PETER'S
ASSEMBLERECKE
für ATARI -Computer

Der USR-Befehl

Wie in der letzten Assembler-Ecke versprochen, werden wir uns diesmal mit einem Thema speziell für Anfänger unter den Maschinensprache-programmierern befassen. Der erste Kontakt eines Basic-Programmierers mit der Welt der Maschinensprache beginnt fast zwangsläufig durch den USR-Befehl, Grund genug, diesen Befehl einmal näher zu durchleuchten. Wer die Assemblerecke bisher schon verfolgt hat, kennt schon einige Beispiele für den Einsatz von USR zur Erweiterung der Basic-Befehle: Im März ging es um einige Befehle zur besseren Handhabung der Player Missile Grafik, in der April-Ausgabe haben wir den USR-Befehl dazu benützt, um Text in den Grafik-Modus 8 zu bringen. Diesmal soll die Funktion und die Möglichkeiten des USR-Befehls anhand eines kurzen Beispiels erklärt werden. Dazu wollen wir den in einigen Basic-Dialekten vorhandenen DEEK-Befehl (Double-Peek, z.B. in Basic XL) mit USR simulieren.

Genau genommen, handelt es sich beim USR-Befehl eigentlich nicht um einen Basic-Befehl (wie PRINT...), sondern vielmehr um eine Funktion, vergleichbar mit der Wurzelfunktion X = SQR (A). Wie wir dem für das Wurzelziehen verantwortlichen Maschinenprogramm mitteilen, von welcher Zahl die Wurzel berechnet werden soll (im Beispiel vom Inhalt der Variablen A), können wir bei USR einem eigenen Assemblerprogramm ebensolche Werte (Argumente) mitteilen. Zusätzlicher Bonus: Wir sind nicht auf ein Argument beschränkt, es sind (fast) beliebig viele möglich.

Ebenso können wir einen Wert an das Basic zurückgeben, allerdings ist in dieser Richtung nur ein einziger Wert möglich. Konkret heißt das, daß bei X = USR (A, B, C) die Argumente B und C an das Maschinenprogramm übergeben und der zurückgegebene Wert in der Variablen X abgelegt wird. Aber wo steckt dann das Argument A? Damit hat es eine besondere Bewandtnis. Das erste USR-Argument enthält immer die absolute Adresse des Maschinenprogrammes. Das kann sowohl als Konstante, z.B. X = USR (1536,B,C) für PAGE 6, oder über eine Variable wie oben geschehen. Da man Variablen im ATARI-Basic lange Namen geben kann, ist es möglich, sich damit eine komfortable Befehlserweiterung zu schaffen. Man definiert z.B. DEEK = 1536 und kann dann die im Anschluß gezeigte Double-Peek Funktion mit X = USR (DEEK,A) aufrufen.

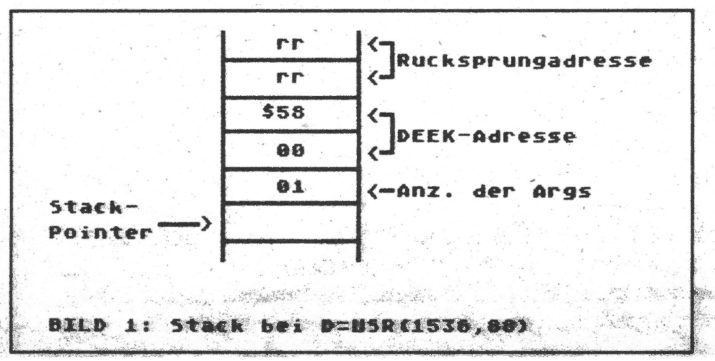

BILD 1: Stack bei D=USR(1536,88)

Noch ein paar Feinheiten: Man kann ein Maschinenprogramm mit USR auch aufrufen, ohne Argumente (abgesehen von der Adresse!) zu übergeben, aber es wird immer ein Wert zurückgegeben. Hat man das Maschinenprogramm, das nun mal eben keine Werte zurückzugeben braucht, dann wählt man eine nicht benützte Variable (DUMMY-Variable) als Ziel. Minimalform des USR-Befehls ist also X = USR (A) mit Dummy-Variable und Adresse. Selbstverständlich ist es auch möglich, statt einfacher Variablen ganze Ausdrücke einzusetzen, so daß X = USR (PEEK, 40000 + I) durchaus gültig ist.

Das Maschinenprogramm wollen wir der Einfachheit halber in PAGE 6 legen. Wir definieren in Basic später eine Variable PEEK = 1536, womit wir das Befehlsformat festlegen können. Bei D =USR (PEEK, A) steht A für die gewünschte Adresse der DEEK-Funktion und D für die ausgegebenen Daten. Wenn Sie z.B. die Adresse des Bildschirmspeichers wissen möchten, dann genügt ?USR (DEEK,88) anstatt vorher ?PEEK (88) + PEEK (89) * 256. Ist doch nett, oder?

Doch halt! Bevor Sie sich jetzt gleich ans Gerät werfen, um das Maschinenprogramm im Listing 1 einzutippen, sollten wir uns erst den Mechanismus der Übergabe von Argumenten an das Maschinenprogramm ansehen. Grundsätzlich gilt, daß nur Ganzzahlwerte von 0-65535 (jeweils also zwei Bytes) übergeben werden können. BASIC schiebt diese Werte nacheinander auf den Hardwarestack (in Page 1), so daß sie vom Maschinenprogramm dort wieder abgeholt werden können. Bild 1 zeigt die Verhältnisse am Stack beim USR-Befehl D = USR (DEEK,88). Wie Sie daraus entnehmen können, tut BASIC noch mehr, es legt nämlich die Anzahl der Argumente als letztes Byte am Stack ab. Damit können Sie nachprüfen, ob auch die richtige Anzahl von Argumenten für Ihre spezielle Anwendung eingegeben wurde.

Im DEEK-Programm wollen wir auf solche Schnörkel verzichten, Wir nehmen daher das erste Byte mit PLA vom Stack und überschreiben es gleich mit dem nächsten PLA. Jetzt haben wir den höherwertigen (!) Teil (das MSB) der DEEK-Adresse im Akku, den wir gleich in eine Zeropagespeicherzelle ablegen. Nebenbei bemerkt: Für USR-Routinen können Sie die Speicherzellen $CB-$D1 ohne Bedenken verwenden. Der höchste PLA liefert jetzt den niederwertigen Teil der PEEK-Adresse (das LSB, im Beispiel $58), den wir ebenfalls in die Zeropage eintragen. Hätten wir einen USR mit mehreren Argumenten vorliegen, so würde sich dieses Schema wiederholen: der nächste PLA liefert das MSB des Arguments 2, dann das LSB des Arguments 2 usw.

Wir sind aber bereits nach dem ersten und einzigen Argument fertig und können gleich zur Verarbeitung übergehen. Die gewünschten Daten können wir leicht mit zwei indirekten Ladebefehlen erhalten, wir brauchen sie jetzt nur noch ans BASIC zurückzugeben. Keine Angst, das ist einfacher als es klingt. Wir müssen sie nur in die zwei Zeropage-Speicherzellen $D4 und $D5 eintragen, das BASIC erledigt den Rest. Diese beiden Speicherzellen liegen im sogenannten Floating-Point Register 0 und wer-

den meist mit dem Label FR0 gekennzeichnet. Wir legen also das LSB des gelesenen Datums in FR0 ab, das MSB kommt nach FR0+1 und schließlich beenden wir das Maschinenprogramm mit RTS.

Der einfachste Weg, so ein USR-Unterprogramm in ein eigenes BASIC-Programm zu implementieren, führt über einen BASIC-Loader (Listing 2). Sie können den mühevollen Weg gehen und alle vom Assembler ausgespuckten Hexzahlen umrechnen und per Hand in DATAs eintragen oder aber, sie nehmen den wesentlich eleganteren Weg und lassen das von einem Programm machen (wofür hat man denn einen Computer?!). Z.B. können Sie dazu das DATGEN-Programm aus einer früheren CK benutzen.

Ach ja, ein paar Fallstricke im Umgang mit dem USR-Befehl muß ich noch nennen: 1. Geben Sie einem USR-Befehl immer die benötigte Anzahl von Argumenten. Fehler können meist nur mit der Reset-Taste behoben werden. 2. Vermeiden Sie es, Maschinenprogramme, die nicht extra dafür geschrieben sind, mit USR aufzurufen. Beispiel: Betriebssystemroutinen wie CIO oder SETVBV, der Effekt gleicht dem des ersten Punktes. Den Grund dafür kennen Sie bereits, es ist nämlich das Byte am Stack, das die Anzahl der Argumente angibt. Denn auch wenn Sie kein Argument angeben, wird eine 0 am Stack abgelegt, die beim RTS-Befehl des Maschinenprogrammes als Teil der Rücksprungadresse zum BASIC aufgefaßt wird. Abhilfe: Schreiben Sie einen kleinen Sprunghandler, der das Byte vom Stack nimmt und dann das gewünschte Programm per JMP aufruft. 3. Wenn Sie eine kleine Routine schreiben, um die benötigte Anzahl der Argumente mit der tatsächlich angegebenen zu vergleichen, dann sollten Sie nicht vergessen, die ungewünschten Argumente im Fehlerfall vom Stack zu nehmen, ein einfaches RTS führt meist zur Katastrophe (s.o.). Listing 3 gibt Ihnen einen Vorschlag, wie man's machen kann.

Wenn Ihr Programmiereifer geweckt worden ist, dann können Sie jetzt an Ihren ATARI stürzen und sich selbst an einem USR-Befehl versuchen: Programmieren Sie einen DOKE-Befehl, der ähnlich wie POKE funktioniert, aber nur Zwei-Byte-Zahlen im Speicher ablegt. Viel Spaß!

Peter Finzel

Listing 1

```
                0100 ;*******************************
                0110 ;LISTING 1: USR-BEFEHL
                0120 ;
                0130 ;Double-Peek Funktion
                0140 ;AUFRUF: D=USR(1536,<Adresse>)
                0150 ;*******************************
                0160 ;
    =00D4       0170 FR0 =    $D4      Ergebnisregister
    =00CB       0180 HLFREG = $CB      Hilfsregister in Zeropage
                0190 ;
0000            0200      *=  $0600    Programm in PAGE 6
                0210 ;
0600 68         0220      PLA          Anzahl der Argumente
0601 68         0230      PLA          MSB der Adresse
0602 85CC       0240      STA HLFREG+1 in Zeropage damit
0604 68         0250      PLA          LSB der Adresse
0605 85CB       0260      STA HLFREG
0607 A000       0270      LDY #0       jetzt geht's an die Arbeit...
0609 B1CB       0280      LDA (HLFREG),Y LSB des Datums indirekt laden
060B 85D4       0290      STA FR0      und in Ergebnisregister eintragen
060D C8         0300      INY          MSB des Datums in naechster Adresse
060E B1CB       0310      LDA (HLFREG),Y
0610 85D5       0320      STA FR0+1    MSB in Ergebnisregister
0612 60         0330      RTS          das war's...
                0340 ;
```

Listing 2

```
100 REM LISTING 2: BASIC-Loader fuer USR-DEMO
110 GOSUB 1000
120 DEEK=1536
130 ? "Der Screen beginnt bei ";USR(DEEK,88)
190 END
1000 REM * BASIC-LOADER fuer DEEK
1010 S=0:RESTORE 1100
1020 FOR A=1536 TO 1554:READ D:POKE A,D:S=S+D:NEXT A
1030 IF S<>2892 THEN ? "DATEN-FEHLER!":STOP
1090 RETURN
1100 DATA 104,104,133,204,104,133,203,160,0,177,203,133,212,200,177
1110 DATA 203,133,213,96
```

Listing 3

```
                0100 ;***********************************************
                0110 ;LISTING 3: Programmteil zum Pruefen der
                0120 ;           korrekten Anzahl der Argumente
                0130 ;***********************************************
                0140 ;
    =0003       0150 ARGANZ = 3       Sollwert der Anzahl (z.B. drei)
                0160 ;
                0170 ;...
0000 68         0180     PLA          tatsaechliche Anzahl vom Stack
0001 AA         0190     TAX          ins X-Register
0002 E003       0200     CPX #ARGANZ  ist's richtig?
0004 F00A       0210     BEQ STIMMT   na bitte, stimmt -->
                0220 ;
0006 E000       0230     CPX #0       etwa gar kein Argument??
0008 F005       0240     BEQ STCKOK   dann gleich zum RTS-->
000A 68         0250 LEEREN PLA       Argument vom Stack
000B 68         0260     PLA
000C CA         0270     DEX          ein Arg. weniger am Stack
000D D0FB       0280     BNE LEEREN   immer noch was da?!-->
                0290 ;
000F 60         0300 STCKOK RTS       Der Stack ist wieder ok, zurueck!
                0310 ;
0010            0320 STIMMT ;         ... weiter im Programm...
                0330 ;
```