```
2820 data16,d4,68,8d,0f,d4,68,8d
2830 data0e,d4,ac,ee,07,c8,8c,12
2840 datad4,68,8d,01,d0,68,8d,00
2850 datad0,a9,09,8d,15,d0,ba,e4
2860 data02,f0,3d,68,8d,08,d4,68
2870 data8d,07,d4,ac,ee,07,c8,8c
2880 data0b,d4,68,8d,03,d0,68,8d
2890 data02,d0,a9,0b,8d,15,d0,ba
2900 datae4,02,f0,1c,68,8d,01,d4
2910 data68,8d,00,d4,ac,ee,07,c8
2920 data8c,04,d4,68,8d,05,d0,68
2930 data8d,04,d0,a9,0f,8d,15,d0
2940 data4c,00,33,a2,a6,86,fc,a2
2950 data30,86,fd,ad,ee,07,29,04
2960 dataf0,09,a9,10,8d,ee,07,85
2970 data02,10,2f,ad,ee,07,29,70
2980 datad0,0d,a9,d8,85,fc,a9,14
2990 data8d,ee,07,a0,0a,10,2b,ad
3000 dataee,07,0a,8d,ee,07,85,02
3010 datac9,40,f0,04,a9,00,f0,02
3020 dataa9,0a,a2,27,9d,58,da,ca
3030 data10,fa,a5,fc,18,69,0a,85
3040 datafc,a5,02,0a,85,02,90,f2
3050 dataa0,0a,b1,fc,99,97,06,88
3060 datad0,f8,4c,bd,32,a2,0d,bd
3070 datae8,06,29,7f,9d,e8,06,ca
3080 data10,f5,ae,ef,07,e0,0f,d0
3090 data03,20,ab,35,e0,7f,d0,05
3100 data20,ab,35,a2,ff,8a,18,69
3110 data10,8d,ef,07,8d,18,d4,0a
3120 data85,02,a2,06,86,fd,a2,e8
3130 data86,fc,a2,03,a5,02,0a,85
3140 data02,90,0b,a0,03,b1,fc,09
3150 data80,91,fc,88,10,f7,a5,fc
3160 data18,69,05,85,fc,ca,d0,e4
3170 data4c,bd,32,ad,f2,07,49,07
3180 data8d,f2,07,8d,17,d4,a0,6e
3190 datab9,f8,da,49,0e,99,f8,da
3200 data88,10,f5,60,4d,41,52,54
3210 data49,4e,20,41,48,4c,42,4f
3220 data52,4e,20,33,34,31,38,20
3230 data55,53,4c,41,52
4000 ende
ready.
```

Listing »Synthesizer« (Schluß)

```
10 *save-routine fur c-64 synthesizer
20 *bitte im direktmodus eingeben,
30 *also ohne zeilennummern
40 *
100 poke 781,geraetenummer (1 bzw 8)
110 sys 65466
120 nm$="synthesizer-obj"
130 poke183,len(nm$)
140 poke781,681and255
150 poke782,681/256
160 forc=1to15:poke680+c,asc(mid$(nm$,c)
):next
170 poke250,12032and255
180 poke251,12032/256        *startadresse
190 poke780,250
200 poke781,13790and255
210 poke782,13790/256        *endadresse
220 sys65496                 *save
ready.
```

Listing »Save-Routine« für »Synthesizer«

Mathematical Basic

Fortsetzung von Seite 50

Eine gute Nachricht für alle Freunde des VC 20: Unser »Listing des Monats« macht mit über 50 neuen Befehlen das Programmieren zum Vergnügen.

Leider gibt es aber auch eine schlechte Nachricht. Sie müssen gut und gerne 150 DATA-Zeilen eintippen. Lassen Sie sich aber davon nicht entmutigen. Der Aufwand lohnt sich ganz bestimmt. Das Programm hilft Ihnen bei der wohl unvermeidlichen Suche nach Tippfehlern durch das blockweise Bilden von Prüfsummen.

Befehle des Mathematical Basic

':
erlaubt eine REM-Anweisung innerhalb einer Zeile ohne sofortige Programmfortsetzung ab folgender Zeile.

BEEP h, l:
gibt einen Ton variabler Höhe (h) und Länge (l) aus. h ist im Bereich von 0 bis 126 und l von 0 bis 255 definiert. Beträgt l=0, so wird kein Ton ausgegeben.

CATALOG:
bringt das Inhaltsverzeichnis der Diskette auf den Bildschirm. Dieser Befehl ist nur für das Gerät mit der Nummer 8 definiert. Wird während der Ausgabe die STOP-Taste betätigt, so wird die Ausgabe sofort abgebrochen. Wird die Leertaste gedrückt, so wird die Ausgabe nur unterbrochen und kann mit einer weiteren Betätigung dieser Taste fortgesetzt werden.

COLOR
c,h,r: setzt die Farben für Cursor (c), Hintergrund (h) und Rahmen (r). r und c erstrecken sich von 0 bis 7 und h von 0 bis 15. Die einzelnen Farben bezüglich der Nummercodes können dem VC 20-Handbuch entnommen werden.

CLS:
löscht den Bildschirmspeicher.

DEFUSRn TO x:
definiert einen der neun möglichen USR-Vektoren. Die einzelnen USR-Funktionen werden durch das Zeichen n unterschieden. n kann dabei A, B, C, D, E, F, G oder H sein. Die neunte USR-Funktion ist die, welche über den Vektor in den Adressen 1 und 2 angesprungen wird. Soll dieser Vektor definiert werden, so muß n weggelassen werden. Die Variable x steht für den Vektor selbst, also der Adresse, ab der die USR-Funktion starten soll. Beispiel: DEFUSRA TO 30000. Die USR-Funktion USRA erhält die Einsprungadresse 30000.

DEGREE:
stellt die Routinen für trigonometrische Funktionen auf Normalgrad (0 bis 360) ein.

DELETE a-b:
löscht Zeilen eines Basic-Programms von Zeile a bis Zeile b. Für a und b sind nur Konstanten erlaubt.

DIRECTORY:
siehe CATALOG

DLOAD pn$:
lädt ein Programm mit dem Namen pn$ vom Floppy-Laufwerk mit der Nummer 8. Außer dem Namen sind keine weiteren Parameter erlaubt.

DO:
erlaubt zusammen mit UNTIL eine Schleife. Der zwischen diesen beiden Schlüsselwörtern liegende Programmteil wird so oft wiederholt, bis der Ausdruck, der UNTIL folgt, nicht mehr auf logisch 0 ist. Dazu ein Beispiel: DO GET a$: UNTIL a$=" ". Das Programm verläßt die Schleife nicht eher, bis die Leertaste gedrückt wird. DO-UNTIL-Schleifen können acht Ebenen

tief geschachtelt werden. Eine neunte DO-Anweisung würde die Meldung »out of memory« zur Folge haben. Sollte UNTIL einmal ohne einen vorangegangenen DO-Befehl aufgerufen werden, so gibt der VC 20 »until without do« aus.

DSAVE pn$:
sichert ein Programm mit dem Namen pn$ auf dem Floppy-Laufwerk mit der Nummer 8. Außer dem Namen sind keine weiteren Parameter zulässig. Sollte dem Befehl DSAVE, DLOAD oder DVERIFY kein Parameter folgen, so wird automatisch »*« als Parameter gesetzt.

DVERIFY pn$:
vergleicht das Programm mit dem Namen pn$ im Speicher mit dem gleichen auf der Diskette. Als Parameter ist nur der Name erlaubt.

EXECUTE a$:
wandelt den String a$ in Interpretercode und führt die darin enthaltenen Anweisungen aus. Beispiel: a$="a=22":EXECUTE a$. Der Variablen a wird der Wert 22 zugewiesen. Der Befehl darf nicht im Direkt-Modus gegeben werden und der Befehlsstring (a$) darf eine Länge von 88 Zeichen nicht überschreiten.

GRAD:
stellt die trigonometrischen Routinen auf Neugrad (0 bis 400).

IF:
Die IF...THEN-Anweisung ist in ihren Aufbaumöglichkeiten erweitert worden. Es ist nicht mehr zwingend erforderlich, den THEN-Dummy zu schreiben. Beispiel: IF A=4 THEN PRINT B$ läßt sich auch als IF A=4 PRINT B$ eingeben.

INITIALIZE:
initialisiert die Floppy mit der Nummer 8.

LOCATE z,s:
setzt den Cursor in Zeile z und Spalte s. z liegt im Bereich von 1 bis 23 und s von 1 bis 22. Folgende Variationsmöglichkeiten sind gegeben: LOCATE z definiert nur eine neue Zeilenposition. LOCATE ,s setzt den Cursor innerhalb einer Zeile nur an eine neue Spaltenposition.

LPRINT:
verhält sich wie PRINT. Die Zeichen werden aber nicht auf den Bildschirm, sondern auf den Drucker (Nummer 4) ausgegeben. LPRINT und PPRINT sollten nur bei angeschlossenen Geräten benutzt werden, da sonst der IEC-Bus blockiert wird.

RADIAN:
stellt die trigonometrischen Routinen auf Bogenmaß ein.

RENUMBER z,s:
reorganisiert die Zeilennummern eines Basic-Programms. z ist dabei die Startzeile und s die Schrittweite. z und s dürfen nur Konstanten sein. Sollten dem Befehl keine Parameter folgen, so gilt z = 100 und s = 10.

RESTORE TO z:
positioniert den DATA-Pointer auf die Zeile z. Das Wort TO kann dabei wegfallen. RESTORE allein setzt den Pointer wie gewohnt auf den Programmstart.

RETURN TO z:
ermöglicht es, aus einem Unterprogramm in eine bestimmte Zeile z zurückzukehren.

RUN "name":
hat die gleiche Funktion wie DLOAD. Zusätzlich wird aber noch der String RUN +CHR$(13) in den Tastaturpuffer geschrieben. Das hat zur Folge, daß das gerade geladene Programm sofort gestartet wird.

PPRINT:
verhält sich wie PRINT. Die Zeichen werden aber, anstatt auf den Bildschirm, auf den Plotter (Nummer 6) ausgegeben.

QUIT:
führt ein Total-Reset aus. Mathematical Basic wird dabei gelöscht.

UNITL a: siehe DO.

Die Funktionen des Mathematical Basic

! a:
ist ein Äquivalent zur CHR$(a)-Funktion. Hierbei sind aber keine Klammern nötig. a ist nur als Konstante erlaubt.

&:
ist das »Hexadezimal-Vorzeichen« für Hex-Konstanten. Beispiel: &a02b ist das Äquivalent für 41003. Die Anzahl der Hex-Ziffern ist beliebig, es wird maximal eine 16-Bit-Zahl errechnet.

←:
entspricht dem Ausdruck CHR$(13) und kann auch genauso gehandhabt werden.

[a]:
hat die gleiche Wirkung wie die ABS-Funktion. Unterscheidet sich aber durch die bessere Selbstdokumentation.

ACS(a):
berechnet abhängig von der jeweiligen Einstellung durch RADIAN, DEGREE oder GRAD den Arcus-Cosinus von a. Die hier angesprochene Abhängigkeit gilt für alle trigonometrischen Funktionen.

ACT(a):
bestimmt den Arcus-Cotangens von a.

ASN(a):
ergibt den Arcus-Sinus von a.

CHR$(z,I):
ist eine Variante von CHR$(z). z entspricht dabei dem Zeichen des normalerweise nur 1 Byte langen Strings. Mit I läßt sich nun aber zusätzlich auch die Länge variieren. I ist dabei von 0 bis 255 definiert.

COT(a):
berechnet den Cotangens von a.

CRSCOL:
holt die momentane Spaltenposition des Cursors.

CRSLIN:
holt die momentane Zeilenposition des Cursors.

CVF(a$):
wandelt den fünf Zeichen langen String a$ in eine Fließkommazahl um.

CVI(a$):
wandelt den zwei Zeichen langen String a$ in eine Integerzahl um.

DEC(a$):
berechnet aus dem 4-Byte-String a$, der sich aus Hex-Ziffern aufbaut, eine dezimale 16-Bit-Zahl.

DIV(a,b,):
ist gleichwertig mit dem Ausdruck INT(a/b).

FRC(a):
holt die Nachkommazahl von a.

FUNCTION(a$):
wandelt den maximal 88 Zeichen langen String a$ in Interpretercode und berechnet ihn. Beispiel: y=FUNCTION("5+2"). y wird der Wert 7 zugewiesen. Diese Funktion darf nicht im Direktmodus stehen.

HEX$(a):
wandelt die 16-Bit-Zahl a in einen 4-Byte-Hex. String.

INSTR(a$,b$):
testet, an welcher Stelle a$ sich in b$ befindet. Ist a$ nicht in b$ enthalten, so ist das Ergebnis 0, ansonsten entspricht es der Position von a$ in b$. a$ und b$ müssen mindestens 1 Zeichen lang sein und a$ darf nicht länger als b$ sein. INSTR darf nicht im Direktmodus stehen.

LGD (a):
berechnet den dekadischen Logarithmus von a.

LGU(a,b):
bestimmt den Logarithmus von a zur Basis b.
MKF$(a):
wandelt die Zahl a in einen fünf Zeichen langen String.
MKI$(a):
wandelt die Integerzahl a in einen zwei Zeichen langen String.
MOD(a,b):
bestimmt den ganzzahligen Rest aus der Division von a durch b.
RANDOM:
entspricht dem Ausdruck RND(1).
RANDOM(a,b):
entspricht dem Ausdruck RND(1)*b+a.
TIME$:
ist eine modifizierte Form von TI$. Der Unterschied liegt darin, daß bei TIME$ noch zwei Trennzeichen eingefügt werden.
USRn:
entspricht im Prinzip der USR(a)-Funktion. Unterschiede liegen darin, daß n (was der Kennung A, B, C, D, E, F, G oder H entspricht) folgen muß, aber keine »Klammer-auf-Klammer-zu«-Sequenz. Diese Routine (Aufruf durch JSR$cef1) muß zusätzlich an USR-Routinen, die für die Standard-USR-Funktion ausgelegt sind, angehängt werden. Der Vorteil bei den USRn-Funktionen liegt nun darin, daß man den USR-String mit mehreren Parametern übergeben kann. Die dazu nötigen Routinen sind:

cefa	Test auf Klammer auf und nächstes Zeichen holen.
cd9e	Ausdruck holen und String- und Integer-Flag setzen.
cd8d	Test auf numerischen Ausdruck.
cd8f	Test auf String.
cef7	Test auf Klammer zu und nächstes Zeichen holen.
cefd	Test auf Komma und nächstes Zeichen holen.

Damit sind alle neuen Befehle und Funktionen vorgestellt. Bleibt nur noch, Ihnen viel Spaß beim Programmieren mit »Mathematical Basic« zu wünschen. (Wolfgang W. Wirth/ev)

```
100 rem-----------------------------------
-----------------------------------
101 rem" * * *       M A T H E M A T I C
 A L   B A S I C   V1.02        * * * "
102 rem-----------------------------------
-----------------------------------
103 rem" createt in october 1984 by W.Wi
rth,Th. Heuss Ring 20,6556 Woellstein "
104 rem-----------------------------------
-----------------------------------
105 rem
106 rem
107 rem
108 clr:if peek(55)+peek(56)*256>29211 t
hen poke 55,27:poke56,114:clr
109 poke 36879,14:print chr$(147);chr$(1
3);chr$(14);chr$(5);spc(17);chr$(34)
110 dim cs(14)
111 rem
112 print"   Ladeprogramm fur"
113 print
114 print" ";chr$(18);" Math. Basic  V1.
02 "
115 print
116 print" Das Programm wird"
117 print
118 print" jetzt abgelegt."
119 print
120 print
121 rem
122 rem--------------------------- hilfs
routine ablegen -----------------------
123 rem
124 poke 0,76
125 poke 1,12
126 poke 2,2
127 rem
128 for i=512 to 543
129 :  read j
130 :  cs=cs+j
131 :  poke i,j
132 next
133 rem
134 if cs=2635 then 148
135 rem
136 print
137 print
138 print" Datafehler im"
139 print
140 print" Hilfsrout.-Block !":end
141 rem
142 rem--------------------------- hilfs
routinenblock -------------------------
143 rem
144 data136,177,34,201,58,144,2,233,8,23
3
145 data47,96,32,130,215,32,,2,133,102,3
2,
146 data2,10,10,10,10,5,102,76,148,215
147 rem
148 rem--------------------------- haupt
programm ablegen ----------------------
149 rem
150 for i=0 to 3556
151 :  print i;chr$(145)
152 :  read j$:l=usr(j$)
153 :  cs(i/240)=cs(i/240)+l:poke 29211+
i,l
154 next
155 rem
156 rem--------------------------- fehle
rerkennung ----------------------------
157 rem
158 for i=0 to 14
159 :  read cs:if cs(i)=cs then 163
160 :  print" Fehler im Block von"
161 :  print" Zeile";171+i*10;"bis";180+
i*10:print
162 :  print:ef=1
163 next
164 rem
165 if ef then end
166 sys 30414
167 rem
168 rem--------------------------- mathe
matical basic v1.02 -------------------
169 rem
170 data48,d2,48,d2,48,d2,48,d2,48,d2,48
,d2,48,d2,48,d2,20,f1,ce,20,82,d7,c9,04
171 datad0,62,88,b1,22,a2,0f,dd,73,73,f0
,05,ca,10,f8,30,53,8a,ea,ea,ea,ea,a2,04
172 data6a,66,62,66,63,ca,d0,f8,88,10,e0
,a2,90,38,4c,49,dc,20,f1,ce,a5,14,48,a5
173 data15,48,20,f7,d7,a9,04,20,7d,d4,a8
,a5,14,20,81,72,a5,15,20,81,72,68,85,15
174 data68,85,14,4c,fb,d6,48,29,0f,20,8c
,72,68,4a,4a,4a,4a,aa,bd,73,73,88,91,62
175 data60,4c,58,d6,4c,48,d2,20,a6,d3,20
,fa,ce,20,9e,cd,20,a3,d6,c9,00,f0,ed,48
```

Listing »Mathematical Basic«