

Kaum zu glauben, aber die Assemblerecke kann bereits das erste kleine Jubiläum feiern: Seit einem halben Jahr konnten Sie hier regelmäßige Beiträge zur Programmierung Ihres Atari-Computers in Maschinensprache finden. Wie der Zuspruch aus dem Leserkreis zeigt, wird das wohl auch in Zukunft so bleiben. Die Themen werden wie bisher bunt gemischt bleiben, vom Scrolling bis zur Erweiterung des Betriebssystems, wobei die Anfänger unter den Maschinenprogrammieren nicht vergessen werden. Zu Anfang der Assemblerecke waren auch ACTION! Programme angekündigt, die bisher noch nicht berücksichtigt wurden. Aus gutem Grund, denn niemand weiß genau, wieviele ACTION!-Module ihren Weg nach Deutschland gefunden haben und wie viele Leute an ACTION!-Listings interessiert sind. Unser Vorschlag: Wer etwas über ACTION! lesen will, der zücke seinen Füller und schreibe uns ein paar Zeilen. Bei ausreichender Resonanz wird dann die (schon gut gefüllte) ACTION!-Trickkiste geöffnet.

Die Mini-RamDisk

Diesmal gibt es einen Leckerbissen speziell für alle Besitzer von XL-Computern mit 64K: Eine Mini-RamDisk, mit der sich der verborgene Speicherplatz parallel zum Betriebssystem nutzen lässt. Die Mini-RamDisk (MRD) wird nur mit dem Gerätenamen X: angesprochen und kann jeweils ein einziges File aufnehmen. Dieses File kann, wie Sie später aus der Speicherbelegung noch sehen werden, bis etwa 13 KByte lang sein. Das entspricht etwas mehr als 100 Disk-Sektoren. Der große Vorteil einer RAM-Disk ist die hohe Zugriffsgeschwindigkeit, die in unserem Fall bei etwa 5000 Zeichen pro Sekunde liegt (Atari Disk-Laufwerk: etwa 900 Zeichen/ Sek.). Wie können Sie nun ein solches Programm verwenden? In BASIC läßt sich damit ein Programm mit SAVE"X:" zwischenspeichern oder man kann auch mit LIST"X:" ganz bestimmte Programmteile in ein anderes Programm übernehmen. Sie können auch eine beliebige Hilfsdatei in der MRD anlegen (z. B. ein Hi-Res Bild). Mit anderen Worten: X: kann überall dort verwendet werden, wo D: zum Einsatz kam (Ausnahme: NOTE und POINT sind nicht implementiert).

Arbeitet mit Atari-Schreiber

Auch beim Programmieren in ACTION! oder Assembler ist die MRD eine große Hilfe. Sie können zum Beispiel ein File in der MRD ablegen, das die wichtigsten Einsprung- und Hardwareadressen enthält und mit INCLUDE"X:" ins Programm eingebunden wird. Dank Mini-RamDisk geht das ohne Diskzugriff in Sekundenschnelle vor sich. Ich war selbst überrascht, daß die MRD anstandslos mit dem Atari-Schreiber (ROM-Modul) zusammenarbeitet und sich z. B. ideal zum Einsetzen von Adressen in Briefen eignet.

Tipps zum Tippen

Für alle Nur-Benutzer ist das Listing 1 gedacht. Es handelt sich dabei um ein Basic-Programm, das das eigentliche AUTORUN.SYS File auf Diskette schreibt. Dabei ist folgende Vorgehensweise angebracht: Zuerst Listing 1 eintippen und davon eine Sicherheitskopie auf Diskette anfertigen. Dann eine Diskette mit DOS-II einlegen und das Programm mit RUN starten. Jetzt wird ein AUTORUN.SYS File auf der Disk angelegt. Achten Sie aber darauf, daß sich nicht schon ein solches auf der Disk befindet, es würde gelöscht werden! Sollten irgendwelche Unstimmigkeiten in den DATA-Zeilen auftreten, so wird Ihnen das mit Angabe der fehlerhaften Zeile mitgeteilt. Das Prinzip des Loaders wurde aus der amerikanischen Zeitschrift „Analog“ übernommen (Thanks, Tom). Jetzt booten Sie die Diskette mit dem AUTORUN.SYS-File neu (BASIC aktiv, d. h. nicht OPTION drücken) und haben somit bereits die Mini-Ramdisk installiert. Einen schnellen Test können Sie gleich mit dem Basic-Loader ausführen: Laden Sie diesen von Diskette und schreiben Sie ihn mit SAVE"X:" in die MRD. Geben sie anschließend NEW und LOAD"X:" ein, und schon ist das Programm wieder im Speicher.

Mini-RamDisk intern

Natürlich wäre die Assemblerecke nicht komplett, wenn wir nicht einen Blick in das Innenleben der MRD werfen würden. Das Programm belegt den Speicherplatz gleich anschließend ans DOS-II (ab \$1F00) und ist ca. 380 Bytes lang. Um ein Überschreiben durch das Anwenderprogramm zu verhindern, wird der LOMEM-Vektor (\$2E7, \$2E8) auf das Ende des MRD-Programmes versetzt. Dieser Zustand wird durch Einbinden der FMS-Initialisierung auch dann beibehalten, wenn RESET gedrückt wurde (siehe Assemblerlisting).

Das Mini-RamDisk Programm hat, obwohl es der Name vermuten lässt, nichts mit dem Diskettenbetriebssystem zu tun. Es arbeitet auf der Basis eines hardwareunabhängigen Gerätetreibers (beim Atari "Handler" genannt). Dieser Gerätetreiber muß nur einige wenige Grundfunktionen kennen. Darunter fallen Funktionen wie: ein Byte schreiben, ein Byte lesen, Gerät vorbereiten (öffnen) und eine Close-Funktion. Die Einsprungadressen der Gerätefunktionen werden in der sogenannten "Handlertabelle" zusammengefasst und deren Adresse wiederum in der Geräte-Tabelle HATABS eingetragen. Wenn nun z.B. in Basic ein Ausgabebefehl ausgeführt werden soll, so fließen die Daten vom Basic über das Central Input/Output-Utility (CIO) an den angesprochenen Gerätetreiber.

Der MRD-Gerätetreiber benutzt die Grundfunktionen, um die vom CIO gelieferten Daten im RAM "unter" dem Betriebssystem einzutragen. Wie Sie sicherlich wissen, kann bei XL(und XE-) Computern mit minimal 64K das Betriebssystem ROM durch RAM ersetzt werden, indem das Bit 0 der Adresse \$D301 auf Null gesetzt wird. Hierbei gibt es einige Besonderheiten zu berücksichtigen: Der normale Zeichensatz verschwindet mit dem Ausschalten des ROMs, so daß es im Interesse einer flimmerfreien Bildschirmdarstellung nötig ist, den Zeichensatz zuvor ins RAM zu kopieren. Weiterhin muß auch für die Interrupts vorgesorgt werden, deren Vektoren ebenfalls im ROM verewigt sind. Und schließlich ist noch zu beachten, daß der Bereich von \$D000 bis \$D7FF immer für Hardware I/O reserviert ist. Aus all diesen Überlegungen resultiert die Speicherbelegung nach Bild 1, die im MRD-Programm durch die MEMTAB Tabelle berücksichtigt wird.

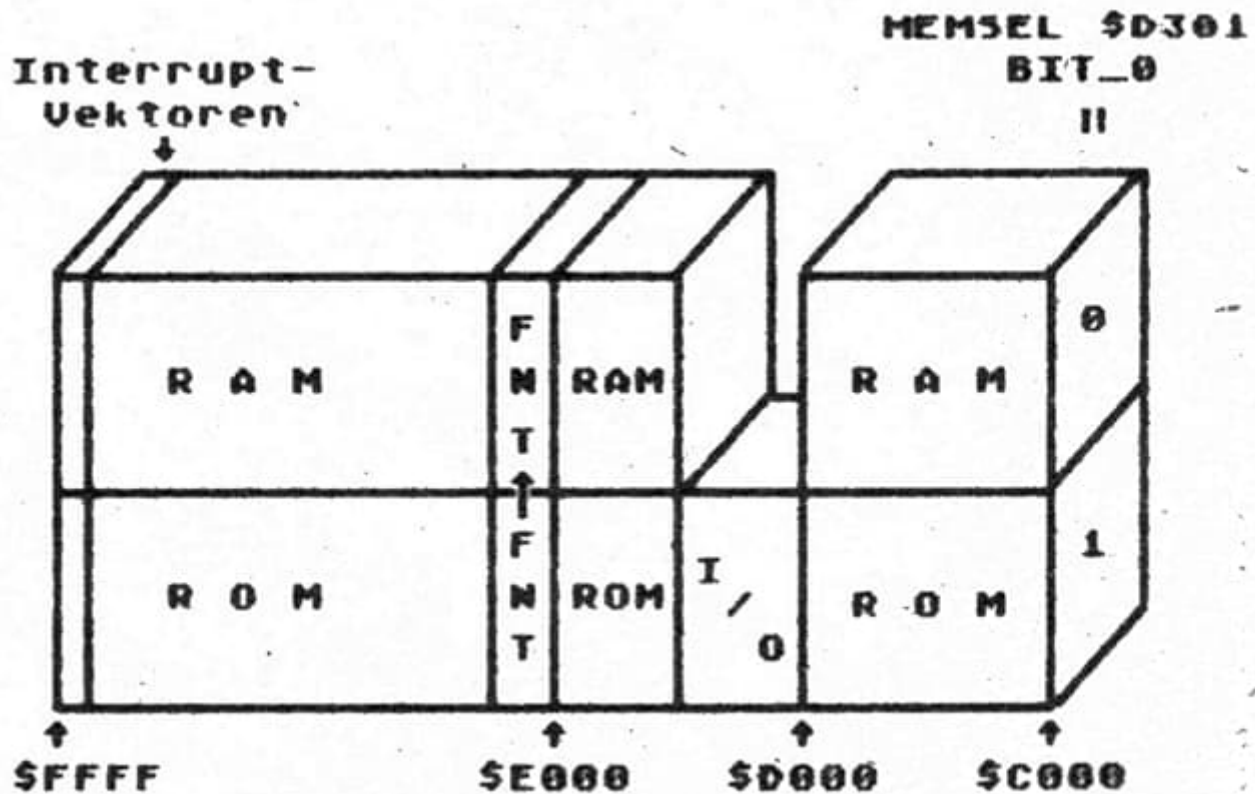


Bild 1: Speicherverteilung im XL-Computer

Für andere DOS-Versionen

Um das MRD-Programm mit anderen DOS-Versionen betreiben zu können, muß deren höchste Adresse als Programmbeginn des MRD-Programmes gewählt werden. Dies ist nur durch erneutes Assemblieren des Listings 2 mit geänderter Anfangsadresse möglich. Für DOS-XL (von OSS) verwenden Sie bitte *=\$2200, für DOS-III *=\$1DOO (mit File-Buffern).

Das Assemblerlisting wurde mit MAC/65 erstellt, kann aber auch mit der Editor/Assembler Cart-ridge assembliert werden. In der nächsten Assemblerecke dreht sich wieder alles um Scrolling. Im 2. Teil geht es um die horizontale Bildverschiebung. Dann sind auch wieder die Besitzer der älteren 400/ 800-Geräte mit von der Partie.

Peter Finzel

```

0100 ;*****
0110 ;
0120 ;      Mini-RAMDISK Handler
0130 ;      fuer XL-Computer
0140 ;      Peter Finzel      1985
0150 ;
0160 ;*****
0170 ;
0180 ;Hardware-Variable
0190 ;
=D40E      0200 NMIEEN      = $D40E
=D301      0210 MEMSEL     = $D301
=E000      0220 ZSATZ      = $E000      ROM-Zeichensatz
=FFFA      0230 NMIVVEC     = $FFFA      Hardware-Vektoren
=FFFE      0240 IRQVEC     = $FFFE
0250 ;
0260 ; O.S. Variable/Konstante.
0270 ;
=00E0      0280 FR1        = $E0        Zwischenspeicher
=000C      0290 DOSINI     = $0C        FMS Init.-Vektor
=02E7      0300 LOMEM      = $02E7
=031A      0310 HATABS     = $031A
=034A      0320 ICAUX1     = $034A      AUX1 Byte des IOCB#0.
0330 ;
=FFF0      0340 RDTOP      = $FFF0      Hoechste benutzbare Adresse
=0001      0350 STOK       = 1          Status ist O.K.
=0088      0360 STEOF      = $88        End-of-File Fehlercode
=00A9      0370 STFULL     = $A9        DISK-FULL Fehler!
=0008      0380 OPDOUT     = $08        Open-Output Anweisung
=0100      0390 HI         = $0100      Divisor fuer MSB
=00FF      0400 LD         = $FF        Maske fuer LSB
0410 ;
0420 ;
0430 ;*****
0440 ;MiniRamDisk-Handler wird vorbereitet
0450 ;*****
0460 ;
0000      0470             *= $1F00      ;ab DOS-Obergrenze
0480 ;
1F00 20EC1F 0490 START     JSR DISET      DOSINI einrichten
1F03 200920 0500             JSR XLINIT   Zusatz-RAM vorbereiten
1F06 A200    0510 DIEIN     LDX #0        Handler Tabelle nach
1F08 BD1A03 0520 WEITER     LDA HATABS,X   Gerateeintraege absuchen
1F0B F00A    0530             BEQ EINTRAG  leere Eintrag?->
1F0D C958    0540             CMP #'X     'X' schon vorhanden?
1F0F F024    0550             BEQ FERTIG   Ja, dann nicht nochmal -->
1F11 EB      0560             INX
1F12 EB      0570             INX
1F13 EB      0580             INX          ; naechsten Eintag
1F14 D0F2    0590             BNE WEITER   und weiter geht's-->
1F16 60      0600             RTS         Wie bitte? Nicht moeglich!!
0610 ;
1F17 A958    0620 EINTRAG   LDA #'X       MRD wird mit X:
1F19 9D1A03 0630             STA HATABS,X angesprochen
1F1C A936    0640             LDA #XTAB&LD Adresse der Handler-
1F1E 9D1B03 0650             STA HATABS+1,X tabelle in HATABS
1F21 A91F    0660             LDA #XTAB/HI eintragen
1F23 9D1C03 0670             STA HATABS+2,X
1F26 A900    0680             LDA #0       neues Ende von HATABS
1F28 9D1D03 0690             STA HATABS+3,X festlegen
0700 ;
1F2B A97D    0710             LDA #PGMEND&LD jetzt noch LOMEM
1F2D BDE702 0720             STA LOMEM     auf das Ende des
1F30 A920    0730             LDA #PGMEND/HI MiniRamDisk Pgms
1F32 BDEB02 0740             STA LOMEM+1   setzen
1F35 60      0750 FERTIG    RTS
0760 ;
0770 ;*****
0780 ;Die Handlertabelle...
0790 ;*****
0800 ;
1F36 471F    0810 XTAB      .WORD OPEN-1
1F38 521F    0820             .WORD CLOSE-1
1F3A A91F    0830             .WORD GET-1
1F3C 681F    0840             .WORD PUT-1
1F3E 441F    0850             .WORD STATUS-1
1F40 441F    0860             .WORD XIO-1
1F42 4C451F 0870             JMP INIT
0880 ;
0890 ;*****
0900 ;Leere Eintraege
0910 ;*****
1F45         0920 STATUS
1F45         0930 XIO
1F45 A001    0940 INIT      LDY #STOK      Status ist O.K.
1F47 60      0950             RTS

```

```

0960 ;
0970 ;*****
0980 ;Open-Routine
0990 ;*****

1000 ;
1F4B A900 1010 OPEN LDA #0 Zeiger auf Buffer-
1F4A BD6C20 1020 STA ZEIGER Anfang setzen
1F4D BD6D20 1030 STA ZEIGER+1
1F50 A001 1040 LDY #STOK kein Fehler moeglich!
1F52 60 1050 RTS
1060 ;
1070 ;*****
1080 ;CLOSE-Routine
1090 ;*****
1100 ;
1F53 BD4A03 1110 CLOSE LDA ICAUX1,X Wurde eine neue
1F56 2908 1120 AND #OPOUT Datei geschrieben?
1F58 F00C 1130 BEQ CLENDE Nein, nur gelesen-->
1140 ;
1F5A AD6C20 1150 LDA ZEIGER Zeiger als ENDE merken.
1F5D BD6E20 1160 STA RDEnde
1F60 AD6D20 1170 LDA ZEIGER+1 jetzt MSB
1F63 BD6F20 1180 STA RDEnde+1
1190 ;
1F66 A001 1200 CLENDE LDY #STOK Alles klar!
1F68 60 1210 RTS
1220 ;
1230 ;
1240 ;*****
1250 ;Ein Byte in MiniRD schreiben
1260 ;*****
1270 ;
1F69 48 1280 PUT PHA ;Datenbyte merken
1F6A AC6C20 1290 LDY ZEIGER Zeiger nach Zeropage
1F6D B4E0 1300 STY FR1
1310 ;
1320 ;MSB in physikalische Adresse umrechnen
1330 ;
1F6F AD6D20 1340 LDA ZEIGER+1 jetzt MSB
1F72 48 1350 PHA ;MSB-Zeiger
1F73 4A 1360 LSR A ;geteilt durch 4
1F74 4A 1370 LSR A
1F75 AA 1380 TAX ;als Index in Memtab
1F76 68 1390 PLA ;Zeiger
1F77 2903 1400 AND #3 Bit 1 & 2 maskieren
1F79 1D7020 1410 ORA MEMTAB,X physikalische Adresse
1F7C B5E1 1420 STA FR1+1
1430 ;
1440 ;Noch Platz in MiniRD?
1450 ;
1F7E C0F0 1460 CPY #RDTOP&LO MRD voll (LSB)
1F80 D004 1470 BNE PLATZ noch Platz vorhanden-->
1F82 C9FF 1480 CMP #RDTOP/Hi MSB noch pruefen
1F84 F020 1490 BEQ VOLL kein Platz mehr!!-->
1500 ;
1510 ;Datenbyte ins RAM eintragen
1520 ;
1F86 AD01D3 1530 PLATZ LDA MEMSEL auf Zusatzram schalten
1F89 29FE 1540 AND #FE Bit 0 gleich 0 waehlt RAM
1F8B BD01D3 1550 STA MEMSEL
1F8E 68 1560 PLA ;Datenbyte wieder holen
1F8F A000 1570 LDY #0
1F91 91E0 1580 STA (FR1),Y und in MRD schreiben
1F93 AD01D3 1590 LDA MEMSEL O.S. wieder ein
1F96 0901 1600 ORA #1 Bit 0 auf 1 waehlt ROM
1F98 BD01D3 1610 STA MEMSEL
1620 ;
1630 ;File-Zeiger auf naechstes freies Byte richten
1640 ;
1F9B EE6C20 1650 INC ZEIGER Zeiger weiter
1F9E D003 1660 BNE PUT1 MSB ist O.K.-->
1670 ;
1FA0 EE6D20 1680 INC ZEIGER+1 MSB Uebertrag
1690 ;
1FA3 A001 1700 PUT1 LDY #STOK keinerlei Fehler
1FA5 60 1710 RTS
1720 ;
1FA6 68 1730 VOLL PLA ; Datenbyte vom Stapel
1FA7 A0A9 1740 LDY #STFULL MRD ist voll
1FA9 60 1750 RTS

```

```

1760 ;
1770 ;*****
1780 ;Ein Byte aus MiniRD holen
1790 ;*****
1800 ;
1FAA AC6C20 1810 GET      LDY ZEIGER      Zeiger nach Zeropage
1FAD 84E0    1820      STY FR1
1FAF AD6D20 1830      LDA ZEIGER+1      jetzt MSB
1FB2 CC6E20 1840      CPY RDEND      letztes Byte im File?
1FB5 D005    1850      BNE NOCH      noch Bytes vorhanden-->
1FB7 CD6F20 1860      CMP RDEND+1    MSB noch pruefen
1FBA F02D    1870      BEQ LEER      nichts mehr da!-->
1FBC 4B      1880 NOCH     PHA          ;phys. Adresse ermitteln
1FBD 4A      1890      LSR A          ;geteilt durch 4
1FBE 4A      1900      LSR A
1FBF AA      1910      TAX          ;als Index in Memtab
1FC0 6B      1920      PLA          ;Zeiger
1FC1 2903    1930      AND #3        Bit 1 & 2 maskieren
1FC3 1D7020 1940      ORA MEMTAB,X    physikalische Adresse
1FC6 85E1    1950      STA FR1+1    in Zeropage
1FC8 AD01D3 1960      LDA MEMSEL      Zusatz-RAM ein
1FCB 29FE    1970      AND #$FE
1FCD 8D01D3 1980      STA MEMSEL
1FD0 A000    1990      LDY #0
1FD2 B1E0    2000      LDA (FR1),Y    Byte aus MRD holen
1FD4 4B      2010      PHA          ;Datenbyte merken
1FD5 AD01D3 2020      LDA MEMSEL      auf OS zurueckschalten
1FDB 0901    2030      ORA #1
1FDA 8D01D3 2040      STA MEMSEL
1FDD EE6C20 2050      INC ZEIGER      Zeiger weiter
1FE0 D003    2060      BNE GET1      MSB ist O.K.-->
1FE2 EE6D20 2070      INC ZEIGER+1    MSB Uebertrag
1FE5 6B      2080 GET1     PLA          ;Datenbyte
1FE6 A001    2090      LDY #STOK      keinerlei Fehler
1FEB 60      2100      RTS
2110 ;
1FE9 A0BB    2120 LEER     LDY #STEOF    MRD ist leer!
1FEB 60      2130      RTS
2140 ;
2150 ;*****
2160 ;Sicherung gegen SYSTEM RESET
2170 ;*****
1FEC A50D    2180 DISET     LDA DOSINI+1    wurde DOSINI schon
1FEE C920    2190      CMP #DIPGM/HI    berichtet?
1FF0 F010    2200      BEQ DISEND      ja, stimmt schon!-->
2210 ;

1FF2 8D0520 2220      STA DIPGM+2    sonst die Adresse in
1FF5 A50C    2230      LDA DOSINI      den JSR-Befehl ein-
1FF7 8D0420 2240      STA DIPGM+1    setzen!
1FFA A903    2250      LDA #DIPGM&LO    DOSINI auf neue
1FFC 850C    2260      STA DOSINI      Routine richten
1FFE A920    2270      LDA #DIPGM/HI
2000 850D    2280      STA DOSINI+1
2002 60      2290 DISEND     RTS
2300 ;
2310 ;modifizierte DOSINI-Routine
2320 ;
2003 204015 2330 DIPGM     JSR $1540      ;!! Hier wird DOSINI
2006 4C061F 2340      JMP DIEIN      ;eingesetzt!!
2350 ;
2360 ;
2370 ;*****
2380 ;Initialisierungsroutine fuer Zusatz-
2390 ;speicher $C000-$FFFF
2400 ;*****
2410 ;
2009 7B      2420 XLINIT     SEI          ;alle Interrupts
200A A900    2430      LDA #0          ;ausschalten
200C 8D0ED4 2440      STA NMIEIN
200F AD01D3 2450      LDA MEMSEL      Zusatz-Ram anwaehlen
2012 29FE    2460      AND #$FE
2014 8D01D3 2470      STA MEMSEL
2017 A96B    2480      LDA #INTEND&255    Interruptvektoren
2019 BDFAFF 2490      STA NMIVC      auf 'Dummy' RTI-
201C BDFEFF 2500      STA IRQVEC     Befehl richten
201F A920    2510      LDA #INTEND/256
2021 BDFBFF 2520      STA NMIVC+1
2024 BDFFFF 2530      STA IRQVEC+1

```

```

2540 ;
2550 ; ROM-Zeichensatz ins RAM kopieren
2560 ;
2027 A200      2570      LDX #0
2029 AD01D3    2580 ZCOPY LDA MEMSEL      jetzt wieder ROM
202C 0901      2590      ORA #1
202E BD01D3    2600      STA MEMSEL
2031 BD00E0    2610      LDA ZSATZ,X      vier Bytes Zeichensatz
2034 48        2620      PHA              ;auf den Stack
2035 BD00E1    2630      LDA ZSATZ+256,X
2038 48        2640      PHA
2039 BD00E2    2650      LDA ZSATZ+512,X
203C 48        2660      PHA
203D BD00E3    2670      LDA ZSATZ+768,X
2040 48        2680      PHA
2041 AD01D3    2690      LDA MEMSEL      RAM einschalten
2044 29FE      2700      AND #$FE
2046 BD01D3    2710      STA MEMSEL
2049 68        2720      PLA              ;vom Stack ins RAM
204A 9D00E3    2730      STA ZSATZ+768,X
204D 68        2740      PLA
204E 9D00E2    2750      STA ZSATZ+512,X
2051 68        2760      PLA
2052 9D00E1    2770      STA ZSATZ+256,X
2055 68        2780      PLA
2056 9D00E0    2790      STA ZSATZ,X
2059 E8        2800      INX
205A D0CD      2810      BNE ZCOPY      ;256 Durchlaeufer-->
2820 ;
205C AD01D3    2830      LDA MEMSEL      fertig, OS-ROM ein
205F 0901      2840      ORA #1
2061 BD01D3    2850      STA MEMSEL
2064 A940      2860      LDA #$40      NMI wieder ein
2066 BD0ED4    2870      STA NMEN
2069 58        2880      CLI              ;IRQ wieder an
206A 60        2890      RTS
2900 ;
2910 ;
206B 40        2920 INTEND RTI          ;fuer Interrupts in RAM-Pha:
2930 ;
2940 ;*****
2950 ;Interne Variablen
2960 ;*****
2970 ;

```

PAGE 6

```

206C 0000      2980 ZEIGER .WORD 0      Zeiger auf aktuelles Byte
206E 0000      2990 RDEnde .WORD 0      log. Endadresse
3000 ;
3010 ;*****
3020 ;Umrechnungstabelle zur Bestimmung der physikalisch
3030 ;Adresse ($C000 - $FFF0, mit Unterbrechungen) aus
3040 ;der logischen Position im File ($0000-$33F0)
3050 ;*****
3060 ;
2070 C0C4C8CC  3070 MEMTAB .BYTE $C0,$C4,$C8,$CC
2074 D8DCE4E8  3080      .BYTE $D8,$DC,$E4,$E8
2078 ECF0F4F8  3090      .BYTE $EC,$F0,$F4,$F8
207C FC        3100      .BYTE $FC
3110 ;
3120 ;
=207D        3130 PGMEND = *          ;LOMEM-Grenze liegt hier
3140 ;
3150 ;RUN-Adresse zum selbstaendigen Programmstart
3160 ;
207D        3170      *= $02E0
02E0 001F      3180      .WORD START
02E2        3190      .END

```

ASSEMBLY ERRORS: 0 2012B BYTES FREE

```

10 REM ** AUTORUN.SYS GENERIERUNG **
20 REM ** P. FINZEL 1985
100 DIM D$(80):OPEN #1,8,0,"D:AUTORUN.
SYS"
110 ? "AUTORUN.SYS wird generiert...":
?
120 READ D$:READ P:IF D$="*" THEN 200
130 S=0: ? "*";
140 FOR I=1 TO LEN(D$) STEP 2
150 H=ASC(D$(I,I))-48:L=ASC(D$(I+1,I+1))-48
160 D=(H-(H>9)*7)*16+L-(L>9)*7:S=S+D:P
UT #1,D
170 NEXT I:IF S=P THEN 120
180 ? :? :? "DATENFEHLER IN ZEILE ";PE
EK(183)+PEEK(184)*256:CLOSE #1:STOP
200 REM * FILE SCHLIESSEN
210 CLOSE #1
220 ? :? :? "AUTORUN.SYS ordnungsgemae
ss erzeugt!"
230 END
1100 DATA FFFF001FFB1F20EC1F200920A200
BD1A03F00AC958F024E8EB,2854
1110 DATA E8D0F260A9589D1A03A9369D1B03
A91F9D1C03A9009D1D03A9,2546
1120 DATA 7D8DE702A9208DE80260471F521F
A91F681F441F441F4C451F,2095
1130 DATA A00160A9008D6C208D6D20A00160
BD4A032908F00CAD6C208D,2267
1140 DATA 6E20AD6D208D6F20A0016048AC6C
2084E0AD6D20484A4AAA68,2545
1150 DATA 29031D702085E1C0F0D004C9FFF0
20AD01D329FEBD01D368A0,3244
1160 DATA 0091E0AD01D309018D01D3EE6C20
D003EE6D20A0016068A0A9,2775
1170 DATA 60AC6C2084E0AD6D20CC6E20D005
CD6F20F02D484A4AAA6829,2805
1180 DATA 031D702085E1AD01D329FEBD01D3
A000B1E048AD01D309018D,2736
1190 DATA 01D3EE6C20D003EE6D2068A00160
A08B60A50DC920F0108D05,2746
1200 DATA 20A50C8D0420A903FC1F7C20850C
A920850D602040154C061F,1815
1210 DATA 78A9008D0ED4AD01D329FEBD01D3
A96B8DFAFF8DFEFA9208D,3603
1220 DATA FBFF8DFFFA200AD01D309018D01
D3BD00E048BD00E148BD00,3227
1230 DATA E248BD00E348AD01D329FEBD01D3
689D00E3689D00E2689D00,3055
1240 DATA E1689D00E0E8D0CDAD01D309018D
01D3A9408D0ED458604000,2951
1250 DATA 000000C0C4C8CCD8DCE4E8ECF0F4
F8FCE002E102001F,3392
1260 DATA *,0

```



```

10 REM ** AUTORUN.SYS GENERIERUNG *
20 REM ** P. FINZEL 1985
100 DIM D$(80):OPEN #1,8,0,"D:AUTORUN.SYS"
110 ? "AUTORUN.SYS WIRD GENERIERT...":?
120 READ D$:READ P:IF D$="*" THEN 200
130 S=0: ? "*";
140 FOR I=1 TO LEN(D$) STEP 2
150 H=ASC(D$(I,I))-48:L=ASC(D$(I+1,I+1))-48
160 D=(H-(H>9)*7)*16+L-(L>9)*7:S=S+D:PUT #1,D
170 NEXT I:IF S=P THEN 120
180 ?:: ? "DATENFEHLER IN ZEILE ";PEEK(183)+PEEK(184)*256:CLOSE #1:STOP
200 REM * FILE SCHLIESSEN
210 CLOSE #1
220 ? : ? : ? "AUTORUN.SYS ORDNUNGSGEMAE55 ERZEUGT!"
230 END

1100 DATA FFFF001FFB1F20EC1F200920A200BD1A03F00AC958F024E8E8,2854
1110 DATA E8D0F260A9589D1A03A9369D1B03A91F9D1C03A9009D1D03A9,2546
1120 DATA 7D8DE702A9208DE80260471F521FA91F681F441F441F4C451F,2095
1130 DATA A00160A9008D6C208D6D20A00160BD4A032908F00CAD6C208D,2267
1140 DATA 6E20AD6D208D6F20A0016048AC6C2084E0AD6D20484A4AAA68,2545
1150 DATA 29031D702085E1C0F0D004C9FFF020AD01D329FE8D01D368A0,3244
1160 DATA 0091E0AD01D309018D01D3EE6C20D003EE6D20A0016068A0A9,2775
1170 DATA 60AC6C2084E0AD6D20CC6E20D005CD6F20F02D484A4AAA6829,2805
1180 DATA 031D702085E1AD01D329FE8D01D3A000B1E048AD01D309018D,2736
1190 DATA 01D3EE6C20D003EE6D2068A00160A08860A50DC920F0108D05,2746
1200 DATA 20A50C8D0420A903FC1F7C20850CA920850D602040154C061F,1815
1210 DATA 78A9008D0ED4AD01D329FE8D01D3A96B8DFAFF8DFEFA9208D,3603
1220 DATA FBFF8DFFFA200AD01D309018D01D3BD00E048BD00E148BD00,3227
1230 DATA E248BD00E348AD01D329FE8D01D3689D00E3689D00E2689D00,3055
1240 DATA E1689D00E0E8D0CDAD01D309018D01D3A9408D0ED458604000,2951
1250 DATA 000000C0C4C8CCD8DCE4E8ECF0F4F8FCE002E102001F,3392
1260 DATA *,0

```