Ins ROM mit Dir!

Wie man seine eigenen Routinen, Programme und RSX-Erweiterungen ins ROM bekommt.

Der CPC kennt mehrere Typen von ROMs. Das erste ist Typ 0, der Vordergrundprogramme, wie z.B. das eingebaute BASIC, enthält. Beim ROM auf der CPC-Platine, als On-Board-ROM bezeichnet, ist Bit 7 gesetzt. Sollten einem Vordergrundprogramm die 16 KB nicht reichen, gibt es noch den Typ 2, die Erweiterungs-ROMs. Von dem können bis zu drei ROMs an den nachfolgenden Selekt-Adressen angehängt werden. Die Selekt-Adresse ist die Nummer des ROMs, AMSDOS hat beispielsweise eine 7. Welches ROM welche Selekt-Adresse hat, kann in der Regel in den Erweiterungen mittels Jumper o.ä. vom Benutzer selbst festgelegt werden. Die Routinen in den ErweiterungsROMs können dann mit einem RST 2 aufgerufen werden. Uns interessiert Typ 1, das Hintergrund-ROM. ROMs werden, vom Lower-ROM, das das Betriebssystem enthält, immer in das obere Speicherviertel, also im Adreßbereich von &C000 bis &FFFF eingeblendet. Die Selekt-Adresse kann beim CPC464 nur &1 bis &7 sein, beim CPC464+, 664 und 6128(+) &1 bis &F (=15), da der Vector KL ROM WALK nur diese Adressen abfrägt. Hardwaremäßig können am CPC aber bis zu 252 ROMs angeschlossen werden, vorausgesetzt, man bastelt sich ein ROM-Board, das eine derartige Kapazität bietet. Damit diese dann aber auch vom Betriebssystem angesprochen werden, wäre ein umfangreicher Eingriff in die Firmware nötig.

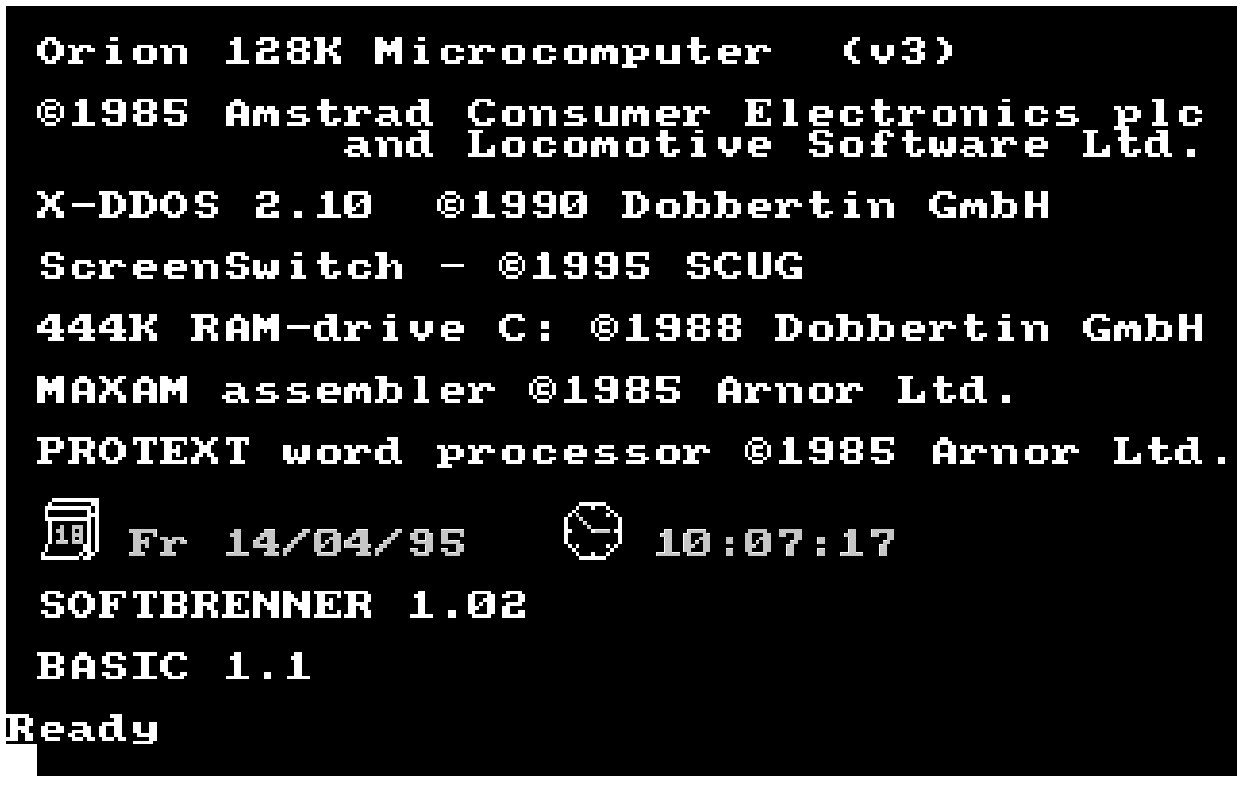
Jetzt zum Aufbau eines ROMs. An Adresse &C000, also im ersten Byte, steht die Typ-Nr., meistens ist das Typ 1. Die drei darauffolgenden Bytes enthalten in der Regel die Versionsnummer. Im Beispiellisting ist das 1.02, was durch die Bytes 1, 0 und 2 im ROM festgelegt wird. Weil diese drei Bytes das Betriebssystem nicht das geringste interessieren, könnten sie aber genausogut das Datum oder den Kosenamen der Freundin, z. B. Evi, enthalten. Die beiden Bytes danach sind dagegen enorm wichtig: Sie enthalten die Adresse der Namenstabelle. Wie beim Z80 üblich, steht das Lower- wieder von dem Higherbyte, aber das erledigt ja der Assembler für uns. Jeder Eintrag in dieser Tabelle kann bis zu 16 Bytes lang sein und wird dadurch abgeschlossen, daß Bit 7 am letzten Zeichen des Eintrags gesetzt ist. Zuerst kommt der Name des ROMs. Damit der nicht mit einem RSX-Befehl verwechselt wird, sollte ein Leerzeichen, Kleinbuchstabe o.ä. eingebaut werden. Danach kommen die Befehle. Am Schluß dieser Tabelle steht eine 0. An Adresse &C006 kommt dann der Sprungbefehl zur Initialisierungsroutine des ROMs, danach die Sprungbefehle zu den einzelnen RSX-Befehlen in der Reihenfolge, in der die RSX-Kommandos in der Namenstabelle stehen.

Was hat denn nun die Initialisierungsroutine zu tun? In BC bekommt sie die Adresse des letzten Bytes des verfügbaren, in HL das letzte Byte des noch freien und in DE das erste Byte des Arbeitsspeichers. HL kann um die Menge Speicher gesenkt werden, die das ROM selbst an Speicher benötigt, BC und DE sollten unverändert an das Betriebssystem zurückgegeben werden. Wenn am Ende der Routine das Carry-Flag gesetzt ist, heißt das für das Betriebssystem: Alles in Ordnung. Vielleicht baust Du aber noch eine Prüfsumme ein und genau die stimmt nicht, was darrauf schließen läßt, daß in dem EPROM mindestens ein Bit umgekippt ist. In diesem Fall sollte das Carry-Flag nicht gesetzt sein und das ROM ist damit tot. Selbstverständlich kann diese Initialisierungsroutine auch noch eine kleine Meldung ausgeben, im Beispiellisting wäre das *ScreenSwitch - ©1995 SCUG*, Bildschirmfarben oder Modus wechseln, die Nationalhymne spielen...

Weil der Arbeitsspeicher den ROMs dynamisch zur Verfügung gestellt wird, kann man nicht davon ausgehen, daß ein ROM immer denselben Speicherbereich zur Verfügung hat. Wenn vor AMSDOS, das normalerweise den Speicher ab &A700 belegt, noch andere ROMs initialisiert werden, verschiebt sich diese Adresse plötzlich nach unten. Deshalb wird bei jedem Aufruf eines RSX-Kommandos diese Adresse in IY übergeben, so daß die Routinen dann beispielsweise mit LD r,(IY+x) oder LD (IY+x),r darauf zugreifen können.

Das alles klingt jetzt vieleicht ein wenig kompliziert, aber ein Blick auf das Listing sorgt sicher für Klarheit. Das Beispielprogramm wurde für Maxam geschrieben. Es erzeugt eine Datei, die beispielsweise mit dem Softbrenner in die Inicron ROM-RAM-Box geladen werden kann.

Eine andere Möglichkeit wäre es, die Software in ein EPROM (Erasable



ROMs und ihre Einschaltmeldungen

Programmable Read Only Memory) zu brennen. Ein EPROM ist ein integrierter Schaltkreis, der RAM (Random Access Memory, Schreib/Lese-Speicher) und ROM in (Read Only Memory, Nurlese-Speicher) in sich vereinigt. Dabei erfolgt das Hineinschreiben von Daten mit einer höheren Spannung, der Programmierspannung, als der üblichen Betriebsspannung von +5 Volt Gleichspannung, was nur dann möglich ist, wenn die Speicherzellen des EPROMs leer oder gelöscht sind. Sind Daten im EPROM vorhanden, können sie nur noch gelesen, aber nicht neue hinzugefügt werden und zwar auch dann nicht, wenn noch Speicherzellen im EPROM frei sind. Das Löschen der Daten im EPROM erfolgt mit ultraviolettem Licht (UV), weshalb EPROMs auf der Gehäuseoberfläche ein Quarzglasfenster haben. Dieses muß mit einem UVundurchlässigem Aufkleber abgedeckt sein, wenn das EPROM Daten enthält, da das natürliche Sonnenlicht einen hohen UV-Anteil hat und bei längerem Einstrahlen Daten im EPROM gelöscht würden. Schon bei kurzer Sonneneinwirkung besteht die Gefahr, daß einzelne Bits umkippen.

Stephan Sommer/<dg>

```
Ready
|help
|HELP,n  zeigt die Befehle von ROM n

ROM  0:BASIC                  1.20 foreground
ROM  1:SB ROM                 1.02 back &9F81
ROM  2:TIME ROM+              1.10 back &9F85
ROM  3:PROTEXT                1.22 back &9FEC
ROM  4:MAXAM                  1.14 back &A1F0
ROM  5:RAM DISC               1.01 back &A2F4
ROM  6:ScreenSwitch           1.02 back &A2F8
ROM  7:XDDOS ROM              2.10 back &A2FC
ROM  8:UTOPIA                 1.25 back &AB00
Ready
|help,6

ROM  6:ScreenSwitch           1.02 back &A2F8
  50HZ                    60HZ
  75HZ
Ready
```

ScreenSwitch neben den anderen ROMs mit seinen Befehlen

```
                        ;ScreenSwitch ROM
                        ;(c)1995 SCUG

org &c000               ;Startadresse
limit &ffff             ;ROM darf hoechstens bis &FFFF gehen

txt_output equ &bb5a

write"c:scrswtch.rom"   ;Programmcode in SCRSWTCH.ROM speichern

db 1                    ;typ 1=Hintergrund-ROM
db 1,0,2                ;Version 1.02
dw namtab               ;zeiger auf namenstabelle
jp init                 ;sprungbefehl zur
initialisierungsroutine
jp cmd50hz              ;sprungbefehle zu den rsx-routinen
jp cmd60hz              ; /
jp cmd75hz              ;/
.namtab db"ScreenSwitc","h"+&80;Name des ROMs
db"50H","Z"+&80         ;namenstabelle
db"60H","Z"+&80         ; /
db"75H","Z"+&80         ;/
db 0                    ;nullbyte markiert tabellenende

.init push hl           ;hl sichern
ld hl,initmeld          ;hl auf die einschaltmeldung
ld b,29                 ;laenge der meldung in backcounter
.l1 ld a,(hl)           ;zeichen in hl
call txt_output         ;zeichen ausgeben
inc hl                  ;hl auf naechstes zeichen
djnz l1                 ;schleifenende
scf                     ;carry-flag setzen
pop hl                  ;hl wiederherstellen
ret                     ;routine ende
.initmeld db" ScreenSwitch - ",164,"1995 SCUG",10,10,13;Meldung

.cmd50hz ld bc,&bc04    ;bildwiederholfrequenz auf 50 hz stellen
out (c),c               ;          /
ld bc,&bd26             ;         /
out (c),c               ;        /
ld bc,&bd00             ;       /
out (c),c               ;      /
ld bc,&bc07             ;     /
out (c),c               ;    /
ld bc,&bd1e             ;   /
out (c),c               ;  /
ret                     ; /
                        ;/
.cmd60hz ld bc,&bc04    ;bildwiederholfrequenz auf 60 hz stellen
out (c),c               ;          /
ld bc,&bd1f             ;         /
out (c),c               ;        /
ld bc,&bc05             ;       /
out (c),c               ;      /
ld bc,&bd00             ;     /
out (c),c               ;    /
ld bc,&bc07             ;   /
out (c),c               ;  /
ld bc,&bd1b             ; /
out (c),c               ;/
ret                     ;routine beenden

.cmd75hz ld bc,&bc04    ;bildwiederholfrequenz auf 75 hz stellen
out (c),c               ;          /
ld bc,&bd1b             ;         /
out (c),c               ;        /
ld bc,&bc05             ;       /
out (c),c               ;      /
ld bc,&bd02             ;     /
out (c),c               ;    /
ld bc,&bc07             ;   /
out (c),c               ;  /
ld bc,&bd19             ; /
out (c),c               ;/
ret                     ;routine beenden
```