LCTOOLSV3: Sequenzielle Dateiarbeit

WeRo, Stand: 05.07.2016

Die bislang in LCTOOLS eingebauten USB-Lade- und Speicherroutinen verarbeiten immer eine ganze Datei. Beim Laden wird die gesamte Datei in den Speicher geholt, beim Schreiben die gesamte Länge aus dem Speicher auf das USB-Medium geschrieben. Soll nur ein Teil einer Datei geladen oder nur ein paar Bytes gespeichert oder geändert werden, so sind die bisherigen Funktionen nicht geeignet. Mit den neuen zusätzlichen Funktionen lässt sich diese "stückweise" Arbeit realisieren. Dass dies etwas langsamer geht, ist nicht zu vermeiden...

Für diese Dateiarbeit wird nicht das Z80-Headersave-Format benutzt, sondern eine reine Binärdatei. Die Dateiendung kann frei gewählt werden. Die Struktur der Datei und den Umgang damit muss das jeweilige Programm organisieren.

Mit per Sprungverteiler definierten festen Adressen sind folgende Funktionen erreichbar:

#A049	Dialog "Dateinamen abfragen"
#A04C	Öffnen für Schreiben sequenziell
#A04F	Öffnen für Lesen sequenziell/ Test auf Vorhandensein/ Dateigröße ermitteln *)
#A052	Schließen Datei
#A055	Löschen Datei
#A058	1 Byte schreiben sequenziell
#A05B	1 Byte lesen sequenziell
#A05E	Filepointer positionieren

*) Rückgabewert Dateigröße in `MENGE: EQU #236A ;ARBEITSZELLE 32Bit`

Hinweise zum sequenziellen Schreiben:
Öffnet man eine noch nicht existierende Datei zum Schreiben, so wird sie angelegt. Der Filepointer (=aktuelle Schreib-/ Leseposition) befindet sich am Anfang.

Wird eine vorhandene Datei zum Schreiben geöffnet, so befindet sich der Filepointer hinter dem Dateiende. Geschriebene Bytes werden also normalerweise angehängt. Setzt man den Filepointer zuvor woandershin, so kann bei Bedarf das Byte auf dieser gesetzten Position überschrieben werden. **Soll die Datei als solches in ihrer vollen Länge erhalten bleiben, so ist vor dem nachfolgenden Schließen der Datei der Filepointer dann zwingend auf das Dateiende zu setzen. Ansonsten wird der Dateiinhalt nach dem geschriebenen Byte gelöscht, d.h. die Datei gekürzt!** Um das Dateiende zu ermitteln, benutzt man die beim Öffnen (automatisch) ermittelte Dateilänge und berücksichtigt evtl. angehängte Bytes.

Aus einer zum Schreiben geöffneten Datei kann auch gelesen werden, aber nicht umgekehrt.

Anwendungsbeispiele:

Dateiname vereinbaren

Jeder Umgang mit Dateien erfordert einen Dateinamen. Dieser wird in einem Buffer bereitgestellt, das Register IX muss auf den Bufferanfang weisen. Der Dateiname darf maximal 8 Zeichen lang sein und es muss die gewünschte Endung angegeben sein. Abzuschließen ist der Dateiname mit #0D,#00.

fest im Anwenderprogramm vorgegeben	wählbar per Eingabe
;... LD IX,NAM ;Zeiger ;... NAM: DEFM "TEST.BIN" ;DATEINAME DEFB #0D,0 ;ABSCHLUSS!	;DATEINAME ANGEBEN: CALL #A049 ;EINGABE OHNE ENDUNG OR A ;FEHLER? JP NZ,FEHLER ;JA, ABBRUCH LD A,(IX+0) ;1. ZEICHEN CP "." ;NUR ENTER? JP Z,FEHLER ;JA, ABBRUCH! ;IX=BUFFERANFANG ;ENDUNG IST IMMER Z80, DA EINGABE- ROUTINE STANDARD FÜR Z80-DATEIEN!

	;GGF. NOCH DATEIENDUNG ANPASSEN:
	PUSH IX ;Anfang Dateiname
	POP HL
	ct1: LD A,(HL)
	INC HL
	cp "."
	JR nz,ct1 ;Punkt suchen
	LD (HL),"B" ;B ablegen
	INC HL
	LD (HL),"I" ;I ablegen
	INC HL
	LD (HL),"N" ;N ablegen

USB initialisieren

```
CALL  #A018             ;USB_INIT, DAS DAUERT ETWAS...
OR    A                 ;FEHLERFREI?
JP    NZ,FEHLER         ;NEIN
```

Test auf Vorhandensein einer Datei

```
;DATEINAME VEREINBAREN (IX)...
;USB INITIALISIEREN...
```

```
CALL  #A04F             ;ÖFFNEN: TESTEN OB DATEI VORHANDEN
OR    A                 ;FEHLER?
JP    NZ,FEHLER         ;A=2 BEI "NOT FOUND"
                        ;A=0 WENN DATEI VORHANDEN
CALL  #A052             ;DATEI WIEDER SCHLIESSEN, WENN KEINE OPERATION WEITER
```

Datei löschen

Soll z.B. eine Datei unter gleichem Namen neu geschrieben werden, so ist sie erst zu löschen:

```
;DATEINAME VEREINBAREN (IX)...
;USB INITIALISIEREN...
```

```
CALL  #A055             ;DATEI LÖSCHEN
```

Sequenzielles Lesen von Bytes aus Datei

Beispiel: 4. und 5. Byte der Datei soll gelesen werden:

```
;DATEINAME VEREINBAREN (IX)...
;USB INITIALISIEREN...
```

```
CALL  #A04F             ;TESTEN OB DATEI VORHANDEN, WENN JA: ÖFFNEN ZUM LESEN
OR    A                 ;FEHLER?
JP    NZ,FEHLER         ;A=2 BEI "NOT FOUND"
```

```
LD    HL,0              ;OBERE 16 BIT
LD    DE,4              ;UNTERE 16 BIT: DAS 4. BYTE SOLL GELESEN WERDEN
CALL  #A05E             ;ZU LESENDE BYTEPOSITION SETZEN
```

```
;ZWEI AUFEINANDERFOLGENDE BYTES LESEN+AUSGEBEN
CALL  #A05B             ;BYTE LESEN => A
CALL  #A040             ;HEXADEZIMAL AUSGEBEN AUF SCHIRM
CALL  #A05B             ;BYTE LESEN => A
CALL  #A040             ;HEXADEZIMAL AUSGEBEN AUF SCHIRM
```

```
CALL  #A052             ;DATEI SCHLIESSEN
```

Sequenzielles Schreiben in eine Datei

Beispiel: ein Byte an vorhandene Datei anhängen

```
;DATEINAME VEREINBAREN (IX)...
;USB INITIALISIEREN...
```

```
;DATEI FÜR SEQ. SCHREIBEN ÖFFNEN
CALL  #A04C             ;WENN DATEI SCHON VORHANDEN, DANN IST ZEIGER AM
                        ;DATEIENDE, NEUE DATEN WERDEN ANGEHÄNGT
                        ;WENN NICHT VORHANDEN, DANN NEUE DATEI,ZEIGER AM ANFANG
```

```
LD    A,"*"             ;SOLLBYTE
CALL  #A058             ;SCHREIBEN (ANHÄNGEN)
```

```
;LD   A,"x"             ;GGF. WEITERE BYTES SCHREIBEN
;CALL #A058
```

```
CALL  #A052             ;DATEI SCHLIESSEN
```

Wahlfreier schreibender Zugriff

Beispiel: Das 10. Byte soll einen neuen Wert "*" erhalten

```
;DATEINAME VEREINBAREN (IX)...
;USB INITIALISIEREN...
```

```
;DATEILÄNGE BESTIMMEN
CALL  #A04F             ;TESTEN OB DATEI VORHANDEN, WENN JA: ÖFFNEN ZUM LESEN
OR    A                 ;FEHLER?
JP    NZ,FEHLER         ;A=2 BEI "NOT FOUND"
                        ;DATEINLÄNGE NUN IN (MENGE)
```

```
CALL  #A052             ;DATEI SCHLIESSEN
```

```
;ÖFFNEN ZUM SCHREIBEN
CALL  #A04C             ;ZEIGER AM DATEIENDE, NEUE DATEN WERDEN ANGEHÄNGT
```

```
;POSITIONIEREN
LD    HL,0              ;OBERE 16 BIT
LD    DE,10             ;UNTERE 16 BIT: DAS 10. BYTE SOLL GESCHRIEBEN WERDEN
CALL  #A05E             ;BYTEPOSITION SETZEN
```

```
LD    A,"*"             ;SOLLBYTE
CALL  #A058             ;SCHREIBEN (ÜBERSCHREIBEN DES ALTEN BYTES)
```

```
;POSITIONIEREN AUF ENDE - WICHTIG, SONST WIRD DATEI GEKÜRZT!!!
;IN (MENGE) STEHT DATEILÄNGE
LD    DE,(MENGE)        ;UNTERE 16 BIT
LD    HL,(MENGE+2)      ;OBERE 16 BIT => DE/HL= ZEIGER AUF DATEIENDE
CALL  #A05E             ;BYTEPOSITION SETZEN
```

```
CALL  #A052             ;DATEI SCHLIESSEN
```