LC-TOOLS

für den LC-80ex

Version 1.0

Inhaltsverzeichnis

Übersicht

Nach der erfolgreichen Erprobung der seriellen Übertragung (Laden/Sichern von Dateien) zwischen LC-80ex und einem (PC-)Terminal soll dieses Werkzeug durch USB-Unterstützung weiter ausgebaut werden. Fast alle Funktionen sind per LC-80ex-Hextastatur aufrufbar.

In der Version 1 stehen nunmehr zur Verfügung:

Einsprungadresse	Funktion	Aufruf per
A000	Laden per RS232	LC-80ex-HEX-Tastatur
A003	Laden per RS232 mit offset	
A006	Sichern per RS232	
A009	Laden per USB	
A00C	USBINIT	
A00F	Sichern per USB	
A012	externer Aufruf Laden per USB	MC-Programm
A015	externe Fehlerrücksprungadresse	
A018	externer Aufruf Sichern per USB	
B000	Aufruf "rdk Tiny-Basic" (mit USB-SAVE/LOAD/FILES-Unterstützung)	LC-80ex-HEX-Tastatur
B003	Warmstart rdk Tiny-Basic	

Alle Laden/Sichern-Funktionen benutzen das Headersave-(z80-)Format. Andere Dateiformate werden nicht unterstützt.

Die LC-Tools sind in einem 8k-(E)PROM auf A000...BFFF untergebracht. Dabei besteht noch relativ viel Freiraum für mögliche Erweiterungen:

```
A000...AFFF:      ca. 2k frei
B000...BFFF:      ca. 1,2k frei
```

RS232-Schnittstelle

Terminal-Anschluss

Per RS232 ist sowohl ein "dummes" Terminal anschließbar als auch ein PC mit Terminalemulation (z.B. HTerm). Es wird gegenwärtig das Protokoll

2400 Baud, 8 Datenbits, 1 Stopbit, keine Parität, kein Handshake

benutzt, welches für einen Systemtakt des LC-80ex von 1,8432 MHz gilt. Die LC-80ex-Hardware ist dazu wie folgt einzustellen:

JP9: Systemtakt 1/2, JP12: SIO-Port B0

Die Verwendung des Standardtaktes 921,6 kHz ist ebenso möglich, allerdings halbiert sich die Baudrate auf 1200.

Die parallele Nutzung der seriellen Schnittstelle für Bildschirm und Tastatur zur Bedienung des TinyBasic steht etwas im Widerstreit zur Benutzung für das Laden und Speichern von Daten:

- Entweder benutzt man zusätzlich den zweiten Kanal der SIO (was einer zweiten seriellen Strippe am LXC-80ex und einem zweiten Terminal entspricht) oder
- man schafft sich ein "Spezialterminal", welches per manueller Umschaltung auf Kommunikation (Bildschirm und Tastatur) oder Datenablage (Ein-/Ausgabepuffer) zu schalten ist.

Da beides einen gewissen Aufwand erfordert, entstand u.a. deshalb die USB-Schnittstelle für den Datenverkehr...

Laden von Daten per RS232

1. Nach RESET am LC-80 eingeben:

Taste	=> Anzeige	Beschreibung
ADR	X.X.X.X.X X	Adresseingabe
A 0 0 0	A.0.0.0.X X	A000 = Laderoutine starten
EX	(dunkel)	Empfang beginnt, LC-80 wartet auf serielle Daten.
	E r r - F	Dateifehler (keine korrekte z80-Datei) bzw.
	X X X X X	Anzeige Ladeadresse + Dateityp bzw.
		Autostart, wenn Dateityp= "A"

2. Am PC wird nun mittels Terminalprogramm das gewünschte in den LC-80 zu ladende Programm gesendet. Die empfangenen Daten kommen automatisch an der richtigen Speicherposition des LC-80ex zur Ablage.
3. Ist die im z80-File verankerte Anzahl Bytes übertragen, so wird das geladene Programm automatisch an der im z80-File angegebenen Startadresse gestartet, falls der Typeintrag "A" (= ausführbares Programm mit Autostart) lautete. Bei allen anderen Typen unterbleibt der Autostart. In der Anzeige steht dann die Ladeadresse sowie der Dateityp. Eine beliebige Taste führt zum Grundzustand. Man kann dann andere Aktivitäten vornehmen (z.B. weitere Daten nachladen).

Laden mit variabler Ladeadresse

Obwohl eigentlich bei einem z80-File die Lage im Adressraum festgelegt ist, kann mit dieser Funktion unabhängig von der im z80-File enthaltenen Anfangsadresse geladen werden.

1. Nach RESET am LC-80 eingeben:

Taste	=> Anzeige	Beschreibung
ADR	X.X.X.X.X X	Adresseingabe
A 0 0 3	A.0.0.3.X X	A003 = Laderoutine starten
EX	a d r -S	S = Startdresse eingeben!
4 0 0 0	4 0 0 0 -S	Startadresse=Ladeadresse
+	(dunkel)	Empfang beginnt, LC-80 wartet auf serielle Daten.
	E r r - F	Dateifehler (keine korrekte z80-Datei) bzw.
	X X X X X	Anzeige Ladeadresse + Dateityp

2. Es wird nun an die eingegebene Adresse geladen. Die Länge des einzulesenden Files ergibt sich aus den Anfangs-und Endeangaben im z80-Header.
3. Ist die enthaltene Anzahl an Bytes übertragen, steht dann die Ladeadresse sowie der Dateityp im Display. Eine beliebige Taste führt zum Grundzustand. Man kann dann andere Aktivitäten vornehmen (z.B. weitere Daten nachladen). Ein Autostart unterbleibt hier immer.

Sichern von Daten per RS232

Das erfolgt analog zur Bedienung der Kassettensicherung, wobei die Eingabe eines Dateinamens entfällt und stattdessen zwei neue Parameter ("Autostartadresse" und "Dateityp") abgefragt werden.

1. Am PC ein Terminalprogramm starten (und wenn nötig auf Empfang stellen).
2. Nach RESET am LC-80 eingeben:

Taste	**=> Anzeige**	**Beschreibung**
ADR	X.X.X.X.X X	Adresseingabe
A 0 0 6	A.0.0.6.X X	A006 = Sicherungsroutine starten
EX	a d r - S	S = Startadresse eingeben!
2 0 0 0	2 0 0 0 - S	Anfangsadresse, z.B. 2000H
+	a d r - E	E = Endadresse eingeben!
2 1 F F	2 1 F F - E	Endadresse, z.B. 21FFH
+	a d r - A	A=Autostartadresse eingeben!
2 0 0 3	2 0 0 3 - A	Autostartadresse, z.B. 2003 (nur bei Dateityp A bedeutsam, ansonsten beliebig)
+	t Y P -	Dateityp eingeben
C	t Y P - C	C=Dateityp (A...F) siehe Anlage Übergeht man die Typ-Eingabe einfach mit [+], so wird standardmäßig der Typ "A" verwendet.
+		Anzeige dunkel Datenübertragung startet, TAPE-LED leuchtet
	L C - 80	wenn fertig: TAPE-LED aus automatischer Reset => Monitor

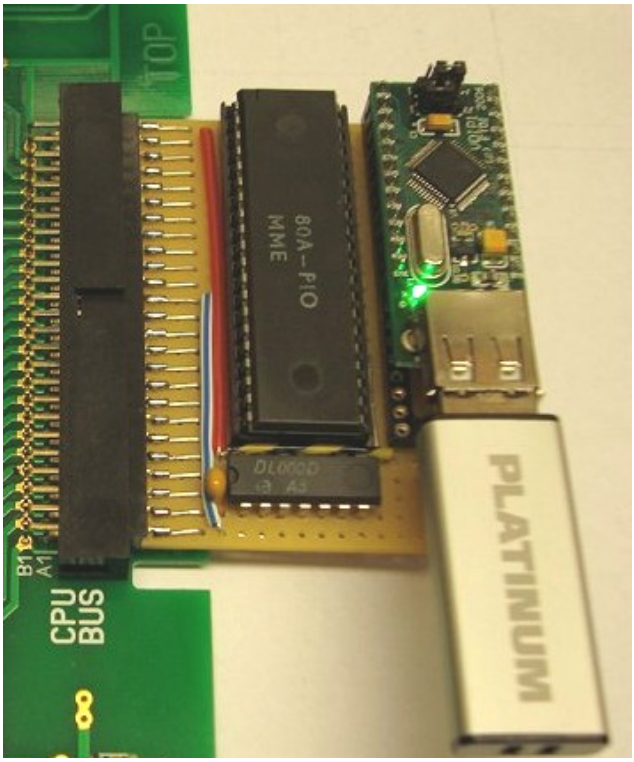
3. Am PC kann nun der Empfangspuffer als Datei abgespeichert werden. Nur dort ist ein Dateiname zur Ablage erforderlich (z.B. TEST.Z80).

USB-Schnittstelle

Ein USB-Anschluss (VINCULUM) am Systembus ermöglicht sowohl Laden und Speichern aus dem (per Terminal bedientem) TinyBasic als auch extern (z.B. per LC-80ex-Monitor). Der Anschluss anderer USB-Geräte außer Speichermedien ist nicht vorgesehen.

Realisierung

Es wurde die herkömmliche Beschaltung und Ansteuerungeines VDIP1 (oder auch VDIP2) gewählt, wie sie bereits in zahlreichen anderen Kleincomputern (AC1, LLC2, Z9001,...) zum Einsatz kam.



Musteraufbau ("gefädelt")

Da hierfür zwei komplette Ports einer PIO benötigt werden, schied der Einsatz am USER-Port des LC-80ex aus (was jedoch den Vorteil hat, dass dieser weiterhin verfügbar bleibt).

Das VDIP wurde mit einer 3. PIO am Systembus angeschlossen. Der Aufwand ist minimal (Schaltplan siehe Anlage).

Die Adressierung erfolgt analog der bisherigen Systematik mit /CE=A6. Damit erhält das VDIP die IO-Adressen BC...BF. Interrupt wird nicht benutzt.

Inbetriebnahme

(ich beschreibe hier meine Vorgehensweise, es geht sicher auch anders...)

1. Hardware-Adapter aufbauen und an LC-80ex anstecken.
2. Einen USB-Stick frisch mit FAT formatieren.
3. Als erste Datei die aktuelle Firmware (FTRFB.FTD, Version 3.69 im Paket) auf den USB-Stick aufspielen.
4. Jumper auf dem VDIP beide auf 1-2, dann USB-Stick an Adapter anstecken
5. LC-80ex einschalten. Das VDIP lädt die Firmware vom USB-Stick und blinkt dann unaufhörlich. Eine Weile blinken lassen...
6. LC-80ex ausschalten, am VDIP Jumper ändern: J3: 1-2, J4: 2-3 (parallele Betriebsart)
7. Nun ist das USB-Interface betriebsbereit. Bei jedem neuen Einschalten (sowie beim Zugriff durch Software) blinken die LEDs kurz.

Fehlermeldungen

Tritt beim Zugriff auf USB ein Fehler auf, so wird als Sammelmeldung entweder "ERR -U" angezeigt (LC-80ex-Display) oder "USB Error" (Bildschirmausgabe bei TinyBasic).

Randbedingungen

Generell gilt:

1. Die Save/Load-Routinen unterstützen nur z80-Headersave-Files. Die Dateinamen haben ".z80" als Erweiterung, die jedoch nicht mit anzugeben ist!
2. Es ist immer darauf zu achten, dass firmwarebedingt Dateinamen maximal 8.3 Zeichen lang sein können und keine Leer-/Sonderzeichen aufweisen.

USB-SAVE/LOAD aus dem LC-80ex-Monitor

Der Aufruf ist vom LC-80ex wie folgt möglich (wobei im Gegensatz zu den RS232-Routinen immer ein Dateiname anzugeben ist):

Laden z80-File:

Taste	**=> Anzeige**	**Beschreibung**	
ADR	X.X.X.X.X X	Adresseingabe	
A 0 0 9	A.0.0.9.X X	A009 = Laderoutine starten	
EX	F I L E - F	F = Filename eingeben *)	
+		Anzeige dunkel, USB-LEDs blinken, dann entweder:	
	E r r - U	Fehler (z.B. kein USB-Stick dran) oder	weiter mit bel. Taste (RESET)
	n o F I L E	Datei wurde nicht gefunden oder	
	2 0 0 0 C	Anzeige der Ladeadresse + Typ oder	
	x x x x x x	Autostart (Dateityp war A)	

Sichern z80-File:

Taste	**=> Anzeige**	**Beschreibung**	
ADR	X.X.X.X.X X	Adresseingabe	
A 0 0 F	A.0.0.F.X X	A00F = Sicherungsroutine starten	
EX	a d r - S	S = Startadresse eingeben!	
2 0 0 0	2 0 0 0 - S	Anfangsadresse, z.B. 2000H	
+	a d r - E	E = Endadresse eingeben!	
2 1 F F	2 1 F F - E	Endadresse, z.B. 21FFH	
+	a d r - A	A=Autostartadresse eingeben!	
2 0 0 3	2 0 0 3 - A	Autostartadresse, z.B. 2003 (nur bei Dateityp A bedeutsam, ansonsten beliebig)	
+	t Y P -	Dateityp eingeben!	
C	t Y P - C	C=Dateityp (Eingabe mit + übergehen: "A")	
+	F I L E - F	F = Filename eingeben *)	
+		Anzeige dunkel, USB-LEDs blinken	
	E x F I L E	Datei existiert bereits (Abbruch)	weiter mit bel. Taste (RESET)
	L C - 80	wenn fertig: RESET	

*) Infolge der Beschränkung durch die HEX-Tastatur sind die "Namen" nur aus den Zeichen 0...9 und A...F zu bilden und nur 4 Zeichen lang.

USB-SAVE/LOAD extern

In eigenen MC-Programmen kann ebenfalls die USB-Schnittstelle benutzt werden. Alle nötigen Angaben sind zuvor in Registern bereitzustellen:

Laden:

```
LD    A,"C"       ;Dateityp
LD    IX,nnnn     ;Anfangsadresse Dateiname
CALL  #A012       ;Laden aufrufen
                  ;geladen wird immer an die im z80-Header enthaltene Anfangsadresse
                  ;wenn geladen, Rückkehr zum Caller
                  ;bei Fehler Rücksprung nach A015h
```

Sichern:

```
LD    A,"A"       ;Dateityp
LD    HL,aadr     ;Anfangsadresse
LD    DE,eadr     ;Endeadresse
LD    BC,sadr     ;Startadresse
LD    IX,nnnn     ;Anfangsadresse Dateiname
CALL  #A018       ;Sichern aufrufen
                  ;wenn fertig, Rückkehr zum Caller
                  ;bei Fehler oder existenter Datei Rücksprung nach  A015h
```

Als Fehlerrücksprung ist auf A015h ein Sprung zu Adresse 0 eingetragen, bei Bedarf dort zu ändern (Stack wird momentan nicht bereinigt!).

Der Dateiname ist in einem Puffer als 0-terminierter ASCII-String bereitzustellen, wobei IX dann auf den Pufferanfang weist. Beispiel:

"T"	"E"	"S"	"T"	"."	"Z"	"8"	"0"	#0D	#00

USB-SAVE/LOAD aus Tiny Basic

Die USB-Schnittstelle wird auch von "rdk Basic V3.2+ LC-80ex" benutzt. Dafür gibt es folgende Anweisungen:

- SAVE Sichern des aktuellen BASIC-Files
- LOAD Laden eines BASIC-Files
- FILES Anzeige des Inhaltsverzeichnisses

TinyBasic benutzt im z80-Format den Dateityp "b". Andere Dateiarten werden abgewiesen. Weitere Beschreibung siehe unter TinyBasic.

USB-INIT

Alle Aufrufe von LOAD/SAVE per HEX-Tastatur sowie deren externen Aufrufe führen automatisch zunächst eine Initialisierung der USB-Schnittstelle durch. Auch in TinyBasic ist das der Fall.

Das manuelle Initialisieren der USB-Schnittstelle kann bei Bedarf per HEX-Tastatur oder externen Aufruf erfolgen:

```
ADR   A00C  EX
```

rdk Basic V3.2+ LC-80ex

Basis:

- TinyBASIC (TDL-Derivat) "rdk Prompt Basic V3.2 3k"
- Beschreibung siehe "Rolf Dieter Klein: BASIC-INTERPRETER" (FRANZIS) oder z.B. hier: http://hc-ddr.hucki.net/wiki/doku.php/z1013:software:tinybasic:rdk

Anpassung für LC-80ex:

- Speicherbelegung geändert
- ? als Abkürzung für PRINT sowie CLS hinzugefügt
- USB-Schnittstelle mit SAVE, LOAD & FILES als Direktkommandos hinzugefügt
- IO-Routinen für Kommunikation per seriellem Terminal (Danke, Holm!)
- Umfang Interpreter (ohne USB-Routinen): <3k

Eckdaten

Speicherbelegung:

2000...<23FF	Arbeitszellen, BASIC-Stack
2400... MAXRAM	BASIC-Text (+42 Byte Eingabepuffer am oberen Ende) Es gibt keine automatische Erkennung des RAM-Endes. Dieses ist in Speicherplatz B006 einzutragen!
B000...BFFF	Interpreter
B000	Kaltstart
B003	Warmstart
B006	MAXRAM (Standardwert: 3FFF)

Variablen:

A...Z	numerisch, -32768...32767
@(I)	numerisches Feld
1 String:	Eingabe per Anweisung I$ nnnn Ausgabe per Anweisung O$ nnnn nnnn=Speicheradresse, z.B. TOP

Zeilenaufbau:

```
HH LL xx ... xx 0D
|  |  |         +-- Endekennung Zeile
|  |  +------------ BASIC-Zeile ASCII-Text, keine Token!
|  +--------------- Zeilennummer low Byte
+------------------ Zeilennummer high Byte
```

Mehrere Anweisungen in einer Zeile:	Trennung durch Semikolon!
höchste Zeilennummer:	32767
max. Zeilenlänge:	Standard 40 Zeichen (130 Zeichen nach END)
Zeileneditor:	keiner (Rückschritt = einzig zulässige Korrekturtaste)

Wichtige Steuerzeichen/-tasten

^C	03	Abbruch bei LIST, RUN, LOAD
^L	0C	Bildschirm löschen + Home
^O	0F	Ausgabe unterdrücken
	08	Kursor <-
	7F	Backspace

Es werden nur Großbuchstaben (sowie Ziffern und Sonderzeichen) korrekt verarbeitet.

Übersicht der Anweisungen

Direktbefehle		
LIST	Listen Basicprogramm	auch LIST zeilennummer
RUN	Ausführen Basicprogramm	
NEW	Löschen des Basicprogramm	
BYE	Beenden Basic	
END nnnn	Speicherende neu festlegen	maximal: END **32767**
SAVE	Programm per USB sichern	als z80-File mit Typ "b" Dateiname wird abgefragt
LOAD	Programm per USB laden	
FILES	Listet Files des USB-Sticks	
Programmierbare Anweisungen:		
LET	arithmetische Zuweisung	A=1 (das LET kann auch entfallen)
IF	Bedingung	
GOTO	Unbedinger Sprung	GOTO 100 oder auch X=10;GOTO X
GOSUB	Unterprogrammsprung	
RETURN	Unterprogrammrücksprung	
REM	Kommentar	
FOR... TO...STEP... NEXT...	Schleife Angabe Schleifenindex nötig!	FOR I=0 TO 10 STEP 2 ... NEXT I
INPUT	Eingabe numerisch nur Zahlen, keine Strings!	INPUT A INPUT "Zahl=",A
PRINT	Ausgabe Variable oder Literal	Abkürzung "?" möglich, Trenner "," unterdrückt anschließendes NL #n für Formatierung auf n Stellen
STOP	Beenden des Programmablaufs	
CALL nnnn	Maschinenprogramm aufrufen	CALL nnnn
OUTCHAR	Einzelzeichenausgabe	OUTCHAR(12)
OUT	Ausgabe an Port	OUT(4)=A
0$ nnnn	Ausgabe eines Textes	O$ nnnn bzw. I$ nnnn nnnn =Speicheradresse nnnn=TOP verwendet freie Adresse
I$ nnnn	Eingabe eines Textes	
POKE	Direkter Speicherzugriff	POKE nnnn,bb
TAB	Tabulator ausgeben	TAB(10)
BYTE	Ausgabe hexadezimal in Byteform	BYTE(15) ergibt "0F"
WORD	Ausgabe hexadezimal in Wortform	WORD(2000) ergibt "07D0"
CLS	Bildschim löschen	
Funktionen:		
RND(n)	Zufallszahl zwischen 0 und n	A=RND(100)
ABS(n)	Betrag	A=ABS(B)
FRE	Freier Speicher	A=FRE

PEEK(n)	Speicherzelle lesen	A=PEEK(nnnn)
INCHAR	Einzelzeicheneingabe	A=INCHAR
HEX(n)	Umwandung hex=>dez	? HEX(FFFF)
IN(n)	Port lesen	A=IN(4)
TOP	Erste freie Speicherzelle	A=TOP
LEN	Länge des Strings (bei I$)	A=LEN
CSTS	Tastaturstatus	A=CSTS
'	Einzelzeichenumwandlung	A='A' ergibt in A den Wert 65
Vergleichsoperatoren:		
>=	größer oder gleich	
<>	ungleich	
>	größer	
<=	kleiner oder gleich	
<	kleiner	
=	gleich	beachte: auch Zuweisung!

Fehler- und sonstige Meldungen

Es gibt wie bei den meisten TinyBasics nur drei Fehlermeldungen:

HOW?	Der Interpreter hat die Anweisung verstanden, sie fordert jedoch etwas Unmögliches...(z.B. Division durch Null)
WHAT?	Der Interpreter versteht die Anweisung nicht (Befehl falsch geschrieben bzw. nicht existent).
SORRY?	Eine (eigentlich korrekte) Anforderung ist vom Interpreter unter den aktuellen Bedingungen (z.B. wegen Speichermangel) nicht zu erfüllen.

Im Zusammenhang mit der USB-Schnittstelle gibt es folgende (Klartext-)Meldungen:

Name too long (8)	Dateiname ist zu lang, darf maxinmal 8 Zeichen lang sein. Erweiterung z80 ist nicht mit anzugeben!	
Not found.	Beim Laden angegebene Datei nicht gefunden	
No TinyBASIC.	Angegebene Datei war keine TinyBasic-Datei	
No z80.	Angegebene Datei war keine (korrekte) z80-Datei	
File exists, overwrite ? (J)	Beim Sichern wurde ein bereits vorhandener Dateiname festgestellt. mit der Eingabe "J" kann diese Datei überschrieben werden. Ansonsten erfolgt Abbruch des Speicherns.	
USB error.	Beim USB-Zugriff ist ein Fehler aufgetreten (zB. kein Adapter angesteckt, USB-Stick nicht angesteckt oder sonstige Ursachen).	
No save.	kein Programm zum Sichern im Speicher	
loading..	Datei wird geladen	vor dem eigentlichen Lade/Speichervorgang wird die USB-Schnittstelle (nochmals) initialisiert (auch bei FILES)
saving...	Datei wird gesichert	

Anmerkungen

- Die für das TinyBasic nötigen Terminaleinstellungen entsprechen den o.a. genannten.
- Die Anweisungen BYTE, WORD und TAB geben direkt auf dem Schirm aus. Sie sind keine Funktionen, denen man zur Ausgabe ein PRINT voranstellen müsste.
- MIT END lässt sich programmtechnisch die benutzte obere RAM-Grenze verändern.
  - ➔ Maximal darf nnnn=32767 betragen, was einem freien Speicher von 24317 Bytes entspricht (Voraussetzung: RAM bis 7FFF bestückt).
  - ➔ Mit END wird gleichzeitig die maximale Zeilenlänge von standardmäßig 40 Zeichen auf 130 Zeichen erhöht.
- Mit BYE wird die Steuerung an den LC-80ex zurückgegeben (Warmstart).
- FILES zeigt alle auf dem USB-Stick vorhandenen Dateien an (Reihenfolge entspricht dem Schreiben auf USB, keine Sortierung).
- SAVE Das aktuell im Speicher befindliche Programm wird als z80-File auf USB-Stick abgelegt. Ist bereits ein gleichnamiges File vorhanden, so erfolgt Rückfrage, ob Überschreiben. Wenn <>"J", Abbruch der Speicheroperation.
- LOAD Laden eines BASIC-Programms. Wird im Ladeversuche keine z80-Datei oder kein Typ "b" gefunden, wird mit entsprechender Meldung abgebrochen. Ansonsten wird das Programm geladen.
- Dateinamen sind nicht hinter SAVE/LOAD anzugeben sondern werden separat nachfolgend abgefragt!
- FILES, SAVE und LOAD initialisieren sicherheitshalber immer erst die USB-Schnittstelle, deswegen dauern die Operationen etwas länger als eigentlich nötig.

Anlage

z80 Headersave

Offsets:

0	ANFANGSADRESSE LOW	
1	ANFANGSADRESSE HIGH	
2	ENDEADRESSE LOW	
3	ENDEADRESSE HIGH	
4	AUTOSTARTADRESSE LOW	
5	AUTOSTARTADRESSE HIGH	
6...0B	(FREI)	"LC80ex"
0C	TYP ("A..."F")	"b" BEI TINY-BASIC-PROGRAMM
0D...0F	3X 0XD3 "MAGIC-BYTE" (Z80-HEADERSAVE-KENNUNG)	
10...1F	PROGRAMM-NAME ASCII	SPEICHERNAME BEI TINY-BASIC-PROGRAMM ANSONSTEN NUR "LC80ex"

```
PUFFER:     EQU  2348H          ;32 Byte Headersave-Puffer
LADADR:     EQU  PUFFER+#20     ;2 Bytes für Ladeadresse
MENGE:      EQU  PUFFER+#22     ;4 Bytes für Lademenge
TYP:        EQU  PUFFER+#26     ;Dateityp
```

Zur Verwendung der Dateitypen gibt es bislang (je nach Rechner) unterschiedliche Definitionen, jedoch keine verbindlichen Vorgaben. Für den LC-80ex wird vorgeschlagen (wie schon bei den RS232-Routinen erwähnt):

A	ausführbares MC-Programm, Autostart
C	ausführbares MC-Programm
D	Daten
b	TinyBasic-Programm ab 2400h

WeRo 01/2016, Stand: 25.01.2016

Schaltplan USB-Adapter