# AVR-ChipBasic2 - Bibliotheken

## V1.36 (c) 2006-2011 Jörg Wolfram

# 1 Die Festkomma-Bibliothek FIXLIB (Bibliothekscode 0x10)

## 1.1 Allgemeines

Diese Bibliothek erweitert den BASIC-Computer um Festkomma-Berechnungen. Festkomma bedeutet, daß die Stellen vor und nach dem Komma fest vorgegeben sind. Bei dieser Bibliothek lässt sich die Stellenzahl per Initialisierungskommando einstellen. Intern ist eine Darstellung gewählt, die jeweils 2 Stellen (Wertebereich 0...99) in einem Byte zusammenfasst. Somit sind Vor- und Nachkommastellen auch nur in 2-er Schritten einstellbar.

- 2...32 Vorkomma- und 0...30 Nachkommastellen wählbar
- Anzahl der verfügbaren Variablen von der Anzahl der Stellen abhängig
- Belegt den Array-Bereich 256...511
- Formatierte Textausgabe
- Eingabe über Text im Array möglich
- Integer- Datenaustaisch mit dem BASIC-System
- 4 Grundrechenarten, Vergleiche und weitere Funktionen

Systembedingt haben nur Parameter die auch angegeben wurden einen definierten Wert. Bei einem **CALL L,2,0** ist nur der Parameter 1 definiert, der für die Funktion notwendige Parameter 2 aber nicht. Somit wird der Variable 0 der Wert zugewiesen, der sich zu diesem Zeitpunkt gerade an dieser Stelle im RAM befindet.

## 1.2 Funktionsübersicht

| **Funktionsnummer** | **Funktion** |
|---|---|
| 0 | Dummy-Funktion, falls die Bibliothek aus dem Hauptmenu aufgerufen wird |
| 1 | Initialisierung |
| 2 | Integer-Zuweisung Variable=Wert |
| 3 | Zuweisung über Textstring im Array |
| 4 | liefert Integerwert der Variablen |
| 5 | Textausgabe der Variablen |
| 6 | Absolutwert einer Variablen |
| 7 | Invers-Wert (0-Variable) |
| 8 | Addition V3 = V1 + V2 |
| 9 | Subtrakion V3 = V1 - V2 |
| 10 | Multiplikation V3 = V1 * V2 |
| 11 | Division V3 = V1 / V2 |
| 12 | Konstantenzuweisung mit vordefinierten Konstanten |
| 13 | Vorkomma-Anteil einer Variablen (INT) |
| 14 | Nachkomma-Anteil einer Variablen (FRAC) |
| 15 | Vergleich zweier Variablen |
| 16 | Variable kopieren |
| 17 | Variable / 2 |
| 18 | Variable * 2 |
| 19 | Script im Arrayspeicher starten |

### 1.2.1 Funktion 0 : Dummy

Dies ist eine Dummy-Funktion, die lediglich 0 zurückliefert falls die Bibliothek aus dem Hauptmenu aufgerufen wird.

| **Parameter** | **Bedeutung** |
|---|---|
| Parameter 1 | — |
| Parameter 2 | — |
| Parameter 3 | — |
| Rückgabewert | 0 |

### 1.2.2 Funktion 1 : Initialisierung

Diese Funktion muss vor allen anderen Funktionen aufgerufen werden, um die notwendigen Initialisierungen vorzunehmen. Der Rückgabewert entspricht der Anzahl der Variablen, die eingerichtet wurden und hängt von der Zahl der Vor- und Nachkommastellen ab.

| **Parameter** | **Bedeutung** |
|---|---|
| Parameter 1 | Anzahl der Vorkomma-Bytes - 1 (0. . . 15) |
| Parameter 2 | Anzahl der Nachkomma-Bytes |
| Parameter 3 | — |
| Rückgabewert | Anzahl der initialisierten Variablen |

### 1.2.3 Funktion 2 : Wertzuweisung von Integerwert

Setzt eine Variable auf einen Integerwert. Die Nachkommastellen werden auf 0 gesetzt

| **Parameter** | **Bedeutung** |
|---|---|
| Parameter 1 | Variablennummer |
| Parameter 2 | Variablenwert |
| Parameter 3 | — |
| Rückgabewert | 0 |

### 1.2.4 Funktion 3 : Wertzuweisung von Textstring

Setzt eine Variable auf einen durch einen Textstring im Array definierten Wert. Als Trennzeichen zwischen Vorkomma- und Nachkommastellen kann sowohl das Komma als auch der Punkt genutzt werden. Beim ersten nicht-numerischen Zeichen im String wird die Konvertierung abgebrochen.

| **Parameter** | **Bedeutung** |
|---|---|
| Parameter 1 | Variablennummer |
| Parameter 2 | Textstart im Array |
| Parameter 3 | — |
| Rückgabewert | 0 |

### 1.2.5 Funktion 4 : Integerwert der Variablen

Gibt den Integerwert der Variablen aus. Dabei wird entsprechend der Nachkommastellen gerundet.

| **Parameter** | **Bedeutung** |
|---|---|
| Parameter 1 | Variablennummer |
| Parameter 2 | — |
| Parameter 3 | — |
| Rückgabewert | Integer-Wert der Variablen |

### 1.2.6 Funktion 5 : Textausgabe der Variablen

Gibt den Inhalt der Variablen als Textstring aus. Mit der Angabe von minimalen Vorkommabytes kann eine rechtsbündige Darstellung bewirkt werden. Ist die Anzahl der maximalen Nachkommabytes kleiner als die der Variablen, wird entsprechend gerundet.

| **Parameter** | **Bedeutung** |
|---|---|
| Parameter 1 | Variablennummer |
| Parameter 2 | Bit 7...4: minimale Vorkommabytes -1 Bit 3...0: Nachkommabytes |
| Parameter 3 | (optional Ausgabekanal) |
| Rückgabewert | 0 |

### 1.2.7 Funktion 6 : Absolutwert einer Variablen

Der Absolutwert der Variable 1 wird in die Variable 2 gespeichert. V2 = ABS(V1)

| **Parameter** | **Bedeutung** |
|---|---|
| Parameter 1 | Variablennummer von Variable 1 |
| Parameter 2 | Variablennummer von Variable 2 |
| Parameter 3 | — |
| Rückgabewert | 0 |

### 1.2.8 Funktion 7 : Invers-Wert einer Variablen

Der Invers-Wert von Variable 1 wird in die Variable 2 gespeichert. V2 = -V1

| **Parameter** | **Bedeutung** |
|---|---|
| Parameter 1 | Variablennummer von Variable 1 |
| Parameter 2 | Variablennummer von Variable 2 |
| Parameter 3 | — |
| Rückgabewert | 0 |

### 1.2.9 Funktion 8 : Addition

Die Summe von Variable 1 und Variable 2 wird in die Variable 3 gespeichert. V3 = V1 + V2

| **Parameter** | **Bedeutung** |
|---|---|
| Parameter 1 | Variablennummer von Variable 1 |
| Parameter 2 | Variablennummer von Variable 2 |
| Parameter 3 | Variablennummer von Variable 3 |
| Rückgabewert | 0 |

### 1.2.10 Funktion 9 : Subtraktion

Die Differenz von Variable 1 und Variable 2 wird in die Variable 3 gespeichert. V3 = V1 - V2

| **Parameter** | **Bedeutung** |
|---|---|
| Parameter 1 | Variablennummer von Variable 1 |
| Parameter 2 | Variablennummer von Variable 2 |
| Parameter 3 | Variablennummer von Variable 3 |
| Rückgabewert | 0 |

### 1.2.11 Funktion 10 : Multiplikation

Das Produkt von Variable 1 und Variable 2 wird in die Variable 3 gespeichert. V3 = V1 * V2

| **Parameter** | **Bedeutung** |
|---|---|
| Parameter 1 | Variablennummer von Variable 1 |
| Parameter 2 | Variablennummer von Variable 2 |
| Parameter 3 | Variablennummer von Variable 3 |
| Rückgabewert | 0 |

### 1.2.12 Funktion 11 : Division

Der Quotient von Variable 1 und Variable 2 wird in die Variable 3 gespeichert. V3 = V1 / V2

| **Parameter** | **Bedeutung** |
|---|---|
| Parameter 1 | Variablennummer von Variable 1 |
| Parameter 2 | Variablennummer von Variable 2 |
| Parameter 3 | Variablennummer von Variable 3 |
| Rückgabewert | 0 |

### 1.2.13 Funktion 12 : Konstantenzuweisung

Für 8 vordefinierte Konstanten lässt sich die Wertzuweisung über diese Funktion vereinfachen.

| **Parameter** | **Bedeutung** |
|---|---|
| Parameter 1 | Variablennummer |
| Parameter 2 | Konstantennummer (0. . . 7) |
| Parameter 3 | — |
| Rückgabewert | 0 |

Die 8 Konstanten haben 15 gültige Nachkommastellen und sind wie folgt definiert:

| **Konstantennummer** | **Wert** |
|---|---|
| 0 | 0 |
| 1 | 1 |
| 2 | PI |
| 3 | e (Eulersche Zahl) |
| 4 | ln(2) |
| 5 | ln(10) |
| 6 | SQRT(2) |
| 7 | SQRT(3) |

### 1.2.14 Funktion 13 : Vorkomma-Anteil einer Variablen

Der Vorkomma-Anteil der Variable 1 wird in die Variable 2 gespeichert. Genaugenommen werden nur die Nachkomma-Bytes auf 0 gesetzt V2 = INT(V1)

| **Parameter** | **Bedeutung** |
|---|---|
| Parameter 1 | Variablennummer von Variable 1 |
| Parameter 2 | Variablennummer von Variable 2 |
| Parameter 3 | — |
| Rückgabewert | 0 |

### 1.2.15 Funktion 14 : Nachkomma-Anteil einer Variablen

Der Vorkomma-Anteil der Variable 1 wird in die Variable 2 gespeichert. Genaugenommen werden nur die Vorkomma-Bytes auf 0 gesetzt V2 = FRAC(V1)

| **Parameter** | **Bedeutung** |
|---|---|
| Parameter 1 | Variablennummer von Variable 1 |
| Parameter 2 | Variablennummer von Variable 2 |
| Parameter 3 | — |
| Rückgabewert | 0 |

### 1.2.16 Funktion 15 : Vergleich zweier Variablen

Variable 1 wird mit Variable 2 verglichen. Ist Variable 1 größer als Variable 2, dann ist der Rückgabewert 1, ist Variable 1 kleiner als Variable 2, ist das Ergebnis 2 und bei Gleichheit beider Variablen wird 0 zurückgeliefert.

| **Parameter** | **Bedeutung** |
|---|---|
| Parameter 1 | Variablennummer von Variable 1 |
| Parameter 2 | Variablennummer von Variable 2 |
| Parameter 3 | — |
| Rückgabewert | Vergleichsergebnis (1,0,2) |

### 1.2.17 Funktion 16 : Variable kopieren

Variable 1 wird in die Variable 2 kopiert.

| **Parameter** | **Bedeutung** |
|---|---|
| Parameter 1 | Variablennummer von Variable 1 |
| Parameter 2 | Variablennummer von Variable 2 |
| Parameter 3 | — |
| Rückgabewert | 0 |

### 1.2.18 Funktion 17 : Multiplikation mit 2

Variable 1 wird durch Bitschieben mit 2 multipliziert und das Ergebnis in die Variable 2 gespeichert. Diese Funktion ist wesentlich schneller als eine entsprechende Multiplikation. V2 = V1 * 2

| **Parameter** | **Bedeutung** |
|---|---|
| Parameter 1 | Variablennummer von Variable 1 |
| Parameter 2 | Variablennummer von Variable 2 |
| Parameter 3 | — |
| Rückgabewert | 0 |

### 1.2.19 Funktion 18 : Division durch 2

Variable 1 wird durch Bitschieben durch 2 dividiert und das Ergebnis in die Variable 2 gespeichert. Diese Funktion ist wesentlich schneller als eine entsprechende Division. V2 = V1 / 2

| **Parameter** | **Bedeutung** |
|---|---|
| Parameter 1 | Variablennummer von Variable 1 |
| Parameter 2 | Variablennummer von Variable 2 |
| Parameter 3 | — |
| Rückgabewert | 0 |

### 1.2.20 Funktion 19 : Starten eines Scripts im Array

Da die Bibliothek im Wesentlichen nur die Grundrechenarten bereitstellt, gibt es eine kleine Script-Engine mit der auch komplexere Funktionen relativ einfach berechnet werden können. Dazu muss der Code im ersten Drittel des Arrays (Zellen 0…255) stehen.

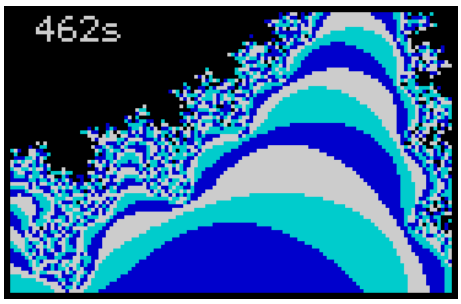
| **Parameter** | **Bedeutung** |
|---|---|
| Parameter 1 | Startadresse im Array |
| Parameter 2 | opt. Parameter 1 |
| Parameter 3 | opt. Parameter 2 |
| Parameter 4 | opt. Parameter 3 |
| Rückgabewert | Rückgabewert, je nach Exit-Befehl |

Für Schleifen etc. stehen vier 8-Bit Countervariablen zur Verfügung Jeder Befehl besteht aus 2 Bytes, die noch in Halbbytes (Nibbles) unterteilt sind. Ein x steht dafür, dass der Wert des entsprechenden Nibbles nicht ausgewertet wird. Trotzdem ist es sinnvoll diese Nibbles auf 0 zu setzen. Für Schleifen etc. stehen vier 8-Bit Countervariablen zur Verfügung, beim Start eines Scripts werden die ersten drei mit den LOW-Bytes der Parameter 1…3 vorbelegt, Countervariable 4 wird auf 0 gesetzt. Systembedingt haben nur Parameter die angegeben wurden einen definierten Wert. Bei einem **CALL 7,19,17** ist nur der Parameter 1 auf 17 gesetzt, Parameter 2 und 3 sind undefiniert.

| **Code** | **Bedeutung** |
|---|---|
| 0 0 0 V | Script beenden, Variablenwert (Integer) von Variable v wird zurückgegeben |
| 0 1 x M | Script beenden, der Wert von Counter M wird zurückgegeben |
| 0 2 n n | E wird zur Script-Position nn gesprungen |
| 0 3 C V | Variable V wird mit der Konstanten C initialisiert |
| 0 4 x V | Variable V wird mit dem Wert des optionalen Parameters 1 initialisiert |
| 0 5 x V | Variable V wird mit dem Wert des optionalen Parameters 2 initialisiert |
| 0 6 x V | Variable V wird mit dem Wert des optionalen Parameters 3 initialisiert |
| 1 0 W V | W = V |
| 1 1 W V | W = ABS(V) |
| 1 2 W V | W = -V |
| 1 3 W V | W = INT(V) |
| 1 4 W V | W = FRAC(V) |
| 1 5 W V | W = V * 2 |
| 1 6 W V | W = V / 2 |
| 1 8 W V | überspringt die nächste Anweisung, wenn V gleich W ist |
| 1 9 W V | überspringt die nächste Anweisung, wenn V ungleich W ist |
| 1 A W V | überspringt die nächste Anweisung, wenn V größer W ist |
| 1 B W V | überspringt die nächste Anweisung, wenn V kleiner W ist |
| 4 U W V | W = U + V |
| 5 U W V | W = U - V |
| 6 U W V | W = U * V |
| 7 U W V | W = U / V |
| 8 M n n | der Counter M wird auf nn gesetzt |
| 9 M n n | zum Counter M wird nn addiert |
| A M n n | überspringt die nächste Anweisung, wenn der Counter M gleich nn ist |
| B M n n | überspringt die nächste Anweisung, wenn der Counter M ungleich nn ist |
| C M x n | der Counter M wird mit Parameter N initialisiert |
| andere Codes | Keine Funktion (NOP) |

## 1.3 Ein Fraktalprogramm als Beispiel

In den Zeilen 7 bis 10 werden die Koordinaten für X1, X2, Y1 und Y2 eingetragen.




```
PROGRAM :Fractal 2
01 LFIND L,$1001:A=120:B=76
02 IF L=0 ? "keine Matlib":END
03 A=120:B=76:C=4:VM 2:VID 0
04 Z=80:PAL 0,0,1,5,7
05 CALL L,1,1,3:CALL L,2,16,4
06 'Koordinatenvorgaben
07 DA 0,"-0.95#":CALL L,3,0,0
08 DA 0,"-0.72#":CALL L,3,1,0
09 DA 0,"0.17#":CALL L,3,2,0
10 DA 0,"0.33#":CALL L,3,3,0
11
12 'DX(V4) und DY(V5) berechnen
13 CALL L,2,6,A
14 CALL L,9,1,0,7:CALL L,11,7,6,4
15 CALL L,2,6,B
16 CALL L,9,3,2,7:CALL L,11,7,6,5
17 CALL L,16,0,15
18 TSET 0
19 FOR Y=0 TO B-1:FOR X=0 TO A-1
20 CALL L,12,8,0:CALL L,12,9,0
21 I=0
22 CALL L,10,8,8,10:'X^2
23 CALL L,10,9,9,11:'Y^2
24 CALL L,8,10,11,6:CALL L,15,6,16
25 R=~R:IF R=1 GOTO 34
26 CALL L,9,10,11,13:'XT
27 CALL L,8,0,13,13
28 CALL L,10,8,9,14:'YT
29 CALL L,17,14,14:CALL L,8,2,14,14
30 CALL L,16,13,8:CALL L,16,14,9
31 'Iteration zu ende
32 I=I+1:IF I<Z GOTO 22
33 PLOT Y,X,0:GOTO 35
34 D=I%(C-1):PLOT Y,X,D+1
35 IF KEY(0)=$F5 THEN VID 0
36 IF KEY(0)=$F6 THEN VID 1
37 CALL L,8,0,4,0:NEXT
38 CALL L,8,2,5,2
39 CALL L,16,15,0:NEXT
40 TGET S
41 ? @0,0;(S-T)/10;"s ":VID 1
42 GOTO 41
#
```

# 2 Eine RTC Befehlserweiterung (Bibliothekscode 0x70)

## 2.1 Allgemeines

Diese Bibliothek erweitert den BASIC-Computer um Festkomma-Berechnungen. Festkomma bedeutet, daß die Stellen vor und nach dem Komma fest vorgegeben sind. Bei dieser Bibliothek lässt sich die Stellenzahl per Initialisierungskommando einstellen. Intern ist eine Darstellung gewählt, die jeweils 2 Stellen (Wertebereich 0...99) in einem Byte zusammenfasst. Somit sind Vor- und Nachkommastellen auch nur in 2-er Schritten einstellbar.

- 2...32 Vorkomma- und 0...30 Nachkommastellen wählbar
- Anzahl der verfügbaren Variablen von der Anzahl der Stellen abhängig
- Belegt den Array-Bereich 256...511
- Formatierte Textausgabe
- Eingabe über Text im Array möglich
- Integer- Datenaustaisch mit dem BASIC-System
- 4 Grundrechenarten, Vergleiche und weitere Funktionen

Systembedingt haben nur Parameter die auch angegeben wurden einen definierten Wert. Bei einem **CALL L,2,0** ist nur der Parameter 1 definiert, der für die Funktion notwendige Parameter 2 aber nicht. Somit wird der Variable 0 der Wert zugewiesen, der sich zu diesem Zeitpunkt gerade an dieser Stelle im RAM befindet.

## 2.2 Funktionsübersicht

| **Funktionsnummer** | **Funktion** |
|---|---|
| 0 | Dummy-Funktion, falls die Bibliothek aus dem Hauptmenu aufgerufen wird |
| 1 | Initialisierung |
| 2 | Integer-Zuweisung Variable=Wert |
| 3 | Zuweisung über Textstring im Array |
| 4 | liefert Integerwert der Variablen |
| 5 | Textausgabe der Variablen |
| 6 | Absolutwert einer Variablen |
| 7 | Invers-Wert (0-Variable) |
| 8 | Addition V3 = V1 + V2 |
| 9 | Subtrakion V3 = V1 - V2 |
| 10 | Multiplikation V3 = V1 * V2 |
| 11 | Division V3 = V1 / V2 |
| 12 | Konstantenzuweisung mit vordefinierten Konstanten |
| 13 | Vorkomma-Anteil einer Variablen (INT) |
| 14 | Nachkomma-Anteil einer Variablen (FRAC) |
| 15 | Vergleich zweier Variablen |
| 16 | Variable kopieren |
| 17 | Variable / 2 |
| 18 | Variable * 2 |
| 19 | Script im Arrayspeicher starten |

### 2.2.1 Funktion 0 : Dummy

Dies ist eine Dummy-Funktion, die lediglich 0 zurückliefert falls die Bibliothek aus dem Hauptmenu aufgerufen wird.

| **Parameter** | **Bedeutung** |
|---|---|
| Parameter 1 | — |
| Parameter 2 | — |
| Parameter 3 | — |
| Rückgabewert | 0 |

### 2.2.2 Funktion 1 : Initialisierung

Diese Funktion muss vor allen anderen Funktionen aufgerufen werden, um die notwendigen Initialisierungen vorzunehmen. Der Rückgabewert entspricht der Anzahl der Variablen, die eingerichtet wurden und hängt von der Zahl der Vor- und Nachkommastellen ab.

| **Parameter** | **Bedeutung** |
|---|---|
| Parameter 1 | Anzahl der Vorkomma-Bytes - 1 (0. . . 15) |
| Parameter 2 | Anzahl der Nachkomma-Bytes |
| Parameter 3 | — |
| Rückgabewert | Anzahl der initialisierten Variablen |

### 2.2.3 Funktion 2 : Wertzuweisung von Integerwert

Setzt eine Variable auf einen Integerwert. Die Nachkommastellen werden auf 0 gesetzt

| **Parameter** | **Bedeutung** |
|---|---|
| Parameter 1 | Variablennummer |
| Parameter 2 | Variablenwert |
| Parameter 3 | — |
| Rückgabewert | 0 |

### 2.2.4 Funktion 3 : Wertzuweisung von Textstring

Setzt eine Variable auf einen durch einen Textstring im Array definierten Wert. Als Trennzeichen zwischen Vorkomma- und Nachkommastellen kann sowohl das Komma als auch der Punkt genutzt werden. Beim ersten nicht-numerischen Zeichen im String wird die Konvertierung abgebrochen.

| **Parameter** | **Bedeutung** |
|---|---|
| Parameter 1 | Variablennummer |
| Parameter 2 | Textstart im Array |
| Parameter 3 | — |
| Rückgabewert | 0 |

### 2.2.5 Funktion 4 : Integerwert der Variablen

Gibt den Integerwert der Variablen aus. Dabei wird entsprechend der Nachkommastellen gerundet.

| **Parameter** | **Bedeutung** |
|---|---|
| Parameter 1 | Variablennummer |
| Parameter 2 | — |
| Parameter 3 | — |
| Rückgabewert | Integer-Wert der Variablen |

### 2.2.6 Funktion 5 : Textausgabe der Variablen

Gibt den Inhalt der Variablen als Textstring aus. Mit der Angabe von minimalen Vorkommabytes kann eine rechtsbündige Darstellung bewirkt werden. Ist die Anzahl der maximalen Nachkommabytes kleiner als die der Variablen, wird entsprechend gerundet.

| **Parameter** | **Bedeutung** |
|---|---|
| Parameter 1 | Variablennummer |
| Parameter 2 | Bit 7...4: minimale Vorkommabytes -1 Bit 3...0: Nachkommabytes |
| Parameter 3 | (optional Ausgabekanal) |
| Rückgabewert | 0 |

### 2.2.7 Funktion 6 : Absolutwert einer Variablen

Der Absolutwert der Variable 1 wird in die Variable 2 gespeichert. V2 = ABS(V1)

| **Parameter** | **Bedeutung** |
|---|---|
| Parameter 1 | Variablennummer von Variable 1 |
| Parameter 2 | Variablennummer von Variable 2 |
| Parameter 3 | — |
| Rückgabewert | 0 |

### 2.2.8 Funktion 7 : Invers-Wert einer Variablen

Der Invers-Wert von Variable 1 wird in die Variable 2 gespeichert. V2 = -V1

| **Parameter** | **Bedeutung** |
|---|---|
| Parameter 1 | Variablennummer von Variable 1 |
| Parameter 2 | Variablennummer von Variable 2 |
| Parameter 3 | — |
| Rückgabewert | 0 |

### 2.2.9 Funktion 8 : Addition

Die Summe von Variable 1 und Variable 2 wird in die Variable 3 gespeichert. V3 = V1 + V2

| **Parameter** | **Bedeutung** |
|---|---|
| Parameter 1 | Variablennummer von Variable 1 |
| Parameter 2 | Variablennummer von Variable 2 |
| Parameter 3 | Variablennummer von Variable 3 |
| Rückgabewert | 0 |

### 2.2.10 Funktion 9 : Subtraktion

Die Differenz von Variable 1 und Variable 2 wird in die Variable 3 gespeichert. V3 = V1 - V2

| **Parameter** | **Bedeutung** |
|---|---|
| Parameter 1 | Variablennummer von Variable 1 |
| Parameter 2 | Variablennummer von Variable 2 |
| Parameter 3 | Variablennummer von Variable 3 |
| Rückgabewert | 0 |

#### 2.2.11 Funktion 10 : Multiplikation

Das Produkt von Variable 1 und Variable 2 wird in die Variable 3 gespeichert. V3 = V1 * V2

| **Parameter** | **Bedeutung** |
|---|---|
| Parameter 1 | Variablennummer von Variable 1 |
| Parameter 2 | Variablennummer von Variable 2 |
| Parameter 3 | Variablennummer von Variable 3 |
| Rückgabewert | 0 |

#### 2.2.12 Funktion 11 : Division

Der Quotient von Variable 1 und Variable 2 wird in die Variable 3 gespeichert. V3 = V1 / V2

| **Parameter** | **Bedeutung** |
|---|---|
| Parameter 1 | Variablennummer von Variable 1 |
| Parameter 2 | Variablennummer von Variable 2 |
| Parameter 3 | Variablennummer von Variable 3 |
| Rückgabewert | 0 |

#### 2.2.13 Funktion 12 : Konstantenzuweisung

Für 8 vordefinierte Konstanten lässt sich die Wertzuweisung über diese Funktion vereinfachen.

| **Parameter** | **Bedeutung** |
|---|---|
| Parameter 1 | Variablennummer |
| Parameter 2 | Konstantennummer (0. . . 7) |
| Parameter 3 | — |
| Rückgabewert | 0 |

Die 8 Konstanten haben 15 gültige Nachkommastellen und sind wie folgt definiert:

| **Konstantennummer** | **Wert** |
|---|---|
| 0 | 0 |
| 1 | 1 |
| 2 | PI |
| 3 | e (Eulersche Zahl) |
| 4 | ln(2) |
| 5 | ln(10) |
| 6 | SQRT(2) |
| 7 | SQRT(3) |

#### 2.2.14 Funktion 13 : Vorkomma-Anteil einer Variablen

Der Vorkomma-Anteil der Variable 1 wird in die Variable 2 gespeichert. Genaugenommen werden nur die Nachkomma-Bytes auf 0 gesetzt V2 = INT(V1)

| **Parameter** | **Bedeutung** |
|---|---|
| Parameter 1 | Variablennummer von Variable 1 |
| Parameter 2 | Variablennummer von Variable 2 |
| Parameter 3 | — |
| Rückgabewert | 0 |

### 2.2.15 Funktion 14 : Nachkomma-Anteil einer Variablen

Der Vorkomma-Anteil der Variable 1 wird in die Variable 2 gespeichert. Genaugenommen werden nur die Vorkomma-Bytes auf 0 gesetzt V2 = FRAC(V1)

| **Parameter** | **Bedeutung** |
|---|---|
| Parameter 1 | Variablennummer von Variable 1 |
| Parameter 2 | Variablennummer von Variable 2 |
| Parameter 3 | — |
| Rückgabewert | 0 |

### 2.2.16 Funktion 15 : Vergleich zweier Variablen

Variable 1 wird mit Variable 2 verglichen. Ist Variable 1 größer als Variable 2, dann ist der Rückgabewert 1, ist Variable 1 kleiner als Variable 2, ist das Ergebnis 2 und bei Gleichheit beider Variablen wird 0 zurückgeliefert.

| **Parameter** | **Bedeutung** |
|---|---|
| Parameter 1 | Variablennummer von Variable 1 |
| Parameter 2 | Variablennummer von Variable 2 |
| Parameter 3 | — |
| Rückgabewert | Vergleichsergebnis (1,0,2) |

### 2.2.17 Funktion 16 : Variable kopieren

Variable 1 wird in die Variable 2 kopiert.

| **Parameter** | **Bedeutung** |
|---|---|
| Parameter 1 | Variablennummer von Variable 1 |
| Parameter 2 | Variablennummer von Variable 2 |
| Parameter 3 | — |
| Rückgabewert | 0 |

### 2.2.18 Funktion 17 : Multiplikation mit 2

Variable 1 wird durch Bitschieben mit 2 multipliziert und das Ergebnis in die Variable 2 gespeichert. Diese Funktion ist wesentlich schneller als eine entsprechende Multiplikation. V2 = V1 * 2

| **Parameter** | **Bedeutung** |
|---|---|
| Parameter 1 | Variablennummer von Variable 1 |
| Parameter 2 | Variablennummer von Variable 2 |
| Parameter 3 | — |
| Rückgabewert | 0 |

### 2.2.19 Funktion 18 : Division durch 2

Variable 1 wird durch Bitschieben durch 2 dividiert und das Ergebnis in die Variable 2 gespeichert. Diese Funktion ist wesentlich schneller als eine entsprechende Division. V2 = V1 / 2

| **Parameter** | **Bedeutung** |
|---|---|
| Parameter 1 | Variablennummer von Variable 1 |
| Parameter 2 | Variablennummer von Variable 2 |
| Parameter 3 | — |
| Rückgabewert | 0 |

### 2.2.20 Funktion 19 : Starten eines Scripts im Array

Da die Bibliothek im Wesentlichen nur die Grundrechenarten bereitstellt, gibt es eine kleine Script-Engine mit der auch komplexere Funktionen relativ einfach berechnet werden können. Dazu muss der Code im ersten Drittel des Arrays (Zellen 0...255) stehen.

| **Parameter** | **Bedeutung** |
|---|---|
| Parameter 1 | Startadresse im Array |
| Parameter 2 | opt. Parameter 1 |
| Parameter 3 | opt. Parameter 2 |
| Parameter 4 | opt. Parameter 3 |
| Rückgabewert | Rückgabewert, je nach Exit-Befehl |

Für Schleifen etc. stehen vier 8-Bit Countervariablen zur Verfügung Jeder Befehl besteht aus 2 Bytes, die noch in Halbbytes (Nibbles) unterteilt sind. Ein x steht dafür, dass der Wert des entsprechenden Nibbles nicht ausgewertet wird. Trotzdem ist es sinnvoll diese Nibbles auf 0 zu setzen. Für Schleifen etc. stehen vier 8-Bit Countervariablen zur Verfügung, beim Start eines Scripts werden die ersten drei mit den LOW-Bytes der Parameter 1...3 vorbelegt, Countervariable 4 wird auf 0 gesetzt. Systembedingt haben nur Parameter die angegeben wurden einen definierten Wert. Bei einem **CALL 7,19,17** ist nur der Parameter 1 auf 17 gesetzt, Parameter 2 und 3 sind undefiniert.

| **Code** | **Bedeutung** |
|---|---|
| 0 0 0 V | Script beenden, Variablenwert (Integer) von Variable v wird zurückgegeben |
| 0 1 x M | Script beenden, der Wert von Counter M wird zurückgegeben |
| 0 2 n n | E wird zur Script-Position nn gesprungen |
| 0 3 C V | Variable V wird mit der Konstanten C initialisiert |
| 0 4 x V | Variable V wird mit dem Wert des optionalen Parameters 1 initialisiert |
| 0 5 x V | Variable V wird mit dem Wert des optionalen Parameters 2 initialisiert |
| 0 6 x V | Variable V wird mit dem Wert des optionalen Parameters 3 initialisiert |
| 1 0 W V | W = V |
| 1 1 W V | W = ABS(V) |
| 1 2 W V | W = -V |
| 1 3 W V | W = INT(V) |
| 1 4 W V | W = FRAC(V) |
| 1 5 W V | W = V * 2 |
| 1 6 W V | W = V / 2 |
| 1 8 W V | überspringt die nächste Anweisung, wenn V gleich W ist |
| 1 9 W V | überspringt die nächste Anweisung, wenn V ungleich W ist |
| 1 A W V | überspringt die nächste Anweisung, wenn V größer W ist |
| 1 B W V | überspringt die nächste Anweisung, wenn V kleiner W ist |
| 4 U W V | W = U + V |
| 5 U W V | W = U - V |
| 6 U W V | W = U * V |
| 7 U W V | W = U / V |
| 8 M n n | der Counter M wird auf nn gesetzt |
| 9 M n n | zum Counter M wird nn addiert |
| A M n n | überspringt die nächste Anweisung, wenn der Counter M gleich nn ist |
| B M n n | überspringt die nächste Anweisung, wenn der Counter M ungleich nn ist |
| C M x n | der Counter M wird mit Parameter N initialisiert |
| andere Codes | Keine Funktion (NOP) |

## 2.3 Ein Fraktalprogramm als Beispiel

In den Zeilen 7 bis 10 werden die Koordinaten für X1, X2, Y1 und Y2 eingetragen.