Ein Assembler für den LC-80ex
V1.0

WeRo, Stand: 6. Juli 2016

Inhaltsverzeichnis

Einleitung....2
Voraussetzungen....2
Einschränkungen....2
Speicherbelegung....2
Terminal....2
Bedienung....3
Start....3
Kommandoeingabe....3
Quelltext....4
Eingabe/Bearbeiten....4
Assemblieren und Testen....7
Laden & Speichern....8
Schreibweisen und Konventionen....9
Zahlen....9
Marken....9
Ausdrücke....9
Anweisungen....10
Z80-Standard-Mnemonics....10
Assemblerdirektiven....10
Assembleranweisungen....11
Kommandoübersicht....12
Fehlernummern und -Meldungen....13
Hinweise für eigene Programme auf dem LC-80ex....14
Speicherbelegung....14
Programmende....14
Beispielablauf 1: Programm für Bildschirm und QWERTZ-Tastatur....15
Beispielablauf 2: Programm für LED-Anzeige und HEX-Tastatur....17
Subroutinen aus LC-80-Monitor und LCTOOLS3....18
Anpassung existenter Programme....20

***

Die Screenshots in dieser Doku entstanden im Rahmen der Erprobung am AC1-Emulator. Bei Benutzung von TV-Term (Bildschirmbreite 40 Zeichen) am realen LC-80ex werden alle langen Zeilen (Kommentare, Assemblerausgaben) umgebrochen, was die Lesbarkeit etwas erschwert.

Einleitung

Der LC-80 war darauf ausgelegt, dass sich die Nutzer mit der Programmierung in Maschinensprache befassen. Der LC-80ex nun ermöglicht es durch den Anschluss eines Terminals mit Bildschirm und QWERTZ-Tastatur komfortabel zu arbeiten. Nachfolgend wird ein kompakter, an den geringen RAM-Umfang des LC-80ex angepasster, Assembler vorgestellt. Dieser basiert auf einem Z1013-Programm, welches eine Portierung des Programms "**GENS3**" (ZX Spectrum, Schneider CPC64,...) ist. Eigenschaften:

- 2-PASS-Assembler,
- beherrscht bedingte Assemblierung,
- zahlreiche Pseudoanweisungen,
- Markentabelle,
- Kommentare möglich,
- kein separater Linker nötig.

Voraussetzungen

- Der Assembler benutzt LCTOOLS3 (RS232-/USB-Treiber, Drucken, Hilfsfunktionen).
- Er ist für einen RAM-Ausbau von #2000...#7FFF ausgelegt.
- Die Kommunikation erfolgt per RS232 ("TV-Terminal" oder PC-Terminalprogramm).
- Für das Laden- und Speichern per USB (Dateinamen max. 8 Zeichen, keine Leer-/Sonderzeichen!) ist die entsprechende Hardware nötig.

Einschränkungen

- Mit einer vorausgesetzten RAM-Bestückung von #2000...#7FFF ist die Anwendung auf die Erstellung von ca. 5 kB Maschinencode begrenzt.

Speicherbelegung

Für Quelltext, den erzeugten Maschinencode und die Markentabelle wird Platz im RAM benötigt. Der Speicher ist dazu wie folgt organisiert:

- #2000...#23FF RAM: Arbeitszellen, Puffer und Stack
- #2400...#37FF RAM: 5 KB Platz für den Maschinencode
- #3800...#7FF0 RAM: ca. 18 KB Platz für Quelltext und Markentabelle
- #A000...#BFFF (E)EPROM mit LCTOOLS3 + Maschinencodeeditor
- #C000...#DFFF (E)EPROM mit Assembler

Die gewählte RAM-Verteilung (Grenzwert #3800) zwischen Maschinencode und Quelltext entstand experimentell. Sollte sich herausstellen, dass das je nach Umfang der Arbeiten mit dem Assembler unzweckmäßig ist, kann diese Grenze verschoben werden. Patchzellen im Assembler-EPROM:

`000B:01`	00=kein Gerüst laden bei Kaltstart, 01=Minimalgerüst
`000C:00 38`	Kleinere Werte schaffen mehr Platz für Quelltext, dafür steht weniger Platz für Maschinencode zur Verfügung bzw. umgekehrt.
`000E:F0 7F`	RAM-Obergrenze (max. #9FFF). Das empfiehlt sich jedoch nicht, wenn ein Timekeeper (original auf #6000...#7FFF) benutzt wird. Dessen letzte acht Bytes (Uhrenzellen) sind kein "echter" RAM und führen zu Speicherfehlern bzw. die Uhr wird ggf. verstellt. Deswegen erfolgt in der Standardversion eine Begrenzung auf #7FF0, wenn der Timekeeper benutzt wird.

Terminal

Der Assembler ist für den Betrieb am "TV-Terminal" eingerichtet. PC-Terminal-Emulationen können benutzt werden, wenn sich deren Tastatur entsprechend konfigurieren lässt (Kursortasten: #08...#0B, Backspace: #7F).

Bedienung

Start

Kaltstart:	ADR C000 EX Ein evtl. vorhandener Quelltext wird gelöscht und ein "Gerüst" geladen.
Warmstart:	ADR C003 EX Ein im RAM vorhandener Quelltext bleibt erhalten
Beenden:	>Q(uit) Die Steuerung wird wieder an den LC-80ex-Monitor übergeben. Da „MCE“ zahlreiche Systemzellen im Bereich #2000...#23FF benutzt, erfolgt beim Verlassen ein „Kaltstart“ des LC-80ex (also Abspiel der Startmusik). Damit werden die Monitorarbeitszellen wieder ordnungsgemäß belegt.

Nach Kalt- oder Warmstart befindet man sich im Kommandomodus (Eingabeaufforderung ">").

Kommandoeingabe

Als Antwort auf die Eingabeaufforderung muss der Benutzer nun ein Kommando eingeben. Das sind Buchstaben A...Z, die bis auf Ausnahmen die Anfangsbuchstaben der (englischsprachigen) Funktionalitäten sind. Mit dem H-Kommando kann die Liste aller verfügbaren Kommando-buchstaben und deren Bedeutung abgerufen werden.

- Einige Kommandos können mit angehängten Parametern versehen werden, andere wieder benötigen unbedingt (sinnvolle) Parameter. Maximal sind vier Parameter (abhängig vom Kommando) möglich:

```
X a,b,c,d    <ENTER>
| | | +++--- Zeichenketten mit max. 20 Zeichen Länge
| +++------- Zahlen im Bereich von 1...32767
+----------- auszuführendes Kommando
```

- Die Kommandoeingabe ist (im Gegensatz zum Maschinencode-Editor) immer mit <ENTER> abzuschließen.

- Kommandos können als Klein- oder Großbuchstaben eingegeben werden. Leerzeichen zwischen Kommandobuchstabe und Parameter sind nicht nötig.

- Die zuletzt eingegebenen Parameter werden behalten und können beim nächsten Kommando (ohne explizite Angabe) wiederverwendet werden.

- In der Kommandozeile sind folgende Sondertasten wirksam:
  Kursor links oder Backspace: löschen des letzten Zeichens
  Kursor rechts oder TAB: Kursor weiter zur nächsten TAB-Position
  ENTER: Kommandoabschluss
  Strg+C: Abbruch

Beginnt eine Kommandoeingabe (versehentlich) nicht mit einem Buchstaben sondern mit einer Ziffer, so wird das als Eingabe der Zeilennummer einer Quelltextzeile interpretiert!

Quelltext

Bei Kaltstart wird ein "Gerüst" geladen (das lässt sich bei Bedarf ändern, siehe hier). Es stellt das kleinste "Assembler-Programm" dar und ist bereits assemblier- und funktionsfähig.

```
10         ORG  #2400
20         ENT  $
30 M1      RET
```

Der Assembler macht daraus ein Maschinenprogramm ab Adresse #2400 (dafür steht die ORG-Anweisung), bestehend aus einem einzigen Operationscode (C9). Die Pseudo-Anweisung "ENT $" definiert die Startadresse. Das ermöglicht ein bequemes Starten des Maschinenprogramms ab angegebener Adresse aus dem Assembler heraus.

Um ein "richtiges" Programm in Assemblersprache zu stellen, stehen zahlreiche Kommandos zur Verfügung. Prinzipiell wird wie mit einem Texteditor gearbeitet.

Eingabe/Bearbeiten

Nach der Kommandoaufforderung (>) wird eine Eingabe der Ziffern 0...9 als Zeilennummer angesehen. Es kann damit eine Assemblerzeile erstellt werden. Diese hat folgendes Eingabeformat:

50		**M1**		**LD**		**HL,#1000**		**;abc**	
Zeilen-nummer	Leertaste	Marken-name	Kursor-rechts	Mnemo Befehl	Kursor-rechts	Mnemo Operand	Kursor-rechts	Kommentar	ENTER

- Neben Ziffern und einigen Sonderzeichen sind (außer in Kommentaren) nur Großbuchstaben erlaubt.
- Ist keine Marke vorhanden, muss trotzdem mit <Kursor-rechts> tabuliert werden. Ansonsten wird der Ausdruck nicht als Mnemo gewertet sondern als Marke!
- Kommentare beginnen mit einem Semikolon. Sie sind sowohl als ganze Zeile (; als Marke) als auch im Anschluss an den Operanden möglich. Kommentare sollten aber sparsam eingesetzt werden, da sie wertvollen Speicherplatz im RAM belegen!
- Die maximale Länge einer Eingabezeile beträgt 64 Zeichen. Bei einer Bildschirmbreite von 40 Zeichen werden die restlichen Zeichen (der Kommentare) so umgebrochen, dass in der nächsten Zeile nach der Zeilennummernposition fortgesetzt wird.

I (a(,b)	**I**(nsert)	**Einfügen automatisch**

Das I-Kommando kann benutzt werden, um für fortlaufende Eingabe automatisch eine Zeilennummer zu erstellen (wie BASIC-Funktion "AUTO"). Beispiele:

```
I 10,1= erste Zeilennummer 10, Abstand 1
I     = es werden die gespeicherten Parameter benutzt (Standard: 10,10)
```

Nach der Zeilennummer wird das erste Leerzeichen (s.o.) automatisch eingefügt. Man beginnt sofort mit dem Markennamen oder einem <KURSOR-rechts>. Ist keine keine neue Zeile mehr gewünscht, kann mit Strg+C unmittelbar nach der erschienenen (nicht benötigten) Zeilennummer abgebrochen werden.

C a,b	**C**(opy)	**Kopieren einer Zeile**

Soll der Inhalt einer Zeile an einen anderen Stelle nochmals eingefügt werden, so benutzt man das C-Kommando. Beispiel:

```
C10,100     kopiert die gesamte Zeile 10 als neue Zeile 100
            existierte Zeile 100 bereits, so wird diese überschrieben.
```

D a,b	**D**(elete)	**Löschen mehrerer Zeilen**

Um eine einzelne Zeile zu löschen, so genügt die Eingabe einer Zeilennummer und <ENTER>. Mehrere Zeilen auf einmal löscht man mit dem D-Kommando. Beispiel:

```
D50,80      löscht alle Zeilen 50...80 (angegebene Zeilen eingeschlossen)
```

F a,b,f,s	**F**(ind)	**Finden String**

Im Bereich zwischen a und a (angegebene Zeilen nicht eingeschlossen!) wird nach dem String f gesucht. Wurde ein Vorkommen gefunden, so wird die Zeile im Zeileneditor zur Bearbeitung angezeigt. Es kann dann (im Zeileneditor) nach weiteren Vorkommen gesucht werden oder der String f durch den angegeben Ersatzstring s ersetzt werden.

N a,b	(re)**N**(umber)	**Neu nummerieren**

Der Quelltext wird neu nummeriert, beginnend mit Zeile a und der Schrittweite b. Würde sich eine neue Zeilennummer >32767 ergeben, so wird das Kommando nicht ausgeführt und die alte Nummerierung bleibt erhalten. Beispiel:

```
N10,5       Nummeriert alles neu ab Zeile 10, Schrittweite 5
```

L (a(,b))	**L**(ist)	**Listing auf Schirm ausgeben**

Die eingegebenen Zeilen können zur Kontrolle ausgegeben werden. Beispiele:

```
L           gibt den kompletten Quelltext aus
L10         Listen ab Zeile 10 bis Ende
L50,100     Listen Zeile 50 bis Zeile 100
```

Umfasst die Ausgabe mehr als 20 Zeilen (Standard), so hält diese nach der Ausgabe von "!" an. Strg+C bricht ab, jede andere Taste setzt fort.

S n	**S**(top)	**Anhalten bei Ausgabe**

Die Zeilenanzahl, nach der bei der LIST-Ausgabe ein Stopp erfolgt, kann für Dauer der Sitzung mit dem S-Kommando geändert werden: Beispiel:

```
S10         LIST stoppt nun nach 10 Zeilen
```

P (a(,b))	**P**(rint)	**Listing ausdrucken**

Wirkt wie das L-Kommando, aber das Listing wird (zusätzlich) auf den Drucker gegeben. Es gelten die gleichen Parameter, ebenso die Einstellung der Zeilenanzahl (z.B. S50, je nach Blattlänge), ehe ein Stopp eingelegt wird. Nach erfolgtem Stopp kann ein neues Blatt eingelegt werden (oder bei Endlospapier einfach mit <ET> weiter.
Drucker vorher bereitmachen, sonst hängt die Ausgabe fest!

V	**V**(iew)	**Parameter anzeigen**

Das V-Kommando zeigt die aktuellen Parameter an. Beispiel:

```
>V
#3800   20    10    10
|       |     |     +--- Parameter 2
|       |     +--------- Parameter 1
|       +--------------- Stopp-Zeilenanzahl
+----------------------- Anfangsadresse Quelltext
```

E n	**E**(dit)	**Bearbeiten Zeile n**

Es wird die Quelltextzeile mit der angegebenen Nummer in einen Puffer kopiert und angezeigt. In der nächsten Bildschirmzeile erfolgt dann die Bearbeitung. Das geschieht zunächst nur im Puffer, nicht im Quelltext selbst. Damit kann der Originalzustand jederzeit wiederhergestellt werden.

In diesem Modus wird ein Kursor in der Zeile bewegt, der die aktuelle Bearbeitungsposition angibt. Mit verschiedenen Tasten lassen sich dann Unterkommandos zur Bearbeitung erteilen:

Leertaste	gehe zum nächsten Textzeichen (nur bis ans Zeilenende)	
Backspace	gehe auf vorheriges Textzeichen zurück (kein Löschen!)	
Kursor rechts	gehe vorwärts bis zur nächsten TAB-Position	
ENTER	Beenden der Bearbeitung, alle Änderungen übernehmen	
Strg+C	Beenden der Bearbeitung, alle Änderungen verwerfen (wie Q)	
C	(C)HANGE	markiert akt. Position mit +, dann Ändern (Überschreiben) möglich. Backspace in diesem Submodus bewegt den Zeiger 1 Schritt nach links, <Kursor rechts> hat keine Wirkung. In diesem +-Modus bleibt man so lange, bis ENTER gedrückt wird. Der Zeiger steht dann hinter dem zuletzt geänderten Zeichen.
F	(F)IND	Finden des nächsten Vorkommens des (mit F-Kommando definierten) Suchstrings. Wenn nicht gefunden, wird der EDIT-Modus verlassen. Wenn gefunden, dann wird die betreffende Zeile in den Editor geholt.
I	(I)NSERT	Markiert akt. Position mit * , dann Einfügen möglich. Backspace in diesem Submodus bewirkt, dass das Zeichen links vom Zeiger gelöscht wird. <Kursor rechts> bewegt den Zeiger bis zur nächsten TAB-Position und fügt Leerzeichen ein. In diesem *-Modus bleibt man so lange, bis ENTER gedrückt wird. Damit kehrt man in den Hauptmodus zurück, der Zeiger steht hinter dem zuletzt eingefügten Zeichen.
K	(K)ILL	Löschen des Zeichens an der aktuellen Zeigerposition
L	(L)IST	Zeile komplett anzeigen, Zeiger auf den Zeilenanfang
Q	(Q)UIT	Beenden der Bearbeitung, alle Änderungen verwerfen
R	(R)ELOAD	Buffer erneut laden, alle Änderungen verwerfen
S	(S)UBSTITUTE	Ersetzen des gefundenen Strings durch den vorher definierten String und dann automatisch wieder FIND. Damit lässt sich Schritt für Schritt ein Textfile durcharbeiten.
X	E(X)TEND	Erweitern der Zeile (Anhängen weiterer Zeichen). Kursor geht ans Zeilenende, dann automatisch INSERT.
Z	(Z)AP	Löschen aller Zeichen ab Kursor bis Zeilenende

Assemblieren und Testen

Ist der Quelltext geschrieben (oder per USB geladen), kann er in Maschinencode übersetzt werden.

A	**A**(ssemble)	**Assemblieren**

Das A-Kommando startet den Assembliervorgang. Abgefragt wird "Options (1/2/4/8/16/32/?)". Normalerweise reicht hier <ENTER>. Bei Bedarf kann per Zifferneingabe eine zusätzliche Funktion gewählt werden (? zeigt diese Kurzhilfe zu Optionen an):

```
  1= mit Markentabelle
  2= nur testen, kein Code
  4= kein Listing
  8= Ausgabe auch auf Drucker
 16= anderer Adressbereich
 32= kein Test auf RAM
 (mehrere Optionen: Addition)
```

Option=4 (keine Ausgabe des Listings) beschleunigt die Assemblierung erheblich!

Ergibt das Assemblieren Fehler, so werden die (erste) fehlerhafte Zeile und die Art des Fehlers (*ERROR* xx) angezeigt und der Vorgang angehalten. Mit der Taste "E" kann die betreffende Zeile sofort in den Editor geholt und bearbeitet werden. Jede andere Taste setzt den Assembliervorgang (mit Fehler) fort, sofern möglich. Die undokumentierten ("illegalen") Z80-Befehle bewirken unterschiedliche Fehlermeldungen, z.B. *ERROR*01.

Überschreitet die Länge des Quelltextes bzw. die der erzeugten Markentabelle die RAM-Obergrenze, so ergeht die Nachricht "OUT OF SPACE". Die den Fehler verursachende Quellzeile wird in den Editor geholt und kann bei Bedarf geändert (gekürzt) werden.

Überschreitet der erzeugte Maschinencode die festgelegten Speichergrenzen, so erfolgt die Nachricht "BAD ORG". Auch in diesem Fall wird die fehlerverursachende Zeile in den Editor geladen.

Spezialfall: Option=16
Hiermit ist es möglich, Maschinencode für andere Adressbereiche als #2400...#37FF zu erzeugen, z.B. auf #1000. Im Quelltext ist dazu lediglich wie gewohnt die entsprechende ORG-Adresse anzugeben. Eine Ausführung aus dem Assembler heraus (G-Kommando) ist jedoch nicht möglich. Das gilt sowohl für adressgebundene als auch relokatible Programme. Eine ENT-Anweisung wird ignoriert. Der generierte Maschinencode ist zunächst mit dem O-Kommando zu sichern und kann dann (außerhalb des Assemblers) zur Ausführung auf seine originale Adresse geladen werden.

Nach erfolgreicher Assemblierung ist die Anzahl noch freier Bytes (S=Quelltext, O=Objectcode) ersichtlich. Ist die Direktive "ENT" enthalten und OPTION<>16, so wird auch noch die Startadresse angezeigt. Danach kann das erzeugte Programm mit dem G-Kommando getestet werden:

G	**G**(o)	**Anwenderprogramm starten**

Das G-Kommando startet das Programm aus dem Assembler heraus. Wurde das Programm fehlerfrei abgearbeitet und endet es mit RET (C9), so kehrt die Steuerung zum Assembler zurück (Promptzeichen >).

M	**M**(CE)	**Debugger starten**

Es wird der Maschinencode-Editor aus LCTOOLS3 aufgerufen. Wurde vom Assembler Maschinencode erzeugt, so steht er in "MCE" zur Verfügung und kann dort getestet werden (z.B. Einzelschrittbetrieb). **Bislang nicht gespeicherter Assemblerquelltext geht verloren. Es ist kein anschließender Warmstart des Assemblers (#C003) zulässig!**

Laden & Speichern

Die USB-Unterstützung (LCTOOLS3) ermöglicht bequemes Laden und Speichern. Dateinamen dürfen max. 8 Zeichen lang sein und keine Leer-/Sonderzeichen enthalten. Eine Dateierweiterung (Z80 bzw. SRC) ist **nicht** mit anzugeben! Vor jedem der nachfolgend genannten USB-Kommandos erfolgt zunächst immer eine Initialisierung der USB-PIO. Das bedeutet zwar kleinen Zeitverzug, sichert aber eine korrekte Funktion auch nach Fehlern im Anwenderprogramm (die ggf. die PIO umprogrammiert haben) oder nach einem USB-Stick-Wechsel.

U	**U**(SB dir)	**USB Inhalt**

Das U-Kommando zeigt die auf dem USB-Stick befindlichen Dateien an. Das erfolgt unsortiert, die Reihenfolge entspricht i.a. in der Reihenfolge des Schreibens auf den Stick. Sind mehr als 20 Dateien enthalten, so wird die Ausgabe angehalten ("!"). Ein beliebiger Tastendruck setzt fort, Strg+C bricht eine weitere Ausgabe ab.

Die folgenden Kommandos fordern zunächst zur Eingabe eines Dateinamens auf. Ein Abbruch des Vorgangs ist durch <ENTER> bei leerem Namen oder Strg+C möglich.

O	**O**(bject code)	**Speichern Objektcode**

Nach erfolgreicher Assemblierung kann das Programm mot dem O-Kommando als lauffähiger Objektcode (Maschinencode) abgespeichert werden. Das erfolgt im z80-Headersave-Format mit Dateityp "C". Die benötigten Angaben (Anfangs- und Endeadresse) werden selbständig ermittelt. Wurde eine vom Programmanfang abweichende ENT-Direktive angegeben, so gilt diese Adresse als Startadresse, ansonsten ist Startadresse = Anfangsadresse.

Existierte bereits eine gleichnamige Datei, so erfolgt eine Abfrage "File exists, overwrite? (Y)". Ist kein gültiger Maschinencode im Speicher, so erfolgt keine Reaktion auf das O-Kommando.

R	**R**(ead)	**Laden Quelltext**

Der abgespeicherte Quelltext kann mit dem R-Kommando auch wieder geladen werden. Geladen werden nur Dateien mit der Endung *.SRC (diese Erweiterung nicht mit angeben!). Es können auch *.SRC Dateien geladen werden, die mit "MCE" als Disassemblat erzeugt und abgespeichert wurden. Die Formatierung ist zwar etwas abweichend, kann aber mit Suchen/ Ersetzen (Find+Substitute: Leerzeichen->TAB) leicht angepasst werden.

Überschreitet die Länge des zu ladenden Quelltextes den verfügbaren Platz im RAM (ca. 17kB), dann erfolgt eine Meldung "OUT OF SPACE" und das Laden wird abgebrochen. Der Datei-Inhalt selbst wird beim Laden nicht geprüft, also keine anderweitigen "fremden" *.SRC-Dateien laden! Ein evtl. im Speicher befindlicher Quelltext wird ohne Rückfrage überschrieben.

W	**W**(rite)	**Speichern Quelltext**

Das erstellte Programm kann mit dem W-Kommando als Assembler-Quelltext abgespeichert werden (zwecks späterer weiterer Bearbeitung oder Archivierung). Eine solche Quelltext-Datei wird mit der Endung *.SRC versehen und hat ein spezielles Format. Da diese Funktion bei der Programmerstellung häufig benutzt wird, um Zwischenstände des gleichen Programms unter dem gleichen Namen zu sichern, wurde hier auf die etwas lästige "Überschreiben?"-Abfrage verzichtet! Ist kein gültiger Quelltext im Speicher, so erfolgt keine Reaktion auf das W-Kommando.

Fehlermeldungen:

USB error.	allgemeine Fehlermeldung, z.B. kein USB-Stick angesteckt
Not found.	die beim Laden angegebene Datei wurde nicht gefunden
Name too long (8)	angegebener Dateiname ist zu lang (max. 8 Zeichen zulässig)

Schreibweisen und Konventionen

Zahlen

- Hexadezimalzahlen: vorangestelltes #
- Binärzahlen: vorangestelltes %
- Dezimalzahlen: ohne Präfix/Suffix

Marken

- Eine Marke (Label) ist ein Symbol, das einen 8- oder 16-bit-Wert repräsentiert. Sie gibt entweder die Adresse einer Anweisung an oder definiert (zusammen mit EQU) eine Konstante.
- Eine Marke muss mit einem Buchstaben beginnen. Es sind nur die ersten 6 Zeichen signifikant! Diese müssen eindeutig sein, d.h. die Zeichenfolge darf nicht noch einmal auftreten (ansonsten '*ERROR* 4, Marke kann nicht redefiniert werden).
- Reservierte Worte (die Registernamen und Flags) dürfen nicht als Markennamen benutzt werde, können jedoch in Markennamen enthalten sein:

A	**B**	**C**	**D**	**E**	**H**	**L**	**I**	**R**	**$**
AF	**AF'**	**BC**	**DE**	**HL**	**IX**	**IY**	**SP**		
NC	**Z**	**NZ**	**M**	**P**	**PE**	**PU**			

- Markenbezeichnungen können einen nachgestellten Doppelpunkt enthalten, müssen das aber nicht.
- Im Operanden kann auch das Symbol "$" als "Zielmarke" benutzt werden, um sich auf den aktuellen Wert des Adresszählers zu beziehen.

Ausdrücke

Als *Ausdruck* in einem Operanden kann entweder eine Konstante, eine Marke oder eine Kombination daraus unter Verwendung eines Rechenoperators benutzt werden.

Als Rechenoperatoren sind möglich: Sie haben keine Wertigkeit sondern werden stur von links nach rechts abgearbeitet! Beispiel: `LD HL,(ANFANG) + #100`	`+` Addition
	`-` Subtraktion
	`&` logisches AND
	`@` logisches OR
	`!` logisches XOR
	`*` Ganzzahlmultiplikation
	`/` Ganzzahldivision
	`?` MODULO-Funktion a ? b = a - (a/b)*b

Es gibt keinen Vergleichsoperator, der als Resultat einen Wahrheitswert liefern würde. Im Fall der IF/ELSE/END-Direktive muss es also heißen:

```
 10 TEST    EQU 0       ;0: FALSCH, <>0 : WAHR
...
100         IF  TEST    ;auch z.B. IF TEST+1, aber nicht IF TEST=0 !!!
...
120         END
```

Anweisungen

Zu den vom Assembler verarbeiteten Anweisungen zählen die Z80-Befehle ("Mnemonics"), Assemblerdirektiven und Assembleranweisungen.

Z80-Standard-Mnemonics

ADC	**CPD**	**DI**	**IN**	**JR**	**NOP**	**POP**	**RLA**	**RRCA**	**SRA**
ADD	**CPDR**	**DJNZ**	**INC**	**LD**	**OR**	**PUSH**	**RLC**	**RRD**	**SRL**
AND	**CPI**	**EI**	**IND**	**LDD**	**OTDR**	**RES**	**RLCA**	**RST**	**SUB**
BIT	**CPIR**	**EX**	**INDR**	**LDDR**	**OTIR**	**RET**	**RLD**	**SBC**	**XOR**
CALL	**CPL**	**EXX**	**INI**	**LDI**	**OUT**	**RETI**	**RR**	**SCF**	
CCF	**DAA**	**HALT**	**INIR**	**LDIR**	**OUTD**	**RETN**	**RRA**	**SET**	
CP	**DEC**	**IM**	**JP**	**NEG**	**OUTI**	**RL**	**RRC**	**SLA**	

keine undokumentierten ("illegalen") Befehle!

Assemblerdirektiven

Diese Pseudo-Anweisungen steuern die Arbeit des Assemblers. Sie werden nicht in Operationscodes umgewandelt.

ORG n	setzt die Anfangsadresse des Maschinencodes auf den Wert von n *(*muss im Bereich #2400...#3800 liegen) ist am Anfang i.A. entbehrlich (ohne: Anfangsadresse=#2400) Eine evtl. zweite ORG-Anweisung muss immer eine höhere Adresse angeben als die vorhergehende. Der Zwischenraum wird mit 00-Bytes gefüllt.	
EQU n	dem EQU muss eine Marke vorausgehen. Damit wird deren Wert auf den Wert von n gesetzt. N darf kein Symbol enthalten, dem noch kein Wert zugewiesen wurde. (*ERROR* 13)	
DEFB n1,n2,...	jedes n muss ein Byte darstellen, z.B. ASCII-Zeichen	
DEFW n1,n2,...	definiert jeweils ein Wort (zwei Bytes, zuerst LSB, dann MSB)	
DEFS n	Reservierung der durch n angegebenen Anzahl Bytes, der Wert der Bytes im reservierten Raum ist unbestimmt!	
DEFM *"HALLO"*	definiert einen ASCII-String, kein Ausdruck möglich	
ENT n	Angabe der Startadresse (wichtig für G-Kommando) nötig auch, wenn Startadresse <> Anfangsadresse wenn n=$, dann Bezug auf akt. Befehlszähleradresse	
IF n	nur wenn Wert von n <> 0 ("WAHR") , dann werden die folgenden Zeilen bis zu ELSE bzw. END ausgeführt.	Bedingungen dürfen nicht geschachtelt werden. Eine Prüfung/ Fehlermeldung erfolgt nicht, aber das Resultat ist undefiniert!
ELSE	Alternative zu IF	
END	Ende der Bedingung	

Für n kann eine Konstante oder ein Ausdruck stehen.

Assembleranweisungen

Eine Assembleranweisung wird im Quelltext aufgeführt und dient ausschließlich zur Gestaltung der Ausgabe des beim Assemblieren erzeugten Listings (nicht beim L-Kommando). Im Normalfall weist dieses folgende Form auf:

```
2400                10        ORG   #2400
2400                20        ENT   $
2400 C9             30 M1     RET
```

Adress-zähler	Maschinencode	Zeilen-nummer	Marke	Befehl	Operand

Mit einer Assembleranweisung lässt sich diese Darstellung bei Bedarf modifizieren. Man kann z.B. einzelne Zeilen nicht ausgeben, den Code weglassen oder Leerräume einfügen.

Eine Assembleranweisung beginnt im Quelltext nach der Zeilennummer und einem Leerzeichen mit einem Stern "*", gefolgt von einem Buchstaben. Dieser bestimmt die Art des Kommandos und damit das Aussehen des Listings.

***E**	erzeugt drei Leerzeilen auf dem Schirm oder Drucker
***Hs**	der String s wird als Überschrift genommen und nach jedem *E ausgegeben.
***S**	die Ausgabe des Listings wird angehalten, eine beliebige Taste setzt es fort
***L-**	beginnend mit dieser Zeile wird die Ausgabe abgeschaltet
***L+**	ab dieser Zeile wird das Listing wieder ausgegeben (Standard)
***D+**	der Wert des Adresszählers wird dezimal ausgegeben
***D-**	der Wert des Adresszählers wird hexadezimal ausgegeben (Standard)
***C-**	kürzt das Listing ab der nächsten Zeile (der Teil "Maschinencode" wird nicht ausgegeben)
***C+**	schaltet Vollausgabe des Listings ein (Standard)

Alle anderen möglichen Assembleranweisungs-Buchstaben werden ignoriert.

Wenn die Assemblierung durch eine Bedingung stellenweise ausgeschaltet wurde, ist natürlich für diesen Bereich der Effekt irgend einer Assembleranweisung ebenfalls ausgeschaltet.

Kommandoübersicht

Kommando	Parameter	Name	Wirkung
A	-	(A)ssemble	Assemblierung starten, abgefragt wird: „Options...“
C	a,b	(C)opy	Zeile a wird dupliziert und als neue Zeile b eingefügt
D	a,b	(D)elete	Zeilenbereich a...b wird gelöscht
E	a	(E)dit	Zeile bearbeiten, siehe extra Kapitel
F	a,b,c,d	(F)ind	Zeilen zwischen a und b (ohne a und b selbst) werden nach dem String c durchsucht und (wenn d angegeben) jedes Vorkommen mit e ersetzt neue Suche mit "F" ohne Parameter möglich
G	-	(G)o	Starten des MC-Programms (ENT nötig)
H	-	(H)elp	Kurzhilfe mit Kommandoliste
I	[a[,b]]	(I)nsert	automat. Einfügemodus (wie "AUTO" in BASIC) optional: a=1. Zeilennummer, b=Abstand Beenden des Modus: Strg+C
L	[a[,b]]	(L)ist	Anzeigen Quelltext, optional: a= ab Zeile, b=bis Zeile
M	-	(M)CE	Debugger starten (#B000)
N	a,b	re(N)umber	Quelltext neu nummerieren a=1. neue Zeilennummer, b=Schrittweite
O	-	(O)bjectcode	Maschinencode speichern per USB
P	[a[,b]]	(P)rint	wie LIST, aber (auch) auf Drucker
Q	-	(Q)uit	Verlassen des Assemblers
R	-	(R)ead	Quelltext laden per USB
S	a	(S)top	Zeilenanzahl für LIST/PRINT setzen (1...255) Nach Stopp: Abbruch mit ^C oder Fortsetzung jede andere Taste ohne Parameter: Standard=20 setzen
U	-	(U)SB	USB Inhaltsverzeichnis
V	-	(V)iew	aktuelle Parameter anzeigen: Anfangsadresse Quelltext, Zeilenanzahl, Parameter 1 und Parameter 2
W	-	(W)rite	Quelltext sichern per USB

Mehrere Parameter sind durch ein Komma zu trennen.

Fehlernummern und -Meldungen

ERROR 1	allgem. Fehler im Zusammenhang mit dieser Zeile
ERROR 2	Mnemonik nicht erkannt
ERROR 3	Anweisung falsch formatiert
ERROR 4	Symbol mehrfach definiert
ERROR 5	Zeile enthält unzulässiges Zeichen
ERROR 6	einer der Operanden ist unzulässig
ERROR 7	eine Marke ist ein reserviertes Wort
ERROR 8	Register passen nicht zusammen
ERROR 9	zu viele Register in dieser Zeile
ERROR 10	8-bit-Ausdruck erwartet, jedoch > 8 Bit geliefert
ERROR 11	Anweisungen JP (IX+n) und JP (IY+n) sind unzulässig
ERROR 12	Assemblerdirektive falsch
ERROR 13	unzulässiger Vorwärtsbezug (EQU auf noch nicht definiertes Symbol)
ERROR 14	Division durch Null
ERROR 15	Überlauf bei Multiplikation
BAD CMD.	Kommandoeingabe ungültig
BAD MEMORY.	kein Platz im RAM mehr, um weiteren Quelltext zu ergänzen
BAD ORG.	Der durch die Adressvorgabe mit ORG bestimmte Anfang bzw. das Maschinenprogramm-Ende überschreiten vorgegebene Grenzen.
OUT OF SPACE!	zu wenig Speicher für Quellcode+Markentabelle.

Hinweise für eigene Programme auf dem LC-80ex

Speicherbelegung

- Zu erstellende neue Maschinenprogramme sollten immer ab #2400 (oder höher) beginnen und dürfen max. bis #37FF reichen (ca. 5kB). Nur dann sind sie in Verbindung mit Assembler und LCTOOLS3 lauffähig.
- Das Assemblieren mit Option=16 ermöglicht prinzipiell auch andere Adressbereiche. Diese Option kann auch beim Nachvollziehen von Beispielen aus der Bedienungsanleitung angewendet werden, wenn Programme ab Adresse #2000 beginnen. Der Test solcher Programme aus dem Assembler heraus ist jedoch nicht möglich.
- Eine ORG-Anweisung am Programmanfang stellt den definierten Beginn des Maschinenprogramms sicher. Fehlt diese, wird immer bei #2400 begonnen.
- Eine ENT-Anweisung ist immer dann nötig, wenn das erstellte Programm
  - ➔ entweder vom Assembler aus (testweise) gestartet werden soll oder
  - ➔ eine von der Anfangsadresse abweichende Startadresse hat.
- Kommentare im Assemblerprogramm sind möglich, belegen aber wertvollen Speicherplatz im Quelltextbereich. Bei längeren Programmen sind sie deshalb auf ein Minimum zu beschränken.
- Für Systemadressen kann man symbolische Namen definieren (z.B. "DAK2 EQU #0483") oder die direkte Adresse (z.B. CALL #0483) benutzen. Letzteres wird nötig, wenn der Speicher nicht reicht (OUT OF SPACE).

Programmende

- Der Ausstieg aus einem Anwenderprogramm muss je nach gewünschter Bedienung überlegt werden. Es kommen infrage:
  - ➔ Endlosschleife, nur mit RES zu verlassen
  - ➔ Sprung zu Adresse 0000 (Software-Reset LC-80ex)
  - ➔ Ein RET ist nur während der Testphase am Assembler sinnvoll. Damit kehrt das Anwenderprogramm an dieser Stelle wieder zum Assembler zurück (sofern nicht die Systemzellen im Bereich #2000...#23FF überschrieben wurden).
- Je nachdem, welche Veränderungen durch das Anwendungsprogramm im Bereich #2000...#23FF vorgenommen werden, erfolgt bei einem Sprung zu Adresse 0000 (RESET) entweder ein Warmstart des LC-80ex oder ein Kaltstart (Anfangsmelodie erklingt).

Beispielablauf 1: Programm für Bildschirm und QWERTZ-Tastatur

Nachfolgend wird die Erstellung des klassischen "HALLO WELT!" als ausführbares Maschinenprogramm mit dem Assembler beschrieben. Ein- und Ausgabe sollen dabei per QWERTZ-Tastatur und Bildschirm erfolgen. Die Formulierung des Ablaufs ist die eigentliche "Kopfarbeit", die der Assembler aber nicht abnehmen kann...

1. Kaltstart des Assemblers, das Gerüst wird geladen
   `RES ADR C000 EX am LC-80ex`

2. I-Kommando für automatisches Einfügen neuer Zeilennummern ab 30
   `>I30  <ET>`
   `Eingabe der Zeilen 30...140`

30	→ LD HL,M2→ ;TEXTANFANG	Die Kommentare müssen nicht mit eingegeben werden und dienen hier nur der Erläuterung. Die CALLS #A024 und #A021 sind im Sprungverteiler von LCTOOLS3 verankert, siehe auch hier. → steht für <KURSOR-rechts>-Taste <= beendet die Eingabe
40	M0→ LD A,(HL)→ ;ZEICHEN HOLEN	
50	→ INC HL→ ;NAECHSTE POSITION	
60	→ CALL #A024→ ;AUSGABE ZEICHEN	
70	→ CP 0→ ;ENDE ERREICHT?	
80	→ JR NZ,M0→ ;NOCH NICHT	
90	M1→ CALL #A021→ ;TASTE?	
100	→ JR Z,M1→ ;NOCH NICHT	
110	→ JP #0000→ ;JA=>RESET	
120	M2→ DEFB #0C→ ;BS LOESCHEN	
130	→ DEFM "HALLO WELT!"	
140	→ DEFB 0→ ;TEXTENDE	
150	Strg+C	

3. Kontrolle Listing:
   `L <ET>`

```
>L
   10          ORG   #2400
   20          ENT   $
   30          LD    HL,M2      ;TEXT
   40 M0       LD    A,(HL)
   50          INC   HL
   60          CALL  #A024      ;AUSGABE
   70          CP    #00
   80          JR    NZ,M0
   90 M1       CALL  #A021      ;TASTATUR
  100          JR    Z,M1
  110          JP    #0000
  120 M2       DEFB  #0C
  130          DEFM  "HALLO WELT!"
  140          DEFB  #00
>
```

4. Quelltext per USB abspeichern
   `>W <ET>`
   `SAVE SOURCE`
   `Filename: WELT    (nur max. 8 Zeichen!)`
   Gespeichert wird nun der Quelltext als "WELT.SRC"

5. Assemblierung starten:
   >A <ET>
   OPTIONS (1/2/4/8/16/32/?): <ET>

```
>A
OPTIONS (1/2/4/8/16/32/?):

*ASSEMBLER LC-80ex*V1.0 USB*

PASS1: ERRORS= 00

2400                10          ORG   #2400
2400                20          ENT   $
2400 211424         30          LD    HL,M2     ;TEXT
2403 7E             40 M0       LD    A,(HL)
2404 23             50          INC   HL
2405 CD24A0         60          CALL  #A024     ;AUSGABE
2408 FE00           70          CP    #00
240A 20F7           80          JR    NZ,M0
240C CD21A0         90 M1       CALL  #A021     ;TASTATUR
240F 28FB          100          JR    Z,M1
2411 C30000        110          JP    #0000
2414 0C            120 M2       DEFB  #0C
2415 48414C4C      130          DEFM  "HALLO WELT!"
2420 00            140          DEFB  #00

PASS2: ERRORS= 00

FREE:   S=18158
        O= 5087

(G)O:   #2400

>
```

6. Programm starten:
   >G <ET>

```
HALLO WELT!
```

   Da der Programmausstieg mit einem RESET endet (Zeile 110) wird nach dem Drücken einer Taste das "Hallo Welt!"-Programm sowie der Assembler verlassen. Die LED-Anzeige des LC-80ex leuchtet wieder.

   Nach einem Warmstart des Assemblers (ADR C003 EX) steht der Quelltext jedoch weiter zur Verfügung.

   Möchte man den Assembler zum Testen nicht verlassen, so ist statt dem RESET-Sprung in Zeile 110 (zumindest zeitweilig) ein RET einzutragen.

7. Maschinencode per USB speichern
   >O <ET>
   SAVE SOURCE
   Filename: WELT (nur max. 8 Zeichen!)
   Gespeichert wird nun das fertige ausführbare Maschinenprogramm "WELT.Z80".

Beachte:	Dieses Maschinenprogramm benutzt LCTOOLS3 und ist auf LC-80ex ohne diesen Treiber nicht funktionsfähig!

Es lassen sich mit dem Assembler jedoch auch "normale" Programme erstellen, die keine speziellen oder nur Routinen aus dem Monitor benutzen.

Beispielablauf 2: Programm für LED-Anzeige und HEX-Tastatur

"Hallo Welt" als abgewandeltes "HELPUS" aus der Originalanleitung:

1. Kaltstart des Assemblers, das Gerüst wird geladen:
   `RES ADR C000 EX am LC-80ex`
2. I-Kommando für automatisches Einfügen neuer Zeilennummern ab 30
   `>I30  <ET>`
3. Eingabe der Zeilen 30...250 (Zeilen 10 und 20 aus dem Gerüst übernommen)

```
 10          ORG  #2400        ;STANDARDANFANG
 20          ENT  $            ;
```

```
 30 DAK2     EQU  #0483
 40 SOUND1   EQU  #0370
 50          LD   HL,HALLO     ;TEXT 1
 60          PUSH HL
 70          LD   IX,WELT      ;TEXT 2
 80 LOOP     EX   (SP),IX      ;IM WECHSEL
 90          LD   B,#80        ;FÜR DAUER
100 LOOP1    CALL DAK2         ;ANZEIGEN
110          DJNZ LOOP1
120          CP   #17          ;MINUS-TASTE GEDRÜCKT?
130          JR   Z,RAUS       ;JA = > ENDE
140          CALL TON          ;NEIN: TONAUSGABE
150          JR   LOOP         ;UND WEITER
160 TON      LD   HL,#0100     ;TONLÄNGE
170          PUSH BC           ;WEIL BC IN LOOP1 und SOUND1 BENUTZT!
180          CALL SOUND1       ;TON
190          POP  BC
200          RET
210 RAUS     POP  HL           ;STACK REPARIEREN (DA PUSH IN ZEILE 60!)
220          RET               ;GGF. ÄNDERN IN JP 0
240 HALLO    DEFB 0,#E8,#C2,#C2,#6F,#6B    ;Codierung Text "HALLO"
250 WELT     DEFB 0,0,#CA,#C2,#CE,#E4      ;Codierung Text "WELT"
```

4. Kontrolle Listing, ggf. Ausdruck:
   `L <ET>` bzw. `P <ET>`
5. Quelltext per USB abspeichern
   `>W <ET>`
   `SAVE SOURCE`
   `Filename: WELT2  (nur max. 8 Zeichen!)`
   Gespeichert wird nun der Quelltext als "WELT2.SRC"
6. Assemblierung starten:
   `>A <ET>`
   `OPTIONS (1/2/4/8/16/32/?):  <ET>`
7. Programm starten:
   `>G <ET>`
   Auf der LED-Anzeige sollte nun "HALLO" und "WELT" im Wechsel erscheinen und danach jeweils ein kurzer Ton ausgegeben werden. Mit Drücken von "-" auf der HEX-Tastatur wird das Programm beendet. Es kehrt zum Assembler zurück. Soll das Programm endgültig fertiggestellt werden, so ist in Zeile 220 das RET durch "JP 0" zu ersetzen (Neuassemblierung nötig!). Es wird dann mit einem RESET des LC-80ex beendet.
8. Maschinencode per USB speichern
   `>O <ET>`
   `SAVE SOURCE`
   `Filename: WELT2  (nur max. 8 Zeichen!)`
   Gespeichert wird nun das fertige ausführbare Maschinenprogramm "WELT2.Z80".

Subroutinen aus LC-80-Monitor und LCTOOLS3

- Werden Routinen aus LCTOOLS3 (z.B. Bildschirmausgabe, Tastatur) benutzt, so sind damit erstellte Maschinenprogramme auch nur auf solchen LC-80ex lauffähig, die LCTOOLS3 installiert haben.
- Soll ein Programm nur für LED-Display, HEX-Tastatur und "Piepser" programmiert werden, so können dazu die Monitorroutinen herangezogen werden. Solche Programme sind dann auch auf allen anderen LC-80ex ausführbar.

Monitor:

Es werden hier nur diejenigen Routinen aufgeführt, die in der Originalanleitung beschrieben werden. Im "Buschendorf"-Monitor gibt es weitere Routinen.

Name	Adr	Wirkung
SOUN1K	#0370	Ausgabe eines Tonsignals von 1 kHz, Dauer in HL
SOUN2K	#0374	Ausgabe eines Tonsignals von 2 kHz, Dauer in HL
SOUND	#0376	Tonsignal für Lautsprecher, Höhe in C, Dauer in HL
RAMCHK	#0452	Test ob ein Speicherplatz (Adr. in HL) im RAM liegt
DAK1	#045A	wie DAK2, aber wartet solange bis Taste gedrückt, in A steht dann interner Tastencode
DAK2	#0483	1x Anzeige alle 6 LED-Stellen (IX=Textanfang = letzte Stelle!), Tastaturabfrage: in A steht Positionscode der Tastatur
ADRSDP	#04B7	DE als Adresse in Adressenspeicher
DADP	#04C3	A als Daten in Datenspeicher
ONESEG	#04CA	Umwandeln einer Ziffer in den entsprechenden 7-Segment-Code
TWOSEG	#04D9	Umwandeln des Inhalts des A-Registers in 2 x 7-Segment-Code Ablage auf Adresse HL bzw. HL+1
MONMUS	#04EA	Spielen der Anfangsmusik
MUSIK	#04EE	Spielen von Musik , IY=Startadresse Daten
DISP3	#0583	Text "HALLO" (im "Buschendorf"-Monitor nicht verfügbar!)
DISP4	#0589	LEER-TEXT
TXTERR	#05A5	Text "error"

Anzeigespeicher LEDs:

#23F7	#23F6	#23F5	#23F4	#23F3	#23F2
Adresse				Daten	

Tastencodes (hexadezimal):

Taste	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	EX	LD	ST	ADR	DAT	+	-
DAK1	**00**	**02**	**02**	**03**	**04**	**05**	**06**	**07**	**08**	**09**	**0A**	**0B**	**0C**	**0D**	**0E**	**0F**	**12**	**1F**	**1E**	**19**	**14**	**10**	**11**
DAK2	**04**	**05**	**06**	**07**	**08**	**09**	**16**	**0B**	**0C**	**0D**	**12**	**0F**	**10**	**11**	**0E**	**13**	**00**	**02**	**01**	**14**	**15**	**0A**	**17**

LCTOOLS3:

Es lassen sich prinzipiell alle im Sprungverteiler aufgeführten Routinen nutzen. Dabei ist zu beachten, dass damit erstellte Maschinenprogramme logischerweise auch nur auf solchen LC-80ex funktionieren, die LCTOOLS3 installiert haben.

Für die Ein- und Ausgabe per Tastatur und Bildschirm/Drucker können z.B. folgende Routinen Anwendung finden:

<table>
<tr><th>Adresse</th><th colspan="2">Wirkung</th></tr>
<tr><td>#A021</td><td colspan="2">Abfrage der QWERTZ-Tastatur ohne Halt
Keine Taste: Rückkehr mit gesetztem Z-Flag, A=0
Taste gedrückt: Rückkehr mit nicht gesetztem Z-Flag, ASCII-Zeichen in A</td></tr>
<tr><td>#A024</td><td colspan="2">Ausgabe A als Zeichen auf Schirm, kein Register wird verändert</td></tr>
<tr><td>#A040</td><td colspan="2">Ausgabe A als HEX-Zahl auf Schirm, kein Register wird verändert</td></tr>
<tr><td rowspan="2">#A043</td><td>wenn vorher HL=0</td><td>wenn vorher HL=nnnn</td></tr>
<tr><td>Hexadezimalausgabe des Registers DE auf Schirm (0000...FFFF)</td><td>Ablage des Wertes im RAM ab Adresse nnnn als HEX-ASCII-String</td></tr>
<tr><td rowspan="2">#A046</td><td>wenn vorher HL=0</td><td>wenn vorher HL=nnnn</td></tr>
<tr><td>Dezimalausgabe des Registers DE auf Schirm (00000...32767)</td><td>Ablage des Wertes im RAM ab Adresse nnnn als DEZ-ASCII-String</td></tr>
<tr><td>#A027</td><td>Drucker ein</td><td rowspan="2">Druck erfolgt parallel zur Schirmausgabe *)</td></tr>
<tr><td>#A02A</td><td>Drucker aus</td></tr>
</table>

*)
Die normale (parallele) Druckausgabe ermöglicht kein Senden von Steuerzeichen außer #0D/#0A. Ist das z.B. für die Einstellung eines Druckers (z.B. Schriftart) nötig, muss es (ohne Benutzung von Drucker ein/aus) "zu Fuß" erfolgen, z.B. so:

```
SIOAC: EQU  #DE
SIOAD: EQU  #DC
...
      CALL  #A01E       ;INITIALISIEREN DRUCKER
...
...                     ;AUSZUGEBENDES (STEUER-)ZEICHEN IN A
      PUSH  AF          ;AUSGABEZEICHEN SICHERN
      XOR   A           ;REGISTER RR0
      OUT   (SIOAC),A   ;AUSWÄHLEN
CHK:  IN    A,(SIOAC)   ;RR0 LESEN
      BIT   2,A         ;SENDEBEREIT?
      JR    Z,CHK       ;NOCH NICHT!
      POP   AF
      OUT   (SIOAD,A    ;ZEICHEN AN DRUCKER SENDEN
;...
```

Anpassung existenter Programme

Sollen vorhandene Programme z.B. aus der Bedienungsanleitung des "LC-80" mit dem LC-80ex-Assembler nachvollzogen werden, so ist folgendes zu beachten:

Form der Listings:
Abgedruckte Listings sind meist eine Mischform, die neben den eigentlichen Z80-Mnemonics auch Adressen und Codebytes enthalten (wie sie ein Disassembler liefern kann), z.B.:

```
2400  21 00 00    START:      LD    HL,0
...
```

Für die Übernahme in den LC-80ex-Assembler sind nur die markierten Teile abzutippen. Fehlt eine ORG-Anweisung, so ist diese der Adresse des ersten Codebytes zu entnehmen (im Beispiel: ORG #2400)

Schreibweise:
Hexadezimalzahlen werden häufig durch ein nachgestelltes H kenntlich gemacht. Im LC-80ex-Assembler müssen diese mit vorangestelltem # beginnen, z.B. 2400H => #2400

Adressbereich:
Vorhandene Programme ab #2000 sind so umzuschreiben, dass sie erst ab #2400 beginnen (im einfachsten Falle sind alle angegebenen Adressen um #0400 zu erhöhen). Dazu müssen

- ORG-Anweisungen,
- direkte Adressbezüge (sollten aber gleich durch Marken ersetzt werden) und
- das high-Byte von Interruptroutinen (der Wert, mit welchem das I-Register geladen wird)

geändert werden.

Includes:
Die z.B. bei *V.POHLERS* (http://www.homecomputer-ddr.de.vu/) zu findenden Programme in Assemblersprache (*.ASM) benötigen meist ein Includefile (LC80.INC), in welchem wichtige System-Adressen definiert sind. Da der LC-80ex-Assembler kein Hinzuladen von Includes ermöglicht, sind aus Gründen der Minimierung des Quelltextes nur die nötigen Definitionen (z.B. Portadressen) aufzunehmen. Geht es mit dem Quelltextspeicher knapp zu, muss man ganz auf die Definitionen verzichten und die Systemadressen in den Befehlen direkt angeben.

Stellt man fest, immer wieder fast die gleichen Definitionen zu benötigen, so speichert man sich diese als Quelle ab.
Die Vorlage-Datei wird dann normal wie jede andere Quell-Datei zu Beginn der Arbeit geladen und unter dem neuen Programmnamen abgespeichert.
	`10          ORG   #2400`
	`20          ENT   $`
	`30 CTC0:    EQU   #EC`
	`30 DAK1:    EQU   #045A`
	`40 DAK2:    EQU   #0483`
	`50 DATLED:  EQU   #23F2`

Makros:
Der LC-80ex-Assembler beherrscht keine Makros. Stößt man auf solche, so sind die Funktionen entsprechend nachzubilden. Beispiel LO und HI:

LO liefert die unteren 8 Bit eines Ausdrucks
HI verschiebt den Ausdruck im 8 Bits nach rechts. War der Ausdruck ein Word, so wird als Resultat das high-Byte zurückgegeben.

Je nach benutztem Assembler können im Original-Quelltext weitere Pseudoanweisungen vorhanden sein. Hierzu ist festzustellen, welcher Assembler benutzt wurde und was die Anweisungen dort bewirken. Die Anweisung "CPU Z80" kann z.B. bedenkenlos weggelassen werden.

Am "Beispiel 10: Uhr mit Wecker" aus der Originalanleitung werden die nötigen Änderungen beschrieben:

Original	LC-80ex-Assembler: Definitionen ergänzen #2000 => #2400 (Adressversatz + #400) für Teststart aus Assembler direkte Adressen => Marken HEX-Werte: # statt H
	CTC0: EQU #EC DAK2: EQU #0483 MONMUS: EQU #04EA ADRSDP: EQU #04B7 DADP: EQU #04C3 DATLED: EQU #23F2
ORG 2000H	ORG #2400 ENT $
EX10: IM 2	EX10: IM 2
LD A,22H ;INTERRUPT-VEKTOR!	LD A,#26 ;#26xx
LD I,A	LD I,A
XOR A	XOR A
OUT (CTC0),A	OUT (CTC0),A
LD A,OA5H	LD A,#A5
OUT (CTC0),A	OUT (CTC0),A
LD A,OE9H	LD A,#F0 ;geändert für 921 kHz
OUT (CTC0),A	OUT (CTC0),A
M1: LD IX,DATLED	M1: LD IX,DATLED
CALL DAK2	CALL DAK2
JR C,M2	JR C,M2
XOR A	XOR A
LD (2216H),A	LD (WFLG),A
M2: LD A,(2216H)	M2: LD A, (WFLG)
CP 55H	CP #55
CALL Z,MONMUS	CALL Z,MONMUS
EI	EI
JR M1	JR M1
ORG 2040H	ORG #2440
PUSH AF	INTR: PUSH AF
PUSH BC	PUSH BC
PUSH DE	PUSH DE
PUSH HL	PUSH HL
LD HL,2210H	LD HL,ZWI
LD B, 15H	LD B, #15
CALL INCT	CALL INCT
JR NZ,EXIT	JR NZ,EXIT
LD B,60H	LD B,#60
CALL Z,INCT	CALL Z,INCT
CALL Z,INCT	CALL Z,INCT
LD B,24H	LD B,#24
CALL Z,INCT	CALL Z,INCT
LD A,(2211H)	LD A,(SEC)
CALL DADP	CALL DADP
LD HL,(2212H)	LD HL,(MIN)
EC DE, HL	Ex DE, HL

```
        CALL   ADRSDP
        LD     HL,(2212H)
        LD     DE,(2214H)
        AND    A
        SBC    HL,DE
        JR     NZ, EXIT
        LD     A,(2211H)
        CP     0
        JR     NZ,EXIT
        LD     A,55H
        LD     (2216H),A
EXIT:   POP    HL
        POP    DE
        POP    BC
        POP    AF
        EI
        RETI
INCT:   LD     A,(HL)
        ADD    A,1
        DAA
        LD     (HL),A
        SUB    B
        JR     NZ,NEXT
        LD     (HL),A
NEXT:   INC    HL
        RET
        ORG    2200H
        DEFW   2040H
        ORG    2210H
        DEFS   1   ; ZWISCHENZÄHLER
        DEFS   1   ; SEKUNDEN
        DEFS   1   ; MINUTEN
        DEFS   1   ; STUNDEN
        DEFS   1   ; WECKMINUTEN
        DEFS   1   ; WECKSTUNDEN
        DEFS   1   ; WECKFLAG
```

```
        CALL   ADRSDP
        LD     HL,(MIN)
        LD     DE,(WMIN)
        AND    A
        SBC    HL,DE
        JR     NZ, EXIT
        LD     A,(SEC)
        CP     0
        JR     NZ,EXIT
        LD     A,#55
        LD     (WFLG),A
EXIT:   POP    HL
        POP    DE
        POP    BC
        POP    AF
        EI
        RETI
INCT:   LD     A,(HL)
        ADD    A,1
        DAA
        LD     (HL),A
        SUB    B
        JR     NZ,NEXT
        LD     (HL),A
NEXT:   INC    HL
        RET
        ORG    #2600 ;INT.-VEKTOR
        DEFW   INTR     ;ADR.INT-ROUTINE
        ORG    #2610
ZWI:    DEFS   1
SEC:    DEFB   0     ; damit wird die Uhr
MIN:    DEFB   0     ; für Demozwecke
STD:    DEFB   #12   ; auf 12:00:00 gesetzt
WMIN:   DEFB   #01   ; um 12:01 erfolgt
WST:    DEFB   #12   ; "Wecken"
WFLG:   DEFS   1
```

Beachten:
Das Programm ist für einen Systemtakt von 900 kHz ausgelegt. Mit obiger Korrektur läuft es auch bei 921 KHz etwa richtig. Beim Test im Assembler (Systemtakt 1,8 MHz) läuft die Uhr logischerweise doppelt so schnell.

Für einen "Echtzeitbetrieb" sind die Uhrenzellen SEC (#2611) bis WST (#2616) vor Starten des assemblierten Programms mit der aktuellen Zeit/Weckzeit zu belegen.