Software-Interrupts ver. 0.41

Gleich zu Beginn der Beschreibungen der integrierten Softwareinterrupts eine Klarstellung (Zitat Wikipedia):

> *Ein Software-Interrupt ist ein expliziter Aufruf einer Unterfunktion (meist einer Betriebssystem-Funktion). Er hat nichts mit einem Interrupt (asynchrone Unterbrechung) zu tun.*

Dieses Zitat beschreibt es sehr treffend. Natürlich hat der CP1+ Clone kein Betriebssystem im klassischen Sinne, aber er besitzt Funktionen, die aus einem laufenden Programm heraus aufgerufen werden können. Viele dieser Funktionen wurden sogar extra dafür geschrieben.

Wenn man möchte könnte man die Funktionen der Software-Interrupts als „Bios“ (Basic-Input-Output-System = grundsätzliches Ein- Ausgabesystem) bezeichnen, da es sich um genau solche Funktionen handelt.

Der CP1+ Clone verwendet folgendes Schema um Software-Interrupts ausführen zu können:

Mnemonic:

- INT Interruptnummer

Registerbelegung:

- Akku und Flags um einen Datumstransfer von oder zum Software-Interrupt vornehmen zu können
- Register B um eine Unterfunktion des Software-Interrupts zu wählen.

Verfügbare Software-Interrupts sind:

Software-Interrupt	Funktion
INT 1	Verzögerungszeiten
INT 2	16-Bit PWM - Frequenzgenerator
INT 3	8-Bit PWM - Frequenzgenerator
INT 4	PWM – Frequenzgenerator für Servomotor
INT 8	Analog – Digital – Wandler
INT 9	Tastatur / Keyboard
INT 10	Displaysteuerung

Da der CP1+ abweichend von üblichen Prozessoren Zahlen in dezimaler Notation darstellt, sind alle Zahlen als Dezimalzahlen zu verstehen (sofern nicht explizit ein anderes Zahlenformat angegeben wird).

Beschreibung der Software-Interrupts

INT 1 : Verzögerungsschleifen

Registerbelegung:

Akku	Multiplikator der Zeitbasis
Register B	Zeitbasis

Register B	Zeitbasis
1	1 [ms]
2	10 [ms]
3	100 [ms]
4	1 [s]

INT 1 stellt flexible Warteschleifen zur Verzögerung von Programmabläufen zur Verfügung. INT 1 besitzt keine weiteren Unterfunktionen.

Der Programmablauf wird um den Wert im Akku multipliziert mit der gewählten Zeitbasis in Register B verzögert.

Beispiel:

```
; Verzoegerungsschleife fuer 2,5 Sekunden
mvi a, 25
mvi b, 3   ; 25 * 0,1s = 2,5 s
int 1
```

INT 2 : 16-Bit PWM Frequenzgenerator

Registerbelegung:

Akku	Frequenz in Hz
Register B	Frequenzbasis
Register C	Tastgrad in %

Register B	Frequenzbasis
0	1 [Hz]
1	10 [Hz]
2	100 [Hz]
3	1 [kHz]

INT 2 stellt einen 16-Bit Frequenzgenerator zur Verfügung. INT 2 besitzt keine weiteren Unterfunktionen.

Die Frequenz des PWM erfolgt durch die Multiplikation des Akkus mit der Zeitbasis in Register B. Der Tastgrad (Duty-Cylcle) der PWM wird durch Register C gewählt, diese Angabe erfolgt in Prozent. Die maximal einstellbare Frequenz ist 65 kHz.

Wird eine Frequenz größer als 65 kHz angewählt, wird der **Fehlercode 22** auf dem Display ausgegeben und das gesamte Programm angehalten.

Die PWM-Frequenz ist am Anschluß P1.3 verfügbar.

Hinweis: Der Tastgrad entspricht dem Verhältnis von Pulsdauer / Periodendauer.

Beispiel:

```
; PWM 20 kHz mit 25 % Tastgrad
mvi a, 200
mvi b, 2      ; 200 x 100 Hz = 20000 Hz
mvi c, 25     ; 25 % Tastgrad
int 2
```

INT 3 : 8-Bit PWM Frequenzgenerator

Registerbelegung:

Akku	Frequenz in Hz
Register B	Frequenzbasis
Register C	Tastgrad in %

Register B	Frequenzbasis
0	1 [Hz]
1	10 [Hz]
2	100 [Hz]
3	1 [kHz]

INT 3 stellt einen 8-Bit Frequenzgenerator zur Verfügung. INT 3 besitzt keine weiteren Unterfunktionen.

Die Frequenz des PWM erfolgt durch die Multiplikation des Akkus mit der Zeitbasis in Register B. Der Tastgrad (Duty-Cylcle) der PWM wird durch Register C gewählt, diese Angabe erfolgt in Prozent. Die maximal einstellbare Frequenz ist 65000 Hz. Die niedrigst einstellbare Frequenz ist 63 Hz.

Wird eine Frequenz größer als 65 kHz angewählt, wird der **Fehlercode 22**, bei einer Frequenzanwahl kleiner als 63 Hz der **Fehlercode 23** auf dem Display ausgegeben und das gesamte Programm angehalten.

Die PWM-Frequenz ist am Anschluß P2.5 verfügbar.

Hinweis: Der Tastgrad entspricht dem Verhältnis von Pulsdauer / Periodendauer.

Beispiel:

```
; PWM 20 kHz mit 25 % Tastgrad
mvi a, 200
mvi b, 2      ; 200 x 100 Hz = 20000 Hz
mvi c, 25     ; 25 % Tastgrad
int 3
```

INT 4 : PWM-Generator Servomotor

Registerbelegung:

Akku	Stellwinkelwert (0..255) des Servomotors

INT 4 stellt die für einen Servomotor notwendig PWM-Steuersignale zur Verfügung. Die PWM-Frequenz ist hierbei fest auf 50 Hz (= 20 ms Periodendauer) eingestellt. Der Tastgrad (Verhältnis von Pulsdauer / Periodendauer) darf hierbei von 3% bis 13% variieren. Dieses entspricht einem Vollausschlag von einem Anschlagpunkt zum anderen.

INT 4 stellt dieses mittels eines 8-Bit Wertes ein, hierbei wird die gesamte mögliche Auflösung des 8-Bit Akkus genutzt. Ein Akkuwert von 0 stellt den Anschlag links ein, ein Akkuwert von 255 den Anschlagwert rechts. Jeder Wert von 0 bis 255 entspricht somit einer Position des Servomotors (der somit 255 unterschiedliche Positionen einnehmen kann). Der Servomotorsteueranschluß ist an P1.3 verfügbar.

INT 4 stellt keine weiteren Unterfunktionen zur Verfügung.

Beispiel:

```
; Stellwinkel Servomotor einstellen in Abhaengigkeit von einer
; an P2.3 anliegenden analogen Spannung

Addr.  Opcode   Mnemonic

                .org 0
000:   04.003   mvi a,003
001:   28.000   mvi b,000
002:   63.008   int 008     ; Analogeingang auf P2.3 waehlen
003:   03.002   cdel 002    ; kurze Wartezeit fuer den ADC
; loop:
004:   28.001   mvi b,001
005:   63.008   int 008     ; INT8, Func. 1 : 8-Bit Analogwert lesen
006:   02.000   cdis        ; und anzeigen
007:   63.004   int 004     ; diesen Wert als Stellvorgabe Servomotor
008:   03.100   cdel 1      ; Wartezeit
009:   09.004   jmp 004     ; fuer immer wiederholen (loop)
```

INT 8 : Analog - Digital - Wandler

INT 8 macht dem CP1+ System einen Analog - Digital -Wandler verfügbar. Für INT 8 existieren insgesamt 4 Unterfunktionen, die über Register B ausgewählt werden.

Register B	Funktion
0	Initialisierung des AD-Wandlers und Wahl des analogen Eingangskanals
1	einlesen der gewählten Meßstelle und Rückgabe des Meßwertes als 8-Bit Wert (0 bis 255)
2	einlesen der gewählten Meßstelle und Rückgabe des Meßwertes als Prozentangabe des Maximalwertes (0 bis 100)
3	einlesen der gewählten Meßstelle und Rückgabe des Meßwertes als Spannungswert mit einer Maximalspannung von 5V, aufgelöst in 100 mV Schritten. (0 bis 50)
10	einlesen eines NTC-Widerstandes an der gewählten Meßstelle mit der Rückgabe der Temperatur in °C im Akku.

INT8 - Funktion 0 :

Registerbelegung:

Register B	Unterfunktionsnummer, muß den Wert 0 enthalten
Akku	Wählt den analogen Eingangskanal. Es stehen insgesamt 4 Eingangskanäle zur Verfügung die mit einem Akkuinhalt von 0 .. 3 ausgewählt werden. Der Akkuinhalt korrespondiert mit den Portpins P2.0 bis P2.3

Initialisierung des AD - Wandlers. Festlegung des Eingangskanals für die analoge Spannungsmessung. Dieser Kanal wird so lange für das Erfassen analoger Spannungen verwendet, bis diese mit der Funktion 0 geändert wird.

INT8 - Funktion 1 :

Registerbelegung:

Register B	Unterfunktionsnummer, muß den Wert 1 enthalten
Akku	wird bei Eintritt in die Funktion 1 nicht ausgewertet, enthält bei Beendigung der INT8 - Funktion 1 das Ergebnis der AD-Wandlung

liest einen analogen Spannungswert auf einem zuvor initialisierten Eingangskanal ein. Das Ergebnis der Analog-Digital-Wandlung wird als 8-Bit Binärwert (0 - 255) im Akku zurückgegeben.

INT8 - Funktion 2 :

Registerbelegung:

Register B	Unterfunktionsnummer, muß den Wert 2 enthalten
Akku	wird bei Eintritt in die Funktion 1 nicht ausgewertet, enthält bei Beendigung der INT8 - Funktion 1 das Ergebnis der AD-Wandlung

liest einen analogen Spannungswert auf einem zuvor initialisierten Eingangskanal ein. Das Ergebnis der Analog-Digital-Wandlung wird als Prozentwert der Maximalspannung (Wert 0 - 99) im Akku zurückgegeben.

INT8 - Funktion 3 :

Registerbelegung:

Register B	Unterfunktionsnummer, muß den Wert 3 enthalten
Akku	wird bei Eintritt in die Funktion 1 nicht ausgewertet, enthält bei Beendigung der INT8 - Funktion 1 das Ergebnis der AD-Wandlung

liest einen analogen Spannungswert auf einem zuvor initialisierten Eingangskanal ein. Das Ergebnis der Analog-Digital-Wandlung wird als Spannungswert mit einer Maximalspannung von 5 V mit einer Auflösung von 100 mV im Akku (Werte 0 - 50) zurückgegeben. Als Referenzspannung dient die eigene 5 V Versorgungsspannung, hier wird von einer hinlänglich guten Konstanz und Genauigkeit ausgegangen. Das Ergebnis der AD-Wandlung kann deshalb nur so gut wie die Versorgungsspannung sein.

Beispiel:

```
; Spannungsmessung an P2.2, zwischen Betriebsspannung und Masse
; ist bspw. ein Trimmer angeschlossen mit Mittelabgriff an P2.2

Addr.  Opcode   Mnemonic

               .org 0
000:   28.000   mvi b,000
001:   04.002   mvi a,002
002:   63.008   int 008       ; initialisieren ADC auf Kanal 2 / P2.2
003:   03.002   cdel 002      ; Wartezeit fuer ADC
; loop:
004:   28.002   mvi b,002
005:   63.008   int 008       ; AD-Wandlung mit Prozentausgabe in A
006:   02.000   cdis          ; Wert anzeigen
007:   03.100   cdel 100      ; Wartezeit
008:   09.004   jmp 004       ; endlos wiederholen (loop)
```

INT8 - Funktion 10 :

Registerbelegung:

Register B	Unterfunktionsnummer, muß den Wert 10 enthalten
Akku	wird bei Eintritt in die Funktion 1 nicht ausgewertet, enthält bei Beendigung der INT8 - Funktion 10 den dezimalen Wert der gemessenen Temperatur in °C
Less-Flag	0 bei positiven Temperaturen, 1 bei negativen Temperaturen

auf dem aktuell gewählten analogen Eingangskanal wird der Spannungswert eines Spannungsteilers bestehend aus 10 KΩ Pullup-Widerstand und 10 KΩ NTC-Widerstandes (B25/80 = 3950) gemessen und in einen Temperaturwert umgerechnet. Die Temperatur wird im Akku zurückgegeben, bei negativen Temperaturen wird das Less-Flag gesetzt.

Beispiel:

```
; Spannungsmessung an P2.2, zwischen Betriebsspannung und Masse
; ist bspw. ein Trimmer angeschlossen mit Mittelabgriff an P2.2

Addr.  Opcode   Mnemonic

               .org 0
000:   28.010   mvi b,010
001:   63.008   int 008       ; INT 8, Funktion 10: Temperaturmessung
002:   51.006   jpl 006       ; bei negativer Temperatur, Program anhalten
003:   02.000   cdis          ; Temperatur anzeigen
004:   03.030   cdel 030      ; warten
005:   09.000   jmp 000       ; und wiederholen
006:   01.000   hlt
```

INT 9 : Tastatur / Keyboard

INT 9 ermöglicht das Auslesen der Tastatur. Für INT 9 existieren insgesamt 3 Unterfunktionen, die über Register B ausgewählt werden.

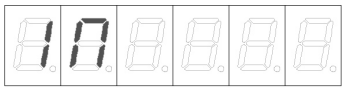
Register B	Funktion
0	einlesen eines 8-Bit dezimalen Zahlenwertes
1	warten auf einen Tastendruck
2	ermitteln, ob eine Taste gedrückt ist (oder nicht)

INT9 - Funktion 0 :

Registerbelegung:

Register B	Unterfunktionsnummer, muß den Wert 0 enthalten
Akku	wird bei Eintritt in die Funktion 0 nicht ausgewertet, enthält bei Beendigung der INT9 - Funktion 0 die eingelesene dezimale Zahl

liest einen maximal 3 stelligen dezimalen Zahlenwert im Bereich von einschließlich 0 bis einschließlich 255 ein. Nach Aufruf von INT9 Funktion 0 erscheint auf dem Display als Aufforderung zur Zahleneingabe:



Eine Zahleneingabe wird mit der Tastenkombination **> Shift - INP <** (als Ersatz für ein Enter) abgeschlossen. Hat die Zahleneingabe einen größeren Wert als 255 erscheint eine Fehlermeldung und es wird erneut eine Zahleneingabe verlangt. Dieser Vorgang wiederholt sich solange, bis eine gültige Zahl eingegeben wurde.

Nach erfolgter Zahleneingabe wird die eingegebene Zahl auf dem Display angezeigt und sie bleibt dort so lange angezeigt, bis auf dem Display andere Ausgaben erfolgen.

Beispiel:

```
; Countdown einer eingegebenen Zahl

Addr.  Opcode   Mnemonic

                .org 0
000:   28.000   mvi b,000
001:   63.009   int 009      ; Interrupt 9, Funktion 0: Zahl einlesen
002:   03.250   cdel 250     ; 1/4 Sekunde Wartezeit nach einlesen
; loop:
003:   02.000   cdis         ; Anzeige
004:   03.250   cdel 250     ; 4 x 1/4 Sekunde warten
005:   03.250   cdel 250
006:   03.250   cdel 250
007:   03.250   cdel 250
008:   25.003   djnz a,003   ; A= A-1, wenn nicht 0 dann wiederholen
009:   01.000   hlt          ; bei Erreichen von 0 Programm anhalten
```

INT9 - Funktion 1 :

Registerbelegung:

Register B	Unterfunktionsnummer, muß den Wert 1 enthalten
Akku	wird bei Eintritt in die Funktion 1 nicht ausgewertet, enthält bei Beendigung der INT9 - Funktion 1 den Wert der gedrückten Taste

wartet auf einen Tastendruck und gibt danach den Wert der gedrückten Taste im Akku zurück.

Beispiel:

```
; Countdownstart nach Tastendruck

Addr.  Opcode   Mnemonic

                .org 0
000:   04.010   mvi a,010     ; Countdown Startwert
001:   39.000   mov e,a       ; Startwert nach Register E
002:   02.000   cdis          ; anzeigen
003:   28.001   mvi b,001
004:   63.009   int 009       ; Interrupt 9, Funktion 1: auf Taste warten
005:   35.000   mov a,e       ; Countdownwert wieder in den Akku
; loop:
006:   02.000   cdis          ; anzeigen
007:   03.250   cdel 250      ; 2 x 1/4 Sekunde warten
008:   03.250   cdel 250
009:   25.006   djnz a,006    ; A= A-1, wenn nicht 0 dann wiederholen
010:   01.000   hlt           ; bei Erreichen von 0 Programm anhalten
```

INT9 - Funktion 2 :

Registerbelegung:

Register B	Unterfunktionsnummer, muß den Wert 2 enthalten
Akku	wird bei Eintritt in die Funktion 0 nicht ausgewertet, enthält bei Beendigung der INT9 - Funktion 2 bei gedrückter Taste den Wert der Taste. Wurde keine Taste gedrückt wird im Akku der Wert 255 zurückgegeben
Zero-Flag	ist eine Taste gedrückt, wird das Zero-Flag zu 1, andernfalls zu 0

stellt fest, ob eine Taste gedrückt ist. Bei gedrückter Taste wird das Zero-Flag gesetzt und der Wert der Taste im Akku zurückgegeben. Ist keine Taste gedrückt, wird der Akku mit dem Wert 255 beschrieben und das Zero-Flag ist gelöscht.

Beispiel:

```
; Zaehler, der bei Tastendruck gestoppt wird

Addr.  Opcode   Mnemonic

                .org 0
000:   04.000   mvi a,000     ; Startzahl
; loop:
001:   02.000   cdis          ; anzeigen
002:   39.000   mov e,a       ; Zaehlerstand in Register E speichern
003:   28.002   mvi b,002
004:   63.009   int 009       ; INT9 Funktion 2: Test auf gedrueckte Taste
005:   11.010   jz 010        ; bei gedrueckter Taste verzweigen
006:   03.100   cdel 100      ; Warteschleife des Zaehlers
007:   35.000   mov a,e       ; Zaehlerstand wieder in den Akku
008:   26.000   inc a         ; und um eins erhoehen
009:   09.001   jmp 001
010:   09.010   jmp 010       ; Programm in Endlosschleife
```

INT 10 : Display

INT 10 ermöglicht den Zugriff auf die 7-Segmentanzeiger. Für INT 10 sind folgende Unterfunktionen vorhanden, die über Register B ausgewählt werden.

Register B	Funktion
0	maskieren der anzuzeigenden Digits
1	dezimale Anzeige des Akkuinhalts ohne Dezimalpunkt
2	16-Bit dezimale Anzeige des Registerpaares C - D
3	hexadezimale Anzeige des Akkuinhalts
4	16-Bit hexadezimale Anzeige des Registerpaares C - D
5	binäre Anzeige des Akkuinhalts
6	User-Bitmap auf einem Digit anzeigen
7	Dezimale Anzeige Akkuinhalt mit Positionsangabe
20	maskieren der Dezimalpunkte
21	einzelnen Dezimalpunkt setzen
22	einzelnen Dezimalpunkt löschen

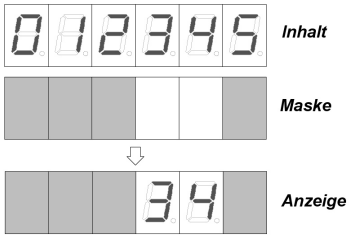
INT10 - Funktion 0 :

Registerbelegung:

Register B	Unterfunktionsnummer, muß den Wert 0 enthalten
Akku	Digitmaske der anzuzeigenden 7-Segment Digits

Unter "maskieren" versteht man die gezielte Auswahl einer Information aus einer Fülle von Informationen. Nur relevante Informationen werden weiter verarbeitet, unrelevantes wird nicht bearbeitet.

In Bezug auf die Digits der 7-Segmentanzeige versteht man unter „maskieren“ das scheinbare Abdecken der Anzeige mittels einer virtuellen Displaymaske:



Angenommener Displayinhalt im Bild links wäre "012345". Diese wird mit einer Abdeckung abgedeckt, die aber an den Digits 3 und 4 (Zählrichtung von links nach rechts mit 0 beginnend) durchsichtig ist. Auf der Anzeige wäre dann nur die im Bild gezeigte Darstellung zu sehen.

Die binäre Kombination von 1 und 0 im Akku bestimmt nun, welche Digits zur Anzeige kommen und welche nicht.

Beispiele (ein "-" bedeutet: Digit dunkel geschaltet)

Zeichen ohne Maske	012345	012345	187	E01234	218
Akkumaske	6 = 000110	15 = 001111	2 = 000010	63 = 111111	4 = 000100
tatsächliche Anzeige	---34-	--2345	----8-	E01234	---2--

Nach der Ausführung von INT10 Funktion 0 werden alle darüber selektierten Digits mit einer 0 solange angezeigt, bis eine Ausgabe über eine der INT10 Funktionen 1 bis 5 erfolgt.

Ein Aufruf des Maschinenbefehls CDIS löscht diese Maske wieder und es erfolgt die CDIS-Standarddarstellung auf der Anzeige (dies entspricht der Standardmaske mit Wert 7, die rechten 3 Digits werden angezeigt).

Wird versucht, eine Displaymaske mit einem Zahlenwert > 63 einzustellen, wird das Programm mit dem Fehlercode 35 gestoppt.

Beispielprogramm:

```
; Demo der Displaymaskierung

Addr.  Opcode   Mnemonic

                .org 0
000:   04.063   mvi a,063     ; Displaymaske alle Digits an
001:   28.000   mvi b,000
002:   63.010   int 010       ; Interrupt 10, Funktion 0 : Displaymaske

; High-Byte = 48, Low-Byte = 57 entsprechen dezimal 12345
003:   29.048   mvi c,048     ; High-Byte von 123456
004:   30.057   mvi d,057     ; Low-Byte von 123456
005:   28.002   mvi b,002
006:   63.010   int 010       ; Interrupt 10, Funktion 2 : 16-Bit dezimal
                              ; anzeigen
007:   03.250   cdel 250      ; Wartezeit um die Anzeige zu sehen
008:   03.250   cdel 250
009:   03.250   cdel 250
010:   04.006   mvi a,006     ; Displaymaske fuer Digit 3 und 4 an
011:   28.000   mvi b,000
012:   63.010   int 010       ; Interrupt 10, Funktion 0 : Displaymaske
013:   29.048   mvi c,048
014:   30.057   mvi d,057
015:   28.002   mvi b,002
016:   63.010   int 010        ; wie in Programmzeile 6
017:   09.017   jmp 017        ; auf dem Display ist "---34-" zu sehen
```

INT10 - Funktion 1 :

Registerbelegung:

Register B	Unterfunktionsnummer, muß den Wert 1 enthalten
Akku	anzuzeigender dezimaler 8-Bit Zahlenwert (0 - 255)

zeigt eine 8-Bit dezimale Zahl auf der 7-Segmentanzeige ohne Dezimalpunktsteuerung. Dezimalpunkte können mittels der Funktionen 20 bis 22 des Interrupts gesteuert werden.

Beispielprogramm:

```
; 2 stellige Zahlenausgabe

Addr.  Opcode   Mnemonic

                .org 0
000:   04.003   mvi a,003     ; Displaymaske 2-stellig
001:   28.000   mvi b,000
002:   63.010   int 010       ; Displaymaske setzen
003:   04.099   mvi a,099     ; auszugebender Zahlenwert
004:   28.001   mvi b,001
005:   63.010   int 010       ; anzeigen
006:   09.006   jmp 006       ; für immer
```

INT10 - Funktion 2 :

Registerbelegung:

Register B	Unterfunktionsnummer, muß den Wert 2 enthalten
Register C	High-Byte des anzuzeigenden dezimalen 16-Bit Zahlenwertes (0 - 65535)
Register D	Low-Byte des anzuzeigenden dezimalen 16-Bit Zahlenwertes

zeigt eine 16-Bit dezimale Zahl auf der 7-Segmentanzeige ohne Dezimalpunktsteuerung. Die höherwertigen 8 Bits werden in Register C, die niederwertigen Bits in Register D übergeben.

Beispielprogramm:

```
; Ausgabe der Zahl 14923

Addr.  Opcode   Mnemonic

                .org0
000:   04.031   mvi a,031     ; Displaymaske 5-stellig
001:   28.000   mvi b,000
002:   63.010   int 010       ; Interrupt 10, Funktion 0: Displaymaske
003:   28.002   mvi b,002
004:   29.058   mvi c,058     ; High-Byte von 14923
005:   30.075   mvi d,075     ; Low-Byte von 14923
006:   63.010   int 010       ; INT 10, Funktion 2: 16-Bit Zahl anzeigen
007:   09.007   jmp 007       ; für immer
```

INT10 - Funktion 3 :

Registerbelegung:

Register B	Unterfunktionsnummer, muß den Wert 3 enthalten
Akku	anzuzeigender hexadezimaler 8-Bit Zahlenwert (0 - 0FFh)

zeigt eine 8-Bit hexadezimale Zahl auf der 7-Segmentanzeige ohne Dezimalpunktsteuerung. Zur Kennzeichnung der Ausgabe als hexadezimalen Wert wird auf dem Display links ein kleines "h" mit ausgegeben. Dezimalpunkte können mittels der Funktionen 20 bis 22 des Interrupts gesteuert werden.

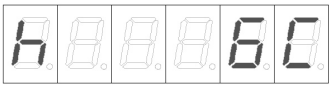
Beispielprogramm:

```
; 2 stellige hexadezimale Zahlenausgabe

Addr.  Opcode   Mnemonic

                .org 0
000:   04.003   mvi a,003     ; Displaymaske 2-stellig
001:   28.000   mvi b,000
002:   63.010   int 010       ; INT 10, Funktion 0: Displaymaske
003:   04.108   mvi a,108     ; auszugebender Zahlenwert
004:   28.001   mvi b,003
005:   63.010   int 010       ; anzeigen
006:   09.006   jmp 006       ; für immer
```

Displayausgabe:



Hinweis: 6C hex entspricht 108 dez

INT10 - Funktion 4 :

Registerbelegung:

Register B	Unterfunktionsnummer, muß den Wert 4 enthalten
Register C	High-Byte des anzuzeigenden hexadezimalen 16-Bit Zahlenwertes (0 - 0FFFFh)
Register D	Low-Byte des anzuzeigenden hexadezimalen 16-Bit Zahlenwertes

zeigt eine 16-Bit hexadezimale Zahl auf der 7-Segmentanzeige ohne Dezimalpunktsteuerung. Zur Kennzeichnung der Ausgabe als hexadezimalen Wert wird auf dem Display links ein kleines "h" mit ausgegeben. Dezimalpunkte können mittels der Funktionen 20 bis 22 des Interrupts gesteuert werden.

Beispielprogramm:

```
; 4-stellige hexadezimale Zahlenausgabe von 3A4Bh

Addr.  Opcode   Mnemonic

                .org0
000:   04.047   mvi a,047     ; Displaymaske fuer 4-stellige Hex-Ausgabe
001:   28.000   mvi b,000
002:   63.010   int 010       ; INT 10, Funktion 0: Displaymaske
003:   28.004   mvi b,004
004:   29.058   mvi c,058     ; High-Byte: 3A hex = 58 dez
005:   30.075   mvi d,075     ; Low-Byte : 4B hex = 75 dez
006:   63.010   int 010       ; INT 10, Funktion 4: Hexadezimalausgabe
007:   09.007   jmp 007
```

Displayausgabe:

INT10 - Funktion 5 :

Registerbelegung:

Register B	Unterfunktionsnummer, muß den Wert 5 enthalten
Akku	anzuzeigender binärer 6-Bit Zahlenwert (0 - 11.1111b)

zeigt eine 6-Bit binäre Zahl auf der 7-Segmentanzeige ohne Dezimalpunktsteuerung. Werden weniger als 6 Bit ausgegeben, erscheint auf dem Display links ein kleines "b" zur Kennzeichnung der Ausgabe als binären Wert. Dezimalpunkte können mittels der Funktionen 20 bis 22 des Interrupts gesteuert werden.

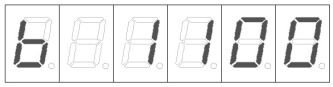
Beispielprogramm:

```
; Binärausgabe der Zahl 12 dez = 1100 bin

Addr.  Opcode   Mnemonic

               .org 0
000:   28.000   mvi b,000
001:   04.047   mvi a,047     ; linkes Digit und 4 Digits rechts
002:   63.010   int 010       ; INT 10, Funktion 0: Displaymaske
003:   28.005   mvi b,005
004:   04.012   mvi a,012     ; 12 dez = 1100 bin
005:   63.010   int 010       ; INT 10, Funktion 5: binäre Displayausgabe
006:   09.006   jmp 006       ; fuer immer wiederholen
```

Displayausgabe:

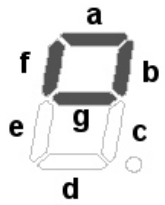


INT10 - Funktion 6 :

Registerbelegung:

Register B	Unterfunktionsnummer, muß den Wert 6 enthalten
Akku	auszugebendes 7-Segment Bitmap
Register C	Ausgabeposition an der das Bitmap ausgegeben wird

unabhängig davon, ob die Displaymaske ein Digit aktiv geschaltet hat oder nicht, wird an der angegebenen Position in Register D ein Bitmap zur Anzeige gebracht. Bit 0 des Akkus korrespondiert mit dem Segment A, Bit 7 korrespondiert mit dem Segment G der Anzeige. Ein gesetztes Bit im Akku schaltet das Segment ein.



Segment	-	g	f	e	d	c	b	a
Akku Bitnummer	7	6	5	4	3	2	1	0
Akku	0	1	1	0	0	0	1	1

Der Akkuinhalt für das dargestellte Bitmap beträgt `0110.0011 bin = 99 dez`

Beispielprogramm:

```
; "HALLO" auf Display anzeigen

Addr.  Opcode   Mnemonic

               .org 0
000:   28.000   mvi b,000
001:   04.000   mvi a,000     ; Maske alle Digits aus
002:   63.010   int 010       ; INT 10, Funktion 0: Displaymaske
003:   28.006   mvi b,006     ; Funktion 6
004:   04.118   mvi a,118     ; Bitmap fuer "H"
005:   29.001   mvi c,001     ; Digit-Position 1
006:   63.010   int 010       ; INT 10, Funktion 6: Bitmapanzeige
007:   04.119   mvi a,119     ; Bitmap fuer "A"
008:   29.002   mvi c,002     ; Digit-Position 2
009:   63.010   int 010       ; anzeigen
010:   04.056   mvi a,056     ; Bitmap fuer "L"
011:   29.003   mvi c,003     ; Digit-Position 3
012:   63.010   int 010       ; anzeigen
013:   29.004   mvi c,004     ; Digit-Position 4
```

```
014:   63.010   int 010       ; anzeigen
015:   04.063   mvi a,063     ; Bitmap fuer "O"
016:   29.005   mvi c,005     ; Digit-Position 5
017:   63.010   int 010       ; anzeigen
018:   09.018   jmp 018       ; fuer immer
```

Displayausgabe:

Beispielprogramm 2:

```
; Segmentlauflicht auf Digit 5

Addr.  Opcode   Mnemonic

                .org 0
000:   04.000   mvi a,000     ; Displaymaske alle aus
001:   28.000   mvi b,000
002:   63.010   int 010       ; INT 10, Funktion 0: Displaymaske
003:   04.001   mvi a,001     ; Lauflichtstart Segment a
004:   28.006   mvi b,006     ; Funktion 6
005:   29.005   mvi c,005     ; Position 5
006:   63.010   int 010       ; INT 10, Funktion 6: Bitmapausgabe
007:   03.128   cdel 128      ; warten
008:   54.000   slc a         ; Akku um eine Position nach links

; bei Erreichen Segment g dieses nicht anzeigen
; stattdessen Akku wieder auf 1 setzen

009:   49.064   cpi a,064     ; 64 = Segment g
010:   11.003   jz 003

; ansonsten geschobenen Akku anzeigen
011:   09.004   jmp 004
```

INT10 - Funktion 7 :

Registerbelegung:

Register B	Unterfunktionsnummer, muß den Wert 7 enthalten
Akku	anzuzeigender dezimaler 8-Bit Zahlenwert (0 - 255)
Register C	Ausgabeposition auf der Anzeige (0 - 3)

gibt eine 8-Bit dezimale Zahl auf der 7-Segmentanzeige ohne Dezimalpunktsteuerung an der in Register C angegebenen Anzeigeposition aus. Die Anzeigeposition 0 korrespondiert mit dem linken Digit der Anzeige. Registerwerte größer als 3 in Register C erzeugen einen Abbruch des gesamten Programms mit dem Fehlercode 37.

Dezimalpunkte können mittels der Funktionen 20 bis 22 des Interrupts gesteuert werden.

Beispielprogramm:

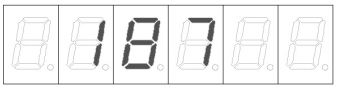
```
; 2 stellige Zahlenausgabe

Addr.  Opcode   Mnemonic

                .org 0
000:   28.000   mvi b,000
001:   04.028   mvi a,028     ; Displaymaske fuer Position 1,2,3
```

```
002:   63.010   int 010       ; INT 10 Funktion 0: Displaymaske
003:   04.187   mvi a,187     ; auszugebender Zahlenwert
004:   28.007   mvi b,007     ; Funktion 7
005:   29.001   mvi c,001     ; Position 1
006:   63.010   int 010       ; INT 10, Funktion 7: dez. Wert ausgeben
007:   09.007   jmp 007
```

Displayausgabe:



INT10 - Funktion 20 :

Registerbelegung:

Register B	Unterfunktionsnummer, muß den Wert 20 enthalten
Akku	Bitmap anzuzeigender Punkte

schaltet nach dem Bitmapmuster des Akkuinhaltes die Punktanzeigen der Digits an oder aus. Bit 0 des Akkus korrespondiert hier mit dem linken Dezimalpunkt der Anzeige, Bit 5 des Akkus korrespondiert mit dem rechten Dezimalpunkt der Anzeige. Wird dem Akku ein größerer Inhalt als 63 zugewiesen wird das gesamte Programm mit einem Fehlercode 35 beendet.

Dezimalpunkt	-	links						rechts
Akku Bitnummer	7	6	5	4	3	2	1	0
Akku	0	1	1	0	0	0	0	1

Der Akkuinhalt beträgt 0110.0001 bin = 49 dez und läßt die Punkte der beiden linksseitigen Digits sowie den Punkt des rechten Digits leuchten.

Beispielprogramm:

```
; Dezimalpunkte anzeigen

Addr.  Opcode   Mnemonic

               .org 0
000:   04.000   mvi a,000
001:   28.000   mvi b,000
002:   63.010   int 010       ; INT 10, Funktion 0: alle Digits aus
003:   04.049   mvi a,049     ; Bitmap der Punkte
004:   28.020   mvi b,020     ; Funktion 20
005:   63.010   int 010       ; INT 10: Punkte entspr. Bitmap anzeigen
006:   09.006   jmp 006       ; fuer immer
```

Beispielprogramm 2:

```
; Lauflicht der Punkte

Addr.  Opcode   Mnemonic

               .org 0
000:   04.000   mvi a,000
001:   28.000   mvi b,000
002:   63.010   int 010       ; INT 10, Funktion 0: alle Digits aus
003:   04.001   mvi a,001     ; Startwert Punktelauflicht
```

```
004:   28.020   mvi b,020
005:   63.010   int 010       ; INT 10, Funktion 20: Punkte anzeigen
006:   03.128   cdel 128      ; warten
007:   54.000   slc a         ; Punktanzeige verschieben
008:   49.064   cpi a,064     ; hoechster Wert erreicht?
009:   11.003   jz 003        ; dann von rechts neu beginnen
010:   09.004   jmp 004       ; mit geschobenem Punkt fortfahren
```

INT10 - Funktion 21 :

Registerbelegung:

Register B	Unterfunktionsnummer, muß den Wert 21 enthalten
Akku	Position des anzuzeigenden Punktes

schaltet an der im Akku angegebenen Position einen Digitpunkt an. Postionsnummer 0 ist, entgegen der Digitnummerierung, rechts. Positionsnummer 5 ist links.

Beispielprogramm:

```
; zeigt 4.3 auf der Anzeige an

Addr.  Opcode   Mnemonic

                .org 0
000:   04.003   mvi a,003     ; Displaymaske fuer 2-stellige Anzeige
001:   28.000   mvi b,000
002:   63.010   int 010       ; INT 10, Funktion 0: Displaymaske
003:   04.043   mvi a,043     ; auszugebender Zahlenwert
004:   28.001   mvi b,001
005:   63.010   int 010       ; INT 10, Funktion 1: dez. Wert ausgeben
006:   04.001   mvi a,001     ; Punktposition 2-te von rechts
007:   28.021   mvi b,021
008:   63.010   int 010       ; INT 10, Funktion 21: Punkt einschalten
009:   09.009   jmp 009       ; fuer immer
```

INT10 - Funktion 22 :

Registerbelegung:

Register B	Unterfunktionsnummer, muß den Wert 22 enthalten
Akku	Position des zu löschenden Punktes

schaltet an der im Akku angegebenen Position einen Digitpunkt aus. Postionsnummer 0 ist, entgegen der Digitnummerierung, rechts. Positionsnummer 5 ist links.

Beispielprogramm:

```
; zeigt 43 auf der Anzeige mit blinkedem Punkt dazwischen an

Addr.  Opcode   Mnemonic

000:   04.003   mvi a,003     ; Displaymaske fuer 2-stellige Anzeige
001:   28.000   mvi b,000
002:   63.010   int 010       ; INT 10, Funktion 0: Displaymaske
003:   04.043   mvi a,043     ; auszugebender Zahlenwert
004:   28.001   mvi b,001
005:   63.010   int 010       ; INT 10, Funktion 1: dez. Wert ausgeben
```

```
; loop:
006:   04.001   mvi a,001     ; Punktposition 2-te von rechts
007:   28.021   mvi b,021
008:   63.010   int 010       ; INT 10, Funktion 21: Punkt einschalten
009:   03.255   cdel 255      ; warten
010:   28.022   mvi b,022
011:   63.010   int 010       ; INT 10, Funktion 22: Punkt loeschen
012:   03.255   cdel 255      ; warten
013:   09.006   jmp 006      ; an - aus fuer immer wiederholen
```