6502 Programmieren - Teil 12#

von Uwe Röder#

Hallo Freaks !!!

In diesem Teil unseres Kurses werde ich Ihnen zeigen, wie die Ausgabe von Zeichen und Texten auf dem Bildschirm mit Hilfe des Editors realisiert wird.

Ich benutze dazu einige Routine von der Bibo-Assembler Tooldisk 1. Damit auch Sie mit diesen Routinen arbeiten können, habe ich sie auf der Magazin-Disk unter dem Namen AS12.ASM im Bibo-Assembler-Format abgelegt.

Um irgendeinen Buchstaben auf dem Bildschirm auszugeben, kann man natürlich, wie für alle anderen Ein- und Ausgabeoperationen, die CIO benutzen. Direkt nach dem Einschalten des Rechners wird für den Editor der Kanal #0 zum Schreiben und Lesen geöffnet.

Der Nachteil bei der Benutzung der CIO ist aber, dass für die Ausgabe nur eines Buchstabens zahlreiche Parameter gesetzt werden müssen. Da diese Parameter in aller Regel nur von der CIO-Routine, nicht aber von den eigentlichen Ausgaberoutinen benötigt werden, drängt es sich also direkt auf, gleich die Editor-Routinen des Betriebssystems zu nutzen, ohne den Umweg über die CIO zu machen.

Dazu müssen wir die Adresse der zu benutzenden Routine aus der entsprechenden Treibertabelle entnehmen. Die Treibertabelle des Editors beginnt bei $E400 und ist wie folgt aufgebaut:

$E400/$E401 Open-Routine -1
$E402/$E403 Close-Routine -1
$E404/$E405 Get-Byte-R. -1
$E406/$E407 Put-Byte-R. -1
...

Diese Routinen werden aufgerufen, indem erst das High- und dann das Low-Byte auf dem Stack abgelegt wird und man den Rechner einen RTS ausführen lässt.

Die Zeichenausgaberoutine sieht daher so aus:

```
00010 PUTCHAR   TAX
00020           LDA $E407
00030           PHA
00040           LDA $E406
00050           PHA
00060           TXA
00070           RTS
```

Der Akku muss vor dem Aufruf der Routine den ASCII-Code des auszugebenden Zeichens enthalten. Da der Akku in der Routine gebraucht wird, um die Adresse der eigentlichen Putbyte-Routine auf dem Stack abzulegen, wird der Inhalt vorübergehend im X-Register gespeichert.

Um nun längere Texte auszugeben, gibt es eine recht einfach zu handhabende Print-Routine, die auf die Putchar-Routine zugreift:

```
00200 PRINT     PLA
00210           STA $D0
00220           PLA
00230           STA $D1
00240 INCP      INC $D0
```

```
00250           BNE .1
00260           INC $D1
00270 .1        LDX #0
00280           LDA ($D0,X)
00290           CMP #$EA
00300           BEQ ENDPR
00310           JSR PUTCHAR
00320           JMP INCP
00330 ENDPR     LDA $D1
00340           PHA
00350           LDA $D0
00360           PHA
00370           RTS
```

Diese Routine wird wie folgt benutzt:

```
00010           JSR PRINT
00020           .AS "TESTTEXT HALLO"
00030           .HX 9B
00040           .HX EA
00050           ... (weiteres Programm )
```

Nach Aufruf der Routine über JSR ist auf dem Stack die Adresse des letzten Bytes des JSR-Aufrufs abgelegt. Diese Adresse ist aber auch genau die Anfangsadresse des Textes weniger 1.

Diese Adresse wird nun vom Stapel genommen, in der Zero-Page ($D0/$D1) abgelegt und um eins erhöht.

Die Adresse $D0/$D1 dient also als Zeiger (Vektor) auf den auszugebenden Buchstaben. Dieser wird in den Akku geladen und mit dem Zahlenwert #$EA verglichen. Ist er ungleich #$EA, so wird er über die Putchar-Routine ausgegeben, die Adresse in $D0/$D1 wird um eins erhöht und der Vorgang wiederholt sich.

War der eingelesene Wert gleich #$EA so wird die Schleife abgebrochen und der Zeiger $D0/$D1 wird auf dem Stack abgelegt. Der Befehl RTS bewirkt nun einen Sprung an die Adresse aus $D0/$D1 plus 1. Diese Adresse ist aber nun genau die erste hinter dem Text, so dass das Programm genau hinter dem Text weitergeführt wird.

Der Hex-Wert EA (die Anweisung .HX EA) wird hier als Textendkennung verwendet. Sie kann natürlich auch gegen eine andere Kennung ausgetauscht werden. Da die Print-Routine am Bildschirm immer bündig schreibt, so wie etwa der Basic-Befehl PRINT"....."; muss zu Ende einer Zeile Return ausgegeben werden. Dies geschieht durch ein Einfügen von .HX 9B in den Text:

```
00010           JSR PRINT
00020           .AS "ZEILE 1"
00030           .HX 9B
00040           .AS "ZEILE 2"
00050           .HX 9BEA
```

Wenn man an einer bestimmten Stelle des Bildschirms schreiben möchte, so kennt man vom Basic den POSITION Befehl. Dieser lässt sich in Assembler sehr leicht realisieren: In der Adresse $54 muss die Zeilenposition (Y-Wert) des Cursors und in der Adresse $55 die Spaltenposition (X-Wert) des Cursors abgelegt werden.

```
00010           LDA #$10
00020           STA $54
00030           LDA #$15
00040           STA $55
00050           JSR PRINT
```

```
00060           .AS "POS. $15,$10"
00070           .HX 9BEA
```

So, dies sollte für diese Ausgabe reichen. Ich kann Ihnen im übrigen die Bibo-Assembler Tooldisk 1 nur wärmstens empfehlen, da sie viele praktische Routinen enthält, die zudem weitestgehend dokumentiert sind.

Bis nächsten Monat
Ihr Uwe Röder
CSM 07/1989

Der Artikel entstammt der Kursreihe ?6502 Programmieren? des Compy Shop Diskettenmagazins. Die Kursreihe besteht aus 14 Kursen, die im Laufe des Jahres 2011 in unregelmäßigen Abständen einzeln veröffentlicht werden, bzw. anschließend als Zusammenzug als ABBUC-Buch ?6502 Programmieren? erscheinen.
Koordination: Volkert Barr (volkert@nivoba.de)
Version 1.1 / 2011-01-23