Autostart für den VC 20

Will man Programme vor unbefugten Eingriffen schützen, so ist dies nur möglich, indem man die Programme auf EPROMs ab $A000 schreibt, die sich dann nach dem Einschalten des VC 20 selbst starten (Modulspiele, Toolkit). Es ist aber auch möglich, solche »Autostart«-Programme mit der Datasette und dem VC 20 zu generieren.

Ein automatischer Programmstart ist überhaupt erst möglich, weil das Betriebssystem des Computers nach dem Laden eines Programmes vom Kassettenrecorder einen indirekten Sprung über einen Vektor der Zeropage zu einer Routine des Betriebssystems durchführt. Beim VC 20 findet dieser Sprung über den Vektor $0302/$0303 zum Basic-Warmstart des Interpreters statt. Dieser Vektor ist jetzt so zu verändern, daß er auf eine selbstgeschriebene Routine zeigt, die dann nach dem Laden eines Programmes von der Kassette sofort gestartet wird. Um nun eine Wirkung zu erzielen, muß dieser Vektor also während des Ladevorgangs an seine ursprüngliche Adresse ($0302/$0303) geschrieben werden. Normalerweise werden aber Programme, die mit SAVE gespeichert wurden, vom Computer als Basic-Programme angesehen und daher nicht unbedingt immer an ihre ursprüngliche Adresse geladen, sondern an den Anfang des Basic-Speichers verschoben. Speichert man dagegen die Programme mit SAVE "Name",1,1 ab, so werden sie später immer in den Speicherbereich geladen, in dem sie zum Zeitpunkt des SAVEns gestanden haben. Setzen wir doch zum Beispiel in einem Programm (Listing 1) den Basic-Warmstartvektor auf $FD22 (High Byte: 253/ Low Byte: 34) und die Basicanfang/ende-Zeiger auf den Bereich des LOAD-Vektors ($2B:01/$2C:03/$2D:04/$2E:03). Wenn wir diesen dann mit SAVE "RESET",1,1 abspeichern, erfolgt nach dem späteren Laden des Programms mit LOAD ein Sprung zu der Adresse, die durch den Vektor gekennzeichnet wird. In diesem Fall wird sofort nach dem Stoppen des Kassettenmotors ein Reset durchgeführt (SYS 64802).

Ein kleiner Trick hilft weiter

Mit dieser Methode läßt sich aber nur der Basic-Warmstartvektor abspeichern, da das Programm im Basicspeicher steht und mit dem SAVE-Befehl nur zusammenhängende RAM-Bereiche gespeichert werden können. Daher wird ein Ladeprogramm benutzt, das das eigentliche Basicprogramm in den Programmspeicher lädt und selbst im unteren Bereich des Kassetten-Puffers ($033C-$03FC) steht. Dort wird es dann als Name (Tape-Header) zusammen mit dem veränderten Vektor auf die Kassette geschrieben. Damit ist auch der Wert des Vektors auf den Bereich des Kassetten-Puffers festgelegt.

Wird jetzt der Kassetten-Puffer mit dem Basic-Warmstartvektor (der auf den Anfang des Ladeprogramms, $0355, zeigt) und dem Befehl SAVE " ",1,1 auf eine Kassette geschrieben und wieder geladen, fängt der Computer nach dem Stoppen des Kassettenmotors sofort an, das Programm ab $0355 auszuführen. Statt des Ladeprogramms kann auch eine Hilfsroutine, die nicht länger als 192 Byte ist, in den Kassettenpuffer geschrieben werden, die dann nach dem Laden, ohne speziellen Startbefehl, sofort zur Verfügung steht. Schließlich wird das zu schützende Basicprogramm nach dem Vektor, ohne Tape-Header (Programmname), abgespeichert. Würde das Basic-Programm normal geSAVEt werden, bräuchte man das Autostart-Programm nur zu übergehen und könnte das zu schützende Basic-Programm auch so laden. Das Lade-Programm im Kassettenpuffer lädt dann das Basic-Programm in den Speicher. Danach werden die Vektoren $0318/0319 (NMI-Vektor) und $0308/$0309 (Text-Stop-Vektor) so geändert, daß erstens die Stop-Taste nicht mehr abgefragt wird und zweitens beim Versuch, das Basic-Programm mit RUN/STOP + RESTORE zu stoppen, das Basic-Programm abgebrochen wird und in einer Endlosschleife die Ausgabe eines Textes (»Autostart ist gesichert«) erfolgt. Zum Schluß wird das Basic-Programm mit RUN (in Maschinensprache: JSR $C659 +JMP $C7AE) gestartet.

Bedienung des Programms »Autostart« (Listing 2)

1. Basic-Lader eintippen und noch einmal genau die DATA-Statements prüfen.
2. Basic-Lader mit RUN starten; daraufhin wird das Maschinenprogramm generiert und vom Basic-Lader aus mit SAVE "Autostart Gen.",1,1 gspeichert.
3. Laden des Maschinenprogramms "Autostart Gen." mit LOAD und Eingabe von: POKE 56,28:NEW.
4. Laden des zu schützenden Programms.
5. Autostart-Generator mit SYS 7168 starten.
6. Programmname auf Aufforderung hin eingeben; das zu schützende Programm wird dann als Autostart-Programm geSAVEt.

Es ist darauf zu achten, daß die Programme, die mit Autostart versehen werden, ausgearbeitete Programme sind, die nicht noch irgendeine Ergänzung oder Verbesserung benötigen. Da die Autostart-Programme nicht mehr zu stoppen sind, kann später auch nicht mehr an ihnen gearbeitet werden, so daß ein unfertiges Programm noch einmal neu eingetippt werden müßte.

```
10 POKE43,1:POKE44,3:POKE45,4:POKE46,3:CLR:
   REM Basicanfang/ende setzen
20 POKE770,34:POKE771,253:     :REM Basic-
   Warmstartvektor
30 SAVE "Reset",1,1
40 POKE770,131:POKE771,196:POKE43,1:POKE44,
   18:POKE45,185:POKE46,18:END
50 REM Zeiger und Vektoren wieder auf alten Wert
   bringen
```

Das Programm "Autostart" läuft nur auf einem VC 20 mit 3,5 KByte Speicher (Grundversion) sowie der Datasette und belegt die obersten beiden Pages von 7168 bis 7616. Die verschlüsselten Programme werden automatisch in den Grundversionsspeicher geschrieben. Daher ist es mit dieser Version nicht möglich, Programme, die auf einer Speichererweiterung laufen, mit "Autostart" zu versehen.

(Dirk Rother/ev)

```
0 POKE36879,8:PRINT"■ AUTOSTART":PRINT"
■COPYRIGHT 1983":PRINT"■BY D.ROTHER"
1 PRINT"■■PROGRAMM WIRD GELADEN (AB7168)
.............."
2 POKE56,28:POKE55,0:CLR
3 FORI=7168TO7617:READA:POKEI,A:SU=SU+A:
NEXT:IFSU<>40847THENPRINT"TIPPFEHLER":EN
D
4 PRINT"■BITTE ■ SPACE ■ DRUECKEN":WAIT1
98,1:PRINT"■MASCHINENPROGRAMM WIRD GESPE
ICHERT■■"
5 POKE56,30:POKE44,28:POKE43,0:POKE46,29
:POKE45,200:SAVE"AUTOSTART GEN.",1,1
6 POKE43,1:POKE44,16:POKE45,172:POKE46,2
3:END
22 DATA169,8,141,15,144,169,3,141,134,2,
32,95,229,162,20,169,32,157,236,28,202,2
08
24 DATA250,169,168,160,28,32,30,203,162,
0,169,0,133,198,165,198,240,252,173,119
26 DATA2,201,13,240,13,157,236,28,41,63,
157,132,30,232,224,20,144,228,165,43,141
28 DATA131,29,165,44,141,132,29,165,45,1
41,133,29,165,46,141,134,29,173,2,3,72,1
73
30 DATA3,3,72,169,3,133,44,133,46,141,3,
3,169,1,133,43,169,4,133,45,169,85,141,2
32 DATA3,169,191,162,236,160,28,32,189,2
55,169,1,170,168,32,186,255,32,86,225,17
3
34 DATA131,29,133,43,133,193,173,132,29,
133,44,133,194,173,133,29,133,45,133,174
36 DATA173,134,29,133,46,133,175,104,141
,3,3,104,141,2,3,76,21,247,65,85,84,79,8
3
38 DATA84,65,82,84,13,32,40,67,41,49,57,
56,52,13,68,46,32,82,79,84,72,69,82,13,1
3
40 DATA80,82,79,71,82,65,77,77,78,65,77,
69,58,13,13,45,45,45,45,45,45,45,45,45,4
5
42 DATA45,45,45,45,45,45,45,45,45,45,13,
13,0,80,71,77,45,83,67,72,85,84,90,32,73
44 DATA32,32,32,32,32,32,32,32,32,91,228
,32,82,253,169,0,141,34,145,173,216,3,13
3
46 DATA193,133,43,173,217,3,133,194,133,
44,173,218,3,133,174,133,45,173,219,3,13
3
48 DATA175,133,46,32,61,246,169,3,141,25
,3,141,41,3,169,161,141,24,3,169,156,141
50 DATA40,3,169,255,141,34,145,32,89,198
,76,174,199,169,0,201,254,96,169,0,141,3
4
52 DATA145,32,95,229,169,8,141,15,144,16
9,1,141,134,2,169,189,160,3,32,30,203,76
54 DATA166,3,65,85,84,79,83,84,65,82,84,
13,32,32,32,73,83,84,13,71,69,83,73,67,7
2
56 DATA69,82,84,0,1,16,171,17,32,32,32,3
2,32,32,32,32,32,32,32,32,32,32,32,32,32
58 DATA32,32,32,32,32,32,32,32,32,32,32,
32,0,0,1,255,255,255,255,255,128,0,0,0,0
60 DATA255,255,255,255,32,32,32,32,32,32
,32,32,32,32,32,0,0
62 DATA2,3,72,173,3
READY.
```

Listing 2. Basic-Lader für "Autostart"

View BAM

Dieses Programm zeigt Ihnen sekundenschnell, wie und wo die Floppy Dateien und Programme abspeichert, welche Blöcke noch frei sind und welche nicht.

Das hier vorgestellte Programm stellt die sogenannte BAM der Disketten grafisch auf dem Bildschirm des C 64 dar. Das gesamte Programm ist in Maschinensprache geschrieben und daher sehr schnell. Es ist dem Viewbam-Programm, das von Commodore auf der Testdiskette mitgeliefert wird, in zweierlei Hinsicht überlegen:
1. ist es, wie bereits erwähnt, um ein Vielfaches schneller und
2. stellt es übersichtlich die gesamte Diskettenbelegung dar, wobei freie, belegte und nicht belegbare Blöcke gut zu erkennen sind.

Eine Hardcopy läßt sich auch leicht anfertigen (siehe Bild 1).

Das Programm braucht inklusive Ladezeit des Blockes 18/0, auf dem die BAM gespeichert ist, unter drei Sekunden zur Ausführung. (Im Vergleich zum Viewbam von Commodore, das 73 Sekunden benötigt.)

Es ist interessant zu sehen, wie das DOS Files abspeichert, welche Blöcke noch frei sind und welche nicht. Ich wurde durch die geringe Geschwindigkeit und die fehlende Möglichkeit des Viewbam von Commodore, alle Blöcke auf einmal zu betrachten, zu diesem Programm animiert.

Die Routine läuft auf dem C 64 von Commodore mit dem Floppy-Laufwerk VC 1541. Die Routine steht im geschützten Speicherbereich $C000-$C21C, dezimal 49152 bis 49693. Man kann sie auch in einem Basicprogramm aufrufen.

Mit diesem Programm wird in grafischer Darstellung ersichtlich, wieviele von den 664 Blöcken besetzt beziehungsweise frei sind.

Nach mühevoller Eingabe der Data-Zeilen und des Ladeprogramms sollte man dieses erst einmal auf Diskette sichern.

Eine Prüfsumme wird berechnet; dadurch vermeidet man eventuelle Eingabefehler.

Zuerst wird die BAM, die sogenannte »Block Availiability Map«, von der Floppy in den Speicher des Computers geladen. Die BAM ist im Block 18/0, das heißt Track 18, Sektor 0, auf jeder Diskette gespeichert.

Die BAM umfaßt 256 Bytes, die im einzelnen folgende Bedeutung haben:

Bytes 0-1	Track und Sektor vom ersten Directory-Block (meistens 18/1).
Byte 2	Hier steht ein »A« (ASCII-Code 65) für das 4040 Format
Byte 3	nicht belegt
Bytes 4-143	Hier stehen die für dieses Programm wichtigen Informationen
Bytes 144-161	Diskettenname, aufgefüllt mit geshifteten Leerzeichen
Bytes 162-163	Disk ID
Byte 164	Geshiftetes Leerzeichen
Bytes 165-166	»2A«-ASCII Repräsentation für die DOS-Version
Bytes 166-167	Geshiftete Leerzeichen
Bytes 168-255	Nullen, unbenutzt

Es existieren 35 Tracks, für jeden Track wurden in dem für uns wichtigen Bereich von Byte 4 bis Byte 143 vier Bytes reserviert. Die Tracks sind auch noch in Sektoren unterteilt, maximal 21 und mindestens 17, dies ist durch die Formel 2*Pi*Radius begründet, das heißt, daß bei kleinerem Radius nach innen weniger Raum zur Verfügung steht.