6502 Programmieren, Teil 13#

von Uwe Röder

In dieser Folge möchte ich Ihnen zeigen, wie man einen Input in Maschinensprache realisiert und wie ein Programm gegen RESET geschützt werden kann.

Zuerst zum Input. Wie üblich für alle Input/Output - Operationen, ermöglicht die CIO natürlich auch die Ein- und Ausgaben an den Editor. Nach Einschalten des Computers und nach jedem RESET wird dem Editor der Kanal 0 zugewiesen. Um nun einen Input auszuführen, müssen wir also nicht erst einen Kanal öffnen.

Für einen Text-Input benutzen wir das CIO-Kommando 5 für GET-Record. Dies bedeutet, daß die Eingabe-Zeile solange editiert werden kann bis Return gedrückt wird. Bei dem Druck auf Return werden maximal soviele Zeichen wie in ICBLEN festgelegt ab ICBADR abgelegt.

Es müssen also folgende Werte im IO-Kontrollblock gesetzt werden:

```
ICCOM  $342      :5 für Get Record
ICBADR $344/$345 :Adresse für Daten
ICBLEN $348/$349 :maximale Anzahl
```

Beispiel:

```
00010           .LI OFF
00020 ------------------------------
00030 ICCOM     .EQ $342
00040 ICBADR    .EQ $344
00050 ICBLEN    .EQ $348
00060 CIO       .EQ $E456
00070 Y         .EQ $680
00080 ------------------------------
00090 S         LDX #0    ; KANAL 0
00100           LDA #5    ; GET RECORD
00110           STA ICCOM,X
00120           LDA #20   ; LEN=20
00130           STA ICBLEN,X
00140           LDA #0
00150           STA ICBLEN+1
00160           STA ICBADR,X
00170           LDA #6    ; ADR=$600
00180           STA ICBADR+1,X
00190           JSR CIO
00200 ------------------------------
00210 OUTPUT    LDA #$9B
00220           JSR PUTCHAR
00230           LDY #0
00240 .1        LDA $600,Y
00250           STY Y
00260           JSR PUTCHAR
00270           LDY Y
00280           INY
00290           CMP #$9B
00300           BNE .1
00310           RTS
00320 ------------------------------
00330 PUTCHAR   TAX
00340           LDA $E407
```

```
00350           PHA
00360           LDA $E406
00370           PHA
00380           TXA
00390           RTS
00400 ------------------------------
```

Mehr Aufwand ist nicht erforderlich. Die Routine Output dient hier nur dazu, um die Daten wieder mit hilfe des Editors auszugeben, um die korrekte Funktion zu zeigen.

Wenn Sie Zahleneingaben haben, müssen Sie je nachdem was Sie mit den Zahlen machen wollen die Floating-Point Routinen benutzen. Sie können den ASCII-Text umwandeln in Zahlen im 6-Byte Floating-Point-Format und diese Zahlen dann wieder in das zwei Byte Integer-Format verwandeln. Wie dies geht wurde in einer der vorherigen Ausgaben des Magazins erklärt, weshalb ich an dieser Stelle nicht weiter darauf eingehe.

Nun zum Thema RESET-Schutz.

Es gibt einige wichtige Speicherstellen in Bezug auf den Reset-Schutz. Dies sind folgende:

```
$09      Boot?
$0A/$0B  DOSVEC
$0244    COLDST
```

Die letzte Adresse kennen wohl die meisten User. Ist der Inhalt dieser Adresse ungleich 0, so wird beim Drücken von Reset neu gebootet.

Um ein Programm vor Reset zu schützen ist also erst einmal unbedingt notwendig, diese Speicherstelle auf 0 zu setzen. Steht in der Speicherstelle Boot? eine 1, so wird nach einem Druck auf RESET der Vektor DOSVEC $A/$B benutzt. Diesen Vektor müssen wir also benutzen, damit unser Programm nach einem RESET wieder fortgesetzt wird.

Wenn das zu schützende Programm kein DOS benötigt, so können wir in DOSVEC einfach die Startadresse des Programms eintragen, $0244 gleich 0 und $09 gleich 1 setzen. Jeder Druck auf RESET startet das Programm dann von neuem.

Bei Programmen, die ein DOS benötigen, ist dies nicht so leicht, da das DOS nach jedem Reset selbst neu initialisiert werden muß und sich dazu eben unseres Vektors DOSVEC bedient.

Wir müssen also zuerst den Inhalt von DOSVEC in eine Zero-Page Adresse retten und dann später in unserer Reset-Routine dafür sorgen, daß ein JSR an diese Adresse ausgeführt wird.

Desweiteren müssen wir darauf achten, daß die Routine, die den Inhalt von DOSVEC rettet, nur genau einmal, nämlich ganz zu Beginn des Programmes, aufgerufen wird und dann nie wieder, auch nicht nach einem RESET. In einem solchen Falle würde nämlich unser eigener RESET Vektor geschützt werden. Die Reset-Routine würde sich dann immer wieder selbst aufrufen und das DOS wäre verloren. Beispiel:

```
00010           .LI OFF
00020 ------------------------------
00030 DOSV      .EQ $D8
00040 ------------------------------
00050 START     LDA $C
00060           STA DOSV
00070           LDA $D
00080           STA DOSV+1
00090           LDA #0
```

```
00100           STA $244
00110           LDA #1
00120           STA 9
00150           LDA #RESET
00160           STA $C
00170           LDA /RESET
00180           STA $D
00190           LDA DOSV
00200           BNE S
00210           LDA DOSVEC+1
00220           BNE S
00270 .1        LDA #S
00280           STA $C
00290           LDA /S
00300           STA $D
00290           LDA /S
00300           STA $D
00310           JMP S
00320 ------------------------------
00330 RESET     JSR DOSINIT
00340           JMP S
00350 ------------------------------
00360 DOSINIT   JMP (DOSV)
00370 ------------------------------
00380 S         ... Start des
                eigentlichen
                Programmes.
```

Eine solche Routine könnte dem eigentlichen Programm vorangestellt werden, um es wirksam zu schützen.

Hier wird erst der Inhalt von DOSVEC gesichert und dann DOSVEC auf eine RESET-Routine gesetzt, die erst das "alte" DOS-Init aufruft und dann zum eigentlichen Programmstart S springt.