PC-NEWS

Das offizielle Mitteilungsblatt
des
PCC-TGM

(Personal Computer Club - Technologisches Gewerbe-Museum)

ASSEMBLER

Hilferuf !!

Nach einigen Jahren der Satzherstellung der PC-News durch Walter Riemer mittels Ventura-Publishers ist es dem genannten "Setzer" wegen stark gestiegener beruflicher wie auch privater Inanspruchnahme nicht mehr möglich, den Satz herzustellen. Das Setzen von Originalbeiträgen anhand übermittelter ASCII-Dateien nimmt, speziell wenn Tabellen, Listen und Grafiken enthalten sind, viel Zeit in Anspruch, und diese Zeit muß noch dazu termingemäß zur Verfügung stehen.

Gesucht wird also ein Mitglied, das diese Arbeit mittels Ventura oder auch einem anderen DTP-Programm gerne übernehmen will und auch die benötigte Zeit dafür aufbringen kann.

Geborene Zeitungsherausgeber, Redakteure und Setzer! Bitte meldet Euch beim Klub!

Näheres zu dieser Zeitschrift:

Bezugsbedingungen: Einzelheft öS 50,-, für Mitglieder des PCC-TGM im Mitgliedsbeitrag enthalten.

Impressum: Medieninhaber: PCC-TGM (Personal-Computer-Club-TGM), Wexstraße 21, Postfach 59, 1202 Wien.

Anrufbeantworter: (0222)/35 23 980 (Herr **Leeb**): Fr: 9.00-12.00h

Mailbox: (0222)/602 10 36 (8-N-1)

BTX: 9122**22584**

TELEBOX: RA2 FIALA.

BTX: Clubseite *5645#. Meinungsaustausch über *35703570##1 E.R.D.E. Kommunikation über *941# für Absenden und *930# für Empfangen von Nachrichten.

Grundlegende Richtung: Auf Anwendungen im Unterricht bezogene Informationen über Personal-Computer-Systeme. Berichte über Veranstaltungen des Vereins. Beratung der Vereinsmitglieder gemäß den Statuten des PCC-TGM.

Layout und Satz: Walter Riemer, Rosengasse 9, 2102 Bisamberg sowie Franz Fiala.

Erscheinungsort: Wien

Redaktion und für den Inhalt verantwortlich: Franz Fiala, Siccardsburggasse 4/1/22, 1100 Wien.

Liebe Clubmitglieder!

Diese Ausgabe der PC-NEWS ist dem Thema 'wie-formuliere-ich-in-Assembler' gewidmet. Sie finden weiter hinten einen möglichst genauen Adressplan eines kompatiblen PC und einige dazu passende Beiträge in Assembler.

Neben den Assemblerbeiträgen finden Sie eine Logo-Writer-Beschreibung und eine Open-Access-Beschreibung von Kollegen Neidhart/Spittal und eine Turbo-Pascal-Beschreibung und vielerlei nützliche Tips von Kollegen Melchart. Für beides dankt die Redaktion; hätten wir doch nur mehr dieser Beiträge.

Daß sich das Layout schon wieder ändert, ist auf chronischen Zeitmangel der Gestalter zurückzuführen. Wie sie aus Einschaltung auf der vorigen Seite ersehen können, suchen wir in dieser Richtung eine Hilfe zur Gestaltung der PC-NEWS.

Clublieferanten

In den vergangenen Jahren sind günstige Bezugsquellen für Clubmitglieder aus verschiedenen Gründen immer wieder versiegt. Seit der letzten Ausgabe der PC-NEWS haben wir aber wiederholt Rückmeldung erhalten, daß die Sonderpreisliste von EXCON für den PCC wirklich günstige Angebote enthält. Viele Kollegen im TGM haben positiv über Einkäufe bei Familie Hanisch (Herr Hanisch ist ehemaliger TGM-Absolvent) berichtet.

Clubbetrieb

Nachdem Arbeiten für den Club immer mehr Routinecharakter annehmen, ist sie als Freizeitbeschäftigung für Vorstandsmitglieder immer weniger tragbar. Für administrative Angelegenheiten ist seit kurzem Frau Jelinek im Clubbüro beschäftigt. Daher kann sich Herr Leeb, eher den technischen Fragestellungen widmen. Er hat auch die ersten BTX-Seiten des PCC erstellt. Da er aber das BTX nur so nebenbei betreuen kann, geht die Informationseingabe ins BTX nur allmählich vor sich.

Wir bemühen uns um Arbeitsteilung. BTX eignet sich gut, auch einmal von Nicht-Wienern betreut zu werden. Unser aktives Kärntner PCC-Team rund um Kollegen Schlatte wird sich bis nach den Ferien mit den BTX-Gegebenheiten auseinandersetzen und Eingabearbeiten übernehemen.

Preisausschreiben

Last-Call-for-Papers: Durch die um Monate verzögerte Aussendung der NEWS-16 (4.Heft 1989) wurde das Preisausschreiben mit Termin Ostern 1989 nur von einigen wenigen als relevant eingestuft, wir haben daher, gemessen an der Mitgliederzahl, nur wenige Einsender. Der Termin für die Einsendung wird daher auf Ende August 1990 verlegt und wir bitten unsere Mitglieder gemäß den Ausschreibungsbedingungen 'Programme für den Unterricht' einzureichen. Bei den Preisen sind wir wegen der geringen Teilnehmerzahl noch unschlüssig, stellen uns aber von, daß der jeweils erste Preis umso hübscher ausfällt, als es der Zahl der Einsendungen entspricht aus der er ausgewählt wurde. Zudem soll auf jeden Einsender ein Gewinn fallen.

BTX

In der Clubgründungsphase gab es im Mitgliederverzeichnis eine eigene Spalte für eine BTX-Nummer. Da diese aber nur bei einigen wenigen Mitgliedern ausgefüllt wurde, ist sie bei der ersten Revision des Verzeichnisses verschwunden.

PCC-TGM und BTX

Wie bereits in der letzten Aussendung zu lesen war, hat der Vorstand beschlossen, das BTX-System zu unterstützen und den Mitgliedern zu empfehlen. In der ersten Phase (zu Beginn des Jahres 1990) wurden einige Erfahrungen gesammelt. Im Frühjahr erhielten alle Mitglieder den BTX-Führer von H. Maurer und wurden zur Teilnahme am BTX-System eingeladen.

Hier noch einmal die wichtigsten Argumente in Kürze:

- Seit BTX mit einem *PC* betrieben werden kann, ist das System für den Club besonders attraktiv.
- An Gebühren fällt nur die einmalige *Einschreibegebühr* von 400 Schilling und während des Betriebes die *Telefonortsgebühr* (40 Schilling pro Stunde) aus ganz Österreich an. BTX ist damit das billigste flächendeckende Medium zur Datenfernverarbeitung.
- Für den PCC ist der *Mitteilungsdienst* und die Möglichkeit, über aktuelle *Angebote* sehr rasch zu informieren, besonders interessant.
- Außer dem österreichischen Angebot können auch die BTX-Netze der Bundesrepublik Deutschland, der Schweiz und von Luxemburg zum *Ortstarif* benutzt werden.
- Für die Verbindung PC - Telefonleitung ist ein Modem erforderlich; geeignet ist sowohl das Post-Modem ("BAG1A" um 70 Schilling Mietgebühr pro Monat) wie auch ein Modem aus unserer Sammelbestellung (siehe auch "*Modem-Aktion*").
- In Wien, Graz, Innsbruck, Klagenfurt und Salzburg wurden weitere Telefonnummern eingerichtet, über die BTX auch mit bis zu *2400 Baud* benutzt werden kann. Da diese Nummern aber keine Sondernummern sind, gelten die normalen Telefongebühren (das heißt, der Ortstarif gilt nur in jeweiligen Ortsnetz samt Umgebung).
- Bei weiterhin anhaltendem Interesse wollen wir über BTX auch interessante Public-domain *Programme* oder Beschreibungen verteilen.

Wenn es mit der Anmeldung *Probleme* geben sollte oder wenn Sie Fragen haben, schreiben Sie mir bitte - am besten per BTX. Bei Problemen werde ich gerne mit den zuständigen Stellen der Post Kontakt aufnehmen.

Meine BTX-Nummer: 912 213 458, meine Adresse: Martin Weissenböck, Gatterburggasse 7, A-1190 Wien.

Sie finden in der folgenden Liste alle Clubmitglieder, die bereits über BTX zu erreichen sind. Noch eine Bitte: wenn Sie BTX-Teilnemer geworden sind, senden Sie bitte eine Mitteilungsseite an den Club: wählen Sie *941#, danach 912 222 584 und schreiben Sie uns, daß Sie neu im System sind. Ihre BTX-Nummer wird dabei automatisch übertragen.

Die Clubmitglieder im BTX:

siehe Seite 2

PCC-Modemaktion

Zur Förderung der BTX-Aktivitäten wurde vom PCC-TGM gemeinsam mit der ADIM, der Arbeitsgemeinschaft für Didaktik, Informatik und Mikroelektronik, eine gemeinsame Bestellaktion für Modems organisiert. Die erste Lieferung wurde mit April an alle Interessenten verschickt.

Die Zusammenarbeit mit der Lieferfirma ist sehr gut. So werden wir jedenfalls über neue Firmwareversionen informiert und können auch wenn nötig bei wesentlichen Verbesserungen den Austausch der EPROMs vorschlagen.

Wegen des großen Interesses und auch wegen neuer Entwicklungen wurde die Aktion jetzt verlängert.

Das Modem Discovery 2400A:

Das **Discovery 2400A** ist ein Hochgeschwindigkeitsmodem mit automatischer Wahl und automatischer Antwort, entwickelt für die Verwendung mit IBM PC/XT/AT-Rechnern zur zuverlässigen Datenübertragen bei Geschwindigkeiten von 0-300, 1200, 1200/75 und 2400 Bit pro Sekunde mit den Protokollen Bell 103/212A, CCITT V.21, V.22, V.22bis und V.23. Das Protokoll V.23 ist für den österreichweiten Zugang zu *BTX* mit 1200/75 Baud notwendig, ferner für die BTX-Netze Deutschlands, der Schweiz und Luxemburgs.

Dieses Modem kann sowohl an Wählleitungen wie auch an Standleitungen angeschlossen werden. Es ist für die asynchrone und die synchrone Datenübertragung geeignet.

Das **Discovery 2400A** hat einen eingebauten nichtflüchtigen Speicher, der die Konfigurationsparamter und 10 Telefonnummern für den späteren Gebrauch speichert.

Der "Voice-Data"-Schalter an der Vorderseite erlaubt die Umschaltung zwischen Sprache und Datenübertragung, ohne die aufgebaute Verbindung zu unterbrechen.

Natürlich ist das Modem mit dem (erweiterten) AT-Befehlssatz ausgerüstet und arbeitet deshalb mit den meisten bekannten Kommunikationsprogrammen zusammen.

Die technischen Daten in Kurzform:

Kompatibilität	0-300, 1200, 2400 und 1200/75 Baud Bell 103/212A, CCITT V.21, V.22, V.22bis, V.23
Betriebsarten	Automatische Wahl, automatische Antwort, vollduplex, halbduplex. Standleitungen und Wählleitungen
Verbindungsaufbau	Tonfrequenzwahl und Impulswahl
Schnittstelle	seriell, RS-232C (V.24), mit 25poliger Buchse
Telefonanschluß	Zwei amerikanische Buchsen (RJ-11 modular) für die Telefonleitung und den Fernsprecher
Lautsprecher	Lautstärke über Hard- und Software einstellbar
Datenformat	Seriell, binär, asynchron, synchron
Empfindlichkeit	-45 dB
Übertragungspegel	-11 dB
Wählvorgang	Wählton, Besetztzeichen, Antwortton, Warten auf Pause, Warten auf zweiten Wählton
Diagnose	Lokale analoge Schleife, lokale analoge Schleife mit Selbsttest, entfernte digitale Schleife, entfernte digitale Schleife mit Selbsttest, lokale digitale Schleife, programmierbarer Zeitgeber für Testzwecke
Netzgerät	220 V Wechselspannung
Gehäuse	Profiliertes Gehäuse aus Aluminiumlegierung
Abmessungen	14,6 cm x 25.4 cm x 3,6 cm, 865 Gramm
Lieferumfang	Discovery 2400A Modem, Netzgerät, amerikanisches Telefonkabel, Benutzerhandbuch.
Zulassung	Zugelassen von der Federal Communications Commission (FCC) gemäß Teil 15 und Teil 68 für den direkten Anschluß an Computer und Telefonsysteme in den Vereinigten Staaten. Leider von der österreichischen Post (noch?) nicht zugelassen.
Softwarekompatibilität	Decodix, DataTalk, Bitcom, CrossTalk, SmartCom II, Carbon Copy, Symphony, Hotline, Remote, Metro, ProComm, Qmodem,

Mirror, PC Talk, Access, Mite, Easylink, CompuServe Videx, Relay und viele andere.

Nachdem die erste Bestellaktion fertig war, wurde von Lieferfirma aus Taiwan ein neuer Typ vorgestellt:

Das **Discovery 2400AM** hat alle Eigenschaften des Modells 2400A, wie sie in der obenstehenden Liste zusammengestellt sind und unterstützt zusätzlich auch das Microcom Networking Protokoll (MNP) der Klassen 4 und 5. Das heißt, daß die Daten bei der Übertragung durch die Modem-Hardware komprimiert und auf ihre Korrektheit überprüft werden. Im Durchschnitt bedeutet dies eine Verdopplung der Übertragungsrate.

Natürlich setzt dies entsprechende Modems an beiden Enden der Übertragungsstrecke voraus; MNP 5 Modems werden von der *Radio Austria* und von den schnellen *BTX*-Anschlüssen verwendet. Bei der BTX-Übertragung treten bei dieser hohen Geschwindigkeit noch Fehler auf, die aber nicht im Modem liegen.

Weitere technische Daten:

Fehlerkorrektur	MNP Klasse 5 Datenkompressionsprogramm
Abmessungen	14,6 cm x 25.4 cm x 3,6 cm, 990 Gramm

Modembestellung:

Die Typen 2400A und 2400AM können bei der ADIM bestellt werden (bitte nicht beim PCC direkt bestellen).

Preise:

* ..... Discovery 2400A ..... 3.048,- inkl. MWSt.
* ..... Discovery 2400A (Vorführmodell) ..... 2.808,- inkl. MWSt.
* ..... Discovery 2400AM ..... 3.540,- inkl. MWSt.

Vorauszahlung:

* ..... Bitte überweisen Sie als Anzahlung 1.500 Schilling an die ADIM auf das Konto 7254.969 bei der PSK (Bankleitzahl 60.000).
* ..... Geben Sie, ob Sie die Type 2400A, 2400A (Vorführgerät) oder 2400AM wünschen.
* ..... Bitte vergessen Sie nicht Ihre Adresse bei der Überweisung.
* ..... Auf Wunsch werden auch alle Verbingskabel besorgt; bitte geben Sie dies ggf. auch an.

Einige Geräte sind lagernd; wenn Modems neu bestellt werden müssen, beträgt die Wartezeit im Durchschnitt vier Wochen.

Tauschaktion:

Wer bei der ersten Bestellung ein Modem Typ 2400A gekauft hat und statt dessen jetzt die Type 2400AM verwenden möchte, möge sich an die ADIM, Postfach 23, A-1191 Wien wenden. Wir werden jene Interessenten, die ein Vorführmodell bestellt haben und jene, die ihr 2400A umtauschen wollen, zusammenbringen: für Interessenten am Typ 2400A verringert sich der Preis auf 2.808 Schilling; die Aufzahlung auf den Typ 2400AM beträgt somit 732 Schilling. Die Interessenten für einen Umtausch bzw. für Vorführgeräte werden in der Reihenfolge des Eintreffens vorgemerkt.

Weitere Kommunikationsprodukte:

Die Firma Datatronics, die die Discovery-Modems herstellt, bietet noch eine Reihe weiterer interessanten Typen an. Derzeit testen wir:

Das **Discovery 2448P**, ein portables externes Modem für bis zu 2400 Baud. Die Größe entspricht etwa einer Zigarrettenschachtel. Der V.23-Modus und die synchrone Datenübertragung werden nicht unterstützt, aber sonst verhält sich das Modem wie sein großer Bruder 2400A. Zusätzlich können mit dem 2448P auch *Telefaxe* mit 4800 Baud abgesandt werden. Die Vorlagen dazu müssen entweder als Text vorliegen oder können auch mit einem Scanner eingespielt werden. Leider können Telefaxe nicht empfangen werden. Das Modem dürfte besonders interessant für "unterwegs" sein. Voraussichtlicher Clubpreis (inklusive Faxsoftware und Mehrwertsteuer): 2.844 Schilling.

Die **Faxkarte 9600F** verwandelt den PC in ein Fax-Gerät. Telefaxe können an Gruppe 3-Faxgeräte gesandt werden und auch von diesen empfangen werden. Auch hier müssen die Vorlagen als Text oder gescanntes Bild vorliegen. Voraussichtlicher Clubpreis (inklusive Faxsoftware und Mehrwertsteuer): 3.840 Schilling.

Wenn Sie an einem dieser Modell Interesse haben, schreiben Sie bitte schon jetzt an die ADIM; wir werden Sie über die Testergebnisse und über Liefertermine informieren.

Die Entwicklung der letzten Monate läßt uns vermuten, daß wir diese Spalte bald wiedereinführen sollten. In der folgenden Tabelle finden Sie alle Mitglieder, die uns ihre BTX-Nummer mitgeteilt haben (BTX-Nummer/Mitgliedsnummer/Name). Für das Aktualisieren der Liste benötigen wir unbedingt Ihre BTX-Nummer.

BTX-Nummer	MNum	Name
221345801		ADIM-Wien
912218106		ADIM-Wien
912415295	1353	Anderle
912615360	733	Brenner
912115060		BRG-Bruck
912218753	104	Bruckner
912216428	551	Callsen-Rauer
	848	Cernusca
912218249	47	Chloupek
912218558	646	Deutsch
912216439	139	Eckl
912218431	1039	Eisenzopf
915520719	725	Feurstein
912218242	77	Fiala
912214463	672	Gassner
	518	Gotschim
	82	Gottfried
912218682	952	Hafenschar
916210260	425	Hasenburger
912218893	506	Herzog
912218898	1097	Hintenaus
911210196		Hirschmann*(Post-Lehrwerkstätte)
912214406	590	Hirz
912916298		HTBLA-Hollabrunn
912213510		HTBLA-Wien-1
912214660		HTBLA-Wien-1, Direktion
912211581		HTBLA-WIEN-1,Abt.N
912615243		HTBLuVA-Wr.Neustadt
912615244		HTBLuVA-Wr.Neustadt
911219987		HTL-Hollabrunn
912217106		HTL-Mödling
911219953		HTL-St.Pölten
912217641		HTL-Wien-22
912218795		HTL-Wien-3 Ungarg.
912215359	581	Hummer
912218465	94	Jordis
917415021	302	Junker
912218694	317	Kliemstein
	453	Klinsky
912218880	1440	Kolacek
912216420	132	König
914210137	1260	Lindner
912218740	194	Lirnberger
912216391	599	Mandl
	72	Martinek
912218220	26	Mayer
912222064		MCCA
912218703	50	Navratil
914710016	311	Neidhart*(BRG-Spittal)
912222588	1161	Neufingerl
912218527	27	Nitsche
912218705	652	Obdrzalek
915210681	830	Oppl

```
912218759     51    Ostermaier
912218336    785    Pany
912222584           PCCTGM *5645#
912218557    121    Reiermann
912211090      1    Reiter
912216422     65    Riemer
912216436    565    Salkovic
915510761    606    Salzmann
912218218     80    Scharl
913110525    533    Scheiber ADIM-Graz
912218862   1372    Scheuer
915210650    383    Schlager
914210023     92    Schlatte
912218432    644    Schleidt-Schuller
912212040    701    Schlögl
912218702    199    Schneeweis
912218930    236    Schwarz H.

912218281   1206    Sicher
912215400    203    Sigart
912218720    173    Skriwanek
912218731    305    Sokol
914210158    102    Sorko
912218716      7    Stani
912216435   1261    Steiner
912218709     20    Streisselberger
916211421    379    Trebuch
912213458    307    Weissenböck
912218769    190    Weltsch
             326    Widder
912213499     29    Winkler
912216313    801    Wurm
912715326    149    Zehetner
912216424    118    Zelinka
```

Clubdisketten

Den Text der vorliegenden PC-NEWS und alle dazugehörigen Programme sowie auch einen Teil des oben angegebenen Adressplans finden Sie auf der Diskette TGM-136.

Weitere Disketten haben wir in unser Verzeichnis aufgenommen:

TGM-136: PC-NEWS-18

```
NEWS17   TXT    178176 05-22-90   7:45a
NEWS17D  DFV      1024 05-22-90   7:45a
N17PGM   ARC     67409 05-22-90   7:57a
BIOS     TXT
AFD      DFV
AFD      TXT
BIOS     DFV
LPTXCHG  ASM
INTPRUEF ASM
INTPRUEF COM
INT_TAB  DAT
SPOOLER  ASM
SPOOLER  EXE
SKC      ASM
SKC      COM
SKC0     ASM
```

```
AUTOASK   COM
AUTOASK   ASM
```

TGM-137: Lohnsteuerberechnungsprogramm

```
FCG        BAT       766 11-24-88   12:27p
HINWEISE   BAT      1592 10-26-88    1:01p
LST        EXE    188912 11-13-88    7:10p
LSTARIFA   DBF       158 10-25-88    2:49p
LSTARIFN   DBF       128 10-25-88    2:49p
LSTNAM     DBF      1186 04-01-90    8:24p
```

TGM-138: DEMO-DISK, LAB-WINDOWS, National Instruments, (HD)

```
SETUP      BAT    SETUP      BAT        337 11-14-89   10:08a
UNPACK     EXE     38951 11-14-89   12:07p
LWDEMO     ZIP    895222 11-17-89    5:35p
```

TGM-139: PSPICE (Dokumentation, ca. 40 Seiten beim Club) (HD)

Diese PSPICE-Version ist eine Demoversion, die aber für Unterrichtszwecke ausreichend ist, die Dokumentation ist eine zum Erlernen geeignete

```
PS-OVL     ARC    395766 05-22-89    9:12p
PS-PARTS   ARC    142460 05-22-89    9:12p
PS-PROBE   ARC    202422 05-22-89    9:13p
PSPICE     BAT        52 06-22-87    7:31a
EXAMPLE1   CIR     10287 05-13-89    6:28p
EXAMPLE1   DAT      8092 05-15-89   12:26p
CSHELL     DOC     24911 10-24-88    8:33a
INSTALL    DOC      4029 11-01-88    9:58a
README     DOC     15443 01-28-89   10:07a
PS         EXE     15824 01-28-89   11:13a
PSPICE1    EXE    213232 04-05-89    9:03p
PSPICE     HLP     30699 12-16-88    2:45p
BIPOLAR    LIB      3297 11-02-88   10:10a
DIODE      LIB      1613 11-02-88   10:04a
JFET       LIB       960 11-02-88   10:15a
LINEAR     LIB      5031 11-02-88   10:12a
MAGNETIC   LIB      2636 11-01-88    8:52a
NOM        LIB      1087 11-01-88    8:45a
PWRMOS     LIB      3535 11-02-88   10:14a
PSPICE     NDX      2192 12-16-88    2:46p
EXAMPLE1   OUT     22380 05-15-89   12:26p
CONFIG     SYS        22 12-15-87   10:47a
PKARC      COM     19573 04-27-87   12:00a
PKXARC     COM     12242 04-27-87   12:00a
```

Farben und Notizdateigröße im SideKick einfach und schnell verändern

Walter Riemer, TGM (Diskette TGM-136)

Der folgende Bericht bezieht sich auf SideKick, Version 1.56A, also die "Urversion" des SideKick. Der Autor benützt insbesondere auf seinem Laptop nach wie vor diesen SideKick. Der Laptop wird fallweise mit externem Monitor betrieben, dann wieder mit dem eingebauten LCD-Display.

1. Side-Kick-Farben

Das LCD-Display ist von Natur aus Invers (dunkle Buchstaben auf hellem Hintergrund), für die Monitore gilt meistens das Umgekehrte. Es ist inzwischen allgemein bekannt, daß dunkle Schrift auf hellem Untergrund besser und augenschonender lesbar ist, als die früher aus auf hellem Darstellung Hell auf Dunkel; deswegen sind moderne Monocolor-Monitore mit einem Inversschalter ausgestattet, sodaß man bei jedem Programm eine "Hell auf Dunkel"-Darstellung am Monitor selbst wählen kann. SideKick bietet die Möglichkeit, die Farben mittels Installationsprogramms SKINST einzustellen, jedoch ist dies umständlich und zeitraubend.

Um bei LCD-Betrieb wie auch bei Monitorbetrieb immer den richtigen SideKick zu haben, wäre es daher wünschenswert, ein Programm zu haben, das ohne viel Aufwand den SideKick zwischen zwei vorgewählten Farbdarstellungen umschaltet.

2. Editor-Dateigröße

Der SideKick belegt je nach Editor-Dateigröße etwa 110 kBytes (bei 45 k) bzw. 80 kBytes (bei 15 k Editor-Dateigröße). Wenn es darum geht, einem Programm möglichst viel RAM-Speicher zu belassen, andererseits aber nicht zu große ASCII-Dateien zu editieren sind (wie etwa beim Programmieren modular aufgebauter dBASE- oder Assemblerprogramme oder beim Protokollieren), wird eine kleinere SideKick-Version von Vorteil sein. Ein Programm zum einfachen Umschalten des SideKick wäre wünschenswert.

3. SKC.COM: SideKickChange-Programm

Das Assembler-Programm SKC.ASM erlaubt beide Umschaltungen. Es belegt als .COM-Datei nicht einmal 500 Bytes. Der Grund für diese Veröffentlichung ist vor allem der, daß in diesem Programm viele "Tricks" angewendet wurden, um möglichst codeeffizient zu programmieren; außerdem wird von den äußerst komfortablen Xenix-orientierten Dateifunktionen Gebrauch gemacht, die im Lehrbuch "Maschinennahes Programmieren unter MS-DOS" zwar erklärt, nicht aber durch ein Beispiel belegt sind.

Zunächst mußte natürlich erforscht werden, wo die Unterschiede, welche die Farben bzw. die Editor-Dateigröße steuern, liegen. Es wurden die gewünschten SideKick-Versionen mit SKINST installiert; dann wurde mit der File-Compare- Funktion von PCTOOLS festgestellt, wo die Unterschiede liegen: Im Sektor 0 ab Displacement 51 (2 Bytes) für die Editor-Dateigröße und im Sektor 16 ab Displacement 102 (33 Bytes) für die Farben.

Aufgabe des Programms SKC ist, nach Aufruf mit 2 Parametern die gewünschten Umstellungen an der Datei SK.COM im aktuellen Verzeichnis vorzunehmen; dann erst sollte der Sidekick geladen werden. Zweck des Programms ist es nicht, den residenten SideKick zu verändern.

Die Parameter sind I oder N (Invers oder Normal) sowie + oder - (große bzw. kleine Editor-Dateigröße). Wenn kein Parameter angegeben ist, wird auch nichts geändert.

Der Ablauf ist im wesentlichen folgender:

Zunächst wird SK.COM eröffnet (Funktion 3Dh). Dann wird die Kommandozeile (im PSP - Program Segment Prefix ab Adresse 80h) auf Großbuchstaben geändert, indem Bit 5 gelöscht wird; dies verändert natürlich auch die Codes von Plus und Minus). Als Nächstes wird mittels SCAN-

Befehls geprüft, ob die vorgesehenen Parameterzeichen enthalten sind; dabei macht eine Schleife Gebrauch von einer Tabelle namens Codes, welche die vier Zeichen enthält. Für jedes enthaltenen Parameterzeichen wird ein zugeordnetes Bit in einem Byte namens Schalter gesetzt. Diese Schalter-Bits haben eine wesentliche Steuerungsfunktion in der anschließend auszuführenden Routine Patch, welche die Veränderung des SK.COM vornimmt.

Je nachdem, welches Schalter-Bit gesetzt ist, wird zu der einen oder anderen Patchtabelle zugegriffen ("Patchen" heißt wörtlich etwa Flicken, Ausbessern und bedeutet in der EDV üblicherweise das direkte Verändern ausführbaren Codes entweder im Speicher oder auf einer Datei). Jede Patchtabelle enthält die Adresse, wo ge"patcht" werden soll (Sektornummer und Offset innerhalb dieses Sektors), die Anzahl zu patchender Bytes sowie die neuen Bytewerte; falls für letztere Null angegeben ist, soll das betreffende Byte nicht verändert werden.

Nach Festlegen der Patchtabelle muß der entsprechende Sektor gelesen werden. Dazu wird mit der Funktion 42h (Dateizeiger verschieben, "Logical Seek")

bestimmt, ab wo SK.COM in den Puffer des Programms SKC zu lesen ist, nämlich ab dem ersten zu patchenden Byte; dann wird gelesen (Funktion 3Hh), in einer kleinen Schleifen-Unterroutine gepatcht, das zugehörige Steuerbit gelöscht, die vorher gelesene Anzahl Bytes (genau so viele, wie zu patchen sind) wieder zurückgeschrieben und neuerlich zum Anfang der Patch-Routine gesprungen. Der Vorgang wird wiederholt, bis keine Steuerbits mehr gesetzt sind, also normalerweise höchsten ein weiteres Mal.

Zuletzt wird SK.COM geschlossen (Funktion 3Eh) und das Programm beendet.

```
        TITLE    'SKC.ASM: SideKick einstellen: 15 k / 45 k, invers / normal'
; Aufruf mit SKC pq, wobei die Parameter p und q sein können:
;       i oder n, - oder + (invers oder normal, klein 15 k oder groß 45 k)
CodeSeg SEGMENT  PARA PUBLIC 'Code'
        ASSUME   CS:CodeSeg,SS:CodeSeg
        ASSUME   DS: CodeSeg,ES:CodeSeg
        ORG      100h
Begin:  JMP      Start
DatNam  DB       'SK.COM'    ; Name der Zieldatei auf dem aktuellen Laufwerk
Displ   DW       0           ; Displacement für Dateizeiger
KdoZLng DB       0           ; Länge der Kommandozeile
Kdo     DW       0           ; Adresse des Kommandos
Codes   DB       'IN'        ; Codezeichen in Kommandozeile
        DB       0Bh         ; "+" nach Bitlöschen
        DB       0Dh         ; "-" nach Bitlöschen
Schalter DB      0           ; Bit 7 = I, 6 = N, 5 = -, 4 = +
TabAktu DW       0           ; zum Sichern der Adresse der aktuellen Tabelle
SKSize  DB       0           ; Sektor
        DB       51          ; Displacement
SKSizeL DB       2           ; Anzahl Bytes
SK15k   DB       98h,3Ah     ; Inhalt für 15 k
SK45k   DB       50h,0C3h    ;        für 45 k
SKColors DB      16          ; Sektor
        DB       102         ; Displacement
SKColoL DB       33          ; Anzahl Bytes ( = Pufferlänge )
SKInvers DB      70h,60h,0,0,60h,0Fh,0,0,0,70h,60h,0Fh,0,0,0,70h,60h,0Fh
        DB       0,0,0,0,60h,0Fh,70h,60h,0Fh,0,75h,0Fh,70h,60h,0Fh
                 ; 0 = unverändert
SKNormal DB      7Fh,70h,0,0,75h,7Fh,0,0,0,06h,70h,70h,0,0,0,06h,70h,70h
        DB       0,0,0,0,06h,70h,06h,70h,70h,0,06h,06h,06h,70h,70h
Handle  DW       0           ; Dateinummer (File Handle)
FehlMdg DB       10,13,'SK.COM nicht da oder schreibgeschützt',10,13,"$"
;
Start:  CALL     OpenSK      ; SK.COM eröffnen
        CALL     KdoZUpCase  ; Kommandozeile in Großbuchstaben umsetzen
        CALL     Analyse     ; Kommandozeile analysieren
        CALL     Patch       ; SK.COM patchen
        JMP      CloseSK
;
OpenSK: ; SK.COM im aktuellen Pfad eröffnen
        MOV      CX,0        ; Dateiattribut 0 (keine besonderen Attribute)
        LEA      DX,DatNam   ; Dateiname
        MOV      AH,3Dh      ; Xenix-orientiertes Eröffnen eines Zugriffspfads
        MOV      AL,2        ; eröffnen für Schreiben und Lesen
        INT      21h         ; Funktion ausführen
        JC       Fehler      ; Wenn Eröffnen nicht möglich (sonst CF=0)
        MOV      Handle,AX   ; Dateinummer (Handle) sichern
        RET
Fehler: LEA      DX,FehlMdg
        MOV      AH,9
        INT      21h
        JMP      Exit
;
```

```
KdoZUpCase:    ; Kommandozeile in Großbuchstaben umsetzen
        XOR     CH,CH
        MOV     CL,CS:80h  ; Kdo-Zeilenlänge steht auf PSP+80h
        MOV     KdoZLng,CL ; Länge sichern
        MOV     DI,81h     ; Adresse erstes Byte der Kommandozeile
        MOV     Kdo,DI     ; sichern
KdoZU0: AND     BYTE PTR [DI],0DFh   ; Bit 5 löschen
        INC     DI
        LOOP    KdoZU0
        RET
;
Analyse: MOV    DI,OFFSET Codes
        MOV     SI,OFFSET Schalter
        MOV     CX,4       ; 4 mögliche Codes
        MOV     AH,80h     ; Bitcode für Code "L"
        PUSH    CS
        POP     ES
Ana0:   MOV     AL,[DI]    ; Codebuchstaben laden
        PUSH    CX
        PUSH    DI
        MOV     CL,KdoZLng ; Länge der Kommandozeile
        MOV     DI,Kdo     ; Adresse des Kommandos
        XOR     CH,CH      ; Nullsetzen
        REPNE SCASB        ; nach Codebuchstaben durchsuchen
        JNZ     AnaLoop    ; wenn nicht gefunden: Schleife fortsetzen
        OR      Schalter,AH ; wenn gefunden: Schalterbit setzen
AnaLoop: SHR    AH,1       ; Schaltermuster für nächsten Code
        POP     DI
        INC     DI         ; nächster Code
        POP     CX
        LOOP    Ana0
        RET
;
Patch:  TEST    Schalter,0C0h  ; I oder N gesetzt ?
        JZ      PatchLS        ; keines gesetzt
        LEA     SI,SKColors    ; eines gesetzt: Parameter für Sektor lesen
        MOV     TabAktu,SI     ; Tabellenadresse sichern
        MOV     BL,[SI+2]      ; Länge der Tabelle
        CALL    Read           ; Sektor lesen
        TEST    Schalter,80h   ; I gesetzt ?
        JZ      PatN           ; nein, also N gesetzt
        LEA     DI,SKInvers    ; Bytetabelle für Invers
        AND     Schalter,3Fh   ; Bits 7 und 6 löschen
PatCall: CALL   Einsetz        ; Bytes aus Tabelle einsetzen
        CALL    Write          ; Sektor zurückschreiben
        JMP     Patch          ; nochmals anfangen für Bits 5 und 4
PatN:   LEA     DI,SKNormal    ; Bytetabelle für Normal
        AND     Schalter,3Fh   ; Bits 7 und 6 löschen
        JMP     PatCall
PatchLS: TEST   Schalter,30h   ; + oder - gesetzt ?
        JZ      PatchEx        ; keines gesetzt: keine Veränderung
        LEA     SI,SKSize      ; eines gesetzt: Parameter für Sektor lesen
        MOV     TabAktu,SI     ; Tabellenadresse sichern
        MOV     BL,[SI+2]      ; Länge der Tabelle
        CALL    READ           ; Sektor lesen
        TEST    Schalter,20h   ; + gesetzt ?
        JZ      PatS           ; nein, also - gesetzt
        LEA     DI,SK45k       ; Bytetabelle für 45 k
        AND     Schalter,0CFh  ; Bits 5 und 4 löschen
        JMP     PatCall
PatS:   LEA     DI,SK15k       ; Bytetabelle für 15 k
        AND     Schalter,0CFh  ; Bits 5 und 4 löschen
        JMP     PatCall
PatchEx: RET
;
Read:   MOV     AX,512         ; Sektorgröße
        XOR     CX,CX
        MOV     SI,TabAktu     ; Tabellenadresse holen
        MUL     WORD PTR [SI]  ; Multiplikator = Sektornummer
        PUSH    BX
        MOV     BL,[SI+1]      ; Displacement
        XOR     BH,BH
        ADD     AX,BX          ; Displacement dazu
        PUSH    CX
        PUSH    AX
        POP     DX
        POP     CX             ; CX:DX = Offset für Logical Seek-Funktion
        MOV     Displ,DX       ; Offset für späteres Write sichern
        CALL    LSeek          ; Dateizeiger stellen
        MOV     AH,3Fh         ; Lesen von Zugriffspfad
        MOV     BX,Handle      ; Dateinummer
        XOR     CX,CX
        MOV     SI,TabAktu     ; Tabellenadresse holen
        MOV     CL,[SI+2]      ; Anzahl zu lesender Bytes
        LEA     DX,Puffer      ; Adresse des Datenpuffers
        INT     21h
        POP     BX
        RET
;
LSeek:  MOV     AH,42h         ; Logical Seek
        MOV     AL,0           ; Zeiger auf Dateianfang + Offset
```

```
        MOV     BX,Handle
        INT     21h
        RET
;
Einsetz: ; Bytes laut Tabelle einsetzen
        MOV     CL,BL           ; Anzahl Bytes
        XOR     CH,CH
        LEA     SI,Puffer
Eins0:  CMP     BYTE PTR [DI],0 ; Eintragen nötig ?
        JE      Eins1           ; nein
        MOV     AL,[DI]
        MOV     BYTE PTR [SI],AL ; ja: Byte einsetzen
Eins1:  INC     DI              ; nächstes Byte in Tabelle
        INC     SI              ; nächstes Byte im Puffer
        LOOP    Eins0
        RET
;
Write:  MOV     DX,Displ    ; Displacement wieder holen
        CALL    LSeek       ; Dateizeiger einstellen
        MOV     BX,Handle   ; Dateinummer (File-Handle) nach BX
        XOR     CH,CH
        MOV     SI,TabAktu  ; Tabellenadresse holen
        MOV     CL,[SI+2]   ; Pufferlänge (=Anzahl zu schreibender Bytes)
        LEA     DX,Puffer
        MOV     AH,40h      ; Funktion Schreiben auf Zugriffspfad
        INT     21h         ; ausführen
        RET
;
CloseSK: MOV    AH,3Eh      ; Funktion Schließen eines Zugriffspfads
        MOV     BX,Handle
        INT     21h
Exit:   MOV     AH,4Ch
        INT     21h
;
Puffer  EQU     $           ; Dateipuffer
;
CodeSeg ENDS
        END     Begin
```

4. Variante mit Hilfetext

Wer nicht Wert darauf legt, daß SKC.COM nur 473 Bytes belegt, kann im Anfangsbereich einige Änderungen vornehmen, sodaß er mit SKC ? einen Hilfetext als Gedächtnisstütze für die praktische Anwendung erhält. Hier nur der relevante Ausschnitt:

(ab HelpMdg neu)

```
FehlMdg  DB     10,13,'SK.COM nicht da oder schreibgeschützt',10,13,"$"
HelpMdg  DB     10,13,'Aufruf mit SKC pq, wobei die Parameter p und q sein '
         DB     'können: i oder n, - oder + ',10,13
         DB     '(invers oder normal, 15 k oder 45 k). SKC ? '
         DB     'gibt Hilfe.','$'
;
Start:   CALL   KdoZUpCase ; Kommandozeile in Großbuchstaben umsetzen
         CALL   Help       ; Hilfe ausgeben
         CALL   OpenSK     ; SK.COM eröffnen
         CALL   Analyse    ; Kommandozeile analysieren
         CALL   Patch      ; SK.COM patchen
         JMP    CloseSK
;
KdoZUpCase:    ; Kommandozeile in Großbuchstaben umsetzen
         XOR    CH,CH
         MOV    CL,CS:80h  ; Kdo-Zeilenlänge steht auf PSP+80h
         MOV    KdoZLng,CL ; Länge sichern
         MOV    DI,81h     ; Adresse erstes Byte der Kommandozeile
         MOV    Kdo,DI     ; sichern
KdoZU0:  AND    BYTE PTR [DI],0DFh   ; Bit 5 löschen
         INC    DI
         LOOP   KdoZU0
         RET
;
Help:    MOV    AL,1Fh     ; "?" ohne Bit 5
         MOV    CL,KdoZLng ; Länge der Kommandozeile
         MOV    DI,Kdo     ; Adresse des Kommandos
         REPNE SCASB       ; nach "?" durchsuchen
         JNZ    HelpEnd    ; wenn nicht gefunden
         LEA    DX,HelpMdg
         JMP    LineOut    ; Hilfetext ausgeben
HelpEnd: RET
;
OpenSK:  ; SK.COM im aktuellen Pfad eröffnen
```

(ab OpenSK: weiter wie in der Grundvariante)

INTPRUEF

Christoph Ferstl, TGM, 4ANA90 (Diskette TGM-136)

Programm zur Überprüfung der Interruptvektortabelle. Wird das Programm zum erstenmal gestartet, so wird die gesamte Vektortabelle erfaßt und in der Datei INT_TAB.DAT gespeichert. Kann die Datei nicht erstellt oder geöffnet werden, wird dies durch eine Fehlermeldung angezeigt. Bei Wiederaufruf wird die aktuelle Vektortabelle mit der alten Tabelle in INT_TAB.DAT verglichen und auf Fehler in der Übereinstimmung geprüft. Die Datei muß dabei im selben Unterverzeichnis stehen, wie das Programm INTPRUEF.COM . Sollten sich Vektore Unt haben, so werden sie am Bildschirm dargestellt. Weiters wird bei Veränderungen abgefragt, ob die Datei INT_TAB.DAT aktualisiert werden soll.

```
                        title INTPRUEF
page 72,80
comment ?   Programm - Datei  \MASM\SOURCE\INTPRUEF.ASM
;
;MASM Version 5.0
code segment
     org 100h
     assume cs:code,ds:code
;
public start
start:              jmp anfang
;
LF              equ  0Ah
CR              equ  0Dh
;
int_tab         db 'INT_TAB.DAT',0
Fehlertext_1    db CR,LF,CR,LF,'Kein Zugriff möglich (Disk vermutlich voll) !',CR,LF,'$'
Fehlertext_2    db CR,LF,CR,LF,'Zu viele Dateien geöffnet !',CR,LF,'$'
Ueberschrift    db CR,LF,'          PRÜFUNG DER INTERRUPTVEKTORTABELLE',CR,LF,'$'
Text            db CR,LF,'   Interrupt   Bestand    Segment:Offset  Vergleich','$'
Meldung_1       db CR,LF,CR,LF,'Aktuelle Vektortabelle in Int_tab.dat gespeichert !','$'
Meldung_2       db CR,LF,CR,LF,CR,LF,'In der Vektortabelle sind keine Veränderungen aufgetreten!','$'
Meldung_3       db 'OK$'
Meldung_4       db 'UNGLEICH$'
Meldung_5       db CR,LF,'Die Datei Int_tab.dat wurde geschlossen.',CR,LF,'$'
Frage_1         db CR,LF,CR,LF,CR,LF,'Soll die aktuelle Vektortabelle abgespeichert werden ?','$'
Neu             db 'neu =>$'
Alt             db 'alt =>$'
Int_num         db 1 dup (0)
Int_seg         dw ?
Int_off         dw ?
Handle          dw ?
Vmodus          db ?
Bseite          db ?
Zeile           db 1 dup (5)
Spalte          db ?
Zeichen         db ?
Schalter        dB 1 dup (0)
Schleife        dw ?
Vek_add_puf     dw 512 dup (0)
;
;
Int_tab_erstellen  proc
                   mov ah,5bh               ;Funktion Datei erstellen
                   mov cx,00                ;Dateiattribut / 0 = normal
                   lea dx,Int_tab           ;Offset des Dateinamens übergeben
                   int 21h
                   jnc OK1                  ;wenn CF=0 wurde Datei erstellt
                   cmp ax,5                 ;wenn CF=1/Fehlercode 5 =>
                   jz Fehler1               ;Zugriff verweigert (Disk voll)
                   cmp ax,80                ;wenn CF=1/Fehlercode 80 =>
                                            ;          Datei existiert bereits
                   mov Schalter,1           ;Schalter setzen: Datei existiert
                   jz Int_tab_oeffnen       ;Datei Int_tab öffnen
              OK1: mov Handle,ax            ;sichern des Dateihandles
                   ret
Int_tab_erstellen  endp
;
Vek_add_schreiben  proc                     ;schreiben in die Datei Int_tab
                   mov ah,40h               ;Funktion Datei schreiben
                   mov bx,Handle            ;Dateihandle übergeben
                   mov cx,400h              ;Anzahl der zu schreibenden Bytes
                   lea dx,Vek_add_puf       ;Offset d. Schreibpuffers übergeben
                   int 21h
```

```
                cmp ax,0                ;wenn ax = 0:Disk voll
                jnz OK3
                cmp ax,400h             ;wenn ax < 400h:Disk voll
                jz OK3
                call Int_tab_loeschen   ;Datei Int_tab löschen
                call Fehler1            ;Zugriff verweigert (Disk voll)
          OK3:  ret
Vek_add_schreiben endp
;
Fehler1         proc                    ;Fehler: Zugriff verweigert
                lea dx,Fehlertext_1
                call String_ausgabe
                jmp Fin                 ;Programm beenden
                ret
Fehler1         endp
;
Fehler2         proc                    ;Fehler: Zu viele Dateien geöffnet
                lea dx,Fehlertext_2
                call String_ausgabe
                jmp Fin                 ;Programm beenden
                ret
Fehler2         endp
;
Int_tab_loeschen proc
                mov ah,41h              ;Funktion Datei löschen
                lea dx,Int_tab          ;Offset des Dateinames übergeben
                int 21h
                ret
Int_tab_loeschen endp
;
Int_tab_oeffnen proc
                mov ah,3dh              ;Funktion Datei öffnen
                lea dx,Int_tab          ;Offset des Dateinamens übergeben
                mov al,02               ;Schreib- und Lesezugriff
                int 21h
                jnc OK2                 ;wenn CF=0 wurde Datei geöffnet
                cmp ax,4                ;wenn CF=1/Fehlercode 4 => zuviele
                jz Fehler2              ;                    Dateien geöffnet
          OK2:  mov handle,ax           ;sichern des Dateihandles
                ret
Int_tab_oeffnen endp
;
Int_tab_schlieszen proc
                mov ah,3eh              ;Funktion Datei schließen
                mov bx,Handle           ;Dateihandle übergeben
                int 21h
                ret
Int_tab_schlieszen endp
;
Int_tab_schreiben proc                  ;Vek_add_puf mit Adressen laden
                mov cx,100h             ;Schleifenzähler auf 256 setzen
       Puf_laden: xor ax,ax             ;ax-Register löschen
                mov al,Int_num          ;al mit laufender Int_num laden
                push cx                 ;sichern des Schleifenzählers
                mov cl,2
                sal ax,cl               ;Displacement: 4 * Int_num
                mov di,ax               ;Index setzen
                call Vek_add_holen      ;Int_add aus Speicher holen
                mov ax,Int_off
                mov Vek_add_puf[di],ax  ;Offset übertragen
                mov ax,Int_seg
                mov Vek_add_puf[di]+2,ax ;Segment übertragen
                inc Int_num             ;nächste Interruptnummer festlegen
                pop cx                  ;Schleifenzähler vom Stack holen
                loop Puf_laden          ;solange cx > 0:Sprung
                                        ;Vek_add in Int_tab schreiben
                call Vek_add_schreiben  ;Vek_add_puf auf Disk schreiben
                ret
Int_tab_schreiben endp
;
Vek_add_holen   proc                    ;aus aktueller Tabelle im Speicher
                mov ah,35h              ;Funktion Interruptadresse lesen
                mov al,Int_num          ;Interruptnummer übergeben
                int 21h
                mov Int_seg,es          ;Segment des Interrupts festhalten
                mov Int_off,bx          ;Offset des Interrupts festhalten
                ret
Vek_add_holen   endp
;
Vek_add_lesen   proc                    ;aus der Datei Int_tab
                mov ah,3fh              ;Funktion Datei lesen
                mov bx,Handle           ;Dateihandle übergeben
                mov cx,400h             ;Anzahl der zu lesenden Bytes
                lea dx,Vek_add_puf      ;Offset des Lesepuffers übergeben
                int 21h
                ret
Vek_add_lesen   endp
;
String_ausgabe  proc
                mov ah,9                ;Funktion: Ausgabe eines Strings
                int 21h
```

```
                ret
String_ausgabe  endp
;
Zeichen_ausgabe proc
                push cx                 ;alten Schleifenzähler sichern
                push dx                 ;dx-Register sichern
                mov cl,4                ;Vorbereitung f. 4 bit-Verschiebung
                mov bl,2                ;neuen Schleifenzähler setzen
                rol dl,cl               ;vertauschen v.unteren 4 mit oberen
        Loop_1: and dl,0fh              ;auszugebende Hexziffer isolieren
                cmp dl,9                ;Vergleich mit 9
                jbe kleiner_gleich_9    ;wenn Zahl kleiner/gleich 9: Sprung
                add dl,37h              ;Ascii-Zeichen A-F vorbereiten
                jmp weiter
 kleiner_gleich_9: add dl,30h           ;Ascii_Zeichen 0-9 vorbereiten
          weiter: mov ah,02             ;Funktion Zeichen ausgeben
                int 21h
                pop dx                  ;dx v.Stack holen f.nächste Hexzif.
                push dx
                dec bl                  ;Schleifenzähler um 1 erniedrigen
                jnz Loop_1              ;Sprung,solange bis dl = 0
                pop dx                  ;dx-Register vom Stack holen
                pop cx                  ;alt. Schleifenzähler v.Stack holen
                ret
Zeichen_ausgabe endp
;
Zeichen_eingabe proc
                mov ah,01               ;Funktion Zeicheneingabe
                int 21h
                mov Zeichen,al          ;Ascii-Wert in Zeichen übergeben
                ret
Zeichen_eingabe endp
;
Alt_ausgabe     proc
                add Zeile,1             ;Cursor in die nächste Zeile
                mov Spalte,17           ;Cursor unter 'Bestand' setzen
                call Cursor_position
                lea dx,Alt              ;Ausgabe von 'alt'
                call String_ausgabe
                mov Spalte,27           ;Cursor unter 'Segment' setzen
                call Cursor_position
                mov dl,byte ptr Vek_add_puf[di]+3 ;altes Segment ausgeben
                call Zeichen_ausgabe
                mov dl,byte ptr Vek_add_puf[di]+2
                call Zeichen_ausgabe
                mov Spalte,35           ;Cursor unter 'Offset' setzen
                call Cursor_position
                mov dl,byte ptr Vek_add_puf[di]+1 ;alten Offset ausgeben
                call Zeichen_ausgabe
                mov dl,byte ptr Vek_add_puf[di]
                call Zeichen_ausgabe
                sub Zeile,1
                ret
Alt_ausgabe     endp
;
Videomodus      proc
                mov ah,0fh              ;Bestimmung des Videomodus
                int 10h
                mov Vmodus,al           ;Videomodus festhalten
                mov Bseite,bh           ;akt. Bildschirmseite festhalten
                ret
Videomodus      endp
;
Bloesch         proc
                mov ah,00               ;Bild löschen d. Videom. setzen
                mov al,Vmodus           ;Videomodus übergeben
                int 10h
                ret
Bloesch         endp
;
Cursor_position proc
                mov ah,02               ;Funktion Cursor positionieren
                mov bh,Bseite           ;Nummer d.aktuellen Bildschirmseite
                mov dh,Zeile            ;Zeilennummer übergeben
                mov dl,Spalte           ;Spaltennummer übergeben
                int 10h
                ret
Cursor_position endp
;
Curs_pos_bestimmen proc
                mov ah,03               ;Funktion Cursorposition bestimmen
                mov bh,Bseite           ;Nummer d.aktuellen Bildschirmseite
                int 10h
                mov Zeile,dh            ;Ausgabe der aktuellen Zeilennummer
                mov Spalte,dl           ;Ausgabe d. akt.  Spaltenposition
                ret
Curs_pos_bestimmen endp
;
;
Anfang:         call Int_tab_erstellen  ;Datei Int_tab erstellen
                                        ;wenn vorhanden: öffnen
```

```
                                        ;wenn nicht vorh.:erstellen+öffnen
                cmp Schalter,1          ;Prüfung ob Int_tab vorhanden
                jz Vergleich            ;Sprung zu Vergleich
;
                call Int_tab_schreiben  ;aktuelle Vektortabelle schreiben
;
                lea dx,Meldung_1        ;Adresse von Meldung_1 übergeben
                call String_ausgabe     ;Meldung_1 ausgeben
                jmp Ende                ;Programm beenden
;
                                        ;Bildschirmaufbau
        Vergleich: mov Schalter,0       ;Schalter rücksetzen
                call Videomodus         ;Vmodus und Bseite feststellen
                call Bloesch            ;Bildschirm löschen
                lea dx,Ueberschrift     ;Programmtitel ausgeben
                call String_ausgabe
                lea dx,Text             ;Textzeile ausgeben
                call String_ausgabe
;
                call Vek_add_lesen      ;Adressen von Int_tab in d.Puffer
;
                mov cx,100h             ;Schleifenregister initialisieren
;
          Loop_2:                       ;Ausgabe der Interruptnummer
                mov Spalte,7            ;Cursor unter 'Interrupt' setzen
                call Cursor_position
                mov dl,Int_num          ;Interruptnummer ins dl-Register
                call Zeichen_ausgabe    ;Ausgabe der Interruptnummer
;
                mov Spalte,17           ;Cursor unter 'Bestand' setzen
                call Cursor_position
                lea dx,Neu              ;Ausgabe von 'neu'
                call String_ausgabe

                                        ;Ausgabe des neuen Segments
                mov Spalte,27           ;Cursor unter 'Segment' setzen
                call Cursor_position
;
                call Vek_add_holen      ;Vek_add aus Speicher bestimmen
                mov dl,byte ptr Int_seg+1 ;höherwertige Byte v.Seg.vorbereit.
                call Zeichen_ausgabe    ;Ausgabe des höherwertigen Bytes
                mov dl,byte ptr Int_seg ;niederwertiges Byte vorbereiten
                call Zeichen_ausgabe    ;niederwertiges Byte ausgeben
;
                                        ;Ausgabe des neuen Offsets
                mov Spalte,35           ;Cursor unter 'Offset' setzen
                call Cursor_position
                mov dl,byte ptr Int_off+1 ;höherwertiges Byte v.Offset vorbe.
                call Zeichen_ausgabe    ;Ausgabe des höherwertigen Bytes
                mov dl,byte ptr Int_off ;niederwertiges Byte vorbereiten
                call Zeichen_ausgabe    ;Ausgabe des niederwertigen Bytes
;
                                        ;Vergleich durchführen
                xor ax,ax               ;ax-Register auf 0 setzen
                mov al,Int_num          ;Interruptnummer übergeben
                push cx                 ;Schleifenzähler sichern
                mov cl,2
                sal ax,cl               ;Displacement: 4 * Int_num
                mov di,ax               ;Displacement setzen
                jmp Bruecke
;
          Hilfe: jmp loop_2
;
        Bruecke: call Alt_ausgabe       ;Ausgabe von 'alt' mit Seg+Off
                                        ;Vergleich durchführen
                mov ax,Int_seg          ;Int_seg vom Speicher laden
                cmp ax,Vek_add_puf[di]+2 ;Vergleich mit Int_seg aus Int_tab
                jnz Ungleich            ;wenn nicht gleich: Sprung
                mov ax,Int_off          ;Int_off vom Speicher laden

                cmp ax,Vek_add_puf[di]  ;Vergleich mit Int_off aus Int_tab
                jnz Ungleich            ;wenn nicht gleich: Sprung
;
                mov Spalte,45           ;Cursor unter 'Status' setzen
                call Cursor_position
                lea dx,Meldung_3        ;Meldung_3 übergeben
                call String_ausgabe     ;Meldung_3 ausgeben
                jmp short Neuer
;
        Ungleich: mov Schalter,1        ;wenn ungleich  => Frage
                mov Spalte,43
                call Cursor_position
                lea dx,Meldung_4        ;Meldung_4 übergeben
                call String_ausgabe     ;Meldung_4 ausgeben
                add Zeile,2             ;Zeilenvariable um 2 erhöhen
           Neuer: pop cx                ;Schleifenzähler vom Stack holen
                inc Int_num             ;nächster Interrupt
                loop Hilfe
;
           Frage: cmp Schalter,1        ;Vergleich, ob Schalter gesetzt
                jnz Alles_klar          ;wenn cx = 0: Frage ausgeben
                lea dx,Frage_1          ;Offset von Frage übergeben

;
Ueberschreiben: call Int_tab_schliesen  ;Int_tab schließen
                call Int_tab_oeffnen    ;Int_tab öffnen
                mov Int_num,0           ;Interruptnummer auf Beginn stellen
                call Int_tab_schreiben  ;aktuelle Vektortabelle schreiben

                call Int_tab_schliesen  ;Int_tab schließen
                lea dx,Meldung_5        ;Meldung_5 ausgeben
                call String_ausgabe
          Fin: mov ah,4ch
                int 21h
;
          Ende:                         ;
;
code ends
     end start

;aktuelle Cursorposition bestimmen
;Cursor um 1 nach rechts
;Cursor auf Eingabeplatz
;J/N eingeben
;Vergleich mit 'J'
;wenn 'J':Sprung zu Int_tab schrei.
;Vergleich mit 'J'
;wenn 'J':Sprung zu Int_tab schrei.
;wenn nicht: Programm beenden

call String_ausgabe
call curs_pos_bestimmen
inc Spalte
call Cursor_position
call Zeichen_eingabe
cmp Zeichen,'J'
jz Ueberschreiben
cmp Zeichen,'J'
jz Ueberschreiben
jmp Ende

;Meldung_2 übergeben
;Ausgabe von Meldung_2
;Programm beenden

Alles_klar: lea dx,Meldung_2
            call String_ausgabe
            jmp short Ende
```

Spooler

Otto Heilig, Franz Lückl, TGM, 4ANA89 (Diskette TGM-136)

Dieses Programm wird ohne Argumente aufgerufen und installiert einen 64k Puffer als Spooler. Defaultmäßig wird der Drucker Nr. 1 verwendet. Bei einem 2. Aufruf erscheint die Abfrage " Druckerpuffer löschen (J/N) ? ", bei der Eingabe von j wird der Spooler-Speicherbereich gelöscht. Das Programm kann nur durch Ctrl-Alt-Del gelöscht werden.

Kommandos des Drucker-Interrupts

0: Drucken eines Zeichen, das sich im AL-Register befindet. Versuch an einen Drucker ein Zeichen zu senden, der ausgeschaltet ist: Bit Nummer 0 im AH-Register gesetzt (Device-Timeout)

1: Der im DX-Register angegebene Drucker wird zurückgesetzt; die INIT-Leitung wird aktiviert.

2: Der Status des gewählten Druckers im AH-Register wird zurückgegeben.

Kommandos des Tastatur-Interrupts

0: Warten auf einen Tastendruck. Wenn eine Taste gedrückt wurde, so ist der Scancode der Taste im AH, das ASCII-Zeichen im AL-Register.

1: Bestimmung des Tastaturstatus. Wird eine Taste gedrückt, so wird das Zeroflag im Prozessorstatusregister zurückgesetzt. Die folgende Programmsequenz wartet also auf einen Tastendruck :

```
KEY:  MOV AH,1
      INT 16h
      JZ KE
```

2: Status der Ctrl-, Alt- und Shifttasten wird ins AL-Register geschrieben. Aufruf mit Interrupt 16 und AH = 2.

```
title   DRUCKER PUFFER FÜR IBM PC's
        page 66,80

        dos     equ 21h                 ; DOS Interrupt
        intr    equ 16h                 ; Keyboard interrupt
        print   equ 17h                 ; BIOS Drucker I/O
        printer equ 0                   ; erster Drucker

comment *

        DOS Drucker-Puffer mit  64 kByte als Pufferspeicher

        *

        d0      segment para 'data'             ; 1. Datensegment
                buf0    dw 4000h dup (?)
        d0      ends

        d1      segment byte 'data'             ; 2. Datensegment
                buf1    dw 4000h dup (?)
        d1      ends

c               segment para 'code'             ; Programm-Segment
                assume  cs:    c, ds:c

                org     100h

pr_buf          proc far
                sti                             ; Setzen des Interrupt-Flags
                push    ds                      ; Register sichern
                push    bx
                push    dx
                push    ax
                pushf
                mov     ax,cs                   ; Datensegment holen
                mov     ds,ax
                mov     bx,[pi]
                cmp     bx,[pp]                 ; Ist etwas zu drucken ?
                je      no_print                ; Nein - weiter mit Interrupt
```

```
                                                ; chain
                mov     ah,2
                mov     dx, [printer]           ; Drucker-Status  holen
                pushf
                call    ds:[pr_intr]            ; BIOS Routine aufrufen
                test    ah,80h                  ; Drucker bereit ?
                jz      no_print                ; Nein
                push    ds
                mov     ds,cs:[d_seg]           ; Datensegment holen
                mov     al,[bx]                 ; Character holen
                pop     ds
                xor     ah,ah                   ; Kommando um ein Zeichen zu
                                                ; drucken
                mov     dx,printer              ; Drucker-Nummer holen
                pushf                           ; Datenregister sichern
                call    ds:  [pr_intr]
                inc     bx
                mov     [pi],bx
no_print:
                popf
                pop     ax
                test    ah,ah                   ; Hole Tastatur-Kommando ?
                jnz     l1                      ; Nein - Rufe BIOS-Routine
;
l0:
                mov     ah,1                    ; Warten auf einen Character
                int     intr                    ; Aufruf der Intr-routine
                jz      l0
                xor     ah,ah
l1:
                pop     dx
                pop     bx
                pop     ds
                jmp     cs:[key_intr]           ; Aufruf der BIOS-Routine
pr_buf          endp

pr_irq          proc    near
                push    ds
                push    bx
                push    dx                      ; Registers abspeichern
                push    ax
                test    ah,ah
                jz      pr_buf0                 ; Ist ein zeichen zu drucken ?
                pop     ax
                mov     ah,90h                  ; Das aufrufende Programm erhält
                                                ; Signal "Drucker fertig"
                jmp     short cont0
pr_buf0:
                mov     bx,cs
                mov     ds,bx                   ; Datensegment holen
                mov     bx, [pp]                ; Zeiger des Druckerpuffers
                                                ; holen
                push    ds
                mov     ds,cs:[d_seg]           ; Puffersegment holen
                mov     [bx],al                 ; Speichere Byte
                pop     ds
                inc     bx                      ; Erhöhe Puffer um 1
                mov     [pp],bx                 ; Wenn 64k erledigt sind ,
                                                ; BX wird geteilt durch 0
                pop     ax
                mov     ah,90h                  ; Das aufrufende Programm erhält
                                                ; Signal "Alles in Ordnung"
cont0:
                pop     dx
                pop     bx                      ; Register zurückholen
                pop     ds
                iret                            ; Return from interrupt
pr_irq          endp

                pp      dw 0
                pi      dw 0
                key_intr dd ?
                pr_intr  dd ?
                d_seg    dw ?
;
start           proc    near
                mov     ax,cs
                mov     ds,ax
                mov     ax,35h shl 8+intr       ; Hole alten Keyboard Vector
                int     dos
                cmp     bx,offset pr_buf        ; Puffer bereits installiert ?
                jnz     not_inst                ; Nein
                mov     ah,9
                mov     dx,offset mess          ; Print Fehlermeldung
                int     dos
                mov     ah,1                    ; Hole Tastatureingabe
                int     dos
                or      al,20h                  ; Umwandeln in Großbuchstaben
                cmp     al,"j"                  ; Pufferspeicher löschen ?
                jne     no_clear                ; Nein
                xor     ax,ax
```

```
                mov     es:[pp],ax              ; Pointer zurücksetzen
                mov     es:[pi],ax
                mov     es,es:[d_seg]           ; Puffersegment holen
                mov     cx,length buf0*2        ; Pufferlänge in words
                mov     di,ax                   ; Start-Adresse
                rep     stosw                   ; Puffer löschen
no_clear:
                mov     ax,4c00h                ; Abbruch des Programmes mit
                                                ; MS-DOS Call
                int     dos
not_inst:
                mov     word ptr [key_intr],bx  ; Sichern für die Interruptkette
                mov     word ptr [key_intr+2],es
                mov     dx,offset pr_buf
                mov     ax,25h shl 8+intr       ; Neuen Tastaturvektor setzen
                int     dos
                mov     ax,35h shl 8+print      ; Alten Drucker-I/O-Interr.-
                                                ; Vektor sichern
                int     dos
                mov     word ptr [pr_intr],bx   ; Sichere alten 17h Vektor
                mov     word ptr [pr_intr+2],es
                mov     dx, offset pr_irq
                mov     ax,25h shl 8+print      ; Neuen Drucker-I/O-Interr.-
                                                ; Vektor setzen
                int     dos
                xor     ax,ax
                mov     word ptr [pi],ax        ; Zähler löschen
                mov     word ptr [pp],ax
                mov     ax,d0
                mov     [d_seg],ax              ; Puffer Segment Adresse lagern
                mov     es,ax
                xor     ax,ax
                mov     di,ax                   ; Start bei 0
                mov     cx,length buf0*2        ; 32768 words zu löschen
                push    cx
                rep     stosw
;
                mov     dx,offset m             ; Drucker Begrüßungsbotschaft
                mov     ah,9
                int     dos
                mov     dx,1000h                ; Puffer Größe (64k) in
                                                ; Abschnitten
                mov     bx,100h                 ; Berechne Programmgröße
                add     bx,offset start
                mov     cl,4
                shr     bx,cl                   ; In Abschnitte umwandeln
                add     dx,bx
                inc     dx
                mov     ax,3100h                ; DOS - Resident Aufruf
                int     dos
                start   endp
                mess    db 13,10,"Drucker-Puffer bereits installiert!",13,10
                        db "Drucker-Puffer löschen (j/n)? $"
                m       db 13,10, "Drucker-Puffer ist jetzt fertig installiert."
                        db " Druckerdaten werden in 64 kByte"
                        db 13,10, "des Systemspeichers gepuffert.",13,10,"$"
c               ends
            end start
```

LPTXCHG

Eduard ERHART, 4ANA90 (Diskette TGM-136)

Das Programm LPTXCHG dient zum Austauschen von Portadressen zwischen zwei parallelen Schnittstellen im PC. Unter MSDOS können üblicherweise vier Drucker angeschlossen werden. Die Gerätetreiber werden dann unter den Namen LPT1 bis LPT4 angesprochen.

Standardsoftware und bedienungsfreundliche Programme erlauben während der Ausführung oder bei der Installation die Auswahl eines beliebigen Druckeranschlußes. Die Standarddruckerausgabe führt MSDOS über LPT1 durch und setzt diesen PRN gleich. Schon in den Betriebssystemprogrammen PRINT und GRAPHICS muß der Port extra angegeben werden, was leicht lästig fallen kann. Auch gibt es eine Vielzahl kleiner Programme, deren Umstellung auf einen anderen Druckerport unmöglich oder zumindest undokumentiert ist.

LPTXCHG, in AUTOEXEC.BAT eingebunden, behebt diesen Mangel.

Bedienung

LPTXCHG wird von der Kommandozeile aus mit Parametern versorgt. Der Aufruf erfolgt durch den Namen und den Ziffern der zu tauschenden Drcukeranschlüsse.

```
LPTXCHG 12
```

Die obige Zeile bewirkt das Vertauschen der Druckerports von LPT1 mit LPT2. Durch das Tauschen der Adressen erspart man sich das Speichern der alten Werte. Alle Ausdrücke, die das Programm an LPT2 schickt, werden nach LPT1 umgeleitet. Durch erneuten Aufruf mit denselben Parametern wird der alte Zustand wiederhergestellt.

Fehlermeldungen

Bei falschen Parameterangaben kennt das Programm drei Fehlermeldungen:

```
"Die Parameter in der Kommandozeile sind unrichtig"
"Es sind keine Paramtere in der Kommandozeile"
"Der angegebene Druckeranschluß ist nicht installiert"
```

Ist im Rechner keine entsprechende Einsteckkarte vorhanden, erkennt das Programm dies und unterbricht die Ausführung.

```
title   lptxchg - vertauscht Portadressen von lptn (n=1..4)

biosseg equ     0040h   ; BIOS-Datensegment

code    segment para
        assume  cs:code,ds:code
        org     100h

main:   jmp     begin   ; Datenbereich überspringen

        ; DATEN
        ; -----

        ; Fehlermeldungen
        ; ---------------

wrong   db      'Die Parameter in der Kommandozeile sind unrichtig$'
noparr  db      'Es sind keine Parameter in der Kommandozeile$'
swap    db      'Portadressen sind ausgetauscht$'
noprint db      'Der angegebene Druckeranschluß ist nicht installiert$'

        ;PROGRAMM
        ;--------
```

```
        ;DS zeigt bereits auf Segmentvorspann

begin:  push    ds              ; ds sichern
        mov     si,80h          ; Offset vom Diskettenzwischenspeicher
        xor     cx,cx           ; CX mit Länge der Kommandozeile laden
        mov     cl,[si]
        jcxz    noparaerr       ; Parameter vorhanden?
        cmp     cl,3            ; stimmt die Länge(2 Zeichen)?
        jne     paraerr


lesen:  mov     al,[si+2]       ; erstes Zeichen ins AL-Register
        mov     bl,[si+3]       ; zweites Zeichen ins bl

        sub     al,48d          ; aus ASCII Dezimalzahl
        sub     bl,48d

        push    bx              ; bx sichern
        add     bl,al           ; Summe der Parameterwerte bilden
        cmp     bl,2            ; Summe muß mindestens 3 sein
        jbe     paraerr ; wenn kleiner, ...
        cmp     bl,8            ; Summe darf max 7 (lpt3+lpt4)
        jae     paraerr         ; wenn größer,...
        pop     bx              ; bx rückholen

;       Adresstabelle
;       lpt     offset
;       --------------
;       1  ->   8
;       2  ->   10
;       3  ->   12
;       4  ->   14
;       offset = lpt*2+6

        mov     cx,biosseg
        mov     ds,cx           ; Bios-datensegment 40hex als neuer
                                ; Datenbereich
        xor     si,si           ; si löschen
        mov     si,ax           ; erste Printernummer
        call    calc            ; offset in si bilden
        mov     cx,word ptr [si]; erste Druckerportadresse zwischenspeichern
        push    si              ; ersten offset zwischenspeichern
        cmp     cx,0            ; installiert???
        je      printer

        xor     si,si           ; si löschen
        mov     si,bx           ; zweite Printernummer
        call    calc            ; offset
        mov     dx,word ptr [si]; zweite Adresse speichern
        cmp     cx,0            ;installiert???
        je      printer

        mov     word ptr [si],cx; erste Adresse an Platz der zweiten speichern
        pop     si              ; ersten offset
        mov     word ptr [si],dx; zweite Adresse an Platz der ersten speichern

        pop     ds              ; altes ds wieder herstellen
        mov     dx,offset swap

message:
        mov     ah,09h          ; Stringausgabe
        int     21h
bye:    mov     ah,4ch          ; zurück zum DOS
        int     21h

noparaerr:
        call    beep
        mov     dx,offset noparr
        jmp     message

paraerr:
        call    beep
        mov     dx,offset wrong
        jmp     message

printer:
        call    beep
        pop     ds
        pop     ds              ; mit altem ds
        mov     dx,offset noprint
        jmp     message

;-------------------------------------------------
beep    proc    near
        mov     ah,2
        mov     dl,7
        int     21h
        ret
beep    endp
;-------------------------------------------------

calc    proc    near            ; si verdoppeln
        add     si,si           ; Adresse bilden laut Formel
        add     si,6
        ret
calc    endp
;-------------------------------------------------
code    ends
        end     main
```

AUTOASK

R. Syrovatka, TGM (Diskette TGM-136: AUTOASK.COM, AUTOASK.ASM)

Das Programm AUTOASK.COM entstand aus der Notwendigkeit heraus, bei einem nicht besetzten Rechner im Netzwerk automatisch die Netz-Software zu laden, wenn er ferneingeschaltet wurde.

In die AUTOEXEC.BAT-Datei des unbesetzten Rechners wurden folgende Zeilen aufgenommen:

```
...
echo SOLL NETZWERK GELADEN WERDEN  J/N
autoask
if errorlevel 1 goto weiter
Netz
...
:weiter
...
```

Nach dem Ferneinschalten, das über eine eigene Netzzuleitung erfolgte, lud der Rechner dann automatisch nach dem BOOTEN die Netz-Software. Wurde dieser Rechner hingegen für eigene Anwendungen selbst hochgefahren, konnte das Laden durch Eingabe von N unterbunden werden, wodurch der gesamte Speicher für die Anwender-Software zur Verfügung stand.

Nun zum Programm selbst. Das Source-Listing ist ziemlich ausführlich dokumentiert, deshalb sollen, vor allem für ASSEMBLER-Neulinge, nur einige Schwerpunkte besprochen werden:

Da das Programm später als .COM-Programm laufen soll, ist die ORG 100h-Direktive erforderlich:

```
            ORG    100h     ;jedes .COM-Programm beginnt an der
                            ;OFFSET-Adresse 0100h mit einem
                            ;ausführbaren Befehl

            ASSUME CS:CSEG,DS:CSEG,SS:CSEG,ES:CSEG

START   PROC   NEAR
            NOP     ;Erster ausführbarer Befehl für autoask.COM
            NOP
            NOP
            JMP   INIT1
```

Bei INIT wird zunächst überprüft, ob beim Aufruf von AUTOASK Parameter angegeben wurden. Als Parameter kommen eine ein- oder zweiziffrige Zahl oder das Fragezeichen in Frage. Alle übrigen Zeichen werden ignoriert. Der Eingabepuffer beginnt bei 80h, wobei als erstes die Anzahl der im Anschluß an den Programmaufruf eingegebenen Zeichen steht (das abschließende CARRIAGE RETURN wird dabei nie mitgezählt!). Wurden keine Parameter angegeben, so wird bei INIT2 der vorgegebene Wert von 5 Sekunden (AH=0, AL=5) in den Speicherplatz für die Verzögerungszeit DLY geladen. Wurde ? übergeben, so wird der HELP-Bildschirm angezeigt:

```
INIT1:
MOV   SI,80h              ;80h = Beginn des Kommandopuffers
      MOV   AL,[SI]   ;Anzahl der Zeichen im Puffer (ohne abschließendes
CR)
      CMP   AL,0      ;wenn 0 keine weitere Eingabe
      JE    INIT2

      XOR   AH,AH
      ADD   SI,AX     ; = letztes Zeichen im Eingabepuffer (vor CR)
      MOV   AL,[SI]
      CMP   AL,"?"          ;bei ? Helpmenü aufrufen und beenden
      JE    HELP
```

Nun erfolgt die Überprüfung, ob die eingegebenen Zeichen tatsächlich Ziffern sind (müssen zwischen 30h und 39h liegen!). Ist dies nicht der Fall wird wieder mit der Standardeinstellung bei INIT2 fortgefahren. Bei einer gültigen Ziffer wird diese vom ASCII-Wert in den Dezimalwert (= Hexwert) umgewandelt und in der entsprechenden Speicherstelle (EINER bzw. ZEHNER) gespeichert.

```
     CMP  AL,30h          ;ASCII-Zahl 0 = 30h
     JC   INIT2           ;wenn CARRY - keine Zahl
     CMP  AL,40h          ;ASCII-Zahl 9 = 39h
     JNC  INIT2           ;wenn noch kein CARRY, dann keine Zahl
     SUB  AL,30h          ;ASCII-Zahl in Dezimalwert umwandeln
     MOV  EINER,AL  ;Einerstelle sichern

     DEC  SI
     MOV  AL,[SI]
     CMP  AL,30h          ;ASCII-Zahl 0 = 30h
     JC   INIT2           ;wenn CARRY - keine Zahl
     CMP  AL,40h          ;ASCII-Zahl 9 = 39h
     JNC  INIT2           ;wenn noch kein CARRY, dann keine Zahl
     SUB  AL,30h          ;ASCII-Zahl in Dezimalwert umwandeln
     MOV  ZEHNER,AL ;Zehnerstelle sichern
```

Die in EINER und ZEHNER gespeicherten Werte werden nun in AL addiert und als "Verzögerungszeit" in DLY gespeichert:

```
INIT2:
     MOV  AL,EINER        ;Verzögerungszeit ermitteln: Für jede
     MOV  AH,ZEHNER       ;Zehnerstelle in AH werden 10 zu AL, in
IN2: SUB  AH,1            ;dem bereits die Einer stehen, addiert.
     JC   IN4
     ADD  AL,10
     JMP  IN2

IN4: MOV  DLY,AL     ;Verzögerungszeit in DLY
```

Bei INIT3 wird der Vektor (Interrupt-Adresse) des USER TIMER INTERRUPTS gesichert und auf die Adresse der AUTOASK-Routine umgeleitet. Fortan wird bei jedem Auftreten des USER TIMER INTERRUPTS (18,2 mal in der Sekunde) die Routine DELAY angesprungen, der Wert in COUNT um 1 verringert und der USER TIMER INTERRUPT an die ursprüngliche Vektoradresse weitergereicht (JMP DWORD PTR CS:TIMER):

```
DELAY:    STI          ;Interrupts wieder zulassen
     PUSH DS
     PUSH CX
     PUSH CS
     POP  DS           ;da Einsprung über INT Datensegment setzen
     MOV  CL,COUNT     ;Timer-Interrupt kommt 18,2 mal pro Sekunde,
     DEC  CL           ;nach 18 mal, also nach etwa 1 Sekunde ist
     JZ   DLY4         ;CL und damit COUNT = Null geworden.
DLY2:MOV  COUNT,CL
     POP  CX
     POP  DS           ;altes Datensegment wiederherstellen
     JMP  DWORD PTR CS:TIMER
```

Ist COUNT schließlich Null geworden (nach 18 mal der Fall, entspricht etwa einer Sekunde) wird DLY um 1 verringert und, falls DLY noch nicht Null ist, der Zähler COUNT erneut mit dem Wert 18 geladen:

```
DLY4:MOV  CL,DLY
```

```
     OR    CL,CL          ;ist DLY schon Null ? Wenn nicht
     JZ    DLY6
     DEC   CL       ;DLY um 1 verringern und COUNT wieder
     MOV   DLY,CL
DLY6:     MOV  CL,18          ;auf 18 stellen für neuen Durchgang
     JMP   DLY2
```

Nach dieser Installation wartet AUTOASK in der "Zeitschleife" solange, bis entweder der Wert in DLY Null geworden ist oder aber eine "gültige Taste" betätigt wurde.

```
;********** ZEITSCHLEIFE ANFANG **********************************

BEGINN: BEEP
BEG2:MOV   AH,0Bh          ;Wurde ein Zeichen über Tastatur eingegeben?
     INT   21h        ;Rückgabe:  AL = 00   kein Zeichen
     OR    AL,AL           ;            AL = FF   Zeichen verfügbar
     JNZ   EINGABE
     MOV   AL,DLY
     OR    AL,AL
     JZ    EIN2
     JMP   BEG2       ;Solange, bis entweder eine Eingabe erfolgt
                      ;oder DLY Null geworden ist.

;********** ZEITSCHLEIFE ENDE ************************************
```

Gültige Zeichen bei der Eingabe sind nur J und N, wobei es gleichgültig ist, ob die Eingabe in Klein- oder Großbuchstaben (unterscheiden sich nur im Bit 5) erfolgt. Dies wird durch Setzen des Bits 5 (OR AL,00100000b) erreicht. Bit 5 ist bei Großbuchstaben ohnedies 1, bei Kleinbuchstaben wird es durch die ODER-Funktion auf 1 gesetzt.

```
EINGABE:
     MOV   AH,07h               ;Console Char.Input ohne Echo, Zeichen in
AL
     INT   21h
     OR    AL,00100000b         ;Groß- in Kleinbuchstaben umwandeln
     CMP   AL,"j"
     JE    EIN2
     CMP   AL,"n"
     JE    EIN4
     JMP   BEGINN

EIN2:     MOV  ERR,0
     JMP   SHORT ENDE

EIN4:     MOV  ERR,1
     JMP   SHORT ENDE
```

Wurde J eingegeben oder ist die Zeit ohne Eingabe abgelaufen, wird ERR = 0 gesetzt, bei einer Eingabe von N wird ERR = 1 gesetzt. Mit ENDE2 wird das Programm beendet und in AL dieser ERRORLEVEL an das aufrufende Programm - meist das Betriebssystem - übergeben. Im Betriebssystem kann dieser ERRORLEVEL, z.B. in einer BATCH-Datei abgefragt werden.

```
ENDE2:    BEEP

     MOV   AL,ERR          ; AL = Beendigungscode (ERRORLEVEL ERR)
     MOV   AH,4Ch          ; Programm beenden
     INT   21h
```

LOGO WRITER 2.0

```
                    LOGO WRITER 2.0
                    H A N D B U C H
                     Version 1.00

TASTENBELEGUNG:

F 1 : selektiert einen Textbereich
F 2 : löscht (selektierten) Text heraus und speichert ihn zum
              Verschieben im Speicher "clipboard"
F 3 : kopiert (selektierten) Text heraus und speichert ihn wie bei F2
F 4 : fügt den, mit <F2> od. <F3>, gewählten Text an beliebiger Stelle
              ein.
F 6 : LÖSCHT , z.B. Textzeilen, aber auch Dateien !!!
F 8 : dient zum Beschriften von Graphikseiten (Text = Graphik !)
              falscher Text wird durch Überschreiben mit dem selben
              Inhalt beseitigt.
F 9 : schaltet die Schildkröte auf die Cursortasten um (turtle move)
F 10 : ruft das Programm HELP.LWR auf (wenn vorhanden)
ESC : verläßt die, mit den F-Tasten, angewählten Modi.

Ctrl - U : (UP) -> schaltet von der Kommandozeile in die Textseite
              Text im Textmodus !!!
Ctrl - D : schaltet in die Kommandozeile zurück
Ctrl - F: schaltet auf die Programmierseite (flip-side) und wieder
              zurück !!
Ctrl Break / Ctrl - S :  stoppt alles
Ctrl <--- : an den Anfang der Zeile
Ctrl ---> : an das Ende der Zeile
Del, Ins, PgUp, PgDn, Home, End : exakte MS DOS Funktion !

STARTEN DES LOGOWRITERS :

nach dem Titelbild erscheint das Inhaltsverzeichnis :


Logowriter----------------contents--------------------------------

New page
Shapes

help
+ alle gespeicherten *.lwr Dateien

Mit dem Cursor wählt man
          - eine neue Seite
          - die vordefinierten Cursorzeichen (shapes)
          - bereits vorhandene Programme
```

ERSTELLEN EINER NEUEN SEITE :

----------????----------

TURTLE
in Position 0/0

Kommandozeile mit blinkendem Cursor

----------Bildschirmende----------

Befehlseingabe in der Kommandozeile :

LOGOWRITER BEFEHLE :

forward_Punktezahl	{fd 50}	bewegt die Schildkröte 50 Bildpunkte vorwärts = nach oben
back_Punktezahl	{bk 30}	rückwärts
right_Winkel	{rt 45}	dreht die Turtle um 45° nach rechts
left_Winkel	{lt 30}	dreht um 30° nach links
hide turtle	{ht}	macht Turtle unsichtbar
show turtle	{st}	zeigt sie wieder
pen erase	{pe}	radiert
pen up	{pu}	Turtle hinterläßt keinen Strich
pen down	{pd}	Strich wieder ein
pixel	{px}	zeichnet einen Strich, wo keiner ist und löscht, wo einer ist.
fill		füllt eine Figur mit der aktuellen Farbe
stamp		hinterläßt einen Abdruck der Turtle
shade		füllt eine Figur mit der aktuellen Turtlefigur

Fill, stamp und shade benötigen unbedingt vorher die Befehle st und pd, weil sonst der Befehl nicht ausgeführt werden kann.

FARB- UND TURTLEÄNDERUNG

setc_Farbe (setcolor)	setzt die Farbe der Turtle fest Farbe: 0 - 15 (EGA)
setbg_Farbe (setbackground)	Farbe des Hintergrundes 0 - 15
setsh_Turtlenummer (set shape)	setzt neue Figur nach dem Shape-katalog. Alle Figuren anschauen : shapes - Ende mit Esc -

Erstellen eigener Shapes : leere Stelle im Shapekatalog anwählen, <Enter> -> Shape Editor : mit <Leer> sowohl einen Punkt setzen als auch löschen - Ende mit Esc.

slowturtle	verringert Turtlegeschwindigkeit, Standard ist fastturtle.
seth_Winkel (set heading)	richtet die Turtle nach dem Winkel aus
show heading	zeigt den Winkel der Turtle an

REORGANISATION

cg	clear graphik	löscht die Graphik und die Turtle steht wieder in der Mitte
ct	clear text	löscht Text im Textmodus
cc	clear command	löscht alles in der Kommandozeile
rg	reset graphik	setzt die Seite in den Urzustand
		-> Einschaltfarbe, Turtle in der Mitte
		-> Text und Graphik gelöscht !
cp	clear page	löscht Graphik/Textseite und Programmseite
restore		stellt die mit CP gelöschten Seiten von der Diskette wieder her
home		bringt Turtle immer in Bildschirmmitte

VERLASSEN DES LOGOWRITERS :
a) mit Abspeichern :

setdisk "Laufwerk	wählt ein Laufwerk zum Abspeichern
show disk	zeigt aktuelles Laufwerk an
savepage "Dateiname	speichert die Datei unter Dateiname.LWR
namepage "Dateiname	vergibt Dateiname und speichert.
dos	kehrt zum Betriebssystem zurück

b) ohne Abspeichern der aktuellen Seite ;
- clear page
mit Ctrl-F auf die Flip Seite und dort Esc, man befindet sich im Inhaltsverzeichnis. Auswahl einer bereits gespeicherten Seite und in deren Kommandozeile "dos" eingeben.

DATEIEN UND INHALT:

filelist	bringt alphabetisch geordnet alle Dateien des aktuellen Verzeichnisses

Eingabe : show filelist

contents	bringt alle "pages" (= *.LWR) des aktuellen Verzeichnisses
printscreen	druckt die komplette Seite
printtext	druckt nur Text im Textmodus

MATHEMAT. OPERATIONEN

show 1 + 3	show 8 - 3	show 2 * 5	show 25 / 5
4	5	10	5

```
ERSTELLEN EINES PROGRAMMES = "procedur"

1) Wechsel auf die Programmierseite mit Ctrl-F
-------------------------flip side---------------------------------

     ---> Programmbefehl :to

to quadrat
repeat 4 [fd 50 rt 90 ]
end


Programmbefehl : repeat_Zahl      Wiederholung Zahl-mal
                 end              Ende der Prozedur
-----------------------------------------------------------------
2) Zurück zur Kommandozeile wieder mit Ctrl-F

3) Eingabe von quadrat : zeichnet ein  Quadrat,  das  Wort  quadrat
wird zu einem Logo Befehl (= tool).


Aufruf von tools in einer procedur :

1) erstellen eines tools : z.B. 6ECK
     to 6eck
     repeat 6[fd 30 rt 60]
     end
2) Einbau von "6eck" in einer neuen Prozedur :
     to stern
     repeat 6[6eck rt 60]
     end
3) Definition von Variablen für Punktezahl und/oder Winkel :
     to kreis  :umfang
     repeat 18[fd :umfang rt 20]
     end
4) Aufruf einer Prozedur in sich selbst :REKURSION
     to muster :umfang
     if :umfang > 100 [stop]          --> Abbruchbedingung,  sonst
                                      läuft die Prozedur unendlich.
     kreis :umfang
     muster :umfang + 5
     end
```

```
Weitere Programmbefehle :

print "TEXT                schreibt den Text in die oberste Zeile und
                           macht einen Zeilenvorschub
print [Liste]             schreibt den Inhalt der [ ]

random  ZAHL               schreibt eine ganzzahlige Zufallszahl
                           kleiner als ZAHL
                           SHOW RANDOM 10      ----> z.B. 4
                           fd random 100       ----> alles < 100
show "TEXT                 schreibt den Text in der Kommandozeile
show [Liste]               schreibt den Inhalt der [ ]

                           show space  : zeigt Speicherplatz in %


                           repeat 5 [show [ Hallo Freunde ]]
                           show  bg          : zeigt Farbe des Hinter-
                                               grundes
                           show  color       :  zeigt  die  aktuelle
                                               Vordergrundfarbe
tell turtle/turtleliste    ruft die 4 möglichen Turtles auf
     tell [0 1 2 3]
     st fd 50
     ---> bewegt alle 4 Turtles 50 Punkte nach oben

     tell 3
     st rt 90
     ---> dreht die 3. Turtle um 90°

     tell all
     --->  ruft alle auf

     show who
     ---> zeigt die aktuellen Turtles an, who  gibt  die  Zahl
          der Turtle aus

    tell all
    st
    each [repeat 4[fd 40 rt 90]]
    ---> läßt alle Schildkröten der Reihe nach den Befehl  in
         [ ] ausführen.

    tell [1 3]
    st
    each [seth (90 * who) fd 50 ]
    ---> multipliziert den Winkel von set heading mit der Zahl
         der aufgerufnen Turtles:
         1. Turtle dreht sich um 90° nach rechts
         3. Turtle dreht sich um 270° über rechts

tone Frequenz Zeit    produziert einen Ton mit der Freqeunz Frequenz
                      mit der Dauer von zeit* 1/20 Sekunde
                      tone 440 20  ---> Kammerton A eine Sekunde
wait Zeit             Pause mit der Dauer Zeit * 1/20 Sekunde
```

OPEN ACCESS II Vers. 2.05

W. Neidhart, BRG Spittal, Tel. 047 62 - 4084

Eine Beschreibung für die Benützer von Computer mit 2 Diskettenlaufwerken, sowie mind. 640 KB RAM

1.) KONFIGURATION der Disketten

OA II ist etwas umfangreicher als OA I und daher empfiehlt sich für die Verwendung mit 2 Diskettenlaufwerken folgende Anordnung:

a) Startdiskette : enthält
MS-DOS, Tbge, ramdisk 64, misce.spi, miscalle.spi, infoe.prt und eine autoexec.bat, die *.spi und *.prt nach C: kopiert.
Misce.spi enthält die komplette SPI-Konfiguration
Miscalle.spi enthält die Fehlermeldungen
Infoe.prt enthält die Druckertreiber, so z.B. den Epson FX-80, so daß man sich den DIO-55D.SYS ersparen kann.

b) SPI Diskette = "Hilfsprogrammdiskette" von SPI, wird nach der Startdiskette ins Laufwerk A: gegeben und startet mit der Eingabe von 'SPI' das Programm. Nach der Auswahl der betr. Option gibt man die entsprechende Diskette nach A:, in B: befindet sich eine formatierte, leere Arbeitsdiskette.

Man benötigt also mind. 4 Disketten :
- MS DOS mit ramdisk
- SPI Startdiskette
- optionale SPI Diskette (Datenbank, Text, Kalkulation etc.)
- Arbeitsdiskette

2.) KONFIGURATION von OA II

Nach dem Startaufruf wählt man im Optionenfenster : Hilfsprogramm - Konfiguration -> und erhält folgendes Menü zur Auswahl:

PARAMETER:

Standarddrucker : mit <f4> sucht man den passenden Drucker, dazu muß aber bereits die "Suchtabelle" definiert sein (s. u.)

Farbeinstellung : hier kann man mit <f6> zwischen verschiedenen Farben für den Bildschirm wählen.

Zwischendatei : hier wird festgelegt, wo OA II bei Bedarf Dateien zwischenlagert, am besten B:, bei genügend RAMdisk auch in C:.

SUCHTABELLE : Es empfiehlt sich mit der Ramdisk die Vorgabe von B:\ , C:\ , A:\ , d.h. OA II sucht zuerst im Laufwerk B: = Arbeitsadiskette, dann in der Ramdisk C: (= dorf findet es die Druckerdatei infoe.prt) und zuletzt in A:.

ZEITTABELLE : zur Steuerung des Desk managers (= <f8>) kann man hier beliebige Zeitzonen definieren, Vorgabe ist New York und San Diego.

FUNKTIONSTASTEN : hier kann man den Tasten <f1> - <f8> beliebige Zeichenfolgen zuordnen (siehe Pkt. 3)

DRUCKER : im Menüpunkt Drucker kann man beliebige Drucker-

konfigurationen für den Epson FX-80 verfassen (Falls der Drucker es überhaupt erlaubt !).

In der Option Hilfsprogramm gibt es noch folgende interessante Punkte:
DB3 => df/if : mit dierser Option kann man alte OA I Datenbanken nach OA II transferieren. DF steht für datafiles, IF steht für indexfiles, OA II trennt jetzt Datenbanken, daher ist eine Transferierung nach OA I NICHT mehr möglich.
ÜBERNEHME_DATEN: damit lassen sich fremde Textdateien (auch ASCII Dateien) einlesen, leider funktioniert in der Vers. 2.05 NICHT die Übernahme von dBase III Dateien. nach Auskunft von Philips ist dieser Fehler SPI bekannt, wird aber auch in der kommenden Version 2.11 nicht beseitigt. Wer noch die Vers. 2.03 zur Verfügung hat, kann dBase III Daten anstandslos übernehmen.
GRAPHIKTREIBER : von der Hilfsprogrammdiskette kann jetzt auch ein EGA Treiber geladen werden, nach meinen Dafürhalten ändert sich am Bildschirm nichts !!.

3.) OPEN ACCESS II TASTATUR

Im Unterschied zu OA I sind nun die folgenden Tasten immer gleich. Man kann mit der Taste <f1> jederzeit die Tastenbelegung und eine dazupassende Information abrufen.
<f2> : bringt einem immer das Hauptmenü
<f3> : spricht den Drucker an
<f4> : Auswahl,Suchen : ersteres in der Datenbank, im Inhaltsverzeichnis, zweiteres in der Textverarb.
<f6> : Ändern, sei es nun Dateinamen, Schriftart o.ä.
<f8> : Desk Manager Menü : Kalender, Uhr, Weltzeiten etc., beim 2.x drücken -> Taschenrechner.
<f10> : wie eh und je : do

Die Tasten f5, f7 und f9 sind von Option zu Option verschieden und sind mit <f1> jederzeit nachzuschauen.

NEU: die Tastenbelegung f1 - f8 mit frei definierbaren Zeichen. Der Aufruf erfolgt über Shift-<fx>, die Definition in der Option Hilfsprogramme, Konfiguration,Funktionstasten.

Neu: alt-<f4> : bringt jederzeit die ASCII Tabelle in ein Fenster, das ausgewählte Zeichen (ab ASCII 33) kann mit <f7> in die Anwendung übernommen werden, z.B. ≈ oder ± , leider scheitert alles am FX-80 !
Genauso kann die mit dem Taschenrechner ausgeführte Berechnung mit <f7> in den Text übernommen werden.
Der Aufruf von Macros erfolgt jetzt mit alt-<f8> und bringt als erstes ein Macrofensteer auf den Bildschirm.

Die Kombination von Crtl-Return bringt eine neue Zeile, die Kombination von Crtl-Backslash löscht eine Zeile. !! Funktioniert nicht mit DIO-55D.SYS, daher entfernen. Durch den Druckertreiber FX 80 ist man auch diese "Altlast" los.

4.) DATENBANK

Die Datenbank wurde völlig überarbeitet und entscheidend verbessert. Die Aufteilung in Hauptmenu I und II ist gefallen und dadurch wesentlich verständlicher geworden.

Man beginnt im Hauptmenu mit AUFBAU und gelangt in ein Untermenü zu ANLEGEN. Hier beginnt man mit dem Aufbau einer Schirmmaske, kenntlich an .SMK.

Die Felddefinition beginnt mit dem 2x-ligen Drücken von <f9> und es erscheint das Felddefinitionsfenster mit den bekannten Punkten. Wenn man auf die shift-<fx> Tasten Graphikzeichen geladen hat, kann man die Schirmmaske auch mit Rahmen u.ä. versehen.

Man kan bis zu 100 Schlüsselfelder definieren und die Schirmmaske ist 15 (!) Bildschirmseiten groß. Mit <PgDn> und <Pgup> kann man weiterblättern - damit ist die Datenbank irrsinnig groß.

Die größte Änderung gegenüber OA I betrifft aber die Möglichkeit, jede Datenbank jederzeit zu ÄNDERN. Man kann mit dem Menüpunkt AUFBAU - DATEI_ÄNDERN jede Schirmmaske frei ändern, dazu noch die Satzlänge u.ä.

Mit dem Menüpunkt EINGABE kann man jetzt Daten eingeben, die Anzahl der Datensätze ist nur durch die Diskettenkapazität begrenzt.

Der Menüpunkt PFLEGE erlaubt ein sortiertes Durchblättern der Datei, mit <f6> kann man jeden Datensatz anwählen/ändern.

Mit AUFBAU - DRUCKMASKE kann man sich zu jeder Datenbank eine Druckmaske (.PMK) anlegen. Der Text wird unterteilt in

Kopfseite
Gruppenseite
Satzseite
Fußseite
Gruppensummenseite
Summenseite und jede Seite kann 16 Zeilen Daten beinhalten.

Die Kopfseite enthält die Überschrift und vorgegeben sind Seitennummerierung (SEITENNR) und das aktuelle Datum (SYSDATUM). Eigenen Text gibt man mit <f9> ein und setzt ihn unter die einfachen Anführungszeichen ' ', Befehlswörter, wie SYSDATUM und FELDNAMEN etc. werden ohne Anführungszeichen geschrieben.

Gruppenseite : hier kann man seine Datensätze in "Gruppen" ausdrucken lassen, z. B. bei einer Schülerliste erklärt man das Feld KLASSE zum Gruppennamen (mit <f2> ins Druckmaskenmenü zu Punkt Gruppendefinition) und kann dann pro Seite eine Klasse ausdrucken. = erste Seite 1.A, zweite Seite 1.B etc.

Satzseite : enthält alle Felder in frei wählbarer Anordnung, mit <f9> kann ein Feld selektiert , mit den Cursortasten positioniert und mit <f10> fixiert werden.

Fußseite : Text für die letzten Zeilen einer Seite, oder SEITENNR

Gruppensummen: zu jeder def. Gruppe können Summen oder andere mathem. Berechnungen (MEAN, COUNT,MAX, MIN, SUM) durchgeführt werden.

Summenseite: wie in der Gruppensumme, nur hier für den gesamten Datensatz.

5.) TEXTVERARBEITUNG

Wegen der wesentliche verbesserten Textverarbeitung bin ich schon sehr früh auf OA II Vers. 2.03 umgestiegen, Vers. 2.05 bringt nur kleine Änderungen. Hingegen enthält die Version 2.11 ein Wörterbuch = Rechtschreibhilfe und eine Silbentrennung, sowie die Möglichkeit, in Spalten zu schreiben.

Das Hauptmenü enthält folgende Optionen:

LADEN : lädt vorhandene Dateien, für Suchen <f4>

ANLEGEN : für Schreiben neuer Dateien, dafür braucht man aber

STANDARD_ÄNDERN : eine der besten Änderungen gegenüber OA I. Da man ja üblicherweise nur wenige verschiedene Briefformate mit entsprechenden Parametern hat, kann man hier Standarddateien definieren und aufrufen. Man legt z. B. folgende Standarddateien an:

ST_BRIEF.DOC
ST_TEST.DOC
ST_SERIEN.DOC

Wenn man diese Dateien auf die Startdiskette kopiert und über die Autoexec.bat nach C: kopiert, kann man sie jederzeit aufrufen. (<f4> sucht die Dateien)

TYP_ÄNDERN :hier stellt man den gewünschten Dateityp ein. .DOC für Dokument mit allen Möglichkeiten (Parameter, Schriftarten etc.) und .TXT für reine ASCII Dateien.

Im Hauptmenü steht einem dann mit <f2> das Kommandomenü zur Verfügung. Neben der Zeilen- und Spaltenangabe in der linken unteren Ecke kann man mit SEITENUMBRUCH jederzeit den Umbruch zu einem gewünschten Drucker aufrufen. Alle anderen Optionen sind selbsterklärend.

ERSTELLUNG VON SERIENBRIEFEN:

Neu: für Serienbriefe muß man die Werte der Datenbank in eine eigene Exportdatei (*.DIF) schicken. Im Serienbrief sind die Felder mit ^@FELDNAME^ zu kennzeichnen, im Druckmenü ist dann die Option Serienbrief_Datei mit <f4> auszufüllen, bei einem einzelnen Serienbrief auch die Option Serienbrief_Datensatz.

Die Textverarbeitung stellt nur die Schrifttypen Fett, Unterstrichen und Kursiv zur Verfügung. Man kann jetzt für andere Schrifttypen eigene Druckerdateien definieren (z. B. fett= subskript, unterstrichen= superskript etc.), dadurch wird die Auswahl aber nicht größer. Elegant aber aufwendig ist die Definition eigener Steuerzeichen in der Druckerdatei unter dem Absatz Zeichenübersetzung -> dort stehen auch die Zeichenübersetzungen für den FX-80 !

6.) KALKULATION

Die Kalkulation ist im Prinzip gleich geblieben, es gibt einige praktische Erweiterungen. Nicht optimal ist die Einbindung der Graphik in die Kalkulation, dadurch ist es nicht mehr möglich, unabhängig von der Kalkulation eine Graphik zu erstellen.
Im gesamten OA II gibt es als Währungssymbol nur DM, ev. wäre es mit den Norton Ut. möglich, dies gegen öS auszutauschen.

7.) PROGRAMMIERER

Dieses Kapitel wurde von mir noch kaum bis gar nicht ausprobiert. Über die einfachen Beispiele der Demo-Dateien bin ich noch nicht hinausgekommen und kann daher darüber NICHTS sagen.

Da es sich hierbei nur um einen frei in die Tasten getippten Bericht handelt, ist mir die Unzulänglichkeit vollkommen klar. Wenn jemand dringend Rat oder Hilfe braucht, stehe ich gerne zur Verfügung. Meine Schultelefonnummer steht am Kopf des Berichtes, privat bin ich unter 0 47 62 - 46 5 42 bis mind. 22 Uhr zu erreichen.

Ich wünsche allen Benützern von OA II viel Erfolg.

W. Neidhart

*) ich schreibe mit OA II über OA II !!

Escape-Sequenzen in WORDSTAR und WORD

Josef MELCHART, TGM Wien

Moderne Drucker bieten etliche Funktionen (z.B. doppelt hohe, doppelt breite Schrift, Überstreichen), die von manchen Textverarbeitungsprogrammen nicht unterstützt werden.

Eine nicht sehr komfortable, aber brauchbare Möglichkeit bieten Escape-Sequenzen, die direkt in den Text geschrieben werden und den Drucker entsprechend steuern. Eine notwendige Voraussetzung dafür ist, daß das Textverarbeitungsprogramm diese Zeichen unverfälscht an den Drucker weiterleitet.

Ich habe WORDSTAR 4.0 und WORD 5.0 mit einem STAR LC-10 Schwarz-Weiß-Drucker auf diese Möglichkeit hin untersucht. Das Ergebnis vorweg: Es funktioniert wieder nur ein Teil. Ein Versuch lohnt sich trotzdem, an das "Handling" gewöhnt man sich rasch.

A) WORDSTAR 4.0

Vorgangsweise:

1) Eingabe des ESC-Zeichens:

Ein Versuch mit ALT-027 scheitert - Wordstar interpretiert dies wie einen Druck auf die Escape-Taste und antwortet mit dem Makro-Menü. Mit einem Trick gelingt es aber doch:

a) Einmalige Erstellung einer ASCII-Datei, die (nur) das ESC-Zeichen enthält: wahlweise

- mit WORD: ALT-027 eingeben und unformatiert abspeichern.

- mit EDLIN: CTRL-V und ALT-091 bzw. CTRL-V und [eingeben. Anzeige des ESC-Zeichens in EDLIN mit ^[.

- mit PCTOOLS oder NU (Norton Utilities): Hexcode eines beliebigen Zeichens (Dummy) mit der Funktion EDIT auf Hex 1B (=ESC-Zeichen) ändern.

b) Hineinkopieren dieser ASCII-Datei ins WORDSTAR-Dokument mit CTRL-K R (Block einlesen).

Anzeige des ESC-Zeichens in WORDSTAR mit ^[(Druckerzeichen mit CTRL-O D sichtbar schalten!)

c) Innerhalb des Textes ist ein Kopieren und Löschen leicht möglich: Das ESC-Zeichen mit CTRL-T (Wort löschen) oder mit CTRL-Y (Zeile löschen) in den "Papierkorb" löschen und mit CTRL-U (Undo) beliebig oft aus dem Papierkorb wieder herausholen (wie in WORD).

2) Eingabe der restlichen Zeichen der Escape-Sequenz:

Ganz normal als Buchstaben nach dem ESC-Zeichen in den Text schreiben.

Und hier tritt die leider gravierende Beschränkung auf: Es sind nur Escape-Sequenzen möglich, die aus ESC-Zeichen + DRUCKBAREN Zeichen bestehen.

Versuche mit nichtdruckbaren Zeichen waren mühsam und nicht von Erfolg gekrönt:

- WORDSTAR akzeptiert die nichtdruckbaren ASCII-Zeichen #0..#31 (Hex 00..1F) nicht als Eingabe in den Text. ALT-00..ALT-031 funktioniert nicht.

- Der Umweg über Dummy-Zeichen und eine nachträgliche Änderung mit PCTOOLS bzw. NU funktionierte auch nicht wie gewünscht - WORDSTAR "zensurierte" die nichtdruckbaren Zeichen beim Ausdruck.

Ergebnis:

Alle Escape-Sequenzen mit nichtdruckbaren Zeichen scheiden aus. Dennoch bleiben Verwendung sich lohnt:

Zum Beispiel für den STAR LC-10 Schwarz-Weiß-Drucker funktionieren in WO folgende Escape-Sequenzen:

```
<ESC>W1  Breitdruck ein
<ESC>W0  Breitdruck aus
<ESC>w1  Doppelt hohe Zeichen ein
<ESC>w0  Doppelt hohe Zeichen aus
<ESC>_1  Überstreichen ein
<ESC>_0  Überstreichen aus
```

Weiters können die Schriftarten auch innerhalb einer Zeile gewechselt werden:

```
<ESC>x1  Schönschrift ein (NLQ)
<ESC>x0  Schönschrift aus (Draft)
<ESC>P   Pica ein (10 Zeichen/Zoll
<ESC>M   Elite ein (12 Zeichen/Zoll
<ESC>p1  Proportionalschrift ein
<ESC>p0  Proportionalschrift aus
<ESC>9   Papierende-Sensor ein
<ESC>8   Papierende-Sensor aus
<ESC>U1  Unidirektionaler Druck ein
<ESC>U0  Unidirektionaler Druck aus
(Weitere ESC-Sequenzen siehe Druckerhandbuch.)
```

Nicht möglich sind leider zum Beispiel:

Vierfach hohe und breite Zeichen, Schmalschrift innerhalb einer Zeile ein/aus, W Zeichensatzes (z.B. Orator), beliebiger Zeilenabstand (in n/216 Zoll).

Serienmäßig ohnehin möglich sind z.B.:

Unterstreichen, Fettdruck, Doppeldruck, Kursivschrift.

Ein Tip:

Ich habe mir die am häufigsten verwendeten Escape-Sequenzen (für Doppelt-breit und Dopp hoch) als Kommentarzeilen (mit ".." am Zeilenanfang) in eine Initialdatei geschrieben, die ich CTRL-K R an den Beginn jedes neuen Dokumentes kopiere. Diese Initialdatei enthält weit Punktbefehle für ein Standard-Seitenformat. Für Endlospapier hat sich folgende Datei mit de Namen # bewährt:

```
.RR---!-----------------------------------------------------R
..Endlospapier: .pl 72
..              .mt 0  { bei n Kopfzeilen: .mt (n+1) }
..              .mb 7  { bei n Fußzeilen:  .mb (n+8) }
.pl 72
.mt 0
.mb 9  { bei 1 Fußzeile }
.po 6
..^[W1    ESC-W1 Breitschrift ein
..^[W0    ESC-W0 Breitschrift aus
..^[w1    ESC-w1 Hochschrift ein
..^[w0    ESC-w0 Hochschrift aus
.lq
```

```
.cw 12
.fo (Dateiname)                    - # -
```

Im aktuellen Dokument brauchen dann je nach Bedarf nur ein paar Einstellungen geändert bzw. unnötige Zeilen gelöscht werden. Das ESC-Zeichen ist durch Kopieren sofort verfügbar.

B) WORD 5.0

Punkto Zeichenformatierung läßt WORD ohnehin nicht viele Wünsche offen. Dennoch sind manche Druckeroptionen interessant, die WORD nicht ausnützen kann.

Die Eingabe nichtdruckbarer Zeichen mit ALT-nnn funktioniert nur für die ASCII-Zeichen #1..#8 und #16..#30 (Hex 01..08 und 10..1E). (Das ESC-Zeichen kann also direkt mit ALT-27 eingegeben werden und wird als Rückwärtspfeil dargestellt.)

Escape-Sequenzen, in denen nur diese Zeichen vorkommen, scheinen zu funktionieren.

Jedenfalls funktionieren folgende Escape-Sequenzen mit dem Star LC-10 Schwarz-Weiß-Drucker:

	Eingabe:	
<ESC>8	ALT-27 8	Papierende-Sensor aus
<ESC>9	ALT-27 9	Papierende-Sensor ein
<ESC>_1	ALT-27 _ 1	Überstreichen ein
<ESC>_0	ALT-27 _ 0	Überstreichen aus
<ESC>W1	ALT-27 W 1	Doppelt breit ein
<ESC>W0	ALT-27 W 0	Doppelt breit aus
<ESC>w1	ALT-27 w 1	Doppelt hoch ein
<ESC>w0	ALT-27 w 0	Doppelt hoch aus
<ESC>h<01>	ALT-27 h ALT-1	Doppelt hoch+breit
<ESC>h<02>	ALT-27 h ALT-2	Vierfach hoch+breit

<ESC>h<00> wäre das Zurückschalten auf normale Zeichengröße, aber das ASCII-Zeichen 00 wird von WORD nicht akzeptiert. Abhilfe: Zuerst auf doppelt breite und doppelt hohe Zeichen schalten und diese dann ausschalten:

<ESC>W1 <ESC>w1 <ESC>W0 <ESC>w0 Normale Zeichengröße

(Weitere Escape-Sequenzen siehe Druckerhandbuch.)

Interessant ist noch das ASCII-Zeichen 08 (Backstep), das ein mehrfaches Überdrucken ermöglicht:

<08> ALT-8 Nächstes Zeichen überdrucken

C) Resumee:

Die Verwendung von Escape-Sequenzen in WORDSTAR und WORD ist unkomfortabel, aber es gelingt damit, wenigstens ein paar besondere Funktionen des Druckers auszunützen.

Als Nachteile sind noch zu nennen:

- Das Bild ist nicht WYSIWYG (What You See Is What You Get), aber damit hat man sich ja schon fast abgefunden.

- Der Blocksatz funktioniert nicht mehr richtig, da die Escape-Sequenzen nicht als Steuerzeichen erkannt und mitgezählt werden.

Bei der Druckersteuerung heißt es also weiterhin "mit Mängeln leben".

TURBO-PASCAL 5.0 MENÜBEFEHLE

Josef MELCHART, TGM Wien

FILE

LOAD (F3): Laden einer Datei. Bei Angabe eines Laufwerks oder Wildcards (*,?) erscheint Verzeichnis.

PICK (ALT-F3): Weiterbearbeitung. Die letzten bearbeiteten Dateien werden gespeichert (samt letzter Cursorposition, Blockmarkierung, etc.) im Pick-File (Name in OPTIONS/DIRECTORIES/PICK-FILE einstellbar)

NEW: Arbeitsspeicher löschen, neue Pascal-Datei erstellen.

SAVE (F2): Abspeichern (gleicher Dateiname).

WRITE TO: Abspeichern unter neuem Dateinamen.
Zusätzlich kann ein Laufwerk oder Verzeichnis angegeben werden, z.B.
B:PRIMZAHL.PAS , C:\TP5\PRIMZAHL.PAS

DIRECTORY: Inhaltsverzeichnis anzeigen (wie DOS-Befehl DIR).

CHANGE DIRECTORY: aktuelles Verzeichnis ändern (wie CD).

OS SHELL: (Operating System Shell) MS-DOS aufrufen,
Hauptspeicherinhalt (Arbeitsdatei) bleibt erhalten.
Rückkehr mit EXIT.

QUIT (ALT-X): Aussteigen aus Turbo-Pascal.

EDIT

Arbeitsdatei editieren.

RUN

RUN (CTRL-F9): Programm laufen lassen. = MAKE (F9) + RUN.
Übergabe von Parametern in OPTIONS/PARAMETERS möglich.

PROGRAM RESET (CTRL-F2): Neustart beim Debugging. Gibt den vom Debugger belegten Hauptspeicher frei. Sollte nach DEBUG aufgerufen werden.

GO TO CURSOR (F4): Programmablauf bis zur Zeile unmittelbar vor dem Cursor. Achtung: Diese Zeile muß vom Programm tatsächlich erreicht werden!

TRACE INTO (F7): Zeilenweise Ausführung (markierte Programmzeile). Unterprogramme ebenfalls zeilenweise ausgeführt.

STEP OVER (F8): Zeilenweise Ausführung wie TRACE INTO (F7), aber Unterprogramme komplett ausgeführt (wie 1 Zeile).

USER SCREEN (ALT-F5): Umschalten zwischen Ausgabebildschirm (User Screen) und Editor (Integrated Development Environment = Integrierte Entwicklungsumgebung).

COMPILE

COMPILE (ALT-F9): Übersetzen (Datei im Editor).
(Mit DESTINATION einstellen, ob im Hauptspeicher oder als .EXE-Datei auf Diskette)

MAKE (F9): Übersetzen (Primary File bzw. Datei im Editor) + Neuübersetzung aller Dateien, auf die sich die Änderung auswirkt (Units .TPU, Include-Files auf der Diskette). = Update.

BUILD: Wie MAKE, aber alle zusammenhängenden Dateien neu übersetzt (egal, ob sich Änderung auswirkt oder nicht).

DESTINATION:
Memory --> Übersetzung im Hauptspeicher
Disk --> .EXE-File auf der Diskette

FIND ERROR: Anzeige eines Laufzeitfehlers (z.B. Division durch 0) als Adresse (Segment:Offset)
--> Debuginformation anwählen, Eingabe der Fehleradresse im Hex-Format, z.B. 2BE0:FFD4.

(1) Laufzeitfehler im Hauptspeicher: Fehleradresse automatisch gespeichert, mit CTRL-Q W anspringen.

(2) Laufzeitfehler unter DOS (in .EXE-Datei):
- Fehleradresse notieren
- Programm in Editor laden (oder als Primary File)
- Destination: Disk
- Fehleradresse eingeben.

PRIMARY FILE: welche .PAS-Datei mit MAKE (F9) oder BUILD (ALT-C B) übersetzt werden soll;
sonst: Datei im Editor übersetzt.

COMPILE (ALT-F9) übersetzt immer Datei im Editor.

GET INFO: Information über aktuelles Verzeichnis, Speicherbelegung, etc.

OPTIONS

COMPILER: Compiler-Optionen wählbar, gleiche Wirkung wie Compileranweisungen im Programm (z.B.
(*$I+*) oder {$I+} Ein/Ausgabe-Überprüfung,
(*$R+*) oder {$R+} Zahlenbereichsüberprüfung.

LINKER:

ENVIRONMENT: (Umgebung)
Backup source files: .BAK-File beim Speichern erstellen (ein/aus)
Edit auto save: automat. Abspeichern bei RUN
Config auto save: Editoreinstellungen automatisch speichern
Zoom windows: voller Bildschirm oder geteilt (Edit-, Output-, Watch-Window)
Screen Size: Anzeige 25 Zeilen (Standard)

```
                        43 Zeilen (EGA)
                        50 Zeilen (VGA)
Tab Size:       Tabulatorenabstand 2..16
                (mit CTRL-O T  Tab-Mode einschalten)

DIRECTORIES: Suchpfade (Laufwerk+Pfad) für die benötigten Dateien:
Turbo:      .TPL (Turbo Pascal Library), Config- und
            Help-Dateien
EXE & TPU:  .EXE und .TPU-Dateien
            (Turbo Pascal Unit = übersetzte Units)
Include:    Include-Dateien
Unit:
Object:     .OBJ (Assembler-) Dateien
Pick file name:     Pfad für Pick-File
Current pick file:  Pfad + Name des aktuellen
                    Pick-Files

PARAMETERS: Übergabe von Parametern (Argumenten) beim Programmstart
(entspricht in MS-DOS z.B.
WORD TEST.TXT  oder TYPE READ.ME)

SAVE OPTIONS: Abspeichern der Optionseinstellungen  auf ein .TP-File
(standardmäßig TURBO.TP)

RETRIEVE OPTIONS: Wiederherstellen einer Optionseinstellung von einem
.TP-File


DEBUG

EVALUATE (CTRL-F4):
(1) Werte von Variablen (Ausdrücken) kontrollieren und verändern.
(Werden nicht ständig angezeigt - Unterschied zu WATCH).
3 Kästchen (Umschalten mit Tab oder Pfeiltasten):
Evaluate:  Variable oder Ausdruck eingeben (automatisch
     wird Wort von Cursorposition genommen, weitere
     Buchstaben mit Rechter Pfeiltaste hereinkopieren).
Result:    zeigt Ergebnis.
New Value: neuen Wert eingeben (Bereichsprüfung aktiv).

(2) Taschenrechnerfunktion:
Ausdruck in Evaluate eingeben (z.B. 2*5/10.4) --> Ergebnis in Result.

Formatzusätze: für gewünschtes Ausgabeformat
    (Eingabe: Variablenname, Beistrich, Formatzusätze):
D  Decimals: Integerwerte dezimal anzeigen.
Fn Floating Point: n = 2..18 für Anzahl der angezeigten
      Stellen (incl. Vorkommastellen + Komma).
H  Hexadecimal: Integerwerte hexadezimal mit $...
M  Memory: Speicherdump einer Variablen (siehe auch S)
P  Pointer als seg:ofs anzeigen
      (sonst als Ptr(seg,ofs))
R  Record: Feldnamen+Werte anzeigen (sonst: nur Werte)
S  String: (zusammen mit M)
      Anzeige von ASCII 0..31 in der Form #0..#31

CALL STACK (CTRL-F3): Liste der erfolgten Unterprogrammaufrufe (mit
Parametern) anzeigen.
```

FIND PROCEDURE: Aufsuchen einer bestimmten Prozedur (Funktion) im geladenen Programm (incl. Units und Include-Dateien).
Vorher kompilieren!

INTEGRATED DEBUGGING:
ON: Breakpoints, Step, Trace möglich
(OPTIONS/COMPILER/DEBUG INFORMATION muß ON sein!)
OFF: Debugging nicht möglich (Hauptspeicherersparnis,
wenn COMPILE/DESTINATION DISK)

STANDALONE DEBUGGING:
OFF: (Normaleinstellung)
ON: Debuginformation wird an .EXE-File angeschlossen
(für Verwendung mit Turbo-Debugger)

DISPLAY SWAPPING: Umschaltung Editor --> Ausgabebildschirm:
SMART: nur bei Ausgabeanweisungen und
Unterprogrammaufrufen.
ALWAYS: bei jeder Anweisung (für Programme, die den
Editor-Bildschirm überschreiben).
NEVER: keine Umschaltung (für Programmteile ohne
Ausgabe).

REFRESH DISPLAY: Wiederherstellen des (irrtümlich überschriebenen) Editor-Bildschirmes.

BREAK/WATCH

"Watch" = Watch Expression = zu beobachtender Ausdruck:
Variable oder Ausdruck, dessen Wert während des Debuggens ständig im Watch-Fenster angezeigt werden soll. Wert nicht veränderbar!
(Veränderung nur mit EVALUATE (CTRL-F4) möglich)

Umschalten zwischen Watch- und Edit-Fenster mit F6 (Switch).

ADD WATCH (CTRL-F7): neue Watch-Variable (Ausdruck) hinzufügen.

DELETE WATCH: Watch-Variable (Ausdruck) löschen.

EDIT WATCH: Watch-Variable (Ausdruck) verändern.
(schnellere Möglichkeit: mit F6 ins Watch-Fenster,
gewünschten Ausdruck anwählen und verändern)

REMOVE ALL WATCHES: alle Watch-Variablen entfernen.

"Breakpoint" = gewünschte Unterbrechung des Programmablaufes vor der markierten Programmzeile.
Weiterlaufen des Programms mit CTRL-F9 (Run)

TOGGLE BREAKPOINT (CTRL-F8): Breakpoint setzen/löschen (Schalter) an der Cursorposition.

CLEAR ALL BREAKPOINTS: alle Breakpoints löschen.

VIEW NEXT BREAKPOINT: bewegt Cursor zum nächsten Breakpoint (ohne Befehlsausführung).

```
FUNKTIONSTASTEN

F1    Help                    ALT-F1    Last Help
F2    Save
F3    Load                    ALT-F3    Pick
F4    Go to Cursor
F5    Zoom   (Window)         ALT-F5    User Screen (Ausgabebildschirm)
F6    Switch (Window)         ALT-F6    Swap (Umschalten zwischen
F7    Trace                               den letzten 2 Dateien)
F8    Step
F9    Make (Update)           ALT-F9    Compile
F10   Menu                    ALT-X     Exit


CTRL-F1    Online Help (zur Cursorposition)

CTRL-F3    Call Stack
CTRL-F4    Evaluate


CTRL-F7    Add Watch
CTRL-F8    Toggle Breakpoint
CTRL-F9    Run


CTRL-BREAK (CTRL-SCROLL LOCK) (2x)    Programmabbruch
```

Papierende-Sensor beim STAR LC-10 Drucker

Josef MELCHART, TGM Wien

Dieser Sensor soll bei Papierende ein Weiterdrucken auf die Walze verhindern. Der Nachteil bei Einzelblatt-Betrieb ist, daß die Seite nicht vollgedruckt werden kann: Der Sensor unterbricht den Druck bereits 2,5 cm oberhalb des unteren Blattrandes.

Möglichkeiten der Abhilfe:

1) Papierende-Sensor hardwaremäßig ausschalten (DIP-Schalter 1-5 auf AUS stellen).

2) Papierende-Sensor softwaremäßig ausschalten (mit <ESC>8 aus, mit <ESC>9 wieder ein).

Nachteil: Irrtümlicher Druck auf Walze möglich.

3) Für sporadischen Einzelblattbetrieb gibt es eine einfache Möglichkeit, ohne den Papierende-Sensor ausschalten zu müssen:

Wenn der Sensor den Druck stoppt (mit einem penetranten Piepston), ein neues Blatt nachschieben und ONLINE drücken. Der Rest der Seite wird dann fertiggedruckt.

Mit diesem einfachen Trick kann die volle Seitenlänge ausgenützt werden.

Adressplan eines kompatiblen PC

00000-003FF Interrupt-Vektoren

BIOS-Vektoren

```
00 0000:0000 CPU   Divide error
01 0000:0004 CPU   Single Step, 6:Debug Exception
02 0000:0008 CPU   NMI
```

This function is called in the event of a memory parity error or may occur in the event of other hardware problems or failures (depends on the specific manufacturer's hardware). Displays the appropriate error message and halts the processor.

```
03 0000:000C CPU   Breakpoint
04 0000:0010 CPU   Overflow
05 0000:0014 BIOS  Print Screen
```

Send the present active display screen contents to the printer if the printer is idle and not out of paper. The status of the print screen function is in prn_screen_stat at 0050:0000. Control-Break will terminate an active print screen operation.

```
06 0000:0018 186   Invalid Opcode Exception
07 0000:001C 186   Co-Processor unavailable
08 0000:0020 BIOS  Timer Ticks-hardware 8259-1, IRQ 0
08 0000:0020 286   LIDT or Double Fault
```

This is the primary timer used to control the clock and other key system resources. It is called indirectly by channel A of the 8253 timer every 18.2 milliseconds. Every timer tick also calls int 1Ch for user needs (int 1Ch points to an iret instruction unless changed by a resident program). The timer interrupt is given the highest maskable interrupt priority upon power up. The main timing functions of int 8 include increment of a 32-bit time since powered on counter, timer_hi and timer_low, and after 24 hours since powered on, timer_rolled is incremented. The last task of int 8 is turning off the floppy drive motor after 2 seconds of non-use. The counter dsk_motor_tmr is decremented upon each occurrence of int 8. When the count reaches zero, the motor is set off, and the motor running flags are cleared in dsk_motor_stat.

```
09 0000:0024 BIOS  Keyboard, BIOS-hardware 8259-1, IRQ 1
09 0000:0024 286   Co-prozessor segment
```

When any key is pressed on the keyboard, the hardware calls this interrupt to service the pressed key or key combination. The hardware provides the key pressed in a non-ASCII scan code format read at i/o port 60h. The servicer acknowledges receipt of the key by toggling bit 7 of port 61h.(Port 61h should be read first, then bit 7 ored on, output to port 61h, then anded off, and resent to port 61h).

The read key is decoded to yield an ASCII character, special function key (such as F1) or a control function like Left Shift Key down. The converted ASCII character is placed into the next available position in the circular queue keybd_queue to keybd_q_end. It is put in the position indicated by keydb_q_tail when it will not cause the loss of earlier entered data. The value keybd_q_head points to the oldest key pressed in the buffer which has not been removed from the queue (the normal process uses int 16h to remove keys from the queue and return the key value to the int 16h caller).

The 16 word queue holds up to 16 keys. If keybd_q_head equals the keybd_q_tail, the queue is empty. Valid keys in the queue comprise the upper byte scan code and the lower byte ASCII character. If the key pressed has no ASCII equivalent (i.e F1 to F12), the lower byte is zero.

Toggle and shift keys are not placed in the buffer, but appear in 2 status registers keybd_flags_1 and keybd_flags_2.

Special key combinations will cause other events to occur:

Ctrl-Alt_Del	-	Reset computer by jumping to power_on_reset
Print screen	-	Call int_5_prn_scrn to print the current screen
Ctrl-Break	-	Call int_1Bh control break key processor (DOS)
Pause	-	Wait until an ASCII key is pressed, without placing the key in the queue

```
0A 0000:0028 BIOS  BIOS-hardware 8259-1, IRQ 2
0A 0000:0028 BIOS  LAN ADAPTER
```


On systems equipped with 2 interrupt controller chips (8259), IRQ 2 is used to support the second interrupt controller. In this case, int 71h (IRQ 9) is used to replace IRQ 2. Hardware calls to int 71h are redirected to this interrupt to maintain compatibility.

```
0B 0000:002C BIOS  Serial, BIOS-hardware 8259-1, IRQ 3
```

Called by the secondary serial port chip.

```
0C 0000:0030 BIOS  Serial, BIOS-hardware 8259-1, IRQ 4
```

Called by the primary serial port chip.

```
0D 0000:0034 BIOS  Disk, BIOS-hardware 8259-1, IRQ 5
0E 0000:0038 BIOS  Diskette, BIOS-hardware 8259-1, IRQ 6
0F 0000:003C BIOS  Printer, BIOS-hardware 8259-1, IRQ 7
10 0000:0040 BIOS  Video
10 0000:0040 286   Invalid TSS Exception
```

Called with: ah = primary function number

Returns: (unless otherwise indicated)
ax - altered (some systems may not alter ax)

Functions:

```
ah = 0    Set video display mode in al.
ah = 1    Set cursor size
            Call with:  ch = top line, 0-32
                        cl = bottom line, 0-32
ah = 2    Set cursor location
            Call with:  bh = page number, 0 = 1st page
                        dh = row (0 for top row)
                        dl = column (0 for leftmost)
ah = 3    Get cursor location
            Call with:  bh = page number, 0 = 1st page
            Returns:    ch/cl = cursor size (top & bottom)
                        dh/dl = row and column of cursor
ah = 4    Get light pen location
            Returns:    ah = 0 not on/unsupported & bx,cx,dx changed
                             1 status valid
                        bx = pixel column
                        cx = horizontal line number
                        dh/dl = row and column
ah = 5    Set Page number al, 0 = 1st page
ah = 6    Up scroll screen
            Call with:  al = # bottom lines to clear, set 0 for all
                        bh = attribute to fill cleared lines
                        ch/cl = row/column of top left scroll corner
                        dh/dl = row/column of bottom right scroll corner
ah = 7    Down scroll screen
            Call with:  al = # top lines to clear, set 0 for all
                        bh = attribute to fill cleared lines
                        ch/cl = row/column of top left scroll corner
                        dh/dl = row/column of bottom right scroll corner
ah = 8    Get character & attribute at cursor
            Call with:  bh = page number, 0 = 1st page
            Returns:    ah/al = attribute/character
ah = 9    Write character & attribute at cursor (graphics modes)
            Call with:  al = character
                        bh = page number, 0 = 1st page
                        bl = attribute, bit 7 = 1 to xor with old color
                        cx = Number of same characters to write
ah = 0Ah  Write character at cursor (graphics modes)
            Call with:  al = character
                        bh = page number, 0 = 1st page
                        cx = Number of same characters to write
ah = 0Bh  Set colors (typically in low res modes)
            Call with:  bh = 0, bl = color low res background, border
                        bh = 1, bl = 0/1 for low res color group 0/1
ah = 0Ch  Write graphics dot
            Call with:  al = color, bit 7 = 1 to xor bit with old color
                        cx/dx = pixel row/pixel column
ah = 0Dh  Read graphics dot
            Call with:  cx/dx = pixel row/pixel column
ah = 0Eh  Write in ASCII mode (cr, lf, bell, and bs as operators)
            Call with:  al/bl = character/color
ah = 0Fh  Get video info
            Returns:    ah = columns active
                        al = active video mode
```

```
                    bh = active page, 0 = 1st page
ah = 10h   Colors  (EGA/VGA)   Called with al = subfunction
             al = 1    bl = color register (0-15)  bh = color to set
             al = 2    es:dx = ptr to change all 16 colors & overscan #
             al = 3    bl = color high bit as: 0-intensity, 1-blink
           For VGA equipped systems only
             al = 7    bl = color register to get in bh
             al = 8    bh = returned overscan value
             al = 9    es:dx = ptr where to put all 16 colors & overscan
             al = 10h  bx = color register to set; ch/cl/dl = grn/blu/red
             al = 12h  es:dx = ptr to change color registers
                       bx = 1st register to set, cx = # of registers
             al = 13h  bl = 0, set color page mode in bh
                       bl = 1, set page;  bh = page number
             al = 15h  bx = color reg to read; ch/cl/dl = grn/blu/red
             al = 17h  es:dx = ptr where to load color regs
                       bx = 1st register to get, cx = # of registers
             al = 1Ah  get color page info; bl = mode; bh = page #
ah = 11h   Reset mode with new character set (leave memory as-is)
             al = 0    Character set load
                       es:bp = ptr to table of chars
                       bl/bh = which block/bytes per char
                       cx/dx = # of characters/where to start in block
             al = 1    bl = block to load the mono character set
             al = 2    bl = block to load the double wide character set
             al = 3    bl = block select related to attribute
             al = 4    bl = block to load the 8 by 16 set (VGA)
             al = 10h - 14h same as 0 - 4, must occur after setting mode
             al = 20h  es:bp = ptr to table of char, using int 1Fh ptr
             al = 21h  es:bp = ptr to table of char, using int 43h ptr
                       bl = 0-dl=# rows; 1-14 rows; 2-25 rows; 3-43 rows
                       cx = number of bytes per character in table
             al = 22h  use 8 by 14 char set, bl=rows (see al=21h)
             al = 23h  use double wide char set, bl=rows (see al=21h)
             al = 24h  use 8 by 16 char set, bl=rows (see al=21h)
           Get table pointer and info
             al = 30h  es:bp = ptr returned; cx=bytes per char; dl=rows
                       bh = 0, get int 1Fh       bh = 4, get double wide
                       bh = 1, get int 43h       bh = 5, get mono 9 by 14
                       bh = 2, get 8 by 14       bh = 6, get 8 by 16 (VGA)
                       bh = 3, get double wide   bh = 7, get 9 by 16 (VGA)
ah = 12h   Miscellaneous functions, bl = sub-function
             bl = 10h  Get info,  bh = 0-now color, 1-now mono
                       ch/cl = info bits/switches
             bl = 20h  Set print screen to work with EGA/VGA
           Functions for VGA only (bl = 30-34h return al = 12h)
             bl = 30h  Set # of scan lines, 0-200, 1-350, 2-400
                         activates after mode change
             bl = 31h  ax = 0/1  allow/prevent palette load with new mode
             bl = 32h  al = 0/1  video off/on
             bl = 33h  al = 0/1  gray scale summing off/on
             bl = 34h  al = 0/1  scale cursor size to font off/on
             bl = 35h  switch between adapter and motherboard video
                       al = 0, adapter off, es:dx = save state area
                       al = 1, motherboard on
                       al = 2, active video off, es:dx = save area
                       al = 3, inactive video on, es:dx = save area
             bl = 36h  al = 0/1  screen off/on
ah = 13h   Write character string (cr, lf, bell, and bs as operators)
             Call with:  al = 0/1 write characters, cursor not/is moved
                            bl = attribute of all characters
                         = 2/3 write characters, cursor not/is moved
                            string alternates character & attribute
                         bh = page number, 0 = 1st page
                         cx = number of characters
                         dh/dl = row and column to start
                         es:bp = ptr to character string
ah = 14h   LCD display support, al = sub-function
             al = 0,   es:di = ptr to font table to load
                       bl/bh = which block/bytes per char
                       cx/dx = # of characters/where to start in block
             al = 1,   bl = block # of ROM font to load
             al = 2,   bl = how to support high intensity
ah = 15h   LCD info table ptr returned in es:di
             ax = 0      there is no alternate adapter
             ax = 5140h  LCD is the alternate
             ax = 5153h  CGA is alternate
             ax = 5151h  MDA is alternate

                 **** VGA SUPPORT ONLY ****
```



```
ah = 1Ah   Get/Set types of displays which are operating
             al = 0, get displays, returns al = 1Ah
                     bl = in use display code
                     bh = alternate display code
             al = 1, set codes, returns al = 1Ah
                     bl = in use display code
                     bh = alternate display code
             codes:  0 = no display
                     1 = mono display
                     2 = CGA
                     4 = EGA, color
                     5 = EGA, mono
                     6 = PGA
                     7 = VGA, mono  (PS/2 MCA only)
                     8 = VGA, color (PS/2 MCA only)
                    Bh = VGA, mono  (PS/2 non MCA)
                    Ch = VGA, color (PS/2 non MCA)
                   FFh = display type not known
ah = 1Bh   Get video system information
             call with:  bx = 0
                         es:di = ptr to buffer where to put info
ah = 1Ch   Video system save & restore, al = sub-function
             call with:  al = 0, get buffer size
                         = 1, save system, buffer at es:bx
                         = 2, restore system, buffer at es:bx
                        cx bit 0 = 1 for hardware registers
                           bit 1 = 1 for software states
                           bit 2 = 1 for colors & DAC registers
```

11 0000:0044 BIOS Equipment
11 0000:0044 286 Segment not present

Returns: ax = equip_install word

```
 15  14  13  12  11  10   9   8
  \   /      game \          /
# of printer  port   # of RS-232
 ports 0-3    used    ports 0-4

 7   6   5   4   3   2   1   0
 \   /   \   /   \   /  Math  |
  # of   video mode   RAM    uP    no
 floppy  at boot up  00=16K        floppy
 drives  00=EGA      01=32K        drive
   1-4   01=CGA-40   10=48K        if 0
 if bit  10=CGA-80   11=64K
  0 = 1  11=MDA-80   (old PCs)

Note: bit 13 = modem on PC lap-tops
      bit  2 = mouse on PS/2
```

Regs used: ax

12 0000:0048 BIOS Memory Size
12 0000:0048 286 Stack Fault Exception

Returns: ax = number of contiguous 1K bytes blocks of memory
from 0 to 640 K.

Regs used: ax

13 0000:004C BIOS Disk Services
13 0000:004C 286 Protection Fault Exception

FLOPPY DISK SERVICES

Note: Floppy disk services may be vectored to this routine from int 40h on newer systems with hard disk support inside the BIOS.

Call with: ah = sub-function number
dl = drive number (unless otherwise noted), drive a: = 0

Returns: carry = 0 if function ok
ah = status (unless otherwise noted)
ah = 0 Function ok
ah = 1 Invalid value passed or unsupported function
ah = 2 Can not locate address mark
ah = 3 Floppy write protected

```
            ah = 4    Sector not found
            ah = 6    Floppy changed line on (1.2 meg drives)
            ah = 8    DMA overrun occurred
            ah = 9    DMA attempted across 64K byte boundary
            ah = 0Ch  Media type not found
            ah = 10h  CRC read error
            ah = 20h  Floppy controller failure
            ah = 40h  Seek operation failed
            ah = 80h  Floppy drive not ready

Functions:
  ah = 0    Floppy disk controller reset
  ah = 1    Get last status
  ah = 2    Read sectors
              Call with:  al = number of sectors to read
                          ch/cl = starting track/starting sector
                          dh = head number, head 0 = 0
                          es:bx = ptr to buffer where to put data
  ah = 3    Write sectors
              Call with:  al = number of sectors to write
                          ch/cl = starting track/starting sector
                          dh = head number, head 0 = 0
                          es:bx = ptr to buffer where to get data
              Return:     al = number of sectors written
  ah = 4    Verify sectors
              Call with:  al = number of sectors to compare
                          ch/cl = starting track/starting sector
                          dh = head number, head 0 = 0
                          es:bx = ptr to buffer where to compare data
              Return:     al = number of sectors checked
  ah = 5    Format track
              Call with:  al = number of sectors to format
                          ch/cl = starting track/starting sector
                          dh = head number, head 0 = 0
                          es:bx = ptr to table of address fields
  ah = 8    Get drive information (not supported by all systems)
              Returns:    ax/bh = 0          size dia.    size dia.
                          bl = drive type: 1=360K 5.25   3=720K 3.5
                                           2=1.2M 5.25   4=1.4M 3.5
                          cx bits 7&6, 15-8 = number of tracks, 0=1 track
                             bits 5-0 = number of sectors per track
                          dh/dl = # of heads/number of floppy drives
                          es:di = ptr to floppy parameter table
  ah = 15h  Get drive type (not supported by all systems)
              Returns:    ah = 0 no drive
                          ah = 1/2  floppy changed line not/is available
  ah = 16h  Get changed floppy status (not supported by all systems)
              Returns:    ah = 0   floppy in drive,     carry = 0
                          ah = 1   bad drive number,    carry = 1
                          ah = 6   floppy out of drive, carry = 1
                          ah = 80h drive not ready,     carry = 1
  ah = 17h  Specify media type for a drive
              Call with:  al = 1  use a 320/360K floppy in 360k drive
                          al = 2  use a 360K floppy in a 1.2M drive
                          al = 3  use a 1.2M floppy in a 1.2M drive
                          al = 4  use a 720K disk in a 720K drive
  ah = 18h  Prepare for format (not supported by all systems)
              Call with:  cx bits 7&6, 15-8 = number of tracks, 0=1 track
                             bits 5-0 = number of sectors per track
              Returns:    es:di = ptr to floppy parameter table

     HARD DISK SERVICES

Call with: ah = sub-function number (or with 80h for hard disk)
           dl = drive number (unless otherwise noted) 0 or 1
                set bit 7 to get return drive parameters

Returns:   carry = 0 if function ok
           ah = status (unless otherwise noted)
             ah = 0    Function ok
             ah = 1    Invalid value passed or unsupported function
             ah = 2    Can not locate address mark
             ah = 3    Write protected
             ah = 4    Sector not found
             ah = 5    Reset failure
             ah = 7    Parameter activity failed
             ah = 8    DMA overrun occurred
             ah = 9    DMA attempted across 64K byte boundary
             ah = 0Ah  Sector flag bad
             ah = 0Bh  Cylinder bad
```



```
             ah = 0Dh  Wrong # of sectors (format)
             ah = 0Eh  Detected control data address mark
             ah = 0Fh  DMA arbitration level has invalid range
             ah = 10h  CRC or EDC (Error Detect & Correct) has an
                         unresolvable error
             ah = 11h  Data corrected by EDC
             ah = 20h  Disk controller failure
             ah = 40h  Seek operation failed
             ah = 80h  Hard disk not ready
             ah = BBh  Error not defined
             ah = CCh  Write error
             ah = E0h  Error register is zero
             ah = FFh  Disk sense error
           If dl bit 7 was set to 1 on entry:
             cx bits 7&6, 15-8 = highest cylinder number
                bits 0-5 = highest sector number
             dh = highest head number
             dl = number of hard disks (1 or 2)

Functions:
  ah = 0    Hard disk controller reset
  ah = 1    Get last status
  ah = 2    Read sectors
              Call with:  al = number of sectors to read
                          cx bits 7&6, 15-8 = cylinder number
                             bits 0-5 = sector number
                          dh = head number, head 0 = 0
                          es:bx = ptr to buffer where to put data
  ah = 3    Write sectors
              Call with:  al = number of sectors to write
                          cx bits 7&6, 15-8 = cylinder number
                             bits 0-5 = sector number
                          dh = head number, head 0 = 0
                          es:bx = ptr to buffer where to get data
  ah = 4    Verify sectors
              Call with:  al = number of sectors to compare
                          cx bits 7&6, 15-8 = cylinder number
                             bits 0-5 = sector number
                          dh = head number, head 0 = 0
  ah = 5    Format cylinder (non-ESDI type disks)
              Call with:  cx bits 7&6, 15-8 = cylinder number
                          dh = head number, head 0 = 0
                          es:bx = ptr to bad sector map (non XT)
                          al = interleave number (XT only)
  ah = 6    Format cylinder & set bad sector flags (XT only)
              Call with:  al = interleave number
                          cx bits 7&6, 15-8 = cylinder number
                          dh = head number, head 0 = 0
  ah = 7    Format drive, begin at specific cylinder (XT only)
              Call with:  al = interleave number
                          cx bits 7&6, 15-8 = cylinder number
                          dh = head number, head 0 = 0
  ah = 8    Get drive parameters
              Returns:    cx bits 7&6, 15-8 = max cylinder number
                             bits 0-5 = max sector number
                          dh = max head number, head 0 = 0
  ah = 9    Set parameters for drive (ignored for ESDI drives)
              Call with   dh = 80h to use int 41h ptr, drive 0
                          dh = 81h to use int 46h ptr, drive 1
  ah = 0Ch  Disk seek
              Call with:  cx bits 7&6, 15-8 = cylinder number
                          dh = head number, head 0 = 0
  ah = 0Dh  Secondary Disk Reset
  ah = 10h  Check if drive is ready
  ah = 11h  Recalibrate drive
  ah = 15h  Get drive type (all but XT systems)
              Returns:    ah = 0 no drive (cx & dx = 0)
                          ah = 3 hard disk
                          cx:dx = size of disk in 512 byte blocks
                          al = 4  use a 720K disk in a 720K drive
  ah = 19h  Park disk heads (PS/2 systems only)
  ah = 1Ah  Format hard disk (ESDI only)
              Call with:  al = number of blocks in defect table (0=none)
                          cl bit 0 = 1 when no primary defect map
                             bit 1 = 1 when no secondary defect map
                             bit 2 = 1 allow logging found surface errors
                                       in the secondary defect map
                             bit 3 = 1 Do extended surface analysis
                             bit 4 = 1 interrupt after each cyldr format
```

14 0000:0050 BIOS Serial Ports Services
14 0000:0050 386 Page Fault exception

```
Call with: ah = sub-function number
           dx = communications number, com1 = 0

Returns:   ah = line status,  bit 0 = Data ready
                              bit 1 = Overrun error
                              bit 2 = Parity error
                              bit 3 = CRC framing error
                              bit 4 = Break detect
                              bit 5 = Xmit buffer register empty
                              bit 6 = Xmit shift out register empty
                              bit 7 = No response, ignore bits 0-6

           al = modem status, bit 0 = Change in Clear To Send
                              bit 1 = Change in Data Set Ready
                              bit 2 = Falling edge ring detect
                              bit 3 = Change in receive detect line
                              bit 4 = Clear to Send (CTS)
                              bit 5 = Data Set Ready (DTS)
                              bit 6 = Ring detected
                              bit 7 = Receive detect line

Functions:
  ah = 0    Reset the specified port
              Call with:  al serial port register values
                          bits 7   6   5   4   3   2   1   0
                               --baud-rate--  -Parity- Stop --Word--
                               000 =  110      bits   bit   size
                               001 =  150                   (bits)
                               010 =  300     00 = off  0=1
                               011 =  600     01 = odd  1=2  10 = 7
                               100 = 1200     10 = off      11 = 8
                               101 = 2400     11 = even
                               110 = 4800
                               111 = 9600
  ah = 1    Transmit a character
              Call with:  al = character to send (not changed on exit)
  ah = 2    Received a character
              Returns:    al = character received
  ah = 3    Get status
  ah = 4    Advanced Initialization (not supported on all systems)
              Call with:  al = 0/1  no break/break
                          bh = Parity,    0 = none     3 = stick odd
                                          1 = odd      4 = stick even
                                          2 = even
                          bl = 0/1 Stop bits set to 1/2
                          ch = Word size, 0 = 5 bits   2 = 7 bits
                                          1 = 6 bits   3 = 8 bits

                          cl = Baud rate, 0 =  110     5 =  2400
                                          1 =  150     6 =  4800
                                          2 =  300     7 =  9600
                                          3 =  600     8 = 19200
                                          4 = 1200
  ah = 5    Advanced port control (not supported on all systems)
              al = 0  Get modem register
                        Returns: bl bit 0 = Data Terminal Ready (DTR)
                                 bit 1 = Request To Send (RTS)
                                 bit 2 = Out 1
                                 bit 3 = Out 2
                                 bit 4 = Loop modem
              al = 1  Set modem register
                        Call with:  bl = value for register (see above)
```

15 0000:0054 BIOS General Services, Cassette

On old PCs this function is only used for cassette servicing. Additional functions have been added for advanced features with newer equipment.

```
Called with: ah = function code

Returns:     CF = 0 if successful
             CF = 1 if failure or function not supported

Functions:
  ah = 0    Cassette motor set on
  ah = 1    Cassette motor set off
  ah = 2    Read cx bytes from cassette, es:bx = ptr to load area
  ah = 3    Write cx bytes to cassette,  es:bx = ptr to read area
  ah = 0Fh  Disk format hook, called from disk format routines
              after a cylinder access completed. Called with:
              al = 1-undergoing surface analysis, 2-if formating
  ah = 21h  Error log (PS 2), al = 0 to read, al = 1 to write

                *** LAP TOP MACHINES ONLY - 40h to 44h ***
  ah = 40h  Systems info in cx, bx - al = 0 to read, 1 to write
              for modem info in bx - al = 2 to read, 3 to write
  ah = 41h  Wait for an event, dx = i/o port to read or use es:di
              as ptr to user event. al = type of event 0-4,11-14
              bh = event mask, bl = # of 55ms counts to timeout
  ah = 42h  Power off, al = 0 or 1 for mode of power off
  ah = 43h  Get system status in al
                  7     6     5     4     3     2     1     0
                low   extrn  bad   pwr up  modem  ports       LCD
              battery  pwr   time  alarm    on     on       missing
  ah = 44h  Modem power, al = 0 turn off,  al = 1 turn on

              *** FUNCTIONS IN RECENT BIOS VERSIONS ONLY ***
  ah = 4Fh  Keyboard input hook, called every keystroke with
              al = scan code. CF = 1 to change or use the scan
              code in al. CF = 0 to ignore the key.
  ah = 80h  Open device number bx, in process number cx.
  ah = 81h  Close device number bx, in process number cx.
  ah = 82h  Terminate device number bx
  ah = 83h  Wait for timeout, and set bit 15 at ptr es:bx when
              timed out. al = 0 to set wait period cx:dx uSec,
              al = 1 to stop timeout timer.
  ah = 84h  Game port read, dx = 0 to read switches into al
              high nibble,  dx = 1 to read linear position
              (Unit 1: ax = x, bx = y   Unit 2: cx = x, dx = y)
  ah = 85h  Get system request key state, al = 0 down, al = 1 up
  ah = 86h  Wait cx:dx microseconds before return
  ah = 87h  Transfer cx words to/from memory above 1 meg
              es:si ptr to table with source & destination ptrs:
              es:si -> zeros          db  16dup(0)
                       src_seg_limit  dw  ?       ; 1-64K bytes
                       scr_ptr_low    dw  ?       ; 0-64K range
                       scr_ptr_high   db  ?       ; 0-16 Meg range
                       scr_rights     db  93h     ; r/w access
                       scr_zero       dw  0
                       dst_seg_limit  dw  ?       ; 1-64K bytes
                       dst_ptr_low    dw  ?       ; 0-64K range
                       dst_ptr_high   db  ?       ; 0-16 Meg range
                       dst_rights     db  93h     ; r/w access
                       dst_zero       dw  0
                       zeros2         db  16dup(0)
              Notes:
                cx = 32K words max xfer, ptr_low & ptr_high form
                a 24 bit physical address (do not use a segment)
              Return codes: al = 0 if ok, al = 1 parity error
                al = 2 other errors, al = 3 gate addr line 20 bad
  ah = 88h  Get extended memory size (above 1 Meg boundary)
              ax = (number of bytes / 1024)
  ah = 89h  Go to protected mode of operation. es:si ptr to table
              comprised of 8 groups of 8 byte ptr sub-arrays. Each.
              group is formed from 8 bytes as follows:
                 seg_limit    dw  ?    ; 1-64K bytes
                 ptr_low      dw  ?    ; 0-64K range
                 ptr_high     db  ?    ; 0-16 Meg range
                 rights       db  93h  ; read/write access
                 zero         dw  0    ; set to zero
              Ptr_low & high form a 24 bit physical address.

              The 8 groups are as follows:
                es:si ->  Group_1   Set to all zero
                          Group_2   ptr to beginning of Group_1
                          Group_3   ptr to Interrupt Descript Table IDT
                          Group_4   ptr to present DS segment
                          Group_5   ptr to present ES segment
                          Group_6   ptr to present SS segment
                          Group_7   ptr to present CS segment
                          Group_8   ptr where to jump to when done
              Also: bh & bl set as offset from beginning of IDT to
                spot where 1st & 2nd set of 8 hardware interrupts begin.
              Returns ah = 0 if ok, regs changed: ax,bp,ds,es,ss,cs
  ah = 90h  Device busy - used to signal DOS when it must wait.
              Type   0  Disk timeout          0FCh  Hard disk timeout
```

```
            codes  1  Floppy timeout                  (PS 2 only)
            (al)   2  Keyboard no timeout    OFDh  Floppy motor timeout
                   3  Mouse timeout          OFEh  Printer timeout
                  80h  Local area network (es:bx = network cntrl block
            Returns CF = 0 if failure, 1 if minimum wait time occurred
  ah = 91h  Interrupt done - used to signal DOS when hardware done
            al = type code (Device busy type codes)
  ah = C0h  Get ROM system information table ptr in es:bx. See data
            area starting at byte "sys_info_size".

             *** PS 2 & COMPATIBLES ONLY - C1h to C4h ***
  ah = C1h  Get Extended BIOS data area segment in es
  ah = C2h  Mouse support (BIOS). Sub-function code in al
              al = 0  if bh = 0 mouse off, bh = 1 mouse on
              al = 1  reset mouse & parameters, returns bh = 0
                      bl altered on return
              al = 2  Set number of updates per second in bh,
                        bh = 0 - 10 updates/sec  4 - 80  updates/sec
                             1 - 20 updates/sec  5 - 100 updates/sec
                             2 - 40 updates/sec  6 - 200 updates/sec
                             3 - 60 updates/sec
              al = 3  Set resolution in counts per millimeter:
                        bh = 0 - 1 cnts per mm   2 - 4 cnts per mm
                             1 - 2 cnts per mm   3 - 8 cnts per mm
              al = 4  Get device type, bh = 0
              al = 5  Initialize pointing device interface data size
                        bh = size 1 to 8 (1 to 8 bytes)
              al = 6  Additional sub-functions in bh:
                        bh = 0  Get status, cl = cnts per mm (see al=3)
                                dl = updates/sec in hex, bl = status:
                                     bit 6 = mode (0=stream, 1=remote)
                                         5 = 0 disabled, 1 enabled
                                         4 = scaling (0=1:1, 1=2:1)
                                         2 = Left button pressed
                                         0 = right button pressed
                        bh = 1  Set scaling to 1:1
                        bh = 2  Set scaling to 2:1
              al = 7  Set location to "call far" when data available
                      in es:bx
            Return code for all sub-functions in al:
              al = 0  successful             3  interface error
                   1  bad function call      4  resend
                   2  invalid input          5  no far call installed
  ah = C3h  Watchdog timer mode al = 0 disable, al = 1 enable,
            bx = set counter value 1 to FFh
  ah = C4h  Bus option select (POS) subfunction in al,
              al = 0  Get base POS adapter register address in dx
                   1  Enable setup for slot bl
                   2  Adapter enable
```

16 0000:0060 BIOS Keyboard-Services
16 0000:0060 286 Coprozessor error Exception

```
Call with:  ah = sub-function number

Functions:
  ah = 0    Get a key, and wait until one is available if none are
            in the queue.
              Returns:    ah = keyboard scan code
                          al = ASCII character, or 0 if non-ASCII
  ah = 1    Get key status
              Returns:    zero flag = 0 if a valid key is in the queue
                            ah = keyboard scan code
                            al = ASCII character, or 0 if non-ASCII
                          zero flag = 1 if no keys in the queue
  ah = 2    Get shift status register
              Returns:    al = bits  7    6    5    4    3    2    1    0
                                     right left ctrl alt scrl  num  cap insrt
                                     -------shifts------ ------toggles------
                                        1 = down          1 = on
  ah = 3    Set repeating character rate (not supported in all systems)
              Call with:  al = 5
                          bh = start delay, 0 = 250 ms    2 =  750 ms
                                            1 = 500 ms    3 = 1000 ms
                          bl = speed in characters per second - values
                               range from 0 = 30 cps, to 1Fh = 2 cps
  ah = 5    Load keyboard queue (not supported in all systems)
              Call with:  ch/cl = scan code/character to load into the queue
              Returns:    al = 0 if ok, 1 if keyboard queue is full
```



```
             **** Support for extended keyboard functions ****
  ah = 10h  Get a key (similar to ah = 0, not supported in all systems)
  ah = 11h  Get key status (similar to ah = 1, not supported in all systems)
  ah = 12h  Get shift info (similar to ah = 2, not supported in all systems)
```

17 0000:0060 BIOS Printer-Services

```
Call with:  ah = function code
            dx = printer number 0-2 (some systems allow 0-3)

Returns:    ah = status bits
                  7      6      5      4      3     2 1     0
                 not  acknow-  no   select  I/O  unused   no
                 busy  ledge  paper         error        response
                 _________from printer_____________

Functions:
  ah = 0  Send character to printer, al = character
  ah = 1  Printer port initialization
  ah = 2  Get printer status in ah
```

18 0000:0060 BIOS ROM Basic

Note: ROM basic points to F600 segment, and has been adjusted to segment F000 to process in-line with the balance of the BIOS code. Internal data references may shift to segment F600 depending on implementation.

19 0000:0064 BIOS Bootstrap-Loader

Floppy drive a: attempts a read from the first sector of the disk. If read properly (i.e a boot disk is in drive a:), control is transferred to the loaded program. The boot sector at track 0, sector 1 is transfered to memory at 0:7C00h. dh = drive which was used (0=A:). Control is passed to 0:7C00h.

1A 0000:0068 BIOS Time of Day

```
Call with:  ah = sub-function number

Functions:
  ah = 0    Get system timer (increments every 54.92 ms from int 8)
              Returns:    al = # of 24 hour periods since read/powered up
                          cx:dx = 32 bit count
  ah = 1    Get system timer & reset 24 hour counter
              Returns:    cx:dx = 32 bit count

             **** System must have CMOS clock for services 2-0Bh ****
  ah = 2    Get time in BCD format
              Returns:    ch/cl = hours/minutes
                          dh = seconds
                          dl = 0/1 if daylight savings time operation off/on
                          carry flag = 0 if clock ok, 1 if stopped
  ah = 3    Set time in BCD format
              Call with:  ch/cl = hours/minutes
                          dh = seconds
                          dl = 0/1 if daylight savings time operation off/on
  ah = 4    Get date in BCD format
              Returns:    cx = four digit year
                          dh/dl = month/day
                          carry flag = 0 if clock ok, 1 if stopped
  ah = 5    Set date in BCD format
              Call with:  cx = four digit year
                          dh/dl = month/day
  ah = 6    Set 24 hour alarm (alarm vectors to int 4Ah)
              Call with:  ch/cl = hours/minutes
                          dh = seconds
  ah = 7    Clear alarm
  ah = 9    Get alarm time & status
              Returns:    ch/cl = hours/minutes
                          dh = seconds
                          dl = 0/1 alarm off/on
  ah = 0Ah  Get days counter
              Returns:    cx = number of days since 1-Jan-80
  ah = 0Bh  Set days counter
              Call with:  cx = number of days since 1-Jan-80
```

1B 0000:006C BIOS KEYBOARD BREAK (when Control-Break pressed)
1C 0000:0070 BIOS Timer-Ticks - Called every 18.2 ms
1D 0000:0074 BIOS Video Initialisation
1E 0000:0078 BIOS Floppy Disk Parm Table Ptr
1F 0000:007C BIOS CGA Graphic Char Font

MSDOS-Vektoren, BIOS, EGA/VGA, Harddisk

```
20 0000:0080 DOS    Program Terminate
21 0000:0084 DOS    Function Call
22 0000:0088 DOS    Terminate Address
23 0000:008C DOS    Ctr-Brk Exit Address
24 0000:0090 DOS    Fatal Error Vektor
25 0000:0094 DOS    Absolute Disk Read
26 0000:0098 DOS    Absolute Disk Write
27 0000:009C DOS    Terminate
28 0000:00A0 DOS    Idle Signal
29 0000:00A4 DOS    TTY Output
2A 0000:00A8 DOS    MS-Net services
2B 0000:00AC
2C 0000:00B0
2D 0000:00B4
2E 0000:00B8
2F 0000:00BC DOS    Print Spool
30 0000:00C0 DOS    Long Jump Interface
31 0000:00C4
32 0000:00C8
33 0000:00CC DOS    Mouse Functions
34 0000:00D0
35 0000:00D4
36 0000:00D8
37 0000:00DC
38 0000:00E0
39 0000:00E4
3A 0000:00E8
3B 0000:00EC
3C 0000:00F0
3D 0000:00F4
3E 0000:00F8
3F 0000:00FC
40 0000:0100 BIOS   Hard Disk Chain
41 0000:0104 BIOS   Hard Disk #1 Parm Table Ptr
42 0000:0108 BIOS   EGA Chain
43 0000:010C BIOS   EGA Parm Table Ptr
44 0000:0110 BIOS   EGA Graphics Char Font
45 0000:0114
46 0000:0118 BIOS   Hard Disk #2 Parm Table Ptr
47 0000:011C
48 0000:0120
49 0000:0124
4A 0000:0128 BIOS   AT Alarm Exit Address
4B 0000:012C
4C 0000:0130
4D 0000:0134
4E 0000:0138
4F 0000:013C
50 0000:0140 BIOS   AT Alarm Interrupt
51 0000:0144 BIOS   Mouse Functions
52 0000:0148
53 0000:014C
54 0000:0150
55 0000:0154
56 0000:0158
57 0000:015C
58 0000:0160
59 0000:0164
```



```
5A 0000:0168 NET    Functions
5B 0000:016C NET    Boot Chain
5C 0000:0170 NET    NetBios entry
5D 0000:0174
5E 0000:0178
5F 0000:017C
```

Anwender-Vektoren

```
60 0000:0180
61 0000:0184
62 0000:0188
63 0000:018C
64 0000:0190
65 0000:0194
66 0000:0198
67 0000:019C DOS    EMS-Functions
68 0000:01A0
69 0000:01A4
6A 0000:01A8
6B 0000:01AC
6C 0000:01B0
6D 0000:01B4 VGA    Suspected VGA Service
6E 0000:01B8
6F 0000:01BC
70 0000:01C0 BIOS   Real-Time Clock, AT-hardware 8259-2, IRQ 8
```

This interrupt services the real-time clock hardware. The hardware supports 2 modes of operation, an interrupt at a specific 24 hour interval (i.e 9:42 am), or repeatedly every 0.976 ms (1,024 Khz). Both modes can operate at the same time if needed.

In the 24 hour alarm mode, the interrupt is vectored here by hardware and interrupt 4Ah is called to alert the application program of the alarm. Int 4Ah is not handled by the BIOS other than to return, and is normally revectored by a particular application using the alarm.

When repeating interrupt mode is active, the 32-bit microsecond counter consisting of timer_clk_low and timer_clk_hi is decremented by 976 us on every interrupt. When the timer reaches zero, the byte pointed to by the offset @timer_wait_off and and segment @timer_wait_seg is set to 80h (this pointer is set by an application program through int 1Ah function ah=6).

```
71 0000:01C4 BIOS   redirected to IRQ2, ATBIOS-hardware 8259-2, IRQ
9
72 0000:01C8 BIOS   unassigned, ATBIOS-hardware 8259-2, IRQ 10
73 0000:01CC BIOS   unassigned, ATBIOS-hardware 8259-2, IRQ 11
74 0000:01D0 BIOS   unassigned, ATBIOS-hardware 8259-2, IRQ 12
75 0000:01D4 BIOS   ATBIOS-hardware 8259-2, IRQ 13
75 0000:01D4 287    287-MATH CO-PROCESSOR
```

The math co-processor 80287 invokes this interrupt. Int 75h calls the non-maskable interrupt int 2 to halt the system (80287 is not used if this vector is left pointing here). Programs which use the 80287 must re-vector this interrupt to use the 80287.

```
76 0000:01D8 BIOS   HARD DISK, ATBIOS-hardware 8259-2, IRQ 14
```

When the hard disk controller has completed its task, it signals completion though hardware activation of int 76h. The status in hdsk_int_flags is set to "done", a value of OFFh. Interrupt 15, function 91h may also be called to signal the interrupt is done.

```
77 0000:01DC BIOS   unassigned, ATBIOS-hardware 8259-2, IRQ 15
78 0000:01E0
79 0000:01E4
7A 0000:01E8
7B 0000:01EC
7C 0000:01F0
7D 0000:01F4
7E 0000:01F8
```

```
7F 0000:01FC
80 0000:0200 - F0 0000:03C0 BASIC
```

Nicht verwendet

```
F1 0000:03C4 - FF 0000:03FC not used, Bios-Stack, User-Vectors
```

00400-004FF BIOS-Datensegment

```
0000:0400 == 0040:0000
0040:0000 dw @rs232_port_1
0040:0002 dw @rs232_port_2
0040:0004 dw @rs232_port_3
0040:0006 dw @rs232_port_4
0040:0008 dw @prn_port_1
0040:000A dw @prn_port_2
0040:000C dw @prn_port_3
0040:000E dw BIOS_data_seg

PS/2:Extended BIOS data pointer
PC,XT,AT & compatible:Printer 4

0040:0010 db Equipment installed info bits (lower)

   7  6    5   4   3  2  1  0
    /     \   /   \  / Math  |
        video mode  RAM    uP  no
disk-       at boot up  00=16K    dsk
ettes       00=EGA      01=32K    driv
 1-4        01=CGA-40   10=48K    if 0
if bit      10=CGA-80   11=64K
0 = 1       11=MDA-80   (old PCs)

0040:0011 db Equipment installed info bits (higher)

15  14  13   12   11  10   9   8
\   /   -  game   0   \  /   -
# of print  port      #of RS-232
ports 0-3   used      ports 0-4
Note: bit 13=modem on PC lap-tops bit  2=mouse on PS/2

0040:0012 db init_test_flag
0040:0013 dw main_ram_size Base memory size 0-1Meg, 1K steps
0040:0015 dw chan_io_size
```

KEYBOARD DATA

```
0040:0017 db keybd_flags_1 Keyboard flag bits

  7   6   5   4    3   2   1   0
ins- cap num scrl alt ctl lef rig
sert --toggles--- --shifts down--

0040:0018 db keybd_flags_2 Keyboard flag bits

   7      6     5     4       3      2     1      0
insert  caps  num  scroll pause  sys   left right
------nov depressed------ lock request -alt-down-

0040:0019 db keybd_alt_num

Alt & digit pad number buffr area

0040:001A dw keybd_q_head

Head ptr of circular key queue empty if head ptr = tail ptr

0040:001C dw keybd_q_tail
```



```
Tail ptr of circular key queue empty if head ptr = tail ptr

0040:001E dw 16 keybd_queue key queue for keyboard
```

DISK DATA

```
0040:003E db dsk_recal_stat

Recalibrate floppy drive bits
   3       2       1       0
drive-3 drive-2 drive-1 drive-0

bit 7 = interrupt flag

0040:003F db dsk_motor_stat

Motor running status & disk write
bit 7=1 disk write in progress
bits 6&5 = drive selected 0 to 3
   3       2       1       0
drive-3 drive-2 drive-1 drive-0
--------- 1=motor on-----------

0040:0040 db dsk_motor_tmr

Motor timer, at 0, turn off motor

0040:0041 db dsk_ret_code

Controller return code
00h = ok
01h = bad command or parameter
02h = can't find address mark
03h = can't write, protected dsk
04h = sector not found
08h = DMA overrun
09h = DMA attempt over 64K bound
10h = bad CRC on disk read
20h = controller failure
40h = seek failure
80h = timeout, no response

0040:0042 db dsk_status_1
0040:0043 db dsk_status_2
0040:0044 db dsk_status_3
0040:0045 db dsk_status_4
0040:0046 db dsk_status_5
0040:0047 db dsk_status_6
0040:0048 db dsk_status_7

Status bytes-disk controller chip       dsk_ctrl_stat
Note: 7 info bytes returned from controller are saved here. Refer to the NEC uPD 765 chip manual for the specific
info, depending on the previous command issued.
```

VIDEO DATA

```
0040:0049 db video_mode

Present display mode(see int 10h)       video_mode

0040:004A dw video_columns
0040:004C dw video_buf_siz

Video buffer size in bytes
Note: size may be rounded up to the nearest 2K boundary. For example, 80x25 mode=4000 bytes, but value may be
4096.

0040:004E dw video_segment      MDA=0B000h, CGA=0B800h, etc.
0040:0050 dw vid_curs_pos0      bits 15-8=row, bits 7-0=column
0040:0052 dw vid_curs_pos1      bits 15-8=row, bits 7-0=column
0040:0054 dw vid_curs_pos2 bits 15-8=row, bits 7-0=column
```

```
0040:0056 dw vid_curs_pos3 bits 15-8=row, bits 7-0=column
0040:0058 dw vid_curs_pos4 bits 15-8=row, bits 7-0=column
0040:005A dw vid_curs_pos5 bits 15-8=row, bits 7-0=column
0040:005C dw vid_curs_pos6 bits 15-8=row, bits 7-0=column
0040:005E dw vid_curs_pos7 bits 15-8=row, bits 7-0=column
0040:0060 dw vid_curs_mode

Active cursor, start & end lines
bits 12 to 8 for starting line
bits 4  to 0 for ending line

0040:0062 db video_page
0040:0063 dw @video_port Video controller base I/O address
0040:0065 db video_mode_reg Hardware mode register bits
0040:0066 db video_color Color set in CGA modes
```

GENERAL DATA

```
Note: next 5 bytes also used for cassette interface in older PCs.

0040:0067 dw @gen_io_ptr ROM initialization pointer
0040:0069 dw @gen_io_seg ROM i/o segment
0040:006B db gen_int_occured Unused interrupt occurred
0040:006C dw timer_low Timer, low word, cnts every 55 ms
0040:006E dw timer_hi Timer, high word
0040:0070 db timer_rolled

Timer overflowed, non-zero when more than 24 hours have elapsed

0040:0071 db keybd_break Bit 7 set if break key depressed
0040:0072 dw warm_boot_flag

Boot (reset) type
1234h=warm boot, no memory test
4321h=boot & save memory (PS/2 with MCA only)
```

HARD DISK DATA

```
0040:0074 db hdsk_status_1

Hard disk status
00h = ok
01h = bad command or parameter
02h = can't find address mark
03h = can't write, protected dsk
04h = sector not found
05h = reset failure
07h = activity failure
08h = DMA overrun
09h = DMA attempt over 64K bound
0Ah = bad sector flag
0Bh = removed bad track
0Dh = wrong # of sectors, format
0Eh = removed control data addr mark
0Fh = out of limit DMA arbitration level
10h = bad CRC or ECC, disk read
11h = bad ECC corrected data
20h = controller failure
40h = seek failure
80h = timeout, no response
AAh = not ready
BBh = error occurred, undefined
CCh = write error, selected dsk
E0h = error register = 0
FFh = disk sense failure

0040:0075 db hdsk_count Number of hard disk drives
0040:0076 db hdsk_head_ctrl Head control (XT only)
0040:0077 db hdsk_ctrl_port Hard disk control port (XT only)
```


PORT TIMER, KEYBOARD DATA

```
0040:0078 db prn_timeout_1

down timer waits for printer to respond (printer 1)

0040:0079 db prn_timeout_2

down timer waits for printer to respond

0040:007A db prn_timeout_3
0040:007B db prn_timeout_4
0040:007C db rs232_timeout_1

Countdown timer waits for RS-232 port to respond (port 1)

0040:007D db rs232_timeout_2
0040:007E db rs232_timeout_3
0040:007F db rs232_timeout_4
0040:0080 dw @keybd_begin Ptr to beginning of keybd queue
0040:0082 dw @keybd_end Ptr to end of keyboard queue
```

ADVANCED VIDEO DATA, EGA/VGA

```
0040:0084 db video_rows Rows of characters on display - 1
0040:0085 dw video_pixels Number of pixels per charactr * 8
0040:0087 db video_options

Display adapter options
bit 7 = clear RAM
bits 6,5 = memory on adapter
   00 - 64K
   01 - 128K
   10 - 192K
   11 - 256K
bit 4 = unused
bit 3 = 0 if EGA/VGA active
bit 2 = wait for display enable
bit 1 = 1 - mono monitor
      = 0 - color monoitor
bit 0 = 0 - handle cursor, CGA

0040:0088 db video_switches

Switch setting bits from adapter
bits 7-4 = feature connector
bits 3-0 = option switches

0040:0089 db video_1_reservd

Video reserved 1, EGA/VGA control
bit 7 = 200 line mode
bits 6,5 = unused
bit 4 = 400 line mode
bit 3 = no palette load
bit 2 = mono monitor
bit 1 = gray scale
bit 0 = unused

0040:008A db video_2_reservd
```

OTHER FLOPPY & HARD DISK DATA

```
0040:008B db dsk_data_rate

Last data rate for diskette
bits 7 & 6 = 00 for 500K bit/sec
           = 01 for 300K bit/sec
           = 10 for 250K bit/sec
bits 5 & 4 = step rate
```

```
0040:008C db          hdsk_status_2
0040:008D db          hdsk_error
0040:008E db          hdsk_int_flags
0040:008F db hdsk_options
```

Bit 0 = 1 when using 1 controller card for both hard disk & floppy

```
0040:0090 db hdsk0_media_st Media state for drive 0
0040:0091 db hdsk1_media_st Media state for drive 1
```

```
  7     6     5     4     3     2     1     0
data xfer rate  two   media  unused -----state of drive-----
00=500K bit/s  step  known bits floppy  drive state
01=300K bit/s              000=  360K in 360K, ?
10=250K bit/s              001=  360K in 1.2M, ?
                           010=  1.2M in 1.2M, ?
                           011=  360K in 360K, ok
                           100=  360K in 1.2M, ok
                           101=  1.2M in 1.2M, ok
                           111=  state not defined
```

```
0040:0092 db hdsk0_start_st Start state for drive 0
0040:0093 db hdsk1_start_st Start state for drive 1
0040:0094 db hdsk0_cylinder Track number for drive 0
0040:0095 db hdsk1_cylinder Track number for drive 1
```

ADVANCED KEYBOARD DATA

```
0040:0096 db keybd_flags_3
```

```
Special keyboard type and mode
bit 7 Reading ID of keyboard
    6 last char is 1st ID char
    5 force num lock
    4 101/102 key keyboard
    3 right alt key down
    2 right ctrl key down
    1 E0h hidden code last
    0 E1h hidden code last
```

```
0040:0097 db          keybd_flags_4 Keyboard Flags (advanced keybd)
```

```
  7     6     5     4 3 2 1 0
xmit   char  Resend Ack   \ /
error was ID Rec'd Rec'd  LEDs
```

REAL-TIME CLOCK & LAN DATA

```
0040:0098 dw @timer_wait_off Ptr offset to wait done flag
0040:009A dw @timer_wait_seg Ptr segment to wait done flag
0040:009C dw timer_clk_low Timer low word, 1 microsecond clk
0040:009E dw timer_clk_hi Timer high word
0040:00A0 dw timer_clk_flag Timer flag
```

```
00h = post acknowledgd
01h = busy
80h = posted
```

```
0040:00A1 db lan_1 Local area network bytes
0040:00A2 db lan_2
0040:00A3 db lan_3
0040:00A4 db lan_4
0040:00A5 db lan_5
0040:00A6 db lan_6
0040:00A7 db lan_7
```

MORE ADVANCED VIDEO DATA

```
0040:00A8 dd @video_sav_tbls
```


Pointer to a save table of more pointers for the video system

offset	type	pointer to
0	dd	Video parameters
4	dd	Params save area
8	dd	Alpha char set
0Ch	dd	Graphics char set
10h	dd	2nd save ptr table
14h	dd	reserved (0:0)
18h	dd	reserved (0:0)

2ND SAVE TABLE (from ptr above)

offset	type	functions & pointers
0	dw	Bytes in this table
2	dd	Combination code tbl
6	dd	2nd alpha char set
0Ah	dd	user palette tbl
0Eh	dd	reserved (0:0)
12h	dd	reserved (0:0)
16h	dd	reserved (0:0)

```
0040:00BC db 72 reserved
0040:00CE dw days_since_1_80 Days since 1-Jan-1980 counter
0040:00F0 db 16 von MSDOS for Anwender reserviert
```

00500-005FF BASIC, DOS

```
0050:0000 db prn_scrn_stat Print screen status
```

```
00h = Print screen ready
01h = Print screen in progress
FFh = Error occurred
```

```
0050:0004 db Statusbyte for Einzellaufwerkmodus
0050:0010 dw BASIC-Segmentadresse
0050:0012 dw Segment for Systemuhr-Interrupt-Routine (BASIC)
0050:0014 dw Offset for Systemuhr-Interrupt-Routine (BASIC)
0050:0016 dw Segment for Break-Interrupt-Routine (BASIC)
0050:0018 dw Offset for Break-Interrupt-Routine (BASIC)
0050:001A dw Segment for Diskettenfehler-Interrupt-Routine (BASIC)
0050:001C dw Offset for Diskettenfehler-Interrupt-Routine (BASIC)
0050:0000 - 0050:00FF DOS, BASIC
```

00600-nnnnn io.sys, msdos.sys (ibmbio.sys, ibmdos.sys)

Die Länge des Betriebssystems hängt von der Versionsnummer ab. Zu den beiden Systemdateien kommt noch der residente Teil von command.com.

nnnnn-9FFFF Freier Arbeitsspeicher

A0000-AFFFF EGA-Grafik-Speicher

B0000-BFFFF CGA/MDA Display-Speicher

```
B000:0000-B000:7FFF Hercules-Karte, 1. Seite
B000:8000-B000:FFFF Hercules-Karte, 2. Seite
B800:0000-B800:3FFF CGA-Karte
B800:0000-B800:0FFF CGA-Karte, 1. Seite (Text)
B800:1000-B800:1FFF CGA-Karte, 2. Seite (Text)
B800:2000-B800:2FFF CGA-Karte, 3. Seite (Text)
B800:3000-B800:3FFF CGA-Karte, 4. Seite (Text)
```

C0000-CFFFF Festplatten-, EGA/VGA-BIOS

D0000-DFFFF ROM-BIOS-Erweiterungen, EMS-Pages

E0000-EFFFF EMS-Pages

F0000-FFFFF ROM-BIOS, ROM-BASIC

```
F000:0000-F000:5FFF 24k ungenutzt
F000:6000-F000:DFFF 32k BASIC
F000:E000-F000:FFFF  8k BIOS
F000:E05B loc Reset
F000:E2C3 loc NMI Entry Point
```

Hard Disk Information Tables

Each sub-table contains a a set of 16 bytes for each particular disk type. Type number specified may differ with manufactures specification. The hard disk table shown here assumes the first entry is type 0.

```
F000:E331 dw hdsk_cylinders Number of cylinders, hdsk_type_0
F000:E333 db hdsk_heads Number of heads
F000:E334 dw hdsk_lo_wrt_cyl Low write current cyl begin (XT)
F000:E336 dw hdsk_precompcyl Write pre-compensation cylinder
F000:E338 db hdsk_err_length Error correction burst length (XT)
F000:E339 db hdsk_misl_bits
```

Miscellaneous bit functions:
bits 0-2 disk option, XT only (XT)
0-2 unused, all others
3 = 1 if > 8 heads
4 unused
5 = 1 for bad map at last cylinder + 1
6 or 7 = 1 no retries

```
F000:E33A db hdsk_timeout Normal timeout  (XT)
F000:E33B db hdsk_fmt_timout Format timeout  (XT)
F000:E33C db hdsk_chk_timout Check timeout  (XT)
F000:E33D dw hdsk_parkng_cyl Parking cylinder number
F000:E33F db hdsk_sectr_trac Number of sectors per track
F000:E340 db hdsk_unused Unused
F000:E331 ds hdsk_type_
F000:E6F2 loc Bootstrap Load
```

System Configuration Table

```
F000:E6F5 dw Config_tbl_size  Size of table in bytes
F000:E6F7 db Config_model Model type
```

OFBh = 80386 model 70-80 types
OFCh = 80286 model 50-60 types, also most 80286/80386 compatibles
OFAh = 8088/86 model 25-30 type

```
F000:E6F8 db Config_sub_model Sub-Model type
F000:E6F9 db Config_BIOS_rev BIOS revsion number
F000:E6FA db Config_features Feature information
```

bit 7=1, hard disk uses DMA 3
bit 6=1, dual interrupt chips
bit 5=1, has real-time-clock
bit 4=1, int 15h, ah=4Fh is supported (keyboard)
bit 3=1, external wait support

bit 2=1, has extended BIOS RAM
bit 1=1, micro-channel
bit 0=1, unused

```
F000:E6FB db Config_info_bytes Information bytes (future use)
```

Baud Rate Table

Table of hex divsors for the serial ports. Table divisors for bauds 110 to 19,200.

```
F000:E729 dw baud_110, baud_rate_tbl
F000:E72B dw baud_150
F000:E72D dw baud_300
F000:E72F dw baud_600
F000:E731 dw baud_1200
F000:E733 dw baud_2400
F000:E735 dw baud_4800
F000:E737 dw baud_9600
F000:E739 dw baud_19200

F000:E82E loc Keyboard Function Call
F000:E987 loc Keyboard Hardware Interrupt
F000:EC59 loc Floppy Disk Function Call
F000:EF57 loc Floppy Disk ISR
```

Floppy Disk Parameters

```
F000:EFC7 db dsk_info_1 Start of ROM BIOS data areas
```

hi nibble = stepping rate in ms
lo nibble = head unload time, ms

```
F000:EFC8 db          dsk_info_2 2nd info byte bit 0 = 0 for DMA
F000:EFC9 db          dsk_motor_delay Delay after use for motor off
F000:EFCA db          dsk_sectr_bytes
```

Bytes per sector 0 = 128 bytes
1 = 256 bytes
2 = 512 bytes
3 = 1024 bytes

```
F000:EFCB db dsk_sector_trac Number of sectors per track
F000:EFCC db dsk_head_gap Gap Length
F000:EFCD db dsk_data_length Data Length
F000:EFCE db dsk_format_gap Format Gap Length
F000:EFCF db dsk_format_byte Format write byte
F000:EFD0 db dsk_settlg_time Head load time, in milliseconds
F000:EFD1 db dsk_startup_tim Motor startup wait time * .125ms
F000:EFD2 LPT-Fuction-Call
F000:F065 Video Function Call
```

Video Hardware Registers

```
F000:F0A4 db video_hdwr_tbl1 mode CGA 40 columns x 25 lines
F000:F0B4 db video_hdrw_tbl2 mode CGA 80 columns x 25 lines
F000:F0C4 db video_hdwr_tbl3 mode CGA graphics
F000:F0D4 db video_hdwr_tbl4 mode MDA 80 columns x 25 lines
F000:F0E4 dw video_buf_size1 Video buffer bytes CGA 40x25
F000:F0E6 dw video_buf_size2 Video buffer bytes CGA 80x25
F000:F0E8 dw video_buf_size3 Video buffer bytes CGA Graphics
F000:F0EA dw video_buf_size4 Video buffer bytes CGA Graphics
F000:F0EC db video_columntbl Video columns per modes 0-7
F000:F0F4 db video_hdwr_mode Video hardware modes (0-7)
```

```
F000:F841 loc Memory size Function call
F000:F84D loc Equipment Check Function call
F000:F859 loc Cassette Function Call
F000:FA6E db video_char_tbl Video characters in graphic modes
F000:FE6E loc Timer Function  Call
F000:FEA5 loc Timer Hardware Interrupt
F000:FEF3 dw int_vec_table Initial interrupt vectors
F000:FF1D dw int_data_table
F000:FF21 dw video_ptr
F000:FF23 dw int_vec_table_2
F000:FF53 loc Dummy Interrupt return
```

This routine processes invalid and unused interrupt requests. The hardware IRQ number is loaded into gen_int_occured, and the interrupt cleared. For software calls to an unused interrupt, a value OFFh is loaded into gen_int_occured, and the routine returns to the caller without changing registers. Alternatively, some systems simply return (iret).

```
F000:FF54 loc Print Screen Function Call
F000:FFF0 loc power_on_reset SYSTEM RESET
```

Reset the computer system. General operation includes a test of of the CPU, ROM checksum, and initialization of hardware including:

- Memory system
- Timer/Counter (which is also used for RAM refresh)
- Interrupt Controller(s)
- DMA Controller(s)
- Keyboard Controller
- Video Controller & Video RAM
- Floppy Controller
- Hard Disk Controller (if present)

Portions of the hardware may also have specific tests made to insure reliable operation. Test failures may display error code on the screen if the video subsystem is operational, or generate beeps or LED blinks to signify the error.

Note: A soft reset uses the warm_boot_flag to skip the memory tests. (i.e. from pressing Ctrl-Alt-Del).

The system checks for installed ROMs by searching memory from C000h to the beginning of the BIOS, in 2K chunks. ROM memory is identified if it starts with the word AA55h. It is followed a one byte field length of the ROM (divided by 512). If ROM is found, the BIOS will call the ROM at an offset of 3 from the beginning. This feature was not supported in the earliest PC machines.

The last task turns control over to the bootstrap loader (assuming the floppy controller is operational)

```
F000:FFF5 da rom_versn_date BIOS version date code
F000:FFFE db model_type
```

Model FFh = PC
FEh = 1st XT
FBh = later XTs
FCh = AT type (286/386)
FAh = models 25/30
F9h = IBM lap-tops
F8h = models 70/80 (80386)

```
F000:FFFF db  model_sub_type
```

100000-FDFFFF Expansion-RAM (AT)

FE0000-FEFFFF System (AT)

FF0000-FFFFFF BIOS (AT)

Personalcomputer XT 88

XT88LC23	-	XET 88 LOW COST	ÖS 6.360,-

* XT-Standard-Gehäuse mit Reset, Turbo u. Schlüsselschalter
* CPU 8088-1, 4,77/12 MHz, Sockel für 8087
* SPEED: Landmark 3.0, Norton SI 2.1
* 512kB RAM, erweiterbar auf 640kB
* 150 W Netzteil
* 4 verfügbare Einschubplätze f. Floppy- u. Harddisks
* Floppy Controller
* 360kB Diskettenlaufwerk (TEAC)
* Mono Graphik Printer-Karte (Herc. kompatibel) oder Color Graphik Printer-Karte (wahlweise)
* Tastatur - 101 Keys, deutsch od. US

XT88ST23	-	XET 88 STANDARD	ÖS 11.160,-

* XT-Standard-Gehäuse mit Reset, Turbo u. Schlüsselschalter
* CPU 8088-1, 4,77/12 MHz, Sockel für 8087
* SPEED: Landmark 3.0, Norton SI 2.1
* 640kB RAM,
* 150 W Netzteil
* 4 verfügbare Einschubplätze f. Floppy- u. Harddisks
* 360kB Diskettenlaufwerk (TEAC)
* 20 MB Festplatte, 65 ms Zugriffszeit (SEAGATE)
* HD-Controller für XT
* Multi I/O Karte für XT mit integriertem Floppy Controller, Game Port, Echtzeituhr, 1 Serielle Schnittstelle, 1 Parallelle Schnittstelle (2. Serielle Schnittstelle optional)
* Mono Graphik Printer-Karte (Herc. kompatibel) oder Color Graphik Printer-Karte (wahlweise)
* Tastatur - 101 Keys, deutsch od. US

Aufpreise für XT STANDARD

APRXT01	EGA-Graphik-Karte	ÖS	900.-
APRXT04	VGA 800x600/8 Bit	ÖS	1.080,-
APRXT05	FESTPLATTE 40 MB 28 ms ST-251-1	ÖS	2.010.-
APRXT03	FESTPLATTE 80 MB 24 ms ST-1096	ÖS	5.502.-

Preise 06/90 incl. 20% MWSt. Änderungen vorbehalten

Tel.: 0222/310-99-74-0
Fax.: 0222/310-99-74-14
EXCON Ing.Günther Hanisch
1090 Wien, Rögergasse 6-8

Personalcomputer AT 286

A286ST23	-	AT 286 STANDARD	ÖS 14.250.-

* Baby AT-Gehäuse mit Reset-, Turbo- u. Schlüsselschalter
* SPEED: Landmark 15.9, Norton SI 13.4
* 200 W Netzteil
* verfügbare Einbauplätze f. Floppy- u.Harddisks: 3x5¼, 1x3½
* CPU 80286-10, 16 Bit, 6/12MHz schaltbar, 0 Wait State
* 1 MB RAM, erweiterbar auf 2/4MB
* 1.2MB Diskettenlaufwerk (TEAC)
* 20 MB Festplatte, 65 ms Zugriffszeit (SEAGATE)
* FDD/HDD-Controller (WD kompatibel) Int. 1:1
* 2 seriell/ 1 parallel Interface
* Mono Graphik Printer-Karte (Herc. kompatibel) oder Color Graphik Printer-Karte (wahlweise)
* erweiterte Tastatur - 102 Keys, deutsch od. US
* EMS Treiber 4.0

A286DL23	-	AT 286 DeLUXE	ÖS 15.540,-

* BABY AT-Gehäuse mit Reset-, Turbo- u. Schlüsselschalter
* SPEED: Landmark 21.0, Norton SI 18.7
* 200 W Netzteil
* verfügbare Einbauplätze f. Floppy- u.Harddisks: 3x5¼, 1x3½
* CPU 80286-16, 16 Bit, 8/16MHz schaltbar, 0 Wait State
* 1 MB RAM, erweiterbar auf 2/4MB
* 1.2MB 5¼" Diskettenlaufwerk (TEAC)
* 20 MB Festplatte, 65 ms Zugriffszeit (SEAGATE)
* FDD/HDD-Controller (WD kompatibel) Int. 1:1
* 2 seriell/ 1 parallel Interface
* Mono Graphik Printer-Karte (Herc. kompatibel) oder Color Graphik Printer-Karte (wahlweise
* erweiterte Tastatur - 102 Keys, deutsch od. US
* EMS Treiber 4.0

Aufpreise für AT 286 STANDARD und AT 286 DeLUXE

APRAT002	FESTPLATTE 40 MB 28 ms ST-251-1	ÖS	2.010.-
APRAT003	FESTPLATTE 80 MB 24 ms ST-1096	ÖS	4.908.-
APRAT007	FESTPLATTE 80 MB IMPRIMIS	ÖS	7.980,-
APRAT010	FESTPLATTE 88 MB ESDI-Contr.	ÖS	13.788.-
APRAT013	FESTPLATTE 150 MB + ESDI-Contr.	ÖS	18.948,-
APRAT011	FESTPLATTE 170 MB + SCSI-Contr.	ÖS	23.328.-
APRAT012	FESTPLATTE 290 MB + SCSI-Contr.	ÖS	29.928.-
APRAT014	FESTPLATTE 320 MB + ESDI-Contr.	ÖS	31.308,-
APRAT004	EGA	ÖS	900.-
APRAT005	VGA 800x600/8 Bit	ÖS	1.080.-
APRAT006	VGA 1024x768/16 Bit (Paradise)	ÖS	1.500.-
APRAT001	VGA 1024x768/16 Bit (Tseng)	ÖS	1.800,-

Preise 06/90 incl. 20% MWSt. Änderungen vorbehalten

Tel.: 0222/310-99-74-0
Fax.: 0222/310-99-74-14
EXCON Ing.Günther Hanisch
1090 Wien, Rögergasse 6-8

Personalcomputer AT 386

A386LC23 - AT 386SX LowCOST ÖS 16.860,-

* BABY AT Gehäuse mit Reset-, Turbo- u. Schlüsselschalter
* CPU 80386SX-16, 16MHz, 0 Wait, Sockel f. 80387SX
* SPEED: Landmark 21.0, Norton SI 18.7
* 1 MB RAM, erweiterbar auf 2/4/8 MB
* 200 W Netzteil
* verfügbare Einbauplätze f. Floppy- u.Harddisks: 3x5¼, 1x3½
* 1.2MB 5¼" Diskettenlaufwerk (TEAC)
* 20 MB Festplatte, 65ms Zugriffszeit (Seagate)
* FDD/HDD-Controller (WD kompatibel) Int. 1:1
* 2 seriell/ 1 parallel Interface
* Mono Graphik Printer-Karte (Herc. kompatibel) oder Color Graphik Printer-Karte (wahlweise)
* erweiterte Tastatur - 102 Keys, deutsch od. US
* EMS Treiber 4.0

A386ST23 AT 386 STANDARD ÖS 23.880,-

* Big-Tower Gehäuse mit Turbo u.Schlüsselschalter
* CPU 80386-20, 25MHz, 0 Wait, Sockel f. 80387 und 80287 wahlweise
* SPEED: Landmark 33.4, Norton SI 28.2
* 1 MB RAM, erweiterbar auf 2/4/8 MB
* 200 W Netzteil
* 6 verfügbare Einbauplätze f. Floppy- u.Harddisks
* 1.2MB 5¼" Diskettenlaufwerk (TEAC)
* 20 MB Festplatte, 65 ms Zugriffszeit (SEAGATE)
* FDD/HDD-Controller (WD kompatibel) Int. 1:1
* 2 seriell/ 1 parallel Interface
* Mono Graphik Printer-Karte (Herc. kompatibel) oder Color Graphik Printer-Karte (wahlweise)
* erweiterte Tastatur - 102 Keys, deutsch od. US
* EMS Treiber 4.0

A386DL23 - AT 386 DeLUXE/25 MHz/64 K Cache ÖS 30.600,-

* Big-Tower Gehäuse mit Turbo u.Schlüsselschalter
* CPU 80386-25, 25MHz, 0 Wait, Sockel f. 80387
* 64k CACHE Memory
* SPEED: Landmark 41.9, Norton SI 31.6
* 1 MB RAM, erweiterbar auf 2/4/8/10/16MB
* 200 W Netzteil
* 6 verfügbare Einbauplätze f. Floppy- u.Harddisks
* 1.2MB 5¼" Diskettenlaufwerk (TEAC)
* 20 MB Festplatte, 65 ms Zugriffszeit (SEAGATE)
* FDD/HDD-Controller (WD kompatibel) Int. 1:1
* 2 seriell/ 1 parallel Interface
* Mono Graphik Printer-Karte (Herc. kompatibel) oder Color Graphik Printer-Karte (wahlweise)
* erweiterte Tastatur - 102 Keys, deutsch od. US

Preise 06/90 incl. 20% MWSt. Änderungen vorbehalten

Tel.: 0222/310-99-74-0
Fax.: 0222/310-99-74-14
EXCON Ing.Günther Hanisch
1090 Wien, Rögergasse 6-8

A386SD23 - AT 386 DeLUXE/33 MHz/64k CACHE ÖS 33.420,-

* Big-Tower Gehäuse mit Turbo u.Schlüsselschalter
* CPU 80386-33, 33MHz, 0 Wait, Sockel f. 80387
* 64k CACHE Memory
* SPEED: Landmark 58.7, Norton SI 45.9
* 1 MB RAM, erweiterbar auf 2/4/8/10/16MB
* 200 W Netzteil
* 6 verfügbare Einbauplätze f. Floppy- u.Harddisks
* 1.2 MB 5¼" Diskettenlaufwerk (TEAC)
* 20 MB Festplatte, 65 ms Zugriffszeit (SEAGATE)
* FDD/HDD-Controller (WD kompatibel) Int. 1:1
* 2 seriell/ 1 parallel Interface
* Mono Graphik Printer-Karte (Herc. kompatibel) oder Color Graphik Printer-Karte (wahlweise)
* erweiterte Tastatur - 102 Keys, deutsch od. US

Aufpreise für AT386 LowCOST, STANDARD und DeLUXE

APRAT002	FESTPLATTE 40 MB 28 ms ST-251-1	ÖS	2.010.-
APRAT003	FESTPLATTE 80 MB 24 ms ST-1096	ÖS	4.908.-
APRAT007	FESTPLATTE 80 MB IMPRIMIS	ÖS	7.980,-
APRAT010	FESTPLATTE 88 MB ESDI-Contr.	ÖS	13.788.-
APRAT013	FESTPLATTE 150 MB + ESDI-Contr.	ÖS	18.948,-
APRAT011	FESTPLATTE 170 MB + SCSI-Contr.	ÖS	23.328.-
APRAT012	FESTPLATTE 290 MB + SCSI-Contr.	ÖS	29.928.-
APRAT014	FESTPLATTE 320 MB + ESDI-Contr.	ÖS	31.308,-
APRAT004	EGA	ÖS	900.-
APRAT005	VGA 800x600/8 Bit	ÖS	1.080.-
APRAT006	VGA 1024x768/16 Bit (Paradise)	ÖS	1.500.-
APRAT001	VGA 1024x768/16 Bit (Tseng)	ÖS	1.800,-

MONITORE

4001MB/W24	14" Monochrom - Monitor (Samsung) * wahlw. Bernstein,Weiss	ÖS	1.890,-
4301ME24	14" E G A - Monitor * Samtron	ÖS	5.820,-
4301MV24	14" Monochrom V G A - Monitor * Samtron * RGB Analog Eingang	ÖS	2.580,-
4300MV24	14" V G A - Monitor * 640 x 480 Bildpunkte * Samtron * RGB Analog Eingang	ÖS	5.820,-
4301MM24	14" MULTISYNC Monitor * 800 x 600 Bildpunkte * RGB Analog od. TTL Eingang	ÖS	7.500.-
4510B024	14" N E C - Multisync 2 A * 800 x 600 Bildpunkte * RGB analog od.TTL Eingang	ÖS	10.380,-

Preise 06/90 incl. 20% MWSt. Änderungen vorbehalten

Tel.: 0222/310-99-74-0
Fax.: 0222/310-99-74-14
EXCON Ing.Günther Hanisch
1090 Wien, Rögergasse 6-8

MONITORE

4302B024	14" N E C - Multisync 3 D * 1024 x 768 Bildpunkte * RGB analog od.TTL Eingang	ÖS 12.300,-
4511MM24	16" N E C - Multisync 4 D * 1024 x 768 Bildpunkte * RGB Analog od. TTL Eingang	ÖS 25.980,-
5412MM24	20" N E C - Multisync 5 D * 1280 x 1024 Bildpunkte * RGB Analog od. TTL Eingang	ÖS 47.280,-

DRUCKER

5008P024	STAR 8 * Laserdrucker * 8 Seiten/min * 300x300 Punkte/Zoll Auflösung * 1 MB RAM * Seriell + Parallel Interface	ÖS 33.480,-
5005P024	NEC P6 + * 24-Nadel-Matrix Drucker * 80 Zeichen * 216 Z/Sek.EDV-Qualität (Pica) * Parallel-Interface	ÖS 10.980,-
5007P024	NEC P7 + * 24-Nadel-Matrix Drucker * 135 Zeichen * 216 Z/Sek.EDV-Qualität (Pica) * Parallel-Interface	ÖS 14.976,-
5006P024	Citizen LSP 120-D * 9-Nadel-Matrix Drucker * 120 Z/Sek.EDV-Qualität (Pica) * 30 Z/Sek. Schönschrift (Pica) * Parallel-Interface	ÖS 2.988,-
5009P024	Citizen Swift * 24-Nadel-Matrix Drucker * 190 Z/Sek. EDV-Qualität * 4 Schriftarten * Parallel-Interface	ÖS 7.200,-

DRUCKER ZUBEHÖR

5007CL24	Farbaufrüstsatz für P6+/P7+	ÖS 2.400,-
5008CL24	Sheetfeeder für P6+	ÖS 4.200,-
5009CL24	Sheetfeeder für P7+	ÖS 5.400,-
5010CL24	Sheetfeeder für Citizen Swift	ÖS 2.268,-

AUF ANFRAGE BIETEN WIR IHNEN AUCH GERNE DRUCKER UND MONITORE BELIEBIGER HERSTELLER NACH IHREN SPEZIELLEN WÜNSCHEN AN.

Preise 06/90 incl. 20% MWSt. Änderungen vorbehalten

Tel.: 0222/310-99-74-0
Fax.: 0222/310-99-74-14
EXCON Ing.Günther Hanisch
1090 Wien, Rögergasse 6-8

DISPLAY - KARTEN

310D0026	Mono/Graphic/Printer - Karte * Hercules-kompatible Karte	ÖS 480,-
311D0026	Mono/Color/Graphik/Printer-Karte umschaltbar Mono/Color-Mode	ÖS 690,-
320D0026	Color Graphic - Printerkarte	ÖS 480,-
330D/B26	E G A - Karte (640 x 480)	ÖS 1.380,-
332D/B26	V G A - Karte / 8-Bit * 800 x 600 Bildpunkte, * 256 k, 16 Farben * Paradise kompatibel	ÖS 1.560,-
333D/B26	V G A - Karte/16-Bit PARADISE * 1024 x 768 Bildpunkte, 512 k * interlaced Modus	ÖS 1.980,-
334D/B26	V G A - Karte/16-Bit TSENG * 1024 x 768 Bildpunkte, 512 k * non-interlaced Modus	ÖS 2.280,-

SCHNITTSTELLEN - KARTEN

200M0026	Multi I/O - Karte für XT 1x Ser/1x Par/1x Game/ Floppy-Ctr. (2.Ser.optional)	ÖS 990,-
630C0026	Multi I/O - Karte für AT 2x Ser/1x Par/1x Game	ÖS 540,-
360D0026	Parallel - Printer - Karte	ÖS 198,-
600C0026	RS232 - 2 Port, Interface f. XT	ÖS 360,-
620C0026	Intelligente RS 232 8-Port Schnittstellenkarte, Interface für AT, mit Treiber für: Xenix/Unix/PC MOS/VM 386	ÖS 9.480,-
631C0026	2 Seriell/ 1 Parallel Karte	ÖS 480.-
601C0026	RS 232 601C0026 RS 232 (4 Port)	

SPEICHERERWEITERUNGEN

ALLE SPEICHERERWEITERUNGS-KARTEN MIT 0k RAM BESTÜCKT

532R0026	RAM-Karte 4/8 MB, für 386	ÖS 1.590,-
540R0026	EMS-Karte 2 MB, 8-Bit für XT	ÖS 1.260,-
541R0026	EMS-Karte 2 MB, 16-Bit für AT	ÖS 1.260,-
542R0026	EMS-Karte 4 MB, 16 Bit für AT	ÖS 1.980.-

RAM

41256010	Dyn.RAM 41256-10 (256kx1)	öS	42,60
41256080	Dyn.RAM 41256-80 (256kx1)	öS	46,80
41640010	Dyn.RAM 4164-10 (64kx1)	ÖS	31,20
41464010	Dyn.RAM 41464-10 (64kx4)	ÖS	43,50
41425610	Dyn.RAM 414256-10 (256kx4)	ÖS	156,--

Preise 06/90 incl. 20% MWSt. Änderungen vorbehalten

Tel.: 0222/310-99-74-0
Fax.: 0222/310-99-74-14
EXCON Ing.Günther Hanisch
1090 Wien, Rögergasse 6-8

RAM

41100010	Dyn.RAM 411000-10	(1024kx1)	ÖS	150,--
SIM25608	SIMM MODULE 80nS	(256kx9)	ÖS	450,--
SIM1MB08	SIMM MODULE 80nS	(1024kx9)	ÖS	1.590,--
SIP25608	SIP MODULE 80nS	(256kx9)	ÖS	528,--
SIP1MB08	SIP MODULE 80nS	(1024kx9)	ÖS	1.680,--

DISKETTEN-LAUFWERKE

800F/J27	5 1/4", 360kB Diskettenlaufwerk * Teac	ÖS	1.080,-
810F/J27	5 1/4", 1,2 MB Diskettenlaufwerk * Teac	ÖS	1.290,-
811F/027	3 1/2", 720kB Diskettenlaufwerk * Teac, ohne Rahmen	ÖS	1.200,-
812F/027	3 1/2", 1,44 MB Diskettenlaufwerk * Teac, ohne Rahmen	ÖS	1.290,-
820F/J027	Rahmen * 5¼" für 3½" Diskettenlaufwerk	ÖS	168.-

FESTPLATTEN

900H0027	20 MB Festplatte, 65ms * Seagate, ST-225, 5¼"/HH	ÖS	3.180.-
920H0027	20 MB Festplatte, 40ms * Seagate, ST-124, 3½"	ÖS	3.600.-
942H0027	40 MB Festplatte, 28ms * Seagate, ST-251-1, 5¼"/HH	ÖS	5.190,-
940H0027	40 MB Festplatte, 17ms * Quantum, 5¼" * incl. AT-Bus-Controller	ÖS	7.500,-
980H0027	80 MB Festplatte, 24ms, * Seagate, ST-1096 N, 3½" * incl. SCSI-Controller ST02	ÖS	9.480,-
981H0027	80 MB Festplatte, 28ms, * Imprimis 5¼"/FH	ÖS	11.160,-
982H0027	88 MB Festplatte, 18ms, * Imprimis 5¼"/HH * ESDI	ÖS	14.640,-
991H0027	150 MB Festplatte, 16ms * Imprimis 5¼"/FH * ESDI	ÖS	19.800,-
993H0027	170 MB Festplatte, 18ms * Imprimis 5¼"/HH * SCSI	ÖS	21.000,-
992H0027	290 MB Festplatte, 16ms * Imprimis 5¼"/FH * SCSI	ÖS	27.600,-
994H0027	320 MB Festplatte, 14,5ms * Imprimis 5¼"/FH	ÖS	32.160,-

Preise 06/90 incl. 20% MWSt. Änderungen vorbehalten

Tel.: 0222/310-99-74-0
Fax.: 0222/310-99-74-14
EXCON Ing.Günther Hanisch
1090 Wien, Rögergasse 6-8

CONTROLLER

400F0026	Floppy-Disk - Controller XT * 2 x DS/DD, 360kB	ÖS	390,-
410F0026	Floppy-Disk - Controller XT/AT * 360/720kB/1.2/1.44MB	ÖS	570,-
420H0026	Hard-Disk - Controller XT * (WD - kompatibel)	ÖS	798,-
431F/H26	Floppy-/Hard Disk - MFM * 2 x Floppy- u.2 x Hard Disk * MFM, Interleave 1:1	ÖS	1.392.-
432F/H26	Floppy-/Hard Disk - ESDI (AT/386) * Western Digital WD 1007 * 2 x Floppy- u.2x Hard Disk	ÖS	4.800,-
433F/H26	Floppy-/Hard Disk - ESDI (AT/386) * ADAPTEC * 2 x Floppy- u.2x Hard Disk	ÖS	3.720,-
434F/H26	Floppy-/Hard Disk - AT-BUS (XT/AT)	ÖS	780,-
435F/H26	Floppy-/Hard Disk - SCSI * ADAPTEC	ÖS	6.900,-
436F/H26	Special Multi I/O Contr. (AT/386) * Floppy-Controller * HD-Controller AT-Bus * 2 Ser./1 Par.	ÖS	1.080,-

MOTHERBOARDS

110X0025	XT-TURBO MOTHERBOARD (10 MHz) * 4,77/10 MHz, 8 Slots, * 0k RAM, aufrüstbar wie folgt: 512k: 4*414256 + 2*41256 640k: 512k + 4*41464 + 2*4164	ÖS	900,-
121A0025	AT MOTHERBOARD (12 MHz) * 6/12 MHz, 8 Slots, CPU 80286-10 * 0 Wait State * Sockel für 80287 Coprozessor * 0k RAM , aufrüstbar wie folgt: 512k: 4*414256 + 2*41256 640k: 512k + 4*41464 + 2*4164 1 MB: 8*414256 + 4*41256 2 MB: 2*SIP Module 1MB 4 MB: 4*SIP Module 1MB	ÖS	2.490,-
122A0025	AT MOTHERBOARD (16 MHz) * 8/16 MHz, 8 Slots, CPU 80286-16 * 0 Wait State * Sockel für 80287 Coprozessor * 0k RAM, aufrüstbar wie folgt: 512k: 18*41256 640k: 18*4164 + 18*41256 1 MB: 36*41256 2 MB: 18*411000 4 MB: 36*411000	ÖS	3.600.-

Preise 06/90 incl. 20% MWSt. Änderungen vorbehalten

Tel.: 0222/310-99-74-0
: 0222/310-99-74-14
EXCON Ing.Günther Hanisch
1090 Wien, Rögergasse 6-8

MOTHERBOARDS

192AM025 80386SX - MOTHERBOARD (16 MHz) ÖS 4.920,-
* CPU 80386SX-16, 6/16 MHz
* 0 Wait State
* 3x8, 3x16 und 1x16 Memory BITS Slots
* SOCKEL 80387SX Coprozessor
* 0k RAM, erweiterbar wie folgt:
  512k: 18*41256
  640k: 18*4164 + 18*41256
  1 MB: 36*41256
  2 MB: 18*411000
  4 MB: 36*411000
  8 MB: 36*411000 + 4*SIMM Module 1 MB

191AM025 80386 - AT MOTHERBOARD (25MHz) ÖS 10.380,-
* BIG-Size, 20/25 MHz, CPU 80386-20
* 0 Wait State
* 3x8, 4x16 und 1x32 Memory BITS Slots
* SOCKEL f. 80287 u. 80387-Coprozessor
* 0k RAM, erweiterbar wie folgt:
  1 MB: 36*41256
  2 MB: 72*41256
  4 MB: 36*411000
  8 MB: 72*411000

190AM025 80386 - BABY MOTHERBOARD (25MHz) ÖS 9.990,-
* XT-Size, 20/25 MHz, CPU 80386-20
* 0 Wait State
* 3x8, 4x16 und 1x32 Memory BITS Slots
* SOCKEL f. 80387-Coprozessor
* 0k RAM, erweiterbar wie folgt:
  1 MB: 4*SIMM Modul 256k
  2 MB: 8*SIMM Modul 256k
  4 MB: 4*SIMM Modul 1MB
  8 MB: 8*SIMM Modul 1MB

193AM025 80386 - CACHE MOTHERBOARD (25MHz) ÖS 16.980,-
* mit 64kB Cache Memory
* BIG-Size, 20/25 MHz, CPU 80386-25
* 0 Wait State
* 3x8, 4x16, und 1x32 Memory BITS Slots
* SOCKEL f. 80287 u. 80387-Coprozessor
* 0k RAM, erweiterbar wie folgt:
  1 MB: 4*SIMM Module 256k
  2 MB: 8*SIMM Module 256k
  3 MB: 12*SIMM Module 256k
  4 MB: 36*411000 oder 4*SIMM Module 1MB
  8 MB: 36*411000 + 4*SIMM Module 1MB
  12 MB: 36*411000 + 8*SIMM Module 1MB
  16 MB: 36*411000 + 12*SIMM Module 1MB

Preise 06/90 incl. 20% MWSt. Änderungen vorbehalten

Tel.: 0222/310-99-74-0
Fax.: 0222/310-99-74-14
EXCON Ing.Günther Hanisch
1090 Wien, Rögergasse 6-8

MOTHERBOARDS

194AM025 80386 - CACHE MOTHERBOARD (33MHz) ÖS 19.800,-
wie 80386-CACHE Motherboard (25Mhz)
jedoch mit:
* 25/33 MHz, CPU-80386-33, 0 Wait State

CO-PROZESSOREN

701CP087	80287-8	ÖS	3.240,-
702CP087	80287-10	ÖS	3.840,-
703CP087	80387SX-16	ÖS	4.950,-
704CP087	80387-20	ÖS	6.630,-
705CP087	80387-25	ÖS	8.760.-
706CP087	80387-33	ÖS	9.990,-
707CP087	8087-2 (bis 8 MHz)	ÖS	2.400,-
708CP087	8087-1 (bis 10 MHz)	ÖS	2.880,-

GEHÄUSE - STROMVERSORGUNG

3100C027 GEHÄUSE FÜR XT ÖS 1.410,-
* inkl. 150 W Netzteil
* 4 Slim Einschubplätze 5¼"
* Reset- u. Turboschalter, Schloß

3201C027 GEHÄUSE FÜR AT ÖS 2.220.-
* inkl. 200 W Netzteil
* 5 Slim Einschubplätze 5¼"
* Reset- u. Turboschalter, Schloß

3202C027 GEHÄUSE FÜR BABY - AT ÖS 2.070,-
* inkl. 200 W Netzteil
* 3x5¼, 1x3¼ Slim Einschubplätze
* Reset- u. Turboschalter, Schloß

3204C027 BABY - TOWER 386D ÖS 2.820.-
* incl. 200W Netzteil
* 3 Einschubpl. 5¼"
* Turbo Sw., LED-Speed Anzeige

3205C027 BIG - TOWER 386D ÖS 3.600,-
* incl. 200W Netzteil
* 6 Slim Einschubplätze 5¼"
* Turbo Sw., LED-Speed Anzeige

1000S027	150W STROMVERSORGUNG für XT	ÖS	990.-
1100S027	200W STROMVERSORGUNG für AT	ÖS	1.320.-
1200S027	200W STROMVERSORGUNG für TOWER	ÖS	1.440.-

Preise 06/90 incl. 20% MWSt. Änderungen vorbehalten

Tel.: 0222/310-99-74-0
Fax.: 0222/310-99-74-14
EXCON Ing.Günther Hanisch
1090 Wien, Rögergasse 6-8

TASTATUREN

2000K027	TASTATUR 84 KEYS (XT/AT) * deutscher Zeichensatz	ÖS	900,-
2001K027	DETTO jedoch mit ASCII-Zeichens.	ÖS	930,-
2100K027	TASTATUR 102 KEYS (XT/AT) * deutscher Zeichensatz	ÖS	900,-
2101K027	DETTO jedoch mit ASCII-Zeichens.	ÖS	930,-

ZUBEHÖR

5000A028	SERIELL MOUSE "WITTY" * MS-PC - kompatibel	ÖS	600,-
5100A028	SERIELL MOUSE "GENIUS" GM 6000 * MS-PC - kompatibel	ÖS	870,-
5110A028	SERIELL MOUSE "GENIUS" F-302 * MS-PC - kompatibel * PS/2 - tauglich	ÖS	990,-
5111A028	Adapterstecker für "GENIUS" F-302 * für PS/2	ÖS	108,-
5150A028	GENIUS Handy Scanner 4500 * 100-400 DPI * DrGenius, Scan Edit, OCR-Software * 32 Graustufen	ÖS	4.490,-
5400A028	MONITOR - STÄNDER	ÖS	300,-
5500A028	DRUCKERSTÄNDER A4 * Druckerständer f.A4 Drucker	ÖS	216,-
5501A028	DRUCKERSTÄNDER A3 * Druckerständer f.A3 Drucker	ÖS	312,-
5502A028	CPU Ständer für PC AT/XT	ÖS	288,-
DS101027	DATA SWITCH RS232 * 3 - Weg	ÖS	300,-
DS201027	DATA SWITCH CENTRONICS *.3 - Weg	ÖS	420,-
5300A028	DRUCKERKABEL PARALLEL 2 m	ÖS	132,-
5306A028	DRUCKERKABEL PARALLEL 6 m	ÖS	264,-
5207A028	DRUCKERKABEL PARALLEL 10 m	ÖS	336,-
5301A028	DRUCKERKABEL SERIELL 2 m	ÖS	180,-
5303A028	TASTATURKABEL 2 m * Verl.kabel f.Keyboard	ÖS	144,-
5304A028	MONITORKABEL 2 m * Verl.kabel f.Monitor	ÖS	168.-
5403A028	FLOPPY KABEL	ÖS	48.-
5402A028	HARDDISK - KABELSET * Kabel f.HD - Controller	ÖS	60,-
5401A028	FD/HD - KABELSET * Kabel f. FD/HD - Controller	ÖS	96,-

Preise 06/90 incl. 20% MWSt. Änderungen vorbehalten

Tel.: 0222/310-99-74-0
Fax.: 0222/310-99-74-14
EXCON Ing.Günther Hanisch
1090 Wien, Rögergasse 6-8

ZUBEHÖR

5302A028	RS 232 ADAPTERKABEL * Kabel f.RS232-Schnittstelle	ÖS	180,-
5003Z028	ADAPTERSTECKER RS 232 * Zusatz f.RS232-Schnittstelle	ÖS	96,-
8250S030	RS232 Schnittstellenerweiterung * 1x82450,1x1488,1x1489 Chips	ÖS	348,-
5502A028	MINITESTER für RS 232	ÖS	144,-
5410A028	Harddiskrack 3½"	ÖS	960,-
5514MF28	MONITOR-COLOR-FILTER 14"	ÖS	156,-
5512MF28	MONITOR-COLOR-FILTER 12"	ÖS	138,-
5106A028	DISKETTENBOX 5¼" * für 100 Stück	ÖS	114,-
5107A028	DISKETTENBOX 3½" * für 100 Stück	ÖS	114,-
5108A028	DISKETTENBOX 5¼", stapelbar * für 100 Stück	ÖS	468,-
5109A028	DISKETTENBOX 3½", stapelbar * für 100 Stück	ÖS	468,-
5700A028	DISKETTEN "NONAME" * 5 1/4" DS/DD, 48 Tpi	ÖS	4,80
5701A028	Maxell - DISKETTEN * 5 1/4" DS/HD, 96 Tpi	ÖS	26,40
5710A028	DISKETTEN "NONAME" * 3 1/2" DS/DD,135 Tpi	ÖS	24,-
5720A028	Maxell - DISKETTEN * 3 1/2", DS/HD, 270 Tpi	ÖS	54,-
5600A028	KONZEPTHALTER A4 * Konzepthalter flexibel	ÖS	264,-
5601A028	KONZEPTHALTER A4 * Konzepthalter mit Standkonsole	ÖS	312,-
5610A028	MONITORSCHWENKARM	ÖS	2.988,-
5620A028	TASTATURLADE, Oberbau	ÖS	828,-
5621A028	TASTATURLADE, Unterbau	ÖS	636,-

UNTERBRECHUNGSFREIE STROMVERSORGUNGEN, STREAMER TAPES

1901S027	UPS 600 VA	ÖS	6.720,-
1902S027	UPS 1000 VA	ÖS	8.880,-
1907S027	STREAMER MAYNARD 60 MB	ÖS	13.560,-
1908S027	STREAMER MAYNARD 150 MB	ÖS	20.640,-
1909S027	STREAMER KASSETTE 60 MB	ÖS	690,-
1910S027	STREAMER KASSETTE 150 MB	AUF ANFRAGE	

BETRIEBSSYSTEM DOS

7000D031	MS-DOS 3.3 (englisch)	ÖS	690,-
7001D031	PC-DOS 4.0 (englisch)	ÖS	1.392,-

Preise 06/90 incl. 20% MWSt. Änderungen vorbehalten

Tel.: 0222/310-99-74-0
Fax.: 0222/310-99-74-14
EXCON Ing.Günther Hanisch
1090 Wien, Rögergasse 6-8

NETZWERK-HARDWARE

ARC01026	ARC-NET (SMC) NETZWERKARTE, 8 Bit * Standard Microsystems kompatibel * 2,5 MB/sec	ÖS	1.320,-
ARC01126	ARC-NET (SMC) NETZWERKARTE, 16 Bit * Standard Microsystems kompatibel * 2,5 MB/sec	ÖS	2.496,-
ARC03026	AKTIVE HUB 8-PORT extern * für max. 600 m Kabellänge * 8-fach Verteiler	ÖS	3.300,-
ARC02026	PASSIVE HUB 4-PORT * für max. 10 m Kabellänge * 4-fach Verteiler	ÖS	198,-
ARC20026	AKTIVE HUB 4-PORT (intern)	ÖS	1.440,-
ARC04026	ARCNET KABEL 5 m	ÖS	156,-
ARC05026	ARCNET KABEL 10 m	ÖS	288,-
ARC08026	ARCNET BNC-CONNECTOR 93 Ohm	ÖS	36,-
ARC07026	BNC TERMINATOR	ÖS	36,-
ARC09026	KABEL KONFEKTIONIEREN	ÖS	360,-
ETH01026	EHTERNET CARD, 8-Bit * Novell-kompatibel * 10 MB/sec	ÖS	2.340,-
ETH01126	EHTERNET CARD, 16-Bit * Novell-kompatibel * 10 MB/sec	ÖS	3.000,-
ETH09026	EHTERNET REPEATER	ÖS	15.072,-
ETH02026	EHTERNET KABEL 5 m	ÖS	156,-
ETH03026	EHTERNET KABEL 10 m	ÖS	288,-
ETH06026	EHTERNET CONNECTOR	ÖS	36,-
ETH07026	T-CONNECTOR	ÖS	36,-
ETH10026	KABEL KONFEKTIONIEREN	ÖS	360,-

NOVELL-NETZWERK-SOFTWARE

ELS10026	ELS-NETWARE LEVEL I (2.0a) * für PC-AT 286/386 Server * für max. 4 Workstations * nur im non-dedicated Mode (Server = Workstation)	ÖS	9.600,-
ELS20026	ELS-NETWARE LEVEL II (2.15) * für PC-AT 286/386 Server * für max. 8 Workstations * dedicated oder non-dedicated mode (Server entweder als Workstation oder nur Server)	ÖS	22.800,-

Preise 06/90 incl. 20% MWSt. Änderungen vorbehalten

Tel.: 0222/310-99-74-0
Fax.: 0222/310-99-74-14

EXCON Ing.Günther Hanisch
1090 Wien, Rögergasse 6-8

NOVELL-NETZWERK-SOFTWARE

ADV21226	ADVANCED NETWARE 286 (2.15) * für PC-AT 286/386 Server * für max. 100 Workstations * dedicated oder non-dedicated mode (Server entweder als Workstation oder nur Server)	ÖS	39.000,-
SFT21226	ADVANCED NETWARE SFT (2.15) * für PC-AT 286/386 Server * für maximal 100 Workstations * dedicated oder non-dedicated mode * mit Festplattenspiegelung * incl. TTS und BRIVE	ÖS	58.800,-
SFT38626	ADVANCED NETWARE 386 (3.0) * für PC 386 Server * max. 200 Workstations * mit Festplattenspiegelung * mit Server-Spiegelung * incl. TTS und BTRIVE	ÖS	98.400,-
DCB00026	DISC COPROCESSOR BOARD (DCB) * Festplattenkanal 1-4 möglich * SCSI Interface für externe oder interne Harddisc	ÖS	9.972,-
0INSTS20	Installation Server	ÖS	9.000,-
0INSTT30	Installation pro Workstation	ÖS	600,-

Mnum: 77 Lfd:204

Dipl.-Ing. Franz FIALA
Siccardsburggasse 4/1/22
1100 Wien

DVR:0596299

Absender:

P C C - T 6 M
Wexstraße 21
Postfach 59
1202 W I E N

P.b.b.
Verlagspostamt
1200 WIEN

orbehalten
ıther Hanisch
ögergasse 6-8