#OFF),
@execute(Verlust.Datentabelle,#TRUE),
@setselection("Verlust.Datentabelle"),
@performkeys("{f9}"),
@performkeys("{ctrl-home}{ctrl-dnarrow}{dnarrow}"),
@echo(#ON),
@keyfilter({ctrl-z},Tastaturfilter),
@quitmenu
```

@echo(#OFF) schaltet die Bildschirmanzeige der nachfolgenden Operationen ab, sie verlaufen gleichsam im Hintergrund - schont die Augen, macht das Programm ein wenig schneller, aber manchmal kann es auch ganz spaßig sein, Framework bei der Arbeit zuzusehen. @echo(#ON) knipst, korrespondierend dazu, die Bildschirmanzeige wieder an.

Tip: Achten Sie darauf, daß @echo(#OFF) an der richtigen Stelle durch @echo(#ON) wieder aufgehoben wird - sonst sehen Sie gar nichts mehr.

Die nächste FRED-Funktion, `@execute`, weist einem Frame, einer Zelle (in einer Tabelle) oder einem Feld (in einer Datenbank) eine Formel zu und führt sie sogleich aus - im Gegensatz zu der Funktion `@setformula`, die eine Formel lediglich zuschreibt, ohne sie auszuführen.

Die allgemeine Schreibweise lautet:

```
@execute(Bezug,Formel)
```

Als Bezug steht in diesem Fall der Frame `VERLUST.DATENTABELLE` - der gesamte Pfad ist hier nicht unbedingt notwendig, schadet indessen nicht. Die Formel besteht aus der FRED-Konstanten `#TRUE` und ist höchst trickreich: Sie bewirkt, daß, wie es im spröden Framework-Jargon heißt, von der Untermenge auf die Gesamtmenge der Sätze umgeschaltet wird. Zu deutsch: Ist die Datenbank selektiert worden, wird sie nunmehr wieder als Ganzes gezeigt - die Formel bewirkt das gleiche wie der Befehl *Öffnen alle* im Menü *Frames*. Und in der Tat ist die Chance groß, daß durch die anderen Module des Programmes `VERLUST` die Datenbank lediglich in selektierter Form vorliegt - das wird durch `@execute(Verlust.Datentabelle,#TRUE)` bereinigt.

■ **Merke:** Bei `@setselection` muß der Frame-Name in Anführungszeichen stehen.

Nun wird wieder mit `@setselection` ein Frame als der aktuelle ausgewählt, nämlich die Datenbank. Hier allerdings ist die gesamte Pfadangabe erforderlich, weil die Formel und die Datenbank sich nicht auf der gleichen Frame-Ebene befinden. Gibt Jonathan Kraft nur `@setselection("Datentabelle")` an, handelt er sich eine Fehlermeldung ein:

Nicht definierter Bezug

Solche Kleinigkeiten bringen unter Umständen ein Programm derart durcheinander, daß es abstürzt - Framework muß neu gestartet werden, was auch bedeutet, daß alles, was zu dem Zeitpunkt noch nicht gesichert war, verloren ist.

Hat nun also Jonathan Kraft glücklich die Datenbank angewählt, öffnet er sie mit `@performkeys("{f9}")` und läßt den Cursor ein wenig umherhüpfen:

- zunächst mit `{ctrl-home}` zum ersten Feld im ersten Satz
- dann mit `{ctrl-dnarrow}` zum letzten belegten Satz
- dann mit `{dnarrow}` eine Zeile abwärts.

Jonathan Kraft will damit erreichen, daß der Cursor, wenn er die Dateneingabe aufruft, im ersten Feld des nächsten noch freien Datensatzes steht und er gleich loslegen kann. Weil die meisten Cursorbewegungen innerhalb einer Datenbank (oder einer Tabelle) relativ zum Standort des Cursors erfolgen und er nicht gewiß sein kann, wo sein Cursor sich gerade befindet, steuert er zunächst einen sicheren Ausgangspunkt an, das erste Feld im ersten Satz, und jongliert von dort aus weiter.

■ **Tip:** Gehen Sie bei programmierten Cursorbewegungen stets von einem eindeutig zu bestimmenden Startpunkt aus.

Und nun kommt der Tastaturfilter ins Spiel, von dem schon des öfteren die Rede war:

```
@keyfilter({ctrl-z},Tastaturfilter)
```

Die Funktion **@keyfilter** verlangt zwei Angaben:

- welche Tasteneingaben abgefangen werden sollen
- was mit ihnen (oder besser: an ihrer Stelle) zu geschehen hat.

Das hat Jonathan Kraft in dem Frame *Tastaturfilter* abgelegt und definiert. Er hätte es auch gleich in der **@keyfilter**-Formel angeben können. Das hätte aber dann nur für diesen einen Fall gegolten. So indes kann er, wann's ihm beliebt, auf den Tastaturfilter zurückgreifen - auch in anderen Modulen seines Programmes.

Zunächst aber hat Jonathan Kraft angeben müssen, welche Tasten er herausgefiltert haben möchte.

Framework unterteilt die gesamte Tastatur in sechs Bereiche:

- die Funktionstasten
- die Sondertasten (also alle Tasten, die nichts schreiben, sondern eine Aktion bewirken - die Cursortasten, die Leertaste, die Tabulator-Taste usw.)
- die alphanumerischen Tasten
- die Makro-Tasten (also alle Kombinationen mit **[ALT]**)
- die Taste **[CTRL]** in Verbindung mit einem Buchstaben
- die Taste **[CTRL]** in Verbindung mit einer Ziffer

Die Besonderheit dabei ist, daß lediglich die ersten beiden Bereiche, die Funktions- und die Sondertasten, individuell herauszufiltern sind, man also gezielt etwa der Funktionstaste **[F1]** eine andere Bedeutung zuschreiben kann.

Alle anderen Bereiche sind nur als Gruppe ansprechbar. Jonathan Kraft hat geschrieben:

```
@keyfilter({ctrl-z},Tastaturfilter)
```

Genausogut könnte die Formel lauten:

```
@keyfilter({ctrl-a},Tastaturfilter)
```

Es macht keinen Unterschied: Wird irgendeine Buchstabentaste in Verbindung mit **[CTRL]** gedrückt, führt sie nur das aus, was im Frame *Tastaturfilter* hinterlegt ist. Welchen Buchstaben Jonathan Kraft als **@keyfilter**-Parameter angibt, ist völlig unerheblich: Alle sind davon betroffen.

Framework verfügt hierfür noch über eine andere Anweisung: die Tastaturbezeichnung `{char}`

filtert alle druckbaren Zeichen und wird mit Vorliebe angewandt, wenn der Bereich der alphanumerischen Tasten angesprochen ist.

Und der Gipfel aller Tastaturfilter ist

```
@keyfilter({all},Tastaturfilter)
```

Damit ist die gesamte Tastatur zunächst einmal außer Betrieb, und jede einzelne Taste kann in dem dafür vorgesehenen Frame (der hier *Tastaturfilter* benannt ist, aber natürlich auch anders heißen kann) neu definiert werden. Ohne die Angabe des Frames hebt diese Anweisung alle vorhandenen Tastaturfilter auf - darauf hat Jonathan Kraft ja auch schon mehrmals zurückgegriffen.

Tip: Wollen Sie mit dem *{all}*-Parameter die gesamte Tastatur filtern, dann erstellen Sie erst den Frame mit den Filterformeln, ehe Sie die *@keyfilter*-Anweisung geben. Sonst kann es passieren, daß mit der Tastatur überhaupt nichts mehr geht.

So weit geht Jonathan Kraft nicht. Er beschränkt sich auf **CTRL**-Kombinationen, und was mit ihnen bewirkt wird, war beim Rundgang durch das Programm schon zu sehen: Sie berechnen aus dem eingegebenen Brutto-Betrag den Netto-Wert und die Mehrwertsteuer und schreiben beides in die entsprechenden Felder.

Die Formeln, wie gesagt, sind im Frame *Tastaturfilter* hinterlegt und sehen beispielsweise so aus (ich führe hier nicht alle auf):

```
;Honorar Zeitungen (7%)
@if(@key={ctrl-z},
    @list(@performkeys("Z"),@mw7)
),

;Ausgaben Bürobedarf (14%)
@if(@key={ctrl-b},
    @list(@performkeys("B"),@mw14)
),

;Ausgaben Porto
@if(@key={ctrl-p},
    @list(@performkeys("P"),@mw0)
),

;Ausgaben Auto (14%); anteilig 80%
@if(@key={ctrl-a},
    @list(@performkeys("A"),@auto)
),

;Ausgaben Bürokosten (14%); anteilig 25 %
@if(@key={ctrl-h},
    @list(@performkeys("H"),@Bürokosten)
),
```

Die Schreibweise scheint etwas komisch: Eine Klammer und ein Komma einsam in einer Zeile für sich. Das indessen dient der Übersichtlichkeit (ebenso wie der Einzugs

der jeweils zweiten Programmzeile): Die Klammer ist die schließende Klammer zu `@if` - so vergißt man sie am wenigsten.

■ **Tip:** Unterteilen Sie Formeln nach logischen Gesichtspunkten in mehrere Zeilen.

Im Gegensatz zu seinem ersten Überschuß-Berechnungsversuch trennt Jonathan Kraft in seinem VERLUST-Programm nicht zwischen Einnahmen und Ausgaben, sondern lediglich zwischen Sachgruppen und unterschiedlichen Mehrwertsteuersätzen.

Jede Sachgruppe bekommt eine `CTRL`-Kombination, zum Beispiel:

- `CTRL`-Z für die Einnahmen Zeitungshonorare
- `CTRL`-V für die Einnahmen Verlagshonorare
- `CTRL`-B für die Ausgaben Bürobedarf
- `CTRL`-A für die Ausgaben Auto

Nutzt Jonathan Kraft das gesamte Alphabet aus, kann er 26 Sachgruppen bilden - das ist hinreichend genug für seine Bedürfnisse.

Das Programm, angewiesen, alle `CTRL`-Kombinationen zu filtern, durchsucht alle entsprechenden Tasteneingaben nach den Bedingungen, die in *Tastaturfilter* aufgestellt sind:

```
@if(@key={ctrl-z},
    @list(@performkeys("Z"),@mw7)
),
```

Solche `@if`-Bedingungen werden in Programmen (nicht nur bei Framework) häufig aufgestellt. Die allgemeine Form einer Bedingung lautet:

```
@if(wenn - dann - sonst)
```

Auf unseren Fall bezogen: *wenn* die Tastenkombination `CTRL`-Z ist, *dann* mache `@list(@performkeys("B"),@mw14)`, *sonst* (wenn also die Bedingung nicht erfüllt wird) - das fehlt bei Jonathan Kraft; er braucht es nicht: wenn die Tastenkombination nicht `CTRL`-Z ist, dann ist sie vielleicht `CTRL`-V oder `CTRL`-A oder eine andere, für die Jonathan Kraft eine Formel definiert hat, und alles weitere interessiert ihn nicht.

Eine `@if`-Bedingung kann nur drei Parameter haben, und jeder Parameter hat seine feste Bedeutung: entweder *wenn* oder *dann* oder *sonst*. Das bringt Jonathan Kraft in Bedrängnis. Wenn die Bedingung erfüllt ist, soll die Formel zweierlei tun: den Buchstaben Z schreiben (`@performkeys("Z")`) und die Formel `@mw7` ausführen.

Damit das überhaupt möglich wird, hat FRED die Funktion `@list`. Was hinter ihr in Klammern angegeben ist, wird nacheinander abgearbeitet - Framework tut so, als sei dies ein einziger Parameter. Das heißt: Der *dann*-Parameter ist in diesem Fall `@list`. Ohne `@list` bekäme `@mw7` die Bedeutung *sonst*.

Der Tastaturfilter arbeitet wieder mit einigen benutzerdefinierten Formeln, die jeweils in einem eigenen Frame abgelegt sind:

- @mw7
- @mw14
- @mw0
- @Auto
- @Bürokosten

@mw7 beispielsweise sieht so aus:

```
@performkeys("{tab}"),
@performkeys("{f2}" & "Brutto/107*100" & "{return}"),
@performkeys("{tab}"),
@performkeys("{f2}" & "Brutto/107*7" & "{return}"),
@performkeys("{tab}{return}{home}")
```

Der gesamte Tastaturfilter macht also folgendes: Wenn die Tastenkombination **CTRL-Z** ist, dann wird

- wird in das aktuelle Feld (»Kategorie«) der Buchstabe Z geschrieben
- springt der Cursor in das nächste Feld (»Netto«)
- wird dort der Formelspeicher geöffnet, die Formel

Brutto/107*100

eingetragen und sogleich mit **RETURN** ausgeführt

Tip: Wenn ein eindeutiger Bezug auf ein Feld oder eine Zelle möglich ist, dann arbeitet die Formelzuweisung mit **@execute** oder **@setformula** schneller.

- springt der Cursor in das nächste Feld (»Mwst«)
- wird dort der Formelspeicher geöffnet, die Formel

Brutto/107*7

eingetragen und sogleich mit **RETURN** ausgeführt

- springt der Cursor zurück in das erste Feld der nächsten Zeile.

@mw14 und **@mw0** veranlassen die gleichen Aktionen, nur sind die Mehrwertsteuersätze anders.

@Auto und **@Bürokosten** sind Sonderfälle und darum etwas komplizierter. Beidesmal darf Jonathan Kraft nicht seine gesamten Aufwendungen als Betriebsausgaben absetzen, sondern nur einen gewissen Anteil.

Die Formeln rechnen diesen Anteil aus, schreiben aber zusätzlich den Gesamtwert in eine eigene Spalte der Datentabelle - als kleine Sicherheit, damit Jonathan Kraft den Beleg nicht mühsam heraussuchen muß, falls er irgendwann den Ausgangswert braucht.

Der nötigen Berechnungen wegen hat Jonathan Kraft in die Datentabelle noch eine Spalte namens »Anteilig« als Zwischenablage eingebaut, sie aber mit **F4** (Größe) so verkleinert, daß sie nicht zu sehen ist. (Wenn der Cursor im Feld »Gesamtwert« steht, taucht der Name der versteckten Spalte in der Statuszeile auf.)

■ **Tip:** Verstecken Sie Felder, die nur für Zwischenergebnisse notwendig sind.

Die Formel für @Auto:

```
@performkeys("{tab}{leftarrow}{leftarrow}{f8}{tab}{tab}{tab}
{tab}{tab}{return}"),
@performkeys("{leftarrow}"),
@performkeys("{f2}" & "Brutto*1" & "{return}"),
@performkeys("{leftarrow}"),
@performkeys("{f2}" & "(Anteilig/114*14)/100*80" & "{return}"),
@performkeys("{leftarrow}"),
@performkeys("{f2}" & "(Anteilig/114*100)/100*80" &
"{return}"),
@performkeys("{leftarrow}{leftarrow}{del}"),
@performkeys("{f2}" & "Anteilig/100*80" & "{return}"),
@performkeys("{dnarrow}{home}")
```

Der Cursor springt dabei wie wild umher:

- Der Gesamtbetrag wird in die Spalte »Gesamtwert« kopiert.
- Aufgrund der entsprechenden Formel übernimmt »Anteilig« den Wert von »Brutto« - derzeit noch identisch mit dem Gesamtwert.
- Nettobetrag und Mehrwertsteuer werden anteilig berechnet.
- Der Brutto-Betrag (noch der Gesamtbetrag) wird gelöscht und durch den anteiligen Brutto-Betrag ersetzt. Die Formel bezieht sich dabei auf das Feld »Anteilig«, und weil sich das wiederum auf »Brutto« bezieht, steht dort jetzt auch der anteilige Brutto-Betrag.
- Zuletzt wird der Cursor wieder ins erste Feld des nächsten Datensatzes geführt.

Warum werden diese Formeln in eigenen Frames abgelegt und nicht im *Tastaturfilter*? Das ist keine Vorschrift, aber praktisch aus mehreren Gründen.

Zunächst: Jonathan Kraft erspart sich Tipparbeit. Die Berechnungsformeln sind ja immer gleich, lediglich die Mehrwertsteuersätze oder der absetzbare Anteil schwanken. Also schreibt Jonathan Kraft seine Formeln nur einmal und weist das Programm an, die Formel sich doch, bitteschön, dort abzuholen.

■ **Zur Erinnerung:** Mit dem Makro-Funktionen **ALT F5** und **ALT F6** aus MACLIB kann man markierten Text zwischenspeichern und wieder abrufen.

Zudem spart Jonathan Kraft Speicherplatz. Das ist ja einsichtig - und bedeutsam: Das Gesamtprogramm VERLUST umfaßt etwa 39000 Byte inklusive der Beispieldaten. Derer sind's aber nur wenige, im praktischen Gebrauch hingegen wachsen sie beängstigend schnell an. Und da spart man, wo's nur geht.

Und schließlich: Allfällige Änderungen der Mehrwertsteuer übersteht Jonathan Kraft gelassen. Er braucht sie nur in die Mehrwertsteuer-Frames einzuarbeiten - und nicht einmal den Namen zu ändern. Denn die Namen haben mit den tatsächlichen Sätzen ja nur gedanklich, nicht jedoch inhaltlich zu tun.

Mit einem Tastaturfilter dieser Art handelt sich Jonathan Kraft jedoch auch Nachteile ein. Weil ja alle **CTRL**-Kombinationen abgefangen werden, kann er die Menüs nur noch mit **INS** öffnen - er nimmt's in Kauf. Welche anderen Möglichkeiten blieben ihm? Er könnte **ALT**-Kombinationen filtern - das verschließt ihm jedoch die Makros. Oder er müßte in seinem Tastaturfilter jede Menge Ausnahmen definieren. Zum Beispiel so:

```
@if(@key={ctrl-l},@performkeys("{ctrl-l}"))
```

CTRL-L wird damit wieder zu **CTRL**-L. Das jedoch sperrt ihm insgesamt neun Buchstaben. Ob das besser ist?

Jedenfalls hat Jonathan Kraft jetzt seine Dateneingabe programmiert und automatisiert und kann zum nächsten Programmpunkt schreiten: der Bilanz.

	A	B
1		Ausgaben
2	Brutto	2988,49
3	Netto	2841,92
4		
5		
6		Einnahmen
7	Brutto	9748,29
8	Netto	9125,49
9		
10		
11		Bilanz
12		
13	Brutto/Netto	6283,57
14	Gewinn brutto	6759,80
15	Mwst	476,23

Bild 34.7: Die Bilanztafel

Die Formel des entsprechenden Menü-Punktes ist einfach. Sie macht die Bilanztafel - ebenso wie die Datentafel ein eigener Frame - zum aktuellen Frame, berechnet sie neu und öffnet sie sodann:

```
@setselection("Verlust.Bilanztafel"),
@performkeys("{f5}"),
@performkeys("{f9}"),
@quitmenu
```


Die eigentliche Arbeit erledigen die Formeln in der Bilanztabelle: Sie holen sich die Werte aus der Datentabelle. In der Zelle B2 (Ausgaben Brutto) sieht die Formel so aus:

```
@execute(Datentabelle,#TRUE),
@execute(Datentabelle,@or(Kategorie="B",Kategorie="P",Kategorie
="D",Kategorie="A",Kategorie="M",Kategorie="W",Kategorie="H",
Kategorie="F",Kategorie="X")),
@sum(Datentabelle.Brutto)
```

Wieder wird hier die Datentabelle mit **@execute(Datentabelle,#TRUE)** zunächst bereinigt, so daß alle Datensätze angezeigt werden - damit sie im folgenden neu selektiert werden können.

@execute weist diesmal tatsächlich eine Formel zu, nämlich eine Selektionsbedingung. Aus der Datentabelle werden all jene Sätze herausgesucht, die in den Bereich der Ausgaben fallen, sodann werden deren Werte mit der **@sum**-Anweisung addiert.

Selektionskriterium ist das Kürzel im Feld »Kategorie«, das die Daten bestimmten Sachgruppen zuordnet. Obschon es kein Problem gewesen wäre, diese Sachgruppenbezeichnungen mit Hilfe der Tastaturfilter-Formeln auszuschreiben, hat es Jonathan Kraft beim Kürzel belassen: Das spart wiederum Speicherplatz und Tipparbeit - zum Beispiel in dieser Formel.

Tip: Reduzieren Sie Zeichenausgaben, wo es nur geht. Das macht die Programme schneller.

Nochmals eine Ersparnis. Für die Zelle B3 (Ausgaben Netto) braucht Jonathan Kraft die Datentabelle nicht erneut zu selektieren - sie ist's bereits, zwischenzeitlich ist keine andere Selektion unternommen worden, also genügt die bloße Additionsfunktion:

```
@sum(Datentabelle.Netto)
```

Für die Einnahmen allerdings muß neu selektiert werden, nach anderen Sachgruppen, und deshalb auch die Datentabelle zuvor wieder bereinigt werden:

```
@execute(Datentabelle,#TRUE),
@execute(Datentabelle,@or(Kategorie="Z",Kategorie="R",
Kategorie="V",Kategorie="S")),
@sum(Datentabelle.Brutto)
```

Die übrigen Formeln in der Bilanztabelle folgen dem gleichen Schema bzw. sind einfache Summierungen.

Auch der Teil *Auswertung nach Sachgruppen* im Untermenü *Auswertung* ist ähnlich aufgebaut - ist ja auch eine Art Bilanz, nur differenzierter.

	A	B	C	D	E
1 Einnahmen (brutto)					
2					
3		Januar	Februar	März	Gesamt
4					
5 Zeitsungen		392,60	546,45	807,63	1746,68
6 Verlage		5350	0	41,84	5391,84
7 Rundfunk					
8 Sonstiges					
9					
10					
11					

Bild 34.8: Die Sachgruppenauswertung

Demgemäß braucht die Formel im Menü lediglich den Cursor auf die Tabelle »Sachgruppenauswertung« zu setzen, sie neu zu berechnen und zu öffnen:

```
@setselection("Verlust.Sachgruppenauswertung"),
@performkeys("{f5}"),
@performkeys("{f9}"),
@quitmenu
```

In der Tabelle selber wird wieder selektiert, diesmal nach Sachgruppen und nach Monaten, bilanziert - und zuvor natürlich die Datentabelle bereinigt. In der Zelle B5 (Zeitsungen Januar) lautet die Formel demgemäß:

```
@execute(Datentabelle,#TRUE),
@execute(Datentabelle,@and(Kategorie="Z",Monat=1)),
@sum(Datentabelle.Brutto)
```

Um einiges kniffliger ist die *Individuelle Auswertung*. Hier geht's wirklich zur Sache: Jonathan Kraft möchte detailliert erfahren, was er wo ausgegeben oder eingenommen hat. Vorgegebene Kriterien scheiden aus. Bei der Dateneingabe kann Jonathan Kraft zu Bereichen zusammenfassen - »Bürobedarf« etwa. Was er hingegen an Bürobedarf im einzelnen benötigt - Papier, Bleistift, Klebstoff, Disketten -, ist im vorhinein nicht festzulegen, das muß bei der Analyse abgefragt werden.

Tip: Achten Sie bei der Dateneingabe auf Tippfehler. Sonst können Sie diese Datensätze nicht selektieren.

Das Programm für die *Individuelle Auswertung* komplett:

```
@local(Kriterium,Ergebnis),
Kriterium:=@inputline("Welche Sachgruppe soll ausgewertet werden?"),
@echo(#OFF),
@execute(Verlust.Datentabelle,#TRUE),
@setselection("Verlust.Datentabelle"),
@performkeys("{f2}{home}{f6}{end}{del}"),
@performkeys("Gegenstand=" & "" & Kriterium & "" &
"{return}"),
Ergebnis:=@currency(@sum(Datentabelle.Brutto)),
@setselection("Individualauswertung"),
@performkeys("{f9}{ctrl-end}" & Kriterium & ":" & "{tab}" &
Ergebnis & "{return}"),
@echo(#ON),
@quitmenu
```

Ein Programmaufbau folgt einer eigenen Logik, die ein Problem nach bestimmten Regeln löst. Für Jonathan Kraft war die Problemstellung etwa derart:

- Er möchte wissen, wieviel er bis dato für eine bestimmte Sachgruppe ausgegeben hat, zum Beispiel für Disketten. Anders formuliert: Die Datentabelle muß nach einem frei definierten Kriterium - zum Beispiel »Disketten« - selektiert und anschließend summiert werden.
- Er möchte seine Ausgaben aufgelistet haben. Anders gesagt: Das Ergebnis des Selektions- und Additionsvorgangs muß in einen Frame ausgegeben werden.

Dafür ist der Frame *Individualauswertung* vorgesehen - ein reiner Text-Frame. Der wird nicht vom Programm jedesmal neu erstellt (das wäre Jonathan Kraft zu zeitaufwendig), sondern ist ständig vorhanden.

Zunächst aber muß Jonathan Kraft seinem Programm sagen, welche Sachgruppe er ausgewertet haben möchte:

```
Kriterium:=@inputline("Welche Sachgruppe soll ausgewertet werden?"),
```

@inputline ist eine FRED-Funktion, die den in Klammern stehenden Text in der Nachrichtenzeile ausgibt - wie mit der Funktion **@prompt**. Im Unterschied dazu aber erwartet **@inputline** eine Eingabe, die mit **RETURN** abgeschlossen und sodann wieder ausgegeben wird.

Tip: Mit **@inputline** kann auch ein Text vorgegeben werden, der vom Benutzer editiert wird - siehe im Referenz-Handbuch auf Seite 11-51.

Wo? Steht eine **@inputline**-Anweisung »hinter« einem leeren Text-Frame, wird die Eingabe in den Text-Frame geschrieben. Genausogut kann man die Ausgabe der Antwort umlenken, in einen anderen Frame beispielsweise.

Eine Art Umlenkung hat auch Jonathan Kraft in seinem Programm festgelegt, bloß ist der Ausgabeort kein Frame, sondern eine vom ihm bestimmte Variable namens **Kriterium**. Was er als Antwort auf die Frage in der Nachrichtenzeile angibt, zum Beispiel »Disketten«, wird also zwischengespeichert und kann weiterverarbeitet werden.

Solche Variablen, die nur für dieses Programm gelten, sonst in keinem, können in beliebiger Anzahl und mit beliebigem Namen definiert werden - stets allerdings, bevor sie benutzt werden:

```
@local(Kriterium,Ergebnis),
```

Im Verlauf des Programmes wird den Variablen dann ein Wert zugewiesen, wozu das Zeichen

```
:=
```

dient. Links davon steht die Variable, rechts die Formel oder Funktion, die den Wert ermittelt - Jonathan Kraft kennt das schon aus den Formelzuweisungen in einer Datenbank (Kapitel 24) und weiß daher auch, daß die gleiche Wirkung mit der Funktion **@set** erzielt werden kann. Die Formel hieße in diesem Fall:

```
@set(Kriterium,@inputline("Welche Sachgruppe soll ausgewertet werden?"))
```

Was in der Variablen **Kriterium** gespeichert wird, ist das Selektionskriterium für die Datenbank - die zunächst wieder bereinigt und dann zum aktuellen Frame gemacht wird:

```
@execute(Verlust.Datentabelle,#TRUE),  
@setselection("Verlust.Datentabelle"),
```

Dann ergibt sich jedoch ein Problem. Im Formelspeicher der Datentabelle steht nunmehr der Wert **#TRUE**, und was immer als neues Selektionskriterium angegeben wird, überschreibt diesen Wert nicht, sondern wird im vorangestellt - weil in der Regel der Einfügemodus aktiviert ist.

Zur Erinnerung: Im Menü *Editieren* kann von Einfügen auf Überschreiben umgeschaltet werden.

Deshalb muß zunächst alles, was im Formelspeicher steht (zur Vorsicht: tatsächlich alles), markiert und gelöscht werden, dann erst wird die neue Formel eingetragen:

```
@performkeys("{f2}{home}{f6}{end}{del}"),  
@performkeys("Gegenstand=" & "" & Kriterium & "" & "{return}"),
```

@performkeys übernimmt den Wert, der in der Variablen **Kriterium** abgelegt ist, also den Gegenstand, nach dem selektiert wird. Deshalb muß der Name der Variablen in der **@performkeys**-Anweisung nicht in Anführungszeichen eingeschlossen sein - dessen Wert in der Selektionsformel hingegen schon. Und weil alle Zeichen, die mit **@performkeys** geschrieben werden sollen, von Anführungszeichen umschlossen sein

müssen, häufen die sich, wenn das Zeichen selber eine Anführung ist. Das abschließende **{return}** sorgt dafür, daß die Formel, also die Selektion, gleich ausgeführt wird - möglich, weil der Formelbereich für diese kurze Eingabe mit **[F9]** (*Zoom*) nicht erweitert werden muß.

Sodann werden die selektierten Datensätze summiert, das Ergebnis jedoch wiederum in einer - anfangs definierten - Variablen gespeichert:

```
Ergebnis: = @currency(@sum(Datentabelle.Brutto)),
```

Jetzt muß nur noch der Text-Frame »Individualauswertung« angewählt und dort hinein alles geschrieben werden:

```
@setselection("Individualauswertung"),
@performkeys("{f9}{ctrl-end}" & Kriterium & ":" & "{tab}" &
Ergebnis & "{return}"),
```

@performkeys übernimmt wieder die Werte der Variablen **Kriterium** und **Ergebnis**.

Das läßt sich natürlich noch weiter ausbauen. Zum Beispiel kann dafür gesorgt werden, daß der Text-Frame zuvor gelöscht wird. Ebenso kann das Tagesdatum mit ausgegeben werden.

Überhaupt: Das Programm VERLUST ist keineswegs eine komplette Lösung. Jonathan Kraft kann es in jeder erdenklichen Weise differenzieren und vervollständigen - theoretisch sogar soweit, daß eine veritable Finanzbuchhaltung zustande kommt. Allerdings ist dann die Frage, ob der Programmieraufwand noch in vertretbarem Verhältnis zum Ergebnis steht. Mal abgesehen davon, was so ein Programm an Speicherplatz kostet ...

Jedenfalls: Jonathan Kraft hat jetzt einen ersten Einblick bekommen, wie das mit dem Programmieren ist. Und daß es ja gar nicht so schwer ist. Daß man daran sogar Geschmack finden kann.

Und der Dichter Jonathan Kraft fühlt sich an seinen großen Kollegen erinnert - na, wie hieß er noch gleich? "Die Geister, die ich rief ..."

* Wer kann das schon gewesen sein? Goethe natürlich. Übrigens zitiert das auch Jonathan Kraft bedauerlicherweise falsch. In Wirklichkeit heißt es: "Die ich rief, die Geister, werd' ich nun nicht los."

Arbeits- erleich- terungen

35 Startroutine

Beim Aufruf von Framework kann man, als sogenannte Befehlszeilenparameter, gleich eine Reihe von Dateien laden, indessen nur eine, die letzte nämlich, versehen mit dem Parameter /X, auch ausführen lassen. Da ist der Gedanke naheliegend, diese letzte ein Programm sein zu lassen, das ja seinerseits wieder andere Programme ausführen kann. Der /X-Parameter versetzt diese Datei gleichsam in einen Makro-Zustand, sämtliche FRED-Funktionen sind also wirksam.

■ **Hinweis:** LADEROUTINE.FW befindet sich auf der Beispieldiskette.

Eine solche Startroutine, hier LADEROUTINE.FW genannt, könnte etwa folgendermaßen aussehen:

```
@performkeys("{ctrl-l}hb:\maclib.fw{return)"),
@setselection("maclib"),
@performkeys("{F5}"),
@setselection("Laderoutine"),
@performkeys("{del}"),
@performkeys("{scroll-lock}{dnarrow}{return}{dnlevel}")
```

Zunächst wird die Makro-Bibliothek MACLIB geladen, neu berechnet, damit die Makros zur Verfügung stehen, sodann wird der Cursor auf LADEROUTINE selbst gesetzt und der Frame aus dem Arbeitsspeicher gelöscht - man braucht ihn jetzt nicht mehr, er nimmt nur Platz weg.

■ **Tip:** Der Frame LADEROUTINE muß geschlossen abgespeichert werden, sonst funktioniert die Löschung nicht.

Als nächstes wird auf das Dateiverzeichnis umgeschaltet - seltsamerweise spricht Framework beharrlich Laufwerk A an (vielleicht ist's bei Ihnen anders, müssen Sie ausprobieren), deshalb ein Schritt abwärts zum nächsten, richtigen Laufwerk. Das Dateiverzeichnis wird geöffnet, der Cursor gleich hineingeführt - alles bereit zum Arbeiten.

Natürlich kann man eine solche Startroutine noch ausbauen und alle Dateien laden lassen, die man so braucht. Und sie kann jederzeit abgeändert werden.

Aus mir nicht ganz einleuchtenden Gründen muß eine derartige Startroutine bei Diskettensystemen auf der Systemdiskette 2 gespeichert werden. Möglicherweise packt man dort auch, wenn der Platz reicht, MACLIB drauf - vielleicht ist in Laufwerk B normalerweise eine andere Datendiskette -, muß dann nur beim Laden von MACLIB auf die richtige Laufwerksangabe achten.

Mit der Startroutine wird Framework dann so aufgerufen:

```
fw laderoutine.fw/x
```

Gegebenfalls ist dafür die Start-Batch-Datei oder AUTOEXEC.BAT abzuändern (siehe Kapitel 5).

36 Tastatur-Bezeichnungen

Beim Programmieren und beim Schreiben von Makros kommt man an lästiger Tipp-Arbeit nicht vorbei. FRED-Funktionen, die man häufiger einsetzt, lassen sich ja als Makros definieren. Genauso die Tastenbezeichnungen - aber die allein sprengen eine Makro-Bibliothek schon.

Hinweis: Das Makro `[ALT]-T` befindet sich in der Makro-Bibliothek `MAKRO` auf der Beispieldiskette.

Das folgende Makro `[ALT]-T` nimmt diese Arbeit ab. Es fordert den Benutzer auf, eine Taste zu drücken; deren Bezeichnung, einschließlich der geschweiften Klammern, wird an der Cursorposition eingefügt.

Die korrekten Tastaturbezeichnungen zu ermitteln, ist nicht weiter schwierig - dafür gibt es eine FRED-Funktion. Das Problem liegt darin, Framework zu überzeugen, daß es hier einen Text vor sich hat, der zu schreiben und nicht mit einem Tastaturbefehl zu verwechseln ist.

Weil dieses Problem schon ein anderer gelöst hat, ist dieses Makro teilweise schlicht geklaut - die gleiche Schwierigkeit war im `MACLIB`-Makro `[ALT][F5]` zu bewältigen, deshalb ist dieser Transformationsteil daraus übernommen. Daran sieht man auch, wie man bestehende Programme für eigene auswerten kann.

```
@local(i, ch, length, tempstr, tempstr2),
@eraseprompt,
@prompt("Drücken Sie eine Taste!",20),
@nextkey,
@eraseprompt,
tempstr:=(@keyname(@key)),
length := @len(tempstr),
i := 1,
@while(i <= length,
  ch := @mid(tempstr,i,1),
  @if (ch = "{",
    tempstr2 := tempstr2 & "{()",
    tempstr2 := tempstr2 & ch
  ),
  i := i + 1
),
@performkeys(tempstr2)
```

37 Klangspiele

Mit Framework kann man auch komponieren; mühsam zwar, aber es geht: mit der FRED-Funktion `@beep`. Sie verlangt zwei Parameter: zunächst die Frequenz des Tones, dann dessen Dauer in Sekunden.

Hinweis: Das Makro `ALT`-R befindet sich in der Makro-Bibliothek MAKRO auf der Beispieldiskette.

Das Makro `ALT`-R spielt eine Tonleiter:

```
@beep(261.63,50), ;c
@beep(293.66,50), ;d
@beep(329.63,50), ;e
@beep(349.23,50), ;f
@beep(392.00,50), ;g
@beep(440.00,50), ;a
@beep(493.88,50), ;h
@beep(523.25,70), ;c
@beep(493.88,50),
@beep(440.00,40),
@beep(392.00,30),
@beep(349.23,20),
@beep(329.63,15),
@beep(293.66,10),
@beep(261.63,10)
```

Wer mit seinen Framework-Eigenkompositionen in die Musikgeschichte eingehen will, muß zumindest viel tippen. Dafür kann er auch Töne hervorzaubern, die jenseits der Hörschwelle liegen: Eine Frequenz von 16000 Hz hört allenfalls noch ein Hund.

Ganz unten, zwischen 0 und 37, ist ebenfalls Stille - womit man auch zeitlich umgrenzte Pausen in ein Programm einbauen kann.

38 Das Demo-Makro

Das Demo-Makro `[ALT]-D` aus Kapitel 25 ist ein Beispiel dafür, wie man »Umschläge« für eigene Programme machen kann. Um die Schreibarbeit möglichst gering zu halten, wird auf die Framework-Wiederhol-Funktion zurückgegriffen: `{CTRL}` in Verbindung mit einer Zahl wiederholt das folgende Zeichen entsprechend oft.

Hinweis: Das Makro `[ALT]-D` befindet sich in der Makro-Bibliothek `MAKRO` auf der Beispieldiskette.

Die Wiederholungsrate kann auch mehrstellig sein, muß dann jedoch aufgeteilt werden:

```
{CTRL-6}{CTRL-9}
```

Das heißt in diesem Fall 69.

Das Zeichen kann dabei auch mit seinem dezimalen ASCII-Code angegeben werden, wodurch man auch Zugriff hat zu den Sonderzeichen, mit denen man malen kann.

```
@echo(#OFF),
@performkeys("{CTRL-N}f{F4}{CTRL-DNARROW}{RETURN}{DNLEVEL}"),
@echo(#ON),
@performkeys("{CTRL-6}{CTRL-9}{176}{RETURN}"
& "{CTRL-6}{CTRL-9}{178}{RETURN}"
& "                {201}{CTRL-2}{CTRL-6}
{205}{187}{RETURN}"),
@beep(523.25,10),
@beep(659.26,50),
@beep(587.33,10),
@beep(440.00,50),
@performkeys("                                {186}    Herzlich willkommen!
{186}{RETURN}"
& "                                {200}{CTRL-2}{CTRL-6}
{205}{188}{RETURN}"
& "{RETURN}"
& "{RETURN}"),
@beep(440.00,50),
@beep(587.33,10),
@beep(659.26,50),
@beep(523.25,50),
@performkeys("                Sie haben mit Junior Framework ein Makro
aufgerufen{RETURN}"
& "{RETURN}"
& "                {CTRL-5}{CTRL-1}{205}{RETURN}"
& "                                Bald sind Sie so fit,{RETURN}"
& "                daß Sie solche Makros auch selber basteln
können!{RETURN}"),
@performkeys("{RETURN}"
& "{CTRL-6}{CTRL-9}{176}{RETURN}"
```

```
& "{CTRL-6}{CTRL-9}{178}"),
@beep(40.00,50),
@beep(100.00,9),
@beep(200.00,8),
@beep(300.00,7),
@beep(400.00,6),
@beep(500.00,5),
@beep(600.00,4),
@beep(700.00,3),
@beep(1000.00,10),
@beep(900.00,9),
@beep(800.00,8),
@beep(700.00,7),
@beep(600.00,6),
@beep(500.00,5),
@beep(400.00,4),
@beep(300.00,3),
@beep(200.00,20),
@prompt("Mit einer beliebigen Taste beenden Sie die
Demonstration",10),
@nextkey,
@performkeys("{uplevel}{del}")
```

39 Sonderdienste

Wenn Sie den mysteriösen Punkt *Sonderdienste* im Menü *Laufwerk* aktivieren, wird ein wohltönendes Demo-Programm gestartet. Was es mit der ganzen Sache auf sich hat, ist jedoch, wenn ich's recht sehe, nirgendwo in den Handbüchern erläutert.

Sonderdienste ist freigehalten für eigene Anwendungen. Mit dieser Funktion können Sie jegliches Programm abrufen und starten - die Sekretärin Katharina Kröll beispielsweise kann ihr Brieffloskel-Makro hineinpacken, der Dichter Jonathan Kraft seine Einnahmen-Überschuß-Rechnung. Sie können auch ein Programm schreiben, das über ein Menü in viele andere Programme verzweigt.

Nur zwei Bedingungen müssen erfüllt sein:

- Das Programm muß UTIL heißen.
- UTIL muß sich auf der Arbeitsfläche befinden.

Es braucht nicht neu berechnet zu werden, auch wenn es sich um ein Makro handelt, das wird automatisch erledigt, sobald Sie das Programm aufrufen.

Wie man ein Programm zum *Sonderdienst* einsetzt, sei am Beispiel der Einnahmen-Überschuß-Rechnung Jonathan Krafts demonstriert:

1. Geben Sie der Datei VERLUST den Namen UTIL.
2. Tauschen Sie mit der *Ersetzen*-Funktion aus dem Menü *Suchen* im gesamten Frame »Verlust« gegen »Util« - sonst stimmen die Bezüge nicht mehr.

Tip: Achten Sie darauf, daß im Menü *Suchen* alle Optionen (Namen, Inhalt, Formeln, Großschreibung) aktiviert und alle Frames geöffnet sind.

3. Aktivieren Sie *Sonderdienste* im Menü *Laufwerk*.

Damit wird das Programm auch schon gestartet - es braucht kein Initialisierungs-Makro, keine Neuberechnung.

Der Menüpunkt muß übrigens nicht *Sonderdienste* heißen. In CONFIG.FW können Sie auch einen anderen Namen festlegen.

40 DOS-Zugriff

Eine der netten Eigenschaften von Framework ist die Tür zum Begriffssystem, mit der man auf die DOS-Ebene huscht und dort alle DOS-Befehle ausführen kann.

1. Erstellen Sie einen neuen Text-Frame.
2. Aktivieren Sie *DOS-Kommandos* im Menü *Laufwerk*.
3. Führen Sie die DOS-Kommandos aus.
4. Kehren Sie mit

`exit`

wieder zu Framework zurück.

Das läßt sich etwas einfacher haben, wenn Sie MACLIB aktiviert haben: Das Makro **[ALT] [F8]** erstellt den DOS-Zugriffs-Frame automatisch.

Wollen Sie auf externe DOS-Befehle zugreifen, müssen Sie bei Diskettensystemen die DOS-Diskette in eines der Laufwerke legen.

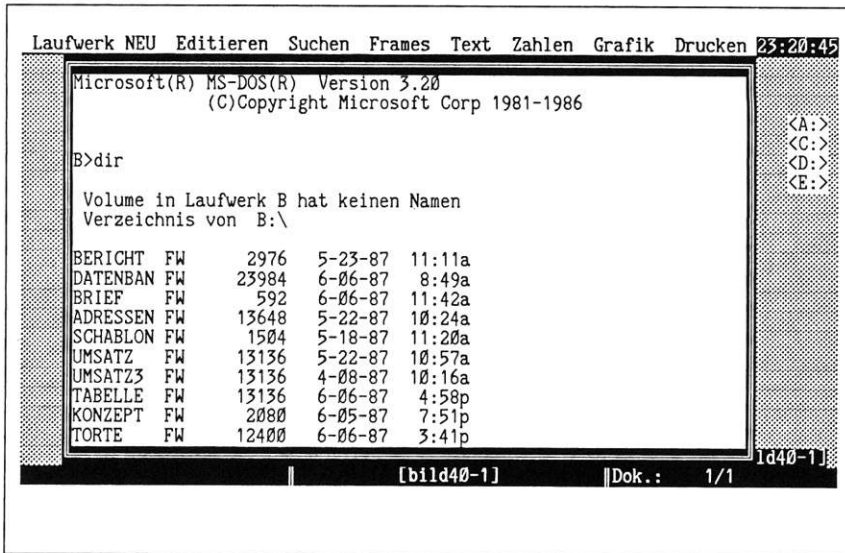


Bild 40.1: Der DOS-Zugriff

All Ihre Kommandos und die DOS-Reaktionen darauf werden, allerdings nur bei Disketten-Systemen, in dem Frame ausgegeben - den Sie wiederum, so Sie möchten, auf

Diskette speichern können. Manchmal möchte man. Zum Beispiel, um sich ein langes Inhaltsverzeichnis in Ruhe anschauen zu können. Oder um sich zu überlegen, welche Dateien man löschen kann. Machen Sie, zum Beispiel, aus dem DOS-Zugriffs-Frame eine Batch-Datei:

Zur Erinnerung: Eine Batch-Datei (zwingend vorgeschriebene Endung: BAT) enthält eine Folge von DOS-Befehlen, die vom Betriebssystem nacheinander abgearbeitet wird.

1. Geben Sie sich das Inhaltsverzeichnis aus, kehren Sie dann wieder zu Framework zurück.
2. Editieren Sie den DOS-Zugriffs-Frame: entfernen Sie alle Dateien, die Sie behalten wollen - übrig bleiben die Dateien, die gelöscht werden sollen.
3. Schreiben, ändern, kopieren Sie (alle Möglichkeiten von Framework stehen ja zur Verfügung!), bis Sie die entsprechen Kommandozeilen haben:

```
del datei1.txt
del datei2.doc
del datei3.fw
```

Tip: Lassen Sie sich das Inhaltsverzeichnis mit *dir/w* ausgeben, dann müssen Sie nicht Datum und Uhrzeit löschen.

4. Lassen Sie den DOS-Zugriffs-Frame in eine *Textdatei für DOS* umwandeln (Menü *Laufwerk*) - das ist dann ein reiner ASCII-Text. Eine normale Framework-Datei enthält Steuerzeichen, die das Betriebssystem nicht versteht.
5. Speichern Sie den Frame, zur Sicherheit unter einem anderen Namen.
6. Gehen Sie durch die Tür ins Betriebssystem.
7. Benennen Sie Ihren Batch-Frame um, weil er von Framework ja die Endung FW bekommen hat; etwa:

```
ren batch.fw batch.bat
```

Wichtig ist die Endung BAT, nicht der Dateiname.

8. Starten Sie die Batch-Datei

Das lohnt sich natürlich nur, wenn viele Dateien gelöscht werden sollen - etwa, wenn man mal wieder die überquellende Festplatte aufräumt.

Der DOS-Zugriffs-Frame hat nur eine begrenzte Speicherkapazität, deren Umfang in CONFIG.FW festgelegt ist. Wird die Grenze überschritten, löscht Framework von Anfang an »first in, first out« sagen die Angelsachsen dazu in einer griffigen Prägnanz, zu der unsere Sprache nicht fähig ist.

Mit dem DOS-Zugriff kann man auch das Primär-Laufwerk wechseln, ohne deswegen CONFIG.FW ändern zu müssen, was ja immer mit einigem Umstand verbunden ist, weil die Änderung erst bei einem Neustart wirksam wird.

1. Gehen Sie mit dem DOS-Zugriff ins Betriebssystem.
2. Wechseln Sie das aktuelle Laufwerk: A statt B oder C.
3. Kehren Sie mit
`exit`
wieder zu Framework zurück.

Die Dateien werden nunmehr auf Laufwerk A gespeichert. Diese Einstellung bleibt solange gültig, bis Sie erneut wechseln oder Framework neu starten.

41 DISKUTIL

DISKUTIL auf der Framework-Dienstprogramm-Diskette ist ein menügesteuertes Hilfsprogramm, um

- Disketten zu testen
- Disketten zu formatieren
- Disketten zu kopieren
- Verzeichnisse anzulegen.

DISKUTIL macht nichts anderes, als die entsprechenden DOS-Befehle auch. Insofern können Sie das alles ebensogut über den DOS-Zugriff erledigen, DISKUTIL ist kein bißchen bequemer zu handhaben.

1. Laden Sie DISKUTIL.
2. Drücken Sie **[F5]** (*Neuberechnung*). DISKUTIL wird daraufhin versteckt.

Merke: Ein Frame wird mit *@hide* versteckt. Er ist dann noch im Arbeitsspeicher, auf ihn kann jederzeit Bezug genommen werden.

3. Aktivieren Sie DISKUTIL mit **[ALT]-D**.

Es ist ratsam, zunächst die *Vorbelegung* anzuschauen und nach Bedarf zu ändern.

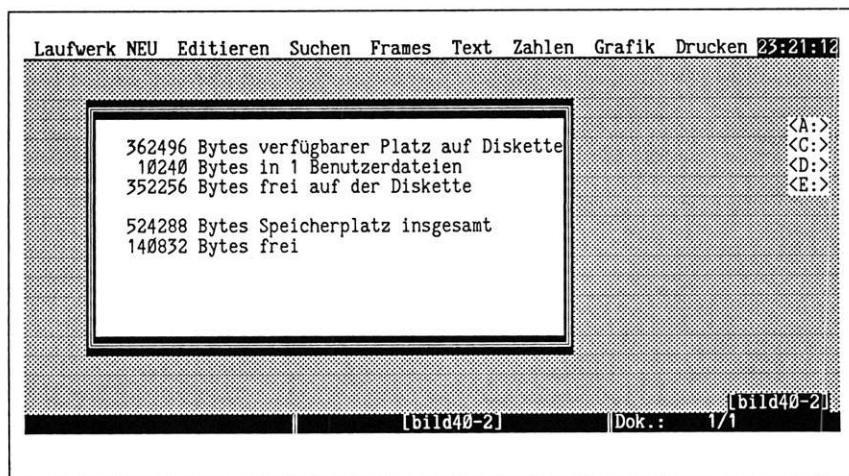


Bild 41.1: Der Test-Dialog von DISKUTIL

DISKUTIL fragt als erstes nach dem Pfad zu den externen DOS-Befehlen. Bei einer Festplatte ist hier das Verzeichnis gefragt, in welchem die DOS-Befehle gespeichert sind; bei einem Diskettensystem wird man praktischerweise Laufwerk A angeben.

Dann will DISKUTIL das zu testende Laufwerk wissen, als nächstes das Laufwerk, auf dem die Disketten formatiert werden sollen. Beides können Sie indes während des Einsatzes von DISKUTIL auch anders bestimmen.

Die Vorbelegungen werden abgespeichert, Sie können das Programm starten. Müssen zuvor aber beim Diskettensystem die DOS-Diskette in das als Pfad definierte Laufwerk legen (wenn nicht, fordert DISKUTIL dazu auf). Oder Sie kopieren sich die entsprechenden DOS-Dateien

- FORMAT
- CHKDSK
- DISKCOPY

auf die Systemdiskette 2, der Platz dafür reicht normalerweise.

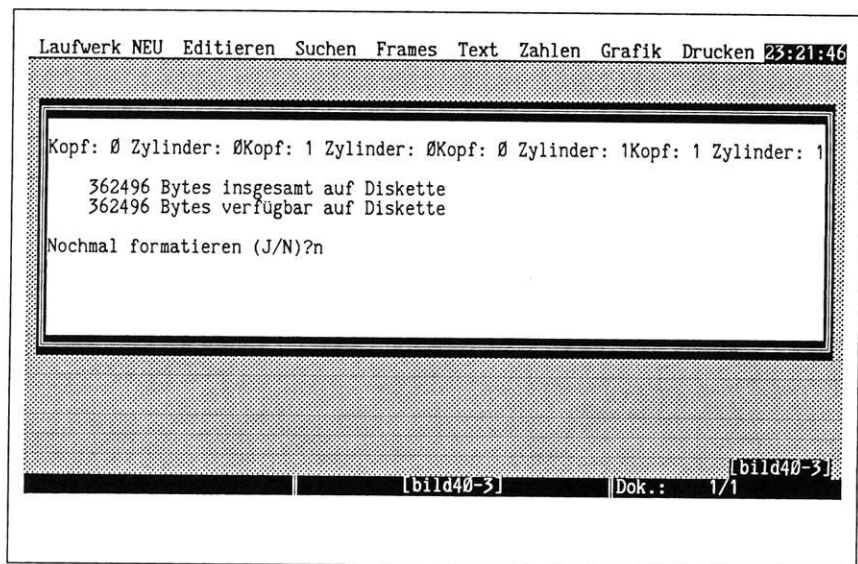


Bild 41.2: Das Formatieren einer Diskette mit DISKUTIL

Trotzdem kann es vorkommen, daß DISKUTIL stur nach der DOS-Diskette verlangt (heißt: nach den DOS-Dateien), sie kriegt und partout nicht funktioniert. Was ist kaputt?

Nichts Irreparables. Vermutlich arbeiten Sie mit einer Betriebssystem-Version, die andere Dateibezeichnungen verwendet, als DISKUTIL es erwartet.

DISKUTIL ist programmiert auf

- FORMAT.COM
- CHKDSK.COM
- DISKCOPY.COM

Manche DOS-Versionen jedoch verwenden statt COM die Endung EXE - das kann DISKUTIL natürlich nicht finden. Sie müssen das Programm ein wenig umschreiben:

1. Öffnen Sie den Formelspeicher der Sub-Frames
 - Format
 - Chkdsk
 - Diskcopy
2. Suchen Sie nach den DOS-Befehlen (die Zeilen beginnen mit **@run**, der FRED-Funktion für die Ausführung von Programmen auf Betriebssystemebene).
3. Ändern Sie die Endungen.

Jetzt läuft's. Beim Formatieren wird vorsichtshalber nochmals nach dem Laufwerk gefragt.

DISKUTIL erstellt jeweils ein Dialog-Fenster und gibt dort die DOS-Meldungen aus. Mit Ausnahme des CHKDSK-Frames werden sie anschließend automatisch gelöscht.

Tip: Möchten Sie die Dialog-Frames erhalten, müssen Sie in den entsprechenden DISKUTIL-Sub-Frames die letzte Zeile `@performkeys("{del}")` löschen.

42 Übernahme aus Fremdprogrammen

Die meisten Programme speichern in ihren Dateien interne Steuerzeichen, die von anderen Programmen nicht zu entziffern sind - das macht den Datenaustausch etwas schwierig.

Bei Textprogrammen ist es noch am problemlosesten. Deren Dateien bestehen - oft, nicht immer - zum überwiegenden Teil aus Text im ASCII-Code, dem Computer-Esperanto, scheren nur - meist, nicht immer - bei den Umlauten aus.

Solche Texte kann man auch in Framework laden und entsprechend bearbeiten. Das ist zwar unter Umständen mühsam, aber es geht. Am einfachsten natürlich, wenn das Textprogramm seine Dateien ins ASCII-Format konvertieren kann. ASCII-Texte kann Framework ohne Probleme lesen.

Bei anderen Anwenderprogrammen ist die Übernahme etwas schwieriger. Framework hat deshalb eine FRED-Funktion und verschiedene Hilfsprogramme integriert, die bestimmte Dateitypen konvertieren. Nämlich:

- dBase-Dateien
- WordStar-Dateien
- Tabellen von Lotus 1-2-3 (und alle im WKS-Format)
- VisiCalc-Dateien (und alle im DIF-Format)

dBase-Dateien

Framework kann direkt Daten der verschiedenen dBase-Versionen übernehmen, ohne daß dazu Konvertierungen erforderlich sind - wie man's eigentlich erwarten sollte bei Produkten aus dem gleichen Hause.

1. Laden Sie die dBase-Datei in Framework. Automatisch wird ein Datenbank-Frame mit den entsprechenden Feldern angelegt, Framework trägt eine Formel ein.
2. Drücken Sie **[F5]** (*Neuberechnung*). Daraufhin werden die Datensätze übernommen.

Das gleiche geht auch manuell mit einer Filterformel, womit die dBase-Datei auch selektiert werden kann:

1. Erstellen Sie einen Datenbank-Frame der erforderlichen Größe.
2. Geben Sie eine Formel ein nach folgendem Schema:

@dbasefilter(Name der dBase-Datei, Selektionskriterium, Erster Datensatz, Letzter Datensatz

3. Drücken Sie **[F5]** (*Neuberechnung*).

Die Datensatznummern können wahlweise angegeben werden und sind eine Möglichkeit, umfangreiche dBase-Dateien so aufzuteilen, daß Framework sie auch bearbeiten kann.

Die drei Hilfsprogramme sind im Frame -READ_ME.FW erläutert, den Sie sich anschauen und ausdrucken können. Wenn Sie nicht möchten: Hier folgen die entsprechenden Ausschnitte daraus.

WordStar-Dateien

Die WS2FW-Programme von der Dienst-Programm-Diskette konvertieren die WordStar-"Dokumenten"-Dateien in Framework-Textdateien. Dieser Prozeß entfernt das achte Bit und setzt die Framework-Attribute für Fettdruck und Unterstreichung. Text, der in WordStar unterlegt dargestellt wird (zur Kennzeichnung von Doppeldruck) erscheint in Framework fettgedruckt. Andere, in WordStar-Texte eingebundene Ctrl-Zeichen, werden automatisch unterdrückt.

Zur Vorbereitung der WS2FW-Hilfsprogramme:

Falls Sie ein doppeltes Diskettenlaufwerk besitzen, kopieren Sie bitte zunächst die Dateien WS2FW.EXE und WS2FW.FW im DOS-Betriebssystem auf die System-Diskette Nr. 2 oder auf die Diskette, auf der sich Ihre WordStar-Dateien befinden.

Ist Ihr Computer mit einem Festplattenlaufwerk ausgerüstet, kopieren Sie bitte die angegebenen Dateien in das Directory, in dem Ihre Framework-System-Dateien stehen.

Zum Start des WS2FW-Hilfsprogramms:

1. Starten Sie Framework und laden Sie dann WS2FW.FW auf die Arbeitsfläche.
2. Drücken Sie die F5-Taste, wenn der Rand des Frames WS2FW.FW hell unterlegt erscheint, um das Programm zu starten.
3. Sie werden vom Programm gebeten, den Namen des Laufwerks bzw. des Verzeichnisses anzugeben, wo sich die Datei WS2FW.EXE befindet. Framework nimmt zunächst an, daß sich die Datei im augenblicklichen Verzeichnis des Boot-Laufwerks befindet. Nach eventueller Korrektur der Laufwerkskennung bzw. des Inhaltsverzeichnisses drücken Sie bitte RETURN.
4. Jetzt bewegen Sie den Cursor bitte auf die Laufwerkssymbole (auf der Arbeitsfläche), danach in das Directory und zuletzt auf den Namen der WordStar-Datei, die Sie konvertieren möchten. Derselbe Name muß dann in der Mitte der Statuszeile erscheinen.
5. Dann drücken Sie {Alt-W} (die Alt-Taste und W gleichzeitig betätigen). Das Konvertierungsprogramm läuft ab und liefert einen Textframe, der den Text der WordStar-Datei enthält. Der konvertierte Text im Frame enthält »feste« Zeilenschaltungen (RETURNS) am Ende jeder Zeile. Falls Sie planen, den Text neu zu editieren, können Sie diese löschen.

Bitte vergessen Sie nicht, den neuen Frame mit konvertiertem und neu editiertem Text auf Diskette zu speichern.

WKS-Tabellen (Lotus 1-2-3)

Die Datei trans123.exe auf der Dienst-Programm-Diskette übersetzt Lotus-1-2-3-".wks"-Dateien in funktionierende Framework-Tabellen. Alle Zahlen, Namen, Formeln und Formate werden in Framework-equivalente Form konvertiert. Es können weder Lotus-1-2-3-, noch Lotus-Symphony-Makros konvertiert werden. Die Konvertierung der Dateien erfolgt auf der DOS-Ebene. Demnach sollten Sie Ihren Computer eingeschaltet und das DOS-Betriebssystem geladen haben.

Merke: Das WKS-Konvertierprogramm funktioniert nur auf DOS-Ebene, nicht in Framework.

Um Lotus-1-2-3-Dateien zu konvertieren:

1. Wenn Ihr Computersystem mit einem Festplattenlaufwerk ausgerüstet ist, legen Sie bitte die Dienst-Programm-Diskette in das A-Laufwerk und vergewissern Sie sich, ob das Symbol C> auf dem Bildschirm erscheint. Ihre Lotus-Dateien sollten sich auf der Festplatte befinden.
Falls Sie über ein doppeltes Diskettenlaufwerk verfügen, legen Sie die Dienst-Programm-Diskette in das A-Laufwerk und Ihre Diskette mit .wks-Dateien in das B-Laufwerk. Vergewissern Sie sich bitte, ob auf der zweiten Diskette genügend Platz für die neuen, zu erstellenden Framework-Dateien vorhanden ist und ob das Symbol B> auf dem Bildschirm erscheint. (Tippen Sie B: und RETURN, falls notwendig.)
2. Geben Sie A:trans123 ein und drücken Sie dann RETURN.
3. Folgen Sie den Anweisungen des Programms. Es wird Sie zuerst nach dem Namen Ihrer 1-2-3-.wks-Datei fragen und dann nach dem neuen Namen, den Sie der .fw-Datei geben möchten. Geben Sie den gesamten Pfadnamen für die Datei an, etwa "B:\budget", und drücken Sie RETURN.
4. Das Programm fragt, wieviel Zeilen und Reihen Sie konvertieren möchten. Wenn Sie dann Ihre kompletten Arbeitsunterlagen konvertieren möchten, drücken Sie RETURN als Antwort auf all diese Fragen.
5. Jetzt wird die .wks-Datei konvertiert und eine neue .fw-Datei mit dem neuen Namen erzeugt.

Anmerkungen:

Wenn kein neuer Name angegeben ist, wird eine .fw-Datei mit dem alten Dateinamen und dem Postfix .fw erzeugt.

Wenn Sie die Anzahl der Reihen und Spalten, die konvertiert werden sollen, angeben, werden die Reihen vom Anfang der Tabelle und die Spalten vom linken Rand der Tabelle aus konvertiert. Um nur einen Teil einer Tabelle zu konvertieren, der nicht in

der ersten Reihe oder mit der ersten Spalte beginnt, sondern mitten im 1-2-3-Programm, sollten Sie diesen Teil zuerst in eine andere Tabelle kopieren und diese dann konvertieren.

DIF-Dateien (VisiCalc)

Das Programm `transdif.fw` (Translate DIF) konvertiert Standard-"DIF"-Dateien in Tabellenkalkulationsframes. DIF ist eine Abkürzung für "Data Interchange Format", welches ein Standard-Datenformat bei vielen Tabellenkalkulationsprogrammen ist. Das FRED-Programm `transdif.fw` wird geladen und arbeitet eine angegebene DIF-Datei ab, erstellt gleichzeitig Framework-Tabellen, welche alle Zahlen, Bezeichnungen und Formate enthalten.

Zur Benutzung des `transdif`-Programms:

1. Starten Sie Framework.
2. Laden Sie die `transdif.fw`-Datei von der Dienst-Programm-Diskette auf Ihre Arbeitsfläche.
3. Laden Sie die DIF-Dateien, die Sie übertragen möchten, auf Ihre Arbeitsfläche (`transdif` erwartet Eingaben aus einem Text-Frame auf der Arbeitsfläche. DIF-Dateien werden automatisch als Text-Frames geladen).
4. Drücken Sie die F5-Taste (NEUBERECHNEN), wenn der Rand des Frames `transdif` hell unterlegt erscheint. Sie erhalten die Anweisung, weitere Tasten zu betätigen und werden nach der Bezeichnung der DIF-Datei gefragt. Nach dem Abschließen der Eingabe mit RETURN erstellt `transdif` automatisch einen Tabellenkalkulationsframe aus den Daten der DIF-Datei.

Vergessen Sie bitte nicht, Ihren auf diese Art und Weise erstellten, neuen Tabellenkalkulationsframe nach der Konvertierung und nach eventuellem Editieren auf Diskette zu sichern.

43 Zeichnen mit FRED

Die Funktion **@draw** erlaubt es, mit Framework zu zeichnen. **@draw** hat entsetzlich viele Parameter, und man muß entsetzlich viel tippen, wenn man zeichnen will, und entsetzlich viel ausprobieren und korrigieren und überlegen ...

Eine Sache ausgesprochen für Tüftler. Hier nur eine kleine und, wie ich zugeben muß, recht läppische Demonstration - als BILD.FW auf der Beispieldiskette.

■ **Hinweis:** Im Referenz-Handbuch, S. 11-40f., sind alle **@draw**-Parameter aufgelistet.

Aus dem Programm mögen Sie einige Prinzipien erschen:

```
@draw(
4,10000,1000,
14,20000,10000,
4,10000,1000,
9,20000,10000,
4,20000,10000,
12,3,
14,30000,20000,
4,20000,20000,
12,7,
14,15000,25000,
12,0,
13,15000,25000,10000,0,360
)
```

Die erste Kennziffer jeder Zeile ist von FRED festgelegt. Zum Beispiel bedeuten:

- 4 Bewegung des abgehobenen Zeichenstiftes
- 14 Zeichnen eines ausgefüllten Rechteckes
- 9 Linie ziehen
- 12 Füllraste
- 13 Kreis

Den Kennziffern folgen dann jeweils die Koordinatenangaben, erst X, dann Y.

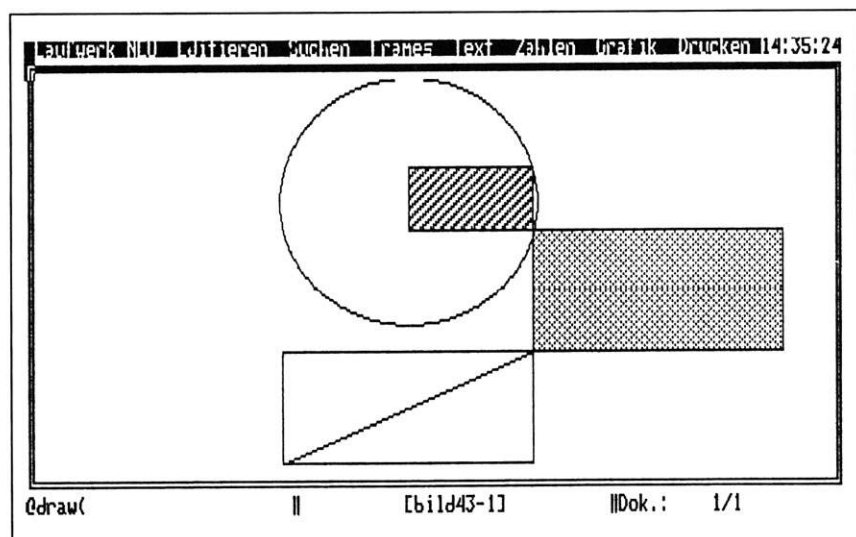


Bild 43.1: Zeichnen mit *FRED*

Framework - ein Überblick

Absatz-Einzug

1. Bringen Sie den Cursor irgendwo in den Absatz.
2. Aktivieren Sie *Einzug Absatz* im Menü *Text*.
3. Geben Sie die Anzahl der Zeichen an, um welche die erste Zeile des Absatzes eingerückt werden soll.

(Kapitel 21)

Adressierung

Framework unterscheidet in Tabellen-Formeln zwischen relativer und absoluter Adressierung. Relative Adressierungen nehmen, wenn sie in eine andere Zelle kopiert werden, deren Koordinaten an. Absolute Adressierungen sind an die angegebenen Zellen gebunden, entnehmen ihre Werte also daraus und nirgendwoher sonst.

Relative Adressierungen sind normale Koordinaten-Angaben, z.B. B5:F5. Absolute Adressierungen werden mit dem Zeichen \$ vor der Spalte und/oder Zeile angegeben. Relative und absolute Adressierungen können untereinander gemischt werden:

- \$F5 absoluter Bezug auf die Spalte F, relativer Bezug auf die Zeile 5
- F\$5 relativer Bezug auf die Spalte F, absoluter Bezug auf die Zeile 5
- \$F\$5 absoluter Bezug auf die Spalte F, absoluter Bezug auf die Zeile 5

(Kapitel 22)

Cursormethode

Statt Koordinatenbezüge in Tabellen- oder Datenbank-Formeln direkt einzugeben, werden sie bei der Cursormethode von Framework übernommen.

1. Öffnen Sie den Formelspeicher, geben Sie, wenn erforderlich, die Rechenfunktion samt der ersten Klammer ein, z.B.

@sum(

2. Drücken Sie **↑** - das aktiviert die Cursormethode
3. Bringen Sie den Cursor (mit den Cursortasten) in die erste Zelle des Bezugs.
4. Drücken Sie **F6** (*Auswahl*).
5. Bringen Sie den Cursor in die letzte Zelle des Bezugs.
6. Drücken Sie **RETURN**.
7. Geben Sie, wenn Sie eine Rechenfunktion benutzen, die schließende Klammer ein.
8. Beenden Sie die Formeleingabe mit **RETURN**.

Die Cursormethode ist immer dann zu empfehlen, wenn große Bereiche erfaßt oder wenn Tabellen miteinander verbunden werden sollen. Grundsätzlich hilft die Cursormethode, Fehler in den Koordinatenangaben zu vermeiden.

(Kapitel 22, 23 und 24)

Dateien laden

1. Wechseln Sie, falls nötig, mit `SCROLL LOCK` zum Dateiverzeichnis, öffnen Sie es, falls nötig, mit `RETURN`.
2. Gehen Sie mit `NUM+` in den Dateiverzeichnis-Frame.
3. Bringen Sie den Cursor auf die zu ladende Datei bzw. markieren Sie mit `F6` (*Auswahl*) mehrere Dateien.
4. Drücken Sie `RETURN`.

Die Dateien werden in dem Zustand geladen, in dem sie abgespeichert worden sind - Größe, Position usw.

Alternativ dazu kann der Name der Datei auch direkt angegeben werden:

1. Öffnen Sie das Menü *Laufwerk*.
2. Aktivieren Sie die Funktion *Hole Dokument mit Namen*. Daraufhin wird die Editierzeile geöffnet.
3. Geben Sie den Namen ein, und zwar mit der Datei-Extension (andernfalls meldet Framework, die Datei sei nicht vorhanden) und mit dem Laufwerks-Buchstaben, wenn ein anderes als das Primär-Laufwerk angesprochen werden soll.

(Kapitel 13)

Dateien löschen

1. Bringen Sie den Cursor auf die gewünschte Datei oder markieren Sie mehrere Dateien mit `F6` (*Auswahl*).
2. Drücken Sie `DEL`.

War die Datei auf der Arbeitsfläche oder im Dokumentenstapel (gleichgültig, ob geöffnet oder geschlossen), wird sie lediglich im Arbeitsspeicher des Computers gelöscht, ist aber auf der Diskette oder Platte noch vorhanden; verloren sind aber sämtliche Änderungen seit dem letzten Sichern. Mit *Rücknahme* (siehe dort) ist die letzte Löschung wieder rückgängig zu machen, vorausgesetzt, zwischenzeitlich ist nichts anderes gelöscht worden.

Auch auf der Diskette oder Platte können Dateien auf die gleiche Weise gelöscht werden. Man muß dazu nur in das Dateiverzeichnis gehen. Eine solche Löschung ist aber endgültig.

(Kapitel 13)

Dateiverzeichnis

Framework-Bezeichnung gleichermaßen für den Stapel der Laufwerke rechts oben auf der Arbeitsfläche wie für den Frame, der das Inhaltsverzeichnis des Laufwerkes bzw. des Verzeichnisses ausgibt.

Ein Dateiverzeichnis-Frame ist ein normaler Frame, das heißt, er kann vergrößert, verschoben oder ausgedruckt werden, nur nicht kopiert oder verlagert.

(Kapitel 13)

Daten eingeben Dateneingabe

Framework unterscheidet bei Tabellen und Datenbanken zwischen Text (Fachbezeichnung: String) und numerischen Werten, wobei auch Zahlen Text sein können.

Grundsätzlich gilt für die Eingabe von Text:

- Besteht der Text nur aus Buchstaben, schreiben Sie einfach drauf los. Framework schaltet automatisch um in die Editierzeile.
- Enthält der »Text« auch Zahlen oder eines dieser Zeichen:

@ + - (# \$! "

muß vor der Eingabe die Leertaste gedrückt werden, worauf Framework wiederum die Editierzeile öffnet; andernfalls versteht Framework die Zeichen als Teil einer Formel.

Für numerische Werte, gemeinhin als Zahlen bekannt, mit denen man rechnet, sieht das so aus:

- Besteht der Text nur aus Zahlen, schreiben Sie einfach drauf los. Framework schaltet automatisch um in die Editierzeile (»implizite Umschaltung«).
- Beginnt die »Zahl« mit einem dieser Zeichen:

@ + - (# \$! "

kann gleichfalls sofort geschrieben werden.

- Alternativ dazu die »explizite Umschaltung«: vor der Dateneingabe F2 (Formel editieren) drücken.

(Kapitel 17, 21)

Datenbank vergrößern

Eine Datenbank wird mit den - in CONFIG.FW definierten - Werten eingerichtet, die im Menü *NEU* angegeben sind, hier aber auch abgeändert werden können. Eine bestehende Datenbank kann nachträglich erweitert werden:

1. Öffnen Sie das Menü *NEU*.
2. Aktivieren Sie *Spalten/Felder (# Einfügen)* bzw. *Zeilen/Sätze (# Einfügen)*. Darauf wird die Editierzeile geöffnet.
3. Geben Sie die Zahl der Felder bzw. Sätze an.

Die Felder/Sätze werden in der Datenbank unmittelbar nach dem Cursor eingefügt.

(Kapitel 21)

Datensätze entfernen

1. Markieren Sie die Datensätze.
2. Aktivieren Sie *Zeilen/Sätze entfernen* im Menü *Editieren*.

Framework warnt, daß die Operation nicht rückgängig zu machen sei - also besser zweimal hinschauen, ob auch die richtigen Datensätze markiert sind.

dBase-Filter

Framework kann direkt Daten der verschiedenen dBase-Versionen übernehmen, ohne daß dazu Konvertierungen erforderlich sind:

1. Laden Sie die dBase-Datei in Framework. Automatisch wird ein Datenbank-Frame mit den entsprechenden Feldern angelegt.
2. Drücken Sie **F5** (*Neuberechnung*). Daraufhin werden die Datensätze übernommen.

Das gleiche geht auch manuell mit einer Filterformel, womit die dBase-Datei auch selektiert werden kann:

1. Erstellen Sie einen Datenbank-Frame der erforderlichen Größe.
2. Geben Sie eine Formel ein nach folgendem Schema:
`@dbasefilter(Name der dBase-Datei, Selektionskriterium, Erster Datensatz, Letzter Datensatz`
3. Drücken Sie **F5** (*Neuberechnung*).

Die Datensatznummern können wahlweise angegeben werden.

(Kapitel 42)

Disketten formatieren, kopieren, testen

Das geht, wenn man nicht den DOS-Zugriff wählt (siehe dort), von Framework aus mit dem Hilfsprogramm DISKUTIL, allerdings auch nicht bequemer:

1. Laden Sie DISKUTIL.
2. Drücken Sie **F5** (*Neuberechnung*).
3. Drücken Sie **ALT-D**.

Daraufhin erscheint in der Editierzeile ein Menü. Achten Sie darauf, daß im angegebenen Laufwerk auch die richtige Diskette ist.

Wenn DISKUTIL nicht funktioniert, müssen Sie nachschauen, ob Ihre DOS-Befehle

- FORMAT
- DISKCOPY
- CHKDSK

die gleichen Datei-Endungen haben, wie sie im DISKUTIL-Programm angegeben sind.

Diese Funktionen sind externe Befehle, DISKUTIL braucht also die DOS-Diskette. Oder man kopiert die DOS-Befehle auf die Systemdiskette 2.

(Kapitel 41)

DOS-Zugriff

Während der Arbeit mit Framework kann auf sämtliche DOS-Befehle zugegriffen werden, ohne Framework deswegen verlassen zu müssen:

1. Erstellen Sie einen neuen Text-Frame.
2. Aktivieren Sie die Funktion *DOS-Kommandos* im Menü *Laufwerk*.
3. Geben Sie die DOS-Kommandos ein. Bei externen Befehlen müssen Sie zuvor die Diskette mit dem Betriebssystem in das Primär-Laufwerk legen.
4. Kehren Sie danach mit

`exit`

wieder zu Framework zurück

5. Löschen Sie, wenn Sie wollen, den Frame *DOS-Zugriff*.

Wieviel Zeichen der Frame maximal aufnehmen darf, kann in CONFIG.FW definiert werden.

In der Makro-Bibliothek MACLIB befindet sich das Makro `[ALT][F8]`, das den DOS-Zugriffs-Frame automatisch einrichtet und die Funktion aktiviert.

Das Framework-Hilfsprogramm DISKUTIL erlaubt menügesteuerten Zugriff auf die DOS-Kommandos zur Diskettenverwaltung (siehe dort).

(Kapitel 40)

Drucken

1. Markieren Sie den Frame, der gedruckt werden soll; es können auch mehrere Frames markiert werden. Soll in einem Konzept-Frame auch der Inhalt der Sub-Frames ausgedruckt werden, müssen diese mit `[F10]` (*Zeigen*) geöffnet werden.
2. Bestimmen Sie im *Drucken*-Sub-Menü das *Ziel für Ausgabe*.

3. Definieren Sie, wie gedruckt werden soll, im *Drucken*-Sub-Menü *Optionen für Ausdruck*: Anzahl der Seiten und Kopien, mit Namen der Frames, nur Formeln usw..
4. Beginnen Sie den Druck mit *Starten* im Menü *Drucken*.
5. Sie können den Druck mit den entsprechenden Funktionen unterbrechen (*Pause*) oder abbrechen (*Beenden*).

Statt eines Ausdrucks auf Papier kann auch eine Druckdatei auf Diskette geschrieben werden (*Textdatei für DOS* im Sub-Menü *Ziel für Ausgabe* bzw. *Textdatei für DOS erstellen* im Menü *Laufwerk*).

(Kapitel 18)

Druckformeln

Allgemeine Bezeichnung für Formeln, welche die Gestaltung eines Dokumentes beeinflussen und erst während des Druckes wirksam werden.

Definiert werden damit Seitenformat, Seitennummern, Kopf- und Fußzeilen (siehe jeweils dort), aber auch

@np	neue Seite; mit einem Wert wird zugleich der Beginn der Seitennummerierung festgelegt
@kp	Frame zusammenhalten: der Frame wird, wenn möglich, zusammenhängend auf einer Seite gedruckt
@pr	erlaubt die Zusammenfassung mehrerer Druckformeln, die damit auch in @if-Bedingungen eingebunden werden können
@print	druckt den in Klammern angegebenen Frame
@st	Drucker-Steuerzeichen: schickt eine Zeichenfolge zum Drucker; damit lassen sich Drucker-Funktionen wie Breitschrift, andere Schriftgröße usw. ansprechen

(Kapitel 19,21)

Ersetzen

Beim Ersetzen wird ein vorgegebener Text gesucht und dann gegen einen anderen ausgetauscht. Das funktioniert in allen Frame-Typen. Bei Datenbank und Tabelle wird jedoch nur spaltenweise gesucht und ersetzt; der Cursor muß zuvor in die betreffende Spalte gebracht werden.

1. Öffnen Sie das Menü *Suchen*.
2. Bestimmen Sie gegebenenfalls die Optionen für die Berücksichtigung von Namen, Inhalt, Formeln bzw. Großschreibung.
3. Aktivieren Sie die Funktion *Ersetzen durch*. Daraufhin öffnet sich der Formelspeicher.
4. Geben Sie den Such-Text ein.

5. Drücken Sie **RETURN**.
6. Geben Sie den Ersatz-Text ein.
7. Drücken Sie **RETURN**.

Ist der Such-Text gefunden worden, haben Sie die Wahl:

- RETURN** der Text wird ersetzt, Framework sucht nach dem nächsten Fundort und wartet dort
- ↓** der Text wird nicht ersetzt, Framework sucht nach dem nächsten Fundort und wartet dort
- HOME** Framework ersetzt den Text und wartet
- END** Framework durchsucht den gesamten Text und ersetzt automatisch überall
- ESC** Abbruch

Framework ersetzt von der Cursorposition an bis zum Dateiende. Gefunden werden auch einzelne Zeichen in Wörtern, zudem kann mit Ersatzzeichen gearbeitet werden:

- ? steht für ein beliebiges Zeichen (M??er z.B. findet Meier, Meyer, Maier, Mayer, Mauer, Maler usw.)
- * steht für mehrere beliebige Zeichen (M*er z.B. findet auch Meer, Meyerbeer, Müller, Malermeister, Mutter usw.)

Mit den Optionen kann definiert werden, ob

- die Namen von Frames
- der Inhalt von Frames
- die Formeln »hinter« Frames oder Zellen
- die Großschreibung

berücksichtigt werden soll. Die Optionen können beliebig kombiniert werden.

Auch die unsichtbaren Zeichen für **TAB** und Leertaste können gesucht und ersetzt werden, sofern sie sichtbar gemacht sind; **RETURN** allerdings nicht.

(Kapitel 20, 28)

Etiketten

Das Framework-Dienstprogramm MAILUTIL vereinfacht den Druck von (Adreß-) Etiketten. Benötigt werden

- eine Datenbank, die DATENBANK heißen und ein Feld haben muß, das den Text für die Etiketten aufnimmt
- eine Schablone: ein umfassender Frame, dessen Sub-Frames gemäß dem Etiketten-Formular angeordnet sind.

1. Laden Sie MAILUTIL, die Datenbank und den Schablonen-Frame.
2. Selektieren Sie, falls nötig, die Datenbank.

3. Bringen Sie den Cursor auf MAILUTIL und drücken Sie **F5** (*Neuberechnung*).
4. Bringen Sie den Cursor auf den Schablonen-Frame.
5. Drücken Sie **ALT**-E.

(Kapitel 21)

Feldbreite

Die Größe eines Feldes in einer Datenbank kann beliebig bestimmt werden; Änderungen eines Feldes in einer Spalte wirken sich auf die gesamte Spalte aus:

1. Markieren Sie ein Feld.
2. Drücken Sie **F4** (*Größe*).
3. Bestimmen Sie mit den Cursortasten **→** und **←** die gewünschte Größe. Im rechten Teil der Statuszeile ist die Größe in Zeichen abzulesen.
4. Beenden Sie mit **RETURN**.

Die Feldbreite hat nichts damit zu tun, wieviel Zeichen ein Feld aufnehmen kann, sondern bestimmt lediglich die Darstellung auf dem Bildschirm. Demzufolge kann man Felder auch verstecken, indem man eine Größe von 0 Zeichen definiert.

Die Feldbreite eines neuen Datenbank-Frames wird in CONFIG.FW festgelegt.

(Kapitel 20)

Felder entfernen

1. Markieren Sie die Felder.
2. Aktivieren Sie *Spalten/Felder entfernen* im Menü *Editieren*.

Framework warnt, daß die Operation nicht rückgängig zu machen sei - also besser zweimal hinschauen, ob auch die richtigen Felder markiert sind.

(Kapitel 20)

Felder verstecken

1. Markieren Sie ein Feld.
2. Drücken Sie **F4** (*Größe*).
3. Bestimmen Sie mit der Cursortaste **→** eine Größe von 0 Zeichen (ablesbar im rechten Teil der Statuszeile).
4. Beenden Sie mit **RETURN**.

Das Feld kann gleichwohl Daten aufnehmen, es ist lediglich auf dem Bildschirm nicht zu sehen.

(Kapitel 20)

Fixieren

Wenn eine Tabelle oder Datenbank größeren Umfang annimmt, sind die Beschriftungen der Zeilen bzw. Spalten nicht mehr zu lesen, wenn man blättert. *Fixieren erste Spalte/Zeile* im Menü *Editieren* hält die Beschriftungen gleichsam fest, wo immer man sich in der Tabelle/Datenbank befindet.

(Kapitel 22)

Formeln editieren/eingeben

1. Drücken Sie **F2** (*Formel editieren*). Framework öffnet den Formelspeicher.
2. Erweitern Sie, wenn die Formeln mehr als eine Zeile umfaßt, mit **F9** (*Zoom*) auf Bildschirmgröße.
3. Beenden Sie mit **ESC**.

RETURN von der Editierzeile aus hat die gleiche Wirkung wie **F5** (*Neuberechnung*), löst also die Formel aus.

Beim Eingeben einer neuen Formel muß der Formelspeicher nicht geöffnet werden, wenn sie mit einem dieser Zeichen beginnt:

@ \$ + - (# \$! "

Framework springt dann automatisch in die Editierzeile.

(Kapitel 17, 22)

Frame-Version

Das Gegenteil der Konzept-Version, mit **F10** (*Zeigen*) zu erreichen: Die Sub-Frames werden geöffnet.

(Kapitel 19)

Frames benennen

Wenn der Cursor auf dem Rand des Frames steht, genügt es, einfach den Namen zu schreiben. Framework springt in die Editierzeile, ein bereits bestehender Name wird gelöscht.

Soll ein Name editiert werden, wird mit der Leertaste in die Editierzeile gewechselt.

Ein Frame-Name kann 32000 Zeichen lang sein. Gespeichert indes werden, den DOS-Konventionen gemäß, nur die ersten acht Zeichen. Framework hängt dem Namen von sich aus die Endung FW an.

(Kapitel 13)

Frames kopieren

1. Markieren Sie den Frame oder, mit **[F6]** (*Auswahl*), mehrere Frames.
2. Drücken Sie **[F8]** (*Kopieren*).
3. Bringen Sie den Cursor auf das Kopierziel: in einen umfassenden Frame hinein. Soll die Kopie auf der gleichen Ebene sein, kann der Cursor bleiben.
4. Drücken Sie **[RETURN]**.

(Kapitel 13, 16)

Frames vergrößern/verkleinern

1. Bringen Sie den Cursor auf den Rand des Frames.
2. Drücken Sie **[F4]** (*Größe*).
3. Bestimmen Sie mit den Cursortasten die neue Größe.
4. Beenden Sie mit **[RETURN]**.

Zweimal **[F4]** bringt den Frame auf volle Bildschirmgröße.

Will man in einem umfassenden Frame die Größe der Sub-Frames gemeinschaftlich ändern, bedient man sich am einfachsten der Funktion *Größe ändern alle* im Menü *Frames*; Voraussetzung ist aber, daß die Sub-Frames in einer Spalte angeordnet sind.

(Kapitel 13)

Frames verschachteln

1. Bringen Sie den Cursor mit **[NUM+]** in den umfassenden Frame.
2. Erstellen Sie einen neuen Frame.

Statt eines neuen Frames kann man natürlich auch einen bestehenden kopieren oder verlagern.

Der umfassende Frame darf keine Daten enthalten, hingegen andere Frames. Der neue (oder kopierte oder verlagerte) Frame befindet sich dann auf deren Ebene.

Am einfachsten geht das Verschachteln von Frames in der Konzept-Version (siehe dort), weil man dann die Gliederungsebenen übersichtlich vor Augen hat.

(Kapitel 19)

Frames verschieben

1. Markieren Sie den Frame.
2. Drücken Sie **[F3]** (*Position*).
3. Schieben Sie den Frame mit den Cursortasten in die gewünschte Position.

(Kapitel 13)

FRED

Die Programmiersprache von Framework mit rund 140 eingebauten Funktionen, aus denen beliebige benutzerdefinierte Funktionen entstehen können.

FRED-Funktionen werden auch im Alltagsgebrauch eingesetzt, zum Beispiel für Berechnungen in Tabellen, für das Seitenformat, für Druckformeln usw.

FRED-Funktionen haben eine bestimmte Syntax:

- Sie beginnen stets mit dem Zeichen @ oder, alternativ, mit \$.
- Die Parameter der Funktion stehen in Klammern und werden durch Kommata getrennt.
- Texte (Strings), die ausgegeben werden sollen, stehen zwischen Anführungszeichen.
- Mehrere Funktionen in Folge werden durch Kommata getrennt.

(Kapitel 31-33)

Fußzeilen

Fußzeilen werden mit Formeln angewiesen (und dementsprechend nicht in den Text, sondern in den Formelspeicher des Frames geschrieben):

@fp	Position: Anzahl der Zeilen zwischen Text und Fußzeile (Voreinstellung: 3). Der Wert 0 unterdrückt die Fußzeile
@fl	Fußzeile links
@fc	Fußzeile zentriert
@fr	Fußzeile rechts
@hf	Fußzeilenbeginn: bestimmt, von welcher Seite an die Fußzeile gedruckt werden soll

In Klammern und zwischen Anführungszeichen folgt jeweils der Text sowie, optional, ein Parameter:

#even	die Fußzeile wird nur auf Seiten mit gerader Seitennummer gedruckt
#odd	die Fußzeile wird nur auf Seiten mit ungerader Seitennummer gedruckt

Zusätzlich kann mit **@pn** die aktuelle Seitennummer in die Kopfzeile integriert werden, deren Anfangswert wiederum mit **@np** bestimmt wird.

Mehrere Anweisungen können kombiniert werden, auch mit anderen FRED-Funktionen. Beispiel:

```
@np(12)
@fl("Bericht vom " & (@date(@today)) & ",Seite " & @pn,#even),
@fr("Bericht vom " & (@date(@today)) & ",Seite " & @pn,#odd),
```

Zunächst wird die Anfangsseitennummer auf 12 gesetzt, in den Fußzeilentext ist das Tagesdatum und die Seitennummer eingeschlossen; auf geraden Seiten stehen die Fußzeilen links, auf ungeraden rechts.

(Kapitel 19)

Globale Formeln

Dem Charakteristikum einer Datenbank entsprechend, lassen sich »hinter« dem Namen eines Feldes Formeln eintragen, die nur dieses Feld betreffen, dafür aber in allen Datensätzen. Framework nennt das eine globale Formel (im Gegensatz zu einer lokalen Formel, die nur für ein Feld eines einzelnen Datensatzes gilt) und verlangt dafür eine etwas kompliziertere Schreibweise:

```
@set(Feldname,Parameter)
```

Identisch damit ist:

```
Feldname:=Parameter
```

Parameter kann eine Formel, ein Text, eine Zahl sein. Beispiel:

```
@set(Umsatz,Absatz*Stückpreis)
```

```
Umsatz:=Absatz*Stückpreis
```

Die Formel steht »hinter« dem Feld *Umsatz* und trägt dort die Multiplikation aus den Feldern *Absatz* und *Stückpreis* ein.

(Kapitel 21, 24)

Grafik erstellen

Eine Grafik setzt Zahlenwerte um und verlangt deshalb eine Tabelle oder Datenbank als Grundlage.

1. Bestimmen Sie in der Tabelle oder Datenbank mit F6 (*Auswahl*) die Zahlenwerte, die in eine Grafik umgesetzt werden sollen.
2. Bestimmen Sie im Menü *Grafik*, ob die Spalte oder die Zeile die X-Achse darstellen soll.
3. Definieren Sie im Sub-Menü *Optionen* die Skalierungswerte der Y-Achse, wenn Sie manuell skalieren wollen.
4. Wählen Sie im Menü *Grafik* den Diagramm-Typ.
5. Aktivieren Sie *Neue Grafik zeichnen*.

Die Grafik wird in einem eigenen Frame erstellt und kann unabhängig von der Tabelle oder der Datenbank gespeichert werden.

Eine vorhandene Grafik kann von einer anderen überlagert werden, allerdings nur unter eingeschränkten Bedingungen:

- Die vorhandene Grafik darf kein Kreis- oder XY-Diagramm sein.
- Die überlagernde Grafik muß ein Linien- oder Punkt-Diagramm sein.

Grafiken können auch mit einer Formel »von Hand« gezeichnet werden. Das Schema:

```
@drawgraph(Bereich,Richtung,Typ,Titel,X-Titel,Y-Titel,Max-
Wert,Min-Wert, Schrittweite)
```

(Kapitel 22)

Inhaltsverzeichnis

Zwei Bedeutungen: einmal das Inhaltsverzeichnis einer Diskette oder eines Verzeichnisses - das wird, entsprechend der Framework-Bezeichnung, unter »Dateiverzeichnis« abgehandelt. Das andere ist das Inhaltsverzeichnis eines Dokumentes. Es wird aus einem Konzept erstellt:

1. Schalten Sie, wenn notwendig, mit **F10** (Zeigen) in die Konzeptversion um.
2. Aktivieren Sie *Ergänzen um Seitennummern* im Menü *Frames*.
3. Aktivieren Sie nach Wunsch auch *Durchnumerieren der Namen*.

(Kapitel 19)

Kommentare

Formeln und Programme erlauben das Einfügen von internen Kommentaren - nützlich, um Funktionen zu dokumentieren und zu erklären. Regeln:

- Der Kommentar muß in der ersten Spalte mit einem Semikolon beginnen.
- Der Kommentar darf beliebig lang sein, doch muß jede Zeile das führende Semikolon haben.

(Kapitel 26)

Konzept-Version

Das Gegenteil der Frame-Version - mit **F10** (Zeigen) schaltet man um. Die Konzept-Version stellt verschachtelte Frames nur mit ihren Namen dar, die Sub-Frames sind also geschlossen.

Das Konzept selber ist eine Ansammlung von verschachtelten Frames auf mehreren Gliederungsebenen. Ein umfassender Frame kann einen Sub-Frame enthalten, der selber wieder umfassender Frame ist für seine Sub-Frames. (Siehe auch »Frames verschachteln«.)

Auf Wunsch können über das Menü *Frames* die Sub-Frames numeriert, mit den Bezeichnungen des Frame-Typs usw. versehen werden, auch kann ein Inhaltsverzeichnis erstellt werden (siehe dort).

Ein neuer Frame kann auch von vornherein als Konzept erstellt werden, wenn man im Menü *NEU* den entsprechenden Frame-Typ wählt.

(Kapitel 19, 22, 24)

Konzept: Abschnitte verschieben

1. Markieren Sie den Sub-Frame.
2. Drücken Sie **F7** (*Verlagern*).
3. Bringen Sie den Cursor an den Zielort.
4. Drücken Sie **RETURN**.

Eingefügt wird der Abschnitt nach dem Cursor. Der Standort am Ziel bestimmt also, auf welche Gliederungsebene der Frame kommt: Steht der Cursor auf einem umfassenden Frame, erhält der verschobene Frame auch diese Ebene. Auf diese Weise kann man Abschnitte auch höher oder tiefer stufen.

Verschieben wird der Abschnitt, sofern er umfassender Frame ist, mitsamt seinen Sub-Frames, die dafür nicht eigens markiert werden müssen - das ist das Prinzip des Konzeptes.

(Kapitel 19, 22)

Konzept: Gliederungsebenen

Ein Konzept besteht aus gestaffelten Gliederungsebenen, deren Hierarchie am deutlichsten wird, wenn man über das Menü *Frames* die Frames durchnummerieren läßt:

- 1 Erste Ebene
 - 1.1 Zweite Ebene
 - 1.2 Zweite Ebene
 - 1.2.1 Dritte Ebene
 - 1.2.2 Dritte Ebene
 - 1.2.3 Dritte Ebene
 - 1.3 Zweite Ebene
 - 1.3.1 Dritte Ebene
 - 1.4 Zweite Ebene
- 2 Erste Ebene
 - 2.1 Zweite Ebene

Klarheit über die Gliederungsebene ist nötig, wenn man Abschnitte verschiebt bzw. kopiert und sie auf diese Weise auch anders einstufen will.

Umfassende Frames, gleich welcher Ebene, werden von Framework mit einem Dreieck gekennzeichnet. `[RETURN]` öffnet bzw. schließt die Ebenen darunter.

(Kapitel 19, 22)

Konzept: Neuer Abschnitt

Weil ein Konzept aus verschachtelten Frames besteht, gelten für einen neu einzufügenden Abschnitt die allgemeinen Verschachtelungs-Prinzipien. Heißt: Um einen Sub-Frame zu erstellen, darf der umfassende Frame keine Daten enthalten. Wie ein neuer Abschnitt eingefügt wird, hängt also davon ab, auf welcher Ebene er eingefügt wird..

Auf gleicher Ebene:

1. Bringen Sie den Cursor auf den Frame.
2. Erstellen Sie über das Menü *NEU* einen neuen Frame.

Der neue Frame wird, wie üblich, nach der Cursorposition eingefügt, deshalb kann er auf seiner Ebene erst an zweiter Stelle stehen - mit einer kleinen Schiebung regelt man das Problem.

Ein neuer Unterabschnitt, also eine neue Unterebene:

1. Bringen Sie den Cursor auf den Frame
2. Drücken Sie `[CTRL]` `[NUM+]`.

Das geht natürlich nur, wenn der Frame leer ist. Ist er's nicht, ist einige Fummelei mit Verschieben notwendig.

(Kapitel 19, 22)

Kopfzeilen

Kopfzeilen werden wie Fußzeilen mit Formeln angewiesen (und dementsprechend nicht in den Text, sondern in den Formelspeicher des Frames geschrieben):

@hp	Position: Anzahl der Zeilen zwischen Text und Kopfzeile (Voreinstellung: 3). Der Wert 0 unterdrückt die Kopfzeile
@hl	Kopfzeile links
@hc	Kopfzeile zentriert
@hr	Kopfzeile rechts
@hf	Kopfzeilenbeginn: bestimmt, von welcher Seite an die Kopfzeile gedruckt werden soll

In Klammern und zwischen Anführungszeichen folgt jeweils der Text sowie, optional, ein Parameter:

#even die Kopfzeile wird nur auf Seiten mit gerader Seitennummer gedruckt
#odd die Kopfzeile wird nur auf Seiten mit ungerader Seitennummer gedruckt

Zusätzlich kann mit **@pn** die aktuelle Seitennummer in die Kopfzeile integriert werden, deren Anfangswert wiederum mit **@np** bestimmt wird.

Mehrere Anweisungen können kombiniert werden, auch mit anderen FRED-Funktionen. Beispiel:

```
@np(12)
@hl("Bericht vom " & (@date(@today)) & ",Seite " & @pn,#even),
@hr("Bericht vom " & (@date(@today)) & ",Seite " & @pn,#odd),
```

Zunächst wird die Anfangsseitennummer auf 12 gesetzt, in den Kopfzeilentext ist das Tagesdatum und die Seitennummer eingeschlossen; auf geraden Seiten stehen die Kopfzeilen links, auf ungeraden rechts.

(Kapitel 19)

Kopieren

1. Markieren Sie mit **[F6]** (*Auswahl*) den Bereich - Textteil, Spalten/Zeilen in Tabelle oder Datenbank, Frames.
2. Drücken Sie **[F8]** (*Kopieren*).
3. Bringen Sie den Cursor an den Zielort.
4. Drücken Sie **[RETURN]**.

Die Markierung bleibt erhalten, so daß Sie unmittelbar anschließend erneut kopieren können.

Das Makro **[ALT][F5]** aus MACLIB stellt den markierten Bereich in einen Puffer, von wo er mit **[ALT][F6]** wieder abgeholt wird. Der Puffer bleibt auch erhalten, wenn der Computer ausgeschaltet wird.

(Kapitel 16, 27)

Lokale Formeln

Eine Formel in einer Datenbank »hinter« einem Feld, die nur für dieses einzelne Feld in diesem Datensatz gilt. Im Unterschied dazu wirkt sich eine globale Formel auf ein Feld in allen Datensätzen aus (siehe dort).

(Kapitel 21)

MAILUTIL

Ein Framework-Dienstprogramm für Serienbriefe und Etikettendruck. MAILUTIL nimmt die Adressen aus einer Datenbank, die DATENBANK heißen muß, und speist sie in einen Brief ein, dessen Name beliebig gewählt werden kann. Im Brief stehen Platzhalter für die Felder der Datenbank (siehe unter »Serienbriefe«).

1. Laden Sie MAILUTIL, die Datenbank und den Frame mit dem Brief.
2. Selektieren Sie, falls nötig, die Datenbank.
3. Bringen Sie den Cursor auf MAILUTIL und drücken Sie **[F5]** (*Neuberechnung*).
4. Bringen Sie den Cursor auf oder in den Brief-Frame.
5. Drücken Sie **[ALT]**-M für Serienbriefe oder **[ALT]**-E für Etiketten.

(Kapitel 21)

Makro-Bibliothek

Eine Sammlung von Makros, die zur Verfügung stehen, solange sich die Makro-Bibliothek auf der Arbeitsfläche befindet. Aktiviert wird sie mit **[F5]** (*Neuberechnung*).

Framework liefert die Makro-Bibliothek MACLIB mit; eine andere mit Namen MAKRO befindet sich auf der Beispieldiskette zu diesem Buch.

Eine Makro-Bibliothek ist ein verschachtelter Frame, dessen Sub-Frames im Formelspeicher das jeweilige Makro enthalten. Ein Sub-Frame ist ein Zuweisungsframe, der Framework mitteilt, wo das jeweilige Makro zu suchen ist. Allgemeine Syntax hierfür:

```
@setmacro({alt-Taste},Frame)
```

(Kapitel 27, 28)

Makros erstellen

Ein Makro verlangt eine Formel, mit der es aktiviert wird und die Framework mitteilt, in welchem Frame welches Makro für welche Taste abgelegt ist. Das Schema:

```
@setmacro({alt-Taste},Frame)
```

Die gleiche Formel ohne Bezug auf den Frame deaktiviert das Makro.

MACLIB beinhaltet das Makro-Erstellungs-Makro **[ALT][F2]**, welches alle notwendigen Schritte erledigt und das Makro in MACLIB speichert:

1. Aktivieren Sie MACLIB mit **[F5]** (*Neuberechnung*).
2. Drücken Sie **[ALT][F2]**.
3. Drücken Sie **[ALT]** sowie die Taste, mit der das Makro abgerufen werden soll - der Beginn der Makro-Definition.
4. Alle folgenden Tastenanschläge werden gespeichert und ergeben das Makro.

5. Drücken Sie abermals **[ALT]** sowie die Taste, mit der das Makro abgerufen werden soll - das Ende der Makro-Definition.

Als Makro-Tasten stehen die Funktionstasten, die Zahlen von 0 bis 9 sowie die Buchstaben des Alphabets zur Verfügung.

In einem Makro werden alle Tastenanschläge gespeichert, auch von Sondertasten wie den Cursortasten. Framework schreibt hierfür verbindliche Schreibweisen vor.

(Kapitel 26)

Masken-Version

Das Gegenteil der Tabellen-Version bei einer Datenbank: Jedes Feld ist ein Frame für sich; die Frames können beliebig vergrößert und positioniert werden. Die Masken-Version kann auch gedruckt werden.

(Kapitel 21)

Neuberechnung

[F5] (*Neuberechnung*) ist wahrscheinlich die am häufigsten benutzte Funktionstaste in Framework. *Neuberechnung* meint zunächst einmal, ganz wörtlich: Eine Tabelle oder Datenbank wird neu berechnet. Im Sub-Menü *Optionen für Neuberechnung* des Menüs *Zahlen* kann bestimmt werden, wie das geschehen soll:

- *Automatisch* (dann wird freilich auch ohne Zutun von **[F5]** neu berechnet, sobald sich Daten ändern) oder *Manuell* (jetzt wird nur mit **[F5]** gerechnet)
- *Standard* (dann wird logisch gerechnet: in Abhängigkeit zu den Formeln, mit denen die Daten verbunden sind) oder *Zeilenweise* (Zeile für Zeile, von oben nach unten).

Neuberechnung meint aber auch, daß Formeln aktiviert werden - also auch Programme, die nicht rechnen im üblichen Sinn. Faustregel: Wo immer sich eine Formel, gleich welcher Art, befindet, wird sie mit **[F5]** (*Neuberechnung*) ausgelöst.

(Kapitel 22)

Primär-Laufwerk

Dasjenige Laufwerk, dem Framework die Daten-Dateien (nicht die Programm-Dateien!) entnimmt und wo die Frames auch gespeichert werden. Das Primär-Laufwerk wird in CONFIG.FW definiert; als Voreinstellung wird das in der alphabetischen Reihenfolge letzte Laufwerk genommen - das aber auch ein logisches, in Wahrheit also nicht vorhandenes Laufwerk sein kann.

Für die Dauer einer Arbeitssitzung kann man mit dem DOS-Zugriff ins Betriebssystem gehen und dort das aktuelle Laufwerk ändern.

(Kapitel 7)

Programm-Bibliothek

Benutzerdefinierte Funktionen (also FRED-Programme) können in der Programm-Bibliothek LIB abgelegt werden (als Sub-Frames). Stößt Framework im Verlauf eines Programmes auf eine solche Funktion und wird in ihr nicht Bezug genommen auf einen bestimmten Frame, sucht Framework in LIB danach - vorausgesetzt, diese Programm-bibliothek befindet sich auf der Arbeitsfläche. Der Name LIB ist Vorschrift.

(Kapitel 33)

Rücknahme

Zu aktivieren im Menü *Editieren*. Macht den letzten Befehl rückgängig - zum Beispiel eine Löschung. Werden aus einer Tabelle oder Datenbank Spalten oder Zeilen entfernt, ist eine Rücknahme allerdings nicht möglich.

Schriftauszeichnung

1. Markieren Sie den Text, das Feld in einer Datenbank oder die Zelle in einer Tabelle.
2. Öffnen Sie das Menü *Text*.
3. Wählen Sie die Schriftauszeichnung.

Fettdruck, Unterstreichen und Schrägschrift können kombiniert werden, allerdings nicht in einem Arbeitsgang.

(Kapitel 19)

Seitenformat

Anweisungen, wie ein Dokument beim Ausdruck auf der Seite positioniert wird. Eine Reihe von Formeln, die in den Formelspeicher »hinter« den Frame einzutragen sind:

@pl	Seitenlänge in Zeilen
@ll	Zeilenlänge in Zeichen
@tm	oberer Rand (Leerzeilen zwischen Papierrand und Text)
@bm	unterer Rand (Leerzeilen zwischen Papierrand und Text)

@po	Zeilenanfang (Randbreite links vom Papierrand bis zum Text in Zeichen)
@sp	Zeilenabstand
@sk	Leerzeilen einfügen (bevor der Ausdruck des Frames beginnt)

Siehe auch »Druckformeln«, »Fußzeilen«, »Kopfzeilen« und »Seitennummer«.

In Text-Frames können linker und rechter Rand auch über das Menü *Text* eingestellt werden. Die Voreinstellungen hierfür wie auch für die anderen Formatierungswerte sind in CONFIG.FW definiert.

(Kapitel 21)

Seitennummern

Framework beginnt die Seitenzählung bei 1. Das kann durch Druckformeln beeinflusst werden:

@np	ein Wert in Klammern definiert den Anfang der Seitenzählung
@pn	bringt die Seitennummer in eine Kopf- oder Fußzeile ein

(Kapitel 19)

Selektieren

Ein Auszug aus einer Datenbank nach bestimmten Bedingungen. Eine Formel dafür (Framework nennt sie einen Filter) ist in den Formelspeicher der Datenbank einzutragen. Allgemeines Schema:

Feldname=Bedingung

Besteht die Bedingung aus Text, muß er in Anführungszeichen stehen.

Feldname und Bedingung können mit logischen Operatoren verknüpft werden, außerdem lassen sich mehrere Bedingungen kombinieren:

>	größer als
<	kleiner als
=	gleich
> =	größer als oder gleich
< =	kleiner als oder gleich
< >	ungleich
@and	und
@or	oder
@not	nicht

In der Bedingung können auch die Ersatzzeichen ? (ein einzelnes Zeichen) und * (mehrere Zeichen) verwendet werden.

Ein Beispiel, der Schreibweise wegen:

```
@or(postleitzahl<7300,name="Müller",umsatz>=3000)
```

Die selektierten Datensätze können beliebig kopiert, verlagert oder gelöscht werden.

Soll von den selektierten Sätzen wieder umgeschaltet werden auf die Gesamtheit aller Datensätze, steht hierfür die Funktion *Öffnen alle* im Menü *Frames* zur Verfügung (der Cursor muß auf dem Rand des Frames sein!). Oder:

1. Öffnen Sie den Formelspeicher mit **[F2]** (*Formel editieren*).
2. Löschen Sie die Selektions-Formel.
3. Drücken Sie **[F5]** (*Neuberechnung*).

Bei der Menü-Methode bleibt die Formel erhalten und kann mit **[F5]** jederzeit wieder aktiviert werden.

(Kapitel 20, 21)

Serienbriefe

Das Framework-Dienstprogramm MAILUTIL vereinfacht Serienbriefe (und Etikettendruck), wobei die »Serie« auch aus einem einzelnen Brief bestehen kann.

MAILUTIL nimmt die Adressen aus einer Datenbank, die DATENBANK heißen muß und speist sie in einen Brief ein, dessen Name beliebig gewählt werden kann. Im Brief stehen Platzhalter für die Felder der Datenbank: die Namen der Felder, von spitzen Klammern umschlossen. Die besondere Variable **<opt>** veranlaßt Framework, leere Felder zu überspringen.

1. Laden Sie MAILUTIL, die Datenbank und den Frame mit dem Brief.
2. Selektieren Sie, falls nötig, die Datenbank.
3. Bringen Sie den Cursor auf MAILUTIL und drücken Sie **[F5]** (*Neuberechnung*).
4. Bringen Sie den Cursor auf oder in den Brief-Frame.
5. Drücken Sie **[ALT]-M**.

(Kapitel 21)

Sonderdienste

Ein Punkt im Menü *Laufwerk*, der nur mit einem kleinen Demo-Programm belegt ist, in den Sie aber hineinpacken können, was Sie wollen: Sie können auf diese Weise eigene Programme über ein Framework-Menü aufrufen und ablaufen lassen. Voraussetzungen:

- Das Programm muß UTIL heißen.
- UTIL muß sich auf der Arbeitsfläche befinden.

Neu berechnet zu werden braucht es nicht, das wird erledigt, wenn Sie den Menüpunkt *Sonderdienste* aktivieren. Dessen Namen übrigens können Sie in CONFIG.FW frei bestimmen.

(Kapitel 39)

Sortieren

Ordnet die Sätze einer Datenbank oder die Zeilen einer Tabelle neu, das heißt, lediglich die Reihenfolge wird geändert, sonst nichts:

1. Markieren Sie das Feld, nach dem sortiert werden soll.
2. Aktivieren Sie *Vorwärts (aufsteig.) sortieren* oder *Rückwärts (absteig.) sortieren* im Menü *Suchen*.

(Kapitel 20, 30)

Spaltenanordnung

Ist diese Funktion (*Ausrichten in Spalte*) im Menü *Frames* aktiviert, sind Sub-Frames innerhalb eines umfassenden Frames nacheinander angeordnet. Das Gegenteil: Die Position der Sub-Frames ist frei wählbar, sie können also auch nebeneinander stehen.

Spalten entfernen

1. Markieren Sie die Spalten.
2. Aktivieren Sie *Spalten/Felder entfernen* im Menü *Editieren*.

Framework warnt, daß die Operation nicht rückgängig zu machen sei - also besser zweimal hinschauen, ob auch die richtigen Spalten markiert sind.

Spaltenbreite

Die Größe einer Spalte in einer Tabelle kann beliebig bestimmt werden; Änderungen einer Zelle in einer Spalte wirken sich auf die gesamte Spalte aus:

1. Markieren Sie ein Feld.
2. Drücken Sie **F4** (*Größe*).
3. Bestimmen Sie mit den Cursortasten **+** und **-** die gewünschte Größe. Im rechten Teil der Statuszeile ist die Größe in Zeichen abzulesen.
4. Beenden Sie mit **RETURN**.

Die Spaltenbreite hat nichts damit zu tun, wieviel Zeichen eine Zelle aufnehmen kann, sondern bestimmt lediglich die Darstellung auf dem Bildschirm. Demzufolge kann man Spalten auch verstecken, indem man eine Größe von 0 Zeichen definiert.

(Kapitel 22)

Spalten verstecken

1. Markieren Sie eine Zelle.
2. Drücken Sie **[F4]** (*Größe*).
3. Bestimmen Sie mit der Cursortaste **[←]** eine Größe von 0 Zeichen (ablesbar im rechten Teil der Statuszeile).
4. Beenden Sie mit **[RETURN]**.

Damit sind alle Zellen dieser Spalte versteckt. Sie können gleichwohl Daten aufnehmen, sie sind lediglich auf dem Bildschirm nicht zu sehen.

(Kapitel 30)

Speicherplatz abfragen

Weil Framework seine Daten-Dateien im Arbeitsspeicher hält, ist deren Umfang begrenzt; den noch freien Platz abzufragen, kann deshalb hin und wieder notwendig werden.

Zwei Möglichkeiten:

- das Makro **[ALT][F7]** in MACLIB; der noch freie Platz im Hauptspeicher wird in der Nachrichtenzeile angegeben und verschwindet beim nächsten Tastendruck wieder
- die Funktion MEMAVAIL in LIB: ein winzig kleiner Frame, der herauskopiert und in irgend eine Ecke der Arbeitsfläche gestellt werden kann, wo er zur steten Verfügung bleibt.

(Kapitel 27, 33)

Suchen

Framework sucht nach einem vorgegebenen Text und hält dort an. Das ist gleichsam die halbe Ersetzen-Funktion, dementsprechend gelten auch die gleichen Regeln.

Gesucht werden kann in allen Frame-Typen, bei Datenbank und Tabelle jedoch nur spaltenweise; der Cursor muß zuvor in die betreffende Spalte gebracht werden.

1. Öffnen Sie das Menü *Suchen*.
2. Bestimmen Sie gegebenenfalls die Optionen für die Berücksichtigung von Namen, Inhalt, Formeln bzw. Großschreibung.

3. Aktivieren Sie die Funktion *Suchen nach*. Daraufhin öffnet sich der Formelspeicher.
4. Geben Sie den Such-Text ein.
5. Drücken Sie **RETURN**.

Ist der Text gefunden worden, haben Sie die Wahl:

- ☐ Framework sucht nach dem nächsten Fundort
- ☐ Framework sucht nach dem vorhergehenden Fundort

Gefunden werden auch einzelne Zeichen in Wörtern, zudem kann mit Ersatzzeichen gearbeitet werden:

- ? steht für ein beliebiges Zeichen (M??er z.B. findet Meier, Meyer, Maier, Mayer, Mauer, Maler usw.)
- * steht für mehrere beliebige Zeichen (M*er z.B. findet auch Meer, Meyerbeer, Müller, Malermeister, Mutter usw.)

Mit den Optionen kann definiert werden, ob

- die Namen von Frames
- der Inhalt von Frames
- die Formeln »hinter« Frames oder Zellen
- die Großschreibung

berücksichtigt werden soll. Die Optionen können beliebig kombiniert werden.

Auch die unsichtbaren Zeichen für **TAB** und Leertaste können gesucht werden, sofern sie sichtbar gemacht sind; **RETURN** allerdings nicht.

(Kapitel 20)

Tabellen verbinden

Woher eine Formel ihre Werte nimmt, ist gleichgültig, es kann also auch eine andere Tabelle oder eine Datenbank sein. Die Tabelle oder Datenbank muß nur erreichbar sein - das heißt, sie muß sich auf der Arbeitsfläche befinden, und der Bezug zu ihr muß eindeutig hergestellt werden.

Am einfachsten und sichersten wird hierfür die Cursormethode benutzt (siehe dort); dann trägt Framework selber die erforderlichen Bezüge ein.

(Kapitel 24, 30)

Tabelle vergrößern

Eine Tabelle wird mit den - in CONFIG.FW definierten - Werten eingerichtet, die im Menü *NEU* angegeben sind, hier aber auch abgeändert werden können. Eine bestehende Tabelle kann nachträglich erweitert werden:

1. Öffnen Sie das Menü *NEU*.
2. Aktivieren Sie *Spalten/Felder (# Einfügen)* bzw. *Zeilen/Sätze (# Einfügen)*. Darauf wird die Editierzeile geöffnet.
3. Geben Sie die Zahl der Felder bzw. Sätze an.

Die Spalten/Zeilen werden in der Tabelle unmittelbar nach dem Cursor eingefügt.

(Kapitel 22)

Tabellen-Version

Das Gegenteil der Masken-Version bei einer Datenbank: Die Datensätze sind Zeile für Zeile untereinander angeordnet. Die übliche Form der Darstellung.

(Kapitel 21)

Text ausrichten

1. Markieren Sie den Text.
2. Öffnen Sie das Menü *Text*.
3. Wählen Sie unter *Ausrichten nach links* (linksbündig), *Verschieben nach rechts* (rechtsbündig), *Blocksatz* und *Zentriert*.

Steht der Cursor auf dem Rand des Frames, gelten die Anweisungen global für den gesamten Frame.

In Tabellen und Datenbanken werden ähnliche Ausrichtungen über das Menü *Zahlen* vorgenommen.

(Kapitel 22)

Unsichtbare Zeichen

Wenn die Funktion *Unsichtbare Zeichen darstellen* im Menü *Editieren* aktiv ist, werden die Zeichen ebenfalls dargestellt, die Framework sonst, der besseren Übersichtlichkeit wegen, versteckt:

- RETURN
- TAB
- Leerzeichen

Verlagern

1. Markieren Sie mit **[F6]** (*Auswahl*) den Bereich - Textteil, Spalten/Zeilen in Tabelle oder Datenbank, Frames.
2. Drücken Sie **[F7]** (*Verlagern*).
3. Bringen Sie den Cursor an den Zielort.
4. Drücken Sie **[RETURN]**.

(Kapitel 16)

Verzeichnis einrichten

Das geht, indem man mit dem DOS-Zugriff (siehe dort) ins Betriebssystem wechselt, oder mit dem Hilfsprogramm DISKUTIL, allerdings auch nicht bequemer:

1. Laden Sie DISKUTIL.
2. Drücken Sie **[F5]** (*Neuberechnung*).
3. Drücken Sie **[ALT]-D**.

Daraufhin erscheint in der Editierzeile ein Menü.

(Kapitel 41)

Verzeichnis wechseln

Die Directory-Hierarchie auf einer Diskette oder Platte behandelt Framework ähnlich wie einen Konzept-Frame:

1. Bringen Sie den Cursor auf den Rand des Dateiverzeichnis-Frames.
2. Drücken Sie **[NUM-]**. Framework listet nun die Verzeichnisse auf, mit einem Dreieck vornedran.
3. Gehen Sie mit **[NUM+]** wieder in den Frame.
4. Bringen Sie den Cursor auf ein Verzeichnis.
5. Drücken Sie **[RETURN]**.

Framework listet alle Dateien des Verzeichnisses auf, sowie, wiederum mit Dreiecken versehen, die Unterverzeichnisse.

Voreinstellungen

Die Datei CONFIG.FW funktioniert innerhalb Framework wie CONFIG.SYS innerhalb des Betriebssystems: Beim Start wird sie durchsucht, sie legt Voreinstellungen fest für:

- Größe der Programmdateien im Arbeitsspeicher
- Größe des Pufferbereichs im Arbeitsspeicher
- Primär-Laufwerk
- Größe des Dateiverzeichnis-Frames
- Größe eines neuerstellten Frames
- Breite der Spalten/Felder in Tabellen und Datenbanken
- Anzahl der Spalten/Zeilen in Tabellen und Datenbanken
- Text-Voreinstellungen (Rand, Absatz-Einzug, **TAB**-Sprung)
- Drucker-Kanäle
- Druckgeschwindigkeit
- Zusammenhalt eines Absatzes beim Druck
- Seitenformat
- maximaler Durchlauf einer **@while**-Schleife
- Zahlenformate
- Größe des DOS-Zugriffs-Frames
- Text anstelle von *Sonderdienste* im Menü *Laufwerk*
- Geschwindigkeit, mit der Frames geöffnet und geschlossen werden
- Einsatz des mathematischen Co-Prozessors 8087
- Zugriffssperre auf Formeln

Sämtliche Änderungen werden erst beim Neustart von Framework wirksam.

(Kapitel 8)

Zahlen ausrichten

1. Markieren Sie eine Zelle bzw. ein Feld oder mehrere.
2. Öffnen Sie das Menü *Zahlen*.
3. Wählen Sie unter *Texte links/Zahlen rechts*, *Linksbündig*, *Rechtsbündig* und *Zentriert*.

Steht der Cursor auf dem Rand des Frames, gelten die Anweisungen global für den gesamten Frame. Zuvor individuell ausgerichtete Zellen oder Felder werden davon jedoch nicht berührt.

(Kapitel 22)

Zahlenformate

Das Zahlenformat aus dem Menü *Zahlen* kann individuell (für die markierten Zellen/Felder) oder global für den gesamten Frame angewiesen werden (dann muß der Cursor auf dem Rand des Frames stehen):

Dezimalstellen	die Anzahl der Dezimalstellen, die angezeigt werden
Standardformat	7654,32

Ganze Zahl	7654
Festpunktzahl	7654,32
Währungsbetrag	DM 7.654,32
Unterteilung in Tausender	7.654,32
Prozent	765432,00%
Exponentendarstellung	7,65E+3

In CONFIG.FW (oder durch entsprechende FRED-Funktionen) kann festgelegt werden:

- das Währungssymbol
- Position des Währungssymbols (vor oder nach der Zahl)
- Zahlenformat Tausender-Punkt und Dezimalkomma oder Tausender-Komma und Dezimalpunkt

(Kapitel 23)

Zeilen entfernen

1. Markieren Sie die Zeilen.
2. Aktivieren Sie *Spalten/Felder entfernen* im Menü *Editieren*.

Framework warnt, daß die Operation nicht rückgängig zu machen sei - also besser zweimal hinschauen, ob auch die richtigen Zeilen markiert sind.

Zellen schützen

1. Markieren Sie die Zellen.
2. Aktivieren Sie *Gegen Änderungen schützen* im Menü *Editieren*.

Ein nur teilweiser Schutz. Die Zellen können zwar mit **[DEL]** nicht gelöscht werden, auch kann direkt kein neuer Wert eingegeben werden - der Formelspeicher indessen läßt sich weiterhin öffnen.

(Kapitel 23)

Zoomen

[F9] (*Zoom*) bringt einen Frame oder Formelspeicher auf volle Bildschirmgröße (und wieder zurück); der Rahmen ist dann nicht mehr sichtbar.

Unerläßlich ist das, wenn man Formeln eingibt oder editiert, die mehr als eine Zeile umfassen. Verlassen wird der (bildschirmgroße) Formelspeicher mit **[ESC]**, was allerdings keine automatische Ausführung der Formel bewirkt. Dazu ist **[F5]** (*Neuberechnung*) nötig.

(Kapitel 13)

Bibliographie

Auch andere Autoren haben schon gedacht, experimentiert, formuliert - warum sollte man das mißachten. Anregungen, Tips, Bestätigungen verdanke ich:

Carr, Robert: A Programmer's Reference. Culver City 1984

Fishback, Clark: Framework: On-the-Job-Applications. Culver City 1985

Kling, Bill: Framework - A Developer's Handbook. Culver City 1985

Kolberg, Michael: Integrierte Datenverarbeitung mit Framework II. Haar 1986

Rodriguez, Suzanne Marie: Introduction to Programming. Culver City 1985

Sedlik, Jay M. (Ed.): Inside Framework. New Jersey 1984 --

Siegert, Andreas: Umsatzsteuerberechnung mit Framework II. PC Magazin 43 (1986)

Stichwortverzeichnis

#even 90, 281, 286

#odd 90, 281, 286

@and 101

@avg 131

@beep I 253

@bm 120, 289

@business 118

@CAGR 212

@currency 118, 197, 247

@date1 91

@date2 91

@date3 91

@date4 91, 196

@Datum5 215, 217

@dbasefilter 263, 274

@decimal 197

@draw 267

@DrawGraph 144

@drawgraph 283

@echo(#OFF) 196, 235

@echo(#ON) 196, 235

@eraseprompt 195

@execute 236, 243

@fc 90, 178, 281

@fl 90, 178, 281

@fp 90, 121, 281

@fr 90, 178, 281

@hc 90, 178, 285

@hf 281, 285

@hide 260

@hl 90, 178, 285

@hlookup 198

@hp 90, 120, 285

@hr 90, 178, 285

@Hundertstel 215, 216

@if 208, 239

@inputline 208, 245

@integer 118, 206

@Jahr 213, 216

@keyfilter 232, 234, 237

@kp 276

@list 239

@ll 120, 289

@local 199, 208, 246

@max 198

@memavail 216, 217, 293

@menu 191, 232

@min 198

@Minute 214, 216

@Monat 213, 216

@nextkey 195

@not 101

@np 89, 178, 276, 290

@or 101

@performkeys 171, 223

@pl 120, 289

@pn 90, 281, 286, 290

@po 120, 290

@pr 276

@print 276

@printreturn 232

@prompt 195, 245

@quitmenu 191, 233

@rept 205

@run 262

@Sekunde 214, 216

@set 158, 199, 246

@setformula 236

@setmacro 175, 183, 287

@setselection 223, 234

@sk 290

@sp 121, 290

@st 276

@Stunde 214, 216

@sum 130, 203, 206

@Tag 213, 216

@time3 216

@tm 120, 289

@today 91, 196

@trace 215, 218

@vlookup 198

@while 208

@Wochentag 215, 217

@Zeitberechnung 212, 216

A

Absatz-Einzug 122, 271

Abschnitts-Balken 136

absolute Adressierung 129, 154

Adressierung 271

- absolut 129, 154

- relativ 128

Arbeitsfläche 45

Arbeitsspeicher 32, 36, 54, 293

Argument 205

ASCII-Format 263

AUTOEXEC.BAT 28, 30, 251

B

Balkendiagramm 134, 135

Batch-Datei 29, 30, 258

Bedingungen 208

Befehlszeilenparameter 30, 251

Benutzeroberfläche 12, 64

Bezug 128, 207

Blocksatz 69

C

CONFIG.FW 33, 35, 36, 51, 58,
121, 209, 296

Cursorbewegungen 64

Cursormethode 130, 161, 271

D

Dateiverzeichnis 37, 45, 51, 273

Datenbank vergrößern 273

Datenbank-Größe 38

Datenbanken erweitern 58

Dateneingabe 71, 117, 273

Datensatz 65

Datenspeicher 52

Datensätze entfernen 274

dBase 263, 274

Debugging 218

DIF-Dateien 266

Disketten

- formatieren 260, 274

- kopieren 260, 274

- testen 260, 274

DISKUTIL 32, 260, 274, 275

Dokumentenstapel 45

DOS-Zugriff 35, 40, 257, 260,
274, 275

Drucken 73, 275

Drucker-Steuerzeichen 276

Druckformeln 39, 88, 120, 276

Druckgeschwindigkeit 38

E

Editierzeile 46, 64
Einfügen 71
Ersetzen 276
Etiketten 112, 277

F

Fehlersuche 218
Feld 65
Feldbreite 37, 278
Felder entfernen 278
Felder verstecken 278
Fettdruck 69, 87, 289
Filter 290
Filter-Formel 97
Fixieren 279
Formatierung 70
Formel editieren 71
Formeleingabe 71
Formeln
- editieren 279
- eingeben 279
Formelspeicher 52
Frame-Größe 37
Frame-Typen 53, 57, 85
Frame-Version 61, 85, 279
Frames
- benennen 58, 279
- erstellen 57
- kopieren 68, 280
- laden 53, 272
- löschen 59, 67, 272
- markieren 66
- öffnen 54
- schließen 55
- sichern 57
- speichern 56
- umfassender Frame 61, 79
- vergrößern 55, 280
- verkleinern 280
- verlagern 60, 69
- verschachteln 60, 162, 280
- verschieben 55, 280
- zoomen 62
Framework installieren 21
- Disketten-Systeme 22

- Drucker und Plotter 26
- Festplatte 24
- Monitor 25

Framework starten 28
Framework-Bildschirm 45
FRED 203, 281
Funktion 205
Funktionstasten 49
Fußzeile 89, 121, 281

G

Globale Formeln 115, 158, 282
Grafik erstellen 282
Grafik zeichnen 141
Grafik-Typ 132

H

Hilfe-System 50

I

Inhaltsverzeichnis 87, 283

J

Joker 103

K

Kommentare 206, 283
Konzept-Version 61, 77, 283
- Abschnitte verschieben 81
- Gliederungsebenen 79
- Neuer Abschnitt 81
Konzept
- Abschnitte verschieben 284
- Gliederungsebenen 284
- Neuer Abschnitt 285
Koordinaten 65
Kopfzeile 89, 120, 285
Kopieren 68, 286
Kreisdiagramm 137

L

LIB 211, 289
Linie 138
linksbündig 69

Lokale Formeln 115, 286
Löschen 67

M

MAILUTIL 107, 277, 287, 291
Makro 203
Makro-Bibliothek 287
Makros
- erstellen 287
manuelle Skalierung 142
Marker 66
Markieren 66
Maske 157
Masken-Version 288
Maskendarstellung 99
Menüs 47
Menüzeile 45

N

Nachrichtenfeld 46
Nachrichtenzeile 46
Neuberechnung 131, 288
Normalschrift 69

O

<opt> 112

P

Parameter 205
Primär-Laufwerk 35, 37, 258, 288
Programm-Bibliothek 211, 289
Punkte 139

R

Randeinstellungen 120, 122, 289
rechtsbündig 69
relative Adressierung 128
Rücknahme 67, 289

S

Schablone 119
Schleifen 208
Schriftauszeichnung 69, 87, 126, 289

Schrägschrift 69, 87, 289
 Seitenformat 38, 39, 120, 289
 Seitenlänge 120, 289
 Seitennummer 87, 290
 Selektieren 96, 290
 Serienbriefe 107, 291
 Sonderdienste 40, 256, 291
 Sortieren 94, 292
 Spalte 65
 Spalten entfernen 292
 Spalten verstecken 293
 Spaltenanordnung 292
 Spaltenbreite 37, 127, 292
 Startroutine 251
 Statuszeile 46
 String 117, 206, 273
 Suchen 104, 293

T

Tabelle vergrößern 295
 Tabellen erweitern 58
 Tabellen verbinden 160, 294
 Tabellen-Größe 38
 Tabellen-Version 295
 Tabellendarstellung 99
 Tastaturbezeichnungen 252

Tastaturfilter 237
 Text ausrichten 295
 Textdatei für DOS 74, 258
 Turmdiagramm 136

U

überschreiben 71
 Unsichtbare Zeichen 295
 Unterstreichen 69, 87, 289
 UTIL 256, 291

V

Variablen 208
 Vergleichsoperator 97
 Verlagern 69, 296
 Verzeichnis anlegen 260
 Verzeichnis einrichten 296
 Verzeichnis wechseln 296
 VisiCalc 266
 Voreinstellungen 296

W

Wiederhol-Funktion 254
 WKS-Tabellen 265

WordStar-Dateien 264
 WS2FW-Hilfsprogramme 264

X

X-Achse 132
 X-Y 140

Y

Y-Achse 132

Z

Zahlen ausrichten 297
 Zahlenformat 40, 152, 297
 Zeile 65
 Zeilen entfernen 298
 Zeilenabstand 121, 290
 Zeilenlänge 120, 289
 Zelle 65
 Zellen schützen 152, 298
 zentriert 69
 Zoomen 298

Bitte schneiden Sie diesen Coupon aus, und schicken Sie ihn in einem Kuvert an:
Markt & Technik Verlag AG, Buchverlag, Hans-Pinsel-Straße 2, 8013 Haar



Computerliteratur und Software vom Spezialisten

Vom Einsteigerbuch für den Heim- oder Personalcomputer-Neuling über professionelle Programmierhandbücher bis hin zum Elektronikbuch bieten wir Ihnen interessante und topaktuelle Titel für

• Apple-Computer • Atari-Computer • Commodore 64/128/16/116/Plus 4 • Schneider-Computer • IBM-PC, XT und Kompatibel
sowie zu den Fachbereichen Programmiersprachen • Betriebssysteme (CP/M, MS-DOS, Unix, Z80) • Textverarbeitung • Datenbanksysteme • Tabellenkalkulation • Integrierte Software • Mikroprozessoren • Schulungen.
Außerdem finden Sie professionelle Spitzen-Programme in unserem preiswerten Software-Angebot für Amiga, Atari ST, Commodore 128, 128 D, 64, 16, für Schneider-Computer und für IBM-PCs und Kompatibel!

Fordern Sie mit dem nebenstehenden Coupon unser neuestes Gesamtverzeichnis und unsere Programmservice-Übersichten an, mit hilfreichen Utilities, professionellen Anwendungen oder packenden Computerspielen!

Adresse:

Name

Straße

Ort

Bitte schicken Sie mir:

- ☐ Ihr neuestes Gesamtverzeichnis
☐ Eine Übersicht Ihres Programmservice-Angebotes aus der Zeitschrift

- ☐ Außerdem interessiere ich mich für folgende/n Computer:

(PS: Wir speichern Ihre Daten und verpflichten uns zur Einhaltung des Bundesdatenschutzgesetzes)



Zeitschriften • Bücher
Software • Schulung

Markt & Technik Verlag AG, Buchverlag, Hans-Pinsel-Straße 2,
8013 Haar bei München, Telefon (089) 46 13-0

Markt & Technik Verlag AG
- Unternehmensbereich Buchverlag -
Hans-Pinsel-Straße 2
D-8013 Haar bei München

Das Buch zum aktuellen Thema: **dBase IV deutsch Menüsystem**

»dBase IV deutsch Menüsystem« ist zugleich Einführungs- und Nachschlagewerk für die Arbeit mit dem neuen Menüsystem von dBase, dem sogenannten »Regiezentrum«. dBase IV setzt mit dem Regiezentrum einen neuen Standard und eröffnet eine weitere Dimension der Anwendung: Es können nun alle Möglichkeiten von dBase genutzt werden, ohne daß der Benutzer Kenntnis von der Befehls- und Programmiersprache haben muß. Das Regiezentrum vereinfacht die Arbeit mit dBase, da die Bedienerführung keine ausgedehnten Vorkenntnisse erfordert. Dasselbe gilt für das vorliegende Buch. Einsteiger können sofort mit der Erstellung und Anwendung von Datenbanken beginnen. In dBase Erfahrene können sich rasch mit den menügesteuerten Funktionen und den neuen Arbeitstechniken von dBase IV vertraut machen. Das Buch ist so konzipiert, daß unterschiedliche Benutzer Unterstützung für ihre Arbeit mit dem Programm finden.

Die einzelnen Kapitel sind jeweils einem Arbeitsbereich gewidmet. Es werden dabei folgende Themen behandelt: - Einführung in grundlegende Arbeitstechniken und -hilfen - Datenbank-Dateien anlegen und bearbeiten - Abfragen durchführen und »Sichten« erstellen - Gestaltung von Masken für die Bildschirmeingabe und -ausgabe - Gestaltung von druckfertigen Berichten - Serienbriefe erstellen - Etiketten entwerfen und erstellen - Möglichkeiten der Programmentwicklung.

Lieferbar 1. Quartal 1989,
ca. 350 Seiten
Bestell-Nr. 90738
ISBN 3-89090-738-5
ca. DM 69,- (sFr 63,50/öS 538,-)

In Vorbereitung:

Krieger, Zander & Partner GmbH
dBase IV deutsch Programmgenerator
Bestell-Nr. 90739,
ISBN 3-89090-739-3
ca. DM 69,- (sFr 63,50/öS 538,-)

163/812



Markt & Technik-Produkte erhalten
Sie bei Ihrem Buchhändler, in
Computer-Fachgeschäften oder in den
Fachabteilungen der Warenhäuser

Ausklappbare Menü-Übersicht

Laufwerk

Hole Dokument mit Namen {}

Zwischendurch abspeichern
Weglegen
Arbeitsfläche räumen

Textdatei für DOS erstellen
DOS-Kommandos
Sonderdienste

ENDE Framework

NEU

Konzept

Frame (leer / Text)

Tabellenkalkulation

Datenbank

Breite (# Spalten/Felder): {14}

Länge (# Zeilen/Sätze): {14}

Spalten/Felder (# Einfügen): {1}

Zeilen / Sätze (# Einfügen): {1}

Editieren

Rücknahme

Spalten/Felder: Entfernen
Zeilen / Sätze: Entfernen
-- Fixieren erste Spalte/Zeile

-- Gegen Änderungen schützen
Änderungen zulassen

-- Vorhandenes überschreiben
-- Unsichtbare Zeichen darstellen

Suchen

Vorwärts (aufsteig.) sortieren
Rückwärts (absteig.) sortieren

Suchen nach:

Ersetzen durch: {}

JA Namen berücksichtigen
JA Inhalt berücksichtigen
-- Formeln berücksichtigen

JA Großschreibung ignorieren

Frames

Öffnen alle
Schließen alle
Größe ändern alle
Löschen alle

-- Ausrichten in Spalte
JA Position frei wählbar

-- Ergänzen um Seitennummer
-- Durchnummerieren der Namen
-- Typbezeichnung anfügen
JA Namen anzeigen
-- Rahmen unterdrücken

Text

-- Normalschrift
-- **Fettdruck**
-- Unterstreichen
-- Schrägschrift

-- Ausrichten nach links
-- Verschieben nach rechts
-- Blocksatz
-- Zentriert

Linker Rand {0}
Rechter Rand {65}
Einzug Absatz {0}
Tabulatorsprung {8}

Zahlen

Dezimalstellen: {2}
JA Standardformat
-- Ganze Zahl
-- Festpunktzahl
-- **Währungsbetrag**
-- Unterteilung in tausender
-- Prozent
-- Exponentendarstellung

JA Texte links / Zahlen rechts
-- Linksbündig
-- Rechtsbündig
-- Zentriert

► Optionen für Neuberechnung

Grafik	
Neue Grafik zeichnen Überlagern vorhandene Grafik	
JA Balken nebeneinander -- Abschnitts-Balken -- Kreis -- Linie -- Punkte, markiert -- X-Y	
-- Spalte markiert X-Achse JA Zeile markiert X-Achse	
► Optionen	
X-Achse, Titel:	{ }
Y-Achse, Titel:	{ }
-- Manuelle Skalierung Y-Achse	
Kleinsten Y-Wert:	{ }
Größter Y-Wert:	{ }
Schrittweite der Skala:	{ }
Absetzen Kreissegment: {0}	

Drucken	
Starten Beenden	
-- Pause -- Warten bei neuer Seite	
► Ziel für Ausgabe ► Optionen für Ausdruck	
JA Erster Drucker -- Zweiter Drucker -- Plotter -- Textdatei für DOS	

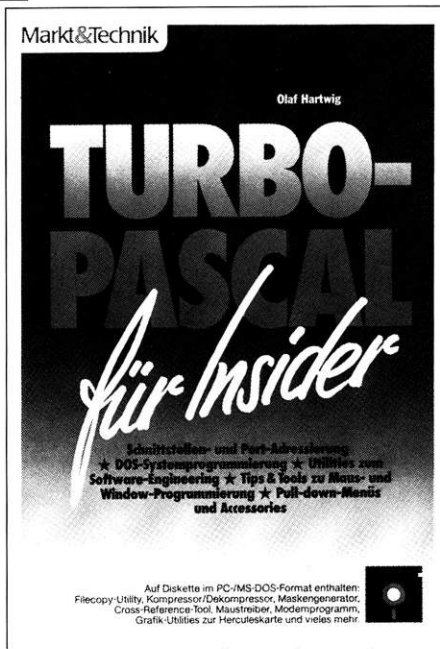
Drucken	
Starten Beenden	
-- Pause -- Warten bei neuer Seite	
► Ziel für Ausgabe ► Optionen für Ausdruck	
Anfangen mit Seite: {1} Ende nach Seite: {9999}	
Kopien, Anzahl: {1}	
-- Ohne geschlossene Frames -- Nur Formeln JA Mit Namen der Frames	

Funktionstasten-Belegung:			
HILFE	F1	F2	FORMEL EDITIEREN
POSITION	F3	F4	GRÖSSE
NEUBERECHNUNG	F5	F6	AUSWAHL
VERLAGERN	F7	F8	KOPIEREN
ZOOM	F9	F0	ZEIGEN

Bücher zu Programmiersprachen

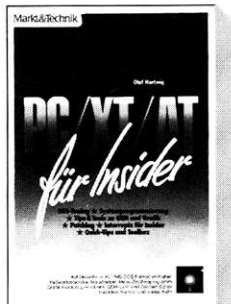


J. Hückstädt
Programmieren mit Turbo-BASIC
1987, 300 Seiten,
inkl. Diskette
Jeder BASIC-Programmierer, der bereits praktische Erfahrung mitbringt, wird mit diesem Buch schnell in Turbo-BASIC eingearbeitet. Sie finden hier nicht nur die allgemein bekannten Programmierertechniken, sondern werden auch in das strukturierte Programmieren eingeführt. Selbst wenn Sie Ihre Programme weiterhin in gewohnter Weise schreiben und mit Turbo-BASIC kompilieren, ist Ihnen diese Lektüre ein wertvoller Begleiter.
Bestell-Nr. 90542
ISBN 3-89090-542-0
DM 59,-
(sFr 54,30/öS 460,20)



O. Hartwig
Turbo-Pascal für Insider
1987, 274 Seiten,
inkl. Diskette
Eine wahre Fundgrube an Insiderwissen für jeden Turbo-Pascal-Anwender! Schwerpunktmäßig werden die Adressierung des Betriebs-

systems, direkte Schnittstellen- und Portadressierung, Utilities zum Software-Engineering, Grafik, Color und Sound, Tips & Tools behandelt.
Bestell-Nr. 90540
ISBN 3-89090-540-4
DM 59,-
(sFr 54,30/öS 460,20)

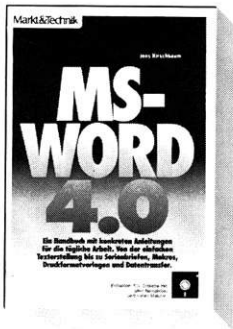


O. Hartwig
PC/XT/AT für Insider
1987, 272 Seiten,
inkl. Diskette
Sind Sie engagierter PC- und AT-Anwender? Interessieren Sie sich für Insiderwissen zu Ihrem System? Dann ist dieses Buch genau das Richtige für Sie! Schwerpunktmäßig werden das DOS-Tuning, die System- und Interruptprogrammierung, die direkte Modifizierung des GEM-Systems, universelle Grafik-Tools und die PC- und AT-Toolbox mit zahlreichen weiteren Utilities behandelt. Mit speziellen Tips zu Schneider-PC und -AT.
Bestell-Nr. 90541
ISBN 3-89090-541-2
DM 59,-
(sFr 54,30/öS 460,20)



Markt & Technik-Produkte erhalten Sie bei Ihrem Buchhändler, in Computer-Fachgeschäften oder in den Fachabteilungen der Warenhäuser.

Bücher ^{zu} Text- verarbeitung

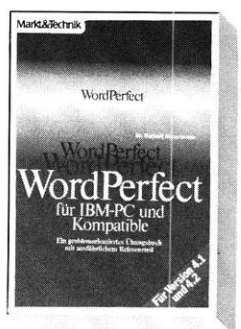


J. Kirschbaum
MS-Word 4.0
1988, 541 Seiten, 2. überarbeitete Auflage, inkl. Diskette
Dieses Handbuch mit verbindlichen Anleitungen für die tägliche Arbeit erleichtert dem Anfänger den Einstieg und dient dem fortgeschrittenen Anwender als ausführliches Nachschlagewerk. Von der einfachen Texterstellung bis zu Serienbriefen, Makros, Druckformatvorlagen und Datentransfer. Ein ausführliches Glossar und ein Anhang mit den wichtigsten Fehlermeldungen sowie Tips und Tricks für die Praxis und viele Übungsbeispiele auf Diskette.
Bestell-Nr. 90664
ISBN 3-89090-664-8
DM 69,-
(sFr 63,50/öS 538,20)



H. Niemeier
Microsoft Word Druckformatvorlagen
1988, 187 Seiten, inkl. Diskette
Dieses Buch bereitet Ihnen einen leichten Weg zur optischen Gestaltung

von Texten wie Brief, Rechnung, Bewerbung, Lebenslauf, bis hin zum druckfertigen Buch.
Bestell-Nr. 90588
ISBN 3-89090-588-9
DM 69,-
(sFr 63,50/öS 538,20)



Dr. N. Hesselmann
WordPerfect für IBM-PC und Kompatible
2. Quartal 1988, ca. 300 Seiten
Dieses Buch fängt dort an, wo die meisten aufhören!
Im ersten Teil erhalten Sie eine fundierte Einführung in das Textverarbeitungsprogramm WordPerfect. Der zweite Teil ist von der Struktur her wie ein Nachschlagewerk aufgebaut. Im dritten Teil sind alle für die Arbeit mit WordPerfect erforderlichen Informationen in tabellarischer Form nach unterschiedlichen Gesichtspunkten geordnet.
Bestell-Nr. 90485
ISBN 3-89090-485-8
DM 59,-
(sFr 54,30/öS 460,20)

Markt&Technik
Zeitschriften · Bücher
Software · Schulung

Markt&Technik-Produkte erhalten Sie bei Ihrem Buchhändler, in Computer-Fachgeschäften oder in den Fachabteilungen der Warenhäuser

Framework

Der Autor:

RUDI KOST, Jahrgang 1949, war lange Zeit Redakteur und Ressortleiter bei Tageszeitungen. Er arbeitet nun als freier Journalist und Publizist – nicht nur im Bereich Textverarbeitung. Mit seinen Rezensionen, Aufsätzen und Buchveröffentlichungen (u. a. »Über George Smiley«, »Was ist los mit Trimmel?«) hat er sich als Film- und Theaterkritiker sowie als Krimi-Spezialist einen Namen gemacht.

Als integriertes Programmpaket mit den Teilen Textverarbeitung, Tabellenkalkulation, Datenbank und Präsentationsgrafik, deckt Framework vielfältige Anwendungsbereiche ab und ist dank seines Funktionsumfangs und seiner Flexibilität auch für den professionellen Einsatz geeignet. Schritt für Schritt wird der Anwender zunächst in den praktischen Umgang mit Framework eingeführt und erfährt,

- was Frames sind und wie man mit ihnen umgeht,
- wie man den Funktionsumfang der einzelnen Programmbereiche nutzt,
- wie das integrierende Prinzip von Framework auszuschöpfen ist und wie man demzufolge zwischen den Programmteilen Daten austauscht und Textverarbeitung, Tabellenkalkulation, Datenbank und Grafik miteinander verbindet.

Anhand von Modellbeispielen aus dem Alltag, die detailliert erläutert werden, bekommt der Anwender Handreichungen,

- wie man eine Adreßdatenbank aufbaut, sortiert, selektiert und sie zur Grundlage einer Serienbrief-Funktion macht,
- wie man mit der Tabellenkalkulation eine »Was-wäre-wenn-Analyse« durchführt und sie bei der Verkaufsplanung einsetzt,
- wie man Tabellen oder Datenbanken auswertet und die Ergebnisse in einen Geschäftsbericht übernimmt,
- wie man eine einfache Einnahmen-Überschuß-Berechnung konzipiert und ausbaut.

Besonderer Nachdruck wird auch auf Makros und auf FRED, die Programmiersprache von Framework, gelegt. Der Benutzer lernt auf leicht faßliche Weise, und wiederum anhand von Modellfällen,

- wie man Makros definiert und sich mit ihnen die Arbeit erleichtert,
- wie man mit FRED eigene Programme schreibt und damit Arbeitsabläufe automatisiert und vereinfacht.

Alle Übungsdateien, Makros und Beispielpprogramme sind auf der beiliegenden Diskette enthalten und so gestaltet, daß sie sofort als Anwendungen für Framework einsetzbar sind, aber auch ohne Mühe variiert und den eigenen Bedürfnissen angepaßt werden können.

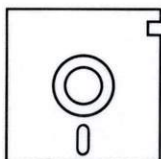
Das Buch wendet sich ebenso an den privaten Anwender wie an die Sekretärin, den Handwerker, den Sachbearbeiter, den Vereinsvorsitzenden oder den Selbständigen.

Hardware-Anforderungen:

IBM PC/XT (oder Kompatible) mit mindestens 384 Kbyte Arbeitsspeicher, zwei Diskettenlaufwerke (360 Kbyte) oder ein Diskettenlaufwerk und Festplatte.

Software-Anforderungen:

Programmpaket Framework, Markt & Technik Verlag AG, Bestell-Nr. 56525



ISBN N 3-89090-500-5



DM 49,-
sFr 45,10
öS 382,20