Unterbrechen Sie mich bitte!

Im Gegensatz zum Bereich zwischenmenschlicher Beziehungen, wo jemanden zu unterbrechen als plumpe Unhöflichkeit eingestuft wird, ist dies einem Computer gegenüber nicht nur ein beliebtes, sondern sogar erwünschtes Verfahren effektvoller Programmgestaltung. Für die meisten Anwender aber sind solche, »Interrupt« genannte, Methoden — leider — mehr oder weniger »Böhmische Dörfer«.

Anhand der vielseitigen Hardware des Commodore 64 wollen wir nun einmal den Schleier des Geheimnisses ein wenig lüften, Ihnen mit einem Demonstrationsprogramm einige Anwendungsbeispiele zeigen und Sie auf Ihrem Bildschirm kein blaues, aber ein buntes Wunder erleben lassen.

Wenn Sie Ihren C 64 einschalten, und der Cursor blinkt, dann haben Sie bereits einen Interrupt erzeugt. Es handelt sich dabei um eine gezielte Programmunterbrechung, die auf ein bestimmtes Ereignis fixiert ist, so zum Beispiel den Inhalt einer Speicherstelle. Tritt der erwünschte Zustand ein, so räumt die Hardware diesem Vorgang Priorität vor allen anderen Aufgaben ein. Gleich was der Computer im Augenblick macht, er wird unverzüglich seine Tätigkeit einstellen und ein spezielles Unterprogramm abarbeiten, das ihm zuvor als Interruptroutine deklariert wurde. Im Betriebssystem-ROM befindet sie sich von Adresse $EA31 (dezimal 59953) bis $ECB8 (60600). Hier wird etwa 60mal in jeder Sekunde geprüft, ob eine — und falls ja, welche — Taste gedrückt wurde; hier wird das Blinken des Cursors erzeugt und der Motor der Datasette ein- oder ausgeschaltet. Wenn der Interrupt beendet ist, setzt der Computer die Bearbeitung des laufenden Programms exakt an der Stelle fort, an der er sich vor Eintritt des Interrupts gerade befand. Gewöhnlich merkt man von solchen Intermezzi nichts, weil der Prozessor eine ungeheure Geschwindigkeit entwickelt, deren Arbeitstakte sich bereits im Bereich von Mikrosekunden (millionstel Sekunden) bewegen.

Noch eine andere Einrichtung der von Ihnen benutzten Geräte ist mit extremer Schnelligkeit ausgestattet: Der Elektronenstrahl, der vor Ihren Augen 25mal in jeder Sekunde ein neues Bild auf die Mattscheibe zeichnet. Nach diesen grundsätzlichen Überlegungen wollen wir aus solchen Voraussetzungen einen Wettstreit der Systeme entwickeln und den Computer gegen den flotten Strahl antreten lassen. Unglaublich, werden Sie jetzt wahrscheinlich sagen, aber warten Sie ab! Möglich ist das nämlich, weil der für die Bildorganisation zuständige Video-Interface-Controller (VIC) stets genau darüber »im Bild« ist, welche Rasterzeile des Monitorbildes augenblicklich geschrieben wird. Das ist die Bedingung dafür, daß auf dem Schirm auch ein Bild im geordneten Zusammenhang wiedergegeben wird. Der VIC, dessen Register unter den Adressen 53248 ($D000) bis 53294 ($D02E) erreichbar sind, führt in der Speicherzelle 53266 ($D012) Buch über die jeweilige Zeilennummer. Darüber war schon einmal zu lesen, daß es unsinnig wäre, dort etwas hineinzuschreiben, weil man den Elektronenstrahl der Bildröhre gar nicht steuern könne. Und das ist nur zum Teil richtig, denn steuern können wir den Strahl nicht, wohl aber können wir ihn steuern lassen.

Dafür ist besagtes Register — wie übrigens viele andere auch — mit einer Doppelfunktion ausgerüstet. Abhängig von der jeweiligen Zugriffsart, Lesen oder Schreiben (PEEK oder POKE), erreicht man unter derselben Hausnummer verschiedene Adressaten. Wird das Register gelesen, so erfährt man die gerade bearbeitete Rasterzeile, wird es beschrieben, bleibt der übermittelte Wert gespeichert und dient dem internen Vergleich, ob er mit der aktuellen Zeile identisch ist. Wenn dieser Fall eintritt, so reagiert die Hardware selbständig darauf und erzeugt einen Interrupt — falls ein solcher vorgesehen war. Zuvor muß dem Computer nämlich noch mitgeteilt werden, daß dies ein Interruptauslöser sein soll. Die Anmeldestelle ist die Adresse 53274 ($D01A), das Interrupt-Masken-Register.

Diese Speicherstelle korrespondiert ständig mit ihrem Nachbarn 53273 ($D019), dem Interrupt-Request-Register. Ist die vorgewählte Rasterzeile erreicht, signalisiert 53266 dieses Ereignis durch Kippen des Bits 0 im Request-Register: Es nimmt den Wert 1 an. Ist Bit 0 auch in der Maske gesetzt, wird der Interrupt-Pin des VIC aktiv und löst die Unterbrechung aus. Im Demo-Programm schreiben wir zu diesem Zweck eine 1 in die Maske, und fortan gilt für uns die Gesetzmäßigkeit, daß Bildhintergrund und -rand unifarben dargestellt werden, nicht mehr. Wir ändern nämlich ab einer beliebigen Rasterzeile die Bildfarben, um sie einige Zeilen weiter abermals umzuschalten. Da das schneller vor sich geht, als das träge menschliche Auge es registrieren kann, resultiert daraus kein wirres Flackern, sondern ein konstantes mehrfarbiges Bild, das im Extremfall (siehe Beispiel »Regenbogen«) sogar alle 16 möglichen Farben des Randes und Hintergrunds gleichzeitig wiederzugeben in der Lage ist.

Noch aber funktioniert der neue Interrupt nicht, denn der C 64 weiß noch nichts von unserem Programmsegment, mit dem wir den Farbwechsel vornehmen wollen. Dazu müssen wir ihm die Adresse der Routine im Interrupt-Vektor ab Speicherstelle 788 ($0314) hinterlegen: Lowbyte in 788, Highbyte in 789. Aber auch jetzt wird das Ergebnis immer noch nicht unseren Erwartungen entsprechen, da uns laufend der immer noch aktivierte Systeminterrupt in die Quere kommt und nach der Vektorenänderung ebenfalls in der neuen Routine unkontrolliert arbeitet. Dieser Interrupt stammt aus einer ganz anderen Quelle, vom Complex-Interface-Adapter (CIA), auf den wir gleich noch zu sprechen kommen. Softwaremäßig können wir ihn durch Setzen der Interruptflagge (SEI) nicht unterbinden, weil damit auch der ebenfalls maskierbare Rasterzeilen-Interrupt abgeschaltet würde. Zwei Möglichkeiten gibt es, das Problem zu lösen: Der CIA-Interrupt wird belassen, muß dann aber zu Beginn der Interrupt-Routine abgefragt (zum Beispiel durch Prüfen des Bits 0 im Request-Register 53273) und gegebenenfalls durch eine Umleitung über Sprungbefehle unschädlich gemacht werden, oder er wird eliminiert durch Schreiben des Wertes 127 in die Speicherstelle 56333 ($DCOD). Weil wir in unserem Demo-Programm weder Cursor-Blinken noch die Tastaturabfrage benötigen, entscheiden wir uns für letztere und löschen gleich zu Programmbeginn (siehe Assembler-Listing) den CIA-Interrupt völlig.

Außerdem ist zu berücksichtigen, daß der Elektronenstrahl auf allen Seiten ein Stück über den Rand des auf dem Monitor sichtbaren Bildes hinausschreibt. Dadurch entstehen Zeilennummern bis 280, so daß von Register 53266 aus bei einem Überlauf ein Highbyte nach Bit 7 des Registers 53265 übertragen werden muß. In dieser als »SCREEN« definierten Speicherstelle liegen allerdings auch einige andere Funktionen, die gegebenenfalls durch eine logische OR-Verknüpfung be-

Basic-Lader von »Rasterzeilen-Interrupt«

```
5 RUN100
10 ************************************
15 *                                  *
20 *     RASTERZEILEN-INTERRUPT       *
25 *                                  *
30 *          (BASIC-LOADER)          *
35 *                                  *
40 *        EIN  DEMO-PROGRAMM        *
45 *                                  *
50 *     FUER DEN  COMMODORE 64       *
55 *                                  *
60 *         VON HELMUT WELKE         *
65 *                                  *
70 *         MOEHNESTRASSE 26         *
75 *                                  *
80 *         5760  ARNSBERG 1         *
85 *                                  *
90 *        TEL. (02932) 23627        *
95 *                                  *
99 ************************************
100 :
110 PRINTCHR$(147)"LOADING":AD=49152
120 READX:IFX<0GOTO300
130 POKEAD,X:AD=AD+1:S=S+X:GOTO120
200 :
300 IFAD=49737ANDS=64373THENSYS49152
500 PRINT:PRINT"DATAS FEHLERHAFT"
600 :
700 :
800 :
1000 DATA 32,68,229,169,127,141,13,220
1010 DATA 141,4,220,141,5,220,162,0
1020 DATA 142,26,208,202,134,251,169
1030 DATA 11,141,17,208,133,253,169
1040 DATA 14,141,33,208,133,254,169
1050 DATA 44,160,193,141,20,3,140,21
1060 DATA 3,160,1,140,18,208,140,26
1070 DATA 208,136,169,25,32,21,193,166
1080 DATA 254,169,2,32,21,193,232,134
1090 DATA 254,224,200,144,244,166,253
1095 :
1100 DATA 169,2,32,21,193,232,134,253
1110 DATA 224,115,144,244,162,7,160
1120 DATA 13,24,32,240,255,169,200,160
1130 DATA 193,32,30,171,169,150,160
1140 DATA 0,32,21,193,169,27,141,17
1150 DATA 208,169,200,32,21,193,140
1160 DATA 26,208,140,32,208,32,68,229
1170 DATA 162,4,142,33,208,160,3,24
1180 DATA 32,240,255,169,254,160,193
1190 DATA 32,30,171,169,30,160,0,32
1195 :
1200 DATA 21,193,169,88,160,193,141
1210 DATA 20,3,140,21,3,162,25,134,253
1220 DATA 142,18,208,173,25,208,141
1230 DATA 25,208,160,1,140,26,208,136
1240 DATA 169,2,32,21,193,232,134,253
1250 DATA 224,170,144,244,169,100,32
1260 DATA 21,193,32,68,229,140,26,208
1270 DATA 169,115,160,193,141,20,3,140
1280 DATA 21,3,173,25,208,141,25,208
1290 DATA 160,0,132,251,200,140,18,208
1295 :
1300 DATA 140,26,208,169,0,32,21,193
1310 DATA 169,15,141,134,2,162,3,24
1320 DATA 32,240,255,169,221,160,193
1330 DATA 32,30,171,169,20,160,0,32
1340 DATA 21,193,206,134,2,16,237,169
1350 DATA 1,168,32,21,193,76,0,192,141
1360 DATA 6,220,140,7,220,169,17,141
1370 DATA 14,220,169,89,141,15,220,173
1380 DATA 15,220,74,176,250,96,165,251
1390 DATA 240,9,16,15,169,0,170,164
1395 :
1400 DATA 253,208,14,169,2,162,1,164
1410 DATA 254,208,6,169,7,162,255,160
1420 DATA 1,134,251,140,18,208,141,32
1430 DATA 208,173,25,208,141,25,208
1440 DATA 76,129,234,173,33,208,73,2
1450 DATA 168,41,2,240,6,165,253,24
1460 DATA 105,21,44,165,253,141,18,208
1470 DATA 140,33,208,76,79,193,198,251
1480 DATA 166,251,16,4,162,15,134,251
1490 DATA 189,152,193,72,188,184,193
1495 :
1500 DATA 189,168,193,170,104,142,18
1510 DATA 208,140,17,208,141,32,208
1520 DATA 141,33,208,76,79,193,4,11
1530 DATA 5,9,3,10,1,14,7,15,2,13,0
1540 DATA 8,6,12,1,11,254,235,219,203
1550 DATA 179,162,138,122,106,82,75
1560 DATA 59,51,33,27,155,27,27,27,27
1570 DATA 27,27,27,27,27,27,27,27,27
1580 DATA 27,14,8,31,196,69,77,79,45
1590 DATA 208,82,79,71,82,65,77,77,17
1595 :
1600 DATA 17,17,17,17,13,29,29,29,29
1610 DATA 29,29,29,29,29,210,193,211
1620 DATA 212,197,210,218,197,201,204
1630 DATA 197,206,45,201,78,84,69,82
1640 DATA 82,85,80,84,0,156,213,78,68
1650 DATA 32,78,85,78,32,32,32,32,17
1660 DATA 17,17,83,69,72,69,78,32,211
1670 DATA 73,69,32,32,32,32,17,17,17
1680 DATA 78,79,67,72,32,68,69,78,13
1690 DATA 17,17,17,17,17,29,29,29,29
1695 :
1700 DATA 29,29,29,29,29,29,210,160
1710 DATA 197,160,199,160,197,160,206
1720 DATA 160,194,160,207,160,199,160
1730 DATA 197,160,206,0
1740 DATA -1

READY.
```

rücksichtigt werden müssen. Das Demo-Programm verwendet diese Funktionen indem durch Löschen des Bits 4 der Bildschirm ausgeblendet wird, so daß auf dem Schirm nur noch ein ganzflächiger Rand erscheint, obwohl auch weiterhin Texte auf den jetzt unsichtbaren Hintergrund ausgegeben werden können. Beispiel: POKE 53265, PEEK (53265) AND 255-16 läßt den Hintergrund verschwinden, POKE 53265, PEEK(53265) OR 16 zaubert ihn dann wieder herbei.

Der Rasterzeilen-Interrupt ist eine Spezialität des C 64, die ihn zu einem äußerst vielseitigen Gerät macht. Die Programmier-Profis der Videospiel-Produzenten benutzen ihn nicht allein dafür, oben blauen Himmel und unten braune Erde darzustellen, sondern auch, um ein nur partielles Scrolling zu erzielen, um Text und hochauflösende Grafik zu mischen, um gleichzeitig normale und Multicolorzeichen zu benutzen und und und... Aber damit erschöpfen sich die Interruptmöglichkeiten des Commodore 64 noch nicht. Bit 1 (Wert = 2) der Mas-

ke in 53274 erzeugt einen Interrupt, wenn ein Sprite Berührung mit einem Zeichen hat, Bit 2 (Wert = 4) wenn Sprite mit Sprite zusammenstößt, Bit 3 (Wert = 8) wenn ein Impuls vom Lightpen kommt, oder der Feuerknopf eines am Controlport 1 angeschlossenen Joysticks gedrückt wird und schließlich Bit 7 (Wert = 128), wenn eines der genannten Ereignisse eingetreten ist.

Und dann ist da noch der bereits genannte CIA-Interrupt, der von einem gleich doppelt vorhandenen Baustein stammt. Beiden sind jedoch im C 64 teilweise unterschiedliche Aufgaben zugewiesen. CIA 1 hat die Basisadresse 56320 ($DC00), CIA 2 eine solche von 56576 ($DD00). Hier erfolgt beispielsweise die Tastaturdekodierung, die Abfrage von Joysticks, Paddles und Lightpen, die serielle Datenübertragung zu einer Schnittstelle, hier befinden sich die Echtzeituhren und die Timer. Wenden wir uns letzteren in CIA 1 zu.

Der Timer ist ein 16-Bit-Zählregister, das nach dem Starten ohne weiteres Zutun mit einer konstanten Geschwindigkeit dekrementiert, das heißt jeweils um 1 abwärts gezählt wird. Durch die zuvor beim VIC schon erwähnte Doppelfunktion besteht die Möglichkeit, den Timer ab einem bestimmten Wert zählen zu lassen: Lesen der Adresse 56324 ($DC04) liefert den aktuellen Stand des Lowbytes, Hineinschreiben den Startwert des Timers, ebenso beim Highbyte unter der Adresse 56325. Auf diese Weise wird der Systeminterrupt erzeugt, da der Computer in der Initialisierungsphase den Timer mit dem Wert 16421 (= 37 low und 64 high) lädt. Wenn der Timer über Null hinauszählt und damit einen sogenannten Unterlauf erzeugt, wird das durch Setzen des Bits 0 im Interrupt-Control-Register 56333 ($DC0D) signalisiert. Auch dieses arbeitet wieder doppelt: Lesen ergibt die Interruptanforderung, ein Schreibzugriff erzeugt die Maske, die darüber entscheidet, welches Ereignis Interruptauslöser sein soll. Besondere Beachtung beim Beschreiben des Registers verdient Bit 7, das darüber bestimmt, ob die nachfolgend gesetzten Bits 6 bis 0 in der Maske gesetzt oder gelöscht werden. Alle übrigen bleiben unangetastet. Deshalb löscht 127 (Bit 7 nicht gesetzt!) im Demo-Programm sämtliche Maskenbits (siehe Bitmuster im Assembler-Listing, Zeile 1010), deshalb setzt 129 (Bitmuster 10000001) das Bit 0 der Maske und schaltet damit auf Interrupt durch den Timer. Erzeugen Sie doch einmal im Direktmodus auf dem Bildschirm ein längeres Zählintervall durch Heraufsetzen des Timer-Highbytes: POKE 56325, 255 läßt den Cursor sehr träge werden, POKE 56325, 5 versetzt ihn in nervöses Flattern.

Dieser Timer besitzt einen Zwillingsbruder mit den Adressen 56326 und 56327, der nicht nur gleichartig konstruiert ist und ebenfalls eigenständig einen Interrupt auf Bit 1 des Control-Registers erzeugen kann, sondern sich auch mit dem Timer A koppeln läßt. Beide Timer können nämlich auf verschiedene Taktquellen gelegt, unterschiedlich getriggert werden. Im Demo-Programm nutzen wir das aus, indem wir Timer A Systemtakte zählen lassen (Bit 5 des Registers 56334 gelöscht — Standardeinstellung), Timer B hingegen nur die Unterläufe von Timer A durch Setzen des Bits 6 im Register 56335. Dadurch erhalten wir einen 32-Bit-Zähler, der beliebige Zeitverzögerungen ermöglicht, eleganter als mit jeder ausschließlich softwaremäßig realisierten Warteschleife, weil die Timer von keinem Interrupt unterbrochen werden.

In der Warteschleife des Demo-Programms (Listing ab Zeile 9010), die als Subroutine angelegt ist, wird mit den Timern kein Interrupt erzeugt, sondern lediglich eine Zeitverzögerung erzielt. Dazu wird zunächst Timer B mit den vom Hauptprogramm im Akku und im Y-Register übergebenen Werten geladen, während Timer A konstant mit einem mittleren Wert arbeitet. Dann werden beide gestartet durch Setzen des Bits 0 im Register 56334 für Timer A und 56335 für Timer B. Aus Zeile 9030 des Listings ist ersichtlich, daß Bit 3 für Timer A gelöscht ist, was Continuous- oder Dauerbetrieb zur Folge hat. Jedesmal, wenn der Timer einen Unterlauf hat, lädt er umgehend wieder den zwischengespeicherten Startwert und beginnt erneut zu zählen. Den anderen schalten wir hingegen auf One-Shot (Zeile 9050), einen »Einzelschuß«. Bei einem Unterlauf lädt er zwar wieder den Startwert, bleibt aber stehen, wodurch sein Start/Stoppbit automatisch gelöscht wird. Wir prüfen dies, indem wir Bit 0 logisch nach rechts ins Carry verschieben, wo es bequem mit einem Branchbefehl untersucht werden kann. Erst der One-Shot-Betrieb des Timers B stellt sicher, daß ein Unterlauf auch erkannt wird, weil er im Dauerbetrieb vorübergehen könnte, während sich der Prozessor im Interrupt befindet. Außerdem verlangt ein gesetztes Bit 4 in der Warteschleife für beide Timer Force-Load, einen unbedingten Ladevorgang. Unabhängig davon, ob der Timer gerade läuft oder nicht, wird der Startwert geladen. Mit einem gelöschten Bit 4 kommt ein neuer Startwert erst dann zum Tragen, wenn er nach dem nächsten Unterlauf geladen wird.

Neben den Unterläufen der beiden Timer kann das Control-Register 56333 auf Bit 2 auch einen Interrupt erzeugen bei Übereinstimmung der Echtzeituhr 56328 bis 56331 mit einer vorgewählten Alarmzeit, auf Bit 3 durch ein volles oder leeres Schieberegister (56332) und auf Bit 4 durch den Impuls einer externen Signalquelle. Ein gesetztes Bit 7 zeigt hier an, daß mindestens eines der gesetzten Bits auch in der Maske gewählt ist, also ein Interrupt stattfindet. Das Interrupt-Flag wird aber bereits durch Lesen des Registers gelöscht.

Diese Kurzabhandlung vermag nur ansatzweise darzustellen, wie flexibel das Instrumentarium ist, das hier dem Anwender zur Verfügung steht. Speziell die zahlreichen Interruptmöglichkeiten verlangen geradezu nach Anwendung, wobei wir abschließend auf eine bisher nicht erwähnte noch zu sprechen kommen müssen. Denn das Demo-Programm läuft in einer Endlosschleife, die nicht ohne weiteres abgebrochen werden kann. Außerdem wird ja durch den lahmgelegten Systeminterrupt ohnehin kein Tastendruck mehr erkannt. Den Ausweg aus diesem Dilemma, haben die Konstrukteure geschaffen, als sie den sogenannten NMI erfanden. Das ist die Abkürzung für nicht maskierbarer Interrupt, eine hardwareseitige Unterbrechungsmöglichkeit mit so hoher Priorität, daß selbst ein gesetztes Interruptflag keine Rolle spielt. Wir können den NMI auslösen, indem wir die STOP-Taste gedrückt halten und gleichzeitig auf die RESTORE-Taste klopfen. Und schon ist alles wieder normal: Bild und Interrupt. Sie können zwar gleich wieder mit SYS 49152 ins Demo-Programm starten, doch vielleicht lassen Sie sich selbst einmal etwas einfallen. Nur zu — unterbrechen Sie doch Ihren Commodore 64 bitte mal ...

(Helmut Welke/aa)

Die Symboltabelle für das Maschinensprache-Programm »Rasterzeilen-Interrupt«

CHRCOL	0286	CLEAR	E544
COLOR	C198	CRA	DC0E
CRB	DC0F	EXIT	C147
FLAG	00FB	GETPAR	C17D
GOLD	C141	GRUND	D021
HIGH	C1B8	ICR	DC0D
INIT	C000	INTRO	C1C8
IREQU	D019	IRMASK	D01A
IRQ1	C12C	IRQ2	C158
IRQ3	C173	IRQOFF	EA81
LINE	00FD	LOOP1	C03E
LOOP2	C04C	LOOP3	C0B7
LOOP4	C0F9	LOW	C1A8
NAME	C1DD	OBEN	C168
PLOT	FFF0	PRINT	AB1E
RAND	D020	RASTER	D012
ROT	C139	SCREEN	D011
TEXT	C1FE	TIME	C115
TIMERA	DC04	TIMERB	DC06
VECTOR	0314	WAIT	C125

Assemblerlisting des »Rasterzeilen-Interrupt«

```
                    R A S T E R Z E I L E N - I N T E R R U P T

                    Ein Demo-Programm fuer den COMMODORE 64

          von Helmut Welke, Moehnestrasse 26, 5760 Arnsberg 1, Telefon (02932) 23627

                          ***** ASSEMBLERLISTING *****


Pass 2
30:   c000                    .opt p1,oo
                              ;
50:   c000                    *=  $c000   ;startadresse 49152
                              ;
100:  c000              clear =   $e544   ;bildschirm loeschen
110:  c000              icr   =   $dc00+13 ;interrupt-control-register
120:  c000              irmask =  $d000+26 ;interruptmaske
130:  c000              rand  =   $d000+32 ;farbe bildrand
140:  c000              grund =   rand+1  ;farbe bildhintergrund
150:  c000              screen =  $d000+17 ;bildschirm-steuerregister
160:  c000              raster =  screen+1 ;elektronenstrahlzeile
170:  c000              vector =  $0314   ;interrupt-sprungadresse
180:  c000              timera =  $dc00+4 ;16-bit-zaehler
190:  c000              timerb =  timera+2 ;dito
200:  c000              fla9  =   $fb     ;freie speicherstelle
210:  c000              cra   =   icr+1   ;steuerregister timer a
220:  c000              crb   =   cra+1   ;steuerregister timer b
230:  c000              line  =   $fd     ;freie speicherstelle
240:  c000              irqu  =   irmask-1 ;interrupt-register
250:  c000              irqoff =  $ea81   ;rueckkehr vom interrupt
260:  c000              Plot  =   $fff0   ;cursor setzen/holen
270:  c000              Print =   $abie   ;string ausgeben
280:  c000              chrcol =  $0286   ;aktuelle zeichenfarbe
                              ;
                              ;*** programmvorbereitung ***
                              ;
1000: c000 20 44 e5 init      jsr  clear
1010: c003 a9 7f              lda  #%01111111 ;alle interruptmoeglichkeiten
1020: c005 8d 0d dc           sta  icr      ;loeschen
1030: c008 8d 04 dc           sta  timera   ;zaehlwerk auf
1040: c00b 8d 05 dc           sta  timera+1 ;32639 einstellen
1050: c00e a2 00              ldx  #0
1060: c010 8e 1a d0           stx  irmask   ;kein vic-interrupt
1070: c013 ca                 dex
1080: c014 86 fb              stx  fla9     ;zaehler
1090: c016 a9 0b              lda  #11
1100: c018 8d 11 d0           sta  screen   ;bildschirm ausblenden
1110: c01b 85 fd              sta  line     ;zwischenspeicher zeile
1120: c01d a9 0e              lda  #14      ;hellblaue farbe
1130: c01f 8d 21 d0           sta  grund
1140: c022 85 fe              sta  line+1   ;wie 1110
                              ;
                              ;*** interrupt-beispiel bundesflagge ***
                              ;
1150: c024 a9 2c              lda  #<ir41   ;zeiger
1160: c026 a0 c1              ldy  #>ir41   ;auf
1170: c028 8d 14 03           sta  vector   ;neue
1180: c02b 8c 15 03           sty  vector+1 ;interrupt-routine
1190: c02e a0 01              ldy  #1
1200: c030 8c 12 d0           sty  raster   ;irr-ausloeser rasterzeile 1
1210: c033 8c 1a d0           sty  irmask   ;irr-maske auf rasterzeilen
1220: c036 88                 dey
1230: c037 a9 19              lda  #25
1240: c039 20 15 c1           jsr  time     ;zeitverzoegerung
1250: c03c a6 fe              ldx  line+1
1260: c03e a9 02    loop1     lda  #2
1270: c040 20 15 c1           jsr  time
1280: c043 e8                 inx
1290: c044 86 fe              stx  line+1   ;eine zeile weiter
1300: c046 e0 c8              cpx  #200     ;bis zeile 200
1310: c048 90 f4              bcc  loop1    ;kleiner - dann weiter
1320: c04a a6 fd              ldx  line     ;dito obere zeile
1330: c04c a9 02    loop2     lda  #2
1340: c04e 20 15 c1           jsr  time
1350: c051 e8                 inx
1360: c052 86 fd              stx  line
1370: c054 e0 73              cpx  #115
1380: c056 90 f4              bcc  loop2
                              ;
                              ;*** titel-einblendung ***
                              ;
1390: c058 a2 07              ldx  #7
1400: c05a a0 0d              ldy  #13
1410: c05c 18                 clc
1420: c05d 20 f0 ff           jsr  Plot
1430: c060 a9 c8              lda  #<intro ;stringadresse low
1440: c062 a0 c1              ldy  #>intro ;und high
1450: c064 20 1e ab           jsr  Print    ;titel drucken
1460: c067 a9 96              lda  #150
1470: c069 a0 00              ldy  #0
1480: c06b 20 15 c1           jsr  time
1490: c06e a9 1b              lda  #27
1500: c070 8d 11 d0           sta  screen   ;bildschirm oeffnen
1510: c073 a9 c8              lda  #200
1520: c075 20 15 c1           jsr  time
                              ;
                              ;*** interrupt-beispiel gleitzeile ***
                              ;
1530: c078 8c 1a d0           sty  irmask   ;interrupt aus
1540: c07b 8c 20 d0           sty  rand     ;schwarze umrandung
1550: c07e 20 44 e5           jsr  clear
1560: c081 a2 04              ldx  #4       ;farbe purpur
1570: c083 8e 21 d0           stx  grund
1580: c086 a0 03              ldy  #3
1590: c088 18                 clc
1600: c089 20 f0 ff           jsr  Plot
1610: c08c a9 fe              lda  #<text
1620: c08e a0 c1              ldy  #>text
1630: c090 20 1e ab           jsr  Print
1640: c093 a9 1e              lda  #30
1650: c095 a0 00              ldy  #0
1660: c097 20 15 c1           jsr  time
1670: c09a a9 58              lda  #<ir42
1680: c09c a0 c1              ldy  #>ir42
1690: c09e 8d 14 03           sta  vector   ;neue
1700: c0a1 8c 15 03           sty  vector+1 ;interrupt-routine
1710: c0a4 a2 19              ldx  #25
1720: c0a6 86 fd              stx  line
1730: c0a8 8e 12 d0           stx  raster
1740: c0ab ad 19 d0           lda  irqu
1750: c0ae 8d 19 d0           sta  irqu     ;interrupt-fla9 loeschen
1760: c0b1 a0 01              ldy  #1
1770: c0b3 8c 1a d0           sty  irmask
1780: c0b6 88                 dey
1790: c0b7 a9 02    loop2     lda  #2
1800: c0b9 20 15 c1           jsr  time
1810: c0bc e8                 inx
1820: c0bd 86 fd              stx  line     ;eine zeile weiter
1830: c0bf e0 aa              cpx  #170
1840: c0c1 90 f4              bcc  loop3
1850: c0c3 a9 64              lda  #100
1860: c0c5 20 15 c1           jsr  time
                              ;
                              ;*** interrupt-beispiel regenbogen ***
1870: c0c8 20 44 e5           jsr  clear
1880: c0cb 8c 1a d0           sty  irmask
1890: c0ce a9 73              lda  #<ir43
1900: c0d0 a0 c1              ldy  #>ir43
1910: c0d2 8d 14 03           sta  vector
1920: c0d5 8c 15 03           sty  vector+1
1930: c0d8 ad 19 d0           lda  irqu
1940: c0db 8d 19 d0           sta  irqu
1950: c0de a0 00              ldy  #0
1960: c0e0 84 fb              sty  fla9
1970: c0e2 c8                 iny
1980: c0e3 8c 12 d0           sty  raster
1990: c0e6 8c 1a d0           sty  irmask
2000: c0e9 a9 00              lda  #0
2010: c0eb 20 15 c1           jsr  time
2020: c0ee a9 0f              lda  #15      ;hellgrau
2030: c0f0 8d 86 02           sta  chrcol   ;zeichenfarbe
2040: c0f3 a2 03              ldx  #3
2050: c0f5 18                 clc
2060: c0f6 20 f0 ff           jsr  Plot
2070: c0f9 a9 dd    loop4     lda  #<name
2080: c0fb a0 c1              ldy  #>name
2090: c0fd 20 1e ab           jsr  Print    ;text ausgeben
2100: c100 a9 14              lda  #20
2110: c102 a0 00              ldy  #0
2120: c104 20 15 c1           jsr  time
2130: c107 ce 86 02           dec  chrcol   ;farbe - 1
2140: c10a 10 ed              bpl  loop4    ;weiter bis einschl. 0 (=schwarz)
2150: c10c a9 01              lda  #1
2160: c10e aa                 tax
2170: c10f 20 15 c1           jsr  time
2180: c112 4c 00 c0           jmp  init     ;auf ein neues
                              ;
                              ;*** warteschleife ***
                              ;
9010: c115 8d 06 dc time      sta  timerb   ;zaehler low
9020: c118 8c 07 dc           sty  timerb+1 ;und high laden
9030: c11b a9 11              lda  #%00010001 ;force-load/continuous/systemtakt
9040: c11d 8d 0e dc           sta  cra      ;timer a starten
9050: c120 a9 59              lda  #%01011001 ;force-load/one-shot/takt timer a
9060: c122 8d 0f dc           sta  crb      ;timer b starten
9070: c125 ad 0f dc wait      lda  crb      ;b-control lesen
9080: c128 4a                 lsr          ;bit 0 ins carry schieben
9090: c129 b0 fa              bcs  wait     ;timer b laeuft noch
9100: c12b 60                 rts          ;zurueck wenn aus
                              ;
                              ;*** interrupt-routinen ***
                              ;
10010: c12c a5 fb     ir41    lda  fla9     ;sektorenfla9
10020: c12e f0 09             beq  rot
10030: c130 10 0f             bpl  gold
10040: c132 a9 00             lda  #0
10050: c134 aa                tax
10060: c135 a4 fd             ldy  line
10070: c137 d0 0e             bne  exit
10100: c139 a9 02     rot     lda  #2
10110: c13b a2 01             ldx  #1
10120: c13d a4 fe             ldy  line+1
10130: c13f d0 06             bne  exit
10200: c141 a9 07     gold    lda  #7
10210: c143 a2 ff             ldx  #255
10220: c145 a0 01             ldy  #1
10500: c147 86 fb     exit    stx  fla9     ;zeiger auf sektor
10510: c149 8c 12 d0          sty  raster   ;naechste interruptzeile
10520: c14c 8d 20 d0          sta  rand     ;farbe setzen
10530: c14f ad 19 d0          lda  irqu
10540: c152 8d 19 d0          sta  irqu     ;ir4-fla9 loeschen
10550: c155 4c 81 ea          jmp  irqoff   ;und ausprung
                              ;
11000: c158 ad 21 d0 ir42     lda  grund
11010: c15b 49 02             eor  #%00000010 ;farb-flipflop 4/6
11020: c15d a8                tay
11030: c15e 29 02             and  #%00000010
11040: c160 f0 06             beq  oben     ;purpur an der reihe
11050: c162 a5 fd             lda  line
11060: c164 18                clc
11070: c165 69 15             adc  #21      ;+ 21 rasterzeilen
11080: c167 2c                .byte$2c      ;= bit (versteckter ladebefehl)
11090: c168 a5 fd     oben    lda  line
11100: c16a 8d 12 d0          sta  raster   ;zeile und
11110: c16d 8c 21 d0          sty  grund    ;farbe setzen
11120: c170 4c 4f c1          jmp  exit+8   ;ausprung
                              ;
12000: c173 c6 fb     ir43    dec  fla9
12010: c175 a6 fb             ldx  fla9     ;offset fuer tabellen
12020: c177 10 04             bpl  getpar
12030: c179 a2 0f             ldx  #low-color-1
12040: c17b 86 fb             stx  fla9     ;offset maximal
12050: c17d bd 98 c1 getpar   lda  color,x  ;farbe holen
12060: c180 48                pha          ;zunaechst auf stapel
12070: c181 bc b8 c1          ldy  high,x   ;rasterzeile highbyte
12080: c184 bd a8 c1          lda  low,x    ;und lowbyte
12090: c187 aa                tax
12100: c188 68                pla          ;farbe zurueckholen
12110: c189 8e 12 d0          stx  raster   ;zeile low
12120: c18c 8c 11 d0          sty  screen   ;und high
12130: c18f 8d 20 d0          sta  rand     ;und randfarbe und
12140: c192 8d 21 d0          sta  grund    ;hintergrundfarbe setzen
12150: c195 4c 4f c1          jmp  exit+8   ;fertig
                              ;
                              ;*** interrupt-tabellen ***
                              ;
15000: c198 04 0b 05 color    .byte4,11,5,9
15010: c19c 03 0a 01          .byte3,10,1,14
15020: c1a0 07 0f 02          .byte7,15,2,13
15030: c1a4 00 08 06          .byte0,8,6,12
                              ;
16000: c1a8 01 0b fe low      .byte1,11,254,235
16010: c1ac db cb b3          .byte219,203,179,162
16020: c1b0 8a 7a 6a          .byte138,122,106,82
16030: c1b4 4b 3b 33          .byte75,59,51,33
                              ;
17000: c1b8 1b 9b 1b high     .byte27,155,27,27
17010: c1bc 1b 1b 1b          .byte27,27,27,27
17020: c1c0 1b 1b 1b          .byte27,27,27,27
17030: c1c4 1b 1b 1b          .byte27,27,27,27
                              ;
                              ;*** tabellen hauptprogramm ***
                              ;
20000: c1c8 0e 08 1f intro    .byte14,8,31
20010: c1cb c4 45 4d          .asc "Demo-Programm"
20020: c1d8 11 11 11          .byte17,17,17,17,17
20030: c1dd 0d 1d 1d name     .byte13,29,29,29,29
20040: c1e2 1d 1d 1d          .byte29,29,29,29,29
20050: c1e7 d2 c1 d3          .asc "RASTERZEILEN-Interrupt"
20060: c1fd 00                .byte0
                              ;
21000: c1fe 9c        text    .byte156
21010: c1ff d5 4e 44          .asc "Und nun     "
21020: c20a 11 11 11          .byte17,17,17
21030: c20d 53 45 48          .asc "sehen Sie     "
21040: c21a 11 11 11          .byte17,17,17
21050: c21d 4e 4f 43          .asc "noch den"
21060: c225 0d 11 11          .byte13,17,17,17,17,17
21070: c22b 1d 1d 1d          .byte29,29,29,29,29
21080: c230 1d 1d 1d          .byte29,29,29,29,29
21090: c235 d2 a0 c5          .asc "R E G E N B O G E N"
21100: c248 00                .byte0

end of assembly 4c000 - 4c248

no errors
```