Assembler für Fortgeschrittene#

von Uwe Röder, CSM 01/1990

Viele Maschinensprache-Programmierer werden schon vor dem Problem gestanden haben, dass sie nicht wussten, wie man Dateien oder Files in den Speicher lädt.

Es gilt zuerst zu unterscheiden um was für ein Files es sich handelt. Wenn es sich um ein reines Datenfile handelt, wie zum Beispiel einen Zeichensatz, so müssen wir die Ein- und Ausgabe-Routinen aus der CIO benutzen. Diese werden uns vom DOS zur Verfügung gestellt. Egal über welches DOS man verfügt, die Benutzung der Routinen über die CIO ist immer gleich (jedenfalls bei den wichtigen Routinen).

Möchte ich ein File Laden, das über einen Fileheader verfügt, also ein File, das ich auch über das DOS-Menue laden kann, so muss man direkt auf Routinen des DOS zugreifen. In aller Regel sind alle DOS in diesem Punkt verschieden, so dass ich an dieser Stelle nur Aussagen über das BIBO-DOS machen kann.

Zuerst aber zu dem Laden eines reinen Datenfiles ohne Header.

Um ein solches File zu laden, müssen wir, wie schon gesagt, die CIO benutzen.

Die CIO verfügt über acht Kanäle, von denen der User aber nur sieben frei benutzen sollte. Dies liegt daran, dass der Editor nach dem Anschalten oder Drücken von Reset über Kanal 0 betrieben wird.

Jeder dieser Kanäle verfügt über einen eigenen Kontroll-Block, das ist ein kleiner Speicherbereich in dem vor Einsprung in die CIO-Routine genau festgelegt wird was die Routine tun soll. Diesen Kontroll-Block nennt man IOCB = Input Output Control Block.

Die Länge eines IOCB's beträgt 16 Bytes. Diese Bytes haben folgende Bedeutungen:

$340 (832) - ICHID

Wenn der Kanal unbenutzt ist, enthält dieses Byte den Wert $FF (255). Die ist zum Beispiel nach einem CLOSE-Befehl der Fall.

$342 (834) - ICCOM

Diese Speicherstelle ist wohl die wichtigste, da dieses Byte vor Aufruf der CIO das Befehls-Byte enthalten muss. Je nach Befehl ändert sich teilweise die Bedeutung der folgenden Speicherstellen. Auch müssen nicht immer alle Bytes definiert werden. Am Ende dieser Liste der Speicherstellen, wird genau angegeben, welche Befehle grundsätzlich möglich sind und welche Bytes dafür wie gesetzt werden müssen.

$343 (835) - ICSTAT

In diesem Byte und in dem Y-Register befindet nach dem CIO-Aufruf der Statuswert der Operation. Wenn er ungleich 1 ist, ist ein Fehler aufgetreten.

$344/$345 (836/847) - ICBAL/ICBAH

Allgemein ist dies ein Zeiger auf einen bestimmten Speicherplatz. Bei einem OPEN Befehl muss zum Beispiel die Adresse des ersten Bytes der Dateispezifikation (D:FILENAME.EXT";#$9B) hier abgelegt werden. Diese nennt man auch "Filespec" (engl. Abk. für Filespecification).

Beim Lesen oder Schreiben von Daten, ist dies die Adresse des ersten Bytes des Speicherbereichs bei dem die Daten abgelegt werden, oder welcher geschrieben wird.

$348/$349 (840/841) - ICBLL/ICBLH

Beim Schreiben und Lesen von Daten muss hier die Länge des Datenblocks festgelegt werden. Ein Zeichensatz ist zum Beispiel 1024 Bytes, ein GR.8+16 Bild ist 7680 Bytes lang.

Gibt man hier als Länge 0 an, so wird dennoch immer ein Byte übertragen. Dieses steht dann im Akku. Nach dem CIO-Aufruf findet man in diesen Registern die Anzahl der in Wirklichkeit übertragenen Bytes. Wenn zum Beispiel ein Fehler aufgetreten ist, so kann diese Anzahl geringer sein als erwünscht.

Dies sind also die einzelnen Register eines IOCB's. Wie gesagt gibt es acht IOCBs. Die hier angegebenen Adressen beziehen sich eigentlich nur auf den IOCB 0. Um nun beliebige IOCBs anzusprechen kommt zum Beispiel folgende Methode in Frage:

```
S       LDX #$10  ($20/$30...$70)
LDA ...
STA ICCOM,X
LDA ...
STA ...,X
JSR $E456
...
```

Im X-Register steht hier der Abstand zwischen erstem Byte des ersten IOCB's und erstem Byte des gewünschten IOCB's. Da die Länge eines IOCB's genau $10 (16) ist, beträgt dieser Abstand $10*Kanalnummer (16*Kanalnummer).

Da jeder eigene Gerätetreiber schreiben kann, kann nie eine genaue Liste der möglichen Kommando-Bytes (ICCOM) gegeben werden. Generell sind aber folgende Kommando-Bytes für ICCOM üblich:

$3 - OPEN

Mit diesem Aufruf wird ein Kanal geöffnet. Dazu müssen ICBAL/ICBAH unbedingt immer die Adresse der Filespezifikation beinhalten. Diese sieht im Bibo-Assembler-Format beispielsweise wie folgt aus:

```
.DA "D1:FILENAME.EXT",#$9B
```

Wie man sieht, ist dies nichts neues. Es müssen die Gerätespezifikation (C:,E:,D:...), der Filename und als Endkennung das EOL-Zeichen (#$9B) angegeben werden. Basic-Usern möchte ich empfehlen den OPEN oder CLOSE-Befehl des ATARI-BASICS zu benutzen, da er das gleiche bewirkt, aber einfacher zu handhaben ist.

$5 - Get Record

Ein File wird solange eingelesen, bis ein EOL-Zeichen (#$9B) eingelesen wird. ICBAL/ICBAH enthält die Zieladresse des Records. ICBLL/ICBLH enthält die Anzahl der einzulesenden Bytes. Ist diese

überschritten ohne dass ein EOL auftrat, so wird solange weiter gelesen, bis eines auftritt. Der Lesevorgang wird dann mit einer Error-Meldung abgebrochen.

$9 - Put Record

Es wird ein File mit der Länge ICBLL/ ICBLH geschrieben. Die Adresse der Daten muss in ICBAL/ ICBAH abgelegt sein.

$7 - Get Characters

Einlesen eines Datenfiles. (zum Beispiel Font, Bild...) ICBLL/ICBLH - Anzahl der einzulesenden Bytes ICBAL/ICBAH - Adresse bei welcher die Daten abgelegt werden sollen

$B (11) - Put Characters

Schreiben eines Datenfiles ICBLL/ICBLH - Anzahl der zu schreibenden Bytes ICBAL/ICBAH - "Source"-Adresse

$C (12) - CLOSE

Mit diesem Kommando wird das entsprechende File geschlossen. Beim Schreiben auf Disk werden einige Bytes erst jetzt geschrieben. Auch der Eintrag in die VTOC erfolgt nun. Ohne den Close Befehl wären diese Daten verloren gewesen. Daher darf man ihn auf keinen Fall nach Beendigung einer Operation vergessen!!!

$D (13) - STATUS

Dieser Befehl liest vier Statuswerte ein, die er bei $2EA (746) ablegt. Ist der entsprechende Kanal noch nicht geöffnet muss ICBAL/IBAH die Filespezifikationsadresse enthalten.

Zur genauen Benutzung der CIO befindet sich noch ein Beispielprogramm im BIBO- ASSEMBLER-Format im Anhang.

Um nun Files mit einem Fileheader mit dem BIBO-DOS zu laden muss man nur eine Speicherstelle und einen Einsprungvektor kennen:

RUNFLG $735
RUNAD $70F

Um ein File zu laden muss man in den Akku und Y-Register die Adresse des Files laden. Ein JSR zur Adresse RUNAD bewirkt dann automatisch ein Laden des Files. Der Wert in RUNFLG bewirkt, ob das geladene File dann gestartet werden soll oder nicht.

RUNFLG=0 File wird geladen und gestartet
RUNFLG=FF File wird nur geladen

Beispiel:

```
00010 RUNFLG       .EQ $735
00020 RUNAD        .EQ $70F
00030 ----------------------------
00040 START        LDA #0
00050              STA RUNFLG
```

```
00060               LDA #FILE
00070               LDY /FILE
00080               JSR RUNAD
00090               RTS
00100 ------------------------------
00110 FILE          .DA "D:FILE.EXT"
00120               .DA #$9B
00130 ------------------------------
```

In diesem Fall wird ein File geladen und gestartet. Wichtig ist aber, dass diese Files über einen Fileheader verfügen. Dieser enthält eine Filekennung ($FFFF), die Start- und die Endadresse. Wenn das so geladene File nicht automatisch vom DOS gestartet werden soll muss in RUNFLG der Wert $FF stehen.

So, dies sollte mal wieder reichen. Ich hoffe Sie sind mit dem Artikel klargekommen und können nun optimal mit Files umgehen.

Anhang:#

Beispiel:#

```
00010           .LI OFF
00020 ------------------------------
00030 ICCOM     .EQ $342
00040 ICBADR    .EQ $344
00050 ICBLEN    .EQ $348
00060 ICAUX     .EQ $34A
00070 CIOV      .EQ $E456
00080 ------------------------------
00090 S         LDX #$10  ;KANAL 1
00100           LDA #3 ; OPEN
00110           STA ICCOM,X
00120           LDA #FILE
00130           STA ICBADR,X
00140           LDA /FILE
00150           STA ICBADR+1,X
00160           LDA #4 ; GETBYTES
00170           STA ICAUX,X
00180           JSR CIOV
00190 ------------------------------
00200           LDX #$10
00210           LDA #7 ; GETBYTES
00220           STA ICCOM,X
00230           LDA #$0
00240           STA ICBADR,X
00250           LDA #$44
00260           STA ICBADR+1,X
00270           LDA #0
00280           STA ICBLEN,X
00290           LDA #$4
00300           STA ICBLEN+1,X
00310           JSR CIOV
00320 ------------------------------
00330           LDX #$10
00340           LDA #12 ;CLOSE
00350           STA ICCOM,X
00360           JSR CIOV
```

```
00370 ------------------------------
00380          LDA #$44
00390          STA 756
00400          RTS
00410 ------------------------------
00420 FILE     .DA "D:FONT.FNT",#$9B
00430 ------------------------------
```