Rolf-Dieter Klein

Floating Point Unit 68881

Für den NDR-Klein-Computer 68020

Der Gleitkomma-Prozessor 68881 realisiert die Gleitkomma-Arithmetik so, wie sie im IEEE-Standard (P754) definiert ist. Darüber hinaus besitzt er einige Zusatzfunktionen (z. B. trigonometrische Funktionen). Er kann als Coprozessor zum 68020 im NDR-Klein-Computer eingesetzt werden.

Ein Coprozessor ergänzt den Befehlssatz eines Hauptprozessors. Er wird nicht wie ein Peripheriebaustein angesprochen, sondern übernimmt bei bestimmten Befehlen im Programm nahtlos deren Bearbeitung als wäre er der Hauptprozessor selbst. Unter Umständen ist sein Verhalten so mit dem des Hauptprozessors verzahnt, daß der den Datentransfer vom und zum Speicher über seine Adreßleitungen unterstützt. Es wird so keine überflüssige Zeit mit Kommunikation verloren und alle Adressierarten des Hauptprozessors stehen zur Verfügung. Die FPU 68881 (von „Floating Point Unit") kann man sowohl als Peripheriebaustein einsetzen, als auch als Coprozessor. Mit dem 68000 zusammen ist sie nur als Peripherie-Baustein zu betreiben, weil der 68000 die notwendigen Signalisierungs-Leitungen nicht besitzt. Mit dem 68020 bildet sie als Coprozessor zusammen ein starkes Gespann.

Die Register

Bild 1 zeigt die Registerstruktur der FPU. Sie besitzt acht Datenregister mit je 80 Bit, die für Gleitkomma-Operationen universell verwendet werden können. Jedes dieser Register kann eine Gleitkommazahl mit 64 Bit Mantisse, 15 Bit Exponenten und einem Vorzeichen-Bit aufnehmen.
Intern besitzt die FPU eine 67 Bit breite arithmetische Logik, so daß Zwischenergebnisse mit Schutzstellen (schützen vor zu grober Rundung) berechnet werden. Ferner gibt es eine 67-Bit-Barrel-Schiebe-Einheit für schnelles Schieben. Barrel-Register sind so konstruiert, daß sie Schiebeoperationen parallel in einer Taktphase ausführen. Normale Schieberegister benötigen mehrere Takte.
Das Steuer-Register FPCR enthält die (programmierbare) Betriebsart der FPU. Dazu gehört zum Beispiel die Art der Rundung. Einmal kann man „to the nearest" runden, dann wird ein nicht als Maschinenzahl existierendes Ergebnis so gerundet, daß die nächstliegende Maschinenzahl entsteht. Man kann aber auch immer die nächst kleinere Maschinenzahl aufsuchen oder die nächst größere – oder die zwischen Null und der nicht-Maschinenzahl nächstliegende Maschinenzahl. Mit demselben Register kann man acht verschiedene Ausnahmebedingungen freigeben, die dann einen Interrupt auslösen können, z. B. Division durch Null, Überlauf, Unterlauf oder inexaktes Ergebnis (wenn z. B. eine Rundung nötig war).

Das Statusregister enthält Informationen über den Stand einer Operation. Da gibt es das Negativ-Flag, für ein Ergebnis kleiner Null, das Zero-Flag, wenn das Ergebnis Null ist, Infinity-Flag, falls durch einen Überlauf ein nicht darstellbarer Zahlenwert entstand (mit dem man aber sinngemäß weiterrechnen kann), und NAN (Not a Number) für Ergebnisse, die nicht mehr eindeutig als Zahlenwert definierbar sind. Zum Beispiel wird auch die Division durch Null angezeigt.

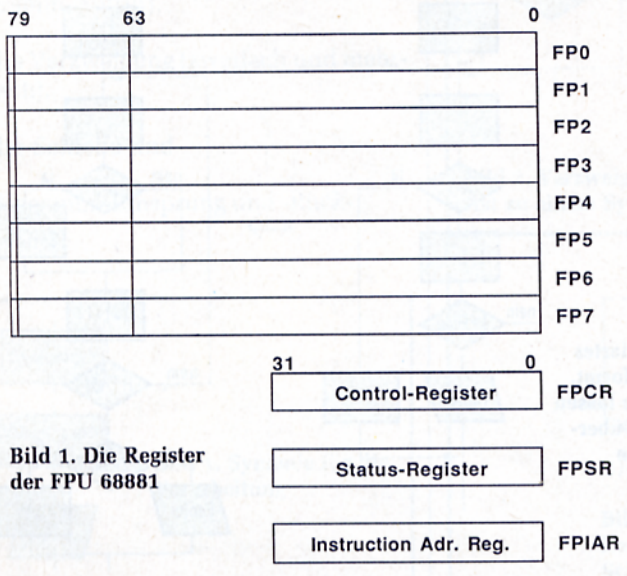

Bild 1. Die Register der FPU 68881

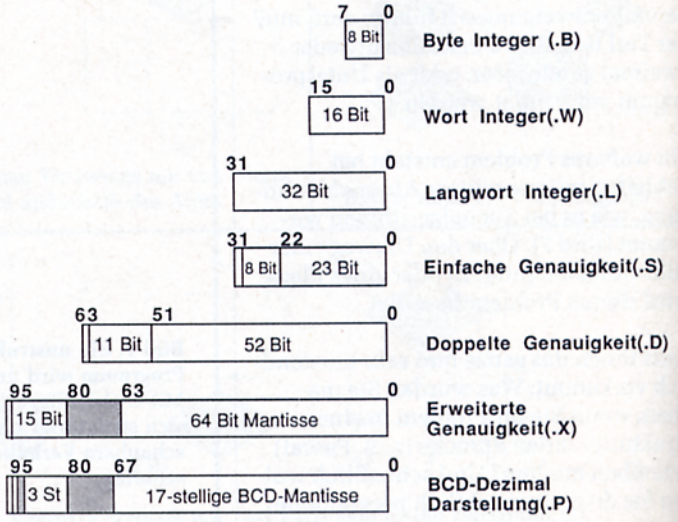

Bild 2. Diese Datenformate kann man verwenden

Prinzip: Parallel-Arbeit

Die FPU 68881 und der Prozessor 68020 können auch parallel arbeiten. Zur Fehlerbehandlung wird daher in einem Register FPIAR die Adresse der zuletzt ausgeführten Operation festgehalten. Wenn dann zum Beispiel eine Ausnahmebedingung zu einem Interrupt führt, kann per Software festgestellt werden, welche Instruktion diesen Fehler verursacht hat. Dabei kann der 68020 vielleicht schon ganz andere Befehle ausführt haben.
Intern arbeitet die FPU normalerweise mit 80 Bit (Extended Precision), wenn man dies nicht anders einstellt. Man kann beim Datentransport in den Hauptspeicher oder in Register der CPU 68020 andere Datenformate angeben. *Bild 2* zeigt eine Übersicht.
Im Befehlscode wird der jeweils verwendete Datenmodus angegeben. In Assembler normalerweise, indem man dem 68881-Befehl – durch einen Punkt getrennt – den Modus als Buchstaben hinten anfügt. Im Byte-Modus kann man Byte-Größen direkt in die FPU laden oder als Ergebnis erzeugen. Sie werden als Integer-Größen interpretiert. Wenn man ein Ergebnis als Byte-Größe ablegt, so wird es zuvor auf den ganzzahligen Teil gerundet. Der Wert darf natürlich nicht mehr als +127 oder -128 betragen, sonst gibt es einen Über- bzw. Unterlauf. Ferner kann man Wort- oder Langwort-Größen verwenden. Es gibt natürlich auch Formate, die speziell auf Gleitkommazahlen zugeschnitten sind. Da ist zunächst die einfache Genauigkeit, bei der eine Gleitkommazahl als 32-Bit-Größe dargestellt wird. Zahlen doppelter Genauigkeit sind mit 64 Bit dargestellt. Will man noch mehr Genauigkeit, so kann man das interne Format verwendet, das allerdings 96 Bit benötigt. Die FPU 68881 kann BCD-Zahlen verarbeiten. Wer schon einmal eine Gleitkomma-Arithmetik geschrieben hat, weiß wie mühsam und zeitaufwendig die Umwandlung vom binären Gleitkommaformat in das BCD-Format ist. Sie braucht man aber immer dann, wenn man Ergebnisse in Zahlenform auf dem Bildschirm ausgeben oder von der Tastatur eingeben will.

Raffinierte Befehle

Mit dem Co-Prozessor-Konzept wird es nicht nur möglich, neue Arithmetik-Befehle einzuführen, sondern auch bedingte Sprünge, deren Bedingung sich direkt auf die FPU-Ergebnisse beziehen. *Bild 3* zeigt die neuen Bedingungen. Dabei findet man neben bekannten Abfragen (wie

Bedingungen

IEEE Non Aware Tests:

EQ	Equal	Gleich
NE	Not Equal	Ungleich
GT	Greater Than	Größer als
NGT	Not Greater Than	Nicht Größer als
GE	Greater Than or Equal	Größer als oder Gleich
NGE	Not (Greater Than or Equal)	Nicht (Größer als oder Gleich)
LT	Less Than	Kleiner als
NLT	Not Less Than	Nicht Kleiner als
LE	Less Than or Equal	Kleiner als oder Gleich
NLE	Not (Less Than or Equal)	Nicht (Kleiner als oder Gleich)
GL	Greater or Less Than	Größer oder Kleiner als
NGL	Not (Greater or Less Than)	Nicht (Größer oder Kleiner als)
GLE	Greater, Less or Equal	Größer, Kleiner oder Gleich
NGLE	Not (Greater, Less or Equal)	Nicht (Größer, Kleiner oder Gleich)

bei den obigen Bedingungen wird das BSUN-Bit (Ausnahme-Bedingung) gesetzt, wenn die NAN (not a number) Bedingung gesetzt ist, ausser bei EQ und NE.

IEEE Aware Tests:

EQ	Equal	Gleich
NE	Not Equal	Ungleich
OGT	Ordered Greater Than	Geordnet Größer als
ULE	Unordered or Less or Equal	Ungeordnet oder Kleiner oder Gleich
OGE	Ordered Greater Than or Equal	Geordnet Größer als oder Gleich
ULT	Unordered or Less Than	Ungeordnet oder Kleiner als
OLT	Ordered Less Than	Geordnet Kleiner als
UGE	Unordered or Greater or Equal	Ungeordnet oder Größer oder Gleich
OLE	Ordered Less Than or Equal	Geordnet Kleiner als oder Gleich
UGT	Unordered or Greater Than	Ungeordnet oder Größer als
OGL	Ordered Greater or Less Than	Geordnet Größer oder Kleiner als
UEQ	Unordered or Equal	Ungeordnet oder Gleich
OR	Ordered	Geordnet
UN	Unordered	Ungeordnet

das BSUN-Bit wird nicht gesetzt

Divers:

F	False	Nicht erfüllt
T	True	Erfüllt
SF	Signalling False	Erfüllt + BSUN-Bit setzen, falls NAN
ST	Signalling True	Nicht erfüllt + BSUN-Bit setzen, falls NAN
SEQ	Signalling Equal	Gleich + BSUN-Bit setzen, falls NAN
SNE	Signallung Not Equal	Ungleich + BSUN-Bit setzen, falls NAN

Bild 3. Bedingungen, die man abfragen kann

Befehlsliste:

FABS.<size> <ea>,FPn FABS.X FPm,FPn FABS.X FPn	Absolut Betrag
FACOS.<size> <ea>,FPn FACOS.X FPm,FPn FACOS.X FPn	Arcus Cosinus
FADD.<size> <ea>,FPn FADD.X FPm,FPn	Addition
FASIN.<size> <ea>,FPn FASIN.X FPm,FPn FASIN.X FPn	Arcus Sinus
FATAN.<size> <ea>,FPn FATAN.X FPm,FPn FATAN.X FPn	Arcus Tangents
FATANH.<size> <ea>,FPn FATANH.X FPm,FPn FATANH.X FPn	Arcus Tangents Hyperbolikus
FBcc.<weite> <marke>	Bedingter Sprung
FCMP.<size> <ea>,FPn FCMP.X FPm,FPn	Vergleich
FCOS.<size> <ea>,FPn FCOS.X FPm,FPn FCOS.X FPn	Cosinus
FCOSH.<size> <ea>,FPn FCOSH.X FPm,FPn FCOSH.X FPn	Cosinus Hyperbolikus

Bild 4. Die umfangreiche Befehlsliste der FPU

FDBcc Dn,<marke>	Prüfen, Dekrementieren und bedingter Sprung
FDIV.<size> <ea>,FPn FDIV.X FPm,FPn	Division
FETOX.<size> <ea>,FPn FETOX.X FPm,FPn FETOX.X FPn	e-Funktion, e hoch x
FETOXM1.<size> <ea>,FPn FETOXM1.X FPm,FPn FETOXM1.X FPn	e-Funktion, (e hoch x) minus 1
FGETEXP.<size> <ea>,FPn FGETEXP.X FPm,FPn FGETEXP.X FPn	Exponent Laden, ohne Bias
FGETMAN.<size> <ea>,FPn FGETMAN.X FPm,FPn FGETMAN.X FPn	Mantisse Laden, Bereich 1.0 .. 2.0
FINT.<size> <ea>,FPn FINT.X FPm,FPn FINT.X FPn	Ganzzahliger (Integer) Anteil
FINTRZ.<size> <ea>,FPn FINTRZ.X FPm,FPn FINTRZ.X FPn	Ganzzahliger (Integer) Anteil gegen Null gerundet (abschneiden)
FLOG10.<size> <ea>,FPn FLOG10.X FPm,FPn FLOG10.X FPn	10er Logarithmus ($\log_{10}$)
FLOG2.<size> <ea>,FPn FLOG2.X FPm,FPn FLOG2.X FPn	2er Logarithmus ($\log_2$)
FLOGN.<size> <ea>,FPn FLOGN.X FPm,FPn FLOGN.X FPn	natürlicher Logarithmus ($\log_e$)
FLOGNP1.<size> <ea>,FPn FLOGNP1.X FPm,FPn FLOGNP1.X FPn	natürlicher Logarithmus plus 1 ($\log_e(x+1)$)
FMOD.<size> <ea>,FPn FMOD.X FPm,FPn	Modulo Rest
FMOVE.<size> <ea>,FPn FMOVE.<size> FPm,<ea>	Transport
FMOVE.P FPm,<ea>,{Dn} FMOVE.P FPm,<ea>,{#k}	Transport im BCD-Mode, mit Genauigkeitsangabe
FMOVE.L <ea>,FPcr FMOVE.L FPcr,<ea>	Transport von System-Control-Registern
FMOVECR.X #index,FPn index:	Constante aus FPUROM laden $00 pi $0B $\text{Log}_{10}(2)$ $0C e $0D $\text{Log}_2(e)$ $0E $\text{Log}_{10}(e)$ $0F 0.0 $30 ln(2) $31 ln(10) $32 10 hoch 0 $33 10 hoch 1 $34 10 hoch 2 $35 10 hoch 4 $36 10 hoch 8 ... ... $3F 10 hoch 4096
FMOVEM.X <liste>,<ea> FMOVEM.X Dn,<ea> FMOVEM.X <ea>,<liste> FMOVEM.X <ea>,Dn	Transport mehrerer Register
FMUL.<size> <ea>,FPn FMUL.X FPm,FPn	Multiplikation
FNEG.<size> <ea>,FPn FNEG.X FPm,FPn FNEG.X FPn	Negation
FNOP	keine Operation
FREM.<size> <ea>,FPn FREM.X FPm,FPn	IEEE Rest Bilden
FPRESTORE <ea>	Internen FPU-Zustand zurückladen
FSAVE <ea>	Internen FPU-Zustand sichern
FSCALE.<size> <ea>,FPn FSCALE.X FPm,FPn	Exponent skalieren (Multiplikation mit 2 hoch n)
FScc.B <ea>	True ($FF) oder False (0) setzen
FSGLDIV.<size> <ea>,FPn FSGLDIV.X FPm,FPn	Division mit einfacher Genauigkeit (Ergebnis)
FSGLMUL.<size> <ea>,FPn FSGLMUL.X FPm,FPn	Multiplikation mit einfacher Genauigkeit (Ergebnis)
FSIN.<size> <ea>,FPn FSIN.X FPm,FPn FSIN.X FPn	Sinus
FSINCOS.<size> <ea>,FPs:FPc FSINCOS.X FPm,FPs:FPc	Sinus und Cosinus gleichzeitig
FSINH.<size> <ea>,FPn FSINH.X FPm,FPn FSINH.X FPn	Sinus Hyperbolikus
FSQRT.<size> <ea>,FPn FSQRT.X FPm,FPn FSQRT.X FPn	Quadratwurzel
FSUB.<size> <ea>,FPn FSUB.X FPm,FPn	Subtraktion
FTAN.<size> <ea>,FPn FTAN.X FPm,FPn FTAN.X FPn	Tangents
FTANH.<size> <ea>,FPn FTANH.X FPm,FPn FTANH.X FPn	Tangents Hyperbolikus
FTENTOX.<size> <ea>,FPn FTENTOX.X FPm,FPn FTENTOX.X FPn	10 hoch x
FTRAPcc FTRAPcc.W #data FTRAPcc.L #data	Bedingter Trap
FTST.<size> <ea> FTST.X FPm	Test eines Operanden
FTWOTOX.<size> <ea>,FPn FTWOTOX.X FPm,FPn FTWOTOX.X FPn	2 hoch x

```
Rolf-D.Klein 68020/68881 Assembler 5.0 (C) 1985, Seite 1
229C00                  * BENCHMARK-PROGRAMM
229C00                  *
010000                   ORG $10000
010000                  START:
010000  4EB9 00218A18    JSR @CI         * START BEI TASTENDRUCK
010006  223C 00030D40    MOVE.L #200000,D1  * GESAMT 1 000 000 MAL
01000C  F23C 4C00        FMOVE.P #1.000002,FP0
010010  00000001
010014  00000200
010018  00000000
01001C  F23C 4C80        FMOVE.P #1.0,FP1
010020  00000001
010024  00000000
010028  00000000
01002C                  LP:
01002C  F200 00A2        FADD FP0,FP1    * HIER BEFEHLE 5 MAL ABLEGEN.
010030  F200 00A2        FADD FP0,FP1
010034  F200 00A2        FADD FP0,FP1
010038  F200 00A2        FADD FP0,FP1
01003C  F200 00A2        FADD FP0,FP1
010040  5381            SUBQ.L #1,D1
010042  66E8            BNE.S LP
010044  4E75            RTS
010046
010046                   END
228B16  Ende-Symboltabelle
```

Bild 5. Ein Einfach-Benchmark-Programm

```
Rolf-D.Klein 68020/68881 Assembler 5.0 (C) 1985, Seite 1

229C00                  ******************************
229C00                  * FPU DEMO PROGRAMM          *
229C00                  * ROLF-DIETER KLEIN, 860529  *
229C00                  * V 1.0                      *
229C00                  ******************************
229C00
010000                   ORG $10000     * START DES HAUPTPROGRAMMS
010000
010000                  START:
010000  4EB9 0021C3EE    JSR @CLRSCREEN  * BILDSCHIRM LOESCHEN
010006  4EB9 0021B530    JSR @CUROFF     * OHNE CURSOR
01000C  4240             CLR D0
01000E  4241             CLR D1     * BILDSEITENWECHSEL AUS
010010  4EB9 00219FD6    JSR @SETFLIP
010016  4240             CLR D0
010018  4241             CLR D1     * BILDSEITE 0
01001A  4EB9 00218E24    JSR @NEWPAGE
010020                  *
010020  4241             CLR D1     * XPUNKT = 0
010022                  LOOP:
010022                  *
010022  F201 5000        FMOVE.W D1,FP0    * UMRECHNEN IN X-WERT DER FUNKTION
010026  F23C 5420        FDIV.D #512,FP0   * NORMIEREN
01002A  40800000
01002E  00000000
010032  F239 4823        FMUL.X XMAX,FP0   * MAXIMALER WERT IN X-RICHTUNG
010036  000100AC
01003A  6100 0052        BSR FUNKTION      * BERECHNEN DER FUNKTION
01003E  F239 4820        FDIV.X YMAX,FP0   * NORMIEREN
010042  000100B8
010046  F23C 5423        FMUL.D #127,FP0   * AUSLENKUNG IN PIXEL
01004A  405FC000
01004E  00000000
010052  F23C 5422        FADD.D #128,FP0   * BILDMITTE Y = 0
010056  40600000
01005A  00000000
01005E  F202 7000        FMOVE.W FP0,D2    * IN INTEGER, FUER ZEICHNEN
010062  4EB9 00218EFA    JSR @MOVETO       * STARTPOSITION
010068  343C 0080        MOVE #128,D2      * SENKRECHTE LINIE ZEICHNEN
01006C  4EB9 00218F64    JSR @DRAWTO       * ZUR BILDMITTE
010072                  *
010072  0641 0001        ADD #1,D1  * XPUNKT := XPUNKT + 1
010076  0C41 0200        CMP #512,D1
01007A  6D00 FFA6        BLT LOOP   * SOLANGE KLEINER, AUSGEBEN
01007E                  *
01007E                  WA:
01007E  4EB9 00218A18    JSR @CI    * WARTEN BIS M GEDRUECKT
010084  0C00 004D        CMP.B #'M',D0
010088  6600 FFF4        BNE WA     * PROGRAMMENDE
01008C  4E75             RTS
01008E
01008E                  * FUNKTION, BEISPIEL GEDAEMPFTE SCHWINGUNG
01008E                  FUNKTION:       * X IN FP0, ERGEBNIS F(X) IN FP0 ANSCHLIESSEND
01008E  F200 0080        FMOVE FP0,FP1   * MERKEN , HIER BEISPIEL F(X) = SIN(X) * E ^ ( -K * X)
010092  F200 048E        FSIN FP1
010096  F23C 5423        FMUL.D #-0.04,FP0  * -K * X
01009A  BFA47AE1
01009E  47AE147B
0100A2  F200 0010        FETOX FP0       * E HOCH (-K*X)
0100A6  F200 0423        FMUL FP1,FP0    * ERGEBNIS IN FP0
0100AA  4E75             RTS
0100AC
0100AC  40050000        XMAX: DC.X 100.0  * 0..XMAX IN X-RICHTUNG
0100B0  C8000000
0100B4  00000000
0100B8  3FFF0000        YMAX: DC.X 1.0    * -YMAX .. YMAX AMPLITUDE
0100BC  80000000
0100C0  00000000
0100C4
0100C4                   END

228CD8  Ende-Symboltabelle
```

Bild 6. Das Programm stellt eine Funktion grafisch dar

verwendete Testprogramm, in das hier beispielsweise die Instruktion FADD zum Test eingesetzt ist. Die leere Schleife benötigt zur Ausführung 0.64 Sekunden. Es werden immer 200000 Schleifendurchläufe ausgeführt und – um den Overhead klein zu halten – wird der zu testende Befehl fünfmal hingeschrieben. Die gestoppte Zeit kann man dann also durch 1000000 teilen und erhält dann die Ausführungszeit für eine FPU-Instruktion. Die Zeit wurde für einen Betriebstakt von 12 MHz für CPU und FPU gemessen.
Ergebnisse: FSUB4.1 µs, FADD 4.1 µs, FMUL 5.6 µs,FDIV 8.2 µs, FSIN 32.6 µs, FEXP 32.2 µs.
Bild 6 zeigt ein praktisches Anwendungsbeispiel. Aufgabe ist es, eine Funktion graphisch auf dem Bildschirm

EQ) auch neue (wie OGE usw.). Diese zusätzlichen Bedingungsabfragen berücksichtigen alle IEEE-Empfehlungen für die Gleitkomma-Arithmetik. So kann die FPU auch mit Zahlen umgehen, die zu groß geworden sind, oder bei denen sich Fehler aufgrund von Rundungsoperationen eingeschlichen haben.
Bild 4 zeigt die umfangreiche Befehlsliste der FPU. Neben den vier Grundrechenarten findet man dort auch Befehle für trigonometrische Operationen,in denen sogar die hyperbolischen Funktionen enthalten sind.
Wer sich im Detail für diese Befehle interessiert, dem sei das Handbuch zur FPU 68881 empfohlen, das viele hundert Seiten dick ist und von Motorola geliefert wird.

Hohe Geschwindigkeit

Die Rechengeschwindigkeit der FPU kann sich sehen lassen. Ein paar Benchmarks sollen das zeigen. *Bild 5* zeigt das

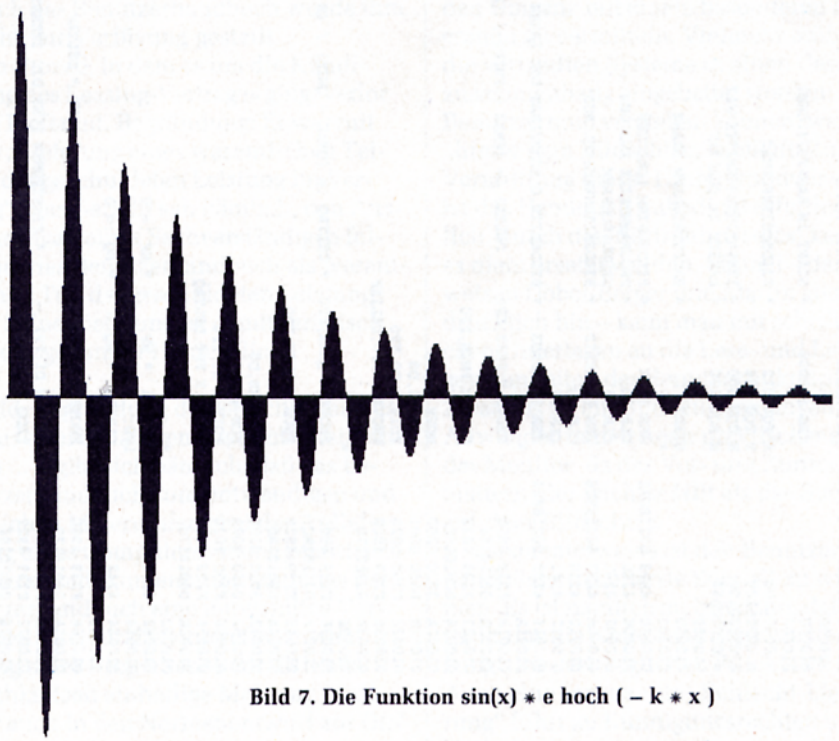

Bild 7. Die Funktion sin(x) * e hoch (– k * x)

```
Rolf-D.Klein 68020/68881 Assembler 5.0 (C) 1985, Seite 1
229C00
010000                    ORG $10000      * START IM HAUPTSPEICHER
010000
= 00000200                A EQU 512       * BILDSCHIRMFORMAT
= 00000100                B EQU 256       *
= 0000012C                K EQU 300       * ITERATIONSSCHRITTE MAX
= 00000064                M EQU 100       * RADIUS IM QUADRAT
010000
010000                    START:          * HAUPTPROGRAMM START
010000                    *
010000  4EB9 00218DB8      JSR @CLR        * BILDSCHIRM LOESCHEN
010006  4240               CLR D0          * PROGRAMMTEILE AUS DEM
010008  4241               CLR D1          * GRUNDPROGRAMM
01000A  4EB9 00219FD6      JSR @SETFLIP    * BILDSEITENWECHSEL AUS
010010  4240               CLR D0
010012  4241               CLR D1
010014  4EB9 00218E24      JSR @NEWPAGE    * BILDSEITE 0 SCHREIBEN UND LESEN
01001A                    *
01001A  F239 4800          FMOVE.X PMAX,FP0   * DELTAP := (PMAX - PMIN) / (A - 1)
01001E  00010194
010022  F239 4828          FSUB.X PMIN,FP0
010026  00010188
01002A  303C 0200          MOVE #A,D0
01002E  0440 0001          SUB #1,D0
010032  F200 5020          FDIV.W D0,FP0
010036  F239 6800          FMOVE.X FP0,DELTAP
01003A  00010170
01003E                    *
01003E  F239 4800          FMOVE.X QMAX,FP0  * DELTAQ := (QMAX - QMIN) / (B - 1)
010042  000101AC
010046  F239 4828          FSUB.X QMIN,FP0
01004A  000101A0
01004E  303C 0100          MOVE #B,D0
010052  0440 0001          SUB #1,D0
010056  F200 5020          FDIV.W D0,FP0
01005A  F239 6800          FMOVE.X FP0,DELTAQ
01005E  0001017C
010062                    *
010062  4241               CLR D1  * NP      * NP := 0
010064                    LP0:
010064  4242               CLR D2  * NQ      * NQ := 0
010066                    LP1:
010066  F239 4900          FMOVE.X DELTAP,FP2   * P0 := PMIN + NP * DELTAP
01006A  00010170
01006E  F201 5123          FMUL.W D1,FP2        * STARTPUNKT FUER DIE ITERATION
010072  F239 4922          FADD.X PMIN,FP2      * P0 IN FP2 MERKEN
010076  00010188
01007A                    *
01007A  F239 4980          FMOVE.X DELTAQ,FP3   * Q0 := QMIN + NQ * DELTAQ
01007E  0001017C
010082  F202 51A3          FMUL.W D2,FP3
010086  F239 49A2          FADD.X QMIN,FP3      * Q0 IN FP3 MERKEN
01008A  000101A0
01008E                    *
01008E  F23C 4600          FMOVE.S #0.0,FP4     * X0 := 0 , X IN FP4
010092  00000000
010096  F23C 4680          FMOVE.S #0.0,FP5     * Y0 := 0 , Y IN FP5
01009A  00000000
01009E  4243               CLR D3               * KK := 0  ITERATIONSZAEHLER
0100A0                    *
0100A0                    LP2:                    * ITERATIONSSCHLEIFE
0100A0  F200 1300          FMOVE FP4,FP6          * XNEXT := X * X  - Y * Y + P
0100A4  F200 1B23          FMUL FP6,FP6           * X IN FP4 , QUADRAT BILDEN
0100A8  F200 1400          FMOVE FP5,FP0          * Y IN FP5
0100AC  F200 0023          FMUL FP0,FP0           * QUADRAT BILDEN
0100B0  F200 0328          FSUB FP0,FP6           * SUBTRAKTION
0100B4  F200 0B22          FADD FP2,FP6           * P IN FP2X, ERGEBNIS XNEU IN FP6
0100B8                    *
0100B8  F200 1380          FMOVE FP4,FP7          * YNEXT := 2 * X * Y + Q
0100BC  F200 17A3          FMUL FP5,FP7           *
0100C0  F23C 57A3          FMUL.D #2.0,FP7        *
0100C4  40000000
0100C8  00000000
0100CC  F200 0FA2          FADD FP3,FP7           * ERGEBNIS VON YNEXT IN FP7
0100D0                    *
0100D0  F200 1A00          FMOVE FP6,FP4          * X := XNEU
0100D4  F200 1E80          FMOVE FP7,FP5          * Y := YNEU
0100D8                    *
0100D8  0643 0001          ADD #1,D3              * KK := KK + 1, ITERATIONSZAEHLER
0100DC  F200 1B23          FMUL FP6,FP6           * RADIUS PRUEFEN
0100E0  F200 1FA3          FMUL FP7,FP7           * X * X + Y * Y < R * R
0100E4  F200 1F22          FADD FP7,FP6           * FP6 ENTHAELT ZWISCHENERGEBNIS
0100E8                    * X
0100E8  F23C 5338 0064     FCMP.W #M,FP6          * M IST R * R
0100EE  F293 000E          FBGE L4                * GROESSER, DANN ITERATIONS BEENDET
0100F2  0C43 012C          CMP #K,D3              * AUCH BEENDET, WENN MAX K ERREICHT
0100F6  6700 0006          BEQ L4
0100FA  6000 FFA4          BRA LP2                * SONST WEITER ITERIEREN
0100FE                    L4:
0100FE  0243 0001          AND #1,D3           * UMRECHNEN AUF SCHWARZ/WEISS
010102  6700 0012          BEQ L5              * PUNKT AUSGEBEN, WENN BIT 0=1
010106  4EB9 00218EFA      JSR @MOVETO         * POSITIONIEREN D1=X, D2=Y
01010C  103C 0080          MOVE.B #$80,D0      *
010110  4EB9 00218D7E      JSR @CMD            * DORT PUNKT SETZEN
010116                    L5:
010116  0642 0001          ADD #1,D2           * NEUES Y WAEHLEN
01011A  0C42 0100          CMP #B,D2           * KLEINER ALS MAX, DANN ZURUECK
01011E  6F00 FF46          BLE LP1
010122  0641 0001          ADD #1,D1           * NEUES X WAEHLEN
010126  0C41 0200          CMP #A,D1           * KLEINER ALS MAX, DANN ZURUECK
01012A  6F00 FF38          BLE LP0
01012E                    WA:
01012E  4EB9 00218A18      JSR @CI             * BILD STEHEN LASSEN UND AUF
010134  0C00 004D          CMP.B #'M',D0       * TASTE "M" WARTEN.
010138  6600 FFF4          BNE WA              * PROGRAMMENDE
01013C  4E75               RTS
01013E                    AUS:                    * TESTAUSGABE, WENN MAN
01013E  48E7 6000          MOVEM.L D1-D2,-(A7)     * FP0 AUF DEM BILDSCHIRM SEHEN
010142  41F9 000101B8      LEA BUFFER,A0           * MOECHTE.
010148  303C FFEF          MOVE #-17,D0
01014C  4EB9 0022183A      JSR @PRINTFP0
010152  41F9 000101B8      LEA BUFFER,A0
010158  303C 0022          MOVE #$22,D0
01015C  323C 000A          MOVE #10,D1
010160  343C 0064          MOVE #100,D2
010164  4EB9 0021939E      JSR @WRITE
01016A  4CDF 0006          MOVEM.L (A7)+,D1-D2
01016E  4E75               RTS
010170
010170                    * VARIABLE
010170  00000000          DELTAP: DC.X 0
010174  00000000
010178  00000000
01017C  00000000          DELTAQ: DC.X 0
010180  00000000
010184  00000000
010188
010188                    * PARAMETER FUER AUSSCHNITT, HIER SEEPFERDCHENTAL,
010188                    * SIEHE "SCHOENHEIT IM CHAOS", VON MAPART, BREMEN
010188  BFFE0000          PMIN: DC.X -0.74591       * -2.25  DAS GESAMTE APFELMAENNCHEN
01018C  BEF3F52F
010190  C2656ABE
010194  BFFE0000          PMAX: DC.X -0.74448       * 0.75   WIRD DARGESTELLT, WENN
010198  BE963DC4
01019C  86AD2DCB
0101A0  3FFB0000          QMIN: DC.X 0.11196        * -1.5   MAN DIE ZAHLEN IM KOMMENTAR
0101A4  E54B48D3
0101A8  AE685DB7
0101AC  3FFB0000          QMAX: DC.X 0.11339        * 1.5    ABLEGT.
0101B0  E839042D
0101B4  8C2A454E
0101B8
0101B8                    BUFFER: DS.B 80
010208
010208                    END

228C5A  Ende-Symboltabelle
```

Bild 8. Programm zur Mandelbrotmenge

Bild 9. Ein Ausschnitt aus dem Seepferdchental der Mandelbrotmenge

darzustellen. Hinter der Marke FUNKTION steht die Formel zur Berechnung der Funktion. Der Eingangsparameter wird dazu in Register FP0 übergeben und der Wert der Funktion steht anschließend wieder in diesem Register. In der Speicherzelle XMAX wird die Größe für die X-Achse eingetragen. Als Startwert wurde hier fest der Wert 0 verwendet. In YMAX steht der Funktionswert, der als Maximum im Intervall 0..XMAX verwendet wird (ein geschicktes Programm kann das auch automatisieren).
Das Hauptprogramm berechnet für jeden x-Wert den dazugehörigen Funktionswert und gibt ihn als senkrechte Linie aus. Die x-Achse verläuft dabei in der Bildschirmmitte, so daß man auch negative Werte darstellen kann.
Bild 7 zeigt das Ergebnis auf dem Bildschirm. Für die Berechnung und Ausgabe auf dem Schirm wurden 260 ms benötigt, die Berechnung selbst dauert nur 100 ms. Man könnte die Kurve also schon bewegt darstellen, denn immerhin sind etwa 4 Bilder pro Sekunde möglich.
Bild 8 zeigt ein anderes Beispiel. Bei der Berechnung der Mandelbrotmenge, die man immer wieder in der Literatur findet, sind sehr viele Rechenschritte nötig. Das Programm ist hier in der Lage, *Bild 9* in ca. 16 Minuten aufzubauen. Ohne FPU würde man dazu ungefähr einen halben Tag benötigen. Dargestellt ist ein stark vergrößerter Ausschnitt der Mandelbrotmenge. Hier könnte man übrigens auch mit Integer-Arithmetik nicht mehr viel ausrichten.

Benutzerhandbuch: Juristisch unverzichtbar

Schlechte oder nicht vorhandene Handbücher – wer mit gekaufter Software arbeiten muß, kann oft ein Lied davon singen. Daß man als Kunde in diesem Fall aber nicht alles hinnehmen muß, zeigt eine Entscheidung des Landgerichts Mannheim vom 8.10.84 (AZ 24 0 62/83), in dem die juristische Bedeutung des Handbuches deutlich wird.
Im entschiedenen Fall hatte ein Fensterbauer beim Lieferanten einen Bürocomputer mit Programmen für Fensterbau, Lohn-, Gehalts- und Finanzbuchhaltung bestellt. Daß die Auslieferung über eine Leasing-Gesellschaft erfolgte, ist hier ohne Bedeutung. Er unterzeichnete eine Übernahmebestätigung, in der er ausdrücklich bestätigte, daß er die darin bezeichneten Anlageteile sowie „Standardsoftware" bereits am 18.6.1980 „fabrikneu und in einwandfrei funktionsfähigem Zustand" übernommen hatte. Im Laufe des Prozesses wurde es unstreitig, daß er einen Teil der vereinbarten Lieferung – ein Programm zur Unterstützung von Kalkulation und Fertigung von Holzfenstern – nicht erhalten hatte.
Auch das Benutzerhandbuch wurde ihm nicht zur Verfügung gestellt.
Den Prozeß, in dem es um die Kündigung des Leasing-Vertrages ging, verlor der Lieferant. Begründung: Er war mit der „Lieferung eines wesentlichen Teiles der geschuldeten Leistung im Verzug". Das Gericht sah nämlich, den Ausführungen eines Sachverständigen folgend, das Benutzerhandbuch als wesentlichen Teil der geschuldeten Leistung an. Seine Lieferung ist damit juristisch eine sogenannte Hauptpflicht.

In der Begründung des Gerichts heißt es: „Aus dem Gutachten des Sachverständigen ... ergibt sich nämlich, daß das Benutzerhandbuch zum notwendigen und üblichen Lieferumfang bei der Anlieferung eines Computersystems gehört. Als Programmdokumentation stellt das Benutzerhandbuch eine notwendige Voraussetzung für die sinnvolle Anwendung eines Programmes dar. Diese in Fachkreisen einhellige Meinung ist daher auch in den Normenentwurf zur DIN 66.230 eingegangen.
Das Benutzerhandbuch kann durch eine Schulung der Benutzer nicht vollständig ersetzt werden. Vielmehr ist nach den überzeugenden Ausführungen des Sachverständigen davon auszugehen, daß die Schulung der Benutzer sowie die schriftliche Dokumentation – das Benutzerhandbuch – zusammengehören und sich wechselseitig ergänzen".

Diese Ausführungen rundet das Gericht damit ab, daß seiner Ansicht nach (und hier folgte es wieder dem Sachverständigen) die Schulung des Benutzers schon aus Zeitmangel nicht auf alle denkbaren Sonderfälle eingehen könne. Diese müßten daher schriftlich dokumentiert werden. Eine Einweisung des Benutzers in alle denkbaren Fälle über den Bildschirm sei unpraktisch.

Damit steht das Gericht im Einklang mit der herrschenden Literaturmeinung. So hat etwa Zahrnt in einer Anmerkung diesem Urteil ausdrücklich zugestimmt (Betriebs-Berater 1985, S. 145). Er berichtet aus Fällen aus seiner Praxis, in denen ein Anwender mit seinem Bürocomputer insbesondere deshalb nicht zurecht kam, weil ihm die Bedienungsanweisung fehlte.
Auch Döbel-Berger und Martin (Zeitschrift für Datenverarbeitung, Online, Heft 10/84 ab S. 88) halten das Handbuch für unverzichtbar. Es reicht ihrer Meinung nach nicht aus, lediglich spezielle Bedienungsanleitungen zur Handhabung des Systems zu geben oder nur den Umgang mit einem besonderen Programm zu vermitteln. Vielmehr müßten die verkauften Systeme in einen Gesamtzusammenhang eingebettet werden: Der Benutzer muß verstehen können, was „hinter dem Bildschirm" abläuft. Zur Vermittlung dieses Hintergrundwissens ist das Handbuch unbedingt erforderlich, durch den Bildschirm selbst kann es nicht ersetzt werden. Einmal – darauf weisen Döbel-Berger und Martin ausdrücklich hin – kann man mit schriftlichen Unterlagen an die Lese- und Lerngewohnheiten der Benutzer anknüpfen. Zweitens ist ein Buch für den ungeübten Anwender immer noch praktischer als der Monitor, da das Vor- und Zurückblättern hier leichter fällt als das Suchen auf dem Bildschirm.
Konsequenterweise ist den Benutzerhandbüchern äußerste Sorgfalt zu widmen. Es ist nämlich zu beachten, daß nicht nur die Nichtlieferung eine Verletzung einer vertraglichen Pflicht ist: Auch eine sogenannte „Schlecht-Lieferung" führt zu Kündigungsrechten und ähnlichem. *Hermann Kahlen*