Erweiterter S/P-BASIC-Interpreter Version 3.2 für den BCS3

Autor: Frank Prüfer

Der 4 KByte große BASIC-Interpreter für den BCS3 [1] in der Version „BASIC-SE 3.1" entsprechend [2] wurde in der Praxis umfassend erprobt. Die Erweiterungen und Ergänzungen gegenüber der Version 2.4 gemäß [1] waren sehr zu begrüßen, jedoch ergaben sich dabei aus der Sicht des Anwenders noch einige Probleme, die die Nutzbarkeit des Systems teilweise einschränken.

Vom Verfasser erfolgte deshalb mithilfe eines Reassemblers eine Erforschung des Maschinencodes und darauf aufbauend eine vollständige Überarbeitung des gesamten Interpreters. Obwohl das System von Eckhard Schiller bereits sehr gut durchdacht war, konnten doch durch konsequentes Weiterverfolgen der vorliegenden, teilweise ausgezeichneten Grundgedanken noch über 100 Byte "herausoptimiert" werden, wodurch für verschiedene Erweiterungen und kleine Verbesserungen Platz geschaffen wurde, ohne dabei die Hardwaregrenze von 4 KByte ROM überschreiten zu müssen.

Im folgenden wird, aufbauend auf [2], der neue BASIC-Interpreter für den BCS3 vorgestellt. **Dabei wird in diesem Artikel ausschließlich auf die Änderungen bzw. Neuerungen gegenüber der Version „BASIC-SE 3.1" lt. [2] eingegangen.**

Das Hexadezimallisting zu diesem Programm zeigen die Tafeln 1...4.

Der Einsatz der geänderten Software erfordert keinerlei Hardware-Umbau gegenüber [2]. Anders formuliert: Auf einem mit dem Interpreter in der Version 3.1 bereits fehlerfrei lauffähigen BCS3 ist die neue Version 3.2 des BASIC-Interpreters nach Umprogrammierung der EPROMs sofort nutzbar.

Der Name wurde geändert in „**S/P-BASIC V3.2**" („S/P" steht für „Schiller/Prüfer").

Konkret wurde folgendes modifiziert/erweitert:

1. Berechnung von Ausdrücken

In der Version 3.1 verbirgt sich hier ein störender Programmfehler: Verkettete Ausdrücke mit mehreren verschiedenen Operatoren wurden völlig falsch berechnet. Z.B. ergab 12+4/2*2 fehlerhafterweise 20. Richtig müßte diese Aufgabe unter Berücksichtigung der in S/P-BASIC zum Einsatz gelangenden "gleitenden Priorität" bei der Arithmetik als Ergebnis 13 liefern. (Multiplikation vor Division; d.h. nach interner Auflösung des Ausdrucks in die Form „12+(4/(2*2))".)
Diese Mängel sind in der Version 3.2 beseitigt worden.

Zur Information hier noch einmal die Reihenfolge der insgesamt 10 logischen und arithmetischen Operatoren von S/P-BASIC V3.2 in aufsteigender Priorität:

a) Vergleiche: **= # < >**
b) Logische Verknüpfungen: **OR AND**
c) Arithmetik: **+ - / ***

Wichtig ist insbesondere bei der Arithmetik, daß in S/P-BASIC sowohl Addition und Subtraktion als auch Multiplikation und Division jeweils nicht gleichwertig sind, sondern Subtraktion immer vor Addition und Multiplikation immer vor Division ausgeführt wird, auch wenn diese Operatoren weiter rechts im Ausdruck stehen. (Siehe das Beispiel für die „interne Klammerung" von Ausdrücken oben.)

Bei verschiedenen Funktionen wurde in der Version 3.1 kein Test auf Integer-Parameter durchgeführt; z.B. bei **IN(X)**, **PEEK(X)**, **USR(X)**, obwohl dies in [2] so behauptet wurde. Dieser Fehler ist jetzt beseitigt worden; die Übergabe von Nicht-Integer-Parametern führt nunmehr zu einer entsprechenden Fehlermeldung: „?I ERROR"

2. Zufallszahlengenerator

Der Zufallszahlengenerator von [2] ist zwar an das für BASIC allgemein übliche angenähert (0 < RND < 1); für die meisten Anwendungen als Heimcomputer ist aber ohnehin eine Normierung der Zufallszahl auf Integerwerte erforderlich (z.B. bei Spielen). Des weiteren fehlte die Möglichkeit, einen zufälligen Startwert für den Zufallsgenerator festzulegen und die erzeugten Zufallszahlenfolgen wiesen

einen geringen Umlaufzyklus auf (Wiederholung derselben Werte nach kurzer Zeit). In der Version 3.2 gilt nunmehr für die Arbeit mit Zufallszahlen:

- **RND(X)** ergibt eine Integer-Zufallszahl zwischen 1 und X; X muß Integer sein. Bei kleinem X dauert die Berechnung ggf. sehr lange, man kann dann aber z.B. statt RND(3) schreiben: INT(RND(2999)/1000+1)
- Mit der Anweisung **RANDOMIZE** kann man einen zufälligen Startwert, der sich aus dem R-Register des U880 ergibt, für den Zufallsgenerator festlegen und so nichtreproduzierbare Zufallszahlenfolgen erzeugen. Werden mit POKE auf die Adresse 3C0CH und 3C0DH zwei beliebige Werte 0...7FH geschrieben, wird der Zufallsgenerator mit einem festen Anfangswert initialisiert und es ergeben sich dadurch reproduzierbare Zufallszahlenfolgen.

3. Arbeit mit Maschinencode-Programmen

Diese gestaltet sich bei [2] wegen der bei "Y=USR(X)" notwendigen "Hilfsvariablen" Y etwas umständlich. Es wurde deshalb zusätzlich die allgemein übliche CALL-Anweisung für Unterprogramme im Maschinencode realisiert:

- **CALL X** ruft ein Maschinencode-Unterprogramm mit der Adresse X auf (X = Integer). Dieses Unterprogramm darf außer den Hintergrundregistern des U880 alle Register benutzen. (Die Hintergrundregister werden im Rahmen des Bildschirminterrupts verwendet und dürfen vom eigentlichen BASIC-Interpreter keinesfalls verändert werden.)

Damit steht „USR(x)" nunmehr für echte Anwenderfunktionen auf Maschinencode-Ebene zur Verfügung, wobei hierfür noch die Möglichkeit geschaffen wurde, auch von S/P-BASIC aus an das Unterprogramm einen Parameter zu übergeben. Es gilt in der Version 3.2 nunmehr für die USR-Funktion:

- **USR(X,Y)** Aufruf eines Maschinencode-Unterprogramms mit der Adresse X (X = Integer). Der Wert von Y wird dem Unterprogramm in den Registern CBLH übergeben (Y kann auch entfallen). Die Rückgabe des berechneten Ergebnisses an BASIC erfolgt ebenfalls in den Registern CBLH.
  Für die Werteübergabe gelten die Aussagen zum Internformat des Interpreters s. [2], S. 562 oben. Strings werden intern wie folgt dargestellt:
  C = 0FEH: Kennung für Zeichenkette
  B = Länge der Zeichenkette (1...0FFH)
  HL = Anfangsadresse der Zeichenkette.

4. Stringverarbeitung

In der Version 3.2 wurde auch die Umkehrung der CHR$(X)-Funktion realisiert:

- **ASC(X$)** liefert den ASCII-Wert des ersten Zeichens von X$.
  Bsp:
  ```
  10 DIM A$(10)
  20 A$='EIN TEXT'
  30 ?%ASC(A$(4,2))
  ```
  würde den Wert 20H (Code für Leerzeichen) ergeben.
  Man kann damit z.B. Groß- in Kleinbuchstaben umwandeln.

Bei allen Zeichenketten darf das letzte Hochkomma dann als Begrenzer entfallen, wenn dieses unmittelbar vor dem Zeilenende steht; d.h. ENTER und zweites Hochkomma sind bei Literalen gleichwertig. Man spart damit bei Befehlen wie LET, PRINT oder REM jeweils ein Byte Speicherplatz für den BASIC-Code.

Nach Eingaben wird jetzt übrigens das in Version 3.1 verbliebene und optisch störende Endekennzeichen auf dem Bildschirm (schwarzes Quadrat) immer gelöscht. Es empfiehlt sich dazu, im Zeichengenerator-EPROM die letzten 8 Byte (Adr. 03F8H bis 03FFH) wie folgt zu ändern: „00 00 00 00 01 00 00 00"

Bisher war keine Abfrage auf ungedrückte Tasten bei **INKEY$** möglich. INKEY$ liefert einen String mit der Länge 1 und dem Tastencode als Inhalt; das ist bei ungedrückter Taste eine binäre 0. Diese könnte man theoretisch mit CHR$(0) abfragen, was aber in der Version 3.1 aufgrund eines Programmfehlers nicht ging (Kollision mit einer bereits anderweitig verwendeten Merkzelle im RAM).
Jetzt ist diese Möglichkeit gegeben – z.B.:

```
40 IF INKEY$=CHR$(0) GOTO 40: REM 'Warten auf Tastendruck
```

5. LIST-Kommando

Die feststehende **LIST**-Zeilenzahl von 12 erlaubte es in Version 3.1 nicht, bei einer kleinen Bildschirm-Zeilenzahl alle BASIC-Zeilen anzuzeigen, da sie dann teilweise über den oberen Rand des Bildschirms hinausrollten. In der Version 3.2 wird die LIST-Zeilenzahl automatisch in Abhängigkeit von der beim Start vom Anwender definierten Bildschirm-Zeilenzahl (BSZZ) berechnet, und zwar wie folgt: „INT(BSZZ/2)-1"; bei 21 Bildschirm-Zeilen ergeben sich also 9 LIST-Zeilen.

Die END-Zeile wird jetzt nicht mehr mit ausgegeben, da sie ohnehin nur interpreterintern von Bedeutung ist. (In Version 3.2 hat sie die Nummer 9984 (d.h. 2700H), denn die Zeilen 9984...9998 können generell nicht geLISTet werden.)

Die Ausgabe der freien Bytes im RAM erfolgt nunmehr mit dem ausgeschriebenen Text "XXXX bytes free". Schlüsselworte werden wieder wie bei Version 2.4 [1] in Kleinbuchstaben umgewandelt, was deutlich übersichtlicher ist.

6. PRINT-Anweisung

Zusätzlich besteht bei **PRINT** jetzt die Möglichkeit, eine wahlfreie Tabulierung auszuführen. Diese erfolgt durch:

- **&X** (X = Integer) es werden so viele Leerzeichen ausgegeben, wie der Wert von X ergibt (nur L-Teil signifikant, 0 entspricht 256!), vergleichbar also der SPC(X)-Funktion in anderen BASIC-Systemen und # bei [1].

7. Nutzbarer Zeilennummernbereich

In der Version 3.2 können jetzt auch die Zeilennummern 7680 bis 7935 gelöscht und überschrieben werden, was vorher aufgrund eines Programmierfehlers nicht ging. Der nutzbare Zeilennummernbereich umfaßt die Zahlen von 0 bis 9983; 9984 ist die END-Zeile (vgl. oben unter 5.) - diese darf keinesfalls überschrieben werden, sonst kommt es zum Systemabsturz!

8. Erweiterung des Stackbereichs

In der Version 3.1 beträgt der Stackbereich nur ca. 80 Byte (3C30H...3C7FH). Dies ist für verschiedene Anwendungen entschieden zu wenig. Vor allem, da sich am Anfang des genannten Bereichs noch der Eingabezeilenpuffer befindet, der den Stack weiter verringert; insbesondere dann, wenn die Eingabezeile schon recht umfangreich ist (bei der 3,5-MHz-Version kann sie selbst fast 80 Zeichen lang sein). Deshalb wurde in der Version 3.2 der Stackbereich um 32 Byte erweitert (3C30H...3C9FH), wodurch die Verschachtelungstiefe bei Schleifen, Ausdrücken, Unterprogrammen usw. entsprechend tiefer werden darf. Die Einschränkung für den nutzbaren BASIC-Programmspeicher ist kaum von Bedeutung, da man ohnehin fast immer mit einer Speichererweiterung arbeiten muß (vgl. [2]). Für die 2,5-MHz-Version des BCS3 reduziert sich die maximale Zeilenzahl damit von 29 auf 28, wobei der Fernseh-Bildschirm dann aber recht symmetrisch bezogen auf seinen oberen und unteren Rand ausgefüllt ist. (Die 3,5-MHz-Variante erlaubt - wie bereits mit BASIC-SE 3.1 - weiterhin nur, max. 21 Bildschirmzeilen zu nutzen.)

9. Anmerkung zu den FOR...NEXT-Schleifen

NEXT bezieht sich interpreterintern stets auf die letzte aktive FOR-Schleife (vergleichbar einer schließenden Klammer in allgemeinen Ausdrücken). Zur besseren Übersichtlichkeit kann man nunmehr aber an **NEXT** den Variablennamen anfügen, der jedoch vom BASIC-Interpreter nicht ausgewertet wird.

Und noch ein Hinweis: Aus programmtechnischen Gründen muß der Endwert ausgehend vom Anfangswert durch Addition der Schrittweite stets exakt erreicht werden. Bei "normalen" Schleifen (Anfangswert < Endwert, beide Integer, Schrittweite=1), ist das ohne Bedeutung, kann aber bei anderen Schrittweiten zu Problemen führen (Folge: "rätselhafte" Endlosschleifen). Ggf. muß man Zählschleifen bilden, die die jeweilige Abbruchbedingung mit Vergleichsoperatoren abfragen. („Zählwert > Endwert?")

10. Fehlerbehandlung

In S/P-BASIC V3.2 wird ein Fehler wie folgt gemeldet: „**?x ERROR**"; anschließend wird bei mit RUN gestarteten BASIC-Programmen die Zeile ausgegeben, in der der Fehler aufgetreten ist. Die Fehlerkennzeichen '**x**' wurden gegenüber [2] Tafel 8 z.T. geändert und dem allgemein üblichen englischen Sprachgebrauch angepaßt:

alt	**neu**	
@	**U**	**U**ndefined character error: Nichtdefiniertes Zeichen; weder Zahl noch Schlüsselwort
B	**S**	**S**yntax error
U	**O**	**O**verflow error: Überlauf bei PLOT/UNPLOT oder Feldern; ein Index <= 0 führt ebenfalls zu einem O-Error
Z	**L**	**L**ine not found: Zeile nicht vorhanden
B(nach LOAD)	**T**	**T**ape error: Fehler bei der Magnetbandarbeit (LOAD). Steht vorher der Text "RECORD ERROR" (Satzfehler), so wurde ein Prüfsummenfehler festgestellt, es ist dann unbedingt "NEW" einzugeben! Ansonsten wurde der Blockanfang nicht erkannt. Nach LOAD muß die ENTER-Taste noch während des Kenntones gedrückt werden!
=	**S**	ersetzt durch „**S**" gemäß oben (hinter einem fehlenden Gleichheitszeichen bei "LET" verbirgt sich fast stets eine falsche Anweisung, also ein Syntaxfehler)

11. Abschließende Hinweise

Das Kassettenarbeitssystem wurde vom Prinzip her nicht geändert. Weil aber die Ausgabe im BASIC-Internformat erfolgt, können Programme, die mit BASIC-SE 3.1 auf Kassette ausgelagert wurden, mit S/P-BASIC V3.2 nicht mehr abgearbeitet werden, da nunmehr neue interne Codezeichen (Token) für die einzelnen Schlüsselworte gelten. Sämtliche Programme müssen also mit Version 3.2 neu erfaßt werden.

Will man weiter mit Version 3.1 arbeiten, kann man den oben unter 1. genannten Fehler bei der Berechnung verketteter Ausdrücke beheben, indem man die Inhalte der ROM-Zellen 083EH und 083FH entsprechend [2], Tafel 3 austauscht.

Der Bildschirmtreiber wurde zur symmetrischeren Auslastung des sichtbaren horizontalen Bildschirmbereiches dahingehend geändert, daß die Fernsehzeilen jetzt etwas weiter links beginnen. Sollen sie um 1/2 Zeichenstelle nach rechts verschoben werden, so ist dies möglich, indem man auf Adresse 0051H den Wert 9FH und auf 0053H 23H schreibt. Zur Verschiebung um eine ganze Zeichenstelle nach rechts gilt: 0054H und 0055H neu 00H und E9H.

Die 3,5-MHz-Variante mit 40 Zeichen/Zeile konnte vom Verfasser aufgrund des Fehlens der schnellen Bauelemente nicht praktisch erprobt werden. Tafel 5 zeigt die Änderungen bei Version 3.2, die nur eine sinngemäße Übernahme von [2], Tafel 7 auf die gegenüber BASIC-SE 3.1 teilweise geänderten Adressen darstellen.

Zur schnellen Überprüfung des Datenerhaltes der EPROMs kann man mit folgender Routine die ROM-Quersumme ermitteln:

```
S=0: FOR I=0 TO 0FFFH: S=S+PEEK(I): NEXT I: ?S
```

Das Ergebnis muß **432060** lauten.

Verwendet man zur feineren Eingrenzung eines Fehlers das in [2] Tafel 6 angegebene Programm, so gelten die nachstehenden 16 Einzelprüfsummen der 256-Byte-Blöcke im Hexadezimalformat:

6FDD	**5492**	**6692**	**69BA**	**729A**	**78F4**	**6E45**	**6D01**
7577	**7AAF**	**663E**	**5909**	**5A9C**	**660E**	**6CC3**	**5F53**

Der Verfasser ist der Meinung, daß die beschriebenen Erweiterungen das umständliche Abtippen der Tafeln 1...4 durchaus rechtfertigen. (Leider können programmierte EPROMs immer noch nicht in Zeitschriften gedruckt werden...) Der Interpreter der Version 3.2 lastet nunmehr die zur Verfügung stehende Hardware des BCS3 wirklich optimal aus.

Anhang: Wichtige Adressen (Unterprogramme u.ä.)

hexadezimal	dezimal	
0028	40	Ausgabe eines Zeichens aus A auf den Bildschirm (=Befehl RST 28H; U880-Maschinencode: 0EFH)
0036	54	Sperren des Bildschirminterrupts, der Rechner arbeitet dann bis zur nächsten INPUT- oder PRINT-Anweisung wesentlich schneller
0056	86	Löschen des Bildschirms
00C6	198	Eingabe eines Zeichens von der Tastatur nach A ohne Echoausgabe; wartet auf Tastendruck
00E0	224	Tastaturstatusabfrage: A=0 wenn gerade keine Taste gedrückt, sonst A=Tastencode; wartet nicht auf Tastendruck
0130	304	Eingabe einer mit ENTER abgeschlossenen Zeichenkette von der Tastatur auf den Bildschirm; Ausgang: IY=Anfangsadr. der Zeile, Endekennzeichen der Zeile: <DEL> (ASCII=7FH)
0241	577	Startadresse für BASIC-Kommandoeingabeebene; Einsprung in das Interpreter-Hauptprogramm (entspricht END-Anweisung)
0377	887	Ausgabe alphanumerischer Texte ab DE bis einschließlich Endekennung <NL> (1EH) oder <NUL> (00H; dann wird kein abschließender Zeilenwechsel ausgegeben)
0820	2080	Berechnung beliebiger alphanumerischer Ausdrücke Eingang: IY=Anfangsadresse des ASCII-codierten Ausdrucks, Endekennzeichen z.B. Komma (2CH), Doppelpunkt (3AH) oder <NL> (1EH); der Ausdruck darf alle Operatoren enthalten (AND=0C6H, OR=0C5H); Standardfunktionen müssen im Intern-Format angegeben sein (Code 0BCH...0C4H, vgl. Schlüssel-wort-Tabelle). Ausgang: CBLH: Berechnungsergebnis; IY: Adresse des Endekennzeichens des Ausdrucks; DE und IX: unverändert
0D8B	3467	SAVE-Routine
0E5C	3676	LOAD-Routine
0EF9...0FC9		Schlüsselworttabelle; Aufbau: Schlüsselwort im ASCII-Code (Bit7=0; bei Funktionen jeweils inkl. öffnender Klammer) gefolgt vom zugehörigen Token (Werte 0BCH...0FEH, Bit7=1)
3C00...3C09		Wie [2], Tafel 10
3C0C/3C0D		Zufallszahlenpointer
3C2C...3C9F		Konvertierungspuffer Eingabezeile und Stackbereich
3CA0		Beginn Bildwiederholspeicher, darf max. bis 3FFFH gehen

Literatur

[1] rfe 1/85, S.13-18, E. Schiller: Grundvorstellung des BCS3
[2] rfe 9/86, S.559-563, E. Schiller: Erweiterungen für den BCS3 (BASIC-SE 3.1)