Einige wichtige BIOS-Interrupts

Wolfgang Kippels

31. Juli 2010

Inhaltsverzeichnis

1 Vorwort

Diese Interrupts werden für das Arbeiten unter Assembler im Real-Modus benötigt. Hier findet man den zugehörigen Assembler-Lehrgang:
http://members.dokom.net/w.kippels/ass/as_lehrg.pdf

Die Liste dieser Interrupts und der zugehörigen Funktionen ist nicht vollständig. Ich habe nur die (meiner Meinung nach) wichtigsten Funktionen zusammengetragen.

2 Interrupt 10h: Bildschirmzugriffe

2.1 Funktion 00h: Videomodus setzen

Eingabe: **AH**=00h
AL=Videomodus

Ausgabe: keine

Unterstützte Videomodi:

Modus	Text/Grafik	Auflösung	Farben
00h	Text	40x25	2
01h	Text	40x25	16
02h	Text	80x25	2
03h	Text	80x25	16
04h	Grafik	320x200	2
05h	Grafik	320x200	4
06h	Grafik	640x200	2
0Dh	Grafik	320x200	16
0Eh	Grafik	640x200	16
0Fh	Grafik	640x350	2
10h	Grafik	640x350	16
11h	Grafik	640x480	2
12h	Grafik	640x480	16
13h	Grafik	320x200	256

2.2 Funktion 01h: Cursor-Größe definieren

Eingabe: **AH**=01h
CH=Startzeile des Cursors
CL=Endzeile des Cursors
sinnvolle Werte zwischen 00h und 0Dh,
Startzeile < Endzeile

Ausgabe: keine

2.3 Funktion 02h: Cursor setzen

Eingabe: **AH**=02h
BH=Bildschirmseite (meist 0)
DH=Zeile
DL=Spalte

Ausgabe: keine

2.4 Funktion 03h: Cursor lesen

Eingabe: **AH**=03h
BH=Nummer der Bildschirmseite
CH=Cursor-Anfangszeile
CL=Cursor-Endzeile

Ausgabe: **DH**=Zeile
DL=Spalte

2.5 Funktion 04h: Lichtgriffel lesen

Eingabe: **AH**=04h

Ausgabe: **AH**=Erfolgskennzeichen (siehe unten)
DH=Textzeile des Lichtgriffels
DL=Textspalte des Lichtgriffels
CH=Grafikzeile des Lichtgriffels
BX=Grafikspalte des Lichtgriffels

- Das **AH**-Register zeigt als Flag an, ob der Leseversuch erfolgreich war (das heißt, ob ein Lichtgriffel angeschlossen ist.) Bedeutungen siehe untenstehende Tabelle.
- Die Funktion ermittelt sowohl die aktuellen Positionswerte im Grafik- als auch im Textmodus.

AH	**Bedeutung:**
00h	Lichtgrifel ist **nicht** angeschlossen
01h	Lichtgrifel ist angeschlossen

2.6 Funktion 05h: Bildschirmseite wählen

Eingabe: **AH**=05h
AL=Nummer der Bildschirmseite

Ausgabe: keine

2.7 Funktion 06h: Aufwärts scrollen innerhalb eines Fensters

Eingabe: **AH**=06h
AL=Anzahl der Zeilen
(Bei AL=0 wird das Fenster gelöscht)
BH=Attribut für die neuen Zeilen
CH=Zeile Fenster links oben
CL=Spalte Fenster links oben
DH=Zeile Fenster rechts unten
DL=Spalte Fenster rechts unten

Ausgabe: keine

2.8 Funktion 07h: Abwärts scrollen innerhalb eines Fensters

Eingabe: **AH**=07h
AL=Anzahl der Zeilen
(Bei AL=0 wird das Fenster gelöscht)
BH=Attribut für die neuen Zeilen
CH=Zeile Fenster links oben
CL=Spalte Fenster links oben
DH=Zeile Fenster rechts unten
DL=Spalte Fenster rechts unten

Ausgabe: keine

2.9 Funktion 08h: Zeichen vom Bildschirm lesen

Eingabe: **AH**=08h
BH=Nummer der Bildschirmseite

Ausgabe: **AL**=ASCII-Code des Zeichens
AH=Attribut-Byte des Zeichens

2.10 Funktion 09h: Zeichen mit Farb-Attribut (wiederholt) schreiben

Eingabe: **AH**=09h
AL=ASCII-Zeichencode
CX=Anzahl der Zeichenausgaben
BH=Nummer der Bildschirmseite
BL=Farb-Attribut

Ausgabe: keine

- Es können keine **ASCII-Steuer-Codes** (Zeilenvorschub, Tabulator, Piepton usw.) verarbeitet werden. Sie werden als normale ASCII-Zeichen ausgegeben.
- Im Grafikmodus müssen alle Zeichen (falls mehrere ausgegeben werden sollen) in eine Zeile passen. Es erfolgt **kein** Zeilenumbruch.
- Die aktuelle Curserposition wird durch diese Funktion **nicht** verändert. Soll dies dennoch geschehen, muss anschließend an diese Funktion der Cursor mit der Funktion 02h des Interrupt 10h nachgestellt werden, oder man macht die Ausgabe **anstelle** der **Funktion 09h** mit der Funktion 0Eh dieses Interrupts.
- Ein gesetztes Bit 7 im Farb-Attribut (in **BL**) zeigt an, dass die Farbe XOR-mäßig[1] mit den bereits aktuell vorhandenen Bildschirmdaten an der aktuellen Position verknüpft werden soll.
- Soll nur ein einzelnes Zeichen geschrieben werden, dann wird **CX**=1 gesetzt.

2.11 Funktion 0Ah: Zeichen ohne Farb-Attribut (wiederholt) schreiben

Eingabe: **AH**=0Ah
AL=ASCII-Zeichencode
CX=Anzahl der Zeichenausgaben
BH=Nummer der Bildschirmseite
(Cursorposition bleibt unverändert)

Ausgabe: keine

- Es können keine **ASCII-Steuer-Codes** (Zeilenvorschub, Tabulator, Piepton usw.) verarbeitet werden. Sie werden als normale ASCII-Zeichen ausgegeben.

[1] Exclusives Oder – Dadurch ist der Text bei jeder beliebigen vorhandenen Farbe erkennbar, da er immer mit einer anderen Farbe dargestellt wird.

- Im Grafikmodus müssen alle Zeichen (falls mehrere ausgegeben werden sollen) in eine Zeile passen. Es erfolgt **kein** Zeilenumbruch.
- Die aktuelle Curserposition wird durch diese Funktion **nicht** verändert. Soll dies dennoch geschehen, muss anschließend an diese Funktion der Cursor mit der Funktion 02h des Interrupt 10h nachgestellt werden, oder man macht die Ausgabe **anstelle** der **Funktion 0Ah** mit der Funktion 0Eh dieses Interrupts.
- Soll nur ein einzelnes Zeichen geschrieben werden, dann wird **CX**=1 gesetzt.

2.12 Funktion 0Bh: Farbe setzen

2.12.1 Funktion 0Bh, Unterfunktion 00h: Hintergrundfarbe setzen

Eingabe: **AH**=0Bh
BH=00h
BL=Rahmen- bzw. Hintergrundfarbe

Ausgabe: keine

- Im **Textmodus** stellt der gewählte Farbwert den Wert für den **Bildschirmrahmen** dar, weil die einzelnen Zeichen mit dem **jeweiligen** Attribut ihre Farbe (Zeichen- **und** Hintergrundfarbe) bestimmen.
- Im **Grafikmodus** bestimmt der Wert sowohl die Farbe des Hintergrundes als auch die Farbe des Rahmens.

2.12.2 Funktion 0Bh, Unterfunktion 01h: Farbpalette wählen

Eingabe: **AH**=0Bh
BH=01h
BL=Nummer der Farbpalette

Ausgabe: keine

2.13 Funktion 0Ch: Pixel setzen

Eingabe: **AH**=0Ch
DX=Grafik-Bildschirmzeile
CX=Grafik-Bildschirmspalte
AL=Farbwert

Ausgabe: keine

- Der verwendbare Farbwert ist vom aktuellen Bildschirmmodus abhängig. Je nach Grafikkarte und Auflösung können unterschiedliche Farbwerte verwendet werden. Der Farbwert gibt nur einen **relativen** Wert innerhalb einer Farbpalette und keinen **absoluten** Wert an.
- Abhängig vom verwendetet Grafikmodus muss der Farbwert als 1-Bit-, 2-Bit-, 4-Bit- oder 8-Bit-Zahl angegeben werden.

2.14 Funktion 0Dh: Pixel lesen

Eingabe: **AH**=0Dh
BH=Bildschirmseite
DX=Grafik-Bildschirmzeile
CX=Grafik-Bildschirmspalte

Ausgabe: **AL**=Farbwert

- Der ermittelte Farbwert ist vom aktuellen Bildschirmmodus abhängig. Je nach Grafikkarte und Auflösung können unterschiedliche Farbwerte verwendet werden. Der Farbwert gibt nur einen **relativen** Wert innerhalb einer Farbpalette und keinen **absoluten** Wert an.
- Ein Farbwert von **00h** kennzeichnet beim Einsatz der Standard-Palette einen Punkt in der **Hintergrundfarbe**, also einen **gelöschten** Punkt.

2.15 Funktion 0Eh: Zeichen im TTY-Modus schreiben

Eingabe: **AH**=0Eh
AL=ASCII-Zeichencode
BL=Vordergrundfarbe (nur Grafikmodus)

Ausgabe: keine

- Im Textmodus wird das aktuelle Attribut an der Cursor-Position **nicht** verändert.
- Die Vordergrundfarbe im **BL**-Register wird nur im Grafikmodus korrekt interpretiert. Im Textmodus wird sie nicht beachtet, da hier die Farbe über das **Attribut** gewählt wird.
- Steuerzeichen (Zeilenvorschub, Tabulator usw.) werden als solche ausgeführt.
- Die Cursorposition wird nach jedem Zeichen aktualisiert, der Cursor wird also mitbewegt.
- Beim Erreichen des Zeilenendes wird ein Zeilenumbruch durchgeführt, der Bildschirm wird ggf. gescrollt.

2.16 Funktion 0Fh: Videostatus lesen

Eingabe: **AH**=0Fh

Ausgabe: **AL**=Nummer des Video-Modus (s.u.)
AH=Anzahl der Zeichen pro Zeile
BH=Nummer der aktuellen Bildschirmseite

- Die **Zeichenzahl pro Zeile**, die in **AH** zurückgeliefert wird, ist die maximal in einer Zeile darstellbare Zeichenanzahl.
- Der Wertebereich des Video-Modus ist abhängig von der verwendeten Bildschirmkarte. Die heutigen Karten erlauben alle Modi außer **Modus 7**[2]. Nachfolgend ist eine Tabelle der möglichen Video-Modi dargestellt.

[2]Modus 7 wird nur für die alte Hercules-Schwarzweiß-Grafikkarte verwendet. Aber deren Grafikmodi werden vom BIOS ebenfalls als **Modus 7** gemeldet!

Mögliche Video-Modi:

Nr.	Modus	Grafik	Text	Farben
00	Text	320 × 200	40 × 25	einfarbig
01	Text	320 × 200	40 × 25	16
02	Text	640 × 200	80 × 25	einfarbig
03	Text	640 × 200	80 × 25	16
04	Grafik	320 × 200	40 × 25	einfarbig
05	Grafik	320 × 200	40 × 25	16
06	Grafik	640 × 200	80 × 25	einfarbig
07	(siehe Anmerkung)	720 × 350	80 × 25	einfarbig
13	Grafik	320 × 200	40 × 25	16
14	Grafik	640 × 200	80 × 25	16
15	Grafik	640 × 350	80 × 25	einfarbig
16	Grafik	640 × 350	80 × 25	16 aus 64

2.17 Funktion 10h: Parameter lesen oder setzen

2.17.1 Funktion 10h, Unterfunktion 00h: Veränderte Paletten-Register

Eingabe: **AX**=1000h
BL=Nummer des Paletten-Registers
BH=Farbwert

Ausgabe: keine

- Paletten-Register werden verwendet, um die konkrete Farbe eines Punktes in den 16-Farben-Modi zu ermitteln. Das Bitmuster des Punktes wir dazu als Nummer eines Paletten-Registers interpretiert, erst der Inhalt des Paletten-Registers wird als Farbe an den Monitor geschickt.

2.17.2 Funktion 10h, Unterfunktion 01h: Rahmenfarbe wählen

Eingabe: **AX**=1001h
BH=Farbwert

Ausgabe: keine

2.17.3 Funktion 10h, Unterfunktion 02h: Alle Paletten-Register und die Rahmenfarbe setzen

Eingabe: **AX**=1002h
ES=Segment-Adresse einer Parameter-Tabelle
DX=Offset-Adresse einer Parameter-Tabelle

Ausgabe: keine

- In der Parameter-Tabelle ab Adresse **ES:DX** müssen sich insgesamt 17 Werte von Byte-Größe befinden. Die ersten 16 Werte ergeben die Farbwerte für die 16 Paletten-Register an, der Wert Nummer 17 gibt die Rahmenfarbe an.

2.17.4 Funktion 10h, Unterfunktion 03h: Hintergrund-Intensität setzen

Eingabe: **AX**=1003h
BL=Intensitäts-Byte
Ausgabe: keine

- Das Intensitäts-Byte darf nur die Werte **0** oder **1** annehmen. Ein Wert von **0** bedeutet, dass der Hintergrund **intensiv** dargestellt wird, ein Wert von **1** bewirkt einen **blinkenden** Hintergrund.

2.17.5 Funktion 10h, Unterfunktion 07h: Paletten-Register auslesen

Eingabe: **AX**=1007h
BL=Nummer des Paletten-Registers
Ausgabe: **BH**=Farbwert

- Der Farbwert gibt den Inhalt des gewünschten Paletten-Registers wieder.

2.17.6 Funktion 10h, Unterfunktion 08h: Rahmenfarbe auslesen

Eingabe: **AX**=1008h
Ausgabe: **BH**=Rahmenfarbe

2.17.7 Funktion 10h, Unterfunktion 09h: Palettenregister und Rahmenfarbe auslesen

Eingabe: **AX**=1009h
Ausgabe: Parametertabelle in **ES:DX**

- In der Parameter-Tabelle ab Adresse **ES:DX** muss vor dem Aufruf Platz insgesamt 17 Bytes vorgesehen sein. Alle eventuell vorhandene Werte in dieser Tabelle werden durch neue Werte überschrieben.
- Die ersten 16 Bytes ergeben die Farbwerte für die 16 Paletten-Register an, der Wert Nummer 17 gibt die Rahmenfarbe an.

2.18 Funktion 11h: Zeichensatzfunktionen

2.18.1 Funktion 11h, Unterfunktion 00h: Zeichensatz laden, Reset ausführen

Eingabe: **AX**=1100h
BH=Byte-Anzahl pro Zeichen
BL=Blocknummer
CX=Zeichenanzahl
DX=Bildschirmspeicher-Offset
ES=Segment-Adresse des Zeichensatzes
BP=Offest-Adresse des Zeichensatzes
Ausgabe: keine

- Die Funktion lädt einen benutzerdefinierten Zeichensatz in den Bildschirmspeicher der Grafikkarte. Die Tabelle muss sich an der durch **ES:BP** adressierten Stelle im Speicher befinden.
- Über das Register **BL** wird festgelegt, an welche Zeichensatz-Position der Zeichensatz in den Bildschirmspeicher geladen werden soll. Es ist Platz für 8 Zeichensätze vorhanden, **BL** darf also Werte von 0 bis 7 enthalten.
- Mit dem Register **CX** wird festgelegt, wieviele Zeichen der Zeichensatz umfasst, der in den Bildschirmspeicher geladen werden soll.
- Das Register **BH** bestimmt, aus wievielen Bytes jedes Zeichen besteht. Es sind Werte bis maximal 32 Byte zulässig.
- Das Register **DX** legt den Offset fest, an dem der Zeichensatz vom Beginn des mit **BL** festgelegten Zeichensatzes im Bildschirmspeicher abgelegt werden soll. Wenn die Zeichensatz-Definition mit **ASCII 00h** beginnt, dann muss der Wert **000h** im Register **DX** stehen. Falls der Zeichensatz weiter „oben“ im Bildschirmspeicher beginnen soll (weil beispielsweise nur die Buchstaben von **A** biz **Z** geändert werden sollen), muss das Register **DX** auf einen entsprechend höheren Wert eingestellt werden.
- Jedes Zeichen im Zeichensatz belegt 32 Bytes, unabhängig davon, wie viele Zeilen das Zeichen tatsächlich umfasst.
- Die Funktion führt mit der Grafikkarte einen **Reset** aus.

2.18.2 Funktion 11h, Unterfunktion 01h: Monochrom-Zeichensatz setzen, Reset ausführen

Eingabe: **AX**=1101h
BL=Blocknummer

Ausgabe: keine

- Die Funktion kopiert den 9×14-Punkte-Zeichensatz in den Bildschirmspeicher der Grafikkarte.
- Das Register **BL** legt fest, an welche Position das BIOS den Zeichensatz schreiben soll. Werte von 0 bis 7 sind zulässig.
- Es ist möglich, zunächst mit dieser Funktion den 9×14-Punkte-Zeichensatz in den Bildschirmspeicher der Grafikkarte zu laden und anschließend bestimmte Zeichen des Zeichensatzes zu verändern, indem man diese mit der **Funktion 11h, Unterfunktion 00h** direkt überschreibt oder Teile des Zeichensatzes hinzulädt.
- Die Funktion führt mit der Grafikkarte einen **Reset** aus.

2.18.3 Funktion 11h, Unterfunktion 02h: 8×8-Zeichensatz setzen, Reset ausführen

Eingabe: **AX**=1101h
BL=Blocknummer

Ausgabe: keine

- Die Funktion kopiert den 8×8-Punkte-Zeichensatz in den Bildschirmspeicher der Grafikkarte.
- Das Register **BL** legt fest, an welche Position das BIOS den Zeichensatz schreiben soll. Werte von 0 bis 7 sind zulässig.
- Es ist möglich, zunächst mit dieser Funktion den 8×8-Punkte-Zeichensatz in den Bildschirmspeicher der Grafikkarte zu laden und anschließend bestimmte Zeichen des Zeichensatzes zu verändern, indem man diese mit der **Funktion 11h, Unterfunktion 00h** direkt überschreibt oder Teile des Zeichensatzes hinzulädt.
- In Grafikmodi mit 350 Zeilen erlaubt dieser Zeichensatz die Darstellung von 43 Zeilen Text, in Grafikmodi mit 480 Zeilen lassen sich 60 Zeilen auf dem Bildschirm darstellen.
- Die Funktion führt mit der Grafikkarte einen **Reset** aus.

3 Interrupt 13h: Disketten und Festplattenzugriffe

3.1 Funktion 00h: Laufwerk initialisieren

Eingabe: **AH**=00h
DL=Laufwerknummer

Ausgabe: **Carry-Flag** zeigt Fehler an
AH=Fehlercode

Folgende Laufwerknummern gelten:

0:	alle Laufwerke
1:	Disketten-Laufwerk A
2:	Disketten-Laufwerk B
80h:	1. Festplatte
81h:	2. Festplatte
...	usw.

Ist das **Carry-Flag** gesetzt, gelten in **AH** folgende **Fehlercodes**:

AH	Bedeutung:
00h	Kein Fehler
01h	Illegale Funktionsnummer
02h	Keine Adressmarkierung
03h	Diskette ist schreibgeschützt
04h	Sektor nicht gefunden
05h	Reset war erfolglos
07h	Fehlerhafte Initialisierung
08h	Überlauf des DMA

AH	Bedeutung:
09h	Segment-Überlauf des DMA
10h	Lesefehler
11h	falsche Prüfsumme
20h	Controllerfehler
40h	Spur nicht gefunden
80h	Laufwerk reagiert nicht
BBh	BIOS-Fehler
FFh	Nicht aufschlüsselbarer Fehler

3.2 Funktion 01h: Laufwerkstatus lesen

Eingabe: **AH**=01h
DL=Laufwerknummer
(Laufwerknummer siehe unten)

Ausgabe: **Carry-Flag** zeigt Fehler an
AH=Fehlercode
(Fehlercode siehe **Interrupt 13h, Funktion 00h**)

Folgende Laufwerknummern gelten:

0:	aktuelles Laufwerk
1:	Disketten-Laufwerk A
2:	Disketten-Laufwerk B
80h:	1. Festplatte
81h:	2. Festplatte
...	usw.

3.3 Funktion 02h: Sektoren lesen

Eingabe: **AH**=02h
DL=Laufwerknummer (Laufwerknummer siehe **Interrupt 13h, Funktion 01h**)
DH=Seiten-/Kopfnummer
CH=Spur-/Zylinder-Nummer
CL=Sektor-Nummer
AL=Sektorenanzahl
ES:BX=Adresse des Lesepuffers (512 Byte je Sektor)

Ausgabe: **Carry-Flag** zeigt Fehler an
AH=Fehlercode (Fehlercode siehe **Interrupt 13h, Funktion 00h**)

- Die **Puffergröße** muss so eingerichtet sein, dass die Anzahl der Sektoren auch hineinpassen. Anderenfalls wird in fremden Speicherbereich hineingeschrieben. Ein Sektor belegt 512 Byte.
- Mit der Angabe der **Sektorenzahl** können nur physikalisch aufeinanderfolgende Sektoren für die gleiche Zylindernummer gelesen werden.
- Erfolgt ein Überlauf über die Grenze der aktuellen Spur (durch eine zu hohe Sektorenzahl), dann wird auf dem ersten Sektor des gleichen Zylinders auf dem nachfolgenden Kopf weitergelesen.
- Da mit den 8 Bits des Registers **CH** beim Ansprechen von Festplatten nur maximal 256 Zylinder angesprochen werden können, werden die Bits 6 und 7 des Registers **CL** als logische Bits 8 und 9 der Zylindernummer **CH** verwendet.

3.4 Funktion 03h: Sektoren schreiben

Eingabe: **AH**=03h
DL=Laufwerknummer (Laufwerknummer siehe **Interrupt 13h, Funktion 01h**)
DH=Seiten-/Kopfnummer
CH=Spur-/Zylinder-Nummer
CL=Sektor-Nummer
AL=Sektorenanzahl
ES:BX=Adresse des Schreibpuffers (512 Byte je Sektor)

Ausgabe: **Carry-Flag** zeigt Fehler an
AH=Fehlercode (Fehlercode siehe **Interrupt 13h, Funktion 00h**)

- Die **Puffergröße** muss so eingerichtet sein, dass die Anzahl der Sektoren auch hineinpassen. Anderenfalls wird aus fremden Speicherbereich gelesen. Ein Sektor belegt 512 Byte.
- Mit der Angabe der **Sektorenzahl** können nur physikalisch aufeinanderfolgende Sektoren für die gleiche Zylindernummer geschrieben werden.

- Erfolgt ein Überlauf über die Grenze der aktuellen Spur (durch eine zu hohe Sektorenzahl), dann wird auf dem ersten Sektor des gleichen Zylinders auf dem nachfolgenden Kopf weitergeschrieben.
- Da mit den 8 Bits des Registers **CH** beim Ansprechen von Festplatten nur maximal 256 Zylinder angesprochen werden können, werden die Bits 6 und 7 des Registers **CL** als logische Bits 8 und 9 der Zylindernummer **CH** verwendet.

3.5 Funktion 04h: Sektoren prüfen

Eingabe: **AH**=04h
DL=Laufwerknummer (Laufwerknummer siehe **Interrupt 13h, Funktion 01h**)
DH=Seiten-/Kopfnummer
CH=Spur-/Zylinder-Nummer
CL=Sektor-Nummer
AL=Sektorenanzahl
ES:BX=Adresse des Prüfpuffers (512 Byte je Sektor)

Ausgabe: **Carry-Flag** zeigt Fehler an
AH=Fehlercode (Fehlercode siehe **Interrupt 13h, Funktion 00h**)

- Die **Puffergröße** muss so eingerichtet sein, dass die Anzahl der Sektoren auch hineinpassen. Ein Sektor belegt 512 Byte.
- Mit der Angabe der **Sektorenzahl** können nur physikalisch aufeinanderfolgende Sektoren für die gleiche Zylindernummer gelesen werden.
- Erfolgt ein Überlauf über die Grenze der aktuellen Spur (durch eine zu hohe Sektorenzahl), dann wird auf dem ersten Sektor des gleichen Zylinders auf dem nachfolgenden Kopf weitergelesen.
- Da mit den 8 Bits des Registers **CH** beim Ansprechen von Festplatten nur maximal 256 Zylinder angesprochen werden können, werden die Bits 6 und 7 des Registers **CL** als logische Bits 8 und 9 der Zylindernummer **CH** verwendet.

3.6 Funktion 05h: Spuren/Zylinder formatieren

Eingabe: **AH**=04h
DL=Laufwerknummer (Laufwerknummer siehe **Interrupt 13h, Funktion 01h**)
DH=Seiten-/Kopfnummer
CH=Spur-/Zylinder-Nummer
AL=Sektorenanzahl
ES:BX=Adresse der Formattabelle (siehe unten)

Ausgabe: **Carry-Flag** zeigt Fehler an
AH=Fehlercode (Fehlercode siehe **Interrupt 13h, Funktion 00h**)

Es muss ein Zeiger auf eine Tabelle übergeben werden, in der das Format des zu formatierenden Sektors eingetragen ist. Die Tabelle enthält 4 Byte, wie nachfolgend dargestellt ist.

Aufbau der Formattabelle:

Byte-Nr.	Bedeutung
0	Spur des Sektors
1	Seite des Sektors
2	Logische Sektornummer
3	Byte-Anzahl im Sektor, codiert:
	0: 128 1: 256 2: 3: 1024

- Mit der Angabe der **Sektorenzahl** können nur physikalisch aufeinanderfolgende Sektoren für die gleiche Zylindernummer formatiert werden.
- Da mit den 8 Bits des Registers **CH** beim Ansprechen von Festplatten nur maximal 256 Zylinder angesprochen werden können, werden die Bits 6 und 7 des Registers **CL** als logische Bits 8 und 9 der Zylindernummer **CH** verwendet.

3.7 Funktion 06h: Spur kennzeichnen

Eingabe: **AH**=06h
DL=Laufwerknummer (Laufwerknummer siehe **Interrupt 13h, Funktion 01h**)
DH=Seiten-/Kopfnummer
CH=Spur-/Zylinder-Nummer

Ausgabe: **Carry-Flag** zeigt Fehler an
AH=Fehlercode (Fehlercode siehe **Interrupt 13h, Funktion 00h**)

3.8 Funktion 07h: Zylinder ab Nr. CH formatieren

Eingabe: **AH**=07h
DL=Laufwerknummer (Laufwerknummer siehe **Interrupt 13h, Funktion 01h**)
DH=Kopfnummer
CH=Start-Zylindernummer
CL=Sektornummer
AL=Sektorenanzahl
ES:BX=Adresse der Formatpuffers (512 Byte je Sektor)

Ausgabe: **Carry-Flag** zeigt Fehler an
AH=Fehlercode (Fehlercode siehe **Interrupt 13h, Funktion 00h**)

- Mit dieser Funktion wird immer ein **kompletter Zayilnder** formatiert. Man sollte also immer eine 1 ins **CL**-Register schreiben, der erste Sektor wird normalerweise zuerst formatiert.

- Im Formatpuffer stehen nach dem Funktionsaufruf zwei Byte, die eine Aussage über den Ablauf der Formatierung geben. Ist das erste Byte =0, dann war die Formatierung erfolgreich. Das zweite gibt die logische Sektornummer des ersten formatierten Sektors an.

- Da mit den 8 Bits des Registers **CH** beim Ansprechen von Festplatten nur maximal 256 Zylinder angesprochen werden können, werden die Bits 6 und 7 des Registers **CL** als logische Bits 8 und 9 der Zylindernummer **CH** verwendet.

3.9 Funktion 08h: Format ermitteln

Eingabe: **AH**=08h
DL=Laufwerknummer
(Laufwerknummer siehe **Interrupt 13h, Funktion 01h**)

Ausgabe: **Carry-Flag** zeigt Fehler an
AH=Fehlercode
(Fehlercode siehe **Interrupt 13h, Funktion 00h**)
DH=Kopf-Anzahl
CH=Zylinder-Anzahl
CL=Sektor-Anzahl
DL=Festplatten-Anzahl

- Aus den ermittelten Daten kann die Gesamtkapazität des Laufwerkes bestimmt werden. Man muss dann nur noch 512 (Byte-Zahl je Sektor) mit der Zylinder-Anzahl, der Kopf-Anzahl und der Sektor-Anzahl multiplizieren.

- Da mit den 8 Bits des Registers **CH** beim Ansprechen von Festplatten nur maximal 256 Zylinder angesprochen werden können, werden die Bits 6 und 7 des Registers **CL** als logische Bits 8 und 9 der Zylindernummer **CH** verwendet.

3.10 Funktion 0Ah: Erweiterte Sektoren lesen

Eingabe: **AH**=0Ah
DL=Laufwerknummer
(Laufwerknummer siehe **Interrupt 13h, Funktion 01h**)
DH=Seiten-/Kopfnummer
CH=Spur-/Zylinder-Nummer
CL=Sektor-Nummer
AL=Sektorenanzahl
ES:BX=Adresse des Lesepuffers (**516 Byte** je Sektor!)

Ausgabe: **Carry-Flag** zeigt Fehler an
AH=Fehlercode
(Fehlercode siehe **Interrupt 13h, Funktion 00h**)

- Die **Puffergröße** muss so eingerichtet sein, dass die Anzahl der Sektoren auch hineinpassen. Anderenfalls wird in fremden Speicherbereich hineingeschrieben. Ein Sektor belegt 512 Byte. Dazu kommen jeweils **vier Kontroll-Byte** für die Prüfsumme.

- Mit der Angabe der **Sektorenzahl** können nur physikalisch aufeinanderfolgende Sektoren für die gleiche Zylindernummer gelesen werden.
- Erfolgt ein Überlauf über die Grenze der aktuellen Spur (durch eine zu hohe Sektorenzahl), dann wird auf dem ersten Sektor des gleichen Zylinders auf dem nachfolgenden Kopf weitergelesen.
- Da mit den 8 Bits des Registers **CH** beim Ansprechen von Festplatten nur maximal 256 Zylinder angesprochen werden können, werden die Bits 6 und 7 des Registers **CL** als logische Bits 8 und 9 der Zylindernummer **CH** verwendet.
- Der Controller bildet zu jedem Sektor der Festplatte eine Prüfsumme. Diese dient zum Erkennen von Schreib- und Lesefehlern und wird normalerweise nur intern im Festplattencontroller benutzt. Mit dieser Funktion kann man die Prüfsumme explizit auslesen und kontrollieren oder (mit der nachfolgenden Funktion **0Bh**) auch manipulieren.

3.11 Funktion 0Bh: Erweiterte Sektoren schreiben

Eingabe: **AH**=0Bh
DL=Laufwerknummer (Laufwerknummer siehe **Interrupt 13h, Funktion 01h**)
DH=Seiten-/Kopfnummer
CH=Spur-/Zylinder-Nummer
CL=Sektor-Nummer
AL=Sektorenanzahl
ES:BX=Adresse des Schreibpuffers (**516 Byte** je Sektor!)

Ausgabe: **Carry-Flag** zeigt Fehler an
AH=Fehlercode (Fehlercode siehe **Interrupt 13h, Funktion 00h**)

- Die **Puffergröße** muss so eingerichtet sein, dass die Anzahl der Sektoren auch hineinpassen. Ein Sektor belegt 512 Byte. Dazu kommen jeweils **vier Kontroll-Byte** für die Prüfsumme, die jeweils im Anschluss an die 512 Bytes des Sektors im RAM stehen. **Achtung! Mit dieser Funktion schreibt das BIOS die von Ihnen zu verantwortende Prüfsumme ohne weitere Überprüfung in den Sektor!**
- Mit der Angabe der **Sektorenzahl** können nur physikalisch aufeinanderfolgende Sektoren für die gleiche Zylindernummer geschrieben werden.
- Erfolgt ein Überlauf über die Grenze der aktuellen Spur (durch eine zu hohe Sektorenzahl), dann wird auf dem ersten Sektor des gleichen Zylinders auf dem nachfolgenden Kopf weitergeschrieben.
- Da mit den 8 Bits des Registers **CH** beim Ansprechen von Festplatten nur maximal 256 Zylinder angesprochen werden können, werden die Bits 6 und 7 des Registers **CL** als logische Bits 8 und 9 der Zylindernummer **CH** verwendet.

- Der Controller bildet zu jedem Sektor der Festplatte eine Prüfsumme. Diese dient zum Erkennen von Schreib- und Lesefehlern und wird normalerweise nur intern im Festplattencontroller benutzt. Mit der vorstehenden Funktion **0Ah** kann man die Prüfsumme explizit auslesen und kontrollieren, mit dieser Funktion **0Bh** auch manipulieren. **Hierbei sollte man größte Vorsicht walten lassen, da bei fehlerhafter Prüfsumme sonst der Controller den Sektor *immer* als fehlerhaft erkennt!**

3.12 Funktion 0Ch: Schreib-/Lese-Kopf bewegen

Eingabe: **AH**=0Ch
DL=Laufwerknummer
(Laufwerknummer siehe **Interrupt 13h, Funktion 01h**)
DH=Kopfnummer
CX=Zylinder-Nummer

Ausgabe: **Carry-Flag** zeigt Fehler an
AH=Fehlercode
(Fehlercode siehe **Interrupt 13h, Funktion 00h**)

3.13 Funktion 10h: Überprüfung der Laufwerkbereitschaft

Eingabe: **AH**=10h
DL=Laufwerknummer
(Laufwerknummer siehe **Interrupt 13h, Funktion 01h**)

Ausgabe: **Carry-Flag** zeigt Fehler an
AH=Fehlercode
(Fehlercode siehe **Interrupt 13h, Funktion 00h**)

3.14 Funktion 13h: Spur 0 überprüfen

Eingabe: **AH**=13h
DL=Laufwerknummer
(Laufwerknummer siehe **Interrupt 13h, Funktion 01h**)

Ausgabe: **Carry-Flag** zeigt Fehler an
AH=Fehlercode
(Fehlercode siehe **Interrupt 13h, Funktion 00h**)

3.15 Funktion 14h: Controller testen

Eingabe: **AH**=14h
DL=Laufwerknummer
(Laufwerknummer siehe **Interrupt 13h, Funktion 01h**)

Ausgabe: **Carry-Flag** zeigt Fehler an
AH=Fehlercode
(Fehlercode siehe **Interrupt 13h, Funktion 00h**)

4 Interrupt 14h: Serielle Schnittstelle

Für alle Funktionen des Interrupt 14h gelten folgende Festlegungen:

- Die erste Schnittstelle **COM1** hat die Nummer **00h**, die zweite Schnittstelle **COM2** hat die Nummer **01h**, usw.
- Der **Konfigurationscode** gibt Aufschluss über die Parameter der Schnittstelle. Nachfolgende Tabelle stellt die Zusammensetzung des Konfigurations-Codes dar.

Konfigurationscode:

Bit-Nr.	Bedeutung	
0	Flag für Datenbits:	**0**: 7 Datenbits
		1: 8 Datenbits
1	(muss immer 1 sein)	
2	Flag für Stoppbits	**0**: 1 Stoppbit
		1: 2 Stoppbits
3	Flag für Parität	**0**: keine Parität
		1: Parität
4	Paritätsmodus	**0**: ungerade Parität
		1: gerade Parität
5-7	Code der Baudrate	(siehe unten)

Mögliche Bit-Kombinationen für die Baudrate:

Bit-Code	Bedeutung
000	110 Baud
001	150 Baud
010	300 Baud
011	600 Baud
100	1200 Baud
101	2400 Baud
110	4800 Baud
111	9600 Baud

- Der Code der Baudrate gibt die **Übertragungsgeschwindigkeit** an.
- Der **Zeilen-Status** ist der aktuelle Status der Schnittstelle bzw. der zu empfangenden Daten. Die einzelnen Bits des **Zeilen-Status-Registers** sind Flags für die folgenden Funktionen.

Bedeutungen des Leitungs-Status-Registers:

Bit-Nr.	Bedeutung
0	Daten stehen zum Empfang bereit (Data Ready)
1	Datenüberlauf (Overrun Error)
2	Paritätsfehler (Parity Error)
3	Protokollfehler (Framing Error)
4	Unterbrechung erkannt (Break Identified)
5	Übertragungs-Halteregister leer (Transmission Hold Register Empty)
6	Übertragungs-Schieberegister leer (Transmission Shift Register Empty)
7	Timeout-Fehler (Time Out Error)

Bedeutungen des Modem-Status-Registers:

Bit-Nr.	Bedeutung
0	(D) klar zum Senden (Delta Data clear to send)
1	(D) Modem ist bereit (Delta Data Terminal Ready)
2	(D) Telefon läutet (Delta Trailing Edge Ring Indicator)
3	(D) Datenträgersignal erkannt (Delta Data Carrier Detect)
4	klar zum Senden (Data clear to send)
5	Modem ist bereit (Data Terminal Ready)
6	Telefon läutet (Trailing Edge Ring Indicator)
7	Datenträgersignal erkannt (Data Carrier Detect)

- Jede Steuerleitung ist doppelt vorhanden, einmal mit vorangestelltem **(D)** oder **delta** und einmal ohne dieses Zeichen. Die Bits mit vorangestelltem **(D)** kennzeichnen den **Unterschied** zur letzten Abfrage, die anderen den **aktuellen Zustand**. Wenn man den Status häufig abfragt, kann man anhand der Bits 0 bis 3 feststellen, ob irgendein Statuswechsel bei einem der Signale vorliegt. Liegt er vor, dann kann man den Status mit den Bits 4 bis 7 direkt abfragen.

- Für die Wahl einer Parität genügt es nicht, nur das Auswahl-Bit (Bit 4) im Konfigurationscode für die gerade oder ungerade Parität zu setzen. Will man eine Paritätsprüfung ermöglichen, muss zusätzlich noch das Auswahl-Bit (Bit 3) gesetzt werden. Dieses Bit schaltet die generelle Paritätsprüfung ein.

4.1 Funktion 00h: Initialisierung

Eingabe: **AH**=00h
AL=Konfigurationscode
DX=Schnittstellennummer
(COM1=0, COM2=1...)

Ausgabe: **AH**=Leitungs-Status
AL=Modemstatus

4.2 Funktion 01h: Ein Zeichen senden

Eingabe: **AH**=01h
AL=Zeichen-Code
DX=Schnittstellennummer

Ausgabe: **AH**=Leitungs-Status

- Mit dem **Leitungs-Status** kann auf einen Fehler geprüft werden. Ist Bit 7 (TOE) gleich 0, dann wurde das Zeichen übertragen, anderenfalls können die Fehlermöglichkeiten mit den verbleibenden Bits erkannt werden.

4.3 Funktion 02h: Ein Zeichen lesen

Eingabe: **AH**=02h
DX=Schnittstellennummer

Ausgabe: **AH**=Leitungs-Status
AL=Zeichen-Code des empfangenen Zeichens

- Mit dem **Leitungs-Status** kann auf einen Fehler geprüft werden. Ist Bit 7 (TOE) gleich 0, dann wurde das Zeichen übertragen, anderenfalls können die Fehlermöglichkeiten mit den verbleibenden Bits erkannt werden.
- Man sollte nur Daten empfangen, wenn ein Zeichen zum Empfang bereitsteht (siehe Funktion **03H**).

4.4 Funktion 03h: Status erfragen

Eingabe: **AH**=03h
DX=Schnittstellennummer

Ausgabe: **AH**=Leitungs-Status
AL=Modemstatus

4.5 Funktion 04h: Erweiterte Initialisierung vornehmen

Eingabe: **AH**=04h
DX=Schnittstellennummer
AL=Break-Zustand
BL=Stop-Bit-Anzahl
BH=Parität (s.u.)
CH=Baudrate (s.u.)
CL=Datenbitanzahl (s.u.)

Ausgabe: **AH**=Leitungs-Status
AL=Modemstatus

Parität:

Code:	Bedeutung:
00h	Keine Parität
01h	Ungerade Parität
02h	Gerade Parität
03h	Feste Parität: Ungerade
04h	Feste Parität: Gerade

Datenbitanzahl:

Code:	Bedeutung:
00h	5 Datenbits
01h	6 Datenbits
02h	7 Datenbits
03h	8 Datenbits

Baudrate:

Code:	Bedeutung:
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

- Der Code für den **Break-Zustand** im **AL**-Register darf entweder **00h** oder **01h** sein. Bei einem Wert von **00h** wird ein Break eingesetzt, bei **01h** keins.
- Der Code für die Anzahl der Stop-Bits beträgt entweder **00h** oder **01h**. Bei einem Code von **00h** arbeitet die Schnittstelle mit **einem** Stop-Bit, ist der Code **01h**, werden **zwei** Stop-Bits verwendet. ***Ausnahme:*** Die Kombination aus **01h** für die Stop-Bits und **00h** für die Daten-Bit-Anzahl bewirkt den Gebrauch von 1,5 Stop-Bits.

5 Interrupt 15h: Diverse Funktionen

5.1 Funktion 83h: Flag nach Zeitintervall setzen

Eingabe: **AH**=83h
AL=0
ES:BX=Flagadresse
CX=Timerwert High
DX=Timerwert Low

Ausgabe: keine

Der Timerwert ist die Wartezeit als Doppelwort in Mikrosekunden. Sobald die angegebene Zeit verstrichen ist, wird **Bit 7** des durch **ES:BX** adressierten Bytes gesetzt. Das Programm kann in der Zeit weiterarbeiten. Man muss nur dafür sorgen, dass das Flag regelmäßig geprüft wird.

5.2 Funktion 84h: Joystick

5.2.1 Funktion 84h, Unterfunktion 00h: Joystick Feuerknopfstatus ermitteln

Eingabe: **AH**=84h
DX=0

Ausgabe: **CF** zeigt Fehler an,
z.B. kein Joystick angeschlossen, ansonsten:
AL=Status der Knöpfe

- Das **Carry-Flag** zeigt einen Fehler an. Ist es gesetzt, so ist kein Joystick angeschlossen, oder es wird keiner erkannt. Nur bei gelöschtem Flag enthält das **Al**-Register einen aussagekräftigen Wert.

Bedeutung des Feuerknopfstatus:

Bit-Nr.	Bedeutung:
7	Erster Feuer-Knopf erster Joystick
6	Zweiter Feuer-Knopf erster Joystick
5	Erster Feuer-Knopf zweiter Joystick
4	Zweiter Feuer-Knopf zweiter Joystick

Ein gesetztes Bit zeigt an, dass der entsprechende Knopf gedrückt ist.

5.2.2 Funktion 84h, Unterfunktion 01h: Joystick-Stellung ermitteln

Eingabe: **AH=84h**
DX=1

Ausgabe: **CF** zeigt Fehler an,
z.B. kein Joystick angeschlossen, ansonsten:
AX=X-Stellung des ersten Joystick
BX=Y-Stellung des ersten Joystick
CX=X-Stellung des zweiten Joystick
DX=Y-Stellung des zweiten Joystick

5.3 Funktion 86h: Zeitintervall abwarten

Eingabe: **AH**=86h
AL=0
CX=Timerwert High
DX=Timerwert Low

Ausgabe: keine

Der Timerwert ist die Wartezeit als Doppelwort in Mikrosekunden. Das Programm wird angehalten, bis die angegebene Zeit verstrichen ist. **Der Rechner ist in dieser Zeit für andere Arbeiten blockiert!** Ist das nicht erwünscht, verwendet man besser die Funktion **83h**.

5.4 Funktion 88h: Größe des Extended Memory ermitteln

Eingabe: **AH**=88h

Ausgabe: **AX**=Größe des HI-Speichers oberhalb 1MB (Angabe in KB)

6 Interrupt 16h: Tastatur-Zugriff

6.1 Funktion 00h: Tastaturcode aus Puffer lesen

(Veraltete Funktion, ersetzt durch Funktion 10h)

Eingabe: **AH**=00h **Ausgabe:** **AX**=Tastaturcode (s.u.)

Der **Tastaturcode** besteht aus 2 Byte in **AL** und **AH**. **AL** enthält den **Zeichencode**. Das ist der **ASCII-Code** des Zeichens. In **AH** befindet sich der **Scancode** der Tastatur. Dieser ist abhängig von der Tastatur.

Falls **AL**=0 ist, enthält **AH** den erweiterten Tastaturcode einer Sondertaste.

Nach Aufruf dieser Funktion ist das Zeichen aus dem Puffer gelöscht.

Befindet sich **kein** Zeichen im Puffer, wartet die Funktion so lange, bist ein Zeichen eingegeben wird. **Der Rechner ist für diese Zeit für andere Aufgaben blockiert!** Ist das nicht erwünscht, sollte man besser mit der nachfolgenden Funktion vorher prüfen, ob ein Zeichen vorhanden ist.

Achtung: Nicht alle Sonder-Tasten /-Kombinationen werden unterstützt, daher besser die neuere Funktion **10h** zur Tastaturabfrage verwenden!

6.2 Funktion 01h: Tastaturpuffer prüfen

(Veraltete Funktion, ersetzt durch Funktion 11h)

Eingabe: **AH**=01h **Ausgabe:** **Zero-Flag** zeigt Pufferstatus an (s.u.).
AX=Tastaturcode

Ist das **ZF** gesetzt, befindet sich **kein** Zeichen im Puffer. **AX** ist dann **ungültig**. Ist **ZF nicht** gesetzt, gilt für **AX** das zu Funktion 00h gesagte.

Das Zeichen wird **nicht** aus dem Puffer gelöscht.

6.3 Funktion 02h: Tastaturstatus lesen

(Welche Hilfs-Taste ist gedrückt?)

Eingabe: **AH**=02h **Ausgabe:** **AL**=Tastaturstatus (s.u.)

Tastaturstatus:

Bit-Nr.	Bedeutung:
0	rechte Shifttaste gedrückt
1	linke Shifttaste gedrückt
2	Strg-Taste gedrückt
3	Alt-Taste gedrückt

Bit-Nr.	Bedeutung:
4	Scroll-Lock aktiv
5	Num-Lock aktiv
6	Caps-Lock aktiv
7	Einf-Taste gedrückt

6.4 Funktion 10h: Tastaturcode aus Puffer lesen

(Ersetzt die veraltete Funktion 00h)

Eingabe: **AH**=10h **Ausgabe:** **AX**=Tastaturcode

Der **Tastaturcode** besteht aus 2 Byte in **AL** und **AH**. **AL** enthält den **Zeichencode**. Das ist der **ASCII-Code** des Zeichens. In **AH** befindet sich der **Scancode** der Tastatur. Dieser ist abhängig von der Tastatur.

Falls **AL=0** oder **AL=E0h** ist, enthält **AH** den erweiterten Tastaturcode einer Sondertaste.

Nach Aufruf dieser Funktion ist das Zeichen aus dem Puffer gelöscht.

Befindet sich **kein** Zeichen im Puffer, wartet die Funktion so lange, bist ein Zeichen eingegeben wird. **Der Rechner ist für diese Zeit für andere Aufgaben blockiert!** Ist das nicht erwünscht, sollte man besser mit der nachfolgenden Funktion vorher prüfen, ob ein Zeichen vorhanden ist.

6.5 Funktion 11h: Tastaturpuffer prüfen

(Ersetzt die veraltete Funktion 01h)

Eingabe: **AH**=11h **Ausgabe:** **Zero-Flag** zeigt Pufferstatus an (s.u.).
AX=Tastaturcode

Ist das **ZF** gesetzt, befindet sich **kein** Zeichen im Puffer. **AX** ist dann **ungültig**. Ist **ZF nicht** gesetzt, gilt für **AX** das zu Funktion 10h gesagte.

Das Zeichen wird **nicht** aus dem Puffer gelöscht.

7 Interrupt 17h: Parallele Schnittstelle

7.1 Funktion 00h: Zeichen ausgeben

Eingabe: **AH**=00h
AL=Zeichencode
DX=Schnittstellennummer
(LPT1=0, LPT2=1...)

Ausgabe: **AH**=Schnittstellestatus

Bedeutung der Bits des Schnittstellenstatus:

Bit-Nr.	Bedeutung:
0	Timeout-Fehler
1	ist immer 0
2	ist immer 0
3	Ein-/Ausgabefehler

Bit-Nr.	Bedeutung:
4	Drucker online
5	Papierende
6	Empfangsbestätigung
7	Beschäftigt

7.2 Funktion 01h: Initialisierung

Eingabe: **AH**=01h
DX=Schnittstellennummer
(LPT1=0, LPT2=1...)

Ausgabe: **AH**=Schnittstellestatus
(siehe Funktion **00h**)

7.3 Funktion 02h: Status der Schnittstelle ermitteln

Eingabe: **AH**=02h
DX=Schnittstellennummer
(LPT1=0, LPT2=1...)

Ausgabe: **AH**=Schnittstellestatus
(siehe Funktion **00h**)

8 Interrupt 19h: System booten

Eingabe: keine

Ausgabe: keine

Der Boot-Record wird geladen und gestartet. Dadurch bootet der Rechner neu.

9 Interrupt 1Ah: Zugriff auf Systemuhr

9.1 Funktion 00h: Zeitzähler lesen

Eingabe: **AH**=00h

Ausgabe: **CX**=Zeitzähler High
DX=Zeitzähler Low
AL=24-Stunden-Zähler

- Der Zeitzähler wird 18,2 mal je Sekunde um 1 erhöht. Der 32-Bit-Wert dividiert durch 18,2 ergibt also die Zahl der vergangenen Sekunden seit dem letzten Booten des Rechners. Durch die nachfolgende Funktion kann der Zeitzähler allerdings auch manipuliert werden.
- Der 24-Stunden-Zähler gibt Aufschluss über einen Überlauf des Zeitzählers. Er zählt die 24-Stunden-Überläufe mit, gibt also an, seit wieviel Tagen der Rechner eingeschaltet ist.

9.2 Funktion 01h: Zeitzähler setzen

Eingabe: **AH**=01h
CX=Zeitzähler High
DX=Zeitzähler Low

Ausgabe: keine

9.3 Funktion 02h: RTC-Uhr auslesen

Eingabe: **AH**=02h

Ausgabe: **Carry-Flag** zeigt Fehler an, ansonsten:
CH=Stunde
CL=Minute
DH=Sekunde

- Die Werte sind nur dann aussagekräftig, wenn das **Carry-Flag gelöscht** ist. Ein **gesetztes Carry-Flag** zeigt einen Fehler an, z.B. „Batterie ist leer“.
- Die Werte für Stunde, Minute und Sekunde sind **BCD**-codiert!

9.4 Funktion 03h: RTC-Uhr einstellen

Eingabe: **AH**=03h
CH=Stunde
CL=Minute
DH=Sekunde

Ausgabe: **Carry-Flag** zeigt ggf. Fehler an.

- Die Werte sind nur dann aussagekräftig, wenn das **Carry-Flag gelöscht** ist. Ein **gesetztes Carry-Flag** zeigt einen Fehler an, z.B. „Batterie ist leer“.
- Die Werte für Stunde, Minute und Sekunde müssen **BCD**-codiert abgelegt werden!

9.5 Funktion 04h: RTC-Datum auslesen

Eingabe: **AH**=04h

Ausgabe: **Carry-Flag** zeigt Fehler an, ansonsten:
CH=Jahrhundert
CL=Jahr (zweistellig)
DH=Monat
DL=Tag

- Die Werte sind nur dann aussagekräftig, wenn das **Carry-Flag gelöscht** ist. Ein **gesetztes Carry-Flag** zeigt einen Fehler an, z.B. „Batterie ist leer“.
- Die Werte für Jahrhundert, Jahr, Monat und Tag sind **BCD**-codiert!

9.6 Funktion 05h: RTC-Datum setzen

Eingabe: **AH**=05h
CH=Jahrhundert
CL=Jahr (zweistellig)
DH=Monat
DL=Tag

Ausgabe: **Carry-Flag** zeigt ggf. Fehler an.

- Die Werte sind nur dann aussagekräftig, wenn das **Carry-Flag gelöscht** ist. Ein **gesetztes Carry-Flag** zeigt einen Fehler an, z.B. „Batterie ist leer“.
- Die Werte für Jahrhundert, Jahr, Monat und Tag müssen **BCD**-codiert abgelegt werden!
- ***Achtung!* Für Jahrhundert ist nur 19 oder 20 möglich.**

9.7 Funktion 06h: RTC-Alarmzeit setzen

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

Ausgabe: **Carry-Flag** zeigt ggf. Fehler an

- Der Alarm wurde nur gesetzt, wenn das **Carry-Flag gelöscht** ist. Ein **gesetztes Carry-Flag** zeigt einen Fehler an. In diesem Fall ist entweder schon eine Alarmzeit programmiert oder ein anderer Fehler aufgetreten.
- Es kann jeweils nur **eine einzige Alarmzeit** aktiv sein.
- Die Werte für Stunde, Minute und Sekunde müssen **BCD**-codiert abgelegt werden!

9.8 Funktion 07h: RTC-Alarmzeit löschen

Eingabe: **AH**=07h

Ausgabe: **Carry-Flag** zeigt ggf. Fehler an

10 Interrupt 1Ch: Timer-Folge-Interrupt

Eingabe: keine **Ausgabe:** keine

Der Interrupt wird **automatisch** (also **nicht** durch mit Aufruf durch eine Anwendersoftware) 18,2 mal je Sekunde aufgerufen. In diesen Interrupt können eigene Routinen eingehängt werden, die automatisch regelmäßig wiederkehrend ablaufen sollen.

Dieser Interrupt wird immer im Anschluss an den eigentlichen **Timer-Interrupt 08h** automatisch aufgerufen. Da er dem **Timer-Interrupt** stets folgt, heißt er **Timer-Folge-Interrupt**.