# MicroPC

Technisches Handbuch

# *taskit* Rechnertechnik GmbH

**Seelenbinderstr. 33**
**D-12555 Berlin**

**Telefon 030-611295-0**
**FAX 030-611295-10**
**http://www.taskit.de**

 **V2.26 (08.12.2005)**

# 1. Einführung

Der MicroPC kann überall dort eingesetzt werden, wo geringer Raum- und Strombedarf eine Rolle spielen und man sich dennoch wegen der leichten Programmierbarkeit eine PC-kompatible Lösung wünscht.

Beispiele für Anwendungen:

- mobile oder nichtmobile Datenerfassungsgeräte
- LCD-Terminals
- Mess- und Prüfgeräte
- Alarmanlagen
- jede einfache oder weniger einfache Automatisierungsaufgabe.

Der MicroPC lässt sich wie ein DOS-PC mit allen üblichen DOS-Compilern programmieren, wie Borland- und Microsoft-C, Pascal, Basic u.a. Der Ablauf beim Erstellen eines Programms ist dabei denkbar einfach. Da die Laufwerke des PCs unter dem Terminalprogramm als Laufwerke des MicroPC abgebildet werden, kann man die erzeugte EXE-Datei per DOS-Befehl direkt starten - oder auch per DOS-Copy erst auf die Flash-Disk kopieren und von dort starten. Zur Erleichterung der Inbetriebnahme gibt es ein Starterkit, das eine Entwicklerplatine mit Stromversorgung und die notwendige Software enthält.

Das BIOS-Setup enthält umfangreiche Konfigurationsmöglichkeiten für Speicheraufteilung, serielle Schnittstellen und I/O-Ports. So werden vom BIOS auch zusätzliche serielle Schnittstellen unterstützt. Die I/O-Pins können einzeln oder in Gruppen auf Ein- oder Ausgang eingestellt werden, teilweise auch mit Interrupt-Funktion. Beliebige Hardware-Initialisierungen für die Peripherie können im BIOS-Setup vorgegeben werden. Daneben enthält das BIOS-Setup-Menü die Funktionen zum Laden von BIOS-Updates oder des ROM-DOS sowie zum Laden oder Löschen der Flash-Disk.

# 2. Übersicht der technischen Eigenschaften

## 2.1. CPU

- Intel 386EX "Embedded" Prozessor (Erweiterung des 386SX)
- 25 MHz Takt
- 3,3 V Betriebsspannung

## 2.2. Speicher

- 2 MB Flash-Speicher mit Flashfile-System (optional auch 1MB, 4MB oder 8MB)
- 512 kB oder 1 MB statisches RAM, Batterie-Stützung möglich,
- davon maximal 896 kB DOS-Arbeitsspeicher (bei 1MB RAM Bestückung)
- Serielles EEPROM mit 512 Byte.

## 2.3. Firmware

- PC-kompatibles BIOS, Konfigurationsmenü im BIOS-Setup
- FreeDOS 7.1
- Datalight ROMDOS (optional, lizenzpflichtig)

## 2.4. Schnittstellen

- zwei PC-kompatible serielle Schnittstellen (8250-kompatibel), maximal 781,25 kBaud (= 25 MHZ/32), oder 115,2 kBaud als maximale PC-kompatible Baudrate, TTL-Pegel, durch Anschluß von IF-Modulen (IF232-9DIR, IF485) als RS-232 oder RS-485 konfigurierbar
- PIF-Bus (5V-kompatibel)
- synchrone serielle Schnittstelle bis maximal 6,25 MBaud (= 25 MHz / 4)
- I²C-Bus
- maximal 32 digitale I/O-Ports, einzeln oder in Vierer-Gruppen als Ein- oder Ausgang programmierbar
- 6 freie IRQs
- Echtzeituhr-Alarm Ausgang
- Timer-Clock, Timer-Gate und Timer-Out Signale

Die verschiedenen Funktionen sind teilweise durch Mehrfachbelegung der Pins des Steckverbinders realisiert, es können also nicht alle Funktionen gleichzeitig verwendet werden (s. Tabelle in Kap. 10).

## 2.5. Power-Management

Der Stromverbrauch kann durch die Power-Management Funktionen des BIOS in vielen Fällen drastisch reduziert werden. Dies geschieht durch Heruntertakten, Idle-Mode, Powerdown-Mode oder Deep Powerdown-Mode (Stop-Mode).

Stromverbrauch:

- Aktiv: 135 mA (bei 25 MHz Takt)
- Idle-Mode: 9 mA
- Powerdown: 4 mA
- Deep Powerdown: 0,3 mA

## 2.6. Sonstiges

- drei 8254-kompatible Timer, davon zwei auch extern zu betreiben
- zwei 8259-kompatible Interrupt-Controller
- Echtzeituhr (Real Time Clock, RTC), optional mit Batterie-Stützung
- programmierbarer Watchdog-Timer
- Eindeutige Hardware-Seriennummer, per Software auslesbar (kann für den Kopierschutz von Anwenderprogrammen verwendet werden).
- lieferbar mit erweitertem Temperaturbereich -40°C ... +85°C
- Gehäuse-Abmessungen: 43 x 36,4 x 5 mm (BxTxH)

# 3. MicroPC Starterkit

## 3.1. Inhalt des Starterkits

Das MicroPC Starterkit besteht aus folgenden Bestandteilen :

- MicroPC (CPU-Modul)
- MicroPC-Base (Basis- und Prototyp-Platine)
- Steckernetzteil, Eingang AC 230V, Ausgang DC 9 bis 16V, mindestens 400 mA
- Verbindungskabel (serielles Nullmodem-Kabel mit zwei 9-poligen DSUB-Buchsen)
- Adapterkabel DSUB-9 auf 10polige Stiftleiste
- CD mit Firmware-Dateien zum Produkt (BIOS,Utilities,Libraries) sowie FreeDOS und ROMDOS (Extra-Lizensierung erforderlich)
- Handbuch

## 3.2. MicroPC-Base

Die Platine des Starterkits ("MicroPC-Base") ermöglicht es, auf besonders einfache Weise den MicroPC in Betrieb zu nehmen.

Das MicroPC-Base soll sowohl einfach als auch universell sein. Es sind daher einige Schaltungsteile vorgesehen, die man nicht immer benötigen wird, die den Einsatz für bestimmte Zwecke aber erleichtern. Der Schaltplan der MicroPC-Base Platine befindet sich im Kap. 11 in diesem Handbuch.

Technische Eigenschaften

Stromversorgung:

Aus einer ungeregelten Eingangsspannung von ca. 8 bis 35 V (z.B. Steckernetzteil) werden drei Spannungen erzeugt:

- 3,3 V für das CPU-Modul,
- 5 V für eventuell anzuschließende Peripherie,
- 22V als Kontrastspannung für gewisse LCD-Module (solche, die keine eigene Kontrastspannungserzeugung besitzen).

RS232-Treiber/Empfänger:

für die Signale RxD, TxD, RTS und CTS der beiden seriellen Schnittstellen des MicroPC.

Steckverbinder:

- Steckplatz MicroPC (CompactFlash-Slot),
- 50-polige Stiftleiste im 2,54mm Raster mit den Signalen des MicroPC,
- Zwei 10-polige Wannenstiftleisten im 2,54mm Raster für die RS232-Schnittstellen,
- Zwei einreihige Stiftleisten im 2,54mm Raster für LCD- (X6) und Matrixtastatur-Kabel (X7),
- Zwei Schraubklemmen für die Stromversorgung. Hier kann auch eine DC-Buchse befestigt werden.

## 3.3. Inbetriebnahme

Da die meisten Hilfsprogramme für den MicroPC unter Windows oder DOS laufen, verwendet man einen Entwicklungs-PC mit Microsoft Windows ab Version '95. Von der mitgelieferten CD startet man das Installationsprogramm MicroPCsetup.exe. Hierdurch werden die MicroPC-Dateien im angegebenen Verzeichnis entpackt, außerdem wird optional das Terminalprogramm Vterm32.exe auf dem PC installiert. Alternativ zu Vterm32 gibt es im Unterverzeichnis Util des MicroPC das DOS-Programm VTERM.EXE. Die korrekte Baudrate - beim MicroPC normalerweise 115200 Baud – muß im Terminalprogramm eingestellt werden, andernfalls erscheinen nur "kryptische" Zeichen auf dem Bildschirm.

**Achtung:** Unter bestimmten Bedingungen funktioniert bei eingeschalteter ACPI-Funktion im Powermanagement des PC die Kommunikationsrichtung PC -> MicroPC nicht. Diese Funktion muß dann vorher deaktiviert werden.

Der 10polige Wannen-Stecker COM1(X3) des MicroPC wird mit einer freien seriellen Schnittstelle (COM-Port) des PC mittels Adapterkabel und seriellem Verbindungskabel verbunden. Falls ein anderes Kabel als das mitgelieferte verwendet wird, muß dieses ein sogenanntes Nullmodem-Kabel sein ("gekreuztes Kabel"). Das mitgelieferte Kabel unterstützt nur TxD und RxD. Die Signale CTS und DCD sind auf der jeweils gleichen Seite mit RTS verbunden.

Schließen Sie nun das Steckernetzteil an die DC-Buchse neben den seriellen Ports an. Bei korrekter Einstellung können Sie nun das Hochfahren des BIOS beobachten. Nach dem Speichertest wird DOS gebootet, und sie gelangen zum DOS-Prompt des MicroPC. Unter dem DOS Prompt des Micro-PC kann man wie beim PC gewohnt arbeiten.

## 3.4. Laufwerke des MicroPC

Der Micro-PC stellt mehrere "Laufwerke" zur Verfügung. Laufwerk A: ist eine ROM-Disk, Laufwerk B: eine RAM-Disk, Laufwerk C: eine Flashdisk, von der in der Standardeinstellung auch gebootet wird, und Laufwerk D: ein (optionales) CompactFlash-Modul. Auf diese Laufwerke kann wie auf Disketten- bzw. Festplattenlaufwerke im PC zugegriffen werden. Die Laufwerke A: , B: und D: sind standardmäßig deaktiviert, können aber im BIOS-Setup aktiviert werden. Die RAM-Disk (Laufwerk B:) kann dann sofort verwendet werden. Um Laufwerk A: verwenden zu können, muß über das BIOS-Setup erst ein ROMDISK-Image geladen werden.

## 3.5. Beschleunigen des Boot-Vorgangs

Beim Booten von ROM-DOS kann man mit der F5-Taste die Ausführung von CONFIG.SYS und AUTOEXEC.BAT umgehen bzw. mit F8 die Einzelbefehls-Ausführung starten. Das ROM-DOS wartet einige Sekunden auf eine Tastatureingabe. Durch den Befehl

*switches=/f*

am Anfang der Datei CONFIG.SYS wird diese Wartezeit vermieden. Es ist dann auch kein Eingriff in den Boot-Vorgang möglich.

Werden im BIOS-Setup auch noch die Optionen "Boot Messages" abgeschaltet und "Fast Boot" eingeschaltet, so dauert das Starten des MicroPC nach dem Einschalten oder Reset bis zum Erscheinen des DOS-Prompts weniger als zwei Sekunden.

## 3.6. PC-Laufwerke über Remote-Drive einbinden

Neben der Möglichkeit, Programme und Daten mit dem PC über X-Modem oder Z-Modem auszutauschen, wozu jedes Terminal-Programm, das über diese Funktionen verfügt, geeignet ist, steht bei Verwendung des mitgelieferten Programms VTERM (im Verzeichnis UTIL) ein Verfahren zur Einbindung von PC-Laufwerken zur Verfügung. Hierzu dienen die Programme RDRIVE.EXE, RMAP.EXE und RMCWD.EXE. In der auf der Flash-Disk befindlichen Datei MAP.BAT finden sich Beispiele, wie die Initialisierung der Treiber bewerkstelligt werden kann. Einzelheiten hierzu sind im Kapitel über RDRIVE zu finden.

## 3.7. BIOS konfigurieren

Die Kommunikation erfolgt über die erste serielle Schnittstelle des MicroPC. Die Baudrate beträgt normalerweise 115200 Baud. Die Konfiguration der Karte erfolgt im BIOS Setup. Um in das Setup der Karte zu gelangen, muß während des Bootens die Taste S gedrückt werden.

**Achtung:** Verschiedene Einstellungen im Setup, z.B. das Abschalten der Standard-I/O Schnittstelle, können dazu führen, daß man nicht mehr auf den MicroPC zugreifen kann. Im Falle von falschen Einstellungen im Setup, die den weiteren Zugriff verhindern, kann man durch Kurzschließen des Pins 19 ("PIF-Ready") von X2 nach Masse wieder Standard-Einstellungen für die COM1 bekommen. Das BIOS springt anschließend ins Setup.

Diese Möglichkeit kann durch die Setup-Einstellung "Emergency Jumper: disabled" allerdings auch abgeschaltet werden, um unberechtigte Zugriffe auf die Software zu verhindern (das sollte dann aber erst nach Fertigstellung des Applikationsprogramms geschehen).

In keinem Fall sollte während des Abspeicherns der Setup-Einstellungen die Versorgungsspannung abgeschaltet werden, da sonst das BIOS gelöscht werden könnte und kein weiterer Zugriff auf die Karte mehr möglich ist. Das Abspeichern selbst dauert nur wenige Sekunden, anschließend bootet die Karte neu.

# 4. Programmierung des MicroPC

## 4.1. Verwenden von PC-Compilern

Als DOS-kompatible und weitgehend PC-kompatible CPU-Karte läßt sich der MicroPC grundsätzlich wie ein normaler DOS-PC programmieren. Das heißt, daß die üblichen Programmierwerkzeuge für den PC, so weit sie für DOS-Programme geeignet sind, auch für den MicroPC verwendet werden können. Insbesondere ist dabei an Basic, Pascal und "C" zu denken.

Gewisse Einschränkungen müssen hinsichtlich der Ausgabe beachtet werden: Der jeweilige Compiler muß so eingestellt werden, daß die Ausgabe über DOS- oder BIOS-Funktionen erfolgt. Nur dann kann die Ausgabe auf die serielle Schnittstelle - die ja den Consolen-Port des MicroPC darstellt - umgeleitet werden. Ausgaben, die den Bildschirmspeicher direkt programmieren - eine beliebte, weil schnelle, Methode beim PC - sind auf dem MicroPC normalerweise nicht möglich (jedoch gibt es für einige LCDs BIOS-Erweiterungen, die den virtuellen 8086-Mode des 386-Prozessors verwenden und Zugriffe auf VGA-Chip und Bildschirmspeicher abfangen und kompatibel umsetzen).

## 4.2. Zugriff auf die PIF-Peripherie

Der PIF-Bus wird "I/O-mapped" angesprochen, also durch IN- und OUT-Befehle des 386EX-Prozessors. Dies geschieht auf den 64 I/O-Adressen von 300h bis 33Fh. Der Datenbus des PIF-Bus ist 8 Bit breit.

## 4.3. Testen von Programmen

### 4.3.1. Debugging auf dem PC

Programme für den MicroPC sollten so weit wie möglich auf dem Entwicklungs-PC selbst getestet werden, da hier die PC-Entwicklungsumgebungen und Debugger uneingeschränkt zur Verfügung stehen. Allerdings sind hier zunächst keine Programm-Zugriffe auf Hardware-Erweiterungen über den PIF-Bus möglich. Es ist aber möglich, über die Interface-Karte PIF-ISA-Base den PIF-Bus direkt an den PC anzuschließen. Diese Karte belegt einen AT-Slot (eine PCI-Karte ist derzeit nicht lieferbar), ihre Hardware-Adressen und IRQ lassen läßt sich flexibel konfigurieren. Diese Methode funktioniert für alle PIF-I/O-Karten (nicht jedoch für LCDs, die BIOS-Erweiterungen verwenden).

### 4.3.2. Remote Debugging

Eine effektive Methode, Programme im Quelltext direkt auf dem MicroPC zu debuggen, gibt es in Form des Remote-Kernels des Turbo-Debuggers von Borland. Hierzu muß eine der beiden seriellen Schnittstellen des MicroPC nur für den Debugger frei zur Verfügung stehen, irgendwelche anderen Aktivitäten auf der gleichen Schnittstelle stören die Kommunikation zwischen dem Remote-Kernel und dem Host-Programm. Der Remote-Kernel (tdremote.exe) wird auf die Flashdisk des MicroPC kopiert und dort gestartet.

# 5. Hardware

## 5.1. 386EX-Core

Den CPU Kern des Intel 386EX bildet ein voll statischer 386SX. Dieser hat einen 16 Bit breiten Daten- und einen 26 Bit (386SX: 24 Bit) breiten Adressbus. Es wird ein Adressraum von 64 Mbyte Speicher und 64 kByte I/O bereitgestellt. Der Prozessorkern unterstützt Protected-Mode-Anwendungen.

## 5.2. Speicher

### 5.2.1. RAM

#### *5.2.1.1. RAM Layout*

Der MicroPC kann mit 512 kB oder 1MB SRAM bestückt werden. Der MicroPC des Starterkits ist mit 1MB SRAM bestückt. Dieser Speicher ist im Real Mode des Prozessors, damit also im untersten MB, adressierbar. Ausgenommen ist der Bereich zwischen 896 kB und 1MB – dort befindet sich ein Teil des Flash-Speichers mit dem BIOS (48 kB), dem ROM-DOS-Kernel (48 kB) und optional einer BIOS-Erweiterung (maximal 32 kB). Dieser Flash-Bereich kann bei Bedarf auch auf 256kB oder 512 kB eingestellt werden, so daß nur noch maximal 768 kB oder 512 kB RAM im untersten MB zur Verfügung stehen.

Bei Bestückung mit 1MB SRAM sind also mindestens 128 kB nicht im Real Mode adressierbar. Das gesamte RAM ist aber auch im zweiten MB des Adressraums eingeblendet (als "Extended Memory") und kann dort mit Protected-Mode Funktionen angesprochen werden (BIOS INT 15h, Funktion 87h).

#### *5.2.1.2. Batterie-Pufferung*

Das statische RAM und der Uhren-IC (RTC) können über eine Batterie mit Spannung versorgt werden. Das RAM behält beim Abschalten der Betriebsspannung seinen Inhalt und die Uhr läuft weiter. Diese Backup-Spannung wird über den Vbatt-Pin des I/O-Steckers zugeführt.

Die minimale Spannung für SRAM und RTC beträgt 2V. Bei einer Lithium-Zelle von 3,2 V verbrauchen SRAM und RTC zusammen etwa 2 µA (typischer Wert). Die Batterie wird nicht belastet, wenn die normale Betriebsspannung von 3,3 V anliegt.

#### *5.2.1.3. Goldcap*

Statt einer Batterie kann auch ein "Goldcap", d.h. ein Kondensator mit besonders großer Kapazität (z.B. 1F) verwendet werden. Hiermit ist ein Datenerhalt über einige Stunden möglich. Der Vorteil gegenüber einer Batterie besteht in der relativen Wartungsfreiheit. Den Ladestrom des Goldcaps führt man während des Normalbetriebs z.B. über eine Diode mit Serienwiderstand (zur Strombegrenzung) zu.

### 5.2.2. Flash-Speicher

#### *5.2.2.1. Flash Speicher Layout*

Der MicroPC kann mit 1 MB, 2 MB, 4 MB oder 8MB Flashspeicher bestückt werden. Dieser ist in Blöcken zu 64 kB organisiert, die einzeln gelöscht werden können. BIOS und ROM-DOS-Kernel belegen 96 kB des Flash-Speichers, 32 kB sind für eine BIOS-Erweiterung reserviert. Der Rest steht als Flash-Disk, ROM-Disk oder als frei vom Applikationsprogramm verwendbarer Speicher zur Verfügung. Hierfür gibt es die BIOS-Funktionen des Int 5Fh.

Ein Teil des Flash-Speichers, nämlich 128 kB, 256 kB oder 512 kB (einschließlich BIOS und ROM-DOS), kann im untersten MB des Adressraums, d.h. im Real Mode Adressraum, eingeblendet werden. Hier kann auch Real-Mode Programm-Code untergebracht werden, der direkt im Flash ausgeführt wird. Dieser Teil des Adressraums geht selbstverständlich für das RAM verloren.

#### *5.2.2.2. Beschränkte Zahl von Lösch-Zyklen*

Der Flash-Speicher wird aus "Large Sector Flash-ICs" (z.B. 29LV800 von AMD oder kompatibel) gebildet. Diese sind für eine beschränkte Zahl von Löschzyklen pro Block ausgelegt (üblicherweise

sind eine Million Löschzyklen vom Hersteller garantiert). Dies bedeutet, daß der Flash-Speicher, insbesondere die Flashdisk, nicht für permanente Schreiboperationen eines Programms geeignet ist, da man mit einem entsprechenden Programm die zulässige Zahl von Löschzyklen pro Block in relativ kurzer Zeit überschreiten kann. Für derartige Zwecke muß eine RAM-Disk verwendet werden.

#### *5.2.2.3. Flashdisk*

Der größte Teil des Flash-Speichers wird normalerweise von der Flash-Disk eingenommen. Diese ist wie eine Festplatte organisiert und wird vom ROM-DOS mittels der BIOS-Funktionen des Int 13h angesprochen.

Bei Verwendung von FreeDOS oder MS-DOS befindet sich das Betriebssystem vollständig auf der Flashdisk und wird beim Booten in das RAM geladen. Bei Datalight ROM-DOS ist der DOS-Kernel als BIOS-Extension in einem schreibgeschützten Bereich des Flashspeichers untergebracht. Die Datei COMMAND.COM muß aber auch in diesem Fall auf der Flashdisk vorhanden sein.

Das Flash-File-System des MicroPC ist äußerst stabil gegen plötzliche Stromausfälle, die vor allem bei batteriebetriebenen Geräten leicht auftreten können. Es ist praktisch ausgeschlossen, daß nach einem Stromausfall das Flash-File-System nicht mehr funktioniert. Eventuell können "verlorene Cluster" entstehen, wenn unter DOS noch Dateien geöffnet waren. Diese lassen sich mit dem DOS-Dienstprogramm CHKDSK wieder löschen.

#### *5.2.2.4. Erstellung einer neuen Flashdisk*

- Erstellen Sie auf dem PC ein Verzeichnis, in welchem alle Dateien abgelegt werden, die später in der Flash-Disk des MicroPC erscheinen sollen.
- Alle benötigten Dateien werden in das zuvor erstellte Verzeichnis kopiert. Hierunter sollten zumindest COMMAND.COM und ein Programm zur Datenübertragung (RDRIVE.EXE UND RMAP.EXE oder XLOAD.COM) vorhanden sein. Außerdem kann man hier passende CONFIG.SYS und AUTOEXEC.BAT Dateien erstellen.

  Wenn FreeDOS zum Einsatz kommt, muß diesem Verzeichnis neben der **command.com** auch **kernel.sys** vorhanden sein. Diverse Boot-Parameter können in der Datei **fdconfig.sys** eingestellt werden (**config.sys** bei Datalight ROM-DOS oder MS-DOS). Name und Ort der Start-Batchdatei (wie die **autoexec.bat** bei MS-DOS oder ROMDOS) können in der fdconfig.sys frei definiert werden (Standardname: **fdauto.bat**).
- Erzeugen Sie mit dem Programm FLASHHDD.EXE eine Flash-Image Datei entsprechend der Beschreibung im Kapitel "PC-Programme".
- Starten Sie das Terminalprogramm und den MicroPC und drücken Sie während des Speichertests die Taste <S>. Sie gelangen in das Setup der CPU-Karte.
- Durch <L> oder mit den Cursor-Tasten auf der Tastatur gelangen Sie zum ***FLASH***-Menü. Hier wählen Sie die Funktion ***Update Flashdisk*** aus und bestätigen Sie mit <Enter>. Die Sicherheitanfrage muß mit 'Y' beantwortet werden. Das BIOS löscht darauf den Flashdisk-Bereich und startet anschließend den Datei-Übertragungsmodus (dies erkennt man am Erscheinen der Protokoll-Zeichen "§"). Schicken Sie hierauf die Datei durch das Terminalprogramm ab (ALT+<S> bei Vterm, Datei-Sende-Befehl). Wählen Sie hierzu das Übertragungsprotokoll „X-Modem“ und geben Sie den Pfad der zuvor angelegten Flash-Image-Datei ein. Einige Sekunden nach Beendigung der Übertragung bootet die Karte neu.

## 5.3. Real Time Clock

Der Echtzeit-Uhrenbaustein (Real Time Clock, "RTC") stellt Anwendungen bei Bedarf Datum und Uhrzeit zur Verfügung. Schaltjahre werden ebenso berücksichtigt wie ein 24-Stunden-Modus. Die RTC kann wie beim PC einen Interrupt (IRQ 8) auslösen. Neben dem PC-üblichen terminierten Interrupt (Interrupt zu einer bestimmten Zeit) kann die RTC auch einen zyklischen Interrupt mit 4096 Hz, 1Hz sowie jede Minute, jede Stunde und jeden Tag auslösen. Für das Auslesen und Programmieren stehen wie beim PC die Funktionen des BIOS-Interrupts 1Ah zur Verfügung.

Die RTC kann über den Vbatt-Pin des MicroPC Steckers mit einer Backup-Spannung (z.B. einer Lithium-Batterie) versorgt werden, so daß sie weiterläuft, wenn die Betriebsspannung des MicroPC abgeschaltet wird. Der Stromverbrauch beträgt dabei nur etwa 3µA.

Das low-aktive RTC-Interrupt-Signal (Open-Drain-Ausgang) ist auf den I/O-Stecker des MicroPC gelegt (-WAKEUP) und kann vom Anwender benutzt werden, um Aktionen der Peripherie auszulösen. Dies funktioniert auch im Deep Power Down Mode, also bei abgeschaltetem CPU-Oszillator. Die Versorgungsspannung des MicroPC muß aber eingeschaltet bleiben, die Backup-Spannung über Vbatt reicht nicht aus. Das -WAKEUP Signal kann auch als externer Interrupt-Eingang verwendet werden, wenn der RTC-Interrupt nicht benötigt wird.

## 5.4. TCU (Timer/Counter-Unit)

Die TCU ist weitgehend kompatibel zum 8254 von Intel (und damit zum PC). Nähere Einzelheiten über die Timer findet man im Manual des 386EX-Prozessors von Intel.

Eigenschaften des 8254:

- drei 16-Bit Zähler,
- sechs programmierbare Zählmodi,
- BCD oder Binäres Zählen,
- eigener Interrupt für jeden Zähler (IRQ 0, 10 und 11),
- Taktquelle intern (PSCLK) oder extern wählbar für Timer 0 und Timer 1, bzw. intern oder COMCLK (1,8432) für Timer 2.

Der **Timer 0** wird vom BIOS nach PC-Standard im Mode 3 auf eine Ausgangsfrequenz von 18,206 Hz programmiert. Sein Ausgang ist mit der Interruptleitung IRQ 0 verbunden. Die BIOS-Interrupt-Routine inkrementiert bei jedem Aufruf die 32-Bit Timer-Variable bei RAM-Adresse 0040h:006Ch, welche die Grundlage für die DOS-Systemzeit ist (periodischer Timer-Interrupt, System-Timer).

Der **Timer 1** - beim normalen PC für den DRAM-Refresh zuständig - wie auch **Timer 2** (beim PC für die Ansteuerung des Lautsprechers) stehen zur freien Verfügung. Beim MicroPC können auch Timer 1 und Timer 2 einen Interrupt auslösen (IRQ 10 und 11).

Der interne Takt (PSCLK) wird durch einen Vorteiler aus dem CPU-Takt (25, 20, 8 oder 4 MHz) erzeugt. Der Vorteiler kann auf Werte von 2 bis 513 eingestellt werden. Das BIOS setzt den Wert des Vorteilers so, daß der Takt etwas langsamer ist als der PC-typische Timer-Eingangstakt von 1,193182 MHz, d.h. als nächsthöhere ganze Zahl zum Quotienten aus 25 MHz und 1,193182 MHz. Man erhält so einen Vorteiler von 21 und einen Timer-Eingangstakt von 1,19047 MHz. Beim Heruntertakten des CPU-Taktes durch den BIOS-Int 15h Funktion C311h auf 20, 8 oder 4 MHz wird der Vorteiler auf 17 bzw. 14 bzw. 7 angepasst.

Man beachte, daß beim Ändern des PSCLK-Vorteilers sowie des Timer0-Teilers beim Umprogrammieren des CPU-Taktes eine gewisse Ungenauigkeit des Timer-Ausgangstaktes entsteht. Braucht man eine Zeitbasis, die trotz häufiger Änderungen des CPU-Taktes durch das Power-Management genau ist, so kann man präsize externe Taktsignale verwenden oder – bei Timer 2 - das COMCLK-Signal, welches auch den Grundtakt für seriellen Schnittstellen darstellt. Dieses liegt außer im Deep Powerdown Mode permanent an.

Folgende Timer-Signale sind nach außen auf den I/O-Stecker X1 geführt:

| Timer | Output | Clock | Gate |
|---|---|---|---|
| Timer 0 | Pin 14 | Pin 24 | Pin 45 |
| Timer 1 | Pin 39 | Pin 16 | Pin 46 |
| Timer 2 | --- | --- | --- |

Die Signale von Timer 0 und Timer 1 sind mit anderen Funktionen zusammengelegt. Die jeweils richtige Einstellung kann im BIOS-I/O-Setup (siehe dort) vorgenommen werden.

## 5.5. Watchdog-Timer

Der 386EX-Watchdog-Timer ist ein 32-Bit Zähler, der beim MicroPC als programmierbarer Watchdog Verwendung findet. Der Eingangstakt entspricht dem CPU-Takt (25 MHz bei vollem Takt), die maximale Watchdog-Timeout-Zeit beträgt demnach etwa 172 Sekunden. Der Watchdog löst am Ende der Timeout-Zeit einen Hardware-Reset aus.

Für die Aktivierung und das Rücksetzen des Watchdogs stehen BIOS-Funktionen des Int 15h zur Verfügung. Einmal aktiviert, kann der Watchdog nur durch einen Hardware-Reset deaktiviert werden.

Die Verwendung des Watchdog-Timers als universellen Timer (mit IRQ15) ist beim MicroPC nur in speziellen Versionen der Hardware möglich (anderer Reset-Generator und Verzicht auf die Watchdog-Funktion), da der Timer-Ausgang beim MicroPC grundsätzlich einen Reset erzeugt.

## 5.6. Interrupt Controller

### 5.6.1. Allgemeines

Der 386EX Prozessor besitzt On-Chip zwei 8259-kompatible Interrupt-Controller ("PIC", Programmable Interrupt Controller) und entspricht damit dem PC. Hierdurch stehen wie beim PC 15 Interrupt-Requests (IRQs) zur Verfügung, die teilweise bereits durch Einheiten des MicroPC belegt sind. Für Anwendungen frei sind die IRQs 1, 5, 7, 9, 13 und 14. Wird kein RTC-Interrupt benötigt, kann auch der IRQ8 anderweitig verwendet werden (RTC-On Signal, Open-Drain Ausgang der RTC). Wenn die erste oder zweite serielle Schnittstelle ohne Interrupt betrieben werden kann oder nicht benötigt wird, so steht auch IRQ4 bzw. IRQ3 extern zur Verfügung. Zu beachten ist, daß der IRQ 8 auf dem Stecker invertiert, also low-aktiv ist.

Der PIC2 ist mit seinem Ausgang an den IRQ2-Eingang des PIC1 angeschlossen. Der IRQ2-Eingang ist deshalb als Slave-Eingang konfiguriert, so daß es den IRQ2 im eigentlichen Sinn nicht gibt. Dem Vektor Int 0Ah ist also auch kein IRQ zugeordnet. Der CPU-Kern besitzt einen allgemeinen Interrupt-Eingang. An diesen ist der Ausgang des PIC1 angeschlossen. Daneben gibt es noch den NMI- und den SMI-Interrupt Eingang. Beim MicroPC werden diese nicht verwendet.

Die Einzelheiten der Programmierung der Interrupt-Controller kann man dem Manual des 386EX-Prozessors von Intel oder der gängigen PC-Literatur entnehmen. Hier sind deshalb nur einige Hinweise aufgeführt.

### 5.6.2. Flanken- und Pegeltriggerung

Die Interrupt-Controller arbeiten traditionell beim PC mit Flankentriggerung. Eine Umprogrammierung auf Pegel-Triggerung ist nicht unbedingt zu empfehlen, da bei den BIOS Interrupt-Funktionen, speziell denen der RTC und des Timer 0, Probleme auftreten werden. Falls der RTC-Interrupt nicht benötigt wird, kann der PIC2 auf Pegel-Triggerung eingestellt werden. Der Vorteil der Pegel-Triggerung: mehrere Einheiten können sich – durch Wired-OR bzw. Wired-AND Verknüpfung - einen IRQ teilen (das geht bei Flankentriggerung nicht gut, da Flanken verloren gehen, wenn zwei Interrupt-Signale auf einer Leitung gleichzeitig auftreten). Der Nachteil der Pegel-Triggerung besteht darin, daß die Interrupt-Quelle innerhalb der Service-Routine sofort zurückgesetzt werden muß (was aber z.B. beim Timer gar nicht möglich ist), da sonst weitere Interrupts auftreten.

### 5.6.3. Zugeordnete Interrupt-Vektoren

Jedem PIC kann per Software-Initialisierung ein Bereich von acht aufeinander folgenden Interrupt-Vektoren zugeordnet werden. Beim PC werden traditionell vom BIOS dem PIC1 die Vektoren Int08 bis Int0Fh und dem PIC2 die Vektoren Int70h bis Int77h zugeordnet. Die Vektoren des PIC1 liegen daher entsprechend der PC-Tradition - wie auch die meisten BIOS-Funktionen - in dem von Intel als "reserved" vorgesehenen Bereich von Int0 bis Int1Fh, d.h. sie teilen sich den Vektor mit gewissen Prozessor-Exceptions. In der Praxis führt dies in den meisten Fällen nicht zu Problemen.

### 5.6.4. Maskieren von Interrupts

Um einen IRQ zu aktivieren, muß das zugehörige Bit im Maskenregister des PIC auf 0 gesetzt werden. Für die IRQs des PIC2 muß zusätzlich das Maskenbit 2 des PIC1 (entsprechend IRQ2) auf 0 gesetzt werden. Die Maskenregister liegen bei den I/O-Adressen 21h bzw. A1h für PIC1 bzw. PIC2.

### 5.6.5. Zurücksetzen der Interrupt-Controller

Jeder IRQ-Eingang besitzt im Interrupt-Controller ein In-Service-Bit. Die Interrupt-Service Routinen müssen grundsätzlich das In-Service-Bit des betreffenden IRQ zurücksetzen, da andernfalls kein weiterer IRQ mit gleicher oder niedrigerer Priorität mehr erzeugt werden kann. Dies geschieht normalerweise durch den "unspezifischen Rücksetzbefehl"

```
out [20h], 20h
```

bzw. für den PIC2:

```
out [0A0h], 20h
```

oder in "C":

```
_outp(0x20, 0x20); _outp(0xA0, 0x20);
```

also durch Ausgabe des Byte 20h auf die I/O-Adresse 20h bzw. A0h. Bei den IRQs des PIC2 muß immer auch das In-Service-Bit des IRQ2 zurückgesetzt werden, man muß also beide der angegebenen Out-Befehle ausführen.

Grundsätzlich ist zu beachten, daß die CPU beim Einsprung in eine Interrupt-Routine zunächst alle Interrupts sperrt durch Rücksetzen des Interrupt-Enable Flags. Dies gilt für IRQ-Service-Routinen wie auch für Software-Interrupts (nur im Protected Mode kann hier für jeden Interrupt eine andere Einstellung gewählt werden). Falls notwendig, kann man die Interrupts innerhalb einer Interrupt-Routine durch den „Set Interrupt“ (STI) Befehl wieder freigeben.

### 5.6.6. Nicht-Speicherung von IRQs

IRQs werden beim 8259 nicht zwischengespeichert (auch wenn das Intel-Manual diesen Eindruck zu erwecken scheint). D.h.: wird das IRQ-Signal von der betreffenden Peripherie-Einheit zurückgenommen, bevor die CPU den IRQ bearbeiten kann (z.B. weil die Interrupts in der CPU gerade gesperrt sind), so geht dieser Interrupt verloren. Nur direkt während eines Interrupt-Acknowledge-Zyklus der CPU werden die Zustände der IRQ-Eingänge eingefroren, um eine eindeutige Ausgabe des Interrupt-Vektors zu ermöglichen. Das Interrupt Request Register (IRR) gibt ansonsten nur den Zustand der IRQ-Eingänge wieder. Dies gilt unabhängig davon, ob ein IRQ ausmaskiert ist. Wenn im flankengetriggerten Modus ein IRQ gerade bearbeitet wird (In-Service-Bit gesetzt), wird das betreffende Bit des IRR als 0 gelesen, da das Edge-Sense-Latch den Eingang sperrt. Das Edge-Sense-Latch wird durch einen Low-Impuls an dem IRQ-Eingang zurückgesetzt, auch wenn das In-Service-Bit noch gesetzt ist, so daß man dann den IRQ-Eingang via IRR wieder einlesen kann.

### 5.6.7. Spurious Interrupt

Die Nicht-Speicherung von IRQ-Impulsen macht auch den Default- oder Spurios-Interrupt notwendig. Dieser tritt bei "unsauberen" IRQ-Signalen auf, z.B. bei IRQs von prellenden Tastaturen. Wenn die CPU als Reaktion auf einen IRQ einen INTA-Zyklus durchführt, der PIC den zugehörigen IRQ-Eingang aber bereits vergessen hat (da das Signal zurückgenommen wurde), muß der PIC dennoch einen Interrupt-Vektor auf den Datenbus ausgeben (ein Zufallswert auf dem Datenbus könnte sonst zum Absturz des Rechners führen). Dieses ist dann der "Spurios IRQ7". Er wird auch dann aufgerufen, wenn der IRQ7 ausmaskiert ist. Das In-Service-Bit des IRQ7 wird beim Spurios IRQ nicht gesetzt. Beim PIC2 kann auch ein Spurios IRQ15 auftreten. Ob ein IRQ7 oder ein IRQ15 auftritt, hängt dann vom Timing des Eingangssignals ab. Wegen der Durchlaufverzögerung des PIC2 liegt das IRQ-Signal am PIC1 etwas länger an, so daß eventuell der PIC1 den IRQ nicht als "Spurios" betrachtet und den PIC2 aktiviert, der dann einen Spurious IRQ15 erzeugt.

### 5.6.8. IRQ-Priorität

Die Priorität der IRQs ist beim PC üblicherweise so festgelegt, daß der IRQ0 die höchste Priorität besitzt, die anderen IRQs folgen gemäß ihrer Nummer. Da der PIC2 an IRQ2 angeschlossen ist, liegen die IRQs des PIC2 der Priorität entsprechend vor dem IRQ3. Die Service-Routinen des PIC2 können allerdings nicht von IRQs höherer Priorität des PIC2 unterbrochen werden, solange das In-Service-Bit des IRQ2 nicht zurückgesetzt wurde. Setzt man aber dieses Bit zurück, können sie auch durch IRQs des PIC1 mit niedrigerer Priorität unterbrochen werden.

Die Prioritäten können auch geändert werden, allerdings nur zyklisch innerhalb der einzelnen PICs. D.h. man legt per Befehl den IRQ mit der niedrigsten Priorität fest, woraus sich automatisch die anderen Prioritäten innerhalb dieses PICs ergeben. Die Festlegung der niedrigsten Priorität erfolgt für den PIC1 bzw. PIC2 durch den Befehl

```
out [20h], 0C0h+ IRQ-Nr
```

bzw.

```
out [0A0h], 0C0h+ IRQ-Nr
```

Z.B. legt

```
out [20h], 0C3h
```

den IRQ3 auf die niedrigste Priorität, wodurch der IRQ4 (von COM1) die höchste Priorität bekommt.

## 5.7. Asynchrone serielle Schnittstellen

Zwei zu dem bekannten 16C450 kompatible UARTs sind auf dem 386EX-Prozessor integriert. Im Gegensatz zu den heute auf PC-Mainboards integrierten UARTs (welche zum 16C550 kompatibel sind) besitzen sie daher auch keine FIFOs.

Die erste serielle Schnittstelle (COM1) wird standardmäßig als Ein-/Ausgabeeinheit des MicroPC verwendet (Gerät "CON" von DOS). Man kann für CON aber auch die COM2 oder eine externe serielle Schnittstelle einstellen (möglich sind COM1 bis COM4).

### 5.7.1. Signale der seriellen Schnittstellen

Beide serielle Schnittstellen besitzen die beim PC üblichen acht Signale: Datenleitungen (RXD, TXD), Modem-Status-Eingänge (DSR, CTS, DCD und RI) und Modem-Control-Ausgänge (RTS, DTR). Häufig wird eines der Paare DTR/DSR oder RTS/CTS für "Handshake-Betrieb" verwendet.

Die acht Signale der COM1 können einzeln als Ein- oder Ausgabeports umkonfiguriert werden (s. Kapitel "I/O-Ports").

Die Signale der RTS, DTR, DSR und RI der COM2 können wahlweise auch als Signale der synchronen seriellen Schnittstelle verwendet werden (s. dort).

### 5.7.2. BIOS Funktionen

Das BIOS stellt die PC-üblichen INT 14h-Funktionen zur Bedienung der seriellen Schnittstelle zur Verfügung. Abweichend von der PC-Tradition werden diese beim MicroPC mit Empfangs-Interrupt betrieben. Diese Einstellung kann im BIOS-Setup deaktiviert werden. Die im Setup einstellbare Buffer-Größe ist nur bei Interrupt-Betrieb wirksam. Um einen geringeren Rechenzeitaufwand zu erreichen, werden die seriellen Schnittstellen in vielen Anwendungen auch direkt, d.h. durch Zugriff auf die UART-Register, programmiert. Hierbei wird üblicherweise zumindest der Receive-Interrupt verwendet.

### 5.7.3. Hardware-Interrupt der seriellen Schnittstellen

COM1 und COM2 belegen wie beim PC die IRQ 4 und 3. Falls man für die seriellen Schnittstellen die Interrupts nicht benötigt, können IRQ4 und 3 auch extern für andere Zwecke verwendet werden.
Es gibt pro UART vier verschiedene Interrupt-Quellen (die aber alle die gleiche IRQ-Leitung verwenden):

- Line-Status-Interrupt: bei Overrun-, Parity- oder Framing-Error oder Break ;
- Receive-Interrupt: ein Zeichen wurde komplett empfangen (Receive Buffer full);
- Transmit-Interrupt: ein Zeichen wurde komplett gesendet (Transmit Buffer empty);
- Modem-Status-Interrupt: der Zustand von DCD, RI, CTS oder DSR hat sich geändert *.

Dabei bedeutet

Overrun-Error: Das Receive-Buffer-Register wurde durch ein weiteres Zeichen überschrieben, d.h. das vorige Zeichen wurde vom Prozessor nicht rechtzeitig abgeholt;

Parity-Error: Die Parität oder auch das Paritätsbit (bei forced parity) des empfangenen Zeichens war ungleich dem Komplement des Even Parity Select (EPS) Bit im Line Control Register. Dies geschieht nur bei eingeschaltetem Parity-Bit. Bei "forced parity" kann man das Parity-Bit ähnlich wie ein neuntes Datenbit verwenden, da man es beim Senden mittels des EPS Bits setzen kann und beim Empfangen mittels des Line Status Interrupts auswerten kann.

Framing-Error: Zeichen hatte kein Stop-Bit (oder ein Stop-Bit zuwenig, wenn 1,5 oder 2 Stopbits eingestellt sind). Ein gültiges Stop-Bit bedeutet high-Zustand der RXD-Leitung.

Break-Bedingung: Die RXD-Leitung ging für die Dauer von mehr als einem Zeichen auf low. Das hat immer auch einen Framing-Error zur Folge. Eine Break-Bedingung kann beim Senden durch Setzen des Break Bits des Line-Control-Registers erzeugt werden. Die Mindestzeitdauer von einem Zeichen erreicht man, indem man anschließend ein Zeichen sendet und wartet, bis das Transmit Shift Register leer ist. Danach wird das Break Bit wieder zurückgesetzt. Man kann auf diese Weise dem Empfänger einen Sonderzustand signalisieren, ohne auf bestimmte Byte-Werte oder Byte-Sequenzen zurückgreifen zu müssen.

* bei RI: nur steigende Flanken erzeugen einen IRQ.

### 5.7.4. Register der UARTs

Divisor-Latch low (DLL, Adresse 0)

Divisor-Latch high (DLH, Adresse 1)

Interrupt Enable Register (IER, Adresse 1):
- Bit 0: Receive Interrupt
- Bit 1: Transmit Interrupt
- Bit 2: Line Status Interrupt
- Bit 3: Modem Status Interrupt
- Bit 4..7: 0

Line Control Register (LCR, Adresse 3):
- Bit 0: Word Length Bit 0
- Bit 1: Word Length Bit 1
- Bit 2: No. of Stop Bits (1 or 2)
- Bit 3: Enable Parity Bit
- Bit 4: Select Even Parity
- Bit 5: Select Forced Parity
- Bit 6: Set Break
- Bit 7: Divisor Latch Enable

Interrupt Identification (Status) Register (IIR oder ISR, Adresse 2):
- Bit 0: 0 = Interrupt Pending
- Bit 1..2: 0 = Modem Status Interrupt
  1 = Transmit Interrupt
  2 = Receive Interrupt
  3 = Line Status Interrupt
- Bit 3..7: 0

Modem Control Register (MCR, Adresse 4):
- Bit 0: /DTR
- Bit 1: /RTS
- Bit 2: OUT1: Test Bit für /RI im Loop Back Mode
- Bit 3: OUT2: Test Bit für /DCD im Loop Back Mode; aktiviert den UART-Interrupt*
- Bit 4: Set Loop-Back Mode
- Bit 5..7: 0

Line Status Register (LSR, Adresse 5):
- Bit 0: Received Data Ready
- Bit 1: Overrun Error
- Bit 2: Parity Error
- Bit 3: Framing Error
- Bit 4: Break Condition
- Bit 5: Transmitter Hold Register Empty
- Bit 6: Transmitter Shift Register Empty
- Bit 7: 0

Modem Status Register (MSR, Adresse 6):
- Bit 0: Delta CTS
- Bit 1: Delta DSR
- Bit 2: Delta RI
- Bit 3: Delta DCD
- Bit 4: /CTS
- Bit 5: /DSR
- Bit 6: /RI
- Bit 7: /DCD

Scratch Register (SCR, Adresse 7)

* OUT1 und OUT2 sind ursprünglich universelle digitale Ausgänge der UARTs 8250 und 16C450. Beim PC wird traditionell mittels OUT2 über einen Tristate-Buffer der Interrupt-Ausgang des UARTs zum Interrupt-Controller durchgeschaltet bzw. vom Interrupt-Controller getrennt (deaktiviert). Gewisse UARTs besitzen dieses Gatter bereits intern, so der Exar/Startech ST16C552. Beim 386EX wird durch OUT2 ein Multiplexer gesteuert, durch den alternativ ein externes Signal (Pin) statt des UART-Interrupts zum Interrupt-Controller weitergeleitet wird.

## 5.8. Synchrone serielle Schnittstelle

Der 386EX-Prozessor besitzt neben den beiden asynchronen auch eine synchrone serielle Schnittstelle (SSIO). Für Senden und Empfangen gibt es jeweils eine Daten- und eine Taktleitung. Diese vier Signale liegen auf dem X1-Stecker auf den gleichen Pins wie die Signale DTR, DSR, RI und RTS der COM2.

Die SSIO kann mit maximal dem halben Prozessortakt (CLK2 / 4) betrieben werden, beim MicroPC mit 25 MHz CPU-Takt also mit maximal 12,5 MBaud. Übertragen werden 16-Bit Worte.

Sowohl Empfänger als auch Sender können im Master- oder im Slave-Mode, d.h. mit aktivem oder passivem Takt arbeiten. Der Interrupt IRQ9 wird optional für die Zustände "Transmit Buffer empty" und "Receive Buffer Full" erzeugt.

Bei der Verwendung der synchronen seriellen Schnittstelle sollte man beachten, daß diese herstellerseitig zwei Bugs enthält, die von Intel zwar dokumentiert, jedoch nie behoben wurden:

- Auto-Transmit Mode. Dieser wäre an sich der normale Master-Transmit Mode. Er ist jedoch nur bei maximaler Baudrate richtig verwendbar, da das erste Bit eines Datenwortes unabhängig von der eingestellten Baudrate mit maximalem Takt ausgegeben wird.
- Das Transmit Buffer Empty Bit des Status-Registers gibt keinen korrekten Wert zurück. Man muß vielmehr das Baudrate-Counter Register "pollen" und abwarten, bis das erste Bit vollständig ausgegeben wurde, bevor man den Transmitter abschalten darf.

## 5.9. I/O-Ports

Der MicroPC besitzt auf seinem Stecker maximal 32 frei programmierbare digitale I/O-Ports. Diese können zum größten Teil unabhängig voneinander als Eingang oder Ausgang konfiguriert werden. Die Pins werden teilweise alternativ auch vom Timer, Interrupt-Controller und der ersten seriellen Schnittstelle verwendet (s. BIOS-I/O-Setup).

Von den I/O-Ports gehören 17 zu den internen Port-Registern des Prozessors. Weitere 15 werden durch die Daten-, Adress- und Kontroll-Signale des PIF-Bus gebildet.

### 5.9.1. Prozessor-Ports

Angesprochen werden diese digitalen I/O-Ports als Teil der 8-Bit Ports P1, P2 und P3 des 386EX-Prozessors.

Jeder 8-Bit Port besitzt die Register

- PnPIN (Pin-Zustandsregister) zum Einlesen der aktuell anliegenden Pegel,
- PnLTC (Ausgangsregister, Latch-Register) zum Setzen der Ausgangszustände,
- PnDIR (Richtungsregister) zum Einstellen von Input- oder Output-Mode.

Die Richtung (Eingang oder Ausgang) wird im BIOS-I/O-Setup oder mittels des Richtungs-Registers des Ports festgelegt (0 = Ausgang, 1 = Eingang und Open-Drain Ausgang). Wenn ein Pin als Eingang verwendet werden soll, muß das betreffende Bit des Ausgangsports auf 1 stehen, da nur dann der zugehörige Open-Drain-Ausgang abgeschaltet ist.

| | Zustand | Ausgang | Richtung |
|---|---|---|---|
| P1 | F860h | F862h | F864h |
| P2 | F868h | F86Ah | F86Ch |
| P3 | F870h | F872h | F874h |

Im Einzelnen sind folgende Bits dieser Register auf dem Stecker zugänglich (siehe auch Tabelle im Kap. 10):

| Port-Bit | Alternative Funktionen |
|---|---|
| P1.0 | DCD von COM1 oder IRQ14 oder Gate von Timer 1 |
| P1.1 | RTS von COM1 |
| P1.2 | DTR von COM1 |
| P1.3 | DSR von COM1 oder IRQ9 oder Gate von Timer 0 |
| P1.4 | RI von COM1 |
| P2.0 | –CS0 des PIF-Bus |
| P2.1 | –CS1 des PIF-Bus |
| P2.2 | –CS2 des PIF-Bus |
| P2.3 | –CS3 des PIF-Bus |
| P2.5 | RXD von COM1 |
| P2.6 | TXD von COM1 |
| P2.7 | CTS von COM1 |
| P3.0 | IRQ4 oder Ausgang von Timer 0 (TOUT0) |
| P3.1 | IRQ3 oder Ausgang von Timer 1 (TOUT1) |
| P3.2 | IRQ1 |
| P3.3 | IRQ5 |
| P3.5 | IRQ7 oder SCL des I2C-Bus |

Die alternativen Funktionen können im BIOS-Setup eingestellt werden. In diesem Fall ist der Port-Mode des jeweiligen Pins nicht zugänglich.

Die hier nicht aufgeführten Bits der Register der Ports P1, P2 und P3 sollten durch Anwendungsprogramme nicht geändert werden, da sie innerhalb des MicroPC verwendet werden.

### 5.9.2. PIF-Bus-Ports

Die Signale PD0..PD7, PA0..PA3, –CS0.. –CS3, –RD, –WR und READY des PIF-Bus können alternativ zum Busbetrieb, teilweise auch gleichzeitig, als Digital-Ports verwendet werden. Die Signale –RD, –WR und READY werden im Port-Mode auch mit PA4, PA5 und PA6 bezeichnet, da sie gemeinsam mit PA0..PA3 über das Register 101h angesprochen werden können.

Die Chip-Selects –CS0 bis –CS3 sind im Port-Mode den Prozessor-Ports zugeordnet (s. das vorangehende Kapitel). Die übrigen Pins werden mit Hilfe der I/O-Adressen 100h bis 103h angesprochen (s. auch Tabelle der I/O-Adressen im Kapitel "Tabellen"). Die Signale –RD, –WR und READY können einzeln tristate geschaltet werden, bei PD0..PD7 und PA0..PA3 geschieht dies in Vierer-Gruppen (PD0..PD3, PD4..PD7, PA0..PA3).

Für den PIF-Bus-Betrieb müssen alle Signale außer READY auf Output geschaltet werden. Dies geschieht mit Hilfe des PIF-Konfigurations-Registers 102h, z.B. in 'C' durch den Befehl

```
_outp(0x102, 0x3F)
```

Das Bit 7 des PIF-Konfigurations-Registers dient zur Aktivierung des Fast-PIF-Mode, der die Zahl der Waitstates bei PIF-Bus Zugriffen herabsetzt.

Die Zustände der Pins –RD, –WR und READY können auch über das Toggle-Register 103h geändert werden. Die Ausgabe einer 1 auf das zugehörige Bit dieses Registers bewirkt die Invertierung des Signals, eine 0 läßt es unverändert.

## 5.10. I2C-Bus

Das BIOS stellt einige einfache Funktionen für den Zugriff auf den I2C-Bus zur Verfügung (s. BIOS-Referenz). Sämtliche der auf den Stecker des MicroPC herausgeführten Signale der Ports P1, P2 und P3 können von den BIOS-Funktionen als SDA- und SCL-Signale des I2C-Bus verwendet werden. Die betreffenden Port-Pins werden im I/O-Menü des BIOS-Setups eingestellt. Der Port-Pin P3.5 wird bereits für den internen I2C-Bus als SCL-Signal verwendet (für die RTC) und ist deshalb auch für die Verwendung beim externen I2C-Bus besonders geeignet.

Pull-Up Widerstände in der Größenordnung 1kOhm bis 4,7 kOhm müssen außer beim Port P3.5 vorgesehen werden. Für den I2C-Bus im BIOS-I/O-Setup ausgewählte Pins müssen dort zusätzlich als "Input"deklariert werden.

Da es sich bei den Ports P1, P2 und P3 um CPU-Ports handelt, beträgt der maximal zulässige Signalpegel nur 3,3 V. Um auch 5V-Signale an den I2C-Bus anzuschließen, kann man externe N-Kanal MOSFETs verwenden (je einen für SDA und SCL). Deren Gate wird an die 3,3V Versorgungsspannung angeschlossen, Source an das MicroPC-seitige Bus-Signal und Drain an das Peripherie-seitige (5V) Bus-Signal.

## 5.11. PIF-Bus

### 5.11.1. Überblick

Der PIF-Bus ist ein einfacher 8-Bit-Erweiterungsbus zum Anschluß von Peripherie-Karten an den MicroPC und andere CPU-Module. Die Bus-Architektur ist an die Schnittstellen diverser LCDs angelehnt (deren Stecker-Belegung jedoch nicht einheitlich ist). So lassen sich LCDs mit dem Controller Toshiba T6963C sogar direkt am PIF-Bus betreiben.

Der Adressraum besteht aus 64 I/O-Adressen. Es werden jedoch nicht 6 Adressleitungen verwendet, sondern 4 Chip-Select-Leitungen und 4 Adressleitungen. Von den Chip-Select-Leitungen ist stets nur eine einzige aktiv (1 aus 4 Code). Jedem Chip-Select sind somit 16 I/O-Adressen zugeordnet. Durch dieses Prinzip vereinfacht sich die Adressdekodierung.

In vielen Fällen wird man sogar ganz ohne Adressdekodierung auskommen. So kann man etwa den bekannten PIO-Baustein 82C55 direkt am PIF-Bus betreiben, indem man die Signale -CS0, -RD, -WR, PA0, PA1, die Datenleitungen sowie die Betriebsspannung verwendet. Hierbei würden von den 16 Adressen, die zu Chip-Select 0 (-CS0) gehören, effektiv nur vier verwendet werden, obwohl alle 16 belegt sind. Diese "Verschwendung" von Adressen ist in vielen Systemen, die nur wenig Peripherie benötigen, kein Problem und vereinfacht das Design.

Wesentlich sind die low-aktiven Read- (-RD) und die Write-Leitungen (-WR), von denen bei jedem PIF-Bus-Zugriff genau eine aktiv ist, je nachdem, ob es sich dabei um einen Lese- oder einen Schreibzyklus handelt. Die Daten werden jeweils auf der **steigenden** Flanke, also gegen Ende des Bus-Zyklus, übernommen.

### 5.11.2. Hardware-Design für den PIF-Bus

Die folgenden Punkte müssen beachtet werden, wenn man Hardware für den Anschluß an den PIF-Bus entwirft.

1. Der Zugriff auf PIF-Peripherie erfolgt durch I/O-Befehle. "Memory-Mapped" Zugriffe sind nicht möglich.
2. Die vier Adressleitungen des PIF-Bus entsprechen den untersten vier Adressleitungen des CPU-Busses. Sie können daher jeden Offsetwert von 0 bis 0Fh annehmen.
3. Genau eine Chip-Select Leitung ist bei einem gültigen PIF-Bus Zugriff aktiv (low).
4. Die vier Chip-Select Leitungen werden aus den Adressleitungen A4 und A5 des CPU-Busses dekodiert. Sie entsprechen daher Offset-Werten von 0h, 10h, 20h und 30h.
5. Die Basis-Adresse des PIF-Bus wird zu den genannten Offset-Werten addiert. Beim MicroPC ist sie 300h. Sie kann bei anderen CPU-Karten anders liegen.
6. Genau eines der Signale -RD und -WR ist während eines gültigen PIF-Bus Zugriffs aktiv (low). Die Peripherie muß diese Signale wie auch die Chip-Select-Signale auswerten, andernfalls können fehlerhafte Bus-Zyklen stattfinden.
7. Die Daten werden sowohl beim Lesen als auch beim Schreiben mit der steigenden Flanke des -RD bzw. -WR Signals übernommen.
8. Die Dauer eines PIF-Bus Zyklus kann im PIF-Bus-Konfigurations-Register auf 1µs oder 320ns (bei 25 MHz CPU-Takt) eingestellt werden. Eine Veränderung des CPU-Taktes bewirkt eine entsprechende Veränderung dieser Werte.
9. Ready-Signal: Dieses Signal wird von der Peripherie-Hardware erzeugt, um PIF-Bus Zyklen zu verlängern. Adressen, Chip-Select und -RD oder -WR bleiben so lange gültig, bis die Peripherie das Ready-Signal wieder freigibt (auf high schaltet). Das Signal besitzt auf der CPU-Karte einen Pull-Up-Widerstand. Die Peripherie muß Open-Collector (Open-Drain) Ausgänge verwenden, wenn mehr als eine Peripherie-Einheit das Ready-Signal verwendet.

### 5.11.3. PIF-Bus: Mechanik

Die PIF-Bus Signale liegen bei der MicroPC-Base Platine auf einer 50poligen, zweireihigen Stiftleiste im 2,54mm Raster. Die Pin-Belegung der ersten 26 Pins ist kompatibel zu der bei diversen CPU-Karten von *taskit* verwendeten 26-poligen Wannenstiftleiste. Dadurch können übliche Flachbandkabel mit Pfostenverbindern in Schneidklemmtechnik eingesetzt werden. Bei der Verwendung von

Flachbandkabeln sollte die Kabellänge 30 cm nicht überschreiten, um Störungen durch Übersprechen und Leitungsreflexionen gering zu halten.

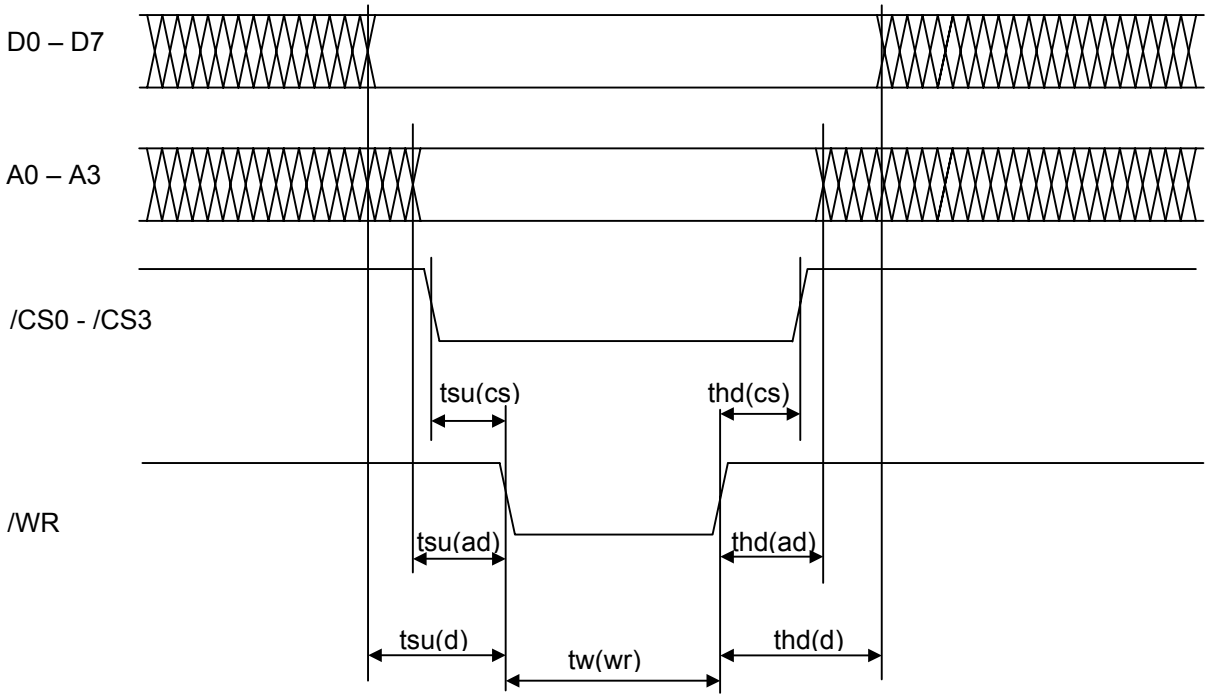
Entfernungen bis ca. 1,5m sind möglich, wenn zusätzliche GND-Leitungen verwendet werden. Insbesondere sollten dann die -RD und -WR Signale durch GND-Leitungen voneinander und von den anderen Signalen abgeschirmt werden. Diese GND-Leitungen sollten an beiden Kabelenden miteinander verbunden sein. Die CPU-Karte muß sich an einem Ende des Kabels befinden. Ausgangsseitige Abschlusswiderstände von 39 Ohm in Reihe zu –RD und –WR sind zu empfehlen, wenn sich die angeschlossenen Peripherie-Module nur im Bereich des Kabelendes befinden.

Verschiedene lieferbare PIF-Cards sind mit einer Buchsen-/Stecker-Kombination ausgestattet, die auch das Stapeln mehrerer Karten ermöglicht.

### 5.11.4. PIF-Bus Signale

| Signal | Pin-Nr. X2 Starterkit-Platine | I/O | aktiv | Beschreibung |
|---|---|---|---|---|
| D0 ... D7 | 11 ... 18 | I/O | high | Datenleitungen |
| –CS0 ... –CS3 | 7, 22, 23, 24 | O | low | Chip-Select. Bei jedem PIF-Bus Zyklus ist genau ein Chip-Select aktiv |
| A0 ... A3 | 8, 9, 20, 21 | O | high | Adressleitungen |
| –RD | 6 | O | low | Read-Signal. Ist bei jedem Lese-Zugriff aktiv. |
| –WR | 5 | O | low | Write-Signal. Ist bei jedem Schreib-Zugriff aktiv. |
| –RESET | 10 | O | low | Reset-Signal. Dies ist das Ausgangssignal des Reset-Generators des MicroPC. |
| /INT | 25 | I | low | Interrupt-Request. Beim MicroPC wird dieses Signal invertiert an den IRQ1 geführt. Es besitzt einen 10kOhm Pull-Up Widerstand. |
| READY | 19 | I | high | Dient zur Verlängerung von PIF-Bus-Zyklen durch Peripherie-Einheiten (low = not ready). Der Bus-Zyklus wird von der CPU erst beendet, wenn das Ready-Signal wieder high ist. |
| VCC | 3 | | | 5V-Versorgungsspannung |
| VEE | 4 | | | Negative Versorgungsspannung für die Kontrastspannung von bestimmten LCDs (wird vom MicroPC nicht verwendet) |
| GND | 1, 2, 26 | | | Masse (negativer Anschluß der Versorgungsspannung) |

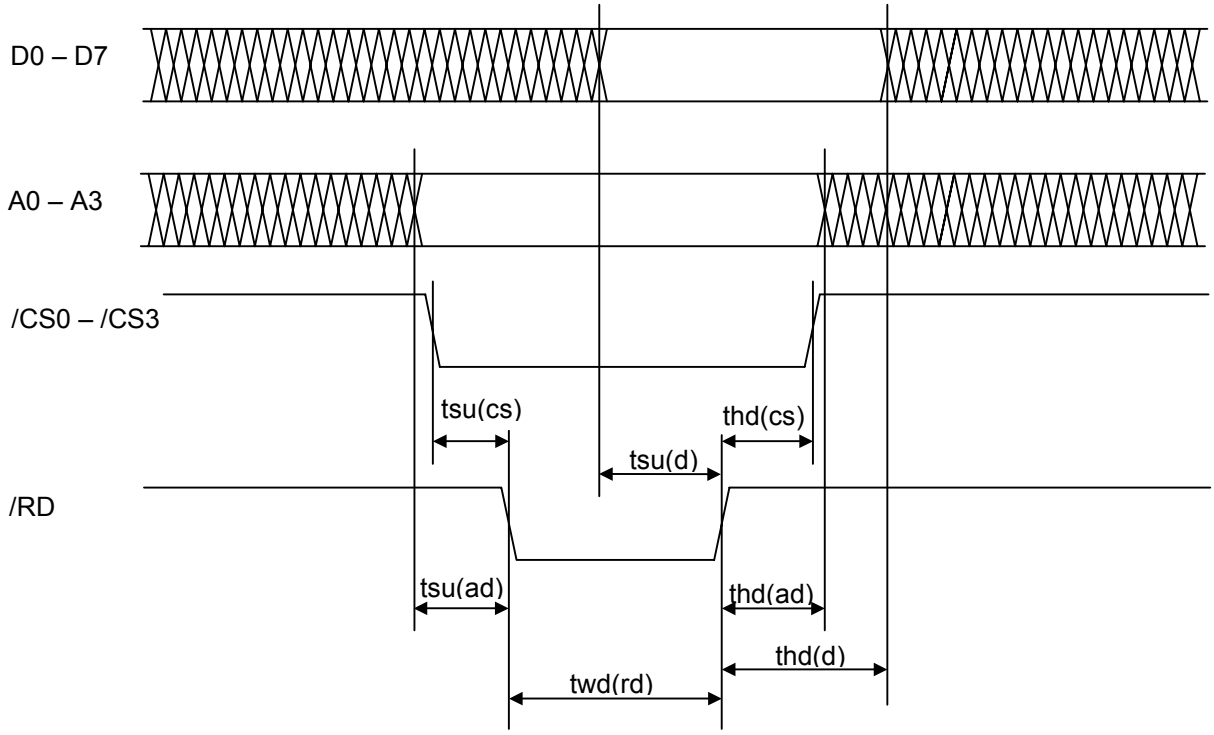
### 5.11.5. PIF-Bus-Timing (Write)




| Symbol | Bezeichnung | Beschreibung | min. | typ. | max. |
|---|---|---|---|---|---|
| tsu(cs) | Chip Select setup time | –CSn low to –WR low | 120 ns | 4 CLK cycles | 160 ns |
| thd(cs) | Chip Select hold time | –WR high to –CSn high | 70 ns | 2 CLK cycles | 80 ns |
| tsu(ad) | Address setup time | A0..A3 valid to –WR low | 120 ns | 3 CLK cycles | |
| thd(ad) | Address hold time | A0..A3 valid after –WR high | 70 ns | 2 CLK cycles | |
| tsu(d) | Data setup time | D0..D7 valid to –WR low | 120 ns | 3 CLK cycles | |
| thd(d) | Data hold time | D0..D7 valid after –WR high | 70 ns | 2 CLK cycles | |
| tw1(wr) | Write pulse width | –WR low to –WR high (fast mode) | 800 ns | 20 CLK cycles | |
| tw2(wr) | Write pulse width | –WR low to –WR high (normal mode) | 160 ns | 4 CLK cycles | |
| tclk | Clock cycle | Clock cycle length at CLK2 = 50 MHz | | 40 ns | |

Bemerkung: Die relativ großzügigen Setup- und -Hold-Zeiten sind zwar beim MicroPC realisiert, dies ist jedoch nicht Standard für alle CPU-Karten mit PIF-Bus. Allgemein werden beim PIF-Bus Setup- und -Hold-Zeiten nur größer als 0 garantiert, d.h. die Chip-Selects, Adressen und Daten sind nur während des /IOW Impulses stabil.

### 5.11.6. PIF-Bus-Timing (Read)




| Symbol | | Beschreibung | min. | typ. | max. |
|---|---|---|---|---|---|
| tsu(cs) | Chip Select setup time | –CSn low to –RD low<br>(Chip Select valid to Read valid) | 120 ns | 4 CLK cycles | |
| thd(cs) | Chip Select hold time | –RD high to –CS high<br>(Chip Select hold after Read invalid) | 70 ns | 2 CLK cycles | |
| tsu(ad) | Address setup time | A0..A3 valid to –RD low | 120 ns | 3 CLK cycles | |
| thd(ad) | Address hold time | A0..A3 valid after –RD high<br>(address hold after Read invalid) | 70 ns | 2 CLK cycles | |
| tsu(d) | Data setup time | D0..D7 valid to –RD high<br>(Data valid to Read invalid) | | | 20 ns |
| thd(d) | Data hold time | D0..D7 valid after –RD high<br>(data hold after Read invalid) | 0 ns | | |
| tw1(rd) | Read pulse width | Normal Mode | 800 ns | 20 CLK cycles | |
| tw2(rd) | Read pulse width | Fast Mode | 160 ns | 4 CLK cycles | |
| tdly(d) | Data delay time | READY valid to data valid delay | | | 20ns |
| tdly(rd) | Read delay time | READY valid to Read invalid delay | 1 CLK cycle | | 2 CLK cycles |
| tclk | Clock cycle | Clock cycle length at CLK2 = 50 MHz | | 40 ns | |

### 5.11.7. Besonderheiten des PIF-Bus beim MicroPC

Der Datenbus wird nur während Lese-Operationen hochohmig geschaltet. In der übrigen Zeit ist er niederohmig und hält das zuletzt geschriebene oder gelesene Byte.

Die Adreßleitungen A0..A3 sowie die Signale –RD und –WR ändern ihren Pegel nur durch Zugriffe auf den PIF-Bus oder das ihnen zugeordnete Portregister 101h. Man kann daher mit Hilfe des Registers 101h eine Verlängerung der Adress-Setup Zeit für PIF-Bus Zugriffe erzielen.

Die vier Chipselects –CS0...–CS3 sind flexibel einsetzbar. Da diese Signale direkt von der Chip-Select-Unit der CPU erzeugt werden, können mit Hilfe der Adreß- und Masken-Register der Chip-Select-Unit sowohl die Lage als auch die Größe des PIF-Adreßraums verändert werden. So kann man statt des üblichen 1-aus-4-Codes der Chip-Selects (entsprechend einem Adreßraum von 64 Adressen) diese so konfigurieren, daß sie wie zusätzliche Adreßleitungen funktionieren. Man erzielt dadurch einen Adreßraum von maximal 256 Adressen.

Daneben können –CS0...–CS3 auch als Port-Pins verwendet werden (Einstellung im BIOS-Setup). Die Chip-Select-Funktionalität (low-Impuls bei PIF-Bus Lese und Schreibzyklen mit gleichzeitigem –RD oder –WR Impuls) ist dann nicht mehr gegeben.

### 5.11.8. 386EX Bus-Monitor und PIF-Bus

**Achtung:** Die folgenden Bermerkungen zum Bus-Monitor betreffen nur solche Exemplare des MicroPC, die vor Juli 2004 gefertigt wurden. Für die später gefertigen MicroPCs ist das Problem hardwareseitig gelöst.

Da der MicroPC den internen Watchdog-Timer der CPU als Watchdog verwendet, steht dieser nicht mehr als Bus-Monitor zur Verfügung. Es kommt dann beim Zugriff auf nicht belegte I/O-Adressen zu einer "Ready-Hang-Condition" (Prozessor bleibt stehen). Ein Beispiel ist der Port 61h (NMI-Status-Register beim PC, beim MicroPC nicht vorhanden), der vom Ethernet-Packet-Driver (und auch anderen Netzwerk-Treibern) eingelesen wird, um anhand der Dauer eines I/O-Read Befehls eine definierte Verzögerung zu erzeugen. Betroffen sind grundsätzlich solche Programme, die bereits fertig vorliegen, ohne daß man die Möglichkeit einer Änderung hat.

Als Bus-Monitor wird deshalb ein ansonsten nicht benutztes Chip-Select der Chip-Select-Unit des 386EX-Prozessors verwendet. Hierfür kommen nur die vier PIF-Chip-Selects in Frage. Von diesen verwenden wir eines, dessen Pin als Port-Pin konfiguriert ist. Das zugehörige Chip-Select ist dann nicht aus dem Prozessor herausgeführt. Es wird so konfiguriert, daß es alle I/O-Adressen umfaßt. Eine "Ready-Hang" Bedingung durch I/O-Befehle kann dann nicht mehr entstehen.

Falls alle PIF-Chip-Select-Signale als Chip-Select verwendet werden, sieht die BIOS-Initialisierung keinen Bus-Monitor vor. Es besteht die Möglichkeit, kritische I/O-Adressen dennoch mit Hilfe von CS0..CS3 abzudecken unter Verwendung der Masken-Register der Chip-Selects (s. Manual des 386EX von Intel). Ein Zugriff auf eine solche Adresse (z.B. Port 61h) ruft dann auch einen low-Impuls der betreffenden CS-Leitung hervor. Das ist normalerweise nicht schlimm, da ein gültiger PIF-Bus Zyklus auch einen low-Impuls der –RD oder –WR-Leitung voraussetzt, der von der Peripherie auscodiert werden sollte. CS-Impulse durch Adressen außerhalb des gültigen PIF-Bus Bereichs rufen dagegen keinen –RD- oder –WR-Impuls hervor.

## 5.12. CompactFlash

CompactFlash Cards (CF-Cards) sind weit verbreitete, international standardisierte Speicher-Module. Sie kommen u.a. in Digital-Kameras zum Einsatz. CF-Cards werden vom BIOS wie Festplatten angesprochen und lassen sich vom PC aus über PCMCIA-Slots (PC-Card Slots) auslesen und beschreiben (mit PCMCIA-Adapter). Die Medien fassen gegenwärtig (April 2005) maximal 8 GB. Daneben gibt es CompactFlash-kompatible 1-Zoll Festplatten von IBM bzw. Hitachi mit 340 MB bis 6 GB (Microdrive™).

Das MicroPC-BIOS unterstützt bis zu zwei an den PIF-Bus angeschlossene CF-Cards.

CF-Cards werden im BIOS-Setup wie Festplatten eingetragen. Normalerweise verwendet man den LBA-Modus, eventuell auch CHS-Modus. Sie bekommt unter DOS den Laufwerks-Buchstaben D:. Falls jedoch im Setup der Eintrag "Swap Hdd0 and Hdd1" auf "disable" gestellt wurde, wird C: zugeordnet. Für die Partitionierung und Formatierung können die DOS-Programme FDISK und FORMAT verwendet werden.

Ein Schaltungsbeispiel für den Anschluß von CompactFlash Cards an den PIF-Bus ist im Anhang angegeben.

**Achtung:** Es wird empfohlen, fabrikneue CF-Cards auf dem MicroPC neu zu partitionieren und zu formatieren. D. h. man sollte mit FDISK alle vorhandenen Partitionen (das ist meist nur eine) zunächst löschen und anschließend neu anlegen. Dadurch wird auch der Bootsektor der Partition gelöscht, was durch FORMAT allein nicht geschieht. Gelegentlich sind CF-Cards mit Formatierungen im Umlauf, deren im Bootsektor gespeicherte CHS-Geometrie nicht mit derjenigen des internen IDE-Controllers übereinstimmt. In der Praxis führt eine solche Formatierung zu Inkonsistenzen zwischen BIOS und DOS, was nach einiger Zeit zum vollständigen Datenverlust führt. Dieses Problem ist bei der Inbetriebnahme der CF-Card nicht direkt erkennbar, diese scheint sich völlig normal zu verhalten.

## 5.13. Power-Management

Durch die Powermanagement-Funktionen des BIOS kann der Stromverbrauch in vielen Fällen drastisch reduziert werden. Das gilt immer dann, wenn die volle CPU-Leistung bei 25 MHz Takt nicht ständig benötigt wird.

### 5.13.1. Ändern des CPU-Taktes

Der CPU-Takt kann per BIOS-Funktion auf 20 MHz, 8 MHz oder 4 MHz herabgesetzt werden. Die Teilerwerte für den Timer 0 werden vom BIOS angepasst.

Man beachte, daß das Umschalten der Taktfrequenz nicht schlagartig erfolgt. Zum Umschalten vom niedrigsten auf den maximalen Takt (4 MHz auf 25 MHz) benötigt der Taktgenerator etwa 4 ms. Hierdurch wird auch die Genauigkeit des Timer 0 beeinflusst. Dessen Teilerwerte (Vorteiler und Timer-Register) werden durch die BIOS-Funktion sofort neu geladen. Auch der aktuelle Zählerstand wird von der BIOS-Funktion nicht beachtet.

### 5.13.2. Idle-Mode

Ein Anwendungsprogramm kann immer dann, wenn keine Aktivität stattfindet, in den Idle- oder Powerdown-Modus umschalten. Erst beim Auftreten eines Interrupts setzt die CPU die Ausführung des Programms fort.

Im Idle-Mode wird nur der Takt für den CPU-Kern (CPU-Core) prozessorintern abgeschaltet, während der Takt für die seriellen Schnittstellen und die Timer weiterläuft.

Der normale Betriebszustand wird durch jeden nicht ausmaskierten Hardware-Interrupt wiederhergestellt (s. auch das Kapitel über die Interrupt-Controller). Man beachte, daß der Timer 0 normalerweise alle 55ms einen IRQ erzeugt und dadurch automatisch den Idle-Mode beendet. Das Anwendungsprogramm muß dafür sorgen, daß nach jedem IRQ der Idle-Mode nach Bedarf wiederhergestellt wird.

Die BIOS-Funktion für den Idle-Mode sieht das gleichzeitige Heruntertakten des Prozessors vor. Der Takt im Idle-Mode wird der Funktion als Parameter übergeben. Hierbei sind wiederum die Umschaltzeiten zu beachten (s.o.).

### 5.13.3. Powerdown-Mode

Der Takt für CPU-Kern und interne Peripherie des 386EX wird gestoppt. Die Timer laufen nur weiter, falls sie mit externem Takt betrieben werden. Die SSIO funktioniert nur noch im Slave-Mode. Da die UARTs einen eigenen Takt besitzen, funktionieren sie auch im Powerdown-Mode (jedoch nicht im Deep Powerdown-Mode).

Wie beim Idle-Mode sieht die BIOS-Funktion für den Powerdown-Mode das gleichzeitige Heruntertakten des Prozessors vor. Der Takt wird der Funktion als Parameter übergeben. Hierbei sind wiederum die Umschaltzeiten zu beachten (s.o.). Für den Deep Powerdown-Mode gibt es eine eigene BIOS-Funktion.

Die Rückkehr aus dem Powerdown-Mode geschieht durch einen Interrupt der RTC, der seriellen Schnittstellen, der Timer oder durch externe IRQs. Diese IRQs dürfen dazu nicht ausmaskiert sein. Die Interrupts der Timer können den Powerdown-Mode nur dann beenden, wenn sie mit externem Takt (Timer 0 und 1) bzw. mit COMCLK als Takt (Timer 2) betrieben werden, da das normale Takt-Signal für die Timer (PSCLK) im Powerdown-Mode abgeschaltet ist.

### 5.13.4. Deep Powerdown-Mode

Im Deep Powerdown-Mode wird der Oszillator-Chip abgeschaltet. Die Stromaufnahme reduziert sich dadurch auf weniger als 1 mA.

Die Zeit zum Wiederhochfahren des Taktes auf 25 MHz beträgt etwa 9ms.

Die Rückkehr aus dem Powerdown-Mode kann nur durch einen IRQ8 erfolgen. Dies ist normalerweise der IRQ der RTC. Das auf den Steckverbinder geführte Signal –IRQ8/TCLK0 ist durch den Open-Drain-Ausgang der RTC belegt. Externe Open-Drain-Signalquellen können hier angeschlossen werden und ebenfalls einen IRQ8 erzeugen.

# 6. PC-Programme

## 6.1. VTERM

VTERM.EXE ist das Standard-Terminalprogramm für den MicroPC und damit die unentbehrliche Verbindung zur MicroPC während der Software-Entwicklung.

### 6.1.1. Kommandozeilen-Parameter

VTERM lässt sich mit folgenden Kommandozeilen-Parametern aufrufen:

-? : Kommandozeilen-Parameter Übersicht
-b(baud): Übertragungsrate einstellen
-c(1-4) : Seriellen Port auswählen
-m : Schwarz/Weiß Darstellung wählen
-o : Logdatei öffnen
-t(AHT) : Terminalemulation wählen

### 6.1.2. VTERM-Kommandos

Folgende Tasten sind mit Funktionen belegt :

ALT-B : Übertragungsrate einstellen
ALT-C : Seriellen Port wählen
ALT-D : Remote-Laufwerke zuweisen
ALT-E : Lokales ECHO ein/ausschalten
ALT-F : Handshake einstellen
ALT-H : Hilfe aufrufen
ALT-O : Logdatei öffnen/schließen
ALT-P : Übertragungsparameter einstellen
ALT-R : Datei empfangen
ALT-S : Datei senden
ALT-T : Terminal Emulation einstellen
ALT-W : Einstellungen sichern
ALT-X : VTERM verlassen
ALT-Y : Bildschirm löschen
ALT-Z : DOS Kommando

Zusätzlich zu den üblichen Terminalfunktionen (Ausgabe auf dem Bildschirm, Eingabe über die Tastatur des Host-PC, sowie Datei-Transfer) erlaubt VTERM den direkten Zugriff durch den MicroPC auf die Laufwerke des PCs mit Hilfe des TSR-Programms RDRIVE.

### 6.1.3. Datei-Transfer mit VTERM

Abgesehen von der Dateiübertragung via RDRIVE, die automatisiert abläuft, muß VTERM zur Unterscheidung von der normalen Bildschirm-Tastatur-Ein/Ausgabe mit besonderen Kommandos auf Datei-Empfang bzw. Datei-Senden eingestellt werden.

Dies betrifft insbesondere die Datei-Transfers des BIOS-Setup (Flash-Update/Backup) und die Kommunikation mit XSEND und XLOAD. Man startet zunächst den Transfer auf dem MicroPC und stellt anschließend mit ALT-R bzw. ALT-S VTERM auf Empfang oder Senden.

Die Übertragung auf der Seite des MicroPC findet im allgemeinen per Xmodem-Protokoll statt, daher muß VTERM ebenfalls auf Xmodem eingestellt werden. Danach wird der Name der Datei eingegeben, die gesendet werden bzw. der Name, unter welchem die zu empfangende Datei abgespeichert werden soll (Xmodem überträgt den Dateinamen nicht).

## 6.2. FLASHHDD

Flashhdd.exe erlaubt die Erstellung einer Flash-Image-Datei basierend auf dem Inhalt eines beliebigen Verzeichnisses. Diese wird per BIOS-Setup als Laufwerk C: in den Flash-Speicher des MicroPC übertragen.

Um DOS booten zu können, müssen sich je nach DOS-Version die folgenden Dateien in dem betreffenden Verzeichnis befinden:

| | |
|---|---|
| Datalight ROM-DOS: | COMMAND.COM |
| MS-DOS: | IO.SYS, MSDOS.SYS, COMMAND.COM |
| FreeDOS: | KERNEL.SYS, COMMAND.COM |

Aufruf:

FLASHHDD [/B<n>] [/S<m>] [/ V ][/?] <*Quellverzeichnis> [<Zieldatei>]*

Optionen:

| | |
|---|---|
| /B<n> | n = Zahl der Blöcke (default 14). |
| /S<m> | m = Blockgröße in kB (64 beim MicroPC). |
| / V | Verbose (Dateien und Verzeichnisse anzeigen) |
| /? | Hilfe anzeigen |
| /?? | Hilfe in Englisch (ab Version 1.32) |
| /M | MS-DOS kompatibles Format der Flashdisk (MS-DOS Systemdateien hinzufügen) |
| /F | FreeDOS kompatibles Format der Flashdisk (ab Version 1.32) |

Der maximale Wert für n hängt von der Bestückung gemäß der folgenden Tabelle ab:

| Gesamtkapazität | n |
|---|---|
| 1 MB | 14 |
| 2 MB | 29 |
| 4 MB | 61 |
| 8 MB | 125 |

Ein zu großer Wert für n sollte vermieden werden, da das DOS die Kapazität aus dem Bootsektor der Flashdisk berechnet. Falls dabei mehr Kapazität berechnet wird als physikalisch vorhanden ist, kommt es zum BIOS-Fehler beim Versuch, auf nicht vorhandene Sektoren zuzugreifen.

Man achte darauf, daß die Zieldatei nicht versehentlich im Quellverzeichnis erstellt wird (sonst Fehler durch Rekursion).

## 6.3. ROMDRV

Romdrv.exe erlaubt die Erstellung einer ROMDISK-Image Datei basierend auf dem Inhalt eines beliebigen Verzeichnisses. Diese wird per BIOS-Setup als Laufwerk A: in den Flash-Speicher des MicroPC übertragen.

Aufruf:

***romdrv <Qellverzeichnis> [<Zieldatei>]***

Man achte darauf, daß die Zieldatei nicht versehentlich im Quellverzeichnis erstellt wird (sonst Fehler durch Rekursion).

## 6.4. Bin2hex

BIN2HEX ist ein Programm für das Erstellen einer Intel-Hex86-Datei aus einer Binärdatei (z.B. ".COM"-File).

## 6.5. Hex2bin

HEX2BIN erzeugt eine Binär-Datei aus einer Intel-Hex86-Datei.

# 7. MicroPC Programme

## 7.1. Einbinden von Remote-Laufwerken mit RDRIVE, RMAP und RMCWD

Dieses Programm ermöglicht das Einbinden der PC-Laufwerke als Laufwerke des MicroPC. Das Programm wird beim Aufruf resident geladen. Danach können Dateien vom und zum Host-PC wie in einem Netzwerk z.B. per copy-Befehl übertragen werden. Dies ist für den MicroPC das Standardverfahren für die Übertragung von Dateien. Außerdem können Programme direkt vom Host-Laufwerk aus auf dem MicroPC gestartet werden, ohne sie erst auf ein lokales Laufwerk zu kopieren.

Aufruf:

rdrive [-?] [-c<n>] [-u]

-? : Hilfe

-c<1..4> : wähle COM-Port 1 bis 4 (default: COM1)
-u : entfernt RDRIVE aus dem RAM-Speicher

Zum Ändern der seriellen Schnittstelle mit -c muß RDRIVE zuvor mit -u entladen werden.

Mit dem Programm RMAP können nun Laufwerke und Verzeichnisse des Host-PC auf Laufwerke bzw. Laufwerksbuchstaben des MicroPC abgebildet werden. Dabei bezeichnet LOCAL den auf dem MicroPC zu verwendenden Laufwerksbuchstaben und REMOTE das Laufwerk oder Verzeichnis des Host-PC.

RMAP /LOCAL=D /REMOTE=C

stellt beispielsweise das Laufwerk C: des Host-PC als Laufwerk D: des MicroPC zur Verfügung. Es gibt keine feste Regel, welche Buchstaben in welcher Reihenfolge benutzt werden müssen – unabhängig davon, ob es sich bei den Remote-Laufwerken um lokale Laufwerke des Host-PC oder um Netzlaufwerke handelt.

Das Abbilden von Verzeichnissen ist genauso einfach:

RMAP /LOCAL=E /REMOTE=C:\Programs\Files386

Die Zuweisungen können jederzeit überschrieben oder mit Angabe des lokalen Laufwerks-Buchstaben gelöscht werden:

RMAP /LOCAL=D

Der Befehl RMAP allein gibt ein Liste der aktuellen Zuweisungen aus.

Das Programm RMCWD.EXE weist automatisch das Verzeichnis, aus dem heraus Vterm gestartet wurde, dem angegebenen LW-Buchstaben zu.

## 7.2. XLOAD

Xload ist ein einfaches Programm, um Dateien vom Host in die RAM- oder Flash-Disk zu übertragen. Hierzu wird das Übertragungsprotokoll XMODEM eingesetzt. Nach dem Aufruf mit

**xload** [com port ] file

wartet Xload, bis auf der Hostseite die Übertragung gestartet wird.

## 7.3. XSEND

Xsend ist ein einfaches Programm, um Dateien vom MicroPC zum Host zu übertragen. Hierzu wird das Übertragungsprotokoll XMODEM eingesetzt. Nach dem Aufruf mit

**xsend** [com port ] file

muß auf der Hostseite der Empfang gestartet werden.

## 7.4. ZTRANS

Ztrans bietet gegenüber Xsend und Xload erweiterte Funktionalität, insbesondere die Übertragung des Dateinamens sowie die Übertragung mehrerer Dateien mit einem Befehl. Das zugrundeliegende Protokoll ist ZMODEM. Dieses wird von VTERM nicht unterstützt. Stattdessen muß ein ZMODEM-fähiges Terminalprogramm eingesetzt werden (z.B. Windows Hyperterminal), oder auf dem Host muß ebenfalls Ztrans gestartet werden.

Aufruf:

***ztrans*** *[/R] [/Bn] [/Cn] [/?] <Datei(en)>*

Optionen

| | |
|---|---|
| /R | Empfangen statt Senden |
| /Bn | Baudrate einstellen |
| /Cn | Schnittstelle wählen |
| /? | Hilfe anzeigen |

Für *<Datei(en)>* sind Wildcards möglich.

# 8. BIOS-Setup

Das Setup des MicroPC bietet diverse Einstellungsmöglichkeiten, um ihn an die Bedürfnisse der Anwendung anzupassen. Um in das Setup zu gelangen, muß nach einem Neustart der Baugruppe am angeschlossenen Terminalprogramm während des Speichertests die Taste **<S>** gedrückt werden. Daraufhin wird der Speichertest abgebrochen und es erscheint das Setup-Hauptmenu. Hier kann mit den Zifferntasten oder den Cursortasten ein Menupunkt ausgewählt werden.

## 8.1. Main Setup

**Date und Time:** Einstellen der Echtzeituhr (RTC). Diese Werte bleiben nach dem Abschalten nur dann erhalten, wenn eine Lithium-Batterie bestückt ist oder eine andere Spannungsquelle an den Vbatt-Anschluß des I/O-Steckers angeschlossen ist.

**Console Port:** Hier wird diejenige serielle Schnittstelle eingetragen, über die das BIOS-Setup sowie das ROM-DOS (Gerät "CON") angesprochen werden sollen. **Achtung:** wird hier "none" oder eine nicht vorhandene Schnittstelle eingestellt, so kann beim nächsten Start nicht mehr auf das Setup und den DOS-Prompt zugegriffen werden. Dies ist zuweilen sinnvoll, um unberechtigte Zugriffe auf den MicroPC zu verhindern. Ein Rücksetzen des Console-Ports ist aber immer noch durch Kurzschließen des Ready-Pins nach GND während des Starts möglich. Das BIOS stellt dann automatisch Default-Werte für die erste serielle Schnittstelle ein und springt ins Setup.

**Real Mode Flash:** Hier wird eingestellt, wieviel vom Flash-Speicher im untersten Megabyte des CPU-Adressraum eingeblendet werden soll. Dieser Speicherbereich ist dann im Real Mode der 386EX-CPU zugänglich. Möglich sind 128 kB (default), 256 kB oder 512 kB. Der zugehörige Adressbereich befindet sich am Ende des untersten Megabytes. 128kB sind immer eingeblendet, da sich hier das Betriebssystem (BIOS und ROM-DOS) befindet.

**Enable ROM-Disk:** Der hier eingestellt Bereich des Flashspeichers kann für eine nicht schreibfähige ROM-Disk verwendet werden oder auch als linear adressierbarer Flashspeicher.

**Enable RAM-Disk:** Die RAM-Disk nutzt den freien RAM-Bereich oberhalb von 1MB, soweit dieser nicht schon unterhalb 1MB eingeblendet ist. Bei 1MB-RAM Bestückung sind mindestens 128 kB für die RAM-Disk verfügbar (entsprechend der Default-Größe des Real Mode Flash). Dieser Bereich kann vergrößert werden, indem mehr Real Mode Flash eingestellt wird.

**CompactFlash 1 und 2:** An den MicroPC können bis zu zwei CompactFlash™ Memory Cards oder Microdrive™ Festplatten angeschlossen werden. Diese können im LBA-Mode oder im Auto-CHS-Mode betrieben werden. Der bei der Formatierung einer konkreten CF Card eingestellte Mode muß später immer verwendet werden, also auch, wenn die CF-Card in anderen Rechnern eingesetzt wird. Der Mode kann nur im Zuge einer Neuformatierung geändert werden.

**CPU Clock (CLK2) in MHz:** Möglich sind Taktraten von 50, 40, 16 und 8 MHz.

## 8.2. Advanced Setup

**Power on messages:** Bei "disabled" werden beim Booten die Copyright-Meldungen, die Meldungen des RAM-Tests und die BIOS-Config-Box unterdrückt.

**System Configuration Box:** Die Anzeige der BIOS-Config-Box kann hiermit unterdrückt werden.

**Display "Hit <S>..." :** Die Anzeige der Meldung "Hit <S> ..." kann hiermit unterdrückt werden.

**Wait For Key on Error :** Hat beim MicroPC keine Bedeutung.

**Fast Boot:** Das BIOS führt nur einen abgekürzten RAM-Test durch (spart Zeit beim Booten).

**ROM-DOS :** aktiviert bzw. deaktiviert das ROM-DOS.

**ROM-DOS Bootdrive :** Legt das Laufwerk fest, von dem ROM-DOS die Dateien command.com, config.sys und autoexec.bat liest.

**Flash File System :** schaltet das Flash File System des BIOS für die On-Board Flash-Disk ab.

**DOS/Non-DOS Flashdisk:** durch "FAT Monitoring" durch das BIOS wird eine erhebliche Geschwindigkeitssteigerung des Flash File Systems der On-Board Flash-Disk erzielt. Dies funktioniert jedoch nur unter DOS. Bei Einsatz eines anderen Betriebssystems muß das FAT Monitoring abgeschaltet werden.

**SRAM und Flash Waitstates:** die Default-Einstellung sollte normalerweise nicht geändert werden. Falls jedoch nicht der maximale CPU-Takt verwendet wird, so kann auch ein anderer Wert eingestellt werden. Dieser errechnet sich wie folgt:

$n \geq 0$

und $n \geq (t_R + 10ns) * f_{CPU} - 1,5$

Dabei bedeutet:

$t_R$ = Zugriffszeit des RAMs / Flashs (nach Datenblatt), n = Zahl der Waitstates

$f_{CPU}$ = Oszillatorfrequenz / 2 (≤ 25 MHz)

Für $t_R$ = 55 ns und $f_{CPU}$ = 25 MHz gilt

$(t_R + 10ns) * f_{CPU} - 1,5 = 0,125$

Der nächstgrößere ganzzahlige Wert für n = 0,125 ist 1. Das 55ns RAM funktioniert also bei 1 Waitstate.

Für $t_R$ = 70 ns und $f_{CPU}$ = 25 MHz gilt

$(t_R + 10ns )* f_{CPU} - 1,5 = 0,5$

Für 70 ns RAM muß daher ebenfalls ein Waitstate eingestellt werden.

Für $t_R$ = 120 ns und $f_{CPU}$ = 25 MHz gilt

$(t_R +10ns) * f_{CPU} - 1,5 = 1,75$

Für 120 ns Flash müssen daher zwei Waitstates eingestellt werden.

Für $t_R$ = 90 ns und $f_{CPU}$ = 25 MHz gilt

$(t_R +10ns) * f_{CPU} - 1,5 = 1,0$

Für 90 ns Flash muß daher ein Waitstate eingestellt werden.

## 8.3. I/O Configuration Setup

**Einstellung der seriellen Schnittstellen:**

**Achtung:** Durch Veränderung der Einstellungen für die seriellen Schnittstellen kann der Zugriff auf den MicroPC vollständig blockiert werden (auch versehentlich). Man sollte also genau wissen, welche Einstellung man zu welchem Zweck verändert.

Das BIOS und das DOS unterstützen maximal vier serielle Schnittstellen. Diese können vom Anwenderprogramm mit Hilfe des BIOS über den Int 14h angesprochen werden oder mit Hilfe des DOS als Geräte (devices) COM1 bis COM4. Weitere serielle Schnittstellen werden von BIOS und DOS nicht unterstützt, diese müssen durch direkten Zugriff auf die Hardware programmiert werden.

**COM PORTS:** Mit TYPE wird der jeweilige UART-Typ für die vier möglichen COM Ports eingestellt. BASE gibt die Basis-Adresse des UARTs an. Mit BAUDRATE wird die Baudrate der Schnittstelle eingestellt. Mit SETTING werden Zahl der Daten-Bits, Parität und Zahl der Stop-Bits eingestellt. Unter INTERRUPT kann angegeben werden, ob die Schnittstelle im Polling-Betrieb oder per Empfangs-Interrupt arbeiten soll. In diesem Fall muß die richtige Interrupt-Leitung eingestellt werden. Verwendet man den Interrupt-Betrieb, kann unter BUFFER die Größe des vom BIOS-Interrupt-Handler verwendeten Empfangspuffers in Byte angegeben werden.

**PRINTER PORTS:** Hiermit wird die Basis-Adresse der optionalen Drucker-Schnittstellen eingestellt.

**X1 Connector Configuration:** Festlegung der Funktion einzelner Pins des I/O-Steckers X1. Diese können als Digital I/O (Input, Open-Drain Oputput oder Output), als Timerfunktionen oder als Interrupts konfiguriert werden.

**PORT INIT :** Hier besteht die Möglichkeit, schaltungsabhängige Initialisierungen an I/O-Ports (also auch an am PIF-Bus angschlossenen Peripherie-Einheiten) sehr früh im Bootvorgang durchzuführen. Man kann maximal vier I/O-Adressen mit ihren Initialisierungswerten angegeben werden. Diese I/O Zugriffe werden wenige Mikrosekunden nach dem Reset durchgeführt.

## 8.4. Flash Setup

**Flash Update Setup**

Im Flash Update Setup können verschiedene Bereiche des Flash-Speichers auf dem MicroPC neu programmiert werden. Dazu wird der aktuelle Inhalt gelöscht und mit den über die serielle Schnittstelle geladenen Daten neu programmiert. Das Laden neuer Daten erfolgt mit dem Übertragungsprotokoll XMODEM. Nach Anwählen eines Menüpunktes beginnt die Karte ein Protokollzeichen ( § ) zu senden. Hierauf muß auf dem Terminal Programm die Übertragung mit XMODEM gestartet werden (bei VTERM: ALT-S, XMODEM, Dateiname). Nach erfolgreichem Download wird die Karte je nach gewähltem Bereich neu gebootet, oder man gelangt zurück in das Setup-Menü.

**Flash Backup Setup**

Hier können einzelne Bereiche des Flash zum PC gesendet werden. Nach Anwählen eines Menüpunktes muß auf dem Terminal-Programm der Empfang mit XMODEM gestartet werden (bei VTERM: ALT-R, XMODEM, Dateiname).

**Flash Erase Setup**

Hier können einzelne Bereiche des Flash auf dem MicroPC selektiv gelöscht werden. Das Löschen erfolgt sektorweise (ein Flash-Sektor enthält 64kB). Das Löschen eines Sektors dauert normalerweise weniger als eine Sekunde.

## 8.5. Exit Setup

**Exit and Save changes:** Setup beenden und Änderungen speichern

**Exit and discard changes:** Setup beenden, ohne Änderungen zu speichern

**Reset to previous values:** Alle seit dem Start des BIOS-Setups durchgeführten Änderungen werden verworfen.

**Reset to default values:** Alle Einstellungen des BIOS-Setups werden auf Standard-Werte gesetzt (Default-Einstellungen).

# 9. BIOS - Referenz

## 9.1. INT 10h - Video Service

### 9.1.1. INT 10h Funktion 00h - Set Video Mode

**Aufruf:** AH = 00h
AL = Video Modus

**Rückgabe:** keine

**Beschreibung:** Da der MicroPC keine Video-Hardware besitzt, dient diese Funktion nur zum Löschen des Bildschirms.

### 9.1.2. INT 10h Funktion 02h - Set cursor Position

**Aufruf:** AH = 02h
DH = Zeile
DL = Spalte

**Rückgabe:** keine

**Beschreibung:** Hiermit kann die Cursor Position verändert werden.

### 9.1.3. INT 10h Funktion 03h - Get current cursor position

**Aufruf:** AH = 03h

**Rückgabe:** AX = 00h
DH = Zeile
DL = Spalte

**Beschreibung:** Hiermit kann die aktuelle Cursor Position erfragt werden.

### 9.1.4. INT 10h Funktion 06h/07h - Scroll current page up/down

**Aufruf:** AH = 05h/06h
BH = Neues Farbattribut

**Rückgabe:** keine

**Beschreibung:** Da der MicroPC keine Video-Hardware besitzt, dienen diese Funktionen nur zum Löschen des Bildschirms mit dem angegebenen Farbattribut.

### 9.1.5. INT 10h Funktion 09h - Write Char/Attribute to Screen

**Aufruf:** AH = 09h
AL = Zeichen
BL = Farbattribut
CX = Anzahl der Zeichen

**Rückgabe:** keine

**Beschreibung:** Das Zeichen wird CX mal mit dem angegebenen Farbattribut ausgegeben. Die Cursor Position wird nicht verändert. Anders als auf einem PC gilt das Farbattribut für alle folgenden Ausgaben mit den Funktionen 00h, 0Ah und 0Eh.

### 9.1.6. INT 10h Funktion 0Ah - Write character to screen

**Aufruf:** AH = 0Ah
AL = Zeichen
CX = Anzahl der Zeichen

**Rückgabe:** keine

**Beschreibung:** Das Zeichen wird CX mal ausgegeben. Die Cursor Position wird nicht verändert.

#### 9.1.7. INT 10h Funktion 0Eh - Write Teletype to screen

**Aufruf:** AH = 0Eh
AL = Zeichen

**Rückgabe:** keine

**Beschreibung:** Das Zeichen in AL wird ausgegeben, wobei die Steuerzeichen 07h (Beep), 08h (Backspace), 0Ah (Linefeed) und 0Dh (Carriage Return) interpretiert werden. Dies ist die schnellste Ausgabemöglichkeit, da keine Escape Sequenzen gesendet werden müssen.

### 9.2. INT 11h - Equipment Check Service

**Aufruf:** keine

**Rückgabe:** AX = Inhalt von 40:10
Bits 15 -14 = Anzahl der Drucker
Bits 13 - 12 = Reserviert
Bits 11 - 9 = Anzahl Disketten
Bit 8 = Reserviert
Bits 5 - 4 = Video Modus
Bit 3 = Reserviert
Bit 2 = Maus installiert
Bit 1 = Coprozessor
Bit 0 = Boot Disk vorhanden

**Beschreibung:** Diese Funktion gibt den Inhalt der Speicherzelle 40:10h zurück.

### 9.3. INT 12h - Memory size

**Aufruf:** keine

**Rückgabe:** AX = Inhalt von 40:13h

**Beschreibung:** Diese Funktion gibt den Inhalt der Speicherzelle 40:13h zurück. Dies gibt den freien Speicher in Kilobytes an.

### 9.4. INT 13h - Disk Services

Da die Flash-Disk des MicroPC wie ein Festplatten-Laufwerk organisiert ist, gilt das folgende auch für sie.

#### 9.4.1. INT 13h Funktion 01h - Read Disk Status

**Aufruf:** AH = 01h
DL = Laufwerk (0 oder 1)

**Rückgabe:** AH = 0 Kein Fehler
= sonst Fehler Code
CF = 0 Kein Fehler
= 1 Fehler

**Beschreibung:** Liest den letzten Fehlercode aus und setzt ihn wieder zurück.

#### 9.4.2. INT 13h Funktion 02h - Read Disk Sectors

**Aufruf:** AH = 02h
AL = Anzahl der Sektoren
CH = Track
CL = Sector
DH = Kopf
DL = Laufwerk (0 oder 1)
ES:BX = Zeiger zum Sektorpuffer

**Rückgabe:** AH = 0 Kein Fehler
= sonst Fehler Code

| | | |
|---|---|---|
| | AL | = Anzahl der gelesenen Sektoren |
| | CF | = 0 Kein Fehler |
| | | = 1 Fehler |

**Beschreibung:** Diese Funktion liest die angegebene Anzahl der Sektoren in einen Puffer ein.

#### 9.4.3. INT 13h Funktion 03h - Write Disk Sectors

| | | |
|---|---|---|
| **Aufruf:** | AH | = 03h |
| | AL | = Anzahl der Sektoren |
| | CH | = Track |
| | CL | = Sector |
| | DH | = Kopf |
| | DL | = Laufwerk (0 oder 1) |
| | ES:BX | = Zeiger zum Sektorpuffer |
| **Rückgabe:** | AH | = 0 Kein Fehler |
| | | = sonst Fehler Code |
| | AL | = Anzahl der gelesenen Sektoren |
| | CF | = 0 Kein Fehler |
| | | = 1 Fehler |

**Beschreibung:** Diese Funktion schreibt die angegebene Anzahl der Sektoren auf das Laufwerk.

#### 9.4.4. INT 13h Funktion 08h - Read Drive Parameter

| | | |
|---|---|---|
| **Aufruf:** | AH | = 08h |
| | DL | = Laufwerk (0 oder 1) |
| **Rückgabe:** | AX | = 0 |
| | CH | = Letzter Track |
| | CL | = Letzter Sektor |
| | DH | = Anzahl der Köpfe |
| | DL | = Anzahl der installierten Laufwerke |
| | ES:DI | = Pointer auf Diskette Parameter Table |
| | CF | = 0 Kein Fehler |
| | | = 1 Fehler |

**Beschreibung:** Diese Funktion liefert die Parameter eines Laufwerkes.

## 9.5. INT 14h – Funktionen der asynchronen seriellen Schnittstellen

### 9.5.1. INT 14h Funktion 00h – Serielle Schnittstelle initialisieren

**Aufruf:** AH = 00h
AL = Parameter
Bits 7 - 5 = Baudrate:
000 = 110 Baud
001 = 150 Baud
010 = 300 Baud
011 = 600 Baud
100 = 1200 Baud
101 = 2400 Baud
110 = 4800 Baud
111 = 9600 Baud
Bits 4 - 3 = Parity
x0 = keine
01 = ungerade
11 = gerade
Bit 2 = Stop Bits
0 = 1 Stop Bit
1 = 2 Stop Bits
Bit 1 - 0 = Datenwortlänge:
00 = 5 Bits
01 = 6 Bits
10 = 7 Bits
11 = 8 Bits
DX = Nr. der seriellen Schnittstelle (0 - 3)

**Rückgabe:** AH = Line Status wie bei Funktion 1
AL = Modem Status

### 9.5.2. INT 14h Funktion 01h – Zeichen senden

**Aufruf:** AH = 01h
AL = Zu sendendes Zeichen
DX = Nr. der seriellen Schnittstelle (0 - 3)

**Rückgabe:** AL = gesendetes Zeichen
AH = Line Status:
Bit 7 = 1 : FIFO Error bei 16C550 kompatiblen externen UARTs; für die UARTs des 386EX-Prozessors ist Bit7 = 0
Bit 6 = 1 : Transmitter shift register empty
Bit 5 = 1 : Transmitter buffer register empty
Bit 4 = 1 : Break condition = RXD low für die Dauer von mehr als einer Wortlänge
Bit 3 = 1 : Framing error = ungültiges Stop-Bit
Bit 2 = 1 : Parity error
Bit 1 = 1 : Overrun error = Receive Buffer wurde überschrieben
Bit 0 = 1 : Receive data ready

### 9.5.3. INT 14h Funktion 02h – Zeichen empfangen

**Aufruf:** AH = 02h

**Rückgabe:** AL = Empfangenes Zeichen
AH = Line Status wie bei Funktion 1, jedoch:
Bit 7 = 1 : Timeout error
DX= Nr. der seriellen Schnittstelle (0 - 3)

**Bemerkung:** Ein Timeout entsteht nach ca. 1 Sekunde. Die seriellen Schnittstellen werden je nach Einstellung im BIOS-Setup (IRQn oder Polled) mit Interrupt oder im Polling-Modus betrieben.

### 9.5.4. INT 14h Funktion 03h – Status einer seriellen Schnittstelle abfragen

**Aufruf:** AH = 03h
DX = Nr. der seriellen Schnittstelle (0 - 3)

**Rückgabe:** AL = Inhalt des Modem Status Registers
AH = Line Status wie bei Funktion 2

**Bemerkung:** Bei Interrupt-Betrieb können Fehlerzustände (Line Status Bits 1 bis 4) nicht eindeutig bestimmten Bytes im Empfangspuffer zugeordnet werden. Das Overrun-Bit dient gleichzeitig zum Anzeigen eines Überlaufs des BIOS-Empfangspuffers.

### 9.5.5. INT 14h Funktion 04h - Extended Init

**Aufruf:** AH = 04h
BH = Parity
00h = no parity.
01h = odd parity.
02h = even parity.
BL - Stop bits
00h = 1 Stop Bit
01h = 2 Stop Bits oder 1,5 bei 5 Datenbits
CH - Data length
00h = 5 Bits
01h = 6 Bits
02h = 7 Bits
03h = 8 Bits
CL - Baudrate

| | |
|---|---|
| 00h = | 110 Baud |
| 01h = | 150 Baud |
| 02h = | 300 Baud |
| 03h = | 600 Baud |
| 04h = | 1200 Baud |
| 05h = | 2400 Baud |
| 06h = | 4800 Baud |
| 07h = | 9600 Baud |
| 08h = | 19200 Baud |
| 09h = | 38400 Baud |
| 0Ah = | 57600 Baud |
| 0Bh = | 115200 Baud |

DX = Nr. der seriellen Schnittstelle (0 - 3)

**Rückgabe:** AH = Line Status wie bei Funktion 1
AL = Modem Status

**Bemerkung:** Die Funktion erlaubt höhere Baudraten als Funktion 00h.

## 9.6. INT 15h - System Services

### 9.6.1. INT 15h Funktion 24h - A20 Gate-Control

### 9.6.2. INT 15h Funktion 87h - Move Memory Block

### 9.6.3. INT 15h Funktion C0h - Get System Config Table

**Aufruf:** AH = C0h

**Rückgabe:** AH = 00h
ES:BX = Adresse System Config Table

**Beschreibung:** Diese Funktion liefert die Adresse der System Configuration Table.

### 9.6.4. INT 15h Funktion A1h - Int 10h / Int 16h I/O umleiten

Die Funktion leitet die Video-Ausgabe des BIOS Int 10h und die Tastatur-Eingabe-Funktionen des Int 16h auf eine serielle Schnittstelle um. Der Parameter 0 (none) schaltet sämtliche Ein- und Ausgaben des Int 10h und Int 16h ab.

**Aufruf:** AH = A1h
BX = COM-Port (1 = COM1, ..., 4 = COM4, 0 = none)

## 9.7. INT 15h Funktion C3h - MicroPC spezifische Funktionen

### 9.7.1. INT 15h Funktion C301h - Watchdog freigeben

**Aufruf:** AH = C3h
AL = 01h
BX = Dauer des Watchdog-Timeout in Milli-Sekunden

**Rückgabe:** --

**Beschreibung:** Diese Funktion aktiviert den Watchdog. Anschließend muß der Watchdog mindestens einmal innerhalb der Watchdog-Timeout-Dauer zurückgesetzt werden, andernfalls wird ein System-Reset ausgelöst.

Einmal freigegeben, kann der Watchdog nur durch einen System-Reset deaktiviert werden. Auch die Timeout-Zeit läßt sich nachträglich nicht mehr ändern.

### 9.7.2. INT 15h Funktion C302h - Watchdog zurücksetzen

**Aufruf:** AH = C3h
AL = 02h

**Rückgabe:** --

**Beschreibung:** Nach Freigabe des Watchdogs muß diese Funktion mindestens einmal innerhalb der Watchdog-Timeout-Dauer aufgerufen werden, andernfalls wird ein System-Reset ausgelöst.

### 9.7.3. INT 15h Funktion C310h – Prozessortakt abfragen

Aufruf: AH = C3h
AL = 10h

**Rückgabe:** AL = 3 : CPU-Clock = 25 MHz (CLK2 = 50 MHz)
= 2 : CPU-Clock = 20 MHz (CLK2 = 40 MHz)
= 1 : CPU-Clock = 8 MHz (CLK2 = 16 MHz)
= 0 : CPU-Clock = 4 MHz (CLK2 = 8 MHz)

Ab Platinenrevision 2 sind auch die folgenden Taktraten möglich:

= 7 : CPU-Clock = 16,67 MHz (CLK2 = 33,33 MHz)
= 6 : CPU-Clock = 12,5 MHz (CLK2 = 25 MHz)
= 5 : CPU-Clock = 10 MHz (CLK2 = 20 MHz)
= 4 : CPU-Clock = 6,5 MHz (CLK2 = 13 MHz)

### 9.7.4. INT 15h Funktion C311h – Prozessortakt setzen

Aufruf: AH = C3h
AL = 11h
BL = Takt wie bei Funktion C310h

**Rückgabe:** --

**Bemerkung :** Der Vorteiler (CLKPRS) für den Eingangstakt von Timer 0 wird bei der Taktumschaltung angepasst.

### 9.7.5. INT 15h Funktion C312h - CPU in den IDLE Mode versetzen

Aufruf: AH = C3h
AL = 12h
BL = *Takt*

**Bemerkung :** CPU-Kern wird angehalten, die interne Peripherie des 386EX (insbesondere die Timer und die SSIO) läuft weiter. Die Rückkehr aus dem Idle-Modus ist nur durch einen (beliebigen) Hardware Interrupt möglich.

### 9.7.6. INT 15h Funktion C313h - CPU in den Powerdown-Mode versetzen

Aufruf: AH = C3h
AL = 13h
BL = Takt wie bei Funktion C310h, oder
BL = 80h: Deep Powerdown Mode

**Bemerkung:** CPU-Kern und interne Peripherie des 386EX wird gestoppt. Die Timer laufen nur weiter, falls sie mit externem Takt betrieben werden. Die SSIO funktioniert nur noch im Slave-Mode. Die UARTs arbeiten beim MicroPC mit einem eigenen Takt, funktionien also auch im Powerdown-Mode.

Rückkehr aus dem Powerdown-Mode durch Interrupt der RTC, der seriellen Schnittstellen oder durch externe IRQs. Die Interrupts von Timer 0 und Timer 1 beenden den Powerdown-Mode nur dann, wenn die Timer mit externem Takt betrieben werden. Da der Timer 2 beim MicroPC nur mit internem Takt betrieben werden kann, kann er die Rückkehr aus dem Powerdown-Mode nicht bewirken.

Im Deep Powerdown Mode wird der Takt komplett abgeschaltet. Man erzielt hierdurch die geringste Leistungsaufnahme. Die Rückkehr aus dem Deep Powerdown Mode kann nur durch den IRQ der RTC oder ein externes, ebenfalls an den RTC-IRQ angeschlossenes Signal bewerkstelligt werden.

### 9.7.7. INT 15h Funktion C314h – Synchrone Serielle Schnittstelle: Senden

Aufruf: AH = C3h
AL = 14h
BX = Datenwort
CL = 4..127: Teiler für Baudrate

**Bemerkung:** Die synchrone serielle Schnittstelle arbeitet mit dieser und der folgenden Funktion ausschließlich im Master-Mode ohne Interrupt. Die Baudrate ergibt sich aus dem CPU-Eingangstakt CLK2 (normalerweise 50 MHz beim MicroPC) nach der Formel:

Baudrate = CLK2 / (8 x (CL + 1))

Es sind also Baudraten zwischen 1,25 MBaud und 48,8 kBaud möglich. Ändern des CPU-Taktes bewirkt eine entsprechende Änderung der Baudrate.

### 9.7.8. INT 15h Funktion C315h – Synchrone Serielle Schnittstelle: Empfangen

Aufruf: AH = C3h
AL = 15h
CL = 4..127: Teiler für Baudrate

**Rückgabe:** AX = Datenwort

### 9.7.9. INT 15h Funktion C320h - EEPROM auslesen

**Aufruf:** AH = C3h
AL = 20h
BH = Adresse im EEPROM

**Rückgabe:** AL = Datenbyte

**Beschreibung:** Diese Funktion liest das Datenbyte im EEPROM an der übergebenen Adresse

### 9.7.10. INT 15h Funktion C321h - EEPROM beschreiben

**Aufruf:** AH = C3h
AL = 21h
BH = Adresse im EEPROM
BL = Datenbyte

**Beschreibung:** Diese Funktion schreibt das Datenbyte im EEPROM an der übergebenen Adresse

### 9.7.11. INT 15h Funktion C322h - I2C-Bus: Byte lesen mit Adreßbyte

**Aufruf:** AH = C3h
AL = 22h
BH = I2C-Bus Adresse
CH = I2C-Chip Adresse

**Rückgabe :** AL = gelesenes I2C-Bus Datenbyte
Carry-Flag = 0: kein Fehler
Carry-Flag = 1: Fehler

### 9.7.12. INT 15h Funktion C323h - I2C-Bus: Byte schreiben mit Adreßbyte

**Aufruf:** AH = C3h
AL = 23h
BH = I2C-Bus Adresse
BL = I2C-Bus Datenbyte
CH = I2C-Chip Adresse

**Rückgabe :** Carry-Flag = 0: kein Fehler
Carry-Flag = 1: Fehler

### 9.7.13. INT 15h Funktion C324h - I2C-Bus: Datenblock lesen mit zwei Adreßbytes

**Aufruf:** AH = C3h
AL = 24h
BX = I2C-Bus Adresse (BH = erstes Adreßbyte, BL = zweites Adreßbyte)
CH = I2C-Chip Adresse
CL = Anzahl zu lesender Bytes (0 == 256)
ES:DI = Far-Zeiger auf Lese-Puffer im RAM

**Rückgabe :** ES:[DI] = gelesene Datenbytes
Carry-Flag = 0: kein Fehler
Carry-Flag = 1: Fehler

### 9.7.14. INT 15h Funktion C325h - I2C-Bus: Datenblock schreiben mit zwei Adreßbytes

**Aufruf:** AH = C3h
AL = 25h
BX = I2C-Bus Adresse (BH = erstes Adreßbyte, BL = zweites Adreßbyte)
CH = I2C-Chip Adresse
CL = Anzahl zu schreibender Bytes (0 == 256)
ES:SI = Far-Zeiger auf Schreibpuffer

**Rückgabe :** Carry-Flag = 0: kein Fehler
Carry-Flag = 1: Fehler

### 9.7.15. INT 15h Funktion C326h - I2C-Bus anfordern

**Aufruf:** AH = C3h
AL = 26h

**Rückgabe :** Carry-Flag = 0: I2C-Bus erfolgreich angefordert
Carry-Flag = 1: I2C-Bus war besetzt

**Beschreibung:** Da I2C-Bus Zyklen sich beim MicroPC nicht überlappen dürfen, sollte der Bus vor jeder Übertragung angefordert werden. Das zugehörige I2C-Bus Flag wird dadurch gesetzt und verhindert, daß weitere I2C-Bus Funktionen die aktuelle Funktion unterbrechen. Dies ist nur von Bedeutung, wenn I2C-Funktionen innerhalb von Interrupt-Routinen ausgeführt werden sollen.

### 9.7.16. INT 15h Funktion C327h - I2C-Bus freigeben

**Aufruf:** AH = C3h
AL = 27h

**Beschreibung:** Das I2C-Bus Flag wird wieder zurückgesetzt. Falls der I2C-Bus zuvor mit der Funktion C326h angefordert wurde, muß er mit dieser Funktion zurückgesetzt werden. Andernfalls sind keine weiteren I2C-Bus Zugriffe mehr möglich.

### 9.7.17. INT 15h Funktion C328h - I2C-Bus: Byte lesen mit 2 Adreßbytes

**Aufruf:** AH = C3h
AL = 28h
BH = I2C-Bus Adresse 0
CH = I2C-Chip Adresse
CL = I2C-Bus Adresse 1

**Rückgabe :** AL = gelesenes I2C-Bus Datenbyte
Carry-Flag = 0: kein Fehler
Carry-Flag = 1: Fehler

### 9.7.18. INT 15h Funktion C329h - I2C-Bus: Byte schreiben mit 2 Adreßbytes

**Aufruf:** AH = C3h
AL = 29h
BH = I2C-Bus Adresse 0
BL = I2C-Bus Datenbyte
CH = I2C-Chip Adresse
CL = I2C-Bus Adresse 1

**Rückgabe :** Carry-Flag = 0: kein Fehler
Carry-Flag = 1: Fehler

### 9.7.19. INT 15h Funktion C32Ah - I2C-Bus: Byte lesen

**Aufruf:** AH = C3h
AL = 2Ah
CH = I2C-Chip Adresse

**Rückgabe :** AL = gelesenes I2C-Bus Datenbyte
Carry-Flag = 0: kein Fehler
Carry-Flag = 1: Fehler

### 9.7.20. INT 15h Funktion C32Bh - I2C-Bus: Byte schreiben

**Aufruf:** AH = C3h
AL = 2Bh
BL = Datenbyte
CH = I2C-Chip Adresse

**Rückgabe :** Carry-Flag = 0: kein Fehler
Carry-Flag = 1: Fehler

### 9.7.21. INT 15h Funktion C32Ch - I2C-Bus: zwei Bytes lesen

**Aufruf:** AH = C3h
AL = 2Ch
CH = I2C-Chip Adresse

**Rückgabe :** AX = gelesenes Datenwort (AH = erstes Byte, AL = zweites Byte)
Carry-Flag = 0: kein Fehler
Carry-Flag = 1: Fehler

### 9.7.22. INT 15h Funktion C32Dh - I2C-Bus: zwei Bytes schreiben

**Aufruf:** AH = C3h
AL = 2Dh
BH = erstes Datenbyte
BL = zweites Datenbyte
CH = I2C-Chip Adresse

**Rückgabe :** Carry-Flag = 0: kein Fehler
Carry-Flag = 1: Fehler

### 9.7.23. INT 15h Funktion C32Eh - I2C-Bus: Datenblock lesen

**Aufruf:** AH = C3h
AL = 2Eh
CH = I2C-Chip Adresse
CL = Anzahl der zu lesenden Bytes (0 == 256)
ES:DI = Far-Zeiger auf Lesepuffer im RAM

**Rückgabe :** [ES:DI] = gelesene Daten
Carry-Flag = 0: kein Fehler
Carry-Flag = 1: Fehler

### 9.7.24. INT 15h Funktion C32Fh - I2C-Bus: Datenblock schreiben

**Aufruf:** AH = C3h
AL = 2Fh
CH = I2C-Chip Adresse
CL = Anzahl der zu schreibenden Bytes (0 == 256)
ES:SI = Far-Zeiger auf Schreibpuffer

**Rückgabe :** Carry-Flag = 0: kein Fehler
Carry-Flag = 1: Fehler

### 9.7.25. INT 15h Funktion C330h – Hardware Serien-Nummer abfragen

**Aufruf:** AH = C3h
AL = 30h

**Rückgabe:** AX = unteres Datenwort der Serien-Nummer
BX = mittleres Datenwort der Serien-Nummer
CX = oberes Datenwort der Serien-Nummer

**Beschreibung:** Diese Funktion liest die sechs Bytes der Hardware-Serien-Nummer des MicroPC aus. Jedes Exemplar des MicroPC besitzt eine eigene, von allen anderen Exemplaren verschiedene Serien-Nummer.

## 9.8. INT 16h - Keyboard Service

### 9.8.1. INT 16h Funktion 00h - Read Keyboard Input

**Aufruf:** AH = 00h

**Rückgabe:** AH = Scancode erweiterte Tasten
AL = Tastenwert

**Beschreibung:** Diese Funktion liest eine Taste ein. Nur einige erweiterte Tasten werden unterstützt, da hierzu ANSI Escape Sequenzen verwendet werden (siehe Kap. 10).

### 9.8.2. INT 16h Funktion 01h - Read Keyboard Status

**Aufruf:** AH = 01h

**Rückgabe:** ZF = 1 - Kein Zeichen vorhanden
= 0 - Zeichen vorhanden

**Beschreibung:** Hiermit wird ermittelt, ob ein Zeichen im Tastaturpuffer vorliegt. Anders als beim PC wird das Zeichen nicht mit zurückgeliefert.

### 9.8.3. INT 16h Funktion 05h – Tastendruck simulieren

**Aufruf:** AH = 05h
CH = Scan-Code der Taste
CL = ASCII-Code der Taste

**Rückgabe:** AL = 0 – Funktion erfolgreich
= 1 – Tastaturpuffer war voll

**Beschreibung:** Mit dieser Funktion kann ein Anwendungsprogramm Werte in den BIOS-Tastatur-Puffer schreiben.

## 9.9. INT 17h - Parallel Service

### 9.9.1. INT 17h Funktion 00h - Print Character

**Aufruf:** AH = 00h
AL = Zeichen
DX = LPT Port (0 - 2)

**Rückgabe:** AH = Drucker Status
Bit 7 = 1 Drucker nicht besetzt
Bit 6 = 1 Acknowledgment
Bit 5 = 1 Out of Paper
Bit 4 = 1 Drucker selektiert
Bit 3 = 1 Drucker Fehler
Bit 2 - 1 = reserviert
Bit 0 = Timeout Fehler

**Beschreibung:** Mit dieser Funktion wird ein Zeichen ausgegeben.

### 9.9.2. INT 17h Funktion 01h - Initialize Printer

**Aufruf:** AH = 01h
DX = LPT Port (0 - 2)

**Rückgabe:** AH = Drucker Status
Bit 7 = 1 Drucker nicht besetzt
Bit 6 = 1 Acknowledgment
Bit 5 = 1 Out of Paper
Bit 4 = 1 Drucker selektiert
Bit 3 = 1 Drucker Fehler
Bit 2 - 1 = reserviert
Bit 0 = Timeout Fehler

**Beschreibung:** Hiermit wird der Drucker zurückgesetzt.

### 9.9.3. INT 17h Funktion 02h - Get Printer Status

**Aufruf:** AH = 02h
DX = LPT Port (0 - 2)

**Rückgabe:** AH = Drucker Status
Bit 7 = 1 Drucker nicht besetzt
Bit 6 = 1 Acknowledgment
Bit 5 = 1 Out of Paper
Bit 4 = 1 Drucker selektiert
Bit 3 = 1 Drucker Fehler
Bit 2 - 1 = reserviert
Bit 0 = Timeout Fehler

**Beschreibung:** Hiermit wird der Status des Druckers ausgelesen.

## 9.10. INT 18h - Boot Failure

**Beschreibung:** Diese Funktion wird nach erfolglosen Bootversuchen angesprungen.

## 9.11. INT 19h - Boot System

**Beschreibung:** Diese Funktion wird nach vollständiger Initialisierung des BIOS angesprungen. Sie wird vom ROM-DOS auf eine eigene Routine gesetzt. Ist kein ROM-DOS vorhanden, oder ist dies deaktiviert, versucht der BIOS-Default-Handler, das Betriebssystem von der Flash- oder ROM-Disk zu laden. Misslingt dies, wird ein INT 18h ausgeführt.

## 9.12. INT 1Ah - Uhren- und Timer-Funktionen

### 9.12.1. INT 1Ah Funktion 00h - Read System timer

**Aufruf:** AH = 00h

**Rückgabe:**
AH = 00h
AL = 24h Überlauf Flag
CX:DX = System Ticks seit Mitternacht

**Beschreibung:** Diese Funktion liest den System Timer aus. Dieser wird 18,2 mal in der Sekunde erhöht.

### 9.12.2. INT 1Ah Funktion 01h - Set System Timer

**Aufruf:**
AH = 01h
CX:DX = System Ticks seit Mitternacht

**Rückgabe:** AH = 00h

**Beschreibung:** Diese Funktion setzt den System Timer. Dieser wird vom Timer 0-Interrupt 18,2 mal in der Sekunde erhöht.

### 9.12.3. INT 1Ah Funktion 02h - Read Real Time Clock

**Aufruf:** AH = 02h

**Rückgabe:**
AH = 00h
AL = Stunden BCD
CH = Stunden in BCD
CL = Minuten in BCD
DH = Sekunden in BCD
CF = 0: OK
CF = 1: Fehler

**Beschreibung:** Diese Funktion liest die Uhrzeit der RTC aus.

### 9.12.4. INT 1Ah Funktion 03h - Set Real Time Clock

**Aufruf:**
AH = 03h
AL = Stunden in BCD
CH = Stunden in BCD
CL = Minuten in BCD
DH = Sekunden in BCD

**Rückgabe:**
AH = 00h
CF = 0: OK
CF = 1: Fehler

**Beschreibung:** Diese Funktion setzt die Uhrzeit der RTC.

### 9.12.5. INT 1Ah Funktion 04h – Read RTC Date

**Aufruf:** AH =04h

**Rückgabe:**
CH =Jahrhundert (19 oder 20)
CL =Jahr
DH =Monat
DL =Tag

**Beschreibung:** Diese Funktion liest das Datum der RTC aus.

### 9.12.6. INT 1Ah Funktion 05h – Set RTC Date

Aufruf: AH =05h
CH =Jahrhundert (19 oder 20)
CL =Jahr
DH =Monat
DL =Tag

**Rückgabe:** CF = 0: OK
CF = 1: Fehler

**Beschreibung:** Diese Funktion setzt das Datum der RTC.

### 9.12.7. INT 1Ah Funktion 06h – Set / Enable RTC Interrupt

**Aufruf:** AH = 06h
CH = Stunde
CL = Minute
DH = Sekunde

**Rückgabe:** CF = 0: OK
CF = 1: Fehler (z.B.: es ist schon ein Interrupt programmiert)

**Beschreibung:** Die Echtzeituhr erzeugt zur programmierten Zeit desselben Tages einen Interrupt. Dies ist ein Impuls von etwa 10..40 ms Dauer. Das Anwenderprogramm kann eine Funktion in den Interrupt 4Ah einklinken, die darauf bei jedem RTC-Interrupt aufgerufen wird. Wird dieser Interrupt verwendet, um aus dem Powerdown-Mode herauszukommen, ist ein Einklinken in den Interrupt 4Ah nicht nötig. Alle Werte, die der Funktion übergeben werden, müssen BCD kodiert sein. Es ist immer nur eine einzige Interrupt-Zeit aktiv. Ist bereits ein Interrupt programmiert, muß dieser zuerst mit Hilfe der Funktion 07h gelöscht werden.

### 9.12.8. INT 1Ah Funktion 07h – Disable RTC Interrupt

**Aufruf:** AH = 07h

**Rückgabe:** CF = 0: OK
CF = 1: Fehler

**Beschreibung:** Mit Hilfe dieser Funktion kann eine einprogrammierte Interrupt-Zeit wieder gelöscht werden. Der RTC-Interrupt wird dann nicht mehr erzeugt.Diese Funktion muß auch immer dann aufgerufen werden, wenn die Interrupt-Zeit geändert werden soll. Erst nach Aufruf dieser Funktion kann mit der Funktion 06h eine neue Zeit programmiert werden.

### 9.12.9. INT 1Ah Funktion 08h : Synchronize System Timer

**Aufruf:** AH = 08h

**Rückgabe:** CF = 0: OK
CF = 1: Fehler

**Beschreibung:** Der System Timer wird mit dem Inhalt der Echtzeituhr synchronisiert.

### 9.12.10. INT 1Ah Funktion 09h : zyklischen Interrupt setzen

**Aufruf:** AH = 09h
AL = 0: Interrupt alle 1/4096 Sekunde
1: Interrupt pro Sekunde
2: Interrupt pro Minute
3: Interrupt pro Stunde
4: Interrupt pro Tag

**Rückgabe:** CF = 0: OK
CF = 1: Fehler

**Bemerkung:** Die Funktion ist hardwareabhängig, also nicht kompatibel zum normalen PC-BIOS.

## 9.13. INT 1Bh bis 1Fh

Diese Interruptvektoren zeigen nicht auf eine ausführbare Funktion, sondern auf verschiedene BIOS-Tabellen.

## 9.14. INT 5Fh - Flash Services

### 9.14.1. INT 5Fh Funktion 00h - Flash Erase Block

**Aufruf :** AH = 00h
DX:DI = 32-Bit Flash-Startadresse + 10000h * Block-Nr.

**Rückgabe :** Carry-Flag = 0: kein Fehler
Carry-Flag = 1: Fehler

**Beschreibung:** (Siehe auch Funktion 02h). Löschen eines Flash-Blocks von 64 kB. Die Blöcke, welche das BIOS und das ROM-DOS enthalten, können mit dieser Funktion nicht gelöscht werden.

### 9.14.2. INT 5Fh Funktion 01h - Flash Read Block

**Aufruf:** AH = 01h
DX:DI = 32-Bit Quell-Adresse (erstes zu lesendes Byte)
= Flash-Startadresse + 10000h*Block-Nr.+ Offset
ES:BX = Ziel-Adresse (Far-Pointer auf Datenpuffer im RAM)
CX = Zahl der zu lesenden Bytes

**Rückgabe :** [ES:BX] = gelesene Daten
Carry-Flag = 0: kein Fehler
Carry-Flag = 1: Fehler

**Beschreibung:** "Offset" gibt die Startadresse relativ zum Blockanfang an. Die Quell-Adresse ist eine 32-Bit-Adresse, da der Flash-Speicher im Protected-Mode angesprochen wird. Die Ziel-Adresse (im RAM) ist eine Real-Mode Adresse, also in der Form *Segment:Offset*. Flash-Start hängt von der Bestückung ab: 1 MB: 3F00000h, 2 MB: 3E00000h, 4 MB: 3C00000h, 8 MB: 3800000h.

### 9.14.3. INT 5Fh Funktion 02h - Flash Write Block

**Aufruf:** AH = 02h
DX:DI =32-Bit Ziel-Adresse
ES:BX = Quell-Adresse (Far-Pointer auf Datenpuffer)
CX = Zahl der zu schreibenden Bytes

**Rückgabe:** Carry-Flag = 0: kein Fehler
Carry-Flag = 1: Fehler

**Beschreibung:** (Siehe auch Funktion 02h). Die Funktion führt keine Lösch-Operation aus. Falls der zu schreibende Bereich nicht gelöscht ist, kehrt die Funktion mit Fehler zurück.

### 9.14.4. INT 5Fh Funktion 03h - Flash Erase and Write Block

**Aufruf:** AH = 03h
DX:DI =32-Bit Ziel-Adresse
ES:BX = Quell-Adresse (Far-Pointer auf Datenpuffer)
CX = Zahl der zu schreibenden Bytes

**Rückgabe:** Carry-Flag = 0: kein Fehler
Carry-Flag = 1: Fehler

**Beschreibung:** (Siehe auch Funktion 02h). Der betreffende Flash-Block wird vor dem Schreiben gelöscht.

### 9.14.5. INT 5Fh Funktion 04h - Read Flash Chip and Manufacturer ID

**Aufruf:**
AH = 04h
DX:DI = 32-Bit Quell-Adresse = 03F0:0000
ES:BX = Ziel-Adresse (Far-Pointer auf Datenpuffer im RAM)

**Rückgabe :**
[ES:BX] = Device ID
[ES:BX + 2] = Manufacturer ID
Carry-Flag = 0: kein Fehler
Carry-Flag = 1: Fehler

**Beschreibung:** Liest den ID-Code der Flash-ICs aus. Je nach Bestückungsvariante können z.B. die folgenden ID-Codes vorkommen (die Liste ist nicht vollständig):

29LV800BT: 22DAh
29LV160BT: 22C4h
29LV320DT: 22F6h
29LV641D : 22D7h

Manufacturer: AMD: 01, Fujitsu: 04, ST: 20h

# 10. Tabellen

## 10.1. I/O-Adressen

| Port Addresse | Funktion |
|---|---|
| 000h – 00Fh | DMA Controller * |
| 020h – 021h | 1. Interrupt Controller (Master) |
| 022h – 023h | 386EX Control Register |
| 040h – 043h | Timer 0, 1 und 2 |
| 080h – 083h | DMA Pageregister * |
| 092h | A20-Gate, CPU Reset |
| 0A0 – 0A1 | 2. Interrupt Controller (Slave) |
| **100h – 103h** | **PIF-Bus Konfiguration:** |
| 100h | Datenport: Setzen/Lesen der Pin-Zustände PD0..PD7 |
| 101h | Kontrollport: Setzen/Lesen der Pin-Zustände PA0..PA6 |
| 102h | I/O-Konfiguration (1 = Output, 0 = Input)<br>Bit0: PD0..PD3,<br>Bit1: PD4..PD7,<br>Bit2: PA0..PA3,<br>Bit3: reserved<br>Bit4: PA4 (–RD),<br>Bit5: PA5 (–WR)<br>Bit6: PA6 (READY)<br>Bit7: PIF-Bus Fast Mode<br>Für gültige PIF-Bus-Zyklen setze man Port 102h auf 3Fh oder BFh |
| 103h | Bit 4: Toggle PA4,<br>Bit 5: Toggle PA5,<br>Bit 6: Toggle PA6<br>Bit 0..3, Bit 7: reserved |
| 2F8h – 2FFh | COM2 |
| 300h – 30Fh | PIF-Bus, –CS0 aktiv |
| 310h – 31Fh | PIF-Bus, –CS1 aktiv |
| 320h – 32Fh | PIF-Bus, –CS2 aktiv |
| 330h – 33Fh | PIF-Bus, –CS3 aktiv |
| 3F8h – 3FFh | COM1 |
| **F000h – FFFFh** | **386EX Peripherie:** |
| F480h – F48Bh | synchrone serielle Schnittstelle |
| F860 | I/O-Port P1 Input |
| F862 | I/O-Port P1 Output |
| F864 | I/O-Port P1 Direction |
| F868 | I/O-Port P2 Input |
| F86A | I/O-Port P2 Output |
| F86C | I/O-Port P2 Direction |
| F870 | I/O-Port P3 Input |
| F872 | I/O-Port P3 Output |
| F874 | I/O-Port P3 Direction |

* DMA wird vom BIOS des MicroPC nicht unterstützt.

## 10.2. Interrupt-Tabelle

| Vektor | Adresse | Verwendung | Service-Routine durch | Typ |
|---|---|---|---|---|
| 00 | 000 | Divide by Zero | BIOS oder Anwendung | CPU-Exception |
| 01 | 004 | Single Step Debug Interrupt | Debugger | CPU-Exception |
| 02 | 008 | Non-Maskable Interrupt (NMI) | Anwendung | Hardware-NMI |
| 03 | 00C | One-Byte Debug Breakpoint | Debugger | CPU-Befehl |
| 04 | 010 | Overflow (aufgerufen durch INTO Befehl) | Anwendung | CPU-Exception |
| 05 | 014 | Array Bounds Check (aufgerufen durch BOUNDS Befehl) | Anwendung | CPU-Exception |
| 06 | 018 | Ungültiger Befehls-Code | Debugger | CPU-Exception |
| 07 | 00C | Coprozessor nicht vorhanden | Anwendung | CPU-Exception |
| 08 | 020 | Timer 0 (System-Timer) | BIOS | Hardware-IRQ0 |
| 09 | 024 | PIF-Bus IRQ (PIF-INT) | Anwendung | Hardware-IRQ1 |
| 0A | 028 | reserviert | BIOS | Software |
| 0B | 02C | serielle Schnittstelle (COM2) | BIOS oder Anwendung | Hardware-IRQ3 |
| 0C | 030 | serielle Schnittstelle (COM1) | BIOS oder Anwendung | Hardware-IRQ4 |
| 0D | 034 | frei für Anwendung | Anwendung | Hardware-IRQ5 |
| 0E | 038 | frei für Anwendung | Anwendung | Hardware-IRQ6 |
| 0F | 03C | frei für Anwendung | Anwendung | Hardware-IRQ7 |
| 10 | 040 | Video Funktionen | BIOS | Software |
| 11 | 044 | System Konfiguration (Equipment Check) | BIOS | Software |
| 12 | 048 | RAM Größe | BIOS | Software |
| 13 | 04C | Disk Funktionen | BIOS | Software |
| 14 | 050 | Serial Port Funktionen (COM1-4) | BIOS | Software |
| 15 | 054 | diverse, u.a. MicroPC-spezifische Funktionen | BIOS | Software |
| 16 | 058 | Keyboard Funktionen | BIOS | Software |
| 17 | 05C | Parallel Port Funktionen (LPT) | BIOS | Software |
| 18 | 060 | Boot Fehler | BIOS | Software |
| 19 | 064 | Boot Loader | BIOS oder DOS | Software |
| 1A | 068 | System-Timer- und RTC-Funktionen | BIOS | Software |
| 1B | 06C | reserviert | BIOS | Software |
| 1C | 070 | Timer User Funktion (aufgerufen von Int 08h) | Anwendung | Software |
| 1D | 074 | reserviert | BIOS | Software |
| 1E | 078 | Disk Parameter Tabelle | -- | BIOS-Tabelle |
| 1F | 07C | reserviert | BIOS | Software |
| 20 – 3F | 080 – 0FC | reserviert für DOS | DOS | Software |
| 40 – 49 | 100 – 124 | reserviert für BIOS | BIOS | Software |
| 4A | 128 | RTC User Funktion (aufgerufen von Int 70h) | Anwendung | Software |
| 4B – 5E | 12C – 178 | reserviert für BIOS | BIOS | Software |
| 5F | 17C | Flash Funktionen | Anwendung | Software |
| 60 – 6F | 180 –1BC | frei für Anwendung | Anwendung | Software |

Fortsetzung Interrupt-Tabelle

| Vektor | Adresse | Verwendung | Service-Routine durch | Typ |
|---|---|---|---|---|
| 70 | 1C0 | Real time Clock (RTC) Alarm | BIOS | Hardware-IRQ8 |
| 71 | 1C4 | synchrone serielle Schnittstelle oder frei für Anwendung | BIOS oder Anwendung | Hardware-IRQ9 |
| 72 | 1C8 | Timer 1 | LCD-BIOS-Extension oder Anwendung | Hardware-IRQ10 |
| 73 | 1CC | Timer 2 | Anwendung | Hardware-IRQ11 |
| 74 | 1D0 | reserviert (DMA) | -- | Hardware-IRQ12 |
| 75 | 1D4 | frei für Anwendung | Anwendung | Hardware-IRQ13 |
| 76 | 1D8 | IDE (Compact-Flash) | BIOS | Hardware-IRQ14 |
| 77 | 1DC | 386EX-Watchdog Timer | Anwendung | Hardware-IRQ15 |
| 78 – FF | 1E0 – 3FC | frei für Anwendung | Anwendung | Software |

## 10.3. Steckerbelegung

| Pin-Nr. | Funktionen | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | PIF-Bus | Timer | SIO | SSIO | Digital Port | IRQ | I²C-Bus[1] | JTAG | Sonstige |
| 01 | GND | | | | | | | | |
| 02 | D0 | | | | PD0 | | | | |
| 03 | D2 | | | | PD2 | | | | |
| 04 | D4 | | | | PD4 | | | | |
| 05 | D6 | | | | PD6 | | | | |
| 06 | –WR | | | | PA5 | | | | |
| 07 | A0 | | | | PA0 | | | | |
| 08 | A2 | | | | PA2 | | | | |
| 09 | –CS0 | | | | P2.0 | | | | |
| 10 | –CS2 | | | | P2.2 | | | | |
| 11 | –RESET | | | | | | | | |
| 12 | | | | | P3.3 | IRQ5 | x | | |
| 13 | VCC | | | | | | | | |
| 14 | | TOUT0 | | | P3.0 | IRQ4[2] | x | | |
| 15 | | | TXD1 | | | | | | |
| 16 | | TCLK1 | CTS1 | | | IRQ13 | | | |
| 17 | | | DSR1 | STXCLK | | | | | |
| 18 | | | RI1 | SSIORX | | | | | |
| 19 | | | RXD0 | | P2.5 | | x | | |
| 20 | | | CTS0 | | P2.7 | | x | | |
| 21 | | | DTR0 | | P1.2 | | x | | |
| 22 | | | RI0 | | P1.4 | | x | | |
| 23 | | | | | | | | TMS | |
| 24 | | /TCLK0 | | | | /IRQ8[3] | | TDI | Wakeup |
| 25 | GND | | | | | | | | |
| 26 | GND | | | | | | | | |
| 27 | D1 | | | | PD1 | | | | |
| 28 | D3 | | | | PD3 | | | | |
| 29 | D5 | | | | PD5 | | | | |
| 30 | D7 | | | | PD7 | | | | |
| 31 | –RD | | | | PA4 | | | | |
| 32 | A1 | | | | PA1 | | | | |
| 33 | A3 | | | | PA3 | | | | |
| 34 | –CS1 | | | | P2.1 | | x | | |
| 35 | –CS3 | | | | P2.3 | | x | | |
| 36 | | | | | P3.2 | IRQ1 | x | | |
| 37 | | | | | P3.5 | IRQ7 | SCL | | |
| 38 | VCC | | | | | | | | |
| 39 | | TOUT1 | DCD1 | | P3.1 | IRQ3[4] | x | | |
| 40 | | | RXD1 | | | | | | |
| 41 | | | RTS1 | SSIOTX | | | | | |
| 42 | | | DTR1 | SRXCLK | | | | | |
| 43 | | | TXD0 | | P2.6 | | x | | |
| 44 | | | RTS0 | | P1.1 | | x | | |
| 45 | | TGATE0 | DSR0 | | P1.3 | IRQ9[5] | x | | |
| 46 | | TGATE1 | DCD0 | | P1.0 | IRQ14[6] | x | | |
| 47 | | | | | | | | | VBATT |
| 48 | READY | | | | PA6 | | | TCK | Emergency Boot |
| 49 | | TGATE2 | | | | | | TDO | |
| 50 | GND | | | | | | | | |

[1] vom BIOS als I²C-Bus Port (SCL, SDA) verwendbare Digital-Ports. Pin 37 wird auch für den internen I²C-Bus als SCL-Signal verwendet.
[2] belegt von SIO-0 (COM1)
[3] belegt von RTC
[4] belegt von SIO-1 (COM2)
[5] belegt von SSIO
[6] belegt von externer CompactFlash Card

## 10.4. Vom BIOS verwendete ANSI Escape Sequenzen

Folgende Escape Sequenzen werden vom Video BIOS (INT 10h) verwendet :

| Escape Sequenz | Bedeutung | INT 10h |
|---|---|---|
| ESC[ # ; # H | Set cursor position | 02h |
| ESC[ 6 n | Status request | 03h |
| ESC[ s | Save cursor position | 09h, 0Ah |
| ESC[ u | Restore cursor position | 09h, 0Ah |
| ESC[ 2J | Erase screen | 00h, 0Eh |
| ESC[ # ; ... ; # m | Set colors | 06h, 07h, 09h |

Folgende Escape Sequenzen werden vom Tastatur-BIOS (INT 16h) erkannt :

| Taste | Normal | + Shift | + Ctrl | + Alt |
|---|---|---|---|---|
| F1 | ESC[M | ESC[Y | ESC[k | ESC[w |
| F2 | ESC[N | ESC[Z | ESC[l | ESC[x |
| F3 | ESC[O | ESC[a | ESC[m | ESC[y |
| F4 | ESC[P | ESC[b | ESC[n | ESC[z |
| F5 | ESC[Q | ESC[c | ESC[o | ESC[@ |
| F6 | ESC[R | ESC[d | ESC[p | ESC[[ |
| F7 | ESC[S | ESC[e | ESC[q | ESC[\ |
| F8 | ESC[T | ESC[f | ESC[r | ESC[] |
| F9 | ESC[U | ESC[g | ESC[s | ESC[^ |
| F10 | ESC[V | ESC[h | ESC[t | ESC[_ |

| | | | |
|---|---|---|---|
| Left | ESC[D | Home | ESC[H |
| Right | ESC[C | End | ESC[F |
| Up | ESC[A | PgUp | ESC[I |
| Down | ESC[B | PgDown | ESC[G |
| Insert | ESC[L | | |

## 10.5. Elektrische Daten

Umgebungstemperatur 25°C, soweit nicht anders angegeben.

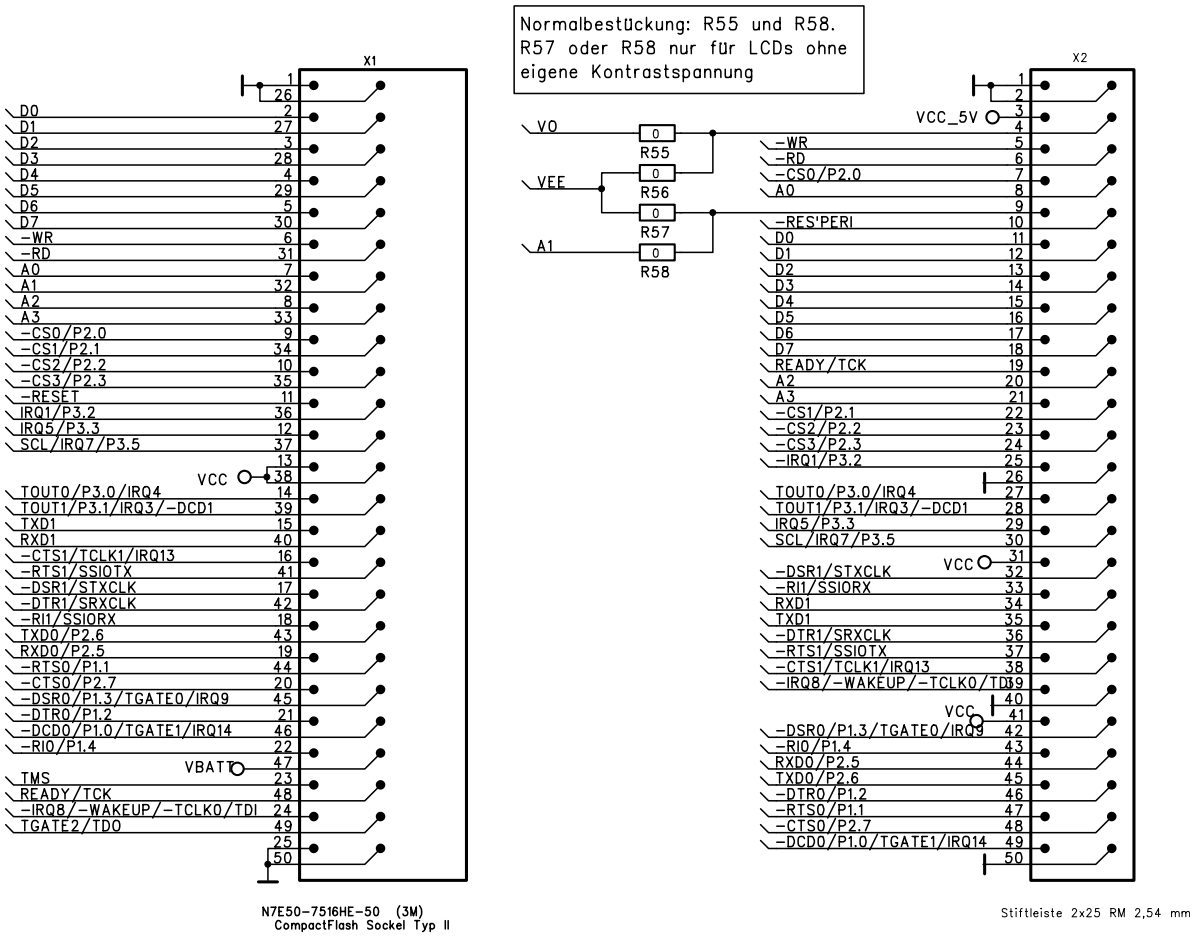
| Symbol | Beschreibung | Parameter | min. | typ. | max. | Einheit |
|---|---|---|---|---|---|---|
| Vcc | Betriebsspannung | | 3,0 | 3,3 | 3,6 | V |
| Vres | Reset-Schwelle | | | 2,9 | | V |
| tres | Dauer des Reset-Impulses | | 150 | | 280 | ms |
| $V_{IH}$ | High-Level Input Voltage | PD0..PD7, PA0..PA3, PIF-RD, PIF-WR, PIF-RDY, | 2 | | 5,5 | V |
| $V_{IH}$ | High-Level Input Voltage | alle anderen digitalen Signale | 2 | | Vcc+ 0,3 | V |
| $V_{IL}$ | Low-Level Input Voltage | | 0 | | 0,8 | V |
| Icc | Betriebsstrom im Normalbetrieb | CLK2 = 50 MHz * | | 135 | | mA |
| | | CLK2 = 40 MHz | | 107 | | mA |
| | | CLK2 = 16 MHz | | 48 | | mA |
| | | CLK2 = 8 MHz | | 27 | | mA |
| | Idle-Mode | CLK2 = 50 MHz | | | | mA |
| | | CLK2 = 40 MHz | | | | mA |
| | | CLK2 = 16 MHz | | | | mA |
| | | CLK2 = 8 MHz | | 8,5 | | mA |
| | Powerdown-Mode | CLK2 = 8 MHz | | 4,4 | | mA |
| | | CLK2 = 0 | | 0,30 | | mA |
| Vbatt | Batteriespannung (für SRAM und RTC) | | 2,0 | 3 | 3,6 (Vcc) | V |
| Ibatt | Batteriestrom bei abgeschalteter Betriebsspannung | 512 kB SRAM Bestückung Batteriespannung = 3V Umgebungstemperatur = 25°C | | 3 | | µA |
| | | Umgebungstemperatur = 70°C | | | 17 | µA |
| | | Umgebungstemperatur = 85°C | | | 22 | µA |

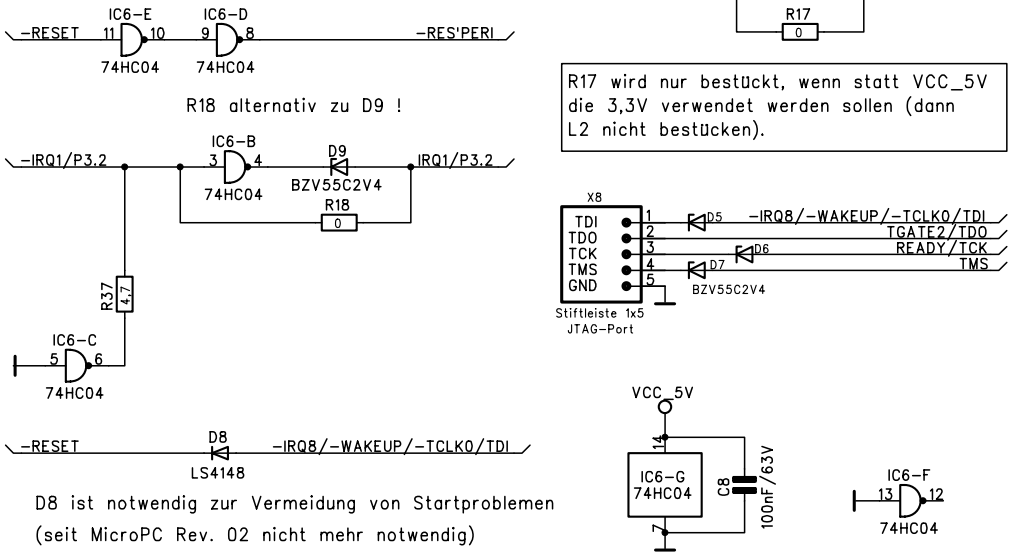
Anmerkungen:

* Die nominelle Taktrate – nach der die Befehlszeiten der CPU angegeben werden – beträgt 50% der Oszillatorfrequenz CLK2.

# 11. Schaltplan MicroPC Starterkit-Platine

## 11.1. BUS-Stecker



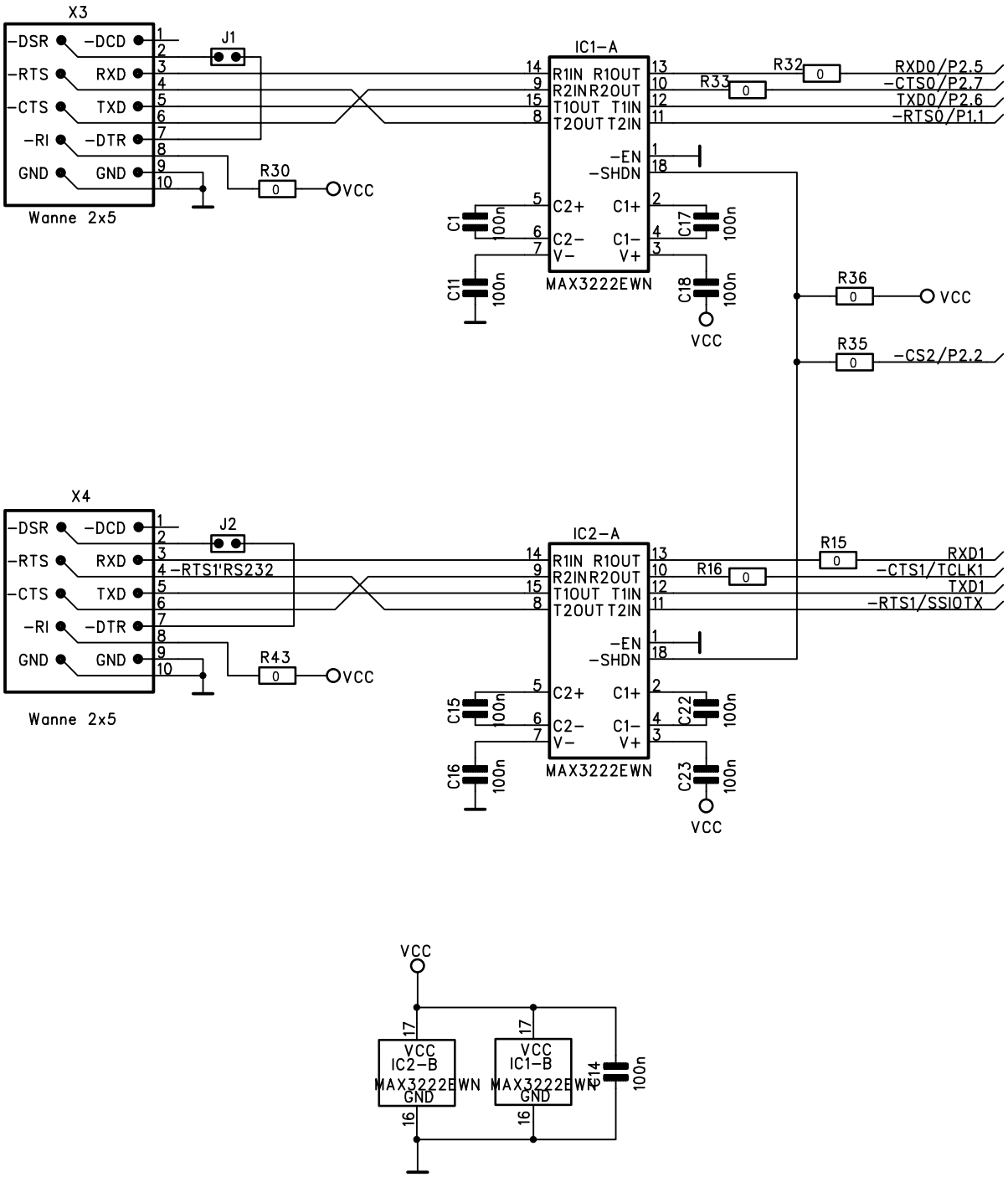



## 11.2. RS232-Treiber

Die beiden RS232-Treiber ICs IC1 und IC2 können bei Bedarf per Digital-Port Ausgang in den Standby-Mode geschaltet werden, in welchem sie nur noch einige µA Strom verbrauchen. Hierzu dient der Port –CS2 im Zusammenhang mit R35. Falls –CS2 als PIF-Bus Chip-Select verwendet wird, sollte R36 bestückt sein.

Die Brücken R30 und R43 ermöglichen es, unter Verzicht auf das RI-Signal eine Versorgungsspannung von 3,3 V auf den SIO-Stecker zu legen.

Falls die Signale einer seriellen Schnittstelle auf dem X1-Stecker verwendet werden sollen, können die Signale RXD und CTS der betreffenden Schnittstelle vom RS232-Treiber abgetrennt werden. Dies geschieht mittels der Brücken R32, R33, R15 und R16.



## 11.3. Keyboard, LCD

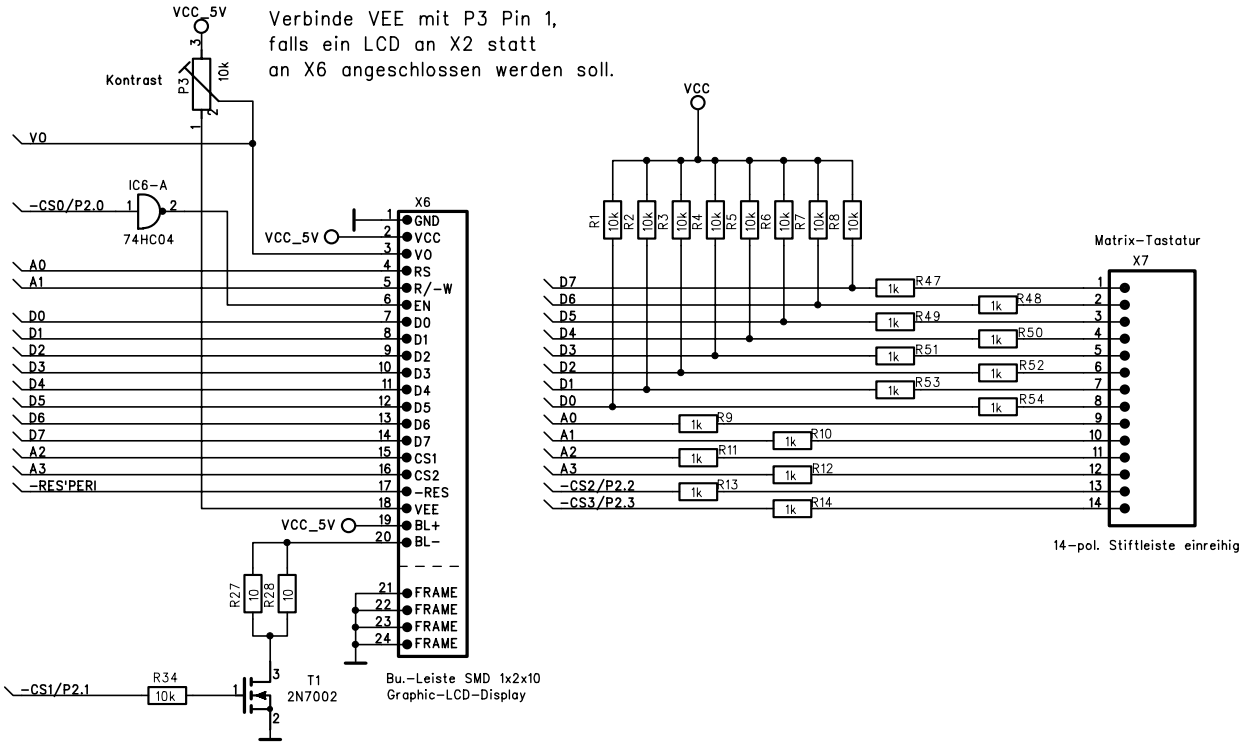
Der Steckverbinder X7 ist vorgesehen für den Anschluß eines LCDs mit dem Controller KS0108 von Samsung (neuerdings auch unter der Bezeichnung: S6B0108) oder dem HD61202 von Hitachi. Da der Bus dieser Controller nicht direkt zum PIF-Bus kompatibel ist, wird er mittels eines Inverters und durch Verwendung zusätzlicher I/O-Adressen (getrennt zum Lesen und Schreiben) angepaßt. Ein LCD belegt dann die I/O-Adressen 304h..30Fh.

Das Poti P3 dient zur Einstellung des LCD-Konrastes.

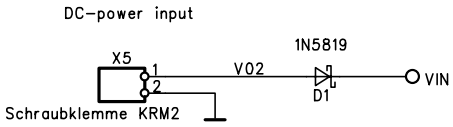
Das Scannen einer 4 x 8 Matrix-Tastatur kann einfach mittels der PIF-Bus Signale D0..D7 und A0..A3 geschehen. Hierfür werden die I/O-Adressen 100h..102h verwendet, so daß keine regulären PIF-Bus Zyklen stattfinden (ein solcher sieht vor, daß –RD oder –WR aktiv wird). Demgemäß belegt der Tastaturport auch keine I/O-Adressen des PIF-Bus. Damit das Drücken der Tasten nicht den PIF-Bus stört, sind Längswiderstände in den Tastaturanschlüssen vorgesehen.

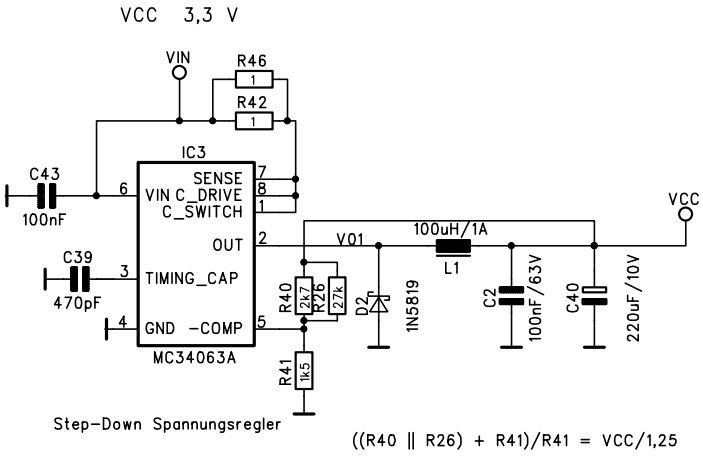
Für 8 x 6 Matrix-Tastaturen werden zusätzlich die Signale -CS2 und -CS3 als Ports verwendet. Der Einsatz von -CS0 und -CS1 (für 8 x 8 Tastaturen) kollidiert bei der MicroPC-Base Platine mit dem Standard-LCD (-CS0 wird vom Controller belegt, -CS1 als Port für LCD-Backlight).

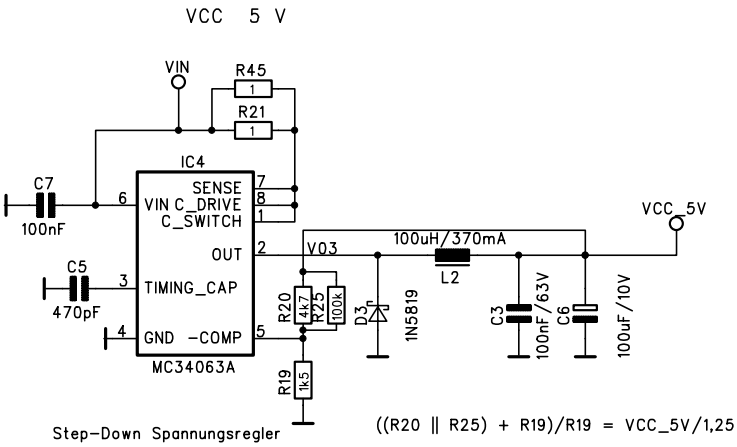
Der Tastaturscan wird zweckmäßigerweise innerhalb der Interrupt-Routine des System-Timers (Timer 0) durchgeführt. Daher ist auch ein Hardware-Interrupt der Tastatur nicht unbedingt notwendig. Allerdings wird ein solcher dann benötigt, wenn die Betätigung der Tastatur automatisch die Beendigung des Idle- oder Powerdown-Modes hervorrufen soll. Ein Hardware-Interrupt kann einfach realisiert werden durch ein 8-fach NAND (74HC30) an D0..D7. Damit nicht jeder PIF-Bus-Zyklus den Tastatur-Interrupt auslöst, wird der verwendete IRQ ausmaskiert und nur bei Aktivierung von Idle- oder Powerdown-Mode geschärft.

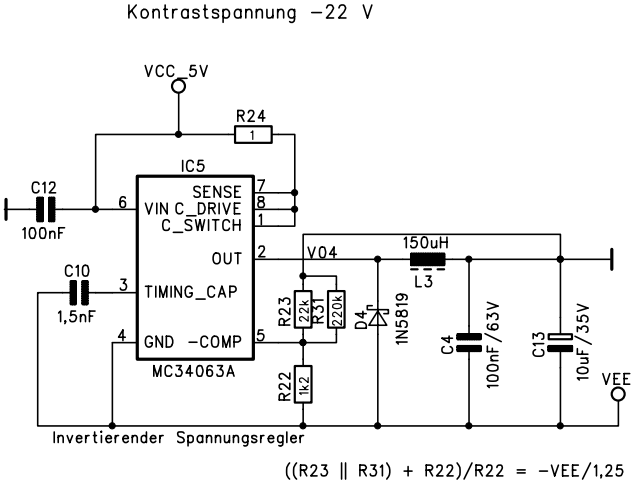


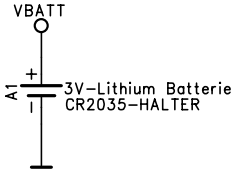
## 11.4. Stromversorgung



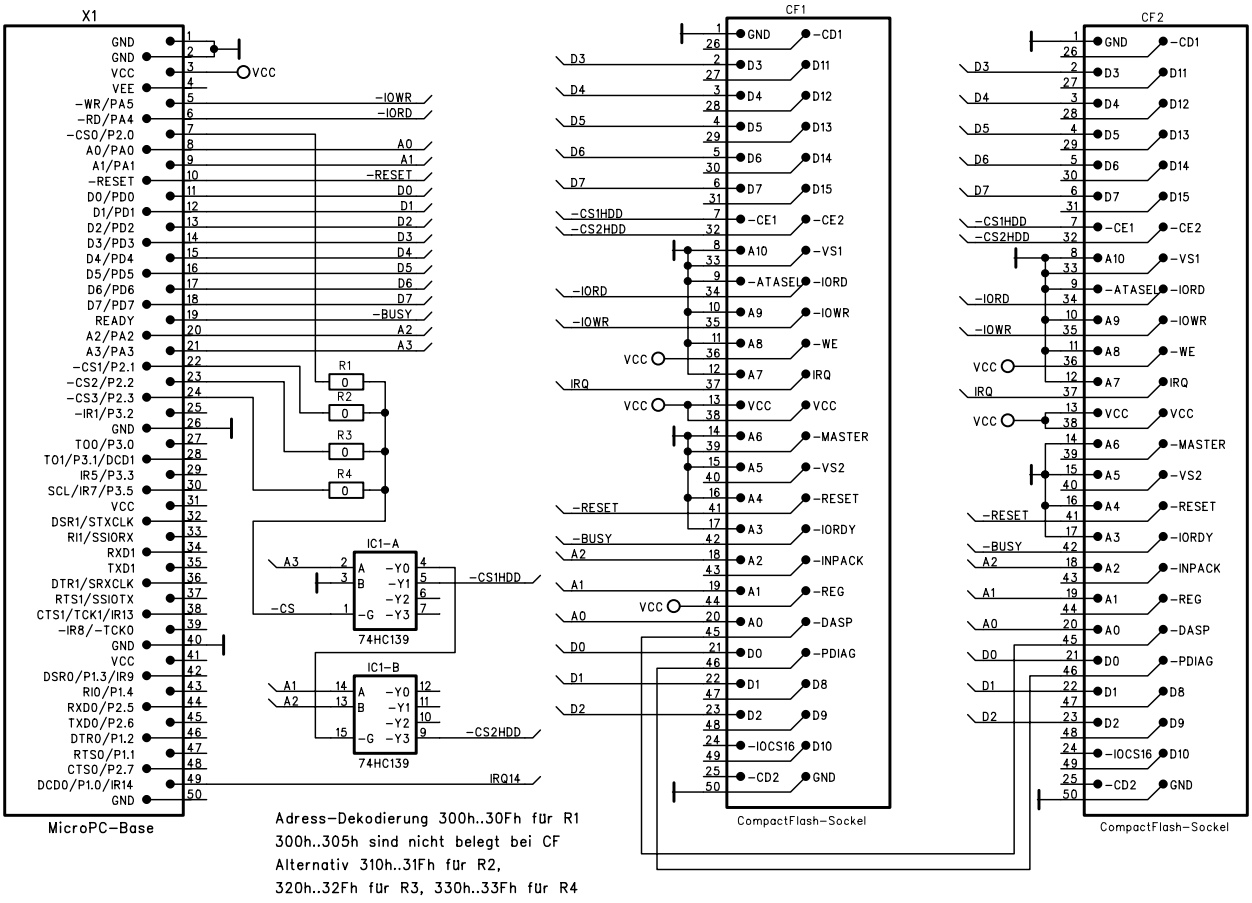



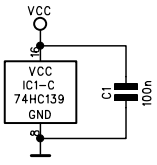



# 12. Schaltplan MicroPC CompactFlash Anschluß

Gemäß dem folgenden Schaltplan kann ein Adapter für zwei CompactFlash Cards aufgebaut werden. Obwohl es möglich ist, einen rein passiven Adapter zu implementieren unter Einsatz von zwei Chip-Selects des PIF-Bus, wird hier ein IC (74HC139, oder 74LCX139) zur Adreßdekodierung verwendet. Dadurch belegen die CompactFlash Cards nur 10 statt 20 I/O-Adressen.

Von den vier Chip-Selects -CS0..-CS3 wird nur eines benötigt, dementsprechend wird auch nur eine der Brücken R1..R4 bestückt. Die BIOS-Funktionen für CompactFlash arbeiten normalerweise mit -CS0, d.h. mit dem Adreßbereich 306h..30Fh.



# 13. Schaltplan MicroPC Ethernet-Adapter