Roland Mester

Relokatible 6502-Programme

Der Befehlssatz des Prozessors 6502 läßt relokatible, d. h. ohne Adressenanpassung verschiebliche Programme nicht ohne weiteres zu. So gibt es etwa keinen relativen, sondern nur einen absolut adressierten Unterprogrammsprung. Ein kleines Hilfsprogramm macht dies aber trotzdem möglich und gestattet so z. B. in jedem Adreßbereich arbeitende EPROMs zu programmieren oder das freie Verschieben von Programm-Modulen im RAM.

```
INIT    LDA #<ROUT    ;ADR LOW DER ROUTINE
        STA VEKTOR
        LDA #>ROUT    ;ADR HIGH DER ROUTINE
        STA VEKTOR+1
```

Bild 1. So wird auf herkömmliche Weise z. B. ein Interrupt-Vektor initialisiert

```
INIT    JSR RELJSR    ;SPRUNG AUF EIN RTS, DAS DIREKT
        .WOR FINDAD-* ;VOR DER ROUTINE STEHT
        LDA JUMPAD    ;BERECHNETE SPRUNGADRESSE
        STA VEKTOR    ;IM VEKTOR ABLEGEN
        INC VEKTOR    ;UND 1 ADDIEREN
        PHP
        LDA JUMPAD+1
        STA VEKTOR+1  ;HIGH-BYTE ABLEGEN,
        PLP           ;KANN ENTFALLEN, WENN KEINE SEITE
        BNE ENDINI    ;GEKREUZT WIRD (INCL. PHP)
        INC VEKTOR+1
ENDINI

FINDAD  RTS           ;FINDEN DER ADRESSE
ROUT    .......       ;START DER ROUTINE
```

Bild 2. Initialisieren eines Vektors auf verschiebliche Programmteile

Das Relativsprung-Hilfsprogramm muß hier stets an einer festen Adresse stehen, z. B. in einem EPROM ab hex 8000. Im verschieblichen Benutzerprogramm sind alle JMP- und JSR-Befehle durch folgende Aufrufe zu ersetzen:

```
JSR RELJMP   ; = JMP LABEL
.WOR LABEL-* ; DISTANZANGABE ZU LABEL
JSR RELJSR   ; = JSR LABEL
.WOR LABEL-* ; DISTANZANGABE ZU LABEL
```

```
8069          PASS 1
8069          PASS 2
0000
0000          ;RELATIVE SPRUENGE MIT 6502
0000          ;ROLAND MESTER, IM JUNI 82
0000
0000                *=$FE
00FE DISPTR         *=*+2           ;INHALT BLEIBT UNVERAENDERT!
0100
0100                *=$A420         ;MONITORVARIABLEN DES AIM 65
A420 SAVEPS         *=*+1           ;WENN SIE BENUTZT WERDEN,
A421 SAVEA          *=*+1           ;IST EIN PROGRAMM-TRACE
A422 SAVEY          *=*+1           ;NICHT MEHR MOEGLICH
A423 SAVEZP         *=*+2           ;DADURCH SOFTWARESCHUTZ!
A425 JUMPAD         *=*+2
A427
A427                *=$8000
8000 RELJMP                         ;ROUTINE NICHT RELOKATIV
8000          204080 JSR SAVE       ;SAVE REG'S AND ZP-VARIABLE
8003          68     PLA            ;LOW ADR OF .WOR  MINUS 1
8004          85FE   STA DISPTR
8006          68     PLA            ;HIGH ADR OF .WOR
8007          85FF   STA DISPTR+1
8009          205680 JSR COMPAD     ;COMPUTE JUMP-ADDRESS
800C          4C2880 JMP RESTOR     ;RESTORE REG'S AND RETURN
800F
800F RELJSR
800F          204080 JSR SAVE
8012          68     PLA            ;LOW ADR OF .WOR  MINUS 1
8013          85FE   STA DISPTR
8015          68     PLA            ;HIGH ADR OF .WOR
8016          85FF   STA DISPTR+1
8018          205680 JSR COMPAD     ;COMPUTE JUMP-ADDRESS
801B          18     CLC
801C          A5FE   LDA DISPTR     ;SKIP OVER DISTANCE,
801E          6902   ADC #2          ;INCREMENT RETURN-ADR BY 2
8020          A8     TAY            ;SAVE LOW ADR
8021          A5FF   LDA DISPTR+1
8023          6900   ADC #0         ;ADD ONLY CARRY
8025          48     PHA            ;SET NEW RET-ADR ON STACK
8026          98     TYA
8027          48     PHA
8028 RESTOR   AD23A4 LDA SAVEZP     ;RESTORE ZP-VARIABLES
802B          85FE   STA DISPTR
802D          AD24A4 LDA SAVEZP+1
8030          85FF   STA DISPTR+1
8032          AD20A4 LDA SAVEPS
8035          48     PHA
8036          28     PLP            ;RESTORE REG'S
8037          AD21A4 LDA SAVEA
803A          AC22A4 LDY SAVEY
803D          6C25A4 JMP (JUMPAD)   ;JMP TO COMPUTED ADDRESS
8040
8040 SAVE     8D21A4 STA SAVEA      ;SAVE ALL REG'S
8043          08     PHP            ;INCL STATUS
8044          68     PLA
8045          8D20A4 STA SAVEPS
8048          8C22A4 STY SAVEY
804B          A5FE   LDA DISPTR     ;AND ZP-VARIABLE
804D          8D23A4 STA SAVEZP
8050          A5FF   LDA DISPTR+1
8052          8D24A4 STA SAVEZP+1
8055          60     RTS
8056
8056 COMPAD   A001   LDY #1         ;COMPUTE JUMP-ADDRESS
8058          38     SEC            ;ADD: 1 +
8059          A5FE   LDA DISPTR     ;LOW ADR OF ORIGIN +
805B          71FE   ADC (DISPTR),Y  ;DISTANCE LOW
805D          8D25A4 STA JUMPAD     ; =JUMPADR LOW
8060          C8     INY
8061          A5FF   LDA DISPTR+1   ;HIGH ADR OF ORIGIN +
8063          71FE   ADC (DISPTR),Y  ;DISTANCE HIGH
8065          8D26A4 STA JUMPAD+1   ; =JUMPADR HIGH
8068          60     RTS
8069
8069                .END
8069           ERRORS= 0000
```

Bild 3. Routinen für das Erstellen frei relokatibler 6502-Programme, erstellt mit einem AIM-65-Assembler

Mit der Anweisung .WOR wird vom Assembler automatisch die Adreßdifferenz zwischen der augenblicklichen Adresse (*) und dem Sprungziel „LABEL" ermittelt. Dieser 2-Byte-Wert wird in der Reihenfolge Low-, High-Byte im Speicher abgelegt. Die Routinen RELJMP und RELJSR holen ihre Aufrufadresse vom Stack und addieren darauf die 2-Byte-Distanzadresse im 2er-Komplement modulo 2^{16}. Dadurch sind auch Rückwärtssprünge möglich.
Es ist noch zu erwähnen, daß der Assembler des AIM-65 negative Distanzen zwar stets korrekt berechnet, aber dennoch ERROR 04 meldet. Von diesem kleinen Schönheitsfehler sollte man sich aber nicht abschrecken lassen. Bei positiven Distanzen erfolgt keine Fehlermeldung.
RELJSR setzt eine neue Rückkehradresse auf den Stack, die gegenüber der alten um 2 erhöht wurde. Damit werden die 2 Byte der Distanzangabe übersprungen.
Sinnvoll sind die Routinen dann einzusetzen, wenn sie immer an einer festen Adresse im Monitor oder einem Utility-EPROM stehen. Besitzer des AIM-65 haben im Monitor zwei unbenutzte Bereiche: Von $EFB2 bis $EFF8 und von $FFC bis $FFF8. Darin können die Routinen untergebracht werden, wenn man die Monitor-ROMs durch EPROMs ersetzt.
Bei einem relokativen Programm kann das Problem auftreten, daß Pointer (z. B. Interruptvektoren) auf verschiebliche Programmteile zu richten sind. Normalerweise wird ein Vektor wie in *Bild 1* initialisiert. Bei verschieblichen Programmen ist die Routine von *Bild 2* als Ersatz zu verwenden.
Es ist noch zu bemerken, daß sich die Laufzeiten relokativer Programme stark erhöhen können. Man kann diesen Nachteil aber meist in Kauf nehmen, zumal es nun nie wieder das Problem gibt, daß sich zwei gleichzeitig benötigte Routinen überlappen.

Textdateien im Eurocom-II-Kassetten-Basic

Beim Eurocom-II ist die Dateibehandlung für den Recorder (Mini-DCR) nicht Bestandteil des TCS-Basic, sondern wird über die EBC-(Extended Basic Command)-Schnittstelle abgewickelt. Da die Speicherung von Strings auf Digitalkassette vom üblichen Basic-Standard doch sehr stark abweicht, hier der Vorschlag eines recht platzsparenden Verfahrens.
Beim Einlesen von Strings müssen diese in ihrer Länge vorher festgelegt sein, und die Leseroutine der DCR-Software liest genau die vorher festgelegte Zahl von Zeichen. Um Strings auf einer Datei zu speichern, muß man also entweder eine feste Länge für alle Strings vereinbaren oder für jeden String die Länge mit auf der Datei speichern. Die Nachteile der festen Stringlänge liegen auf der Hand: da sich alle Strings nach der Maximallänge richten müssen, wird sehr viel Platz auf der Kassette benötigt. Außerdem sind Routinen für das Füllen und Kürzen der Strings notwendig.
Also werden der String **und** seine Länge gespeichert. Für jeden String wird erst die Länge als Floating-Point-Feld und dann der String selbst auf die Datei geschrieben. Damit die Datei beliebig lang sein kann und die Leseroutine trotzdem das Ende einer Textdatei erkennt, wird als letzter String noch die Zeichenfolge „—EOF—" (EOF steht für End Of File) auf die Datei geschrieben.
Damit beim Einlesen der „Empfängerstring" immer auf genau die richtige Länge gebracht werden kann, muß zu Anfang des Programms ein Dummystring erzeugt werden, von dem dann mit dem Befehl LEFT$ ein Happchen passend abgeschnitten wird. Alles Weitere läßt sich den drei Listings im *Bild* entnehmen.
Zum Schluß noch ein Tip. Durch weitere „Spezialstrings" auf der Datei läßt sich eine Strukturierung erzielen, die in manchen Anwendungen höchst wünschenswert sein kann (z. B. „—EOR—" für End of Record, „—EOS—" für End of Segment).

Jürgen Plate

```
  10 REM *** TEST/DEMO fuer FILE-E/A ***
  15 REM
  20 DIM A$(1000),M(0)  REM M HAT EIN ELEMENT
  30 Z$="........'........'........'........'........'"
  35 REM *** Einzelstrings duerfen maximal 50 Zeichen
  36 REM *** lang sein, sonst ist Z$ laenger zu machen.
  40 FOR I=0 TO 100
  50 A$(I)='test '+STR$(I)
  60 NEXT I
  70 A$(101)='--EOF--'  REM *** END OF FILE ***
  80 GOSUB 1000  REM *** DATEI ERZEUGEN
  90 GOSUB 2000  REM *** DATEI LESEN
 100 FOR I= 1 TO 101:PRINT A$(I):NEXT I
 110 END
 120 REM ================================================
 130
 140
 150
1000 REM *** SCHREIBEN A$ AUF DATEI
1001 REM *** DAS LETZTE ZU SPEICHERNDE ELEMENT
1002 REM *** MUSS DER STRING "--EOF--" SEIN.
1003 REM ***
1004 REM *** VERWENDET J,MEDIM(0),M$,Q$,A$(.)
1005 REM ***
1006 REM ***
1010 INPUT "Dateiname",M$:J=0
1015 INPUT "New or Old",Q$
1020 IF Q$="N" THEN POKE HEX("01AA"),1
1030 EBC OPENWF M$
1040 M(0)=LEN(A$(J)):M$=A$(J)
1050 EBC OUTF M
1060 EBC OUTS M$
1070 IF M$<>"--EOF--" THEN J=J+1:GOTO 1040
1080 EBC CLOSEF
1090 IF Q$="N" THEN POKE HEX("01AA"),0
1100 RETURN
1110 END:REM ----------------------------------------
1111
2000 REM *** LESEN A$ VON DATEI
2001 REM *** DER LETZTE STRING MUSS "--EOF--" SEIN.
2002 REM *** VEWENDET M(.),J,M$,Z$,A$(.)
2003 REM *** Z$ MUSS LAENGER ALS DER LAENGSTE STRING
2004 REM *** AUF DER DATEI SEIN.
2005 REM ***
2006 REM ***
2010 INPUT "Dateiname";M$:J=0
2020 EBC OPENRF M$
2030 EBC INF M
2040 M$=LEFT$(Z$,M(0))
2050 EBC INS M$
2055 A$(J)=M$
2060 IF M$<>"--EOF--"THEN J=J+1:GOTO 2030
2070 EBC CLOSEF
2080 RETURN
2090 END:REM ----------------------------------------
```

Brauchbare Dateibehandlung bei der Mini-DCR ermöglicht dieses Hilfsprogramm