RND

*Anmerkung: Dieser Artikel beschreibt den BASIC-Befehl **RND** in BASIC V2 auf dem Commodore 64 .*

BASIC-Schlüsselwort	
Stichwort:	**RND**
Abkürzung :	**R, Umschalt+N**
Typ:	Funktion
Token-Code :	187/$BB
Handhabungsroutine im BASIC ROM :	57495–57592 $E097–E0F8
Liste aller BASIC-Schlüsselwörter	

```
Typ : Funktion
Allgemeine Programmiersyntax : RND(<Zahl>)
```

Die Funktion **RND** erzeugt pseudozufällige [1] Gleitkommazahlen im Bereich von 0,0 (einschließlich) bis 1,0 (ausschließlich). Das Argument <Zahl> kann positiv, negativ oder null sein.

- Durch die Verwendung von **RND(<positive Zahl>)** wird jedes Mal eine andere Zufallszahl aus einer vorgegebenen Sequenz ausgegeben (die Sequenznummer wird intern gespeichert).
- Mit **RND(<negative Zahl>)** springt man zu einem Punkt in der Sequenz, der durch die verwendete negative Zahl bestimmt wird. Wiederholtes Aufrufen von RND mit derselben negativen Zahl führt zum gleichen Ergebnis. Typischerweise wird RND(<negative Zahl>) einmal und anschließend wiederholt RND(<positive Zahl>) aufgerufen.
- Bei **RND(0)** generiert der C64 die Zufallszahl aus dem internen Timer [2] und der Echtzeituhr [3] , allerdings sind die möglichen Werte, solange die RTC nicht läuft, auf nur 16 Bit für die 32 Bit breite Gleitkomma-Mantisse beschränkt.

Bei Verwendung eines ungültigen numerischen Werts wird der BASIC-Fehler **?TYPE MISMATCH ERROR** angezeigt. Fehlt das numerische Argument, wird der Fehler **?SYNTAX ERROR** angezeigt .

Die Laufzeitleistung von RND(0) ist im Vergleich zu RND(1) viel besser und bis zu 5-mal schneller.

Implementierung [Bearbeiten | Quelltext bearbeiten]

Die Implementierung von Commodore [4][5] (bezogen auf die ursprüngliche Diskussion in der Newsgroup comp.sys.cbm [6]) weist einen schwerwiegenden Fehler bei der Erzeugung der Zahlenfolge auf: Trotz des 32-Bit-Startwerts (der einen möglichen Wertebereich von 4 Milliarden ergäbe) könnte die Zufallsfolge ohne Wiederholung auf 723 Werte schrumpfen. Im schlimmsten Fall könnte das Intervall der Folge auf 1 zusammenbrechen und bei der Erzeugung eines einzigen konstanten Wertes hängen bleiben (siehe Beispiel).

Andere Plattformen [Bearbeiten | Quelltext bearbeiten]

RND(0) verwendet möglicherweise bestimmte Hardwareregister, um die Zufälligkeit der generierten Werte zu verbessern.

Plattform	BASIC-Version	Hardwareregister	Notiz
C64	BASIC V2	CIA 1 Timer A und RTC's 1/10s- und 1s-Register	Vorbehalt: Die RTC wird nicht automatisch gestartet, daher tragen ihre Register nichts dazu bei.
VC 20	BASIC V2	VIA 1 Timer A und Timer B	Beide Timer laufen ständig und mischen passende Zufallszahlen. In Emulatorumgebungen treten einige Anomalien auf. Beispielsweise springt beim VICE -Emulator (Version 2.4.20) der Wert von Timer A zwischen 0 und 255 hin und her, was zu einem erheblichen Verlust beim Mischen der Gleitkomma-Mantisse führt und schließlich zu Werten nahe 1 oder 0 führt.
PET CBM 4000 CBM 8000	BASIC 4.0	VIA 1 Timer A und Timer B	Es laufen immer beide Timer, die jeweils passende Zufallszahlen mischen.
Commodore-264-Serie	BASIC 3.5	TED -Timer 1 und Timer 2	Es laufen immer beide Timer, die jeweils passende Zufallszahlen mischen.
C128	BASIC 7.0	CIA 1 Timer A und Timer B	Timer A wird nur während des Datasette -Betriebs verwendet und trägt ansonsten nur 0 Bits zur Mantisse bei.

Die Implementierung in BASIC 4.0+ in der CBM B-Serie verwendet keine hardwarebezogene Quelle für die RND-Funktion und der Parameterwert 0 führt zum gleichen Verhalten wie für >0.

Beispiele [Bearbeiten | Quelltext bearbeiten]

Typische Verwendung [Bearbeiten | Quelltext bearbeiten]

```
X = RND(-TI) : REM Initialisierung
PRINT  INT ( RND (1)*100) : REM Ganzzahlige Zufallszahlen von 0 bis 99
PRINT INT( RND (1)*6)+1 : REM Ganzzahlige Zufallszahlen von 1 bis 6 (für Würfelsimulation)
PRINT INT( RND (1)*49)+1 : REM Ganzzahlige Zufallszahl von 1 bis 49 (für Lottosimulation, zum
Beispiel berühmtes deutsches Lotto 6 aus 49)
PRINT ( RND (0)*101)+100 : REM Zufallszahlen von 100 bis 201
```

RND(0) einschließlich RTC-Werte [bearbeiten | Quelltext bearbeiten]

Lassen Sie die Echtzeituhr zur RND-Funktion beitragen.

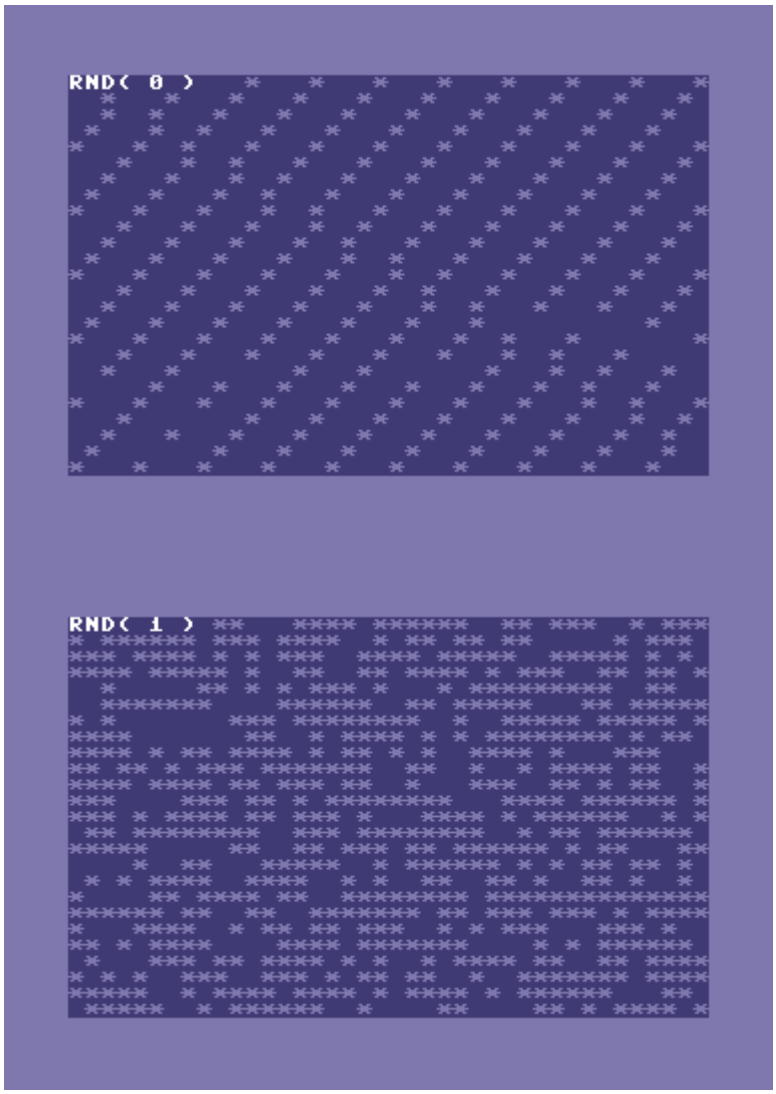
```
POKE 56328,0 : REM Echtzeituhr aktivieren mit 1/10s -> Uhr läuft ($DC08)
PRINT RND (0) : REM Zufallszahl abhängig von Timer und Echtzeituhr
```

RND(0)-Anomalie im Vergleich zu RND(1) [bearbeiten | Quelltext bearbeiten]

Dieses Beispiel zeigt, warum der Befehl `RND(0)` im Vergleich zum Befehl nicht sehr gut ist `RND(1)`:

```
10 REM DIESES PROGRAMM ZEIGT DEN UNTERSCHIED ZWISCHEN RND(0) UND RND(1).
20 REM 1000 STERNE WERDEN FÜR RND(0) GENERIERT,
30 REM. DRÜCKEN SIE NACHDEM DIES ABGESCHLOSSEN IST, EINE BELIEBIGE TASTE, UM RND(1) ZU STARTEN.
40 FÜR I=0 BIS 1
50 PRINT CHR$ (147): REM LÖSCHT BILDSCHIRM
60 POKE 53280,0: REM FESTLEGT RANDFARBE
70 FÜR J=1 BIS 1000
80 POKE 1024+INT( RND (I)*1000),42: NEXT
90 PRINT "{WHITE}{HOME} RND (";I;"){LT. BLUE}": REM {LT. BLUE}=HELLBLAUE
100 POKE 53280,14: REM "SETZT" RANDFARBE ZURÜCK
110 WAIT 198,1: REM WARTET, BIS EINE TASTE GEDRÜCKT WIRD
120 ERHALTEN SIE A$: NÄCHSTES
130 DRUCK CHR$(147)
```

Mit `RND(0)` sehen Sie ein Muster. Dies liegt daran, dass nur 256 Zahlen generiert werden können. Um den Bildschirm mit Sternchen (`* RND(0)`) auszufüllen , benötigen Sie 1000 Zeichen. Beachten Sie jedoch, dass dieser Code nicht prüft, ob eine Stelle mit einem Sternchen gefüllt ist oder nicht. Daher kann es sein, dass ein Sternchen dort eingefügt wird, wo bereits eines war, wodurch es aussieht, als wären weniger Sternchen geschrieben worden.



RND(1) gibt einen konstanten Wert aus [Bearbeiten | Quelltext bearbeiten]

Das folgende Programm [7] initialisiert den PRNG mit einem Startwert, bei dem die Generatorformel denselben Wert erzeugt, was die RND(1)-Funktion nahezu nutzlos macht. Dies könnte als deutlicher Hinweis auf die mangelnde Qualität der Implementierung gewertet werden.

```
1 X=RND(-8008/10371)
2 PRINTRND(1):GOTO2
.166823086
.166823086
.166823086
.166823086
...
```

Wir erhalten 8008/10371 aus der Kettenbruchentwicklung von 1658186197/2147483648 und das ist präzise genug, dass der Gleitkommawert auf einem C64 gleich bleibt. [8]

Referenzen [Bearbeiten | Quelle bearbeiten]

1. ↑ **Wikipedia: Pseudozufälligkeit**
2. ↑ CIA 1-Timer A, der das IRQ- Timing für den KERNAL bereitstellt .
3. ↑ Die RTC wird beim Systemstart nicht aktiviert und trägt ohne explizite Einrichtung nicht zur Zufälligkeit bei.
4. ↑ Disassemblierung $E097/57495: Evaluiere <rnd> auf All_About_Your_64-Online
5. ↑ Auflistung und Diskussion zur Implementierung von RND auf codebase64.org
6. ↑ USENET-Newsgroup comp.sys.cbm: C64 Zufallsgenerator?
7. ↑ Thread: RIP RND(1) auf Forum64.de
8. ↑ Thread: RIP RND(1) (Nachtrag) auf Forum64.de