MoDOS 386

Development Kit

taskit GmbH

Seelenbinderstr. 33
12555 Berlin (Germany)

Telefon +49(0)30 / 611295-0
Fax +49(0)30 / 611295-10

1. Inbetriebnahme

Das MODOS wird betriebsfertig mit geladenen Akkus ausgeliefert. Zum Einschalten wird die Taste "Start" etwa eine halbe Sekunde lang gedrückt. Anschließend läuft ein einfaches Demo-Programm ab.
Die LCD-Beleuchtung kann man durch wiederholtes Betätigen der Start-Taste einstellen.
Das Gerät wird durch die Tastenkombination "Shift" - "Start" (nacheinander drücken) abgeschaltet.

2. Verbindung mit einem PC

Zur Kommunikation mit einem PC wird das MODOS über die MODOS-Charge-Box (diese enthält den RS-232-Treiber) an eine serielle Schnittstelle angeschlossen.

Das zugehörige Terminalprogramm "VTERM.EXE" befindet sich im Verzeichnis "BIOS" auf der beiliegenden Diskette. Zur Bedienung von VTERM werden Tastenkombinatinen mit der "Alt"-Taste verwendet, alle übrigen Tastendrücke werden unverändert an das MODOS weitergegeben. Insbesondere ist also auch "F1" nicht die VTERM Online-Hilfe (sondern Alt-h), die richtige serielle Schnittstelle kann man mit Alt-c einstellen.

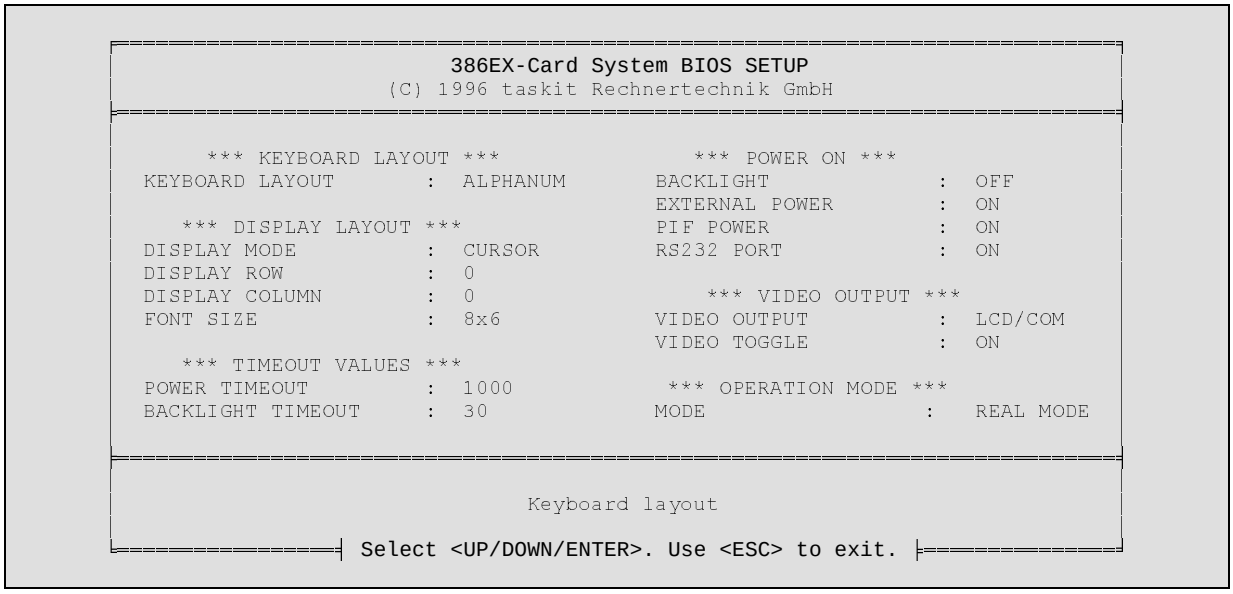
Nach Beendigung des Demo-Programms (mit der Taste "4" - exit to DOS) erscheint der DOS-Prompt auf dem PC-Bildschirm. Jetzt können beliebige DOS-Befehle bzw. -Programme auf dem MODOS ausgeführt werden. Insbesondere kann man mit "copy" Daten und Programme vom MODOS zum PC und umgekehrt transferieren.

3. Batterie-Ladung

Das MODOS wird an die Charge-Box angeschlossen, diese wiederum an das Steckernetzteil. Ein Ladezyklus (bei weitgehend entladenen Batterien) dauert etwa 5 Stunden.

4. MoDOS-Setup

Zusätzlich zum grundlegenden BIOS-Setup der CPU-Karte, welches mit der Taste „S“ (wie **S**etup) beim Booten aufgerufen wird, gibt es beim MoDOS mit 386EX-Card ein Setup der BIOS-Erweiterung, in dem einige MoDOS-spezifische Einstellungen vorgenommen werden können. Dieses Setup während des Bootens durch die Taste „M“ (für **M**oDOS) des PC aufgerufen. Es erscheint unter Vterm die folgende Maske.



KEYBOARD LAYOUT: Hier wird je nach Ausführung des MoDOS entweder die alphanumerische Tastatur mit 45 Tasten oder die numerische Tastatur mit 21 Tasten eingestellt. Eine falsche Einstellung führt zu Fehlfunktion der Tastatur.

DISPLAY MODE: Da der normale DOS-Bildschirm 25x80 Zeichen hat, während auf dem MoDOS-Display nur maximal 8x21 Zeichen dargestellt werden, ist eine Einstellung des LCD-Fensters im DOS-Bildschirm notwendig.

CURSOR	(default) Das LCD-Fenster folgt der aktuellen Cursor-Position.
FIXED	Das LCD-Fenster befindet sich an der durch DISPLAY ROW und DISPLAY COLUMN festgelegten Stelle des DOS-Bildschirms.
COL FIXED	Die linke Spalte des LCDs befindet sich in der durch DISPLAY COLUMN festgelegten Spalte des DOS-Bildschirms, die vertikale Position folgt dem Cursor.
ROW FIXED	Die obere Zeile des LCDs befindet sich in der durch DISPLAY ROW festgelegten Zeile des DOS-Bildschirms, die horizontale Position folgt dem Cursor.
DISPLAY ROW	fixiert die vertikale Position des LCD-Fensters (0 bis 17)
DISPLAY COLUMN	fixiert die horizontale Position des LCD-Fensters (0 bis 59).

4.1. **FONT SIZE** (Größe des Zeichensatzes): Möglich sind die Einstellungen 8 x 6, 8 x 8 und 8 x 16 Pixel pro Zeichen. Dies entspricht 21 x 8, 16 x 8 oder 16 x 4 Zeichen auf dem LCD.

POWER TIMEOUT: Legt fest, nach welcher Zeit das MoDOS sich selbsttätig abschaltet. Möglich sind Werte bis 65534 Sekunden (etwa 18 Stunden). Jede Betätigung der Matrix-Tastatur setzt den Power-Timeout-Zähler auf diesen Anfangswert zurück. Der Wert 65535 (0FFFFh) deaktiviert die Auto-Power-Off-Funktion.

Achtung: die serielle Schnittstelle setzt den Zähler nicht zurück (dies ist dem Anwendungsprogramm vorbehalten). Beim Arbeiten mit Vterm kann es also durchaus zum Auto-Power-Off kommen, wenn ein ungünstiger Wert eingestellt ist.

BACKLIGHT TIMEOUT: Legt fest, nach welcher Zeit die LCD-Beleuchtung sich selbsttätig abschaltet. Es gelten ansonsten die gleichen Bedingungen wie für POWER TIMEOUT.

BACKLIGHT: Gibt an, ob die LCD-Beleuchtung beim Einschalten des Geräts ebenfalls eingeschaltet werden soll.

EXTERNAL POWER: Schaltet die Versorgungsspannung des MoDOS auf den Pin „USER VCC“ des DSUB-Steckers zur Versorgung von externer Peripherie.
PIF POWER: Schaltet die Versorgungsspannung des MoDOS auf den Pin VEE des PIF-Bus. Wird eine an den PIF-Bus angeschlossene Peripherie-Einheit über VEE (statt VCC) versorgt, ist sie damit per Software abschaltbar.

RS232 PORT: Der optionale RS232-Treiber der ersten seriellen Schnittstelle kann hiermit ein- oder ausgeschaltet werden.

VIDEO OUTPUT: Legt fest, ob DOS-Aus- und Eingaben nur über LCD und Matrix-Tastatur oder auch über die serielle Schnittstelle erfolgen.
VIDEO TOGGLE: Legt fest, ob die Einstellung für VIDEO OUTPUT mit „Shift-Enter“ auf der Matrix-Tastatur geändert werden kann.

MODE: Legt fest, ob das Gerät im Real Mode des 386EX oder im virtuellen 8086-Mode (Protected) betrieben werden soll.
Hinweis: Im Protected Mode stellt das Betriebssystem zusätzliche Funktionalität zur Verfügung, insbesondere die VGA-Emulation (im Text-Modus) für die LCD-Ausgabe und den virtuellen Bildschirmspeicher. Allerdings sind die betreffenden BIOS-Funktionen zeitaufwendiger, so daß viele Programme im Real Mode schneller ablaufen. Dagegen ist die Verwendung von Ausgabefunktionen, die direkt auf VGA-Register zugreifen, nur im Protected Mode möglich. Hiervon betroffen sind z.B. die Ausgabefunktionen der Unit CRT von Turbo-Pascal.

5. Funktions-Bibliothek des 386EX-MoDOS

Die Programmierung entspricht im wesentlichen der der 386EX-Card, die den CPU-Kern des MODOS bildet (siehe Handbuch „386EX-Card“).
Für die Ansteuerung von LCD, Matrix-Tastatur und Power-Save eine eigene Programm-Bibliothek. Die Funktionen sind als BIOS-Erweiterung implementiert. Es gibt zwei Versionen: eine zur V40-Card kompatible (hiermit lassen sich Programme für das MoDOS mit V40-Card auf die 386EX-Card portieren), und eine auf die speziellen Eigenschaften des 386EX-Prozessors zugeschnittene (386EX-Version). Für neue Programme ist die letztere Version vorzuziehen. Die Beschreibung der Funktionen der V40-kompatiblen Version entspricht der Kapitels über die Funktionsbibliothek des V40-MoDOS, so daß hier nur die 386EX-Version dargestellt wird.

Die MoDOS-Funktionen sind jetzt zum größten Teil als Funktionen des Int 15h implementiert. Für die **Matrix-Tastatur** gibt es keine speziellen Funktionen mehr, da sie jetzt über die normalen Tastatur-Funktionen des PC ansprechbar ist (Int 16h bzw. die diversen Bibliotheks-Funktionen der Programmiersprachen).

5.1. Allgemeine Funktionen

5.1.1. SetDefaults

Assembler Interface: Int 15h

Parameter:	AX	-	201Ch
Rückgabe:	keine		

C-Library Interface

Funktionsprototyp:	void SetDefaults (void);
Rückgabe:	keine

Die Funktion initialisiert die folgenden Variablen:

PowerCount	= 300 Sekunden
BacklightCount	= 30 Sekunden
BacklightStatus	= Off
ScrollMode	= 0
CursorOffset	= 0
ScrollOffset	= 0

5.1.2. GetBiosVersion

Assembler Interface: Int 15h

Parameter:	AX	-	201Dh
Rückgabe:	DL	-	Nummer der BIOS-Version

C-Library Interface

Funktionsprototyp:	unsigned char GetBiosVersion(void);
Rückgabe:	Nummer der BIOS-Version

5.1.3. Beep

Assembler Interface: Int 15h

Parameter:	AX	-	2050h
	CX	-	Anzahl der Beeps
Rückgabe:	keine		

C-Library Interface

Funktionsprototyp:	void Beep(unsigned int cnt);
Rückgabe:	keine

Die Funktion gibt eine Folge von Piep-Tönen auf dem MoDOS-Beeper aus.

5.1.4. GetBacklight

Assembler Interface: Int 15h

Parameter:	AX	-	200Fh
Rückgabe:	0...3, 0 = Aus		

C-Library Interface

Funktionsprototyp:	unsigned int GetBacklight(void)
Rückgabe:	0...3, 0 = Aus

Status der Hintergrundbeleuchtung abfragen

5.1.5. SetBacklight

Assembler Interface: Int 15h

Parameter:	AX	-	2010h
	DL	-	bkl
Rückgabe:	keine		

C-Library Interface

Funktionsprototyp:	void SetBacklight(unsigned char bkl)
Rückgabe:	keine

Hintergrundbeleuchtung setzen (0...3)
bkl=0: Hintergrundbeleuchtung aus
bkl=3: größte Helligkeit

5.1.6. EnableRedir

Assembler Interface: Int 15h

Parameter:	AX	-	2038h
Rückgabe:	keine		

C-Library Interface

Funktionsprototyp:	void EnableRedir(void)
Rückgabe:	keine

DOS-Standard-I/O findet nur noch durch LCD und Matrix-Tastatur statt, nicht über die serielle Schnittstelle.

5.1.7. DisableRedir

Assembler Interface: Int 15h

Parameter:	AX	-	2039h
Rückgabe:	keine		

C-Library Interface

Funktionsprototyp:	void DisableRedir(void)
Rückgabe:	keine

DOS-Standard-I/O über LCD und Matrix-Tastatur und serielle Schnittstelle einschalten

5.2. Power-Management-Funktionen

5.2.1. GetPowerOffDelay

Assembler Interface: Int 15h

Parameter:	AX	-	2017h
Rückgabe:	DX	-	PowerOffDelay

C-Library Interface

Funktionsprototyp:	unsigned int GetPowerOffDelay(void)
Rückgabe:	PowerOffDelay

Beschreibung siehe GetPowerOffTimer.

5.2.2. SetPowerOffDelay

Assembler Interface: Int 15h

Parameter:	AX	-	2018h
Rückgabe:	keine		

C-Library Interface

Funktionsprototyp:	void SetPowerOffDelay(unsigned int delay)
Rückgabe:	keine

Auomatische Abschaltzeit in Sekunden einstellen (16-Bit Zähler, bis ca. 18 Stunden)
delay = -1: automatisches Abschalten deaktivieren
Siehe auch GetPowerOffTimer

5.2.3. PowerOff

Assembler Interface: Int 15h

Parameter:	AX	-	2019h
Rückgabe:	keine		

C-Library Interface

Funktionsprototyp:	void PowerOff(void)
Rückgabe:	keine

Gerät ausschalten (Einschränkungen: s. QueryPowerOff)

5.2.4. PowerSaveDeep

Assembler Interface: Int 15h

Parameter:	AX	-	201Ah
Rückgabe:	keine		

C-Library Interface

Funktionsprototyp:	void PowerSaveDeep(void)
Rückgabe:	keine

PowerSaveDeep: Gerät in den Stromsparmodus versetzen
Als Parameter wird mit der Funktion SetPowerIdleMode einer der folgenden Werte angegeben:

0: Deep Power Down, CPU-Kern und Peripherie stillgelegt, 20 mA Reststrom
1: Power Idle Mode, nur CPU-Kern angehalten, 30 mA Reststrom
2: Power Save Light, nur LCD-Backlight abschalten

Die CPU kehrt nur durch einen Interrupt Request aus den Modes 0 und 1 zurück, im Mode 0 sogar nur durch externen IRQ (da Timer und UARTs ebenfalls abgeschaltet sind), zum Beispiel durch die Uhr (RTC). Interrupt-Quellen, die den Stromsparmodus nicht verändern sollen, müssen im Interrupt-Controller ausmaskiert werden. Da es meistens nicht sinnvoll ist, den RTC- oder Timer- Interrupt abzuschalten, muß die Funktion PowerSaveDeep in einer Warteschleife permanent aufgerufen werden, um den Stromsparmodus nach Beendigung der Interrupt-Routine sofort wieder einzuschalten. Ein einfaches Beispiel:

```
while(1){
  if (kbhit()){
     ch = getch();
     printf("%c", ch);
     LcdForceUpdate();
  }
  PowerSaveDeep();
}
```

5.2.5. SetPowerIdleMode

Assembler Interface: Int 15h

Parameter:	AX	-	2009h
	DL	-	Power Mode
Rückgabe:	keine		

C-Library Interface

Funktionsprototyp:	void SetPowerIdleMode(unsigned char pow_mode)
Rückgabe:	keine

Als Parameter wird mit der Funktion SetPowerIdleMode einer der folgenden Werte angegeben:

0: Deep Power Down, CPU-Kern und Peripherie stillgelegt, 20 mA Reststrom
1: Power Idle Mode, nur CPU-Kern angehalten, 30 mA Reststrom
2: Power Save Light, nur LCD-Backlight abschalten

5.2.6. GetPowerIdleMode

Assembler Interface: Int 15h

Parameter:	AX	-	2008h
Rückgabe:	DL	-	Power Mode

C-Library Interface

Funktionsprototyp:	unsigned char GetPowerIdleMode(void)
Rückgabe:	Power Mode

(siehe SetPowerIdleMode)

5.2.7. GetPowerStatus

Assembler Interface: Int 15h

Parameter:	AX	-	2Eh
Rückgabe:	DX	-	Power-Status

C-Library Interface

Funktionsprototyp:	unsigned int GetPowerStatus(void)
Rückgabe:	Power-Status

Power-Status = -1: PowerFail (Batteriespannung zu niedrig), Gerät schaltet (im Extremfall) nach wenigen Sekunden ab.

Power-Status = 0: PowerGood

5.2.8. EnablePowerOff

Assembler Interface: Int 15h

Parameter:	AX	-	2030h
Rückgabe:	keine		

C-Library Interface

Funktionsprototyp:	void EnablePowerOff (void)
Rückgabe:	keine

Das Funktionspaar EnablePowerOff und DisablePowerOff kann gleichzeitig von verschiedenen Tasks verwendet werden. Es kann also keine PowerOff-Freigabe durch eine Funktion stattfinden, während sich eine andere in einer kritischen Sektion befindet. Um das Ausschalten zu ermöglichen, muß EnablePowerOff genau so oft aufgerufen werden, wie vorher DisablePowerOff aufgerufen wurde (hierzu wird ein Zähler von 0 bis 255 verwendet).

5.2.9. DisablePowerOff

Assembler Interface: Int 15h

Parameter:	AX	-	2031h
Rückgabe:	keine		

C-Library Interface

Funktionsprototyp:	void DisablePowerOff(void)
Rückgabe:	keine

Abschalten verhindern, z.B. beim Schreiben auf die Flash-Disk

5.2.10. QueryPowerOff

Assembler Interface: Int 15h

Parameter:	AX	-	2032h
Rückgabe:	DH	-	PowerOffRequest
	DL	-	PowerOffDisable

C-Library Interface

Funktionsprototyp: QueryPowerOff(void)
Rückgabe: keine

Rückgabewerte: High-Byte: PowerOffRequest
Low-Byte: PowerOffDisable

PowerOffRequest = -1: PowerOff wurde (z.B. durch Drücken von Shift-Start) in einer kritischen Sektion aufgerufen, in der das Ausschalten vermieden werden muß. Der nächste endgültige Aufruf von EnablePowerOff schaltet das Gerät aus.
PowerOffRequest = 0: normaler Betriebszustand
PowerOffDisable > 0: Ausschalten momentan nicht möglich
PowerOffDisable = 0: Ausschalten ist möglich

5.2.11. ResetPowerOff

Assembler Interface: Int 15h

Parameter:	AX	-	2033h
Rückgabe:	keine		

C-Library Interface

Funktionsprototyp: void ResetPowerOff(void)
Rückgabe: keine

PowerOffRequest und DisablePowerOff zurücksetzen (s. QueryPowerOff)

5.2.12. GetPowerOffTimer

Assembler Interface: Int 15h

Parameter:	AX	-	2034h
Rückgabe:	DX	-	Power-Off Zähler

C-Library Interface

Funktionsprototyp: unsigned int GetPowerOffTimer(void)
Rückgabe: Power-Off Zähler

Der Poweroff-Zähler (16-Bit-Variable) wird bei jedem Interrupt des Uhrenbausteins - normalerweise einmal in der Sekunde - um 1 erniedrigt. Bei 0 wird das Gerät abgeschaltet. Jeder Interrupt der Tastatur oder der seriellen Schnittstelle setzt den Zähler wieder auf den Anfangswert. Die maximale Power-Off Zeit beträgt etwa 18 Stunden(Zählerwert von 65534). Die Funtion

SetPowerOffDelay

setzt den Anfangswert des Zählers und lädt den Zähler mit diesem Wert. Ein Anfangswert von -1 oder 65535 deaktiviert den Zähler.

GetPowerOffDelay

liefert den Anfangswert des Zählers und

GetPowerOffTimer

den aktuellen Zustand des Zählers.

5.2.13. GetBacklightTimer

Assembler Interface: Int 15h

Parameter:	AX	-	2035h
Rückgabe:	DX	-	Backlight Timer

C-Library Interface

Funktionsprototyp:	GetBacklightTimer(void)
Rückgabe:	Backlight Timer

Gibt die Zeit in Sekunden zurück, die bis zum Abschalten der LCD-Beleuchtung verbleibt.

5.2.14. GetPowerOnTimer

Assembler Interface: Int 15h

Parameter:	AX	-	2036h
Rückgabe:	DX	-	Power-On Timer

C-Library Interface

Funktionsprototyp:	unsigned int GetPowerOnTimer(void)
Rückgabe:	Power-On Timer

Gibt die Zeit in Sekunden zurück, die seit dem Einschalten vergangen ist.

5.2.15. SetBacklightOffDelay

Assembler Interface: Int 15h

Parameter:	AX	-	2037h
Rückgabe:	DX	-	BacklightOffDelay

C-Library Interface

Funktionsprototyp:	void SetBacklightOffDelay(unsigned int delay)
Rückgabe:	BacklightOffDelay

Setzt die Zeit in Sekunden, nach welcher die LCD-Hintergrundbeleuchtung abgeschaltet wird.

5.3. LCD-Funktionen

5.3.1. SetCharSize

Assembler Interface: Int 15h

Parameter:	AX	-	2003h
Rückgabe:	DX	-	size

C-Library Interface

Funktionsprototyp:	void SetCharSize(int size)
Rückgabe:	keine

Zeichen-Größe festlegen: size ist einer der folgenden Werte:

0 - 8x6
1 - 8x8
2 - 8x16

5.3.2. GetScrollMode

Assembler Interface: Int 15h

Parameter:	AX	-	200Bh
Rückgabe:	DL	-	Scrollmode

C-Library Interface

Funktionsprototyp:	unsigned char GetScrollMode(void)
Rückgabe:	Scrollmode

Darstellungsart feststellen.

- 0 : Cursor immer im sichtbaren Bereich
- 1 : Spalte fest
- 2 : Reihe Fest
- 3 : Reihe Spalte fest

5.3.3. SetScrollMode

Assembler Interface: Int 15h

Parameter:	AX	-	200Ch
	DL	-	Scrollmode
Rückgabe:	keine		

C-Library Interface

Funktionsprototyp:	void SetScrollMode(unsigned char mode)
Rückgabe:	keine

Darstellungsart festlegen.

- 0 : Cursor immer im sichtbaren Bereich
- 1 : Spalte fest
- 2 : Reihe Fest
- 3 : Reihe Spalte fest

5.3.4. GetWindowPos

Assembler Interface: Int 15h

Parameter:	AX	-	2013h
Rückgabe:	DL	-	X-Wert (linke obere Ecke)
	DH	-	Y-wert (linke obere Ecke)

C-Library Interface

Funktionsprototyp:	void GetWindowPos(unsigned char *wx, unsigned char *wy)
Rückgabe:	wx, wy: linke obere Ecke des sichtbaren Bereichs

Liefert die Position der linken oberen Ecke des LCDs im des 80x25-Zeichen umfassenden DOS-Bildschirm (s. SetScrollMode). Der Wert ist nur von Bedeutung, wenn Scrollmode ungleich 0 ist, d.h. wenn das LCD-Fenster vertikal oder horizontal (oder beides) fixiert ist.

5.3.5. SetWindowPos

Assembler Interface: Int 15h

Parameter:	AX	-	2014h
Rückgabe:	DL	-	X-Wert (linke obere Ecke)
	DH	-	Y-wert (linke obere Ecke)

C-Library Interface

Funktionsprototyp:	void SetWindowPos(unsigned char wx, unsigned char wy)
Rückgabe:	keine

Setzt die Position der linken oberen Ecke des LCDs im DOS-Bildschirm (s. SetScrollMode). Der Wert ist nur von Bedeutung, wenn Scrollmode ungleich 0 ist, d.h. wenn das LCD-Fenster vertikal oder horizontal (oder beides) fixiert ist.

5.3.6. LcdEnableUpdate

Assembler Interface: Int 15h

Parameter:	AX	-	2040h
Rückgabe:	keine		

C-Library Interface

Funktionsprototyp:	void LcdEnableUpdate(void)
Rückgabe:	keine

Bei jedem Timer-Interrupt (18 mal pro Sekunde) wird überprüft, ob der LCD-Inhalt sich geändert hat. Falls ja, wird die Änderung auf dem LCD ausgegeben.

5.3.7. LcdDisableUpdate

Assembler Interface: Int 15h

Parameter:	AX	-	2041h
Rückgabe:	keine		

C-Library Interface

Funktionsprototyp:	void LcdDisableUpdate(void)
Rückgabe:	keine

LcdDisableUpdate: Verhindert Ausgabe auf dem LCD durch die Timer-Interrupt-Routine. Da durch die LCD-Funktion des Timers relativ viel CPU-Zeit benötigt wird, kann es sinnvoll sein, die Timer-Funktion in zeitkritischen Sektionen eines Programms zu deaktivieren.

5.3.8. LcdForceUpdate

Assembler Interface: Int 72h

Parameter: keine
Rückgabe: keine

C-Library Interface

Funktionsprototyp: void LcdForceUpdate(void)
Rückgabe: keine

Der Inhalt des Bildschirmspeichers wird auf dem LCD ausgegeben. Die Ausgabe darf jedoch nicht durch LcdDisableUpdate gesperrt sein.

5.3.9. LcdSetMode

Assembler Interface: Int 10h

Parameter:	AX	-	2000h
	BL	-	mode
Rückgabe:	keine		

C-Library Interface

Funktionsprototyp: void LcdSetMode(unsigned char mode)
Rückgabe: keine

mode = 0 : Normale Ausgabe, Hintergrund wird überschrieben
mode = 1 : Oder-Verknüpfung mit Hintergrund
mode = 2 : Und-Verknüpfung mit Hintergrund
mode = 3 : Exklusiv-Oder-Verknüpfung mit Hintergrund

5.3.10. LcdSetPixel

Assembler Interface: Int 10h

Parameter:	AX	-	2001h
	BL	-	01h
	DH	-	px
	DL	-	py
Rückgabe:	keine		

C-Library Interface

Funktionsprototyp: void LcdSetPixel(unsigned char px, unsigned char py)
Rückgabe: keine

Setzt das Pixel an der Position px, py.

5.3.11. LcdClrPixel

Assembler Interface: Int 10h

Parameter:	AX	-	2001h
	BL	-	01h
	DH	-	px
	DL	-	py
Rückgabe:	keine		

C-Library Interface

Funktionsprototyp: void LcdClrPixel(unsigned char px, unsigned char py)
Rückgabe: keine

Löscht das Pixel an der Position px, py.

5.3.12. LcdDrawLine

Assembler Interface: Int 10h

Parameter:	AX	-	2001h
	DL	-	x1
	DH	-	y1
	CL	-	x2
	CH	-	y2
Rückgabe:	keine		

C-Library Interface

Funktionsprototyp:

```
void LcdDrawLine (unsigned char x1, unsigned char y1,
                  unsigned char x2, unsigned char y2)
```

Rückgabe: keine

Zeichnet eine Linie von Punkt (x1, y1) nach (x2, y2)

5.3.13. LcdDrawPLine

Assembler Interface: Int 10h

Parameter:	AX	-	2003h
	BX	-	pattern
	DL	-	x1
	DH	-	y1
	CL	-	x2
	CH	-	y2
Rückgabe:	keine		

C-Library Interface

Funktionsprototyp:

```
void LcdDrawPLine (unsigned char x1, unsigned char y1,
                   unsigned char x2, unsigned char y2, unsigned int pattern)
```

Rückgabe: keine

Zeichnet eine Linie mit dem Bitmuster pattern von Punkt (x1, y1) nach (x2, y2).

5.3.14. LcdDrawBox

Assembler Interface: Int 10h

Parameter:	AX	-	2004h
	DL	-	x1
	DH	-	y1
	CL	-	x2
	CH	-	y2
Rückgabe:	keine		

C-Library Interface

Funktionsprototyp: void LcdDrawBox(unsigned char x1, unsigned char y1,
unsigned char x2, unsigned char y2)

Rückgabe: keine

Kasten zeichnen. (x1, y1) links oben, (x2, y2) rechts unten.

5.3.15. LcdFill

Assembler Interface: Int 10h

Parameter:	AX	-	2005
	BX	-	pattern
	DL	-	x1
	DH	-	y1
	CL	-	x2
	CH	-	y2
Rückgabe:	keine		

C-Library Interface

Funktionsprototyp: void LcdFill (unsigned char x1, unsigned char y1,
unsigned char x2, unsigned char y2,
unsigned int pattern)

Rückgabe: keine

Gefülltes Quadrat ohne Umrandung zeichnen (x1, y1) links oben, (x2, y2) rechts unten.
Pattern gibt eines von 16 Mustern an (0 - 15).

5.3.16. LcdFBox

Assembler Interface: Int 10h

Parameter:	AX	-	2005
	BX	-	pattern
	DL	-	x1
	DH	-	y1
	CL	-	x2
	CH	-	y2
Rückgabe:	keine		

C-Library Interface

Funktionsprototyp: void LcdFBox (unsigned char x1, unsigned char y1,
unsigned char x2, unsigned char y2,
unsigned int pattern)

Rückgabe: keine

Gefüllten Kasten zeichnen.
(x1, y1) links oben, (x2, y2) rechts unten. Pattern gibt eins von 16 Mustern an (0 - 15).

6. Funktions-Bibliothek des V40-MoDOS

Die Programmierung des V40-MoDOS entspricht im wesentlichen der der V40-Card, die den CPU-Kern des MODOS bildet (siehe Handbuch "V40-Card"). Für die Ansteuerung von LCD und Matrix-Tastatur gibt es eine eigene Programm-Bibliothek. Die Funktionen sind als BIOS-Erweiterung implementiert. Sie werden über den Interrupt 78h erreicht.

6.1. Beschreibung der Systemfunktionen

6.1.1. ModosInit

Beim Start einer Applikation sollte diese Funktion aufgerufen werden, um das LCD-Display und die Tastatur in einen definierten Zustand zu versetzen.

Assembler Interface

Parameter:	AH	-	00h
Rückgabe:	keine		

C-Library Interface

Funktionsprototyp:	int ModosInit(void);
Parameter:	keine
Rückgabe:	keine

6.1.2. PowerOff

Mit dieser Funktion wird das Gerät ausgeschaltet.

Assembler Interface

Parameter:	AH	-	20h
Rückgabe:	keine		

C-Library Interface

Funktionsprototyp:	void PowerOff(void);
Parameter:	keine
Rückgabe:	keine

6.1.3. PifPowerOff

Mit dieser Funktion wird die Spannungsversorgung des internen PIF-Bus abgeschaltet. Hierdurch kann Strom gespart werden, da zusätzliche Hardware nur bei Bedarf eingeschaltet werden kann.

Assembler Interface

Parameter:	AH	-	21h
Rückgabe:	keine		

C-Library Interface

Funktionsprototyp:	void PifPowerOff(void);
Parameter:	keine
Rückgabe:	kein

6.1.4. PifPowerOn

Hiermit wird die Spannungsversorgung für den internen PIF-Bus eingeschaltet. Gleichzeitig wird ein Reset am PIF-Bus ausgelöst, so daß eventuell angeschlossene Hardware sich neu initialisieren muß.

Assembler Interface

Parameter:	AH	-	22h
Rückgabe:	keine		

C-Library Interface

Funktionsprototyp:	void PifPowerOn(void);
Parameter:	keine
Rückgabe:	keine

6.1.5. Beep

Mit dieser Funktion können kurze Tonfolgen am internen Lautsprecher ausgegeben werden.

Assembler Interface

Parameter:	AH	-	50h
	AL	-	Anzahl der Piepse
Rückgabe:	keine		

C-Library Interface

Funktionsprototyp:	void Beep(int count);		
Parameter:	count	-	Anzahl der Tönen (< 255)
Rückgabe:	keine		

6.2. Tastatur

6.2.1. KbdGetch

Mit dieser Funktion wird der Tastaturpuffer abgefragt. Befindet sich kein Zeichen im Tastaturpuffer, so wird das Gerät in den Stopmodus versetzt, in dem es deutlich weniger Strom verbraucht. Sobald eine Taste gedrückt wird, kehrt diese Funktion mit dem Tastenwert zum aufrufenden Programm zurück. Die internen Interrupts der V40-Card werden abgeschaltet, und nur externe Interrupts sind möglich. Um dieses Verhalten zu ändern, kann man sich in den Software-Interrupt 0x7A einhängen, der es ermöglicht, das Eintreten in den Stopmodos zu verhindern (siehe Auschaltverhalten).
Diese Funktion verhält sich ähnlich der Funktion getch() in C, d.h. falls 0x00 zurückgegeben wird, handelt es sich um eine Sondertaste. Den eigentlichen Scancode erfährt man dann über einen erneuten Aufruf von KbdGetch. Die zurückgelieferten Werte orientieren sich an der Tastatur eines PC.

Assembler Interface

Aufruf*Parameter:*	AH	-	10h
Rückgabe:	AL	-	Zeichen aus Tastaturpuffer

C-Library Interface

Funktionsprototyp:	int KbdGetch(void);
Parameter:	keine
Rückgabe:	Zeichen aus dem Tastaturpuffer. Falls dieses 0x00 ist, so steht der Scancode nach einem erneuten Aufruf von KbdGetch() bereit.

6.2.2. KbdKbhit

Mit dieser Funktion kann geprüft werden, ob noch Zeichen im Tastaturpuffer vorhanden sind. Falls noch Zeichen vorhanden sind, liefert diese Funktion einen Wert ungleich Null, ansonsten NULL.

Assembler Interface

Parameter:	AH	-	11h
Rückgabe:	AL	-	00h : Kein Zeichen in Tastaturpuffer
			01h : Zeichen im Tastaturpuffer vorhanden

C-Library Interface

Funktionsprototyp:	int KbdKbhit(void);		
Parameter:	keine		
Rückgabe:	0	-	kein Zeichen im Tastaturpuffer
	1	-	Zeichen im Tastaturpuffer vorhanden

6.2.3. KbdFlush

Mit dieser Funktion kann der interne Tastaturpuffer gelöscht werden. Eventuell im Puffer liegende Zeichen gehen verloren.

Assembler Interface

Parameter:	AH	-	12h
Rückgabe:	keine		

C-Library Interface

Funktionsprototyp:	void KbdFlush(void);
Parameter:	keine
Rückgabe:	keine

6.2.4. KbdPeek

Mit diese kann das nächste Zeichen im Tastaturpuffer erfragt werden, ohne es aus dem Puffer zu entfernen.

Assembler Interface

Parameter:	AH	-	13h
Rückgabe:	AL	-	Erstes Zeichen im Tastaturpuffer NULL falls kein Zeichen vorhanden.

C-Library Interface

Funktionsprototyp:	int KbdPeek(void);
Parameter:	keine
Rückgabe:	nächstes Zeichen im Tastaturpuffer, 0x00 falls kein Zeichen vorhanden.

6.3. LCD - Allgemein

6.3.1. LcdClear

Diese Funktion löscht die gesamte LCD-Anzeige.

Assembler Interface

Parameter:	AH	-	38h
Rückgabe:	keine		

C-Library Interface

Funktionsprototyp:	void LcdClear(void);
Parameter:	keine
Rückgabe:	keine

6.3.2. LcdScrollUp

Mit dieser Funktion wird der Inhalt des LCD-Displays um 8 Pixel nach oben verschoben.

Assembler Interface

Parameter: AH - 3Ch
Rückgabe: keine

C-Library Interface

Funktionsprototyp: void LcdScrollUp(void);
Parameter: keine
Rückgabe: keine

6.3.3. LcdScrollDown

Mit dieser Funktion wird der Inhalt des LCD-Displays um 8 Pixel nach unten verschoben.

Assembler Interface

Parameter: AH - 3Dh
Rückgabe: keine

C-Library Interface

Funktionsprototyp: void LcdScrollDown(void);
Parameter: keine
Rückgabe: keine

6.3.4. LcdBackLight

Mit deser Funktion kann die Helligkeit des LCD-Displays in vier Stufen eingestellt werden. Je heller das Display, umso mehr Strom wird verbraucht.

Assembler Interface

Parameter: AH - 25h
AL - 0 - 3 Helligkeit der LED Hintergrungbeleuchtung
Rückgabe: keine

C-Library Interface

Funktionsprototyp: void LcdBackLight(int led);
Parameter: led - 0 - 3 Helligkeit der LED Hintergrundbeleuchtung
Rückgabe: keine

6.3.5. LcdGetScreen

Mit dieser Funktion kann ein Teil des LCD-Displays ausgelesen werden und in einem Puffer gespeichert werden. Hierzu muß der Anwender einen Puffer mit geeigneter Größe zur Verfügung gestellt werden. Die Puffergröße berechnet sich aus (Größe in Bytes = 16 + (Anzahl der Zeilen / 8) + Anzahl der Spalten.
Es können nur Zeilen, die ein Vielfaches von 8 sind, angegeben werden.

Assembler Interface

Parameter:	AH	-	47h
	CH	-	Zeile links oben
	CL	-	Spalte links oben
	DH	-	Zeile rechts unten
	DL	-	Spalte rechts unten
	ES:DI	-	Segment:Offset vom Puffer
Rückgabe:	keine		

C-Library Interface

Funktionsprototyp:

```
void LcdGetScreen( unsigned char _far *buffer, int x1, int y1, int x2, int y2 );
```

Parameter:	buffer	-	far Pointer auf Puffer
	x1,y1	-	Koordinate links oben
	x2,y2	-	Koordinate rechts unten
Rückgabe:	keine		

6.3.6. LcdPutScreen

Ein zuvor mit GetScreen eingelesener Puffer kann mit dieser Funktion wieder auf das Display an eine neue Position geschrieben werden. Es können nur Zeilen angegeben werden, die ein Vielfaches von 8 sind.

Assembler Interface

Parameter:	AH	-	48h
	DH	-	Zeile
	DL	-	Spalte
	ES:DI	-	Segment:Offset von Puffer
Rückgabe:	keine		

C-Library Interface

Funktionsprototyp:	void LcdPutScreen(unsigned char _far *b, int x int y);		
Parameter:	x,y	-	Koordinate auf dem Display
Rückgabe:			

6.4. LCD - Textanzeige

6.4.1. LcdPutc

Diese Funktion gibt ein Zeichen an der aktuellen Position und im aktuellen Zeichensatz aus. Steuerzeichen wie CR, LF, BS usw. werden ausgewertet. Die aktuelle Position wird hiernach auf den neuesten Stand gebracht.

Assembler Interface

Parameter:	AH	-	32h
	AL	-	Zeichen
Rückgabe:	keine		

C-Library Interface

Funktionsprototyp:	void LcdPutc(int ch);		
Parameter:	ch	-	Zeichen
Rückgabe:	keine		

6.4.2. LcdPutChar

Mit diese Funktion wird ein Zeichen an der angegebenen Position ausgegeben. Steuerzeichen werden nicht interpretiert, und die Cursor Position bleibt unverändert.

Assembler Interface

Parameter:	AH	-	31h
	AL	-	Zeichen
	DH	-	Zeile
	DL	-	Spalte
Rückgabe:	keine		

C-Library Interface

Funktionsprototyp:	void LcdPutChar(int y, int x, int ch);		
Parameter:	x,y	-	Position des Zeichens
	ch	-	Zeichen
Rückgabe:	keine		

6.4.3. LcdInvertChar

Mit dieser Funktion wird das Zeichen an der angegebenen Position invertiert.

Assembler Interface

Parameter:	AH	-	30h
	DH	-	Zeile
	DL	-	Spalte
Rückgabe:	keine		

C-Library Interface

Funktionsprototyp:	void LcdInvertChar(int y, int x);		
Parameter:	x,y	-	Position des Zeichens
Rückgabe:	keine		

6.4.4. LcdInverseOn

Mit dieser Funktion wird die invertierte Darstellung eingeschaltet. Alle folgenden Ausgaben werden invertiert dargestellt.

Assembler Interface

Parameter: AH - 34h
Rückgabe: keine

C-Library Interface

Funktionsprototyp: void LcdInverseOn(void);
Parameter: keine
Rückgabe: keine

6.4.5. LcdInverseOff

Mit dieser Funktion wird zurück in die Normaldarstellung geschaltet.

Assembler Interface

Parameter: AH - 33h
Rückgabe: keine

C-Library Interface

Funktionsprototyp: void LcdInverseOff(void);
Parameter: keine
Rückgabe: keine

6.4.6. LcdShowCursor

Mit dieser Funktion wird der Display Cursor eingeschaltet.

Assembler Interface

Parameter: AH - 36h
Rückgabe: keine

C-Library Interface

Funktionsprototyp: void LcdShowCursor(void);
Parameter: keine
Rückgabe: keine

6.4.7. LcdHideCursor

Mit dieser Funktion wird der Display Cursor ausgechaltet.

Assembler Interface

Parameter: AH - 35h
Rückgabe: keine

C-Library Interface

Funktionsprototyp:	void LcdHideCursor(void);
Parameter:	keine
Rückgabe:	keine

6.4.8. LcdBlinkCursorOn

Hiermit kann der Display Cursor in einen Blinkmodus versetzt werden.

Assembler Interface

Parameter:	AH	-	26h
Rückgabe:	keine		

C-Library Interface

Funktionsprototyp:	void LcdBlinkOn(void);
Parameter:	keine
Rückgabe:	keine

6.4.9. LcdBlinkCursorOff

Mit dieser Funktion wird das Blinken des Cursors ausgeschaltet.

Assembler Interface

Parameter:	AH	-	27h
Rückgabe:	keine		

C-Library Interface

Funktionsprototyp:	void LcdBlinkOff(void);
Parameter:	keine
Rückgabe:	keine

6.4.10. LcdSetCursorSize

Mit dieser Funktion kann die Größe des Display Cursors festgelegt werden. Gültig sind Werte zwischen 0 (Strich) und 7 (Block).

Assembler Interface

Parameter:	AH	-	28h
	AL	-	0 - 7 Größe des Display Cursors
Rückgabe:	keine		

C-Library Interface

Funktionsprototyp:	void LcdSetCursorSize(int size);		
Parameter:	size	-	0 - 7 Größe des Display Cursors
Rückgabe:	keine		

6.4.11. LcdGotoYX

Mit dieser Funktion kann die aktuelle Cursor Position festgelegt werden.

Assembler Interface

Parameter:	AH	-	31h
	DH	-	Zeile
	DL	-	Spalte
Rückgabe:	keine		

C-Library Interface

Funktionsprototyp:	void LcdGotoYX(int y, int x);		
Parameter:	y, x	-	Zeile, Spalte
Rückgabe:	keine		

6.4.12. LcdGetPosition

Mit dieser Funktion kann die aktuelle Cursor Position erfragt werden.

Assembler Interface

Parameter:	AH	-	3Bh
Rückgabe:	DH	-	aktuelle Zeile
	DL	-	aktuelle Spalte

C-Library Interface

Funktionsprototyp:	void LcdGetPos(int *y, int * x);		
Parameter:	y,x	-	Pointer auf integer
Rückgabe:	y,x enthalten nach dem Aufruf die aktuelle Cursor Position.		

6.4.13. LcdSetSize

Mit dieser Funktion kann die Größe der ausgegebenen Zeichen festgelegt werden. Alle darauf folgenden Ausgaben erfolgen mit diesr Zeichensatzgröße. Es stehen drei Zeichengrößen zur Verfügung : 0 - 8x6, 1 - 8x8, 2 - 16x8.

Assembler Interface

Parameter:	AH	-	3Ah
	AL	-	0 : 8x6 Zeichensatz
			1 : 8x8 Zeichensatz
			2 : 16x8 Zeichensatz
Rückgabe:	keine		

C-Library Interface

Funktionsprototyp:	void LcdSetSize(int size);		
Parameter:	size	-	0 : 8x6 Zeichensatz
			1 : 8x8 Zeichensatz
			2 : 16x8 Zeichensatz
Rückgabe:	keine		

6.4.14. LcdSetFont

Mit dieser Funktion können die internen Zeichensätze durch einen eigenen ersetzt werden. Hierzu muß ein Zeiger auf den neuen Zeichensatz übergeben werden. Zurückgeliefert wird ein Zeiger auf den bisherigen Zeichensatz.

Assembler Interface

Parameter:	AH	-	39h
	ES:DI	-	Segment:Offset von Zeichensatz
Rückgabe:	ES:DI	-	Segment:Offset von altem Zeichensatz

C-Library Interface

Funktionsprototyp:	unsigned char _far *LcdSetFont(unsigned char _far * font);
Parameter:	font - far Pointer auf neuen Zeichensatz
Rückgabe:	far Pointer auf alten Zeichensatz.

6.4.15. LcdWrapOn

Diese Funktion bewirkt, daß Ausgaben, die am Zeilenende beginnen, auf der nächsten Zeile fortgesetzt werden.

Assembler Interface

Parameter:	AH	-	3Eh
Rückgabe:	keine		

C-Library Interface

Funktionsprototyp:	void LcdWrapOn(void);
Parameter:	keine
Rückgabe:	keine

6.4.16. LcdWrapOff

Mit dieser Funktion kann verhindert werden, daß Ausgaben, die am Zeilenende erfolgen, auf der nächsten Zeile fortgesetzt werden.

Assembler Interface

Parameter:	AH	-	3Fh
Rückgabe:	keine		

C-Library Interface

Funktionsprototyp:	void LcdWrapOff(void);
Parameter:	keine
Rückgabe:	keine

6.5. LCD - Grafik

6.5.1. LcdGChar

Mit dieser Funktion kann ein Zeichen an einer beliebigen Pixel Position ausgegeben werden.

Assembler Interface

Parameter:	AH	-	40h
	AL	-	Zeichen
	BL	-	00h : Überschreiben
			01h : Löschen
			02h : Invertieren
	DH	-	Zeile
	DL	-	Spalte
Rückgabe:	keine		

C-Library Interface

Funktionsprototyp:	void LcdGChar(int x, int y, int ch, int m);		
Parameter:	x,y	-	Position
	ch	-	Zeichen
	m	-	0 : Überschreiben
			1 : Löschen
			2 : Invertieren
Rückgabe:	keine		

6.5.2. LcdBox

Mit dieser Funktion wird ein leeres Rechteck gezeichnet.

Assembler Interface

Parameter:	AH	-	41h
	CH	-	Zeile links oben
	CL	-	Spalte links oben
	DH	-	Zeile rechts unten
	DL	-	Spalte rechts unten
Rückgabe:	keine		

C-Library Interface

Funktionsprototyp:	void LcdBox(int x1, int y1, int x2, int y2);		
Parameter:	x1, y1	-	Ecke links oben
	x2, y2	-	Ecke rechts unten
Rückgabe:	keine		

6.5.3. LcdFBox

Mit dieser Funktion wird ein mit einem Muster gefülltes Rechteck gezeichnet.

Assembler Interface

Parameter:	AH	-	42h
	BH	-	0-15 : Muster
	CH	-	Zeile links oben
	CL	-	Spalte links oben
	DH	-	Zeile rechts unten
	DL	-	Spalte rechts unten
Rückgabe:	keine		

C-Library Interface

Funktionsprototyp:	void LcdFBox(int x1, int y1, int x2, int y2, int p);		
Parameter:	x1, y1	-	Ecke links oben
	x2, y2	-	Ecke rechts unten
	p	-	Muster (0 - 15)
Rückgabe:	keine		

6.5.4. LcdPLine

Mit dieser Funktion wird eine Linie mit einem Muster gezeichnet. Das Muster wird als Bitmuster angegeben.

Assembler Interface

Parameter:	AH	-	43h
	BH	-	Bitmuster
	BL	-	00h : Überschreiben
			01h : Löschen
			02h : Invertieren
	CH	-	Zeile Start
	CL	-	Spalte Start
	DH	-	Zeile Ende
	DL	-	Spalte Ende
Rückgabe:	keine		

C-Library Interface

Funktionsprototyp:	void LcdPLine(int x1, int y1, int x2, int y2, int p, int m);		
Parameter:	x1, y1	-	Ecke links oben
	x2, y2	-	Ecke rechts unten
	p	-	Bitmuster
	m	-	0 : Überschreiben
			1 : Löschen
			2 : Invertieren
Rückgabe:	keine		

6.5.5. LcdLine

Mit dieser Funktion wird eine Linie auf dem LCD Display gezeichnet.

Assembler Interface

Parameter:	AH	-	44h
	BL	-	00h : Überschreiben
			01h : Löschen
			02h : Invertieren
	CH	-	Zeile Start
	CL	-	Spalte Start
	DH	-	Zeile Ende
	DL	-	Spalte Ende
Rückgabe:	keine		

C-Library Interface

Funktionsprototyp:	void LcdLine(int x1, int y1, int x2, int y2, int m);		
Parameter:	x1, y1	-	Ecke links oben
	x2, y2	-	Ecke rechts unten
	m	-	0 : Überschreiben
			1 : Löschen
			2 : Invertieren
Rückgabe:	keine		

6.5.6. LcdPixel

Hiermit kann ein Pixel auf dem LCD-Display manipuliert werden.

Assembler Interface

Parameter:	AH	-	45h
	BL	-	00h : Überschreiben
			01h : Löschen
			02h : Invertieren
	DH	-	Zeile
	DL	-	Spalte
Rückgabe:	keine		

C-Library Interface

Funktionsprototyp:	void LcdPixel(int x , int y , int m);		
Parameter:	x,y	-	Position
	m	-	0 : Überschreiben
			1 : Löschen
			2 : Invertieren
Rückgabe:	keine		

6.5.7. LcdFill

Mit dieser Funktion wird ein rechteckiger Bereich mit einem Muster gefüllt.

Assembler Interface

Parameter:	AH	-	46h
	BH	-	0-15 : Muster
	CH	-	Zeile links oben
	CL	-	Spalte links oben
	DH	-	Zeile rechts unten
	DL	-	Spalte rechts unten
Rückgabe:	keine		

C-Library Interface

Funktionsprototyp:

```
void LcdFill( int x1, int y1, int x2, int y2, int m );
```

Parameter:	x1, y1	-	Ecke links oben
	x2, y2	-	Ecke rechts unten
	p	-	Muster (0 - 15)
Rückgabe:	keine		

6.6. Ausschaltverhalten

Da der Anwender das Gerät zu jedem beliebigen Zeitpunkt ausschalten kann, besteht die Gefahr, das Gerät in einem undefinierten Zustand zu hinterlassen. So können z.B. unvollendete Schreibzugriffe auf die Laufwerke das Dateisystem zerstören. Daher besitzt das Gerät die Möglichkeit, ein Ausschalten zu verhindern und so alle notwendigen Operationen zu vollenden. Zusätzlich könnte vor dem Abschalten der Programmzustand abgespeichert werden, damit das Anwenderprogramm nach dem Einschalten wieder neu aufsetzen kann. Um dies zu ermöglichen, ruft das BIOS einen definierten Software Interrupt auf, der im Anwenderprogramm abgefangen werden kann. In der Interruptfunktion kann nun dem BIOS mitgeteilt werden, ob es ausschalten kann oder nicht. So kann man in kritischen Sektionen verhindern, daß das Gerät ausgeschaltet wird. Gleiches gilt für den Eintritt in den Stop Modus, nur daß hier der Interrupt 0x7A und nicht der Interrupt 0x79 verwendet wird. Diese Möglichkeit ist im folgenden Programm beispielhaft dargestellt.

```
/*****                          Header - Files                              *****/

#include <stdlib.h>
#include <stdlib.h>
#include <dos.h>
#include "modos.h"

/*****                               Makros                                 *****/
/*****                         Typ Definitionen                             *****/
/*****                        Externe Prototypes                            *****/
/*****                        Externe Variablen                             *****/
/*****                         Public Prototypes                            *****/
/*****                         Static Prototypes                            *****/
/*****                         Public Variablen                             *****/

void _far *old_vec;

/*****                         Static Variablen                             *****/

static int  power_cnt  = 0;
static int  power_flag = 0;

/*****                            Public Code                               *****/

/*****************************************************************************

DisablePowerOff

Diese Funktion markiert den Anfang einen kritischen Sektion. Zusammen mit
DisablePowerOff wird ein Bereich markiert, in dem das Gerät nicht
ausgeschaltet werden darf. Das eventuelle Ausschalten wird bis EnablePowerOff
verzögert.

   DisablePowerOff();
        .
        .
   Kritische Sektion, z.B. Datei- und DOS Funktionen
        .
        .
   EnablePowerOff();

*****************************************************************************/

void DisablePowerOff( void )
{
   /* Der Zähler wird um eins erhöht, um anzuzeigen, daß das Gerät */
   /* nicht ausgeschaltet werden darf.                             */
   power_cnt++;
}
```

```
/****************************************************************************

EnablePowerOff

Diese Funktion markiert das Ende einer kritischen Sektion. Wurde in der
kritischen Sektion <SHIFT-START> betätigt, so wird das Gerät hier
ausgeschaltet.

****************************************************************************/

void EnablePowerOff( void )
{
   /* Eine Kritische Sektion wurde beendet; falls sich der Zähler bei */
   /* Null befindet, kann das Gerät ausgeschaltet werden              */
   if( power_cnt != 0 ) power_cnt--;

   /* Falls innerhalb einer kritischen Sektion ausgeschaltet werden */
   /* sollte, wird hier das Gerät nachträglich ausgeschaltet        */
   if( ( power_cnt == 0 ) && ( power_flag == 1 ) ) PowerOff( );
}

/****************************************************************************

PowerInterrupt

Diese Funktion muß in den Interrupt 0x79 eingehängt werden. Sie wird
jedesmal aufgerufen, wenn das Gerät ausgeschaltet werden soll. In dieser
Funktion sollten keine DOS Aufrufe gemacht werden, da sich DOS in einem
nicht bekannten Zustand befindet.

****************************************************************************/

void __interrupt __far PowerInterrupt( unsigned _es, unsigned _ds,
                                       unsigned _di, unsigned _si,
                                       unsigned _bp, unsigned _sp,
                                       unsigned _bx, unsigned _dx,
                                       unsigned _cx, unsigned _ax,
                                       unsigned _ip, unsigned _cs,
                                       unsigned flags )

{
   /* Falls der Zähler ungleich null ist, so befindet sich das */
   /* Gerät in einer Kritischen Sektion und darf nicht         */
   /* ausgeschaltet werden.                                    */

   if( power_cnt != 0x00 )
   {
      /* Flag setzen, damit das Gerät nach Verlassen einer     */
      /* kritischen Sektion auszuschalten ist                  */
      power_flag = 1;

      /* Warnton ausgeben                                      */
      Beep( 1 );

      /* BIOS durch Setzen von AX benachrichtigen, daß nicht   */
      /* ausgeschaltet werden darf. Falls ausgeschaltet werden */
      /* kann, wird AX unverändert gelassen                    */
      _ax = 0x1234;
   }
}



/****************************************************************************

main

Beispiel zur Anwendung von DisablePowerOff, EnablePowerOff und
PowerInterrupt.

****************************************************************************/
```

```
void main( void )
{
   int key;

   /* Callback Interrupt Vektor auf eigene Funktion setzen */
   old_vec = ( void _far *)_dos_getvect( 0x79 );
   _dos_setvect( 0x79, ( void _far * ) PowerInterrupt );


   if( ModosInit() )
   {
      LcdPuts( " Poweroff - Demo    "
               "Das Auschalten des  "
               "Geräts ist erst nach"
               "der Eingabe von ESC "
               "möglich.\n" );

      /* Anfang einer kritischen Sektion */
      DisablePowerOff();

      do
      {
          if( ( key = KbdGetch() ) == 0 ) KbdGetch();
          LcdPutc( key );
      }
      while( key != 0x1B );

      /* Ende einer kritischen Sektion. Wurde Zwischenzeitlich      */
      /* SHIFT-START gedrückt so wird das Gerät jetzt ausgeschaltet */
      EnablePowerOff();

      /* Falls man bis hierher gelangt, wurde kein SHIFT ENTER      */
      /* gedrückt. Alle Vektoren zurücksetzen.                      */
      _dos_setvect( 0x79, old_vec );
   }
}

/*********************  E n d   o f   F i l e  ***************************/
```

7. Steckerbelegung

7.1. User I/O - DSUB 25

Pin	Verwendung	Pin	Verwendung
1	USER VCC	14	BATT+
2	-ONEXT	15	RxD
3	TxD	16	-CTS
4	-RTS	17	BATT-
5	IN0 / IRQ6	18	GND
6	IN1	19	OUT0
7	IN2	20	OUT1
8	IN3	21	OUT2
9	IN4	22	OUT3
10	IN5	23	OUT4
11	IN6	24	OUT5
12	IN7	25	OUT6
13	OUT7		

7.2. User I/O - 26 polige Wanne (intern)

Pin	Verwendung	Pin	Verwendung
1	USER VCC	2	BATT+
3	-ONEXT	4	RxD
5	TxD	6	-CTS
7	-RTS	8	BATT-
9	GND	10	GND
11	IN0 / IRQ6	12	OUT0
13	IN1	14	OUT1
15	IN2	16	OUT2
17	IN3	18	OUT3
19	IN4	20	OUT4
21	IN5	22	OUT5
23	IN6	24	OUT6
25	IN7	26	OUT7

Bedeutung der Signale :

IN0 - IN7 : TTL Eingänge
IN0 ist gleichzeitig auch IRQ6
OUT0-OUT7: TTL Ausgänge
BATT+, BATT- : Batterie Anschlüsse
VCC, GND: Betriebsspannungs Ausgang 5V
TxD,RxD,CTS,RTS: TLL Signale der internen seriellen Schnittstelle

7.3. I/O Ports

Das MoDOS besitzt jeweils 8 digitale Eingänge und Ausgänge, die extern über den 25-poligen DSUB-Stecker zugänglich sind. Sie können über die folgenden I/O-Adressen angesprochen werden:

MoDOS mit V40-Card: **7Bh**
MoDOS mit 386EX-Card: **32Bh**

8. *Interrupt-Tabelle für MoDOS mit V40-Card*

Vektor	Adresse	Verwendung	Typ
00	0000	Divide by Zero	CPU
01	0004	Single Step	CPU
02	0008	NMI	CPU
03	000C	Breakpoint	CPU
04	0010	Overflow	CPU
05	0014	CHKIND	CPU
06 - 07	0018-001C	reserved	
08	0020	Timer 0 (System Timer)	PIC - IRQ0
09	0024	COM1	PIC - IRQ1
0A	0028	Timer 2 (Beeper)	PIC - IRQ2
0B	002C	PIF	PIC - IRQ3
0C	0030	Real time Clock (RTC)	PIC - IRQ4
0D	0034	Startkey	PIC - IRQ5
0E	0038	IN-0 User I/O Stecker	PIC - IRQ6
0F	003C	Matrix-Keyboard	PIC - IRQ7
10	0040	Video Interrupt	BIOS
11	0044	Equipment Check	BIOS
12	0048	Memory size	BIOS
13	004C	Disk	BIOS
14	0050	COM	BIOS
15	0054	reserved	BIOS
16	0058	Keyboard	BIOS
17	005C	LPT	BIOS
18	0060	Boot failure	BIOS
19	0064	Boot loader	BIOS
1A	0068	Time of Day	BIOS
1B	006C	reserved	BIOS
1C	0070	User Timer Tick (von Int 08 aufgerufen)	BIOS
1D	0074	reserved	BIOS
1E	0078	Disk Par. Table	BIOS
1F	007C	reserved	BIOS
20 - 3F	0080 - 00FC	reserved for DOS	DOS
40 - 61	0100 - 0184	reserved for BIOS	BIOS
62	188	reserved	BIOS
63-77	18C-1DC	Unused	USER
78	1E0	MoDOS-Funktionen	MoDOS
79-FF	1E4-3FC	Unused	USER

9. *Interrupt-Tabelle für MoDOS mit 386EX-Card*

Vektor	Adresse	Verwendung	Typ
00	0000	Divide by Zero	CPU
01	0004	Single Step	CPU
02	0008	NMI	CPU
03	000C	Breakpoint	CPU
04	0010	Overflow	CPU
05	0014	CHKIND	CPU
06 - 07	0018-001C	reserved	
08	0020	Timer 0 (System Timer)	PIC - IRQ0
09	0024	PIF	PIC - IRQ1
0A	0028	2. Interrupt Controller	PIC - IRQ2
0B	002C	COM2	PIC - IRQ3
0C	0030	COM1	PIC - IRQ4
0D	0034	Startkey	PIC - IRQ5
0E	0038	IN-0 User I/O Stecker	PIC - IRQ6
0F	003C	Matrix-Keyboard	PIC - IRQ7
10	0040	Video Interrupt	BIOS
11	0044	Equipment Check	BIOS
12	0048	Memory size	BIOS
13	004C	Disk	BIOS
14	0050	COM	BIOS
15	0054	reserved	BIOS
16	0058	Keyboard	BIOS
17	005C	LPT	BIOS
18	0060	Boot failure	BIOS
19	0064	Boot loader	BIOS
1A	0068	Time of Day	BIOS
1B	006C	reserved	BIOS
1C	0070	User Timer Tick	BIOS
1D	0074	reserved	BIOS
1E	0078	Disk Par. Table	BIOS
1F	007C	reserved	BIOS
20 - 3F	0080 - 00FC	reserved for DOS	DOS
40 - 61	0100 - 0184	reserved for BIOS	BIOS
62	188	reserved	BIOS
63-6F	18C-1BC	Unused	USER
70	1C0	Real time Clock (RTC)	PIC2 - IRQ8
71	1C4	I/O (unassigned)	PIC2- IRQ9
72	1C8	Timer 1 (MODOS-Funkt.)	PIC2 - IRQ10
73	1CC	Timer 2 (Beeper)	PIC2 - IRQ11
74	1D0	DMA	PIC2 - IRQ12
75	1D4	I/O (unassigned)	PIC2 - IRQ13
76	1D8	IDE (Compact-Flash)	PIC2 - IRQ14
77	1DC	reserved	PIC2 - IRQ15
78-FF	1E0-3FC	Unused	USER