

Basic-Erweiterung

Das [Tiny MPBASIC](#) des U883 lässt sich um neue Prozeduren und Funktionen [erweitern](#).
Erstaunlicherweise sind in der Literatur jedoch keine Beispiele zu finden. 2021 habe ich mich mit dieser Thematik beschäftigt und eine erste Erweiterung für das ES4.0 geschrieben:

Download

- [baserw.zip](#)

Erweiterungen für die TINY-Systeme

ES 4.0

Die Erweiterung liegt als Binärdatei vor und muss vorab in den Speicher geladen werden (nach 8000h)

Mit

```
1 PROC SETRR[8,%8000];REM Erweiterung einbinden
```

werden die neuen Prozeduren dem BASIC bekannt gemacht. Danach können die neuen Befehle genutzt werden.¹⁾

Hinweis: in der [ROM-Bank](#) ist die Basic-Erweiterung ab Adresse **3000h** dauerhaft verfügbar. Es ist hierzu SETRR[8,%3000] zu nutzen.

Durch die neuen Befehle werden die Programme auch leichter lesbar (z.B. DRAW statt CALL%, CLS, AT, ...)

Grafik-Befehle

| | | |
|-------------------------------|----------------------------|------------------|
| PROC PLOT[X,Y,farbe] | Punkt setzen | statt LET |
| X=,Y=,Z=; CALL %17FD | | |
| F=PTEST[X,Y] | Punktfarbe abfragen | statt LET X=,Y=; |
| CALL %17FA; PRINT Z | | |
| PROC DRAW[X1,Y1,X2,Y2,farbe] | Linie von X1,Y1 nach X2,Y2 | statt LET |
| V=,W=,X=,Y=,Z=; CALL %17F7 | | |
| PROC CIRCLE[X,Y,radius,Farbe] | Kreis | |

Achtung: es werden die Variablen X,Y,Z und V,W (draw) mitgenutzt!
s. ES4.0-Beschreibung. Diese dürfen im eigenen Programm daher nicht genutzt werden!

Text-Befehle

```

PROC CLS                Bildschirm löschen          statt OPTC[12]
PROC COLOR[FG,HG]      Textfarben setzen          statt OSETEB[%F7A0,%F0]
PROC AT[X,Y]           Textcursor positionieren  statt LET X=,Y=,Z=;C%0827
                                                                bzw.
OSETRR[%5B,..];OSETR[%5B,..]

```

```

10 PROC CLS
20 PROC COLOR[15,0]
30 PROC AT[13,19];PRINT"JU+TE-COMPUTER"
DATA-Befehle
-----

```

Anstelle Schlüsselwort DATA wird REM genutzt.
 Die Daten sind in REM-Zeilen abzulegen, ohne Leerzeichen.
 10 REM 11,22,33

RESTORE[10] setzt den Daten-Zeiger auf Zeile 10
 der Zeiger wird nicht automatisch auf die nächste Zeile gesetzt,
 es muss wieder RESTORE erfolgen
 Y=READ Funktion, liest den nächsten Datenwert aus der Zeile, auf die
 RESTORE gesetzt ist und setzt den Daten-Zeiger auf den folgenden Wert.

Beispiele

Beispiel Punkte setzen/abfragen

```

1 PROC SETRR[8,%8000]
10 PROC CLS
20 PROC PLOT[200,100,15]
30 PROC PLOT[100,100,8]
40 PRINT PTEST[100,100]

```

Farb- und Programmbeispiel JU+TE 07/1990, Seiten 84

```

10 PROC PTC[12]
20 LET A=0,B=0,C=319,D=191,E=6
30 LET V=A,W=B,X=C,Y=B,Z=E/3; CALL %17F7
40 LET V=C,W=D; CALL %17F7
50 LET X=A,Y=D; CALL %17F7
60 LET V=A,W=B; CALL %17F7
70 LET A=A+2,B=B+1,C=C-2,D=D-1,E=E+1
80 IF A<160 THEN GOTO 30
90 END

```

neu (besser lesbar)

```

1 PROC SETRR[8,%8000]

```

```

10 PROC CLS
20 LET A=0,B=0,C=319,D=191,E=6
30 LET F=E/3
35 PROC DRAW[A,B,C,B,F]
40 PROC DRAW[C,D,C,B,F]
50 PROC DRAW[C,D,A,D,F]
60 PROC DRAW[A,B,A,D,F]
70 LET A=A+2,B=B+1,C=C-2,D=D-1,E=E+1
80 IF A<160 THEN GOTO 30
90 END

```

Beispiel Testbild (analog zu R. Weidlich's MC-Programm)

```

1 PROC SETRR[8,%8000]
10 PROC COLOR[2,13];PROC CLS
15 REM GITTERNETZ
20 LET I=1
30 PROC DRAW[16*I,0,16*I,191,15]
40 LET I=I+1; IF I<=19 THEN GOTO 30
50 LET I=1
60 PROC DRAW[0,16*I,319,16*I,15]
70 LET I=I+1; IF I<=15 THEN GOTO 60
100 REM KREISE
110 PROC CIRCLE[160,96,85,15]
120 PROC CIRCLE[24,24,16,15]
130 PROC CIRCLE[296,24,16,15]
140 PROC CIRCLE[24,168,16,15]
150 PROC CIRCLE[296,168,16,15]
200 REM Farbfelder
210 LET C=0,A=12,B=5,I=0
215 PROC COLOR[0,C]
220 PROC AT[A, B];PRINT" ",
230 PROC AT[A, B+1];PRINT" ",
240 LET C=C+15,A=A+2,I=I+1
250 IF I=8 THEN LET B=7,A=12
260 IF I<16 THEN GOTO 215
300 REM Zeichensatz
310 PROC COLOR[15,0]
330 LET C=0
340 PROC AT[C $MOD 16 + 12, C/16 + 10]
345 IF C<32 THEN PROC PTC[14];REM VORAB ESC
350 PROC PTC[C]
360 C=C+1;IF C<128 THEN GOTO 340
400 REM TEXT
410 PROC COLOR[0,15]
420 PROC AT[13,19];PRINT"JU+TE-COMPUTER"
430 PROC AT[18,20];PRINT"*6K*"
9999 PROC AT[0,0];PROC COLOR[2,13];END

```

Beispiel Data

Datas in REM-Zeilen ablegen, ohne Leerzeichen

```

1 PROC SETRR[8,%8000]
5 PROC RESTORE[10]
10 REM 11,22,33
20 PRINT READ,READ
30 LET Y=READ;PRINT Y

```

```

R
00011 00022
00033

```



ES 2k

Die Erweiterung **baserw2k** liegt als Binärdatei vor und muss vorab in den Speicher geladen werden (nach 8000h)

Mit

```
1 PROC SETRR[8,%8000];REM Erweiterung einbinden
```

werden die neuen Prozeduren dem BASIC bekannt gemacht. Danach können die neuen Befehle genutzt werden.

Durch die neuen Befehle werden die Programme leichter lesbar

```

PSET[X,Y]   Punkt zeichnen
PRES[X,Y]   Punkt löschen
PTEST[X,Y]  Punkt abfragen
T=INKEY     aktuell gedrückte Taste
CLS         Bildschirm löschen
AT[X,Y]     Cursor positionieren
RESTORE[Zeile] Datenzeile festlegen
Y=READ      Daten aus REM-Zeilen lesen

```

Übersetzung vorhandenen Codes

```

-----
LET X=.,Y=.;CALL %FCA0          PROC PSET[X,Y]
LET X=.,Y=.;CALL %FCB0          PROC PRES[X,Y]
LET X=.,Y=.;CALL %FCBB; IF Z..  LET Z=PTEST[X,Y]

CALL %C56;LET A=GETR[%6D]$A%7F;PROC SETR[%6D,0]    LET A=INKEY

CALL %8DD                          PROC CLS
PROC SETR[%5B,%20]                  PROC AT[2,0]

```

Beispiele

Test Grafik

```

1 PROC SETRR[8,%8000]
10 PROC CLS
20 PROC PSET[10,10]
30 PROC PSET[0,0]
40 PROC PSET[63,63]
100 PRINT PTEST[10,10]
110 PROC PRES[10,10]
120 PRINT PTEST[10,10]
9999 END

```

Test INKEY

```

1 PROC SETRR[8,%8000]
10 PROC CLS
20 X=INKEY;PRINT X
30 GOTO20
9999 END

```

Test AT

```

1 PROC SETRR[8,%8000]
10 PROC CLS
20 PROC AT[0,0];PRINT"00",
30 PROC AT[1,1];PRINT"11",
40 PROC AT[2,4];PRINT"24",
9999 END

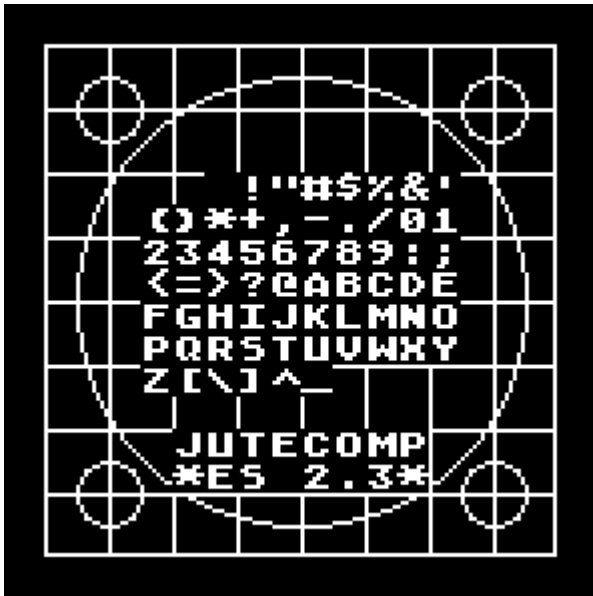
```

```

+
00
 11
   24
END 9999

```

Test Data s.o.



1)
Im Emulator JTCEMU ist die Nutzung der neuen Befehle im Texteditor erst ab Version 2.1 möglich.

From:
<https://hc-ddr.hucki.net/wiki/> - Homecomputer DDR

Permanent link:
<https://hc-ddr.hucki.net/wiki/doku.php/tiny/software/baserw40?rev=1659595258>

Last update: 2022/08/04 06:40

