Float für ES4.0

Harun Scheutzow hat eine Gleitkommarithmetik als Erweiterung des Tiny-MPBASIC programmiert und in JU+TE 11/1990, Seiten 76-77 vorgestellt.

In der aktuellen rombank zum ES4.0 ist das Paket enthalten.

float_fuer_basic.jtc D000..DFFF

Hinweis: In Original war ein kleiner Bug, der für diverse Probleme sorgt. (Korrektur Adr. D8ECh: 8B 38 → 8B 3D)

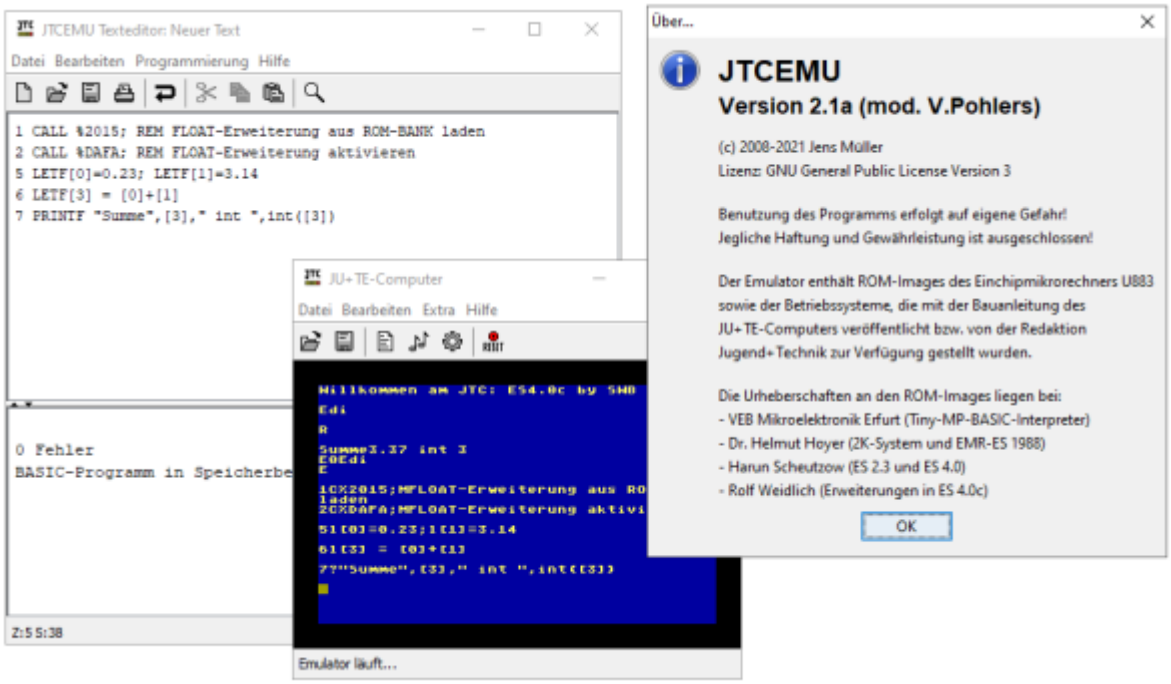
Download

- float_es40.zip

Float-Erweiterung (orig, gepatcht, reass. Quellcode)

Nutzung

Hinweis: Der orginale **JTCEMU** 2.1 unterstützt die Float-Kommandos noch nicht. Ich stelle hier eine nichtoffizielle modifizierte Version bereit, in der die Float-Befehle genutzt werden.




```
1 CALL %2015; REM FLOAT-Erweiterung aus ROM-BANK laden
2 CALL %DAFA; REM FLOAT-Erweiterung aktivieren
```

ab jetzt stehen 3 neue BASIC-Befehle zur Verfügung

Beispiel

```
3 LET A=12,B=5
5 LETF[B]=0.23*A,[0]=3.14
6 LETF[3]=[B]+10.2*[0]
7 PRINTF "Summe ",[3]+100," int ",int([3])
8 LETF C=int([3])
9 PRINT C
```

Eingabe im EDI als

```
1C%2015
2C%DAFA
3LA=12,B=5
5l[B]=0.23*A,[0]=3.14
6l[3]=[B]+10.2*[0]
7?"Summe ",[3]+100," int ",int([3])
...
```

Ausgabe

```
Summe 140.6944 int 40
00040
```

Bemerkung: Zeile 5: Mehrfachausführung eines Kommando (Komma-getrennt)
Zeile 5, 6: Int-Variable als Index, Berechnung mit Int-Variablen und Float-Variablen ist mischbar
Zeile 8: Float-Let-Funktion zur Zuweisung an eine Int-Variable

Befehle

Die kleinste positive Zahl größer als Null ist die 1E-98, die größte positive Zahl ist 9.999999999E99. Für die negativen Zahlen gilt analoges.

[n]

Gleitkomma-Variablen werden mit „[n]“ kodiert.

Die Zahlen werden in Variablen gespeichert. Sie erhalten keinen Namen, sondern eine Nummer. Die Numerierung beginnt mit null und endet mit einer durch die Speichergröße bestimmten Zahl.

Eine Gleitkommavariable wird als [num] geschrieben „num“ ist dabei die Nummer der Variablen und kann aus den Dezimalzahlen 0 bis 65535, den normalen Basic- Variablen A bis Z und den Operationszeichen + (Plus) und - (Minus) beliebig zusammengesetzt sein.

expr ist eine Zusammensetzung aus var, Gleitkommazahlen, den Operationszeichen + - * / und den definierten Funktionen. Dieser Ausdruck wird von **links nach rechts berechnet, ohne irgendwelche Prioritäten** zu beachten. Eine andere Berechnungsreihenfolge kann nur durch das Setzen von runden Klammern erreicht werden.
Die Gleitkommazahlen werden in der üblichen Computernotation eingegeben. Das „E“ als

Exponentenkennzeichen muß groß geschrieben werden. Beispiele: 234, -98.07, 5.6E-12, -34E45

PRINTF „text“expr,„text2“,..

?„text“expr druckt den Text „text“, berechnet dann den Ausdruck expr und druckt das Ergebnis aus. „text“ und expr können in beliebiger Anzahl und Reihenfolge auftreten. Steht nach dieser Folge ein Komma, dann wird nach dem Ausdrucken kein Return-Kode ausgegeben.
interne Kodierung des Befehls als „?“

LETF [n]=expr

lvar=expr berechnet den Ausdruck expr und übergibt den Wert an die Variable var.
interne Kodierung des Befehls als „l“

INPUTF [i]

ivar wartet auf eine Zahleneingabe und trägt die Zahl in die Variable var ein. var steht für eine normale Basic-Variable A bis Z oder eine Gleitkommavariable.
interne Kodierung des Befehls als „i“

Die **Funktionen** müssen aus kleinen Buchstaben bestehen. Das Argument folgt in runden Klammern. Es sind bereits folgende Funktionen definiert:

abs(expr) liefert den absoluten Betrag von expr

sgn(expr) Vorzeichentest, ergibt null, wenn expr null ist und -1 bzw. 1, wenn expr negativ bzw. positiv ist.

int(expr) setzt bei expr alle Nachkommastellen auf null. Trigonometrische und Potenzfunktionen werden später folgen.

Auch in dieser Basic- Erweiterung wird keine umfangreiche Syntaxkontrolle durchgeführt. Folgende Fehler werden trotzdem erkannt und gemeldet:

E0: unbekannter Befehl am Zeilenanfang oder nach einem Semikolon
E3: Syntaxfehler im expr (Fehlende Klammern z. B.)
E12: Zahlenbereichsüberschreitung oder Division durch null

Interna

Die beiden folgenden Abschnitte enthalten Informationen über Programmdetails. Für den Gebrauch der Basic- Erweiterung ist deren Verständnis nicht erforderlich.

Die kleinste positive Zahl größer als Null ist die 1E-98, die größte positive Zahl ist 9.999999999E99. Für die negativen Zahlen gilt analoges. Die Zahlen werden in Variablen gespeichert. Um Platz zu sparen, erhalten sie keinen Namen, sondern eine Nummer. Die Numerierung beginnt mit null und endet mit einer durch die Speichergröße bestimmten Zahl.
Die Anfangsadresse des Variablenspeichers wird auf %D6DC (Lowbyte) und %D6DF (Highbyte) eingetragen. Bei einem Eintrag auf %00 und %C0 beginnt der Variablenspeicher bei %C000[1)]. Bis zum Programmbeginn sind noch 4096 byte frei, die für 682 Variablen zu 6 byte ausreichen. Die höchste nutzbare Variablennummer lautet somit 681.

Der Maschinenprogramm-Aufruf C%DAFA am Anfang eines Basic-Programms startet eine neue Interpretationsroutine. Das Ausführen der bereits bekannten Befehle wird dadurch nicht beeinflußt.

Zahlenformat

Zahlen werden BCD-kodiert in 6 Byte abgelegt:

Intern werden Dezimalzahlen mit einer Genauigkeit von neun bis zehn Stellen benutzt. Das erste Byte enthält im Bit 7 die Vorzeicheninformation: Ist es 1, dann ist die Zahl negativ. Bit 6 bis 0 geben den Exponenten zur Basis 100 (einhundert!) an. Dieser wurde noch zu %40 addiert. Zweites bis sechstes Byte enthalten die Ziffern im gepackten BCD-Kode, wobei das zweite Byte das höchstwertige ist. Nach ihm steht der Dezimalpunkt. Nur die Zahl 0 (Null) hat eine besondere Darstellung: Alle sechs Byte sind %00. Dieses Zahlenformat wird auch im ATARI XL genutzt und vermeidet Rundungsfehler, die sonst beim Umwandeln von Dezimalzahlen in ein Binärformat auftreten.

Rechenroutinen

Die Rechenroutinen nutzen den Registersatz %7X. Die Register %70-%75 werden als FP0 und die Register %76-%7B als FP1 bezeichnet. Ein Zahlenbereichsüberschreiten melden die Routinen FADD, FSUB, FMUL und FDIV mit gesetztem Carry- Flag. Alle Routinen verändern %7X.

```
%D000 FSUB FP0=FP0-FP1
%D003 FADD FP0=FP0+FP1
%D0EE FASC wandelt Zahl aus FP0 in einen ASCII-String um und legt diesen ab
           (%76/77) ab, als letztes Zeichen steht ein Byte %00;
%D1F8 PASC druckt die Zahl aus FP0 aus, benutzt dazu den Puffer %F750-F76F;
%D234 ASCF wandelt den ASCII-String ab (%7C/7D) in eine Zahl in FP0,
(%7C/7D)
           zeigt danach auf das erste nicht umwandelbare Zeichen;
%D424 FDIV FP0=FP0/FP1, wenn FP1 null ist, wird ebenfalls das C-Flag
gesetzt,
           Registersatz %4X wird auf Stack zwischengespeichert;
%D540 FMUL FP0=FP0 FP1, Registersätze %2X und %4X werden auf Stack
           zwischengespeichert;
%D578 HEXF wandelt vorzeichenbehaftete Integerzahl aus %76/77 nach FP0;
%D5E0 FHEX wandelt die Zahl aus FP0 in vorzeichenbehaftete Integerzahl nach
           %76/77.
```

[1)]

Im verbreiteten Paket steht der Zeiger auf 8000h