Tausch-programm für Page Ø und 1 des 6502

SWAP

Wenn im Speicher schon zwei (oder mehrere) umfangreiche Programme, wie zum Beispiel ein BASIC-Interpreter und ein Disketten-Betriebssystem möglichst noch zusammen mit Video-Software abgelegt sind, kommt es sehr leicht zu Schwierigkeiten mit Page Ø und mit dem Stapel auf Page 1. Eins der gängigsten Gegenmittel ist, den Inhalt von Page Ø und Page 1 im RAM auszutauschen, wenn man von einem auf das andere Programm umspringt. Beispiel: Page 1 auf EØØØ...EØFF und den Stapel auf E1ØØ...E1FF setzen. Dadurch ist jede Gefahr, daß die Zeiger von Page Ø oder der Inhalt des Stapels zerstört werden, ausgeschlossen.

Typisch für den Prozessor 6502 ist unter anderem, wie er Page Ø und 1 verwendet. 256 Byte von 0000_{Hex} bis $00FF_{Hex}$ werden mit speziell für diesen Bereich gedachten Operationskodes adressiert. Bei der Adressierung auf Page Ø ist also das höchstwertigste Byte während einer Datenänderung in diesem Bereich nicht besonders gekennzeichnet, weil es schon im Operationskode enthalten ist. Genau diese 256 Byte können auch als 16-bit-Zeiger zur indirekten, indexierten Adressierung des restlichen Speicherbereiches verwendet werden. Die 256 Byte von 0100_{Hex} bis $01FF_{Hex}$ bilden den Stapel des 6502, eine Art Sicherungsregister, das der Prozessor selbst verwaltet. Dabei ist die letzte Information, die in dieses Register hereinkommt, auch die erste, die es wieder verläßt, weil der Prozessor immer nur die zuletzt eingetroffene Information des Stapels bearbeiten kann. Dafür hat er auch einen internen Stapelzeiger.
Es ist sicher nicht schwer einzusehen, daß schon die geringste (unabsichtliche) Änderung der Parameter von Page Ø und 1 den Programmablauf — meist unwiderruflich — durcheinanderbringt. Wenn zwei Programme gleichzeitig ausgeführt werden, ist es daher notwendig, daß sie nicht gegenseitig die gespeicherten Parameter zerstören. Für den Programmierer ist das ein zusätzliches und manchmal unlösbares Problem. Da die vorhin erwähnten Programme ziemlich umfangreich sind, ist es wirklich ratsam, sich vom allem zu befreien, was die Verwendung von Page Ø und 1 behindert.
Mit dem hier vorgestellten Unterprogramm überträgt man den Inhalt von Page Ø und 1 in einen anderen RAM-Bereich, um ihn vor jeder unwillkommenen Abänderung zu schützen. Gleichzeitig wird der Inhalt des bewußten RAM-Bereichs auf Page Ø und 1 übertragen. In der Programmiersprache heißt so etwas "Swap" (englisch für Austausch). Man braucht sich in Zukunft keine Gedanken mehr über den Inhalt von Page Ø und 1 zu machen, wenn man aus einem Programm in das andere geht. Es genügt, das Unterprogramm "SWAP" laufen zu lassen, und alles erledigt sich wie von selbst: Page Ø ($0000_{Hex}...00FF_{Hex}$) und Page 1 ($0100_{Hex}...01FF_{Hex}$) des ersten Programms werden in $E000_{Hex}...E1FF_{Hex}$ gesichert, während der Inhalt von Page Ø und 1 des zweiten Programms (bis dahin in $E000_{Hex}...01FF_{Hex}$) jetzt in $0000_{Hex}...01FF_{Hex}$ übertragen wird. Kehrt man nun vom zweiten wieder ins erste Programm zurück, wird erst wieder das Programm SWAP ausgeführt und alles wird zurücktransportiert. Natürlich können die Adressen $E000_{Hex}...E1FF_{Hex}$ je nach verwendetem Computer geändert werden. Auf jeden Fall aber müssen sie im **RAM-Bereich** sein. Auch das Unterprogramm SWAP muß immer im RAM sein. Warum das so ist? Werfen Sie doch mal einen Blick auf die letzte Programmzeile. Indizierung und Austausch (swap) sind die Eckpfeiler des Programmierens.

Tabelle 1.

```
0010: E200                      ORG   $E200
0020:
0030:
0040:                  ******
0050:                  *SWAP*
0060:                  ******
0070:
0080:
0090:                  SWAP PAGE 0 AND PAGE 1 WITH E000 AND E100
0100:
0110:
0120:
0130: E200             PZ      *     $0000  PAGE ZERO
0140: E200             STACK   *     $0100  STACK AREA
0150: E200             SPZ     *     $E000  SWAPPED PAGE ZERO
0160: E200             SSTACK  *     $E100  SWAPPED STACK AREA
0170:
0180:
0190:
0200: E200 18          SWAP    CLC
0210: E201 68                  PLA
0220: E202 69 01               ADCIM $01    PUT RETURN ADDRESS
0230: E204 8D 2C E2            STA   JMPINS +01 JUST BEHIND A JUMP OP-CODE
0240: E207 68                  PLA
0250: E208 AA                  TAX
0260: E209 90 01               BCC   SW
0270: E20B E8                  INX
0280:
0290: E20C 8E 2D E2    SW      STX   JMPINS +02
0300: E20F A2 00               LDXIM $00    RESET INDEX
0310:
0320: E211 BD 00 01    SWAPST  LDAX  STACK  GET BYTE FROM PAGE 1
0330: E214 BC 00 E1            LDYX  SSTACK GET BYTE FROM SWAP AREA
0340: E217 9D 00 E1            STAX  SSTACK SAVE BYTE FROM PAGE 1 IN SWAP AREA
0350: E21A 98                  TYA
0360: E21B 9D 00 01            STAX  STACK  SAVE BYTE FROM SWAP AREA IN PAGE 1
0370: E21E B5 00               LDAX  PZ     GET BYTE FROM PAGE 0
0380: E220 BC 00 E0            LDYX  SPZ    GET BYTE FROM SWAP AREA
0390: E223 9D 00 E0            STAX  SPZ    SAVE BYTE FROM PAGE 0 IN SWAP AREA
0400: E226 94 00               STYZX PZ     SAVE BYTE FROM SWAP AREA IN PAGE 0
0410: E228 E8                  INX          BUMP THE OFFSET
0420: E229 D0 E6               BNE   SWAPST NOT DONE, KEEP ON
0430:
0440: E22B 4C FF FF    JMPINS  JMP   $FFFF  SELF MODIFYING CODE!!!
0450:
-T

    SYMBOL TABLE 3400 3430
    JMPINS E22B    PZ     0000    SPZ    E000    SSTACK E100
    STACK  0100    SW     E20C    SWAP   E200    SWAPST E211
-
```

Tabelle 1. In das Programm SWAP kommt man mit einem JSR-Befehl hinein, aber nicht mit RTS, sondern mit JMP wieder heraus. Die Rücksprungadresse wird zu Anfang des Programms erst "vom Stapel genommen", korrigiert (Rücksprungadresse = Ausgangsadresse + 1) und dann hinter den Sprungbefehl JMPINS gesetzt.