

Markus Weber

## **Turbo Pascal Tools**

# **Technisch-naturwissenschaftliche Anwendungen**

## **Computergrafische Experimente mit Pascal**

von K.-H. Becker und M. Dörfler

## **Methoden der Numerischen Mathematik**

von W. Böhm, G. Gosse und J. Kahmann

## **Statistische Verfahren**

von J. Bruhn

## **Multivariate Statistik in den Natur- und Verhaltenswissenschaften**

von C.-M. Haf und T. Cheaib

## **Fortgeschrittene Programmiertechniken in TurboPascal**

von E. Hering und K. Scheurer

## **Numerische Mathematik**

von D. Herrmann

## **Wahrscheinlichkeitsrechnung und Statistik**

von D. Herrmann

## **BASIC-Programme zur Numerischen Mathematik**

von J. Kahmann

## **Lineare Optimierung mit BASIC auf dem PC-1500A**

von H. Luther

## **Die endliche Fourier- und Walsh-Transformation mit einer Einführung in die Bildverarbeitung**

von K. Niederdröck

## **Turbo Pascal Tools**

von M. Weber

## **Physikalische Experimente mit dem Mikrocomputer**

von K.-D. Tillmann

Markus Weber

# **Turbo Pascal Tools**

**Mathematische Verfahren und Programmroutinen  
zur Auswertung experimenteller Daten**



**Friedr. Vieweg & Sohn      Braunschweig / Wiesbaden**

CIP-Titelaufnahme der Deutschen Bibliothek

**Weber, Markus:**

Turbo Pascal tools: math. Verfahren u. Programm-  
routinen zur Auswertung experimenteller Daten /

Markus Weber. — Braunschweig; Wiesbaden:

Vieweg, 1987

(Technisch-naturwissenschaftliche  
Anwendungen)

ISBN-13: 978-3-528-04543-2

e-ISBN-13: 978-3-322-86345-4

DOI: 10.1007/978-3-322-86345-4

Das in diesem Buch enthaltene Programm-Material ist mit keiner Verpflichtung oder Garantie irgendeiner Art verbunden. Der Autor und der Verlag übernehmen infolgedessen keine Verantwortung und werden keine daraus folgende oder sonstige Haftung übernehmen, die auf irgendeine Art aus der Benutzung dieses Programm-Materials oder Teilen davon entsteht.

Der Verlag Vieweg ist ein Unternehmen der Verlagsgruppe Bertelsmann.

Alle Rechte vorbehalten

© Friedr. Vieweg & Sohn Verlagsgesellschaft mbH, Braunschweig 1987



Das Werk einschließlich aller seiner Teile ist urheberrechtlich geschützt. Jede Verwertung außerhalb der engen Grenzen des Urheberrechtsgesetzes ist ohne Zustimmung des Verlags unzulässig und strafbar. Das gilt insbesondere für Vervielfältigungen, Übersetzungen, Mikroverfilmungen und die Einspeicherung und Verarbeitung in elektronischen Systemen.

ISBN-13: 978-3-528-04543-2

## Vorwort

Ich habe das vorliegende Buch in der Absicht geschrieben, dem experimentell tätigen Naturwissenschaftler eine Reihe von Routinen und Programmen an die Hand zu geben, die er bei der Aufbereitung von Meßergebnissen auf einem Personal Computer benötigt. In der Kürze war es mir leider nicht möglich, all die Themen anzusprechen, die ich gern behandelt hätte. Ich hoffe, die getroffene Auswahl enthält trotzdem viele interessante Themen.

Ich habe mich bei der Wahl einer geeigneten Programmiersprache für dieses Buch für die Verwendung des TurboPascal Compilers der Firma Borland entschieden. Dieser enthält bereits eine Vielzahl von Befehlen, die über Standard-Pascal hinausgehen und ist andererseits weitgehend zum UCSD-Pascal Standard kompatibel.

Da Pascal von Nikolaus Wirth 1971 ursprünglich als Unterrichtssprache entwickelt wurde, läßt es sich besonders gut zur Demonstration von Programmierprinzipien verwenden. Für den Leser bedeutet die modulare Struktur eine wesentliche Erleichterung, da sich sämtliche Programmschritte fast wie in der theoretischen Formulierung lesen lassen. Gleichzeitig bleibt infolge der guten Strukturierbarkeit eine optimale Übersicht gewahrt.

Die in diesem Band enthaltenen Programme sollen dem Leser, der gerade erst beginnt, sich in die Programmierung von PC's einzuarbeiten, den Einstieg erleichtern und ihn von der Programmierung mehr oder weniger elementarer Routinen befreien. Als Vorbild dienten hier die Programmbibliotheken an den Großrechnern der Universitäten. Der Leser soll aber nicht nur Programme zur Verfügung gestellt bekommen, sondern auch die Möglichkeit diese zu verstehen. Soweit dies möglich ist, wurde deshalb darauf geachtet, die Programme nicht zu speziell auf Turbo Pascal hin zu optimieren und sie auch ausführlich theoretisch zu erläutern. Dabei sollte die Übertragbarkeit auf andere Programmiersprachen gesichert bleiben. Dies war auch ein Grund, warum ich mich für einen Pascal Compiler als Programmiersprache entschieden habe. Die strukturierte Programmierung in Pascal hat sich in den letzten Jahren in fast allen Computerzirkularen als Korrespondenz-Programmiersprache etabliert. Dies ist nicht zuletzt der Einführung von Turbo Pascal zu verdanken, das neben einem günstigen Anschaffungspreis auch noch einen äußerst komfortablen, Wordstar kompatiblen, Editor und eine reichhaltige und offene Programmbibliothek enthält.

Viele Befehle, die Turbo Pascal dem Benutzer zur Verfügung stellt, können von diesem allerdings aus Unkenntnis meist gar nicht genutzt werden. Das heißt nicht, daß sie im Handbuch fehlen würden. Dort werden aber so viele Kenntnisse über Aufbau des Rechners und über das Betriebssystem vorausgesetzt, daß vor allem Anfänger hoffnungslos überfordert sind. Ein Ziel dieses Buches war es daher auch, den Leser mit diesen Kenntnissen zu versorgen, so daß er auch tatsächlich alle zur Verfügung gestellten Befehle ausnutzen kann.

Ein weiteres Anliegen war mir die systematische Darstellung der wichtigsten mathematischen Analysemethoden. Hier sollte in erster Linie die Methode der nichtlinearen Mini-

mierung, das sogenannte "Fitting", ausgiebig behandelt werden. Ich habe festgestellt, daß die meisten Leute, die mit solchen Programmen zur Datenanalyse arbeiten, nicht wissen, was eigentlich im Rechner abläuft und wie die Ergebnisse zu interpretieren sind. Dabei ist es wichtig, gerade darüber Bescheid zu wissen, da alle diese Programme in der Regel nur unter bestimmten Prämissen einwandfrei arbeiten. Wie gut sich ein gemessenes Spektrum auf diese Weise auswerten läßt, hängt meist stark von der Erfahrung des Benutzers ab. Generell ist auf kein gefittetes Ergebnis absolut Verlaß.

Abschließend werde ich noch die elementarsten Simulationsprinzipien auf der Basis der Monte-Carlo-Methode besprechen. Der Leser soll dabei durch die Verwendung nicht-gleichverteilter Zufallszahlen in die Lage versetzt werden, einfache Simulationen selbständig durchzuführen.

Ich möchte mich an dieser Stelle ganz herzlich bei all meinen Freunden und Kollegen bedanken, die zum Gelingen dieses Buches beigetragen haben. Mein besonderer Dank gilt hier Herrn Michael Zelger, der mir über zweieinhalb Jahre, oftmals bis spät in die Nacht, mit Rat und Tat aktiv zur Seite gestanden und als unerbittlicher Diskussionspartner maßgeblich zur Beseitigung der Fehler in den mathematischen Abschnitten des Buches beigetragen hat.

Weiter möchte ich mich ganz herzlich bei Fräulein Barbara Matschke bedanken, die in den Stunden, in denen nichts mehr zu gehen schien, immer für mich da war und auch das gesamte Manuskript kritisch gelesen und mich auf die unverständlichen Passagen aufmerksam gemacht hat.

Meinen Kollegen und Freunden Sigfrido Saibene und Wolfgang Ihra möchte ich sagen, daß ohne ihren unerschütterlichen Glauben an mich dieses Buch wahrscheinlich niemals zustande gekommen wäre. Dasselbe gilt für meine Freunde Ulrich Eschbaumer und Marcus Settles, sowie meine Mutter und Frau Maria Niedermaier. Ich werde Euch das niemals vergessen.

Mein Dank gilt auch Herrn Dr. Tilman Butz und seinen Kollegen vom Institut E15 für nukleare Festkörperphysik der Technischen Universität München, die es mir im Rahmen meiner Arbeit an ihrem Institut ermöglichten, die meisten Programme in diesem Buch zu erstellen.

Zum Schluß noch ein Wort des Dankes nach Wiesbaden an die Mitarbeiter des Vieweg-Verlages, insbesondere an Herrn Wolfgang Dumke und an Frau Gabriele Treiber, die mir in jeder Hinsicht bezüglich meiner vielen Wünsche entgegenkamen.

*Garching/München, im April 1987*

*Markus Weber*

# Inhaltsverzeichnis

|  |           |
|--|-----------|
| <b>Vorwort</b> . . . . .                                       | <b>V</b>  |
| <b>1 Von Pascal bis PASCAL</b> . . . . .                       | <b>1</b>  |
| <b>2 TURBO-PASCAL unter MS-DOS</b> . . . . .                   | <b>7</b>  |
| 2.1 Speicherverwaltung . . . . .                               | 7         |
| 2.2 Die Register der Prozessoren 8086/8088 . . . . .           | 11        |
| 2.3 Das Speicherformat von Variablen . . . . .                 | 13        |
| 2.4 Variablenübergabe an Unterprogramme . . . . .              | 15        |
| 2.5 Interrupts . . . . .                                       | 17        |
| 2.6 Include-Files . . . . .                                    | 18        |
| <b>3 Allgemeine Utility-Routinen</b> . . . . .                 | <b>20</b> |
| 3.1 TYPES.SYS . . . . .  | 20        |
| 3.2 UTILITY.SYS . . . . .                                      | 21        |
| 3.2.1 Time_of_day . . . . .                                    | 21        |
| 3.2.2 Date . . . . .   | 23        |
| 3.2.3 Time_Diff . . . . .                                      | 24        |
| 3.2.4 Hex . . . . .  | 27        |
| 3.2.5 Select_LPT . . . . .                                     | 31        |
| 3.2.6 PlotSmall . . . . .                                      | 33        |
| 3.2.7 Invert_Screen . . . . .                                  | 35        |
| 3.2.8 UpString . . . . .                                       | 37        |
| 3.2.9 Save_Screen und Load_Screen . . . . .                    | 38        |
| 3.3 Convert_Binary_to_Pascal . . . . .                         | 40        |
| <b>4 Stringroutinen</b> . . . . .                              | <b>42</b> |
| 4.1 Edit_String . . . . .                                      | 42        |
| 4.1.1 Tastaturabfrage und Cursorsteuerung . . . . .            | 42        |
| 4.1.2 Cursorbewegung im String . . . . .                       | 44        |
| 4.1.3 Delete und Backspace/Delete . . . . .                    | 45        |
| 4.1.4 Insert- und Overwritemodus . . . . .                     | 46        |
| 4.2 Format_Num_String . . . . .                                | 51        |
| 4.3 Schreiben im Grafikmodus . . . . .                         | 52        |
| <b>5 Variable Speicherverwaltung und DOS-Aufrufe</b> . . . . . | <b>57</b> |
| 5.1 STACK.SYS . . . . .  | 57        |
| 5.1.1 CMP_Less . . . . .                                       | 60        |
| 5.1.2 Die Stackinstallation . . . . .                          | 61        |

|          |  |            |
|----------|--|------------|
| 5.1.2.1  | SetUpStack   | 61         |
| 5.1.2.2  | Dim_Stack  | 62         |
| 5.1.2.3  | Remove_Stack   | 63         |
| 5.1.2.4  | Clear_Stack  | 63         |
| 5.1.3    | PUSH und POP   | 64         |
| 5.2      | DOS-Aufrufe am Beispiel DIR.SYS                      | 65         |
| 5.2.1    | Actual_Drive, Set_Actual_Drive, Bytes_free           | 66         |
| 5.2.2    | Analyse_String                                       | 67         |
| 5.2.3    | Modify   | 70         |
| 5.2.4    | Dir  | 73         |
| <b>6</b> | <b>Elementare Grafikroutinen</b>                     | <b>77</b>  |
| 6.1      | Der Color Graphics Adapter (CGA)                     | 77         |
| 6.2      | GRAPRIM.SYS  | 83         |
| 6.2.1    | Point_Color  | 83         |
| 6.2.2    | Der Bresenham-Algorithmus                            | 84         |
| 6.2.3    | Circle und Ellipse                                   | 92         |
| 6.2.4    | Box  | 96         |
| 6.2.5    | MoveP und RelDraw                                    | 96         |
| 6.3      | Polygonglättung                                      | 98         |
| 6.3.1    | Bernstein- und Bezier-Polynome                       | 98         |
| 6.3.2    | Die Bezier-Interpolation                             | 100        |
| 6.3.3    | BEZIER.SYS   | 104        |
| <b>7</b> | <b>Bitmap-Operationen</b>                            | <b>109</b> |
| 7.1      | WINDOW.SYS   | 109        |
| 7.1.1    | New_Window und Clear_Window_Buffer                   | 111        |
| 7.1.2    | Put_Window   | 113        |
| 7.1.3    | Actual_Window  | 117        |
| 7.1.4    | Slip_Window und Disactivate_Windows                  | 118        |
| 7.1.5    | Redefine_Window                                      | 119        |
| 7.1.6    | Del_Window   | 119        |
| 7.1.7    | Clear_Window   | 120        |
| 7.1.8    | Die Move_Window-Befehle                              | 122        |
| 7.1.9    | Die Window-Zeichenbefehle WPlot und WDraw            | 129        |
| <b>8</b> | <b>Datenrepräsentation</b>                           | <b>131</b> |
| 8.1      | Dreidimensionale Repräsentation gitterförmiger Daten | 131        |
| 8.1.1    | Rotationsmatrizen                                    | 131        |
| 8.1.2    | Zentralprojektionen                                  | 133        |
| 8.1.3    | Datenspeicherung                                     | 134        |
| 8.1.4    | Hidden Lines   | 135        |
| 8.1.5    | 3D-Plot  | 137        |
| 8.2      | Schätzung von Höhenlinien aus experimentellen Daten  | 154        |
| 8.2.1    | Binäre Grenzlinien                                   | 154        |
| 8.2.2    | Der SCHLUMPF   | 155        |



|           |   |            |
|-----------|---|------------|
| 8.2.3     | Höhenlinien und binäre Gitter . . . . .                     | 160        |
| 8.2.4     | Glättung von Höhenlinien . . . . .                          | 161        |
| 8.2.5     | Contour-Plot . . . . .                                      | 163        |
| <b>9</b>  | <b>Mathematische Operationen . . . . .</b>                  | <b>179</b> |
| 9.1       | Die Prozessoren 8086/8088 und 8087 . . . . .                | 179        |
| 9.2       | Maschinenzahlen und Floating-Point-Operationen . . . . .    | 180        |
| 9.2.1     | Die Floating-Point-Darstellung . . . . .                    | 180        |
| 9.2.2     | Fehlerfortpflanzungen . . . . .                             | 183        |
| 9.3       | COMPLEX.SYS . . . . .                                       | 185        |
| 9.3.1     | Die elementaren komplexen Funktionen . . . . .              | 186        |
| 9.3.2     | Multiplikation und Division . . . . .                       | 189        |
| 9.3.3     | Komplexes Quadrat und Potenzfunktion . . . . .              | 191        |
| 9.3.4     | Die Errorfunktion erf . . . . .                             | 193        |
| 9.3.5     | Die Kaiser-Bessel-Funktion . . . . .                        | 200        |
| 9.4       | Das Gaußsche Eliminationsverfahren . . . . .                | 202        |
| 9.4.1     | Aufstellen des Gleichungssystems . . . . .                  | 202        |
| 9.4.2     | Einlesen des Gleichungssystems . . . . .                    | 203        |
| 9.4.3     | Lösung des Gleichungssystems . . . . .                      | 204        |
| 9.5       | Harmonische Analyse – Fast Fouriertransformations . . . . . | 207        |
| 9.5.1     | Fourierreihen . . . . .                                     | 208        |
| 9.5.2     | Diskrete Fouriertransformationen . . . . .                  | 210        |
| 9.5.3     | Die schnelle Fouriertransformation FFT . . . . .            | 214        |
| 9.5.4     | Formulierung des Algorithmus . . . . .                      | 215        |
| 9.6       | FFT.SYS . . . . .   | 219        |
| 9.6.1     | Datenzugriff über variable Pointerfelder . . . . .          | 219        |
| 9.6.2     | Die unbestimmte Variablenübergabe . . . . .                 | 222        |
| 9.6.3     | Die Prozedur FFT . . . . .                                  | 222        |
| 9.6.4     | Datenwichtung . . . . .                                     | 226        |
| <b>10</b> | <b>Nichtlineare Ausgleichsrechnung . . . . .</b>            | <b>228</b> |
| 10.1      | Das lineare Ausgleichsproblem . . . . .                     | 232        |
| 10.2      | Standardminimierungsmethoden . . . . .                      | 235        |
| 10.3      | Die Taylormethode . . . . .                                 | 236        |
| 10.4      | Die Gradientenmethode . . . . .                             | 237        |
| 10.5      | Analyse des Problems . . . . .                              | 238        |
| 10.6      | Theoretische Formulierung des Algorithmus . . . . .         | 239        |
| 10.7      | Der Algorithmus . . . . .                                   | 240        |
| 10.8      | Gewichtetes Fitten . . . . .                                | 245        |
| 10.9      | Fehler- und Korrelationsmatrizen . . . . .                  | 247        |
| 10.10     | Kontrollen . . . . .  | 253        |
| 10.11     | Klassifizierung von Matrizen . . . . .                      | 254        |
| 10.12     | Höhenlinien und Kontrollen . . . . .                        | 255        |
| 10.13     | Freie Parameter . . . . .                                   | 260        |
| 10.14     | Linearfit . . . . .   | 261        |

|                             |                            |            |
|-----------------------------|----------------------------|------------|
| 10.15                       | Das Programm FIT           | 262        |
| 10.15.1                     | FIT.PAR                    | 263        |
| 10.15.2                     | GRAPH.FIT                  | 264        |
| 10.15.3                     | FITM.FIT                   | 267        |
| 10.15.4                     | FIT.PAS                    | 279        |
| <b>11</b>                   | <b>Randomfunktionen</b>    | <b>282</b> |
| 11.1                        | Die Exponentialverteilung  | 285        |
| 11.2                        | Die Lorentzverteilung      | 286        |
| 11.3                        | Die Gaußverteilung         | 290        |
| 11.4                        | Simulation eines Spektrums | 290        |
| <b>Anhang</b>               |                            | <b>295</b> |
| A.                          | SMALL.BIN                  | 295        |
| B.                          | TPLOT.SYS                  | 297        |
| <b>Literaturverzeichnis</b> |                            | <b>299</b> |
| <b>Sachregister</b>         |                            | <b>305</b> |

## 1 Von Pascal bis PASCAL

Es liegt bereits gut 5500 Jahre zurück, daß in Babylon und Ägypten die ersten Zahlensysteme auftauchten. Das besondere Merkmal all dieser Zahlensysteme war, daß sie stets in engem Bezug zu der Anzahl der Finger an der menschlichen Hand standen. So verwendeten die Ägypter bereits ein Zehnersystem, wohingegen die Römer 2500 Jahre später noch nach einem Fünfersystem arbeiteten. Die Ägypter und Babylonier kannten bereits die Zahl Null, die eine unabdingbare Voraussetzung für ein vollständiges Zahlensystem ist.

Es dauerte jedoch noch bis fast ins 8. Jahrhundert nach Christus, bis das dezimale Zahlensystem, von Händlern aus dem indoasiatischen Raum importiert, das Zahlensystem der Römer verdrängte. Erst mit ihm konnte sich die Mathematik schwunghaft weiterentwickeln, da es nun endlich möglich war, in einfacher Weise die vier Grundrechenarten auszuführen. Der entscheidende Vorteil des Dezimalsystems liegt in der Verwendung eines Stellen-systems, in dem jede Stelle mit einer aus 10 möglichen Ziffern besetzt werden kann. Dabei wird die Null als Ziffer und Zahl voll in das System integriert.

Die ersten bildlichen Darstellungen einer mechanischen Rechenhilfe, des Abakus, tauchen auf den Vasen antiker Künstler des 3. bis 4. vorchristlichen Jahrhunderts auf. Allerdings gestalteten sich die Operationen der Multiplikation und der Division auf diesen Geräten noch sehr umständlich.

Es war dann letztendlich die Mathematik, die der Entwicklung der Rechenhilfen im 16. Jahrhundert den entscheidenden Impuls in die richtige Richtung gab. Mit Kopernikus, Galilei und Kepler lebten damals drei Gelehrte, deren Arbeiten umfangreiche Berechnungen, die möglichst schnell durchgeführt werden sollten, erforderten. Der entscheidende Schritt hierzu war die Entdeckung der Potenz- und Logarithmengesetze. Zum ersten Mal erläutert wurde das Rechnen mit Potenzen von dem deutschen Mathematiker Michael Stifel. Für Johannes Kepler schuf der Schweizer Josef Bürgi 1588 eine Logarithmentafel, die er selbst berechnete und für Keplers umfangreiche Berechnungen verwendete. Er weigerte sich jedoch bis ins Jahr 1620 diese Tafeln auch zu veröffentlichen. So bleibt der Ruhm, die erste Logarithmentafel publiziert zu haben, dem englischen Mathematiker Lord John Napier of

Merchiston vorbehalten. Dieser hatte bereits früher ein System von Rechenstäbchen erfunden, das die Ausführung von Multiplikationen erleichterte. Im Jahr 1594 veröffentlichte er ein Logarithmensystem, das später die Bezeichnung natürliches Logarithmensystem erhielt. 1614 folgte ein Buch mit Logarithmen und 1617 stellte er schließlich die erste Logarithmentafel vor.

Der englische Astronom John Briggs schrieb Napiers Logarithmentafel 1624 auf das dem Dezimalsystem entsprechende System zur Basis 10 um. Die Engländer Edmund Gunter und William Oughtred griffen Napiers Idee mit den Rechenstäbchen auf und brachten sie mit den Logarithmentafeln in Verbindung. Daraus entstand das schließlich von Oughtred 1622 vorgestellte Konzept des Rechenschiebers.

Wieder war es Johannes Kepler, der den Anstoß zur Entwicklung der ersten funktionsfähigen Rechen-Maschine gab. Mit seinen mathematischen Forschungen, die umfangreiche und ermüdende Zahlenberechnungen erforderten, traktierte er seine Mitarbeiter. 1623 konnte ihm der Tübinger Professor Wilhelm Schickard die Entwicklung einer zahnradgetriebenen Rechenmaschine mitteilen, die die Grundrechenarten mit sechs Stellen beherrschte. Sie war bereits so weit entwickelt, daß sie den automatischen Zehnerübertrag beherrschte.

In Europa tobte zu diesem Zeitpunkt der Dreißigjährige Krieg, in dessen Wirren Schickards Rechenmaschine zerstört wurde und damit für lange Zeit im Grau der Geschichte verschwand. Erst im 20. Jahrhundert wurden Zeichnungen und Beschreibungen von Schickards Rechenmaschine gefunden. Sie wurde nachgebaut und es konnte bewiesen werden, daß sie tatsächlich funktionierte.

Der Bau der ersten funktionsfähigen Rechenmaschine wurde deshalb lange Zeit dem französischen Physiker und Mathematiker Blaise Pascal zugeschrieben, der sich in Paris außerhalb des vernichtenden Einflußbereiches des Dreißigjährigen Krieges befand. 1642 konnte der damals erst 19jährige der erstaunten Öffentlichkeit seine achtstellige Rechenmaschine präsentieren.

Gottfried Wilhelm von Leibniz, neben Sir Isaac Newton einer der Entdecker der Infinitesimalrechnung, war es schließlich, der eine der wesentlichen Grundlagen für die Entwicklung des digitalen Computers schuf. In den Jahren vor 1673 entwickelte er das Konzept einer Rechenmaschine, die auf der Grundlage von sogenannten Staffelwalzen alle vier Grundrechenarten beherrschen

sollte. Die Fertigungsmethoden seiner Zeit konnten aber nicht die dafür notwendige Präzision gewährleisten, so daß es seiner Maschine ebenso wie den Rechenmaschinen vieler seiner Kollegen erging und sie niemals die vollständige Funktionstüchtigkeit erlangte. Am 15. März 1679 schrieb Leibniz dann die Arbeit, die die Welt verändern sollte. In "De Progressione Dyadica" entwickelt er alle Grundlagen des dualen Zahlensystems, das sich lediglich auf die Zahlen 0 und 1, oder von Leibniz' philosophischem Standpunkt aus gesehen, Sein oder Nicht-Sein, stützte. Er zeigt in dieser Arbeit, daß in diesem einfachsten aller denkbaren Zahlensysteme auch alle Grundrechenarten und Rechenoperationen am elementarsten gelöst werden können. Hiermit war eine der wesentlichsten Grundlagen des Computers geboren.

Die Zeit war jedoch noch nicht reif für Leibniz und seine duale Rechenmaschine. Unkenntnis der Umwelt und mangelhafte Fertigungsmethoden verhinderten den Bau einer solchen Rechenmaschine und es sollte bis ins 20. Jahrhundert dauern, bis diese Idee wieder aufgegriffen wurde. Die Entwicklung ging zunächst einen ganz anderen Weg.

Der Italiener Johannes Polenius und die Deutschen Antonius Braun und Philipp Matthäus Hahn entwickeln in den Jahren zwischen 1709 und 1774 die mechanischen Rechenmaschinen, zum Teil auf der Grundlage von Leibniz' erster Rechenmaschine, bis zur Serienreife weiter. Hahns Schwager Schuster war der erste, der dessen Maschine ab 1789 in größerer Stückzahl herstellte. Ihm folgten in den Jahren 1821 und 1878 der Franzose Charles Xavier Thomas und der Deutsche Arthur Burkardt mit eigenen Entwicklungen, die sie industriell fertigten und vertrieben.

Die nächste grundlegende Entwicklung auf dem Weg zum modernen Computer kam aus Frankreich, genauer gesagt aus Lyon. Dort baute der Mechaniker Falcon 1728 eine Vorrichtung in einen Webstuhl ein, die diesen anhand eines Holzbrettchens, das mit einer Lochkombination versehen war, automatisch steuerte. Dies kann als die Geburt der Lochkarte als Daten- und Programmspeicher angesehen werden. Wieder war es ein Franzose, Joseph-Marie Jaquard, der Falcons Werk fortsetzte und 1805 einen von einem Lochkartenprogramm gesteuerten Webstuhl konstruierte. Somit hielt bereits damals ein Teil des Computers als Mittel zur Automation Einzug in die industrielle Produktion.

Für die Datenverarbeitung wurde die Lochkarte 1886 von dem

Amerikaner Hermann Hollerith entdeckt, dessen elektromechanische Sortier- und Zählmaschine die Auswertung der 11. amerikanischen Volkszählung von 1890 revolutionierte. Der Erfolg war so durchschlagend, daß sich in der Folge alle führenden Industrienationen dazu entschlossen, Holleriths Maschine oder Abwandlungen davon für ihre Volkszählungen zu verwenden.

Bereits im Jahre 1833 hatte der Engländer Charles Babbage einen analytischen Rechenautomaten konstruiert, dessen Verwirklichung jedoch einmal mehr an den fertigungstechnischen Voraussetzungen scheiterte. Seine Konstruktion machte ihn aber zum geistigen Vater der digitalen Rechenmaschine mit Programmsteuerung.

Auf der Grundlage dieses Wissens war es nun nicht mehr schwer, den letzten Schritt zum ersten funktionsfähigen Computer zu tun. 1936 weist der Franzose Valtat auf die Vorteile der Dualzahlen beim Bau von Rechenmaschinen hin und Konrad Zuse beginnt in Berlin, zunächst auf rein mechanischer Basis, mit dem Bau einer programmgesteuerten Rechenmaschine, die er ZUSE Z1 nennt. In den folgenden Jahren entwickelt er diesen Prototyp weiter, wobei er immer mehr mechanische Teile durch elektrische ersetzt. Am 12. Mai 1941 kann er in der ZUSE Z3 die erste funktionsfähige programmgesteuerte digitale Rechanlage vorstellen. Der Zweite Weltkrieg aber verhindert den Gedankenaustausch zwischen den Wissenschaftlern in Deutschland und denen der meisten anderen Industrienationen. So sollte es noch bis zum 7. August 1944 dauern, bis mit dem von Howard Aiken erdachten und gebauten MARK I der erste programmgesteuerte Rechenautomat Amerikas in Betrieb genommen werden konnte.

Ab diesem Zeitpunkt verlief die Entwicklung stürmisch. Bereits 1906 hatte Robert Lieben den Verstärkereffekt der Gitterelektrodenröhre entdeckt. 1919 entwickeln Eccles und Jordan die Flip-Flop-Schaltung in Röhrentechnik und schaffen damit die Grundlage eines vollelektronischen Rechenwerks. 1944 konzipiert John von Neumann den ersten speicherprogrammierten Rechenautomaten EDVAC und wird damit zum Vater des modernen Computers. Zunächst wird jedoch 1945 mit dem von Eckert und Mauchly entwickelten Computer ENIAC die erste vollelektronische Großrechenanlage der Welt in den Vereinigten Staaten fertiggestellt. Erst der 1946 von Maurice V. Wilkes an der Universität von Manchester in England in Röhrentechnik gebaute Rechner EDSAC erfüllt von Neumanns Forde-

rungen nach einer internen Programmspeicherung. Alle bis dahin gebauten Rechenanlagen beruhten auf dem Prinzip eines starren Programms, das extern abgespeichert und von einem Rechenwerk abgearbeitet wurde. Erst mit der flexiblen internen Programmspeicherung war es möglich, auch Verzweigungen und Schleifen in das Programm aufzunehmen und den Rechner auf der Grundlage logischer Entscheidungen zu programmieren.

Die Stunde der zweiten Computergeneration schlug dann am 19. März 1955, als mit TRADIC, auf der Grundlage des von Bardeen, Brattain und Shockley erfundenen Transistors, von den Bell Telephone Laboratories der erste Transistorrechner der Welt vorgestellt wurde. Ihm folgte im Jahre 1957, in der Rechenanlage "2002" der Firma Siemens, der erste voll transistorbestückte Rechner.

Im Jahre 1962 hält die Miniaturisierung Einzug in die Computertechnologie. Mit der Einführung salzkorngroßer Transistoren sinkt der Platzbedarf und steigt die Rechengeschwindigkeit - die Computer der dritten Generation sind geboren. Den nächsten Schritt zu den Computern der vierten Generation ermöglicht 1968 ein winziges Siliziumstück, das bei den Bemühungen der Amerikaner den Mond zu erreichen, als Nebenprodukt aus der Entwicklung abfällt. Mit der Einführung von integrierten Schaltkreisen wird der erste Schritt auf dem Weg zum heutigen Computer getan. Die damit eingeleitete ständige Verkleinerung wurde in den folgenden Jahren ständig weitergetrieben und damit die Geschwindigkeit der Rechner immer weiter erhöht, bis man langsam aber sicher die Grenzen der physikalisch störungsfreien Miniaturisierung erreichte. Diese Grenzen sind vor allem durch thermische und quantenmechanische Effekte gegeben, die es nicht erlauben, bestimmte Abstände zwischen den Leiterbahnen eines Chips zu unterschreiten, da sich diese sonst infolge von Tunnel- und anderen Effekten nicht mehr einwandfrei elektrisch isolieren lassen oder einfach thermisch dissoziieren.

Die Entwicklung geht heute weg von der fortschreitenden Miniaturisierung der Chips und hin zu einer Abänderung des durch John von Neumann eingeführten Konzepts des sequentiellen Computers. Das Handikap fast aller heutigen Computer ist der sogenannte von Neumann'sche Flaschenhals, die CPU oder Central Processing Unit. Sie ist das Nadelöhr, das alle Operationen sequentiell hintereinander passieren müssen. Häufig ist es aber möglich, daß mehrere

Programmteile parallel verarbeitet werden können. Durch Vernetzung vieler handelsüblicher Prozessoren zu einem parallel verarbeitenden Computer ist es möglich, die Rechengeschwindigkeit praktisch beliebig zu steigern. Der im Mai 1986 von der Thinking Mashines Corporation vorgestellte Parallelcomputer CM (Connection Mashine) erreicht mit einer Rechengeschwindigkeit von 7000 MIPS (Million Instructions Per Second) bereits eine höhere Rechengeschwindigkeit als die schnellsten Supercomputer in herkömmlicher Technologie, und das bei einem Preis, der bei 7-20% dieser Rechner liegt.

Die Entwicklung des Computers war immer auch eng mit der Entwicklung der Naturwissenschaften verknüpft. Heute ist diese Beziehung enger denn je. Die modernen Naturwissenschaften, insbesondere Physik und Chemie, sind heute ohne Computer nicht mehr denkbar.



## 2 TURBO PASCAL unter MS-DOS

Bei der Verwendung von Turbo Pascal auf Rechnern der IBM-PC Serie, die mit dem Betriebssystem MS-DOS arbeiten, gibt es eine Reihe von Besonderheiten zu beachten, auf die der Benutzer im Handbuch zu Turbo Pascal nur sehr unzureichend oder gar nicht hingewiesen wird. Hier können nur langjährige Erfahrung oder ein eingehendes Studium der gängigen Literatur weiterhelfen. Andererseits erfordern die meisten Routinen, die sich direkt mit der Systemprogrammierung befassen, eingehende Kenntnisse des Betriebssystems und der Rechnerstruktur. Diese sind verständlicherweise nicht im Turbo Pascal-Handbuch enthalten, da sie dort auch mehr zur Verwirrung als zur Aufhellung beitragen würden. Dies würde nicht störend auffallen, würden durch Turbo Pascal ausreichend viele Befehle zur Verfügung gestellt und eine eigene Systemprogrammierung durch den Benutzer damit überflüssig. Dies ist aber leider nicht der Fall. Die Möglichkeiten des Diskettenzugriffs, der Grafikprogrammierung und der mathematischen Funktionen sind auf ein Minimum beschränkt. Normalerweise wird man damit auf keinen Fall auskommen. Dem Benutzer werden jedoch von Turbo Pascal durch die Verwendung von Include- und in Assembler geschriebenen External- und Inline-Files sowie der Möglichkeit Interrupts aufzurufen genügend Wege geöffnet dieses Manko zu beseitigen.

Wir wollen zunächst einige technischere Dinge besprechen, die dazu dienen sollen, die Bedeutung später häufiger verwendeter Begriffe eindeutig festzulegen.

### 2.1 Speicherverwaltung

Die elementarste Speichereinheit in einem digitalen Computer ist ein **Byte**. Als Byte bezeichnet man einen Verband von 8 **Bits**, die man sich bildlich immer als 8 Glühlämpchen vorstellen kann. Für jedes dieser Bits gibt es zwei Zustandsmöglichkeiten, die wir abkürzend durch 0 (Lämpchen ist ausgeschaltet) und 1 (Lämpchen ist eingeschaltet) bzw., wenn es zur Vermeidung von Mißverständnissen notwendig ist, durch O und L charakterisieren wollen. Dies sind die möglichen Ziffern des von Leibniz erfundenen dualen Zahlensystems, d.h. eines Zahlensystems, das mit nur zwei Ziffern auskommt. Ein Byte ist somit nichts anderes als eine

achtstellige Dualzahl. (Dualzahlen bezeichnet man oft auch nach dem lateinischen bis=zweimal als Binärzahlen.) Da jedes Bit zwei mögliche Ziffern darstellen kann, gibt es also  $2^8=256$  verschiedene Varianten, wie dieses Byte aussehen kann. Sie sind die ersten 256 Zahlen des Dualsystems und können etwa dazu verwendet werden, die Zahlen 0 bis 255 des dezimalen Zahlensystems darzustellen. Sie können aber auch genausogut zur Codierung von Zeichen durch die sogenannten ASCII-Zahlen (American Standard Code for Information Interchange) benutzt werden.

Die Stellen einer Dualzahl werden, wie die einer Dezimalzahl, von rechts nach links durchnummeriert. Es ist wichtig, daß man sich dessen bewußt ist, denn nur so kann man später die Bedeutung der Befehle SHL (Shift Left) und SHR (Shift Right) verstehen. Die Umrechnung vom Dual- ins Dezimalsystem erfolgt nach der Formel

$$(2.1) \quad D = \sum_{i=0}^7 B_i 2^i,$$

wobei D der dezimale Wert und  $B_i$  der Wert (0 oder 1) der i-ten Stelle der Dualzahl ist. Die acht Stellen werden von 0 bis 7 durchnummeriert, da sie gleichzeitig der Exponent der Stelle bezüglich der Basis 2 sind.

Ein **Word**, zu deutsch Wort, ist ein Verband aus zwei Bytes, also insgesamt 16 Bits oder eine 16-stellige Dualzahl. Analog zum Byte lassen sich damit  $2^{16}=65536=64\text{kByte}$  darstellen. Als Kilobyte (griech.:  $\chi\iota\lambda\iota\omicron\iota$  = tausend), kurz **1Kbyte**, bezeichnet man eine Einheit von  $2^{10}=1024$  Bytes, und nicht, wie oftmals fälschlich aufgrund der sonstigen Bedeutung des Zusatzes Kilo angenommen wird, von 1000 Bytes.

Unter einem **Doubleword**, kurz **DWord** oder deutsch Doppelwort, versteht man dann einen Verband von 2 Wörtern, entsprechend 4 Bytes oder 32 Bits.

Ein Wort wird i.a. zur Darstellung einer **Integerzahl**, d.h. einer ganzen Zahl zwischen -32768 und 32767, verwendet. Für die Darstellung beliebiger Zahlen verwendet man in der Regel hingegen sogenannte **Realzahlen**. Mit diesen lassen sich reelle Zahlen zwischen  $10^{-38}$  und  $10^{38}$  (8086-Prozessor) bzw.  $4.19 \cdot 10^{-307}$  und  $1.67 \cdot 10^{308}$  (8087-Coprozessor) darstellen. Auf die Beschränkungen bei den Realzahlen, auch als **Floating-Point-** oder **Gleitkommadarstellung** bezeichnet, werden wir im 9. Kapitel noch zurückkommen.

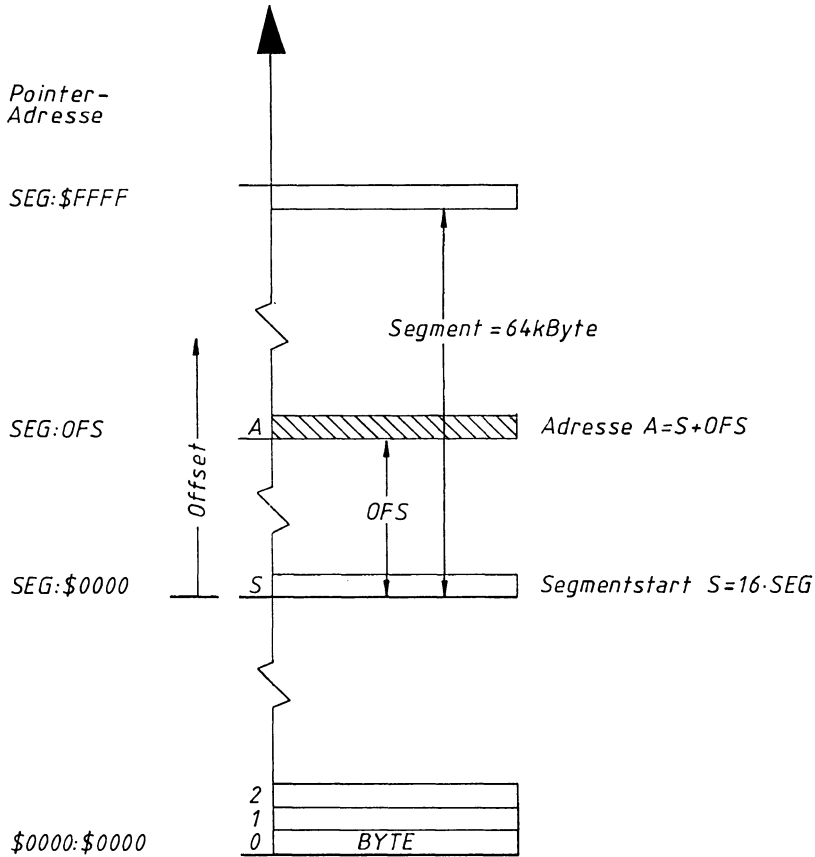
Ein Doppelwort dient i.a. zur Darstellung einer Speicherplatzadresse. Es kann aber auch als sogenannte **Longinteger**-Variable verwendet werden. Diese Variablen erlauben es, Integervariablen zwischen -214748398 und 214748397 darzustellen. Turbo Pascal ist jedoch nicht auf die Verarbeitung von solchen Longintegervariablen vorbereitet, so daß sie lediglich bei Speicherplatzproblemen Verwendung finden. In diesem Fall muß der gespeicherte Wert jeweils in eine Realzahl umgewandelt werden, wenn er für eine Berechnung benötigt wird.

Der Hauptspeicher ist eine Folge von Bytes, die von 0 an durchnummeriert sind. Die Nummer des Bytes ist eine fünfstellige Hexadezimalzahl (vgl. Abschnitt 3.2.4) und wird als die Position oder Adresse des Bytes im Speicher bezeichnet. Eine solche Adresse wird von den Prozessoren in Form von zwei Wörtern, d.h. einem Doppelwort, dargestellt. Das erste dieser beiden Wörter bezeichnet man als **Segment**, das zweite als **Offset** der Adresse. Das Segment SEG bestimmt die Adresse des Bytes in Einheiten von 16 Bytes. Der Offset OFS übernimmt die Feinbestimmung der Adresse, ausgedrückt als Abstand der Adresse in Bytes von der sogenannten Segmentadresse  $16 \cdot \text{SEG}$ . Die Multiplikation der vierstelligen Hexadezimalzahl SEG mit 16 macht aus dieser eine fünfstellige Hexadezimalzahl, wobei die Ziffern einfach um eine Stelle nach links verschoben sind (vgl. Multiplikation mit 10 im Dezimalsystem). Da der Offset insgesamt 65536 Werte annehmen kann, ist durch die Angabe der Segmentadresse ein **64KByte** langer Bereich des Hauptspeichers bestimmt. Diesen bezeichnet man auch als Segment des Hauptspeichers, wovon sich letztendlich auch die Bezeichnung Segment für das erste Wort der Speicherplatzadresse ableitet. Der Offset gibt dann einfach an, das wievielte Byte des Segments gemeint ist.

Die Speicherplatzadresse selbst berechnet sich symbolisch aus Segment und Offset wie folgt:

$$\begin{array}{rcl}
 & \$\text{HHHH} & \text{Segment SEG} \\
 (2.2) \quad + & \$\text{HHHH} & \text{Offset OFS} \\
 \hline
 = & \$\text{HHHHH} & \text{Adresse } 16 \cdot \text{SEG} + \text{OFS}
 \end{array}$$

Da die Adresse eine fünfstellige Hexadezimalzahl ist, sind nur solche Werte für SEG und OFS erlaubt, für die die nach (2.2) berechnete Adresse kleiner oder gleich dem Wert \$FFFFF ist.



**Bild 2-1 Segment und Offset einer Speicherplatzadresse**

Damit ist klar, daß durch die Werte Segment und Offset zwar die Adresse eindeutig bestimmt ist, jedoch nicht umgekehrt. Als Beispiel betrachten wir das Byte, das durch die Angaben Segment=\$B800, Offset=\$0100 spezifiziert ist. Nach (2.2) ist die absolute Adresse des Bytes also

$$\begin{array}{r}
 \$B800 \\
 + \quad \$0100 \\
 \hline
 = \$B8100.
 \end{array}$$

Die Angaben Segment=\$B810, Offset=\$0000 oder Segment=\$A903, Offset=\$0F0D spezifizieren jedoch dasselbe Byte und sind damit gleichberechtigt zu der Angabe \$B800:\$0100 (Kurzschreibweise von Segment=\$B800, Offset=\$0100).