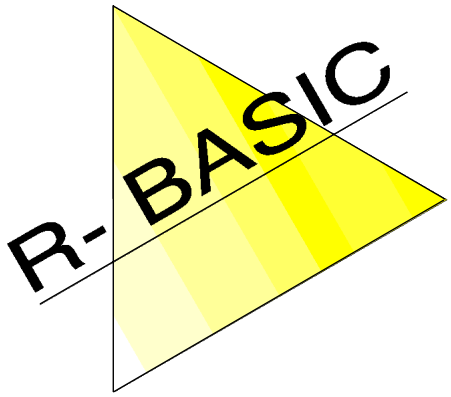
R-BASIC

Einfach unter PC/GEOS programmieren



Programmierhandbuch

Volume 4
Textausgabe, Sound

Version 1.0

(Leerseite)

Inhaltsverzeichnis

2.9 Textausgabe 192
2.9.1 Überblick 192
2.9.2 Der Print-Befehl 194
2.9.3 Der Textbildschirm 196
2.9.3.1 Aufbau des Textbildschirms 196
2.9.3.2 Positionierung von Text 199
2.9.3.3 Ermitteln der Cursorposition 201
2.9.4 Steuerzeichen: Scrollmode, Pagemode und Layoutmode 203
2.9.5 Farben und Schriften 206
2.9.6 Formatierung von Zahlen 207

2.10 Musik und Sound 210
2.10.1 Einfache Töne und Tonfolgen 210
2.10.2 Abspielen von WAV-Dateien 214
2.10.3 Ausgabe von FM-Sounds 219
2.10.3.1 Beschreibung von Instrumenten und Noten 219
2.10.3.2 Spielen von einzelnen Noten 224
2.10.3.3 Ausgabe von FM-Musik 228
2.10.4 Konfiguration der Soundlibrary 234

(Leerseite)

2.9 Textausgabe

Um in R-BASIC Texte auszugeben gibt es prinzipiell zwei Möglichkeiten: Die Verwendung eines Textobjekts (Memo, InputLine, VisText bzw. LargeText) oder die Verwendung des PRINT-Befehls. Die Textobjekte werden ausführlich im Objekthandbuch beschrieben. Die folgenden Kapitel beschäftigen sich mit dem PRINT-Befehl. Dabei wird vorausgesetzt, dass ein Screen existiert. Sehr häufig ist das ein BitmapContent Objekt, das die Anweisung defaultScreen im UI-Code hat.

2.9.1 Überblick

PRINT ist die zentrale Anweisung, um Text oder Zahlen auf dem Bildschirm darzustellen. Die Möglichkeiten von PRINT sind sehr vielfältig. Hier finden Sie deshalb einen Überblick.

Tabelle: Überblick über die PRINT Funktion
<Ausgabeliste> enthält numerische Ausdrücke, String-Ausdrücke oder Steueranweisungen (INK, PAPER, SPC usw.), getrennt durch Komma (tabulierte Ausgabe) oder Semikolon (fortlaufende Ausgabe).

Format	Bedeutung
PRINT **<Ausgabeliste>**	PRINT-Standardformat
PRINT	Cursor in die nächste Zeile setzen
PRINT **INK** (v); **<Ausgabeliste>**	Ausgabe mit Vordergrundfarbe v
PRINT **PAPER** (h); **<Ausgabeliste>**	Ausgabe mit Hintergrundfarbe h
PRINT **COLOR** (v, h); **<Ausgabeliste>**	Farben für diese Ausgabe festlegen
PRINT **TAB** (n); **<Ausgabeliste>**	Ausgabe auf Tabulatorposition n. Siehe auch tabWidth-Systemvariable
PRINT **SPC** (n); **<Ausgabeliste>**	Ausgabe von n Leerzeichen.
Sonderfunktionen	
PRINT **Chr$**(n)	Ausgabe von Steuerzeichen. Bestimmte Zeichen mit n < 32 führen Sonderfunktionen aus.
PRINT **AT** (z, s); **<Ausgabeliste>**	Ausgabe an Cursorposition Zeile (z) und Spalte (s)
PRINT **ATXY** (x,y); **<Ausgabeliste>**	Freie Positionierung an Grafik-koordinaten x, y

Tabelle: Befehle im Zusammenhang mit Textausgaben

Befehl	Bedeutung
LOCATE **z**, **s**	Positionierung des Cursors
WINDOW **za**, **ze**, **sa**, **se**	Text-Fenster einstellen
CsrLin(**n**)	Abfrage der Cursor-Zeile
CsrPos(**n**)	Abfrage der Cursor-Spalte
VGet$	Abfrage des Zeichens an der aktuellen Cursor-Position (Kompatibilitätsbefehl)

Der Cursor

Damit Textausgaben fortlaufend auf dem Bildschirm erfolgen können, verwaltet R-BASIC intern, wohin die nächste Textausgabe erfolgt. Diese Position wird als Cursor-Position bezeichnet. Man sagt z.B. der Cursor steht in Zeile 7 und Spalte 4. Sie können die Cursorposition manuell mit dem Befehl LOCATE verändern.

Farben und Position

Innerhalb einer Print-Anweisung können die Text-Farbe und die Ausgabe-Position geändert werden. Diese Änderungen sind nur für diese eine Print-Anweisung gültig.
Standardmäßig wird der Texthintergrund mit der aktuelle Hintergrundfarbe gelöscht. Wenn Sie Texte transparent ausgeben wollen müssen Sie als Hintergrundfarbe den Spezialwert BG_TRANSPARENT einstellen.

Steuerzeichen

Innerhalb einer Print-Anweisung kann nicht nur Text, sondern auch sogenannte Steuerzeichen ausgegeben werden. Diese Zeichen mit einem ASCII Code unter 32 führen Sonderaufgaben aus. Sie sind weiter unten, in Kapitel 2.9.4 (Steuerzeichen) beschrieben. Insbesondere sei hier auf die Einstellung des Bildschirmmodus (Page-Mode, Scroll-Mode und Layout-Mode) hingewiesen, der sich nur über den Printbefehl ändern lässt.

Volle Kontrolle

Das Verhalten der Print-Anweisung wird über die globale Systemvariable **printFont** gesteuert. Sie enthält alle für die Textausgabe relevanten Daten. Das Feld printFont.**style** enthält die Text-Stile (fett, kursiv usw.). Näheres dazu erfahren Sie weiter unten, im Kapitel 2.9.5 (Farben und Schriften) sowie im Handbuch "Spezielle Themen", Kapitel 2.
Die Formatierung von Zahlen wird über die globale Systemvariable **numberFormat** kontrolliert. Auch hierzu erfahren Sie Näheres weiter unten, im Kapitel 2.9.6 (Formatierung von Zahlen) sowie im Handbuch "Spezielle Themen", Kapitel 1.

2.9.2 Der PRINT-Befehl

PRINT ist die zentrale Anweisung, um Text oder Zahlen auf dem Bildschirm darzustellen.

PRINT

Syntax 1: **PRINT**
Der Cursor wird an den Anfang der nächsten Zeile gesetzt.

Syntax 2: **PRINT <Ausgabeliste>**
<Ausgabeliste> enthält die auszugebenen Daten, jeweils getrennt durch ein Trennzeichen (Komma oder ein Semikolon). Die Liste kann mit einem Trennzeichen abgeschlossen werden.

Elemente der Ausgabeliste können sein:
- numerische Ausdrücke (Zahlen, Variablen, Berechnungen, ...)
- String-Ausdrücke (Variablen, Stringfunktionen, ...)
  Innerhalb von Stringausdrücken können mit der Funktion Chr$() auch Steuerzeichen an den Bildschirm gesendet werden. Eine Liste der Steuerzeichen und ihre Wirkung finden Sie weiter unten.
- Eines der Positionierungs-Kommandos AT, TAB, SPC, ATXY
- Eine Farbanweisung (INK, PAPER, COLOR)

Trennzeichen können sein:
- Ein Komma. Es erfolgt eine tabulierte Ausgabe, d.h. der Cursor wird an die nächste Tabulator-Position gesetzt. Die Tabulatorschrittweite wird von der Systemvariablen tabWidth bestimmt.
- Ein Semikolon. Es erfolgt eine fortlaufende Ausgabe ohne zusätzliche Zwischenräume.

Beispiele:

```
PRINT "Hallo", "Welt"   ' tabuliert
PRINT "Hallo"; "Welt"   ' hintereinander
```

Ausgabe:

```
Hallo        Welt
HalloWelt
```

Steht am Ende einer PRINT-Anweisung eines der Trennzeichen (Komma oder Semikolon), so bleibt der Cursor an der durch das Trennzeichen bestimmten Stelle. Steht kein Trennzeichen am Ende wird der Cursor nach der Ausgabe an den Anfang der nächsten Zeile gesetzt.

Beispiele:

```
A = 100: N = 17
PRINT A, 2*A, sqr(a);                ' keine Zeilenschaltung
PRINT " Otto ist"; N; " Jahre alt"
```

Ausgabe:

```
100      200      10 Otto ist 17 Jahre alt
```

```
A$ = "Ottokar isst Wurst"
PRINT A$;                            ' keine Zeilenschaltung!
PRINT Left$(A$, 7); Right$(A$, 6)
```

Ausgabe:

```
Ottokar isst Wurst Ottokar Wurst
```

```
PRINT AT(0,0); "Ganz links oben"     ' Positionierung unabhängig
                                     ' vom eingestellten Fenster
PRINT COLOR(BLACK, WHITE); " OK "    ' Farbe nur für diese
                                     ' Ausgabe
```

Die Farb- und Positionierungsanweisungen innerhalb des Print-Befehls können beliebig miteinander kombiniert werden.

Beispiel:

```
Print AT(2,2);Paper 0;"Hallo";AT (3,2);Ink 15;"Welt"
```

Beachten Sie, dass in diesem Beispiel die Anweisung PAPER auch auf das Wort "Welt" wirkt. "Welt" erscheint also weiß (15) auf Schwarz (0).

PRINT positioniert den Text immer relativ zum aktuell (mit WINDOW) eingestellten Text-Fenster. Ausnahmen sind die Funktionen PRINT AT und PRINT ATXY. Sie beziehen sich immer auf das maximale Textfenster.
Wurde kein WINDOW-Befehl ausgeführt, ist das maximale Fenster voreingestellt.

Geht der mit Print ausgegebene Text über das Text-Fenster hinaus, so hängt die Reaktion vom eingestellten Modus (PAGE-Mode, SCROLL-Mode oder LAYOUT-Mode) ab. Im LAYOUT-Modus werden Fensterbegrenzungen ignoriert, der Text geht rechts und unten über das Fenster hinaus. Im PAGE- und SCROLL-Mode wird beim Überschreiten der rechten Fensterseite eine neue Zeile eröffnet. Das Überschreiten des unteren Fensterrandes führt im PAGE-Mode dazu, dass der Cursor nach links oben gesetzt wird. Im Scrollmodus wird der Fensterinhalt nach oben geschoben (Scrolling). Dieser Modus steht nur zur Verfügung, wenn der Screen ein BitmapContent-Objekt ist.
Per Default ist für BitmapContent-Objekte der Scroll-Modus voreingestellt. Für alle anderen Objekte ist der Layout-Modus voreingestellt.

2.9.3 Der Textbildschirm

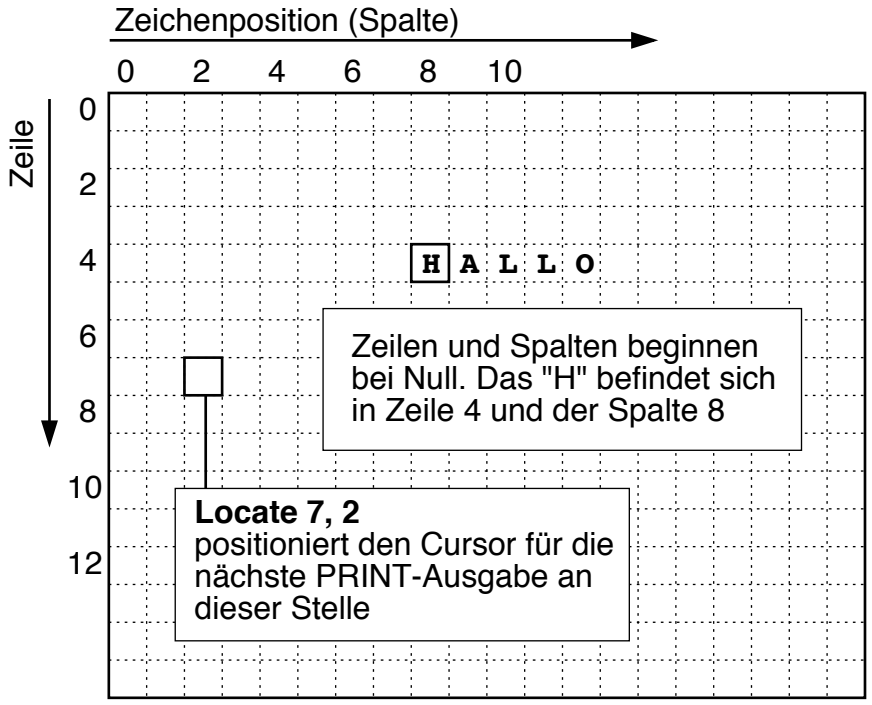
2.9.3.1 Aufbau des Textbildschirms

Der Bildschirm gliedert sich für die Textausgabe in Zeilen und Spalten. Entsprechend der bei Computern üblichen Zählweise beginnen sowohl die Zeilennummern als auch die Spaltennummern bei Null.

LOCATE

Der Befehl PRINT beginnt seine Textausgabe an der aktuellen Cursorposition. Mit der Anweisung LOCATE können Sie diese Position verändern.

Syntax: **LOCATE** **zeile**, **spalte**
zeile: Zeilenposition für die nächste RPINT-Anweisung
spalte: Spaltenposition für die nächste RPINT-Anweisung



Hinweise:

- Ein Standard-Zeichen ist 16 Pixel hoch und 10 Pixel breit. Wenn Sie zum Beispiel ein Ausgabefenster der Größe 640x400 Pixel haben, so fasst der Bildschirm 25 Zeilen zu 64 Zeichen.
- Setzen Sie einen anderen Font (siehe FontSetFixed, FontSetGeos, FontSetBlock), so ändern sich die Werte für Zeichenbreite und Zeilenabstand. Dabei kann es passieren, dass rechts und/oder unten ein Rand bleibt. Der Befehl WINDOW (ohne Parameter) wird beim Aufruf dieser Routinen automatisch ausgeführt und aktiviert das komplette Ausgabefenster (mit Rand, falls erforderlich). Der Rand wird nicht von Text beschrieben, von CLS aber gelöscht.

- Im GEOS-Font-Modus (aktiviert mit FontSetGeos) haben die Zeichen meist keine feste Breite (die meisten GEOS-Fonts sind sog. Proportional-Fonts). Eine feste Zuordnung zwischen Zeichen und Bildschirmposition ist hier nicht sinnvoll. Die Befehle WINDOW und LOCATE rechnen daher mit einer durchschnittlichen Breite von 60% der Zeichenhöhe. Bei Bedarf lässt sich dieser Wert über die System-Variable printFont.charWidth einstellen (Achtung! Profi-Funktion!).

WINDOW

Der Befehl WINDOW (Fenster) schränkt die Textausgabe auf einen bestimmten Bereich des Bildschirms ein. Die Einschränkung gilt nur für Textein- und ausgaben (PRINT, INPUT, CLS, LOCATE einschließlich der darauf wirkenden Farb-Befehle), nicht jedoch für Grafikbefehle.

Syntax 1: **WINDOW** **za**, **ze**, **sa**, **se**

za: erste Zeile
ze: letzte Zeile
sa: erste Spalte
se: letzte Spalte
Die Zählung für Zeilen und Spalten beginnt bei Null.

Syntax 2: **WINDOW**

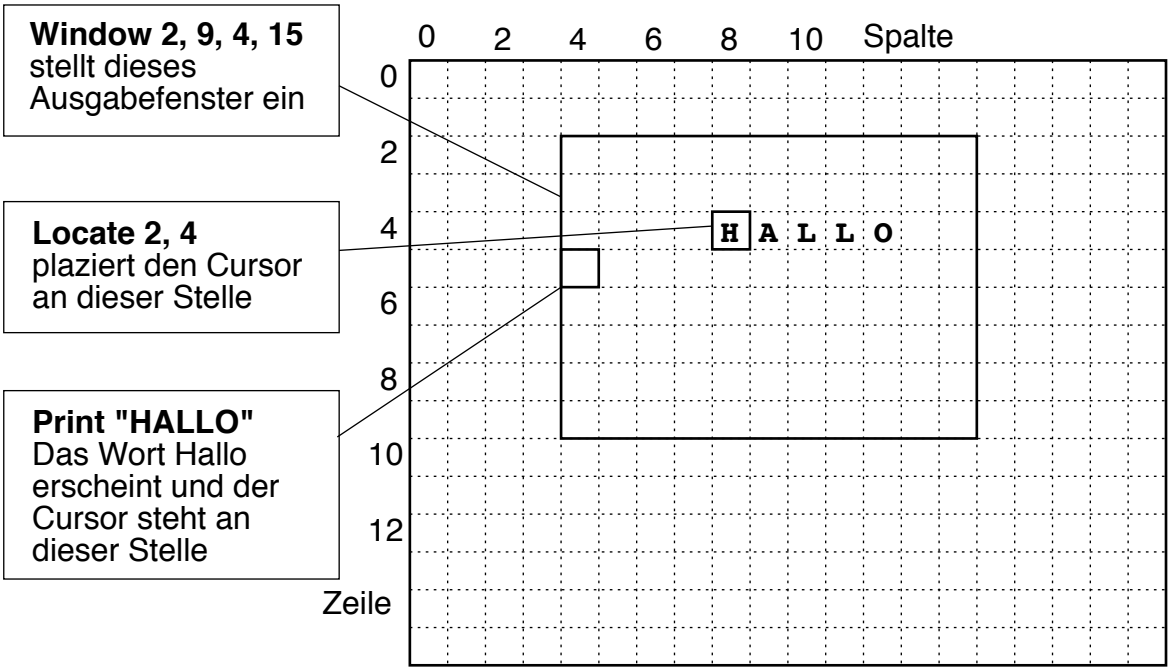
Es wird das maximal mögliche Ausgabefenster eingestellt.

Beispiel:
Die Textausgabe wird auf die Zeilen 10 bis 15 innerhalb der Spalten 20 bis 30 eingeschränkt.

```
WINDOW  10, 15, 20, 30
```

Der Befehl LOCATE bezieht sich immer auf das eingestellte Fenster.
Beispiel:

```
WINDOW 2, 9, 4, 15
LOCATE 2, 4
PRINT "Hallo"
```


Window 2, 9, 4, 15
stellt dieses Ausgabefenster ein
Locate 2, 4
plaziert den Cursor an dieser Stelle
Print "HALLO"
Das Wort Hallo erscheint und der Cursor steht an dieser Stelle
0 2 4 6 8 10 Spalte
0 2 4 6 8 10 12 Zeile
HALLO

2.9.3.2 Positionierung von Text

Die tabWidth - Systemvariable

Werden bei Print die Argumente mit einem Komma getrennt, erfolgt eine tabulierte Ausgabe. Die Systemvariable tabWidth bestimmt dabei die Position der Tabulatoren. Gültige Werte sind ganze Zahlen im Bereich von 1 bis 1024.
tabWidth kann gelesen und geschrieben werden.

Beispiel: Tabulator auslesen und neu setzen

```
Print "Tabulator",  "=", tabWidth    ' Standardwert: 16
tabWidth = 23
Print "Tabulator",  "=", tabWidth    ' 23 erscheint
tabWidth = tabWidth / 2
Print "Tabulator",  "=", tabWidth    ' 11 erscheint (ganzzahlig)
```

Positionierung innerhalb von Print

Innerhalb einer PRINT-Anweisung kann man nicht nur Text ausgeben, sondern auch Positionierungsaufgaben wahrnehmen, damit die Ausgabe ordentlich aussieht. Dafür stehen die Funktionen SPC, TAB, AT und ATXY zur Verfügung.

SPC und TAB

Die Funktion SPC (Spaces) erzeugt eine bestimmte Anzahl von Leerzeichen und gibt diese aus.
Die Funktion TAB (Tabulator, Tabelle) positioniert den Cursor an der angegebenen Spalte in der Zeile, indem so viele Leerzeichen, wie dafür nötig sind, erzeugt und ausgegeben werden.

Syntax: PRINT **SPC**(n); **<Ausgabeliste>**
n: Anzahl der auszugebenden Leerzeichen
<Ausgabeliste> : Liste von Werten, die ausgegeben werden sollen.

Syntax: PRINT **TAB**(n); **<Ausgabeliste>**
n: Position (Spalte), an der der Cursor stehen soll
TAB verwendet **nicht** die globale Variable tabWidth, sondern erwartet direkt die Spalte, relativ zum aktuellen Window.
<Ausgabeliste> : Liste von Werten, die ausgegeben werden sollen.

Beispiele:

```
A = 3
PRINT "Alfa"; SPC(12);"Beta"                ' 12 Leerzeichen
PRINT "Alfa"; TAB(12);"Beta"                ' an Position 12
PRINT A; TAB(8);  A*A;  TAB(24);  A*A*A
```

Ausgabe

```
Alfa            Beta
Alfa        Beta
 3       9              27
```

AT und ATXY

Die Funktion AT ('at' heißt 'an') positioniert den Cursor an der angegebenen Stelle (Zeile, Spalte). Die Funktion ATXY positioniert den Cursor an der angegebenen Grafik-Position (x, y) in Pixeln. Damit ist eine Text-Positionierung unabhängig vom Zeichen-Raster aus Zeilen und Spalten möglich.
Beide Funktionen beziehen sich NICHT auf das aktuelle Text-Window, sondern schreiben den Text bezüglich des maximalen Fensters.

Syntax: PRINT **AT**(**zeile**, **spalte**); **<Ausgabeliste>**
zeile: Zeichen-Zeile, in die die Ausgabe erfolgen soll
spalte: Zeichen-Spalte, in die die Ausgabe erfolgen soll
Zeile und spalte beziehen sich auf das maximale Text-Fenster.
<Ausgabeliste> : Liste von Werten, die ausgegeben werden sollen.

Syntax: PRINT **ATXY**(**x**, **y**); **<Ausgabeliste>**
x, y: Koordinaten, an der der Text erscheinen soll
Die linke obere Ecke des ersten Text-Zeichens erscheint an dieser Stelle. Beachten Sie, dass hier die Grafik-Konventionen für die Koordinaten gelten, d.h. es folgt zuerst die x-Koordinate (entsprechend der Spalte) und dann die y-Koordinate (entsprechend der Zeile). Bei PRINT AT() und LOCATE ist es andersherum (erst die Zeile und dann die Spalte).
<Ausgabeliste> : Liste von Werten, die ausgegeben werden sollen.

Beispiele:

```
PRINT AT(0,7); "****************"
' Ausgabe untereinander:
PRINT AT(3,10); "A";AT(4,10); "B";AT(5,10); "C";
PRINT ATXY(0,300);"xyz"                  ' links, unten (!)
```

Hinweise:

- Hinter SPC, TAB, AT und ATXY sollte immer ein Semikolon (kein Komma) folgen, da ein Komma die Cursorposition wieder ändern kann (Anwahl der nächsten Tabulator-Position).
- Die Klammern um die Argumente sind optional.

```
PRINT SPC 7, "sieben Leerzeichen"
```

- Diese Anweisungen können beliebig mit den Farb-Anweisungen kombiniert werden.
- Sie sollten AT bzw. ATXY nicht mit TAB oder VGet$ kombinieren. Die Ergebnisse sind häufig unerwartet.

2.9.3.3 Ermitteln der Cursorposition

Sie können in R-BASIC sowohl die aktuelle Cursor-Position ermitteln (CsrLin, CsrPos) als auch das auf der Cursor-Position stehende Zeichen lesen (VGet$).

CsrLin

Die Funktion CsrLin (d.h. Cursor Line, Zeile des Cursors) liefert die Zeile, in der sich der Cursor gerade befindet. Der Parameter n bestimmt, worauf sich CsrLin beziehen soll.

Syntax: **<numVar>** = **CsrLin** (**n**)

n > 0: Zeilennummer bezüglich des aktuellen Text-Fensters
n = 0: Zeilennummer bezüglich des maximalen Fensters
n < 0: Sonderfunktion: Anzahl der Zeilen im aktuellen Fenster

---

Beispiel:

```
WINDOW 5, 10, 8, 45
Color 7, 0: CLS
Print  CsrLin(1); CsrLin(0)
Print "Das Fenster hat"; CsrLin(-1); "Zeilen"
```

CsrPos

Die Funktion CsrPos (d.h. Cursor Position, Spalte des Cursors) liefert die Spalte, in der sich der Cursor gerade befindet. Der Parameter n bestimmt, worauf sich CsrPos beziehen soll.

Syntax: **<numVar>** = **CsrPos** (**n**)

n > 0: Spaltennummer bezüglich des aktuellen Text-Fensters
n = 0: Spaltennummer bezüglich des maximalen Fensters
n < 0: Sonderfunktion: Anzahl der Spalten im aktuellen Fenster

---

Beispiel:

```
WINDOW 5, 10, 8, 45
Color 7, 0: CLS
Print  CsrPos(1)
Print  CsrPos(0)
Print "Das Fenster hat"; CsrPos(-1); "Spalten"
```

Hinweis:
Die Funktionen AT und ATXY aktivieren intern kurzzeitig das maximale Textfenster. Verwenden Sie CsrLin, CsrPos und VGet$ deswegen nicht als Argumente einer PRINT-Anweisung, die auch AT oder ATXY enthält.

Fehlerhaft:

```
Print AT ( 10, 9); CsrPos(-1)
```

Korrekt:

```
A = CsrPos(-1)
Print AT ( 10, 9); A
```

VGet$

Die Funktion VGet$ (d.h. Video Get - Hole von der Grafikkarte) liefert das Zeichen, das sich an der aktuellen Cursor-Position befindet.
VGet$ ist ein Kompatibilitätsbefehl für ältere BASIC Programme, die für einen echten Textmodus geschrieben wurden.

Syntax: C$ = VGet$

Anmerkung:

- Nachdem der Bildschirm mit CLS gelöscht wurde, liefert VGet$ an unbeschriebenen Positionen Null-Zeichen (Leerstrings). ASC(VGet$) liefert Null.
- Ein Aufruf der Funktionen FontSetFixed oder FontSetBlock löscht den von VGet$ verwendeten Speicherbereich. VGet$ liefert an nicht erneut beschriebenen Positionen Leerstrings.
- VGet$ steht im GEOS-Font-Modus (nach FontSetGeos) nicht zur Verfügung.
- Für nicht unterstützte Screenobjekte oder wenn der Screen mehr als 8000 Zeichen enthalten kann liefert VGet$ immer ein Null-Zeichen (Leerstring)
- VGet$ steht im LAYOUT-Modus nicht zur Verfügung.

Beispiel:

```
Dim a$
WINDOW
Color 7, 0: CLS
Print "Hallo BASIC"
Locate 0, 7
a$ = VGet$
Print  AT(2, 7); a$
```

Hinweis: Die Funktionen AT und ATXY aktivieren intern kurzzeitig das maximale Textfenster. VGet$ deswegen nicht als Argument einer PRINT-Anweisung, die auch AT oder ATXY enthält.

2.9.4 Steuerzeichen: Scrollmode, Pagemode und Layoutmode

ASCII-Codes im Bereich von Null bis 31 ist kein Buchstabe, sondern eine Steuerfunktion zugeordnet. Einige dieser Codes können von R-BASIC in der PRINT-Anweisung ausgewertet werden. Im Folgenden finden Sie eine Zusammenstellung der verfügbaren Steuerzeichen. Besondere Bedeutung kommt dabei dem PAGE-Mode, dem SCROLL-Mode und dem LAYOUT-Mode zu, da diese nur über die Steuerzeichen und PRINT eingestellt werden können.

SCROLL-Mode

Am Zeilenende wird in die nächste Zeile gewechselt. Wenn das untere Ende des Text-Fensters erreicht wird, schiebt R-BASIC den Fensterinhalt um eine Zeile nach oben. Am unteren Fensterrand entsteht eine Leerzeile. Der Scrollmode steht nur zur Verfügung, wenn der Screen ein BitmapContent-Objekt ist.

PAGE-Mode

Am Zeilenende wird in die nächste Zeile gewechselt. Wenn das untere Ende des Text-Fensters erreicht wird setzt R-BASIC den Cursor wieder nach links oben.

LAYOUT-Mode

Es gibt weder einen automatischen Zeilenumbruch noch eine Begrenzung auf das Textfenster. Der Cursor kann frei positioniert werden, auch mit negativen Koordinaten (links oder oberhalb des Textfensters). Texte können über das Fenster hinausragen (und damit auch unsichtbar werden).

Um ein Steuerzeichen auszugeben verwenden Sie innerhalb einer Print-Anweisung die Stringfunktion Chr$() oder einen Backslash '\', gefolgt vom entsprechenden ASCII-Code.

Syntax: PRINT **Chr$**(**code**); ' z.B. Print Chr$(19);
PRINT "**\code**"; ' z.B. Print "\19";
code: das auszuführende Steuerzeichen

Hinweise:

- Nicht unterstützte (d.h. nicht in der Tabelle aufgeführte) Steuercodes werden ignoriert.
- Nach Chr$(code) sollte ein Semikolon folgen.
- Ist code > 31, erzeugt Chr$() das entsprechende ASCII-Zeichen.
- Im GEOS-Font-Modus arbeiten die meisten Steuercodes nur eingeschränkt (siehe Beschreibung in der Tabelle)
- Im GEOS-Font-Modus existieren keine festen Zeichen-Positionen, da die meisten Buchstaben verschieden breit sind. Die Steuerzeichen arbeiten daher

mit einer "durchschnittlichen" Zeichenbreite. Ob die Funktion für Ihre Zwecke brauchbar ist, müssen Sie ausprobieren.
Grundsätzlich NICHT korrekt arbeiten Codes, die einzelne Zeichen löschen oder einfügen (ASC_CLEAR, ASC_BACKSPACE, ASC_DEL, ASC_INS).

- Mit der Ausnahme der Codes 9 (ASC_TAB) und 13 (ASC_ENTER) können Textobjekte keine Steuerzeichen verarbeiten. In einigen Fällen kann es sogar zum Crash kommen, wenn Sie andere Steuerzeichen an ein Textobjekt übergeben.

Um die symbolischen Namen (z.B. ASC_UP) verwenden zu können, müssen Sie die KeyCodes-Library einbinden.

```
Include "KeyCodes"    ! Groß-/Kleinschreibung beachten
```

Tabelle: ASCII Steuercodes

InKey$-Taste: Gedrückte Taste, damit InKey$ diesen Code zurückgibt.
Bedeutung in Print: Funktion, die mit Print Chr$(code) ausgeführt wird.

ASCII-Code dez. hex.	Konstanten-Name (KeyCodes-Library)	InKey$ Taste	Bedeutung in Print
01 (&H01)	ASC_CLEAR	-	Zeichen löschen, Cursor an nach links
02 (&H02)	ASC_CLEAR_LINE	-	Zeile löschen, Cursor an Zeilenanfang
03 (&H03)	-	-	-
04 (&H04)	-	-	-
05 (&H05)	-	-	-
06 (&H06)	-	-	-
07 (&H07)	ASC_BEEP	-	Signalton (Beep)
08 (&H08)	ASC_BACKSPACE	Backsp.	Rückwärtsschritt. Zeichen auf Cursorposition löschen und Cursor nach links
09 (&H09)	ASC_TAB	TAB	Anwahl der nächsten Tabulatorposition
10 (&H0A)	ASC_DOWN	↓	Cursor 1 Zeile tiefer
11 (&H0B)	ASC_UP	↑	Cursor 1 Zeile höher
12 (&H0C)	ASC_CLS	-	Bildschirm löschen
13 (&H0D)	ASC_ENTER	Enter	Cursor an den Anfang der nächsten Zeile
14 (&H0E)	ASC_LEFT	←	Cursor 1 Zeichen nach links

15 (&H0F)	ASC_RIGHT	→	Cursor 1 Zeichen nach rechts
16 (&H10)	ASC_HOME	-	Cursor nach links oben
17 (&H11)	ASC_PAGE_MODE		PAGE-Mode einstellen
	ASC_PAGE_UP	Bild ↑	
18 (&H12)	ASC_SCROLL_MODE		SCROLL-Mode einstellen
	ASC_PAGE_DOWN	Bild ↓	
19 (&H13)	ASC_LAYOUT_MODE	-	LAYOUT-Mode einstellen
20 (&H14)	ASC_POS_END	Ende	-
21 (&H15)	ASC_POS_1	Pos 1	Cursor an Zeilenanfang
22 (&H16)	ASC_INS	Einfg	ein Zeichen an der Cursorposition einfügen
23 (&H17)	ASC_DEL	Entf	ein Zeichen an der Cursorposition löschen
24 (&H18)	-	-	-
25 (&H19)	-	-	-
26 (&H1A)	-	-	-
27 (&H1B)	ASC_ESC	ESC	-
28 (&H1C)	-	-	-
29 (&H1D)	-	-	-
30 (&H1E)	-	-	-
31 (&H1F)	-	-	-

2.9.5 Farben und Schriften

Standardmäßig verwendet Print die aktuelle Vorder- und Hintergrundfarbe, die mit einem der Befehle INK, PAPER oder COLOR eingestellt wurden. Sie können die Farben jedoch für eine einzelne Print-Anweisung ändern, indem Sie die Funktionen INK, PAPER oder COLOR innerhalb der Print-Anweisung angeben. Diese Farbwerte gelten dann nur für diese eine Print-Anweisung.

INK, PAPER, COLOR

Syntax: PRINT **INK**(**v**); **<Ausgabeliste>**
PRINT **PAPER**(**h**); **<Ausgabeliste>**
PRINT **COLOR**(**v**, **h**); **<Ausgabeliste>**
v: Vordergrund (Text-) Farbe
h: Hintergrundfarbe
<Ausgabeliste> : Liste von Werten, die ausgegeben werden sollen.

Hinweis:
Um Texte oder Blockgrafik-Zeichen transparent auszugeben (d.h. der Hintergrund wird nicht gelöscht) verwenden Sie als Hintergrundfarbe die spezielle Konstante BG_TRANSPARENT (numerischer Wert: 4096).
BG_TRANSPARENT kann sowohl innerhalb der Print-Anweisung als auch mit den BASIC-Befehlen PAPER und COLOR verwendet werden.

Beispiel:

```
PRINT AT (7, 12); COLOR (BLACK, WHITE);  " ACHTUNG! "
PRINT AT (7, 12); PAPER (BG_TRANSPARENT);  " ACHTUNG! "
```

Hinweise:

- Die Farbanweisungen können beliebig mit den Positionierungsanweisungen AT und ATXY kombiniert werden.
- Hinter den Farbanweisungen sollte immer ein Semikolon folgen, damit die Cursorposition nicht verändert wird.

Verwendung von Schriften

Print verwendet immer die aktuell eingestellte Schriftart und Schriftgöße. Die folgende Tabelle enthält die wesentlichen Befehle zum Einstellen von Schriftart und Schriftgröße. Eine ausführliche Beschreibung sowie weitere Befehle zur Verwaltung von Schriften finden Sie im Handbuch "Spezielle Themen", Kapitel 2 (Verwendung von Schriften) und Kapitel 4 (Verwendung des Block-Grafik-Modus). Der Block-Grafik-Modus erlaubt die Ausgabe von selbst erstellten Grafikzeichen mit dem Print-Befehl.

Befehl / Variable	Aufgabe
FontSetFixed (**fontID**, **size [,lineHeight]**)	Einstellen einer Schrift mit fester Zeichenbreite
FontSetGeos(**fontID**, **size [,lineHeight]**)	Einstellen einer beliebigen GEOS-Schriftart
FontSetBlock(**sizex, sizey [, colored]**)	Einstellen des Block-Grafik-Modus
printFont	Systemvariable, enthält alle für die Textausgabe relevanten Daten. Das Feld printFont.style enthält die Text-Stile (fett, kursiv usw.).
printFontStruct	Datentyp der globalen Systemvariablen printFont

Printfont.style enthält Bitflags. Die Arbeit mit Bitflags ist im Kapitel 2.3.5.4 (Sonderfall: Bitflags) erläutert.

Beispiele:

```
'Setzen des Stils "unterstrichen", auch wenn er schon gesetzt ist
 printFont.style = printFont.style  OR  TS_UNDERLINE
```

```
 ' Rücksetzen des Stils "kursiv", auch wenn er nicht gesetzt war
 printFont.style = printFont.style  AND NOT TS_ITALIC
```

2.9.6 Formatierung von Zahlen

Wenn Sie mit dem Print-Befehl eine Zahl ausgeben gibt R-BASIC Vor- und Nachkommastellen aus oder wechselt in die Exponentialdarstellung, je nachdem in welchem Bereich die Zahl liegt.
Zur Einstellung des Zahlenformats stehen in R-BASIC die folgenden Befehle und Variablen zur Verfügung. Sie sind ausführlich im Kapitel 1 des Handbuchs "Spezielle Themen" beschrieben. Wenn Sie zum Beispiel selbst festlegen wollen, wie viele Nachkommastellen ausgegeben werden, können Sie die Systemvariable **numberFormat** manuell ändern.

Befehl / Variable	Bedeutung
numberFormat	Systemvariable. Enthält alle für die Zahlenformatierung relevanten Informationen.
NumberFormatStruct	Typ der globalen Systemvariablen numberFormat.
SetNumberFormat	Einstellen eines Standard-Formats

Für viele Zwecke reichen die R-BASIC Standard-Zahlenformate aus, so dass Sie sich meistens nicht mit der komplexen Belegung der **numberFormat-**Variable

beschäftigen müssen. Im Folgenden finden Sie eine Kurzbeschreibung der Anweisung SetNumberFormat. Eine vollständige Beschreibung sowie weitere Syntaxvarianten finden Sie im Kapitel 1 des Handbuchs "Spezielle Themen".

SetNumberFormat

Stellt das Zahlenformat für die Anzeige von Zahlen ein.

Syntax:	**SetNumberFormat** (**format**) Die Systemvariable numberFormat wird belegt.
format:	einzustellendes Format (ein numerischer Wert, siehe Tabelle)

Für 'format' stehen folgende Werte zur Verfügung:

Wert	Konstante	Wirkung
0	NF_NORMAL	Standardeinstellung von R-BASIC. Automatische Umschaltung in die Exponentialdarstellung.
1	NF_CURRENCY	Währungstypische Darstellung mit 2 Nachkommastellen.
2	NF_MAX_PREC	Maximale Genauigkeit mit 15 Stellen.
3	NF_SCI_3	Wissenschaftliche Darstellung mit 3 Stellen z.B. 34.8E+06
4	NF_SCI_4	Wissenschaftliche Darstellung mit 4 Stellen z.B. 34.82E+06
5	NF_INTEGER	Ganzzahlige Darstellung
6	NF_FIXED_3	Immer 3 Nachkommastellen
7	NF_FIXED_4	Immer 4 Nachkommastellen

Beispiele:

```
SetNumberFormat ( NF_NORMAL )
SetNumberFormat ( NF_SCI_4 )
```

(Leerseite)

2.10 Musik und Sound

Unter R-BASIC gibt es folgende Möglichkeiten Töne auszugeben:

- Ausgabe einfacher Töne oder Tonfolgen (Befehle **Beep**, **StandardSound**, **Sound** und **KCSound**)
- Abspielen von WAV-Dateien
  - **PlayWav** spielt eine WAV-Datei im Vordergrund ab.
  - **PlayWavBG** spielt eine WAV-Datei im Hintergrund ab, wobei eine Warteschlange für mehrere WAV-Dateien organisiert wird.
- Ausgabe vom FM-Sound (einzelne Töne und komplexe Melodien)
  - **PlayNote** spielt eine einzelne Note.
  - **PlayNoteHan** spielt eine einzelne Note, wobei Sie die Tonausgabe vorzeitig stoppen können.
  - **PlayMusic** spielt ein FM-Musikstück.
  - **PlayMusicBG** spielt ein FM-Musikstück im Hintergrund, wobei eine Warteschlange für mehrere Musikstücke organisiert wird.
  - **PlayMusicHan** spielt ein FM-Musikstück, wobei Sie die Tonausgabe vorzeitig stoppen können.

Beachten Sie, dass die Soundausgabe in den Voreinstellungen aktiviert sein muss, damit R-BASIC Töne ausgeben kann. R-BASIC hat weder die Möglichkeit, das zu prüfen, noch die Einstellungen zu verändern.

Beispiele zur Verwendung der Sound- und Musik-Befehle finden Sie im Ordner "R-BASIC\Beispiel\Sound".

2.10.1 Einfache Töne und Tonfolgen

BEEP

Gibt einen oder mehrere kurze Töne aus. Tonhöhe und Länge können nicht verändert werden.

Syntax: **BEEP** [**n**]
n: num. Wert, Anzahl der Töne
ohne n: 1 Ton

Beispiel:

```
BEEP 3 ' Drei kurze Töne
```

StandardSound

Gibt einen der sechs GEOS Standardtöne aus.

Syntax: **StandardSound n**
n: num. Wert, wählt den Standardton aus (siehe Tabelle)

Konstante	Wert	Verwendet für
SS_ERROR	0	Fehler
SS_WARNING	1	Warnung
SS_NOTIFY	2	Information
SS_NO_INPUT	3	Keine Eingabe möglich
SS_KEY_CLICK	4	Tastenanschlag
SS_ALARM	5	Alarm

Beispiel:

```
StandardSound SS_ALARM
```

Sound

Die Anweisung SOUND spielt eine Tonfolge mit bis zu 11 Tönen. Die Töne werden dabei durch ihre Frequenz und die Tondauer bestimmt. Außerdem können Sie die Lautstärke und die verwendeten Instrumente auswählen und Sie können entscheiden, ob die Tonfolge im Vordergrund oder im Hintergrund gespielt wird.
"Im Vordergrund" bedeutet, dass R-BASIC so lange wartet, bis die Tonfolge beendet ist.
"Im Hintergrund" bedeutet, dass R-BASIC weiterarbeitet, noch während die Tonfolge gespielt wird. Dabei können Sie festlegen, dass eine Warteschlange verwendet wird. Das bedeutet, dass die nächste Sound-Tonfolge erst gespielt wird, wenn alle vorher angegebenen Sound-Tonfolgen fertig sind.
Wenn Sie festlegen, dass keine Warteschlange verwendet werden soll, dann werden allen nacheinander angegebenen Sound-Tonfolgen sofort - und damit faktisch gleichzeitig - abgespielt, vorausgesetzt die Sound-Hardware lässt das zu.

Syntax: **Sound "control", <parameterList>**
"control" Zeichenkette, bestehend aus einem der Buchstaben ’q’, ’b’ oder ’f’ sowie einer Zahl.
’q’: ("queue") Spielen im Hintergrund mit Warteschlange
’b’: ("background") Spielen im Hintergrund ohne Warteschlange
’f’: ("foreground") Spielen im Vordergrund
Die Zahl gibt die Lautstärke an. Erlaubte Werte 0 (Stille) bis 100 (maximale Lautstärke)
Wird kein Buchstabe angegeben wird ’f’ angenommen
Wird keine Zahl angegeben wird 100 angenommen.
d.h.: Leerstring bedeutet: volle Lautstärke im Vordergrund

<parameterList> Liste von bis zu 22 numerischen Werten n mit folgender Bedeutung:
n < 0: Der Wert bezeichnet ein Instrument. Siehe unten.
n = 0: Pause. Es folgt eine die Dauer der Pause in Sekunden.
n > 0: Tonfrequenz. Es folgt die Tondauer in Sekunden.

Um die Klangfarbe der Töne zu ändern können Sie jederzeit ein anderes Instrument auswählen. Dazu geben Sie die Nummer des Instruments (siehe nächstes Kapitel) als negative Zahl an. Startwert für jeden Sound-Befehl ist das Instrument 19 (IP_CHURCH_ORGAN).
Tipps:

- Um das Instrument Null (IP_GRAND_PIANO) auszuwählen verwende Sie den Wert -0.2.
- Der Sound-Befehl verwendet die gleiche Hintergrundwarteschlange wie der Befehl PlayMusicBG (Kapitel 2.10.3.3). Sie können also z.B. den Befehl StopMusicBG (siehe Kapitel 2.10.3.3) verwenden, um die Warteschlange für Sound-Befehle zu leeren.

Beispiel Morsecode des Buchstaben M (Lang - Pause - Lang) in voller Lautstärke im Vordergrund. Tonfrequenz: 1000 Hz.

```
Sound "", 1000, 0.3, 0, 0.1, 1000, 0.3
```

Beispiel Morsecode der Buchstaben "MMS" im Vordergrund

```
Sound "", 1000, 0.3, 0, 0.1, 1000, 0.3     ' M
Sound "", 0, 0.3                           ' Pause
Sound "", 1000, 0.3, 0, 0.1, 1000, 0.3     ' M
Sound "", 0, 0.3                           ' Pause
Sound "", 1000, 0.1, 0, 0.1, 1000, 0.1, 0, 0.1, 1000, 0.1   ' S
```

Beispiel Morsecode der Buchstaben "MMS" im Hintergrund. Durch die Angabe des Steuerzeichens "q" (Warteschlange) werden die Töne nacheinander gespielt.

```
Sound "q", 1000, 0.3, 0, 0.1, 1000, 0.3        ' M
Sound "q", 0, 0.3                              ' Pause
Sound "q", 1000, 0.3, 0, 0.1, 1000, 0.3        ' M
Sound "q", 0, 0.3                              ' Pause
Sound "q", 1000, 0.1, 0, 0.1, 1000, 0.1, 0, 0.1, 1000, 0.1  ' S
```

Beispiel mit einem anderen Instrument

```
Sound "", 440, 0.2, 0, 0.1, 440, 0.2, 0, 0.1,\
         -12, \
         440, 0.2, 0, 0.1, 440, 0.2, 0, 0.1, 660, 0.2
```

KCSound

KCSound ist ein Kompatibilitätsbefehl. Er ersetzt den Sound-Befehl des KC85-Homecomputers. Im Original steuert der Befehl die Hardware des Computers (die CTC-Einheit) an. Die CTC teilt die Taktfrequenz dieses Computers (1,75 MHz) herunter und erzeugt so einen hörbaren Ton. Dabei stehen zwei Kanäle zur Verfügung. Jeder Kanal hat einen Vorteiler (erlaubte Werte: 0 und 1) und einen Zähler (erlaubte Werte 0 bis 255). Wird der Zähler auf Null gesetzt, erfolgt keine Tonausgabe auf diesem Kanal.
Die ausgegebene Frequenz berechnet sich aus dem Vorteiler v und dem Zähler z folgendermaßen:

Vorteiler v = 0: f = 1.75E6/(32*z)
Vorteiler v = 1: f = 1.75E6/(512*z)

Syntax: **KCSound z1**, **v1**, **z2**, **v2** [, **laut** [, **time**]]

v1, z1: Vorteiler und Zähler für Kanal 1
v2, z2: Vorteiler und Zähler für Kanal 2
Erlaubte Werte: v1, v2: 0 oder 1
z1, z2: 0 .. 255 (nur ganzzahlig)
Ist z = 0 erfolgt keine Tonausgabe auf diesem Kanal.
laut: Lautstärke (für beide Kanäle), erlaubte Werte: 0 .. 31
time: Zeitdauer (für beide Kanäle), erlaubte Werte: 0 .. 255
Die Zeitbasis ist 1/50s (=20 ms)
time = 0 erzeugt einen Dauerton bis zum nächsten KCSound Befehl (aber maximal 18 Minuten).

Werden "laut" und/oder "time" nicht angegeben, so bleiben die zuletzt verwendeten Werte erhalten.

Hinweise:
- Unter R-BASIC sind für z und time auch größere Werte zulässig.
- Zur Berechnung des Zählers für eine bestimmte Frequenz f kann man folgende Formeln verwenden:
  Vorteiler v = 0 z = 54687.5 / f
  Vorteiler v = 1 z = 3418 / f
- Der Klang kann nicht verändert werden.

Beispiel: Gleichzeitige Ausgabe der Frequenzen 220 Hz (tiefes a) und 330 Hz in voller Lautstärke für 1 Sekunde.

```
KCSound 249, 0, 166, 0, 31, 50
```

Beispiele zur Verwendung der Befehle Sound und KCSound finden Sie in der Beispieldatei "R-BASIC\Beispiel\Sound\Sound Demo".

2.10.2 Abspielen von WAV-Dateien

GEOS verfügt über die Möglichkeit WAV-Dateien (*.WAV") abzuspielen. GEOS selbst kann nur Dateien mit dem Format "8 Bit Mono" abspielen. R-BASIC umgeht dieses Problem, indem es andere Formate (z.B. "16 Bit Stereo") vorher nach 8 Bit Mono konvertiert. Da das WAV Format ein sehr einfaches Datenformat ist und die WAV-Dateien meist nicht sehr groß sind führt das in den meisten Fällen zu keiner hörbaren Verzögerung.
Unter GEOS kann immer nur eine WAV-Datei gleichzeitig abgespielt werden. R-BASIC organisiert eine "Warteschlange", wenn Sie versuchen, mehrere WAV-Sounds gleichzeitig abzuspielen. Sie werden dann nacheinander abgespielt.
GEOS verfügt aktuell nicht über die Möglichkeit, in das Abspielen von WAV-Dateien einzugreifen. So hat man z.B. nicht die Möglichkeit die Lautstärke zu ändern oder das Abspielen abzubrechen. Dieses Problem kann R-BASIC nicht umgehen.

Überblick

Anweisung	Aufgabe
PlayWav "file"	Spielt eine WAV-Datei ab
PlayWavBG "file" [, Handler]	Spielt eine WAV-Datei im Hintergrund ab. Kann anschließend einen Action-Handler aufrufen.
<numVar> = GetWavBGCount()	Liest die Anzahl der WAV-Dateien in der Hintergrundwarteschlange
StopWavBG	Leert die Hintergrundwarteschlange für WAV-Dateien
<numVar> = GetWavInfo ("file", info)	Liest Informationen aus einer WAV-Datei aus.
ConvertWav "srcFile", "destFile"	Konvertiert eine WAV-Datei in das Format 8 Bit Mono

PlayWav

PlayWav spielt eine WAV-Datei ("*.wav") ab. Falls erforderlich wird die Datei vorher in das Format "8 Bit Mono" konvertiert. Dazu wird eine temporäre Kopie erstellt, die Originaldatei wird nicht verändert. Der nächste R-BASIC-Befehl wird erst abgearbeitet, wenn die WAV-Datei vollständig abgespielt wurde.
PlayWav setzt die globale Variable "fileError".
Hinweis: GEOS kann nur eine WAV-Datei gleichzeitig abspielen. Die WAV-Datei wird daher nicht abgespielt, wenn bereits eine WAV-Datei im Hintergrund abgespielt wird. Verwenden Sie PlayWavBG um die Datei in die Warteschlange zu stellen.

Syntax: **PlayWav "filename.wav"**
filename.wav: Name der abzuspielenden Datei. Sie muss sich im aktuellen Ordner befinden. Pfadangaben sind zugelassen. Zur Sicherheit sollten Sie Großbuchstaben verwenden.

Beispiele:

```
  PlayWav "TATA.WAV"
  Print "Fertig"
```

```
DIM t$
  t$ = "TATA.WAV"
  PlayWav "C:\\MUSIC\\CHORD1.WAV"
  PlayWav t$
  Print "Fertig"
```

Die Print-Anweisung wird jeweils ausgeführt nachdem der letzte Ton der Datei "TATA.WAV" verklungen ist.

PlayWavBG

PlayWavBG spielt eine WAV-Datei ("*.wav") im Hintergrund (background) ab. R-BASIC arbeitet weiter während die Datei im Hintergrund abgespielt wird. Sollte bereits eine WAV-Datei im Hintergrund gespielt werden so wird die neue Datei auf die "Warteliste" gesetzt und erst abgespielt, sobald die anderen Dateien auf der Liste abgespielt wurden.
Sie können PlayWavBG den Namen eines Action-Handlers übergeben. Dieser Handler wird automatisch aufgerufen, sobald die Datei vollständig abgespielt wurde. Er muss als TimerAction oder als ButtonAction deklariert sein. Ihr Programm kann so auf dem Laufenden bleiben, ob im Hintergrund noch WAV-Sounds spielen oder nicht.
Der Handler wird auch gerufen, wenn die Datei nicht abgespielt werden konnte. Im Parameter actionData finden Sie einen Fehlercode bzw. Null, wenn die Datei korrekt abgespielt wurde. Eine Liste der Fehlercodes finden Sie im Anhang B. Verwenden Sie die Funktion ErrorText$, um den Fehlercode in einen anschaulichen Text zu konvertieren.
Falls erforderlich wird die Datei vor der Ausgabe in das Format "8 Bit Mono" konvertiert. Dazu wird eine temporäre Kopie erstellt, die Originaldatei wird nicht verändert.
PlayWavBG setzt die globale Variable "fileError".

Syntax: **PlayWavBG "filename.wav"** [, **HandlerName**]

filename.wav:	Name der abzuspielenden Datei. Sie muss sich im aktuellen Ordner befinden. Pfadangaben sind zugelassen. Zur Sicherheit sollten Sie Großbuchstaben verwenden.
HandlerName:	Optional: Action-Handler, der aufgerufen wird, wenn die Datei abgespielt wurde. Er muss als TimerAction oder als ButtonAction deklariert sein.

Beispiele:

```
  PlayWavBG "TRUMPET.WAV"
  Print "Fertig"
```

```
PlayWavBG "TRUMPET.WAV"
PlayWavBG "C:\\MUSIC\\CHORD1.WAV", ZwischenHandler
PlayWavBG "TATA.WAV", FinishHandler
Print "Fertig"
```

Die Print-Anweisung wird jeweils ausgeführt noch während die Datei "TRUMPET.WAV" abgespielt wird. Im zweiten Beispiel werden die Handler "ZwischenHandler" und "FinishHandler" aufgerufen, nachdem die entsprechenden Dateien abgespielt wurden.

```
TIMERACTION ZwischenHandler
Print "Jetzt spielt 'TATA' "
End ACTION
```

```
TIMERACTION FinishHandler
Print "Nun ist Ruhe."
Print ErrorText$( actionData )    ' ggf Fehlercode ausgeben
End ACTION
```

GetWavBGCount

Diese Funktion liefert die Anzahl der WAV-Dateien, die in der Wartschlange sind. Dabei wird die aktuell spielende Datei mitgezählt. Der Rückgabewert 1 bedeutet also, dass gerade eine WAV-Datei im Hintergrund spielt, die Warteschlange aber leer ist.

Syntax: **<numVar>** = **GetWavBGCount** ( )

StopWavBG

StopWavBG löscht die Warteschlange der für die WAV-Ausgabe verantwortlichen Library. Der aktuell im Hintergrund spielende WAV-Sound wird nicht abgebrochen (das wird vom System nicht unterstützt). StopWavBG darf auch gerufen werden, wenn gerade kein WAV-Sound spielt. Es wird empfohlen, StopWavBG im OnExit-Handler zu rufen, wenn man sich sicher ist, ob noch WAV-Sounds in der Warteschlange sind.

Syntax: **StopWavBG**

Beispiel:

```
PlayWavBG "TATA.WAV"
PlayWavBG "CHORD.WAV"
PlayWavBG "TRUMPET.WAV"
PlayWavBG "PIANO.WAV"
StopWavBG
PlayWavBG "BOING.WAV"
```

Beim Aufruf von StopWavBG spielt die Datei TATA.WAV im Hintergrund. CHORD.WAV, TRUMPET.WAV und PIANO.WAV werden aus der Warteschlange entfernt. TATA.WAV wird zu Ende gespielt, danach wird BOING.WAV gespielt.

GetWavInfo

Mit GetWavInfo können Sie verschiedene Informationen über eine WAV-Datei erhalten. GetWavInfo liefert Null wenn die übergebene Datei keine WAV-Datei ist oder nicht geöffnet werden kann. Im ersten Fall enthält die globale Variable fileError den Wert Null, im anderen Fall einen Fehlercode.

Syntax: **<numVar>** = **GetWavInfo** (**"fileName"**, **info**)

"fileName" Name der zu untersuchenden WAV-Datei. Pfadangaben sind zulässig.

info: Numerische Konstante entsprechend der Tabelle unten

Konstante	Wert	Information
WI_FORMAT	0	WAV-Format. Unterstützte Formate: 1: PCM 6: A-LAW
WI_CHANNELS	1	Anzahl der Kanäle (1=Mono, 2=Stereo)
WI_SAMPLE_DEEPTH	2	Bytes pro Sample (1: 8 Bit Mono, 2: 8 Bit Stereo oder 16 Bit Mono ...)
WI_SAMPLING_RATE	3	Anzahl Samples pro Sekunde (Samplingfrequenz in Hz)
WI_PLAY_TIME	4	Spielzeit in Sekunden

ConvertWav

ConvertWav fertigt eine Kopie einer Wav-Datei an und konvertiert sie dabei ins Format 8 Bit Mono. Nur dieses Format kann von GEOS abgespielt werden. Es ist daher sinnvoll alle WAV-Dateien, die Sie in ein Installationspaket packen wollen, vorher nach 8 Bit Mono zu konvertieren. ConvertWav erspart Ihnen die Verwendung eines Fremdprogramms. Es kopiert auch 8 Bit-Mono-Dateien.

Prüfen Sie die globale Variable "fileError" bzw. spielen Sie die Kopie ab, um zu sehen, ob das Konvertieren erfolgreich war.

Syntax: **ConvertWav "source.wav", "dest.wav"**

"source.wav"	Zu konvertierende Datei. Pfadangaben sind zulässig.
"dest.wav"	Zieldatei. Pfadangaben sind zulässig. Falls die Datei schon existiert wird sie ohne Warnung überschrieben.

Hinweis: Es ist nicht sinnvoll, vor jedem Abspielen einer WAV-Datei ihr Format zu prüfen und falls notwendig ConvertWav zu rufen. Die PlayWav-Routinen rufen ConvertWav selbständig, falls es erforderlich ist. Und sie sind dabei wesentlich effizienter als jeder BASIC Code.

2.10.3 Ausgabe von FM-Sounds

Neben der Ausgabe von gesampelter Musik (z.B. der Ausgabe von WAV-Dateien) können Computer auch Musikinstrumente simulieren. Das dazu häufig verwendete Verfahren heißt Frequenz-Modulation (kurz: FM). Die so erzeugten Töne oder Tonfolgen werden daher auch als FM-Sounds bezeichnet. Berühmt wurde dieses Verfahren, weil es in den erfolgreichen SoundBlaster™ Soundkarten verwendet wurde.

2.10.3.1 Beschreibung von Instrumenten und Noten

Dieser Abschnitt enthält die Grundlagen für die folgenden Kapitel (Spielen von einzelnen Noten und Spielen von FM-Musik).

Instrumente

Wie Sie wissen klingt ein bestimmter Ton, z.B. ein tiefes C, auf jedem Instrument anders. Das liegt im Wesentlichen an zwei Dingen:
Einerseits spielen die Anzahl und Amplitude der Oberwellen eine Rolle. Neben der Grundfrequenz enthält jeder Ton noch die doppelte, dreifache, vierfache usw. Frequenz. Je nachdem, wie stark diese Oberwellen sind, klingt der Ton anders.
Der zweite Aspekt ist der Amplitudenverlauf. Zum Beispiel erreicht der Ton eines Saiteninstruments sehr schnell seine volle Lautstärke um dann sofort mehr oder weniger langsam abzuklingen. Bei einem Blasinstrument kann hingegen die volle Lautstärke des Tons über längere Zeit gehalten werden. Hört der Musiker auf zu blasen so klingt der Ton sehr schnell ab. So hat jedes Musikinstrument seine eigene Charakteristik.

Computer simulieren diese Eigenschaften häufig mit einem Verfahren, dass man Frequenzmodulation (FM) nennt. Die so erzeugten Töne oder Tonfolgen bezeichnet man deshalb als FM-Sounds. Dabei wird die Grundfrequenz des Tones in einem bestimmten (sehr schellen) Rhythmus verändert. Dadurch entstehen - je nach Stärke und Geschwindigkeit der Änderung - unterschiedlich viele und unterschiedlich starke Oberwellen. Zusätzlich kann man den Amplitudenverlauf simulieren. Auf diese Weise gibt es extrem viele Möglichkeiten ein Instrument zu simulieren.

Geos bietet Ihnen 175 vordefinierte Instrumente an. Jedes Instrument wird dabei durch eine Zahl (im Bereich von Null bis 174) beschrieben. In der R-BASIC Library "MusicValues" sind entsprechende Konstanten definiert. Sie finden diese auch im R-BASIC Anhang, Kapitel G. Die Namen der Konstanten sind direkt dem PC/GEOS SDK entnommen und sind an die englischen Bezeichnungen der Instrumente angelehnt.
R-BASIC selbst definiert die folgenden Konstanten. Diese Werte sind Kopien der Werte aus der Library, so dass Sie in vielen Fällen ohne das Einbinden der Library auskommen.

Instrument	Wert	Instrument	Wert
IP_GRAND_PIANO	0	IP_ACOUSTIC_BASS	32
IP_ELECTRIC_PIANO_1	4	IP_ORCHESTRA_HIT	55
IP_MARIMBA	12	IP_PAD_METALLIC	93
IP_DRAWBAR_ORGAN	16	IP_KALIMBA	108
IP_REED_ORGAN	20	IP_TAIKO_DRUM	116

Das Beispiel "R-BASIC\Beispiel\Sound\Instrument Demo" erlaubt Ihnen, die Instrumente auf einfache Weise auszuprobieren.

Die Instrumente mit den Nummern 0 bis 127 sind "normale" Instrumente, die jeden Ton spielen können. Die Instrumente mit den Nummern 128 bis 174 hingegen sind Percussion-Instrumente. Laut Dokumentation sollten Sie mit einer genau auf das Instrument angestimmten Frequenz gespielt werden, damit sie so klingen wie gewünscht. Konstanten für die entsprechenden Frequenzen finden Sie ebenfalls in der Library "MusicValues" sowie im Anhang G.
Die Erfahrung zeigt aber, dass viele Percussion-Instrumente auch gut klingen, wenn sie mit anderen Noten gespielt werden. Das oben genannte Beispiel macht davon Gebrauch. Leider bedürfen insbesondere die Percussion-Instrumente unter GEOS einer Überarbeitung. Vielleicht erreichen Sie ja gerade durch Veränderung der gespielten Note den gewünschten Effekt.

Percussion-Instrument	Wert	Standard-Frequenz bzw. Note
IP_ACOUSTIC_BASS_DRUM	128	247 Hz, Note h
IP_HI_BONGO	153	831 Hz, Note g'+ (gis2)
IP_HI_TIMBALE	158	523 Hz, Note c'
IP_OPEN_TRIANGLE	174	831 Hz, Note g'+ (gis2)

Da das Aufsuchen der Standard-Frequenz zu einem Percussioninstrument mühsam und fehleranfällig ist kann man den Routinen PlayNote und AllocNoteHan (siehe nächstes Kapitel) den Frequenzwert Null übergeben. Die Routinen suchen dann selbst den passenden Frequenzwert heraus. Das Gleiche gilt für die Routinen PlayMusic, PlayMusicBG und AllocMusicHan.

GEOS und damit R-BASIC kann einzelne Töne oder ganze Tonfolgen auf diese Weise abspielen. Wie das gemacht wird ist in den folgenden Kapiteln beschrieben.

Die Tonhöhe

Die Tonhöhe einer Note wird durch ihre Frequenz festgelegt. Dabei gelten die folgenden Regeln:

- Der Kammerton 'a' hat eine Frequenz von genau 440 Hz.
- Eine Oktave entspricht einem Frequenzfaktor von genau 2. Das heißt, das hohe C hat genau die doppelte Frequenz wie das mittlere C. Das hohe A hat also eine Frequenz von 880 Hz.
- Benachbarte Halbtöne unterscheiden sich in ihrer Frequenz um den Faktor $\sqrt[12]{2}$. Damit erreicht man bei 12 Halbtonschritten genau einen Faktor von 2, nämlich eine Oktave.

Die zu den einzelnen Tönen gehörenden Frequenzen finden Sie im R-BASIC Anhang G. Da die FM-Sound-Routinen ganze Zahlen erwarten, sind die Werte gerundet. Außerdem sind in der R-BASIC Library "MusicValues" entsprechende Konstanten definiert.

Weil es sehr mühsam und auch nicht üblich ist, Musikstücke durch die Aufzählung der Frequenzen der Töne zu beschreiben, bietet R-BASIC die Routinen PlayMusic, PlayMusicBG und AllocMusicHan, denen man nur die Noten als String (z.B. "cdefgggg") übergeben muss.

Der Notenwert

Dieser Begriff beschreibt, ob es sich z.B. um eine viertel, eine achtel oder eine ganze Note handelt.

Das Tempo

Neben der Tonhöhe und dem Instrument, mit dem eine Note gespielt werden soll, müssen Sie auch angeben, wie lang die Note ist. Dabei geht R-BASIC folgenden Weg:

- Wenn Sie eine einzelne Note spielen (Anweisungen PlayNote und AllocNoteHan) geben Sie die Dauer in Sekunden an. Für eine viertel Sekunde verwenden Sie also den Wert 0.25.
- Bei Musikfolgen (Anweisungen PlayMusic, PlayMusicBG und AllocMusicHan) geben Sie bei der Definition der Musikfolge an, ob es sich z.B. um Achtel-, Viertel oder ganze Noten handelt. Beim Anspielen des Musikstücks legen Sie fest, wie schnell das Stück gespielt werden soll, d.h. wie viele Millisekunden eine 1/128 Note lang ist. So können Sie das gleiche Musikstück unterschiedlich schnell abspielen.
- Beim Befehl SOUND ist die Zeiteinheit der Einzeltöne wieder Sekunden, wobei Bruchteile von Sekunden (z.B. 0.1) erlaubt sind.
- Der Kompatibilitätsbefehl KCSound verwendet als Zeiteinheit 1/50 Sekunde.

Die Lautstärke

Töne oder Musikfolgen können in unterschiedlicher Lautstärke abgespielt werden. Die Lautstärke wird in R-BASIC grundsätzlich als Prozentwert von der Maximallautstärke, also im Bereich von 0 bis 100, angegeben. R-BASIC simuliert das physiologische Lautstärkeempfinden. Das heißt, mit dem Wert 50 klingt jede Note nur etwa halb so laut wie mit dem Wert 100. Das genaue Resultat hängt jedoch auch vom gewählten Instrument ab. Außerdem kann es sein, das bei einem Lautstärkewert von Null der Ton trotzdem ganz leise zu hören ist.

FM-Musik

Der Parameter musicString für die Befehle PlayMusic, PlayMusicBG und AllocMusicHan ist eine Zeichenkette, die genau beschreibt, welche Note, wie lange und mit welchem Instrument gespielt werden soll. Dabei gelten folgende Grundregeln:

- Defaultwert für das zu spielende Instrument ist die Nummer 13 (Xylophon).
- Der Default-Notenwert ist 1/4 Note.
- Die Groß- bzw. Kleinschreibung der Zeichen wird berücksichtigt!
- Zeichen, die nicht in der untenstehenden Tabelle vorkommen, werden ignoriert. Sie können Sie zur Strukturierung Ihres Notenstrings einsetzen. Insbesondere gilt das für **Leerzeichen** - sie bewirken **keine** Pause!

Die folgenden Zeichen können in einem Musicstring vorkommen. Ein X hinter einem Zeichen bedeutet, dass hier eine Zahl (1 bis 3 Ziffern) folgen muss. Vor oder innerhalb dieser Zahlen sind keine Leerzeichen zugelassen.

Zeichen	Bedeutung
c, d, e, f, g, a, h	Noten in der normalen Tonhöhe
C, D, E, F, G, A H	Noten, eine Oktave tiefer
+, –, ’, ., ~	Modifier-Zeichen, die hinter eine Note gesetzt werden können. Es sind mehrere Modifier erlaubt und sie können auch mehrfach auftreten. + Note einen halben Ton höher spielen. – Note einen halben Ton tiefer spielen. ’ Note eine Oktave höher spielen . Note um den halben Notenwert verlängern ~ Note Staccato spielen. (1)
#X	Explizite Angabe der zu spielenden Frequenz. Als Modifierzeichen sind "." und "~" erlaubt. (2)
o+, o–, o=	Oktave höher (o+) oder niedriger (o–) anwählen. Alle folgenden Noten werde eine Oktave höher bzw. tiefer gespielt. o= wechselt in die Standard-Oktave.
iX iX~	Auswahl des Instruments. Defaultwert ist i13 (Xylophon). Sie können hier nicht die IP_-Konstanten verwenden, sondern müssen die Nummer explizit angeben. Der Modifier ~ (z.B. "i16~") bewirkt, dass alle Noten für dieses Instrument Staccato gespielt werden (1).
%X	Ändert den Notenwert. Erlaubte Werte sind %1 (Ganze Note) bis %128 (1/128 Note). Defaultwert ist %4 (1/4 Note).
_	Der Unterstrich _ erzeugt eine Pause. Die Länge der Pause entspricht dem aktuellen Notenwert. Als Modifier ist ’.’ erlaubt.
vX, VX	Volume. Ändert die Lautstärke, in der das Stück gespielt wird. Erlaube Werte sind v0 (0%, lautlos) bis v100 (100%, maximale Lautstärke).

*	Das Sternchen * ist der Marker für die 2. Stimme (Hintergrundstimme).
*vX	Ändert die Lautstärke der Hintergrundstimme. Erlaube Werte sind v0 bis v100.
*iX	Ändert das Instrument für die Hintergrundstimme. Defaultwert ist i128 (Acoustic Bass Drum).
*c, *d , ...	Note für die Hintergrundstimme. Als Modifier-Zeichen sind +, – und ’ erlaubt. (3)
*C, *D , ...	Note für die Hintergrundstimme, eine Oktave tiefer. Als Modifier-Zeichen sind +, – und ’ erlaubt. (3)
*#X	Explizite Angabe der zu spielenden Frequenz für die Hintergrundstimme. (2), (3)
*o+, *o-, *o=	Oktave höher (o+) oder niedriger (o–) für die Hintergrundstimme anwählen. o= wechselt in die Standard-Oktave.
*X	Legt fest, über wie viele Noten der Vordergrundstimme die Hintergrund-Noten gespielt werden sollen. Der Standardwert ist "*1". (4)

Anmerkungen

(1) Staccato: Die Note wird deutlich verkürzt gespielt. Es folgt eine kurze Pause, so dass der Notenwert (Zeit zum Spielen einer Note) insgesamt nicht geändert wird.

(2) Um Percussion-Instrumente automatisch mit der für sie korrekten Frequenz zu spielen können Sie den Frequenzwert Null angeben ("#0").

(3) Eine mit einem Sternchen eingeleitete Hintergrundnote (oder explizite Frequenzangabe) wird nicht sofort, sondern gemeinsam mit der nächsten Vordergrundnote, Frequenzangabe oder Pause gespielt. Die Hintergrundnote wird dabei genauso lange wie die Vordergrundnote gespielt.

(4) Wenn Sie hier z.B. den Wert 3 angeben ("*3"), so werden im Gegensatz zu Anmerkung (3) alle folgenden Hintergrundnoten so lange gespielt, wie 3 Vordergrundnoten dauern.
Tipp: Damit Sie den Unterschied hören, müssen Sie für den Hintergrund ein Instrument verwenden, das den Ton hält (z.B. die Instrumente 16 bis 20).

Beispiele:

```
PlayMusic "cde_f~g~h~c’.", 7
```

```
PlayMusic "i20 c.f~ e%2f_ %4d~g.", 7
```

Beispiel mit Hintergrundnoten:
Die Hintergrundnoten c und f werden gemeinsam mit den Vordergrundnoten e und h gespielt. Vorher wird das Instrument Nr. 3 als Hintergrundinstrument ausgewählt.

```
PlayMusic "*i3 *cedc *fhag", 10
```

2.10.3.2 Spielen von einzelnen Noten

Bevor Sie diesen Abschnitt lesen, sollten die das vorherige Kapitel (Beschreibung von Noten und Instrumenten) gelesen haben.

Um eine Note zu spielen müssen Sie folgendes angeben:
- Das Instrument, mit die die Note gespielt werden soll. GEOS bieten Ihnen 175 Instrumente, davon sind 47 Percussions. In der R-BASIC Library "MusicValues" sind entsprechende Konstanten definiert. Eine entsprechende Liste finden Sie im Anhang G.
- Die Note, die gespielt werden soll. Noten werden durch ihre Frequenz bestimmt. In der R-BASIC Library "MusicValues" sind entsprechende Konstanten definiert. Eine entsprechende Liste finden Sie im Anhang G.
- Die Länge der Note. Das ist die Zeit, für die die Note "gespielt" werden soll. Für einzelne Noten muss unter R-BASIC die Zeit in Sekunden angegeben werden. Eine mit moderatem Tempo gespielte Viertelnote dauert etwa 0.5 s.
- Die Lautstärke. Unter R-BASIC wird die Lautstärke im Bereich von Null bis 100 angegeben. Ein Lautstärkewert von 50 bis 70 ist oft eine gute Wahl.
- Soll die Note im Hintergrund gespielt werden (R-BASIC arbeitet weiter während die Note erklingt) oder soll R-BASIC Warten, bis die Note verklungen ist, bevor das Programm weiterarbeitet.

Intern passiert beim Spielen einer Note folgendes:
- Es wird eine Datenstruktur angelegt, in der die Informationen über die zu spielende Note gespeichert sind.
- Die Note wird gespielt. Das kann auch mehrfach geschehen.
- Die Datenstruktur wird wieder freigegeben. Dazu muss das Spielen der Note vorher sicher beendet sein. Notfalls muss das Spielen der Note abgebrochen werden. R-BASIC erledigt dies automatisch.

Moderne Hardware kann meist mehr als eine Note (oder FM-Musikstück) gleichzeitig spielen. Wenn Sie mehrere Noten im Hintergrund spielen, werden diese wirklich gleichzeitig gespielt. Eine Warteschlange, wie bei WAV-Sounds, existiert nicht.

Zum Spielen einzelner Noten stehen Ihnen die folgenden Befehle zur Verfügung:

Anweisung	Aufgabe
PlayNote instr, freq, time, laut [, delay]	Spielt eine einzelne Note im Vordergrund oder im Hintergrund ab
han = AllocNoteHan (instr, freq, laut)	Speichert die Daten einer Note und liefert ein Handle darauf
PlayNoteHan han , time [, delay]	Spielt eine einzelne Note im Vordergrund oder im Hintergrund ab
StopNoteHan han	Stoppt das Abspielen einer mit PlayNoteHan im Hintergrund gespielten Note
FreeNoteHan han	Gibt die von AllocNoteHan angelegte Datenstruktur frei

Die Anweisung **PlayNote** spielt eine einzelne Note. Der Parameter "delay" (engl: Verzögerung) bestimmt dabei, wie die Note gespielt werden soll. Wird Delay nicht angegeben so wird die Note im Hintergrund abgespielt.

Wert von Delay	Wirkung
delay < 0	Die Note wird im Hintergrund abgespielt.
delay = 0	Die Note wird im "Vordergrund" gespielt. Das heißt, R-BASIC wartet genau die Zeit, die durch den Parameter "time" bestimmt wird, dann wird die Wiedergabe der Note beendet und R-BASIC arbeitet weiter.
delay > 0	Die Note wird im "Vordergrund" gespielt. R-BASIC wartet um die durch "delay" angegebene Zeit (in Sekunden) länger, als durch "time" angegeben, bevor die Wiedergabe der Note abgebrochen wird. Dadurch können die Noten "ausklingen", was bei vielen Instrumenten sinnvoll ist.

PlayNote ist sehr einfach zu benutzen. Allerdings müssen Sie vorher wissen, wie lange die Note gespielt werden soll, denn Sie haben nicht die Möglichkeit, das Spielen der Note vorzeitig abzubrechen. Wenn Sie z.B. ein Piano programmieren wollen ist jedoch genau das nötig. Die Note soll beim Drücken einer Taste erklingen und beim Loslassen wieder enden.
Deswegen bietet R-BASIC die Kommandos **AllocNoteHan**, **PlayNoteHan**, **StopNoteHan** und **FreeNoteHan**. AllocNoteHan legt die notwendige Datenstruktur an und liefert ein Handle darauf. Für ein Piano-Programm könnten Sie z.B. folgendes tun: Wenn der Nutzer eine Taste drückt rufen Sie PlayNoteHan und der Ton erklingt. Den Parameter "time" setzen Sie z.B. auf 600 (10 Minuten), damit der Ton nicht "von allein" aufhört. Lässt der Nutzer die Taste wieder los so rufen Sie StopNoteHan. Wenn Sie die Note sicher nicht mehr brauchen, müssen Sie FreeNoteHan rufen, um die Datenstruktur wieder freizugeben.

PlayNote

Spielt eine einzelne Note im Vordergrund oder im Hintergrund ab.

Syntax: **PlayNote** **instr**, **freq**, **time**, **laut** [, **delay**]

instr: Numerischer Wert im Bereich von 0 bis 174, der das Instrument beschreibt. Die Werte 0 bis 127 sind "normale" Instrumente, die Werte 128 bis 174 sind Percussions.
freq: Frequenz der zu spielenden Note
Empfehlung: Verwenden Sie den Wert Null für die Frequenz um Percussions automatisch mit der korrekten Frequenz zu spielen.
time: Länge der Note in Sekunden. Der Maximalwert für time ist 65 000, das entspricht ca. 18 Stunden.

laut: Lautstärke der Note im Bereich von 0 bis 100. R-BASIC simuliert das physiologische Lautstärkeempfinden. Das heißt, mit dem Wert 50 klingt die Note nur etwa halb so laut wie mit dem Wert 100. Das genaue Resultat hängt jedoch auch vom gewählten Instrument ab.

delay: Bestimmt, wie die Note gespielt wird.
- delay < 0: (Defaultwert) Die Note wird im Hintergrund gespielt.
- delay >= 0: R-BASIC wartet bis die Note gespielt wurde. Die Wartezeit beträgt "time+delay" Sekunden, wobei Bruchteile von Sekunden (z.B. 0.2 s) zugelassen sind. Verwenden Sie delay > 0 um die Note ausklingen zu lassen.

Beispiele:

```
PlayNote   IP_MARIMBA, 262, 0.7, 100
```

Die Note 'c' (262 Hz) wird auf dem Instrument MARIMBA für 0.7 Sekunden mit 100% Lautstärke im Hintergrund gespielt. Das heißt, R-BASIC führt den nächsten Befehl aus, noch während die Note erklingt.

```
PlayNote   IP_REED_ORGAN, 440, 0.6, 80, 0.4
```

Die Note 'a' (440 Hz) wird auf dem Instrument REED_ORGAN für 0.6 Sekunden mit 80% Lautstärke gespielt. R-BASIC wartet eine Sekunde (0.6 + 0.4 = 1) bevor der nächste Befehl abgearbeitet wird.

```
PlayNote   IP_ACOUSTIC_BASS_DRUM, 0, 0.2, 100
```

Das Percussion-Instrument IP_ACOUSTIC_BASS_DRUM wird mit der passenden Frequenz (automatische Auswahl durch den Frequenzwert Null) für 0,2 Sekunden mit voller Lautstärke im Hintergrund gespielt.

AllocNoteHan

Speichert die Daten einer Note in einer internen Datenstruktur und liefert ein Handle darauf.

Syntax: **<hanVar>** = **AllocNoteHan** (**instr**, **freq**, **laut**)

<hanVar>: Variable vom Typ Handle. Das Handle kann an PlayNoteHan, StopNoteHan und FreeNoteHan übergeben werden. Sie müssen das Handle nach Gebrauch mit FreeNoteHan wieder freigegeben.

instr, freq, laut: Siehe PlayNote
Beachten Sie, dass hier keine Zeit angegeben wird. Die Zeit, wie lange die Note erklingen soll, wird an PlayNoteHan übergeben.

PlayNoteHan

Spielt eine einzelne Note im Vordergrund oder im Hintergrund ab. Die Note wird durch ein Handle beschrieben, das von AllocNoteHan geliefert wurde.

Syntax: **PlayNoteHan** **han** , **time**, [, **delay**]

han:	Handle, das von AllocNoteHan geliefert wurde.
time:	Länge der Note in Sekunden. Der Maximalwert für time ist 650, das entspricht knapp 11 Minuten.
delay:	Siehe PlayNote Defaultwert für delay: -1 (Note im Hintergrund abspielen)

StopNoteHan

Stoppt das Abspielen einer mit PlayNoteHan im Hintergrund gespielten Note. StopNoteHan darf auch gerufen werden, wenn die durch "han" spezifizierte Note nicht oder nicht mehr spielt.

Syntax: **StopNoteHan** **han**

han:	Handle, das von AllocNoteHan geliefert wurde.

FreeNoteHan

Gibt die von AllocNoteHan angelegte Datenstruktur frei.

Syntax: **FreeNoteHan** **han**

han:	Handle, das von AllocNoteHan geliefert wurde.

Beispiel:

```
Dim h as HANDLE
  h = AllocNoteHan (IP_REED_ORGAN, 550, 100)
  PlayNoteHan h, 600      ' Ton an (10 min.)
  Pause 20                ' 2 Sekunden
  StopNoteHan h           ' Ton vorzeitig aus

  Pause 10                ' 1 Sekunde

  PlayNoteHan h, 2, 0     ' Ton an (2 Sekunden, im Vordergrund)
  StopNoteHan h           ' Ton aus. Zur Sicherheit
  FreeNoteHan h           ' Fertig
```

2.10.3.3 Ausgabe von FM-Musik

Bevor Sie diesen Abschnitt lesen, sollten die das Kapitel 2.10.3.1 (Beschreibung von Noten und Instrumenten) gelesen haben.
In der Datei "R-BASIC\Beispiel\Sound\FM Musik Demo" finden Sie mehrere Beispiele zur Verwendung der FM-Musik-Routinen.

Zum Spielen eines FM-Musikstücks stehen Ihnen die folgenden Befehle zur Verfügung:

Anweisung	Aufgabe
PlayMusic "noten", tempo [, delay]	Spielt ein FM-Musikstück im Vordergrund oder im Hintergrund ab.
PlayMusicBG "noten", tempo	Spielt ein FM-Musikstück im Hintergrund ab, wobei eine Warteschlange organisiert wird.
<numVar> = GetMusicBGCount()	Liest die Anzahl der FM-Musikstücke in der Hintergrundwarteschlange
StopMusicBG	Leert die Hintergrundwarteschlange für FM-Musikstücke
<hanVar> = AllocMusicHan ("noten")	Speichert die Daten eines FM-Musikstücks und liefert ein Handle darauf.
PlayMusicHan han , tempo [, delay]	Spielt ein FM-Musikstück im Vordergrund oder im Hintergrund ab
StopMusicHan han	Stoppt das Abspielen eines mit PlayMusicHan im Hintergrund gespielten FM-Musikstücks.
<numVar> = GetMusicTime (han)	Ermittelt die Spielzeit für ein FM-Musikstück.
FreeMusicHan han	Gibt die von AllocMusicHan angelegte Datenstruktur frei.

PlayMusic ist sehr einfach zu benutzen. Allerdings haben Sie nicht die Möglichkeit, das FM-Musikstück vorher abzubrechen oder mehrere Musikstücke nacheinander im Hintergrund zu spielen.
Die Kommandos **AllocMusicHan**, **PlayMusicHan**, **StopMusicHan** und **FreeMusicHan** lösen das erste Problem. AllocMusicHan legt die notwendige Datenstruktur an und liefert ein Handle darauf. Zum Spielen des FM-Musikstücks rufen Sie PlayMusicHan, um das Spielen vorzeitig abzubrechen, rufen Sie StopMusicHan. Mit FreeMusicHan geben Sie die von AllocMusicHan belegten Datenstrukturen wieder frei.
Um mehrere FM-Musikstücke im Hintergrund nacheinander zu spielen rufen Sie mehrfach hintereinander **PlayMusicBG**. R-BASIC organisiert automatisch eine Warteschlange, so dass die FM-Musikstücke automatisch nacheinander angespielt werden. Mit **GetMusicBGCount** und **StopMusicBG** haben Sie Zugriff auf diese Warteschlange.

PlayMusic

Spielt ein FM-Musikstück im Vordergrund oder im Hintergrund ab.

Syntax: **PlayMusic** **"musicString"** , **tempo** [, **delay**]

"musicString": Zeichenfolge die angibt, welche Töne gespielt werden und welche Instrumente verwendet werden sollen.
Beschreibung siehe Kapitel 2.10.3.1.

tempo: Bestimmt die Geschwindigkeit, mit der das Musikstück gespielt wird. Tempo gibt an, wie viele Millisekunden eine 1/128 Note dauert. Ein guter Startwert für eigene Musikstücke ist 15. Das entspricht etwa der Metronomeinstellung "120 Schläge pro Minute".

delay: Bestimmt, wie das Musikstück gespielt wird.

- delay < 0: (Defaultwert) Das Musikstück wird im Hintergrund gespielt.
- delay >= 0: R-BASIC wartet bis das Musikstück gespielt wurde. Die Wartezeit ist um "delay" Sekunden größer, als das Musikstück selbst dauert, wobei Bruchteile von Sekunden (z.B. 0.2 s) zugelassen sind.
  Verwenden Sie delay > 0 um die letzten Töne ausklingen zu lassen.

Beispiele:

```
PlayMusic "cdefgahc'", 15
                      ' Tonleiter im Hintergrund spielen
```

```
PlayMusic "cdefgahc'", 15, 0        ' Im Vordergrund spielen
```

```
PlayMusic "i0c i4c i32c i108c", 15, 0
                                    ' Mehrfach Note c spielen
```

Beispiel: Für Percussion-Instrumente (Instrumentennummer >= 128) kann man die "Standard"-Frequenz automatisch auswählen lassen, wenn man als Frequenz den Wert Null übergibt.

```
PlayMusic "i128#0 i158#0_ i128#0#0", 14
```

Hinweis: Die Erfahrung zeigt, dass andere Noten als die "Standard-Frequenz" bei Percussion-Instrumenten ebenfalls gut klingen. Im Einzelfall sollten Sie das ausprobieren. Das Beispiel "R-BASIC\Beispiel\Sound\FM Musik Demo" macht für die Hintergrundstimme davon Gebrauch. Auch das Beispiel "R-BASIC\Beispiel\Sound\Instrument Demo" verwendet diese Variante.

PlayMusicBG

Spielt ein FM-Musikstück im Hintergrund ab, wobei eine Warteschalge organisiert wird. Die an PlayMusicBG übergebene Musikfolge wir erst gespielt, wenn alle vorher an PlayMusicBG übergebene Musikstücke abgespielt wurden.
Hinweis: Der Sound-Befehl (Kapitel 2.10.1) verwendet die gleiche Hintergrund-warteschlange.

Syntax: **PlayMusicBG** **"musicString"** , **tempo**

"musicString": Zeichenfolge die angibt, welche Töne gespielt werden und welche Instrumente verwendet werden sollen.
Beschreibung siehe Kapitel 2.10.3.1.
tempo: Bestimmt die Geschwindigkeit, mit der das Musikstück gespielt wird. Tempo gibt an, wie viele Millisekunden eine 1/128 Note dauert. Ein guter Startwert für eigene Musikstücke ist 15. Das entspricht etwa der Metronomeinstellung "120 Schläge pro Minute".

Beispiel.
Zuerst wird die Tonfolge "cdef" und dann, mit geringerer Geschwindigkeit, die Tonfolge "gahc' " abgespielt. Der Text "Musik spielt" erscheint noch während die Tonfolgen zu hören sind.

```
PlayMusicBG    "cdef", 12
PlayMusicBG    "gahc'", 20
Print "Musik spielt"
```

GetMusicBGCount

Diese Funktion liefert die Anzahl der FM-Tonfolgen, die in der Wartschlange sind. Dabei wird die aktuell spielende FM-Tonfolge mitgezählt. Der Rückgabewert 1 bedeutet also, dass gerade eine FM-Tonfolge im Hintergrund spielt, die Warteschlange aber leer ist.

Syntax: **<numVar>** = **GetMusicBGCount** ( )

Hinweis: Der Sound-Befehl (Kapitel 2.10.1) verwendet die gleiche Hintergrund-warteschlange wie der Befehl PlayMusicBG. GetMusicBGCount zählt also auch Sound-Anweisungen, die in der Warteschlange sind.

StopMusicBG

StopMusicBG löscht die Warteschlange der für Ausgabe von FM-Tonfolgen verantwortlichen Library. Der aktuell im Hintergrund spielende FM-Tonfolge wird nicht abgebrochen. StopMusicBG darf auch gerufen werden, wenn gerade keine FM-Tonfolge spielt. Es wird empfohlen, StopMusicBG im OnExit-Handler zu rufen, wenn man sich sicher ist, ob noch FM-Tonfolgen in der Warteschlange sind.

Syntax: **StopMusicBG**

Hinweis: Der Sound-Befehl (Kapitel 2.10.1) verwendet die gleiche Hintergrund-warteschlange wie der Befehl PlayMusicBG. Sie können also den Befehl StopMusicBG auch verwenden, um die Warteschlange für Sound-Befehle zu leeren.

Beispiel:

```
PlayMusicBG "cdefgahc'_", 7
PlayMusicBG "cdefgahc'_", 7
StopMusicBG
PlayMusicBG "c~c~c~d~e~c~", 7
```

Beim Aufruf von StopMusicBG spielt die erste Tonleiter. Sie spielt zuende. Die zweite Tonleiter wird aus der Warteschlange entfernt. Stattdessen spielt sofort die Tonfolge "c~c~c~d~e~c~".

AllocMusicHan

Speichert die Daten eines FM-Musikstücks in einer internen Datenstruktur und liefert ein Handle darauf. Das Musikstück kann später mit PlayMusicHan abgespielt werden. Sie müssen das Handle nach Gebrauch mit FreeMusicHan wieder freigegeben.

Syntax: **<hanVar>** = **AllocMusicHan** (**"musicString"**)

<hanVar>: Variable vom Typ Handle. Das Handle kann an PlayMusicHan, StopMusicHan und FreeMusicHan übergeben werden.

"musicString": Siehe PlayMusic
Beachten Sie, dass hier kein Tempo angegeben wird. Das tempo wird bei PlayMusicHan angegeben.

PlayMusicHan

Spielt ein Musikstück im Vordergrund oder im Hintergrund ab. Das Musikstück wird durch ein Handle beschrieben, das von AllocMusicHan geliefert wurde.

Syntax: **PlayMusicHan** **han** , **tempo**, [, **delay**]

han: Handle, das von AllocMusicHan geliefert wurde.
tempo: Siehe PlayMusic
delay: Siehe PlayMusic
Defaultwert für delay: -1 (Musik im Hintergrund abspielen)

StopMusicHan

Stoppt das Abspielen einer mit PlayMusicHan im Hintergrund gespielten Musik. StopMusicHan darf auch gerufen werden, wenn das durch "han" spezifizierte Musikstück nicht oder nicht mehr spielt.

Syntax: **StopMusicHan** **han**
han: Handle, das von AllocMusicHan geliefert wurde.

FreeMusicHan

Gibt die von AllocMusicHan angelegte Datenstruktur frei.

Syntax: FreeMusicHan han
han: Handle, das von AllocMusicHan geliefert wurde.

Beispiel:
Die Tonfolge "eefggfed_" wird mehrfach, aber mit verschiedener Geschwindigkeit, im Vordergrund abgespielt. Als Musikinstrument wird "Grand Piano" (Nummer 0) verwendet.

```
DIM th1$
DIM h as handle

  th1$ = "eefggfed_"
  h = AllocMusicHan ( "i0" + th1$)
  PlayMusicHan h, 14, 0
  PlayMusicHan h, 10, 0
  PlayMusicHan h, 5, 0
  StopMusicHan h
  FreeMusicHan h
```

GetMusicTime

GetMusicTime ermittelt die Spielzeit eines FM-Musikstücks in Sekunden, wenn es mit dem angegebenen Tempo-Wert gespielt wird.
Die Funktion GetMusicTime liefert immer die Zeit, die das Musikstück insgesamt spielt. Es ist egal, ob das Musikstück bereits zur Hälfte gespielt ist oder ob es überhaupt schon spielt.

Syntax: **<numVar>** = **GetMusicTime** (**han**, **tempo**)
han: Handle, das vom AllocMusicHan geliefert wurde.
tempo: Siehe PlayMusic / PlayMusicHan

```
DIM mh as Handle
DIM longMusic$, t

  longMusic$ =  .... ' sehr viel Musik
  mh = AllocMusicHan ( longMusic$ )
  PlayMusicHan mh, 14
  PRINT "Sie hören jetzt "; GetMusicTime(mh); " Sekunden Musik."
```

2.10.4 Konfiguration der Soundlibrary

Die Ausgabe von WAV- und FM-Sound wird von einer Library, der RabeSoft-Audio-Library, übernommen. Je nach Systemversion und nach installiertem Sound-Treiber kann die WAV- und FM-Sound Ausgabe möglich sein oder nicht. Zum Beispiel spielt die Sound.geo von GEOS 3.2 Wav-Dateien hörbar zu kurz ab. Die RS-Audio-Library verfügt sowohl über die Möglichkeit, diesen Fehler auszugleichen, als auch (durch Befragen des Nutzers) zu testen, ob dieser Ausgleich nötig ist. Deshalb bietet R-BASIC an, den entsprechenden Konfigurations-Dialog aufzurufen als auch den Konfigurationsstatus abzufragen.

Sie können mithilfe der hier beschriebenen Befehle herausbekommen, ob auf dem System des Nutzers z.B. keine WAV-Dateien abgespielt werden können. Sie können dann stattdessen einen FM-Sound abspielen.

Hinweise:
- Es ist nicht zwingend erforderlich, dass Sie die hier beschrieben Kommandos DoAudioConfig und GetAudioConfig in ihrem Programm verwenden.
- Wenn Sie die Soundausgabe auf einem System verwenden, dass keine Soundausgabe unterstützt, so wird eben kein Sound gespielt. Systemabstürze sind nicht zu erwarten.
- Der Nutzer kann durch eine (fehlerhafte) Konfiguration der Library nicht verhindern, dass Sounds abgespielt werden.

DoAudioConfig

DoAudioConfig ruft den Konfigurationsdialog der RS-Audio-Library auf. Der User hat hier die Möglichkeit, zu testen, ob die WAV und die FM-Ausgabe funktioniert. Die Library merkt sich die Einstellungen in der GEOS.INI unter [Rabe-Soft diverses] "Audio".

Syntax: **DoAudioConfig forceDialog**

forceDialog:	FALSE:	Der Dialog wird nur aufgerufen, wenn die Library noch nicht oder noch nicht vollständig konfiguriert ist.
	TRUE:	Der Dialog wird in jedem Fall aufgerufen.

Hinweis: Durch die Konfiguration der Library kann der User nicht verhindern, dass Sounds abgespielt werden. Er kann dem Programm nur mitteilen, ob bestimmte Sounds auf seinem System hörbar sind. Das bedeutet folgendes:
- Sollte der User in der Konfiguration z.B. angegeben haben, dass das System keine Wav-Ausgabe unterstützt, obwohl das System es tut, so wird PlayWav die Wav-Datei trotzdem abspielen. Das Gleiche gilt für FM-Sounds. Dieser Fall kann z.B. eintreten, wenn der Nutzer einen neuen, besseren Soundtreiber aktiviert.
- Wenn Sie die vom User eingestellte Konfiguration berücksichtigen wollen, so müssen Sie die Funktion GetAudioConfig aufrufen.
- Die Wav-Befehle berücksichtigen in jedem Fall, ob der Nutzer angegeben hat, dass zu kurz gespielte Wav-Dateien verlängert werden sollen. Das bedeutet: hat

der User angegeben, dass Wav-Dateien zu verlängern sind, obwohl dies nicht nötig ist, entsteht eine (kurze) Pause am Ende jeder Wav-Datei.

Empfehlung (siehe auch Beispieldatei "Beispiel\Sound\KeyClick Demo"):
Wenn Sie in Ihrem Programm die Ausgabe von WAV und/oder FM-Sound verwenden sollten Sie im OnStartup-Handler folgende Zeile einbauen:

```
  DoAudioConfig FALSE
```

Sie bewirkt, dass der Konfigurations-Dialog der RS-Audio-Library aufgerufen wird, falls die Library nicht oder nicht vollständig konfiguriert ist.
Außerdem sollten Sie einen Menüpunkt "Audio-Library konfigurieren" anbieten, dessen Actionhandler folgendermaßen aussieht:

```
Buttonaction ConfigureAudioLib
  DoAudioConfig TRUE
End Action
```

Damit hat der Nutzer die Möglichkeit, die Konfiguration der RS-Audio-Library nachträglich zu verändern, z.B. wenn er einen neuen Sound-Treiber ausprobiert.

GetAudioConfig

Die Funktion GetAudioConfig liefert die vom Nutzer angegebenen Konfigurationsdaten, d.h. ob das System Wav- und FM-Sound unterstützt und ob die RS-Audio-Library vollständig konfiguriert ist. Sollte die RS-Audio-Library nicht oder nicht vollständig konfiguriert sein, so nimmt GetAudioConfig an, dass die nicht konfigurierten / getesteten Funktionen unterstützt werden.
Sie können GetAudioConfig z.B. verwenden, um einen FM-Sound abzuspielen, falls die Wav-Ausgabe auf dem System des Users nicht unterstützt wird.

Syntax: **<numVar>** = **GetAudioConfig** ( )

Der Rückgabewert von GetAudioConfig sind Bitflags. Die folgenden Werte sind definiert. Hier nicht aufgeführte Bits sind reserviert!

Konstante	Wert	hex	Bedeutung
SC_WAV_SUPPORTED	1	&h01	Wav-Ausgabe wird unterstützt
SC_FM_SUPPORTED	2	&h02	FM-Sound wird unterstützt
SC_NEED_CONFIG	16	&h10	Die RS-Audio-Library ist nicht oder nicht vollständig konfiguriert.
SC_SOUND_IS_OFF	32	&h32	Die Tonausgabe ist in den Voreinstellungen deaktiviert

Hinweis:
- GetAudioConfig liefert die Konfigurationsdaten der RS-Audio-Library. Es führt selbst keinen Prüfungen durch.