REFERENZ

Hier finden Sie Syntax und Beispiele zu Systemvariablen, Funktionen und Befehlen von Profan.

- Systemvariablen
- Funktionen
- Befehle

Bitte wählen Sie das gwünschte Thema durch Anklicken mit der Maus aus.

Eine Einführung in die Programmierung mit PROFAN² finden Sie in der EINFÜHRUNG

Referenz Systemvariablen

!Input

$DosPath
$DosVer
$Drive
$GetInput
$GetText
$Input
$WinPath
$WinVer

%Button
%Error
%GetCount
%GetCurSel
%Input
%IOResult
%Key
%MCIError
%MenuItem
%MouseKey
%MouseX
%MouseY
%ParCount
%ScanKey
%WinTop
%WinBottom
%WinRight
%WinLeft

&Input

Referenz Funktionen

@! @$ @% @&

@Abs @Add$ @Add @And @AnsiToOem$ @ArcTan

@Chr$ @Cos

@Date$ @Del$ @DiskFree @DiskSize @Div& @Div

@Eof @Equ$ @Equ @Exp

@FindFirst$ @FindNext$

@GetClip$ @GetDir$ @GetKey$ @GetPixel @GT @GT$

@If @Inkey$ @Input$ @Ins$ @InStr @Int

@KeyIn

@Len @List! @List$ @List% @List& @ListBox$ @ListBoxItem$ @LoadFile$ @LT @LT$

@MCISend$ @MenuItem @MessageBox @Mid$ @Mod @Mouse @Mul

@NEq @NEq$ @Not

@OemToAnsi$ @Or @Ord

@Par$ @Pi

@ReadIni$ @RGB @Rnd

@SaveFile$ @ScanKey @Sin @Sqr @Sqrt @Str$ @Sub

@Time$ @TMouse @Trim$

@Upper$

@Val

Referenz Befehle

´(Apostroph)

Add AddFiles AddString Append AppendMenu AppendMenuBar Arc Assign

Beep

Case CaseNot CharSet ChDir Chord ClearClip ClearList Close Cls Color Copy CopyBmp CopySizedBmp CreateMenu

Dec Decimals Declare Def Dim! Dim$ Dim% Dim& DrawIcon DrawSysIcon DrawText

Ellipse Else ElseIf EndIf End EndPrint EndProc EndSub Erase

Fill Font

GoSub ... Return GoTo

If ... ElseIf ... Else ... EndIf IfNot ... Else ... EndIf Inc Input Input#

Let Line LineTo ListBoxItem$ List! List$ List% List& LoadBmp LoadSizedBmp Locate

MessageBox MkDir MoveTo Music

NumWidth

Orientation

Parameters Pie Play PlaySound PopUp Print Print# Proc ... EndProc PutClip

Randomize Rectangle Rem Rename RePaint Reset Return Rewrite RmDir RoundRect Run

SaveBmp ScreenCopy Separator SetErrorLevel SetPixel Shell ShowMin ShowMax ShowNormal Sound StartPrint ... EndPrint StrWidth Sub SubPopUp

TBox TextColor TrackMenu

UseBrush UseCursor UseFont UseIcon UsePen

WaitInput WaitKey WaitMouse WaitScan While ... Wend WhileNot ... Wend Window WindowStyle WindowTitle WriteIni

Systemvariablen

Systemvariablen geben Auskunft über Zustände des Systems, z.B. die aktuelle Position der Maus, das aktuelle Laufwerk, etc.

Es gibt vier Typen von Systemvariablen:

- $Name - Der Ergebnistyp ist ein String
- %Name - Der Ergebnistyp ist ein Integer
- &Name - Der Ergebnistyp ist ein LongInt
- !Name - Der Ergebnistyp ist ein Float

Funktionen

Eine Funktion gibt einen Wert zurück, der in der Regel von den Parametern der Funktion abhängig ist. Überall da, wo in PROFAN² Variablen oder konstante Werte (Literale) eingesetzt werden können, dürfen auch Funktionen stehen. Der Rückgabewert der Funktion sollte der gleiche Typ sein, wie der erwartete Wert.

Alle Funktionen in PROFAN² beginnen mit einem @. Dies ist deshalb, damit der Interpreter das Vorliegen einer Funktion schneller erkennen kann.

Funktionen, deren Funktionsnamen mit dem $ endet, geben in vielen Fällen einen String zurück, die übrigen Funktionen einen numerischen Wert. In einigen Fällen enden auch Vergeleichsfunktionen, die entweder 1 oder 0 zurückgeben mit einem $, um deutlich zu machen, das hiermit Strings verglichen werden.

Wenn der Rückgabewert einer Funktion uninteressant ist, kann die Zuweisung auch weggelassen werden. Bis auf die immer noch notwendigen Klammern, wird die Funktion dann wie ein Befehl verwandt.

Beispiel:

```
@LISTBOX$("Das ist die Liste",0)
```

In diesem Fall wird die Listbox nur zum Anschauen angezeigt oder aber das Ergebnis der Auswahl über die Systemvariable $GETTEXT gelesen.

Befehle

Jede Programmzeile beginnt mit einem Befehl, wobei auch Funktionen wie Befehle verwandt werden können. So darf also auf keinen Fall bei einer Zuweisung das LET weggelassen werden. Nach dem Befehlswort muß immer mindestens 1 Leerzeichen stehen, ansonsten wird der Befehl nicht als solcher erkannt.
In jeder Programmzeile darf auch immer nur ein Befehl stehen. Mehrere Befehle in einer Zeile (wie z.B. in BASIC, PASCAL oder C) sind nicht zulässig. Einige Ausnahme ist der CASE-Befehl.

Literale und Variablen

Literale sind direkt ausgeschriebene Werte, also Strings in Anführungszeichen oder Zahlen in Ziffern geschrieben. Beispiele für Literale:

```
"Basic ist eine Programmiersprache"
4.567
1234
234456
```

Variablen hingegen sind Platzhalter bzw. Speichernamen (Bezeichner) für Werte:

```
Text$
Ergebnis!
Anzahl%
GrosserWert&
```

Das letzte Zeichen (Postfix) gibt an, welchen Typ die Variable hat.

!Input - $Input - %Input - &Input

!Input

Der zuletzt mit Input eingegebene Float-Wert.

$Input

Der zuletzt mit Input eingegebene String.

%Input

Der zuletzt mit Input eingegebene Integer-Wert.

&Input

Der zuletzt mit Input eingegebene LongInt-Wert.

Systemvariablen
Übersicht

$DRive

Das aktuelle Laufwerk (ohne Pfadangabe).

Systemvariablen
Übersicht

$GetInput

Der zulezt mit der Funktion @Input$ eingegebene Text.

Systemvariablen
Übersicht

$GetText

Der zuletzt in einer Listbox (Funktion @ListBox$) ausgewählte Text.

Systemvariablen
Übersicht

%Button

Der in einer MessageBox gedrückte Knopf:

1 - OK
2 - Abbrechen (Cancel)
3 - Abbrechen (Abort)
4 - Wiederholen
5 - Ignorieren
6 - Ja
7 - Nein

Systemvariablen
Übersicht

%Error

Der zuletzt aufgetretene Fehlercode. Wie %IOResult wird %Error beim Auslesen wieder auf 0 gesetzt. Die Error-Werte:

0 - kein Fehler

1 - Warnung

2 - Fehler

Systemvariablen
Übersicht

%GetCount

Die Anzahl der Einträge in der ListBox-Liste. Beispiel:

```
ClearList
AddFiles "C:\WINDOWS\*.EXE"
Print %GetCount;" EXE-Dateien"
```

Systemvariablen
Übersicht

%IOResult

Nach einer Dateioperation oder einer Directory-Suche enthält diese Variable den entsprechenden Wert:

Die Werte entsprechenden den von Turbo-Pascal her bekannten Ergebnissen:

- 0 - kein Fehler aufgetreten
- 2 - Datei nicht gefunden
- 3 - Pfad nicht gefunden
- 5 - Zugriff verweigert (ReadOnly?)
- 12 - Ungültiger Dateimodus
- 15 - Laufwerksnummer unzuzulässig
- 16 - Verzeichnis kann nicht gelöscht werden (noch Dateien drin?)
- 17 - RENAME nicht über Laufwerksgrenzen möglich (siehe Rename)
- 18 - Kein weiterer Eintrag (bei @FindFirst/@FindNext)
- 100 - Lesefehler von Diskette/Platte
- 101 - Schreibfehler auf Diskette/Platte
- 102 - Dateinummer ist keiner Datei mit ASSIGN zugeordnet (siehe Assign)
- 103 - Datei nicht offen (Reset, Rewrite oder Append fehlt)
- 104 - Datei nicht zum Lesen geöffnet (Reset fehlt)
- 105 - Datei nicht zum Schreiben geöffnet (Rewrite oder Append fehlt)
- 106 - Falsches Format (bei Input #N,..)

Systemvariablen
Übersicht

%Key

Der ANSI-Code der zuletzt gedrückten Taste. Anwendung nach den Befehlen WAITKEY und WAITINPUT:

```
WaitInput

If @Equ(Key,90)
   Print "Du hast 'Z' gedrückt"
EndIf
```

Systemvariablen
Übersicht

%MCIError

Der zuletzt gemeldete Fehler beim Senden eines MCI-Strings zur Ansteuerung eines Multi-Media-Treibers (etwa CD, Soundkarte, MIDI-Gerät, etc.). Aussagekräftiger sind allerdings die von @MCISend$ zurückgelieferten Fehlermeldungen (in der Landessprache).

Systemvariablen
Übersicht

%MenuItem

Die Identifikationsnummer des zuletzt angewählten Menüpunktes. Wurde das Copyright-Zeichen angeklickt, ist die Nummer 254, wurde kein Menüpunkt angeklickt ist sie 0. Die Nummern 1 .. 253 können vom Programmierer verwandt werden.

Systemvariablen
Übersicht

%MouseKey

Die zuletzt gedrückte Maustaste:

0 - keine Taste gedrückt
1 - linke Maustaste
2 - rechte Maustaste

Die Position der Maus findet sich in %MouseX und %MouseY.

Systemvariablen
Übersicht

%MouseX - %MouseY

%MouseX

Die aktuelle X-Position der Maus.

%MouseY

Die aktuelle Y-Position der Maus.

Anwendung besonders nach den Befehlen WAITMOUSE und WAITINPUT.

Systemvariablen
Übersicht

%ParCount

Die Anzahl der beim Programmaufruf übergebenen Kommandozeilen-Parameter. Parameter 0 ist beim Interpreter der Interpreter selber, bei der Runtime das Runtime-Modul und bei einer EXE-Datei die EXE-Datei.
Parameter 1 ist bei Interpreter und Runtime das ausgeführte Programm, bei einer EXE-Datei nicht sinnvoll. Er muß das "!" sein. Weitere Hinweise siehe unter @PAR$.

Systemvariablen
Übersicht

%ScanKey

Der Scancode (virtueller Code) der zuletzt gedrückten Taste. Wichtige Scancodes:

16 - Shift
17 - Strg
27 - Esc
33 - BildHoch
34 - BildRunter
35 - Ende
36 - Pos1
37 - Links
38 - Hoch
39 - Rechts
40 - Runter
45 - Einfg
46 - Entf
112 - F1
... - ...
123 - F12

Systemvariablen
Übersicht

@!(N) - @$(N) - @%(N) - @&(N)

@!(N)

```
N : Integer (0 ... 8)

Ergebnis: Float
```

Der N-te übergebene Parameter in einer selbstdefinierten Funktion (oder Prozedur) als Float.

@$(N)

```
N : Integer (0 ... 8)

Ergebnis: String
```

Der N-te übergebene Parameter in einer selbstdefinierten Funktion (oder Prozedur) als String.

@%(N)

```
N : Integer (0 ... 8)

Ergebnis: Integer
```

Der N-te übergebene Parameter in einer selbstdefinierten Funktion (oder Prozedur) als Integer.

@&(N)

```
N : Integer (0 ... 8)

Ergebnis: LongInt
```

Der N-te übergebene Parameter in einer selbstdefinierten Funktion (oder Prozedur) als LongInt.

Der 0. Parameter ist nach Aufruf einer Prozedur oder einem Unterprogramm mit RETURN <Wert> zurückgegebene Wert.

Funktionen
Übersicht

@Abs(N)

```
N : Wert

Ergebnis: Wert
```

Absolutwert der Zahl N.

Funktionen
Übersicht

@Add$(S1,S2)

```
S1 : String
S2 : String

Ergebnis: String
```

Verknüpfung der Strings S1 und S2. S2 wird an S1 angehängt.

Funktionen
Übersicht

@Add(N1,N2)

```
N1 : Wert - Summand
N2 : Wert - Summand

Ergebnis: Wert
```

Die Summe von N1 und N2.

Funktionen
Übersicht

@And(N1,N2)

```
N1 : Wert
N2 : Wert

Ergebnis: Wert
```

Die Werte von N1 und N2 werden mit der AND-Funktion verknüpft. Das Ergebnis ist 0 wenn ein oder beide Werte 0 sind.

Vergleiche: @OR, @NOT

Funktionen
Übersicht

@AnsiToOem$(S) - @OemToAnsi$(S)

@AnsiToOem$(S)

```
S : String

Ergebnis: String
```

Der String S wird von ANSI-Code in den ASCII-Code (OEM) umgewandelt.

@OemToAnsi$(S)

```
S : String

Ergebnis: String
```

Der String S wird von ASCII-Code in den Ansi-Code umgewandelt.

Diese Umwandlungen betreffen insbesondere die deutschen Sonderzeichen. Der ASCII-Code wird unter DOS verwandt, der ANSI-Code unter Windows.

Funktionen
Übersicht

@ArcTan(N)

```
N : Wert (Winkel in Bogenmaß)

Ergebnis : Float
```

Der Arcustangens des Winkels N.

Funktionen
Übersicht

@Chr$(N)

```
N : Wert (0 .. 255)

Ergebnis: String (1 Zeichen)
```

Das Zeichen mit dem ANSI-Code N.

Funktionen
Übersicht

@Cos(N)

```
N : Wert (Winkel in Bogenmaß)

Ergebnis : Float
```

Der Cosinus des Winkels N.

Funktionen
Übersicht

@Del$(S,N1,N2)

```
S  : String
N1 : Wert - Position
N2 : Wert - Zeichenzahl

Ergebnis: String
```

Ab der Position N1 werden N2 Zeichen aus dem String entfernt.

Funktionen
Übersicht

@DiskFree(S)

```
S : String - Laufwerksbezeichnung

Ergebnis: LongInt
```

Das Ergebnis ist der freie Speicher auf Laufwerk S in kB. Ist das Ergebnis z.B. 430, so sind noch 430 kB frei. Ist das Laufwerk nicht vorhanden oder nicht lesbar, ist das Ergebnis 0. Beispiel:

```
Print @DiskFree("C:");" kB frei"
```

Funktionen
Übersicht

@DiskSize(S)

```
S : String - Laufwerksbezeichnung

Ergebnis: Wert
```

Das Ergebnis ist der Gesamtspeicher auf Laufwerk S in kB. Ist das Ergebnis z.B. 43000, so hat das Laufwerk 43000 kB bzw. 43 MB. Ist das Laufwerk nicht vorhanden oder nicht lesbar, ist das Ergebnis 0. Beispiel:

```
Print @DiskSize("C:");" kB"
```

Funktionen
Übersicht

@Div&(N1,N2) - @Div(N1,N2)

@Div&(N1,N2)

```
N1 : Wert
N2 : Wert

Ergebnis: Integer / LongInt
```

Ganzzahliges Ergebnis der Division von N1 durch N2.

@Div(N1,N2)

```
N1 : Wert
N2 : Wert

Ergebnis: Wert
```

Ergebnis der Division von N1 durch N2.

Bei PROFAN 1.x-Programmen sollte vor dem Neucompilieren überall das @Div durch @Div& ersetzt werden, da PROFAN 1.x nur ganzzahlige Rechenoperationen beherrschte.

Funktionen
Übersicht

@Eof(#N)

```
N : Integer - Dateinummer (1..8)

Ergebnis: Integer (0 oder 1)
```

Wenn mit dem letzten Lesen aus der Datei das Ende der Datei erreicht wurde, ergibt @Eof(#N) den Wert 1, ansonsten ist es 0.

(Wegen Kompatibilität zu PROFAN 1.x kann das # auch weggelassen werden.)

Funktionen
Übersicht

@Equ$(S1,S2) - @Equ(N1,N2)

@Equ$(S1,S2)

```
S1 : String 1
S2 : String 2

Ergebnis: Integer (0 oder 1)
```

Das Ergebnis ist WAHR (1), wenn beide Strings gleich sind, UNWAHR (0), wenn die Strings verschieden sind.

@EQU(N1,N2)

```
N1 : Wert - Wert 1
N2 : Wert - Wert 2

Ergebnis: Integer (0 oder 1)
```

Das Ergebnis ist WAHR (1), wenn beide Werte gleich sind, UNWAHR (0), wenn die Werte verschieden sind.

Funktionen
Übersicht

@Exp(N)

N - Wert

Ergebnis - Float

Exponentialfunktion

Funktionen
Übersicht

@FindFirst$(S) - @FindNext$()

@FindFirst$(S)

```
S : String - Datei-Maske

Ergebnis: String
```

Die Funktion sucht die erste Datei im aktuellen Verzeichnis, die der Dateimaske entspricht. Das Ergebnis ist die gefundene Datei. Handelt es sich um ein Verzeichnis, steht es in eckigen Klammern.

@FindNext$()

```
 Ergebnis: String
```

Findet die nächste Datei, die zur mit @FINDFIRST$(S) gegebenen Maske paßt. War die Suche erfolgreich, ist %IORESULT = 0! Beispiel:

```
Print @FindFirst$("*.PRF")
WhileNot %IOResult
  Print @FindNext$()
Wend
```

Funktionen
Übersicht

GetDir$(S)

```
S : Laufwerkskennzeichen

Ergebnis: String (Pfad)
```

Der Pfad des angegebenen Laufwerks wird ermittelt, wobei "@" für das aktuelle Laufwerk steht.
Beispiel:

```
Let CPfad$=@GetDir$("C:")
Let Pfad$=@GetDir$("@")
```

Funktionen
Übersicht

@GetKey$()

```
Ergebnis: String (1 Zeichen)
```

Wartet auf einen Tastendruck und gibt das Ergebnis zurück.

Funktionen
Übersicht

@GT(N1,N2) - @GT$(S1,S2)

@GT(N1,N2)

```
N1 : Wert - Wert 1
N2 : Wert - Wert 2

Ergebnis: Wert (0 oder 1)
```

Das Ergebnis ist WAHR (1), wenn Wert 1 größer als (Greater Than) Wert 2 ist.

@GT$(S1,S2)

```
S1 : String 1
S2 : String 2

Ergebnis: Integer (0 oder 1)
```

Das Ergebnis ist WAHR (1), wenn String 1 größer als (Greater Than) String 2 ist.

Funktionen
Übersicht

@If(N1,N2,N3)

```
N1 : Wert - Bedingung
N2 : Wert - Ergebnis bei erfüllter Bedingung
N3 : Wert - Ergebnis bei nicht erfüllter Bedingung

Ergebnis : Wert, abhängig von der Bedingung
```

In Anlehnung an C gibt es das IF auch als Funktion. Die Funktion @IF hat drei Argumente: Das erste Argument ist die Bedingung; das zweite Argument ist der Wert, der zurückgegeben wird, wenn die Bedingung erfüllt ist und das dritte Argument ist schließlich der Wert, der bei unerfüllter Bedingung zurückgegeben wird. Beispiele:

```
LET Text$ = @IF(@Equ(A%,1), "A ist gleich 1", \
                            "A ist ungleich 1")
PRINT Text$

LET A% = @IF(@Gt(A%,1000),@Add(A%,200),@Add(A%,50))
PRINT A%
```

Im zweiten Beispiel wird zu A% 200 addiert wenn es größer als 1000 ist; im anderen Fall kommen nur 50 dazu.

Funktionen
Übersicht

@Inkey$()

```
Ergebnis: String (1 Zeichen)
```

Gibt als Ergebnis das Zeichen der aktuell gedrückten Taste zurück.

Funktionen
Übersicht

@Input$(S1,S2,S3)

```
S1 - String: Prompt (Frage)
S2 - String: Fensterüberschrift
S3 - String: Vorgabestring bzw. Vorgabewert

Ergebnis: String bzw. Wert (Eingabe)
```

Eine Dialogbox zur Eingabe eines Strings wird auf den Bildschirm gebracht. Diese Funktion sollte wo möglich anstelle des INPUT-Befehles verwandt werden.

Ab PROFAN² 2.5 kann sowohl die Vorgabe als auch das Ergebnis numerisch sein. Die Funktion kann so zur Eingabe von Strings und numerischen Werten verwandt werden. Beispiele:

```
Let Ort$=@Input$("Ort eingeben:,"Test","Köln")

Let Z&=@Input$("Neuer Wert:","Test",Z&)
```

Funktionen
Übersicht

Ins$(S1,S2,N)

```
S1 : String
S2 : String
N  : Wert

Ergerbnis: String
```

Der String S1 wird in S2 an Position N eingegefügt.

Funktionen
Übersicht

@Instr(S1,S2)

```
S1 : String
S2 : String

Ergebnis: Wert
```

Das Ergebnis gibt an, an welcher Position S1 in S2 vorkommt. Kommt S1 in S2 nicht vor, ist das Ergebnis 0.

Funktionen
Übersicht

@Int(N)

```
N : Wert

Ergebnis : Integer / LongInt
```

Ganzzahliger Anteil von N. Es wird nicht gerundet. (Gerundet wird bei Beschränkung der auszugebenden Stellen mit NUMWIDTH.)

Funktionen
Übersicht

@KeyIn(S)

```
S : String

Ergebnis: Wert (0 oder 1)
```

Das Ergebnis ist dann 1, wenn die zuletzt gedrückte Taste im String S vorkommt. Beispiel:

```
WaitKey
Case @KeyIn("AaBb"):Print "A oder B"
```

Funktionen
Übersicht

@Len(S)

```
S : String

Ergebnis: Wert
```

Länge des Strings S

Funktionen
Übersicht

@List!(N) - @List$(N) - @List%(N) - @List&(N)

@LIST!(N)

```
N : Integer - Index (0 .. 999)

Ergebnis: Float
```

Das Ergebnis ist der N. Eintrag der Float-Liste. Die Liste wird mit dem Befehl LIST! gefüllt.

@List$(N)

```
N : Integer - Index (0 .. 999)

Ergebnis: String
```

Das Ergebnis ist der N. Eintrag der Stringliste. Die Liste wird mit dem Befehl LIST$ gefüllt.

@List%(N)

```
N : Integer - Index (0 .. 999)

Ergebnis: Integer
```

Das Ergebnis ist der N. Eintrag der Integerliste. Die Liste wird mit dem Befehl LIST% gefüllt.
(Aus Gründen der Kompatibilität zu PROFAN 1.x kann auch noch die Form @List(N) verwandt werden.)

@List&(N)

```
N : Integer - Index (0 .. 999)

Ergebnis: LongInt
```

Das Ergebnis ist der N. Eintrag der LongInt-Liste. Die Liste wird mit dem Befehl LIST& gefüllt.

Funktionen
Übersicht

@ListBox$(S,N)

S : String - Fenster-Überschrift

N : Wert - Typ der Listbox 0 oder 1

Ergebnis: Ausgewählter Sring

Der Inhalt der ListBox-Liste wird zur Auswahl (oder zum Durchblättern) angeboten. Bei 0 wird eine kleine, alphabetisch geordnete Listbox angezeigt, bei 1 eine große Listbox in ursprünglicher Ordnung (ideal für Textbetrachtung). Beispiel:

```
ClearList
AddFiles "*.BMP"
Let Wahl$=@ListBox$("Wähle Bild:",0)
LoadBmp Wahl$,10,10
```

Die Listbox-Liste wird ADDFILES und ADDSTRING gefüllt und mit CEARLIST geleert; %GETCOUNT enthält die Anzahl der Einträge und mit @LISTBOXITEM$ kann man einen einzelnen Eintrag auslesen. Siehe auch das Demo "Schnelleinstieg"!

Funktionen
Übersicht

@ListBoxItem$(N)

```
N : Integer (0 ... 999)

Ergebnis: String
```

Der N-te Eintrag der ListBox-Liste wird ermittelt.

Funktionen
Übersicht

@LoadFile$(S1,S2)

```
S1 : String - Überschrift
S2 : String - Dateimaske (mit Pfad)

Ergebnis: Dateiname (mit Pfad)
```

Es wird eine Dateiauswahlbox zum Laden (Öffnen) mit der Überschrift S1 geöffnet. Das Ergebnis ist die gewählte Datei (mit Pfad), die geöffnet werden kann. Beispiel:

```
LET NAME$=@LOADFILE$("ÖFFNE:","*.EXE")
```

Funktionen
Übersicht

@LT(N1,N2) - @LT$(S1,S2)

@LT(N1,N2)

```
N1 : Wert 1
N2 : Wert 2

Ergebnis: Wert (0 oder 1)
```

Das Ergebnis ist WAHR (1), wenn Wert 1 kleiner als (Less Than) Wert 2 ist.

@LT$(S1,S2)

```
S1 : String 1
S2 : String 2

Ergebnis: Integer (0 oder 1)
```

Das Ergebnis ist WAHR (1), wenn String 1 kleiner als (Less Than) String 2 ist.

Funktionen
Übersicht

@MCISend$(S)

S : String - MCI-Kommando

Ergebnis: String (MCI-Ergebnis)

Es wird der String S als Kommando gesandt. Das Ergebnis (oder ein Fehler) wird zurückgegeben. Beispiele:

Ausgabe einer "WAV"-Datei:

```
@MCISend$("open gong.wav type \
          waveaudio alias gong")
@MCISend$("play gong wait")  'Datei spielen
@MCISend$("close gong") '... schließen
```

Das `wait` veranlaßt den PC zu warten, bis der Vorgang beendet ist.

WAV-Datei per Mikro aufnehmen:

```
@MCISend$("open new type waveaudio \
       alias test")     'Neue WAV-Datei
@MCISend$("set test time format \
       milliseconds")  'Zeitformat msec
@MCISend$("record test from 0 to \
       5000 wait")     '5 sec Aufnahme
@MCISend$("Play test from 0 wait")
                       'abspielen
@MCISend$("save test test.wav")
                       'speichern
```

CD (Titel 3 - 5) abspielen:

```
@MCISend$(open cdaudio alias cd")
Print "Länge: ";@MCISend$\
      ("status cd length")
Print "Titelzahl: ";@MCISend$\
      ("status cd number of tracks")
@MCISend$("set cd time format tmsf")
@MCISend$("play cd from 3 to 5 wait")
@MCISend$("stop cd")
@MCISend$("close cd")
```

Auf diese Weise lassen sich alle Multimediageräte und auch Video für Windows steuern. Näheres finden Sie in der Anleitung und in der Fachliteratur.

Funktionen
Übersicht

@MenuItem(N)

```
N : Wert - Menü-Kennzeichen

Ergebnis: Wert (1 oder 0)
```

Wenn der aktuell gewählte Menüpunkt mit N (0 bis 254) übereinstimmt, ist das Ergebnis 1, anderenfalls ist es 0. Ist N gleich 0, dann ist das Ergebnis 1, wenn kein Menüpunkt angewählt wurde. 254 steht für die Anwahl des Copyright-Zeichens.

Funktionen
Übersicht

@MessageBox(S1,S2,N)

```
S1 : String - Meldungstext
S2 : String - Überschrift
N  : Integer - Art der MessageBox

Ergebnis -  Integer: gedrückter Knopf
```

Eine MessageBox wird auf dem Bildschirm gebracht. N setzt sich zusammen aus BUTTONS + ICON + DEFAULT + FENSTERART.

Werte für BUTTON:

0 - OK
1 - OK Abbrechen
2 - Abbrechen Wiederholen Ignorieren
3 - Ja Nein Abbrechen
4 - Ja Nein
5 - Wiederholen Abbrechen

Werte für ICON:

0 - Kein Icon
16 - STOP
32 - "?"
49 - "!"
64 - "i"

Werte für DEFAULT:

(Der Wert gibt an, welcher Knopf defaultmäßig angewählt ist:)

0 - erster Knopf
256 - zweiter Knopf
512 - dritter Knopf

Werte für FENSTERART:

0 - "normales" Fenster.
4096 - großes, nicht verschiebbares "Fehler"-Fenster.

Das Ergebnis ist der gedrückte Knopf:

1 - OK
2 - Abbrechen (Cancel)
3 - Abbrechen (Abort)
4 - Wiederholen

5 - Ignorieren

6 - Ja

7 - Nein

Funktionen
Übersicht

@Mid$(S,N1,N2)

```
S  : String
N1 : Integer - Position
N2 : Integer - Anzahl

Ergebnis: String
```

Das Ergebnis ist ein String, der N2 Zeichen ab Position N1 aus String S enthält.

Funktionen
Übersicht

@Mod(N1,N2)

```
N1 : Integer
N2 : Integer

Ergebnis: Integer
```

Das Ergebnis ist N1 modulo N2.

Funktionen
Übersicht

@Mouse(X1,Y1-X2,Y2)

```
X1,Y1 : Integer - linke obere Ecke
X2,Y2 : Integer - rechte untere Ecke

Ergebnis: Integer (0 oder 1)
```

Das Ergebnis ist 1, wenn beim letzten Mausklick die Maus im angegeben Bereich war. Beispiel:

```
WaitMouse
Case @Mouse(10,10-60,60):Gosub "Info"
```

Funktionen
Übersicht

@Mul(N1,N2)

```
N1 : Wert
N2 : Wert

Ergebnis: Wert
```

Multiplikation von N1 mit N2.

Funktionen
Übersicht

@Neq(N1,N2) - @Neq$(N1,N2)

@Neq(N1,N2)

```
N1 : Wert - Wert 1
N2 : Wert - Wert 2

Ergebnis: Integer (0 oder 1)
```

Das Ergebnis ist UNWAHR (0), wenn beide Werte gleich sind, WAHR (1), wenn die Werte verschieden sind.

@Neq$(S1,S2)

```
S1 : String 1
S2 : String 2

Ergebnis: Integer (0 oder 1)
```

Das Ergebnis ist UNWAHR (0), wenn beide Strings gleich sind, WAHR (1), wenn die Strings verschieden sind.

Funktionen
Übersicht

@Not(N)

```
N : Wert

Ergebnis: Integer (0 oder 1)
```

Das Ergebnis ist 1, wenn N den Wert 0 hat, ansonsten ist es 0.

Funktionen
Übersicht

@Or(N1,N2)

```
N1 : Wert
N2 : Wert

Ergebnis: Integer
```

ODER-Funktion: Das Ergerbnis ist ungleich 0, wenn N1 ODER N2 (oder beide) ungleich 0 sind. Anders ausgedrückt: Es ist nur dann 0, wenn beide Argumente 0 sind.

Funktionen
Übersicht

@Ord(S)

```
S : String

Ergebnis: Integer
```

Der ANSI-Code des ersten Zeichens von S.

Funktionen
Übersicht

@Par$(N)

```
N : Integer - Parameter-Nummer

Ergebnis: String - Parameter
```

Die Kommandozeilenparameter werden ausgelesen. Die Anzahl wird mit %PARCOUNT festgestellt.

Aufrufmöglichkeiten:

Interpreter:

```
      PROFAN <name.prf> <par2> <par3> ...
```

Runtime (bzw. .EXE-Datei als Runtime):

```
      PROFRUN <name.prc> <par2> <par3> ...
bzw.  <name.exe> <name.prc> <par2> <par3> ...
```

.EXE-Datei:

```
      <name.exe> ! <par2> <par3> ...
```

Bei der .EXE-Datei muß der erste Parameter das Ausrufezeichen sein, ansonsten würde das integrierte Runtimemodul versuchen, den erstn Parameter als Dateiname zu interpretiereen und mit einer Fehlermeldung reagieren.

Funktionen
Übersicht

@Pi()

```
Ergebnis: Float - PI
```

Die Kreitzahl PI (3,41...)

Funktionen
Übersicht

@ReadIni(S1,S2,S3)

```
S1 - String: Dateiname
S2 - String: Anwendungsname (Rubrik)
S3 - String: Eintrag

Ergebnis: String (Wert des Eintrages)
```

Ein Wert aus einer INI-Datei wird gelesen. Ermittlung des Druckers:

```
Let Drucker$=@ReadIni$("WIN.INI",\
              "Windows","Device")
```

Mit WriteIni können INI-Dateien erzeugt und beschrieben werden.

Funktionen
Übersicht

@RGB(R,G,B)

```
R,G,B : Wert - Farbwerte (0 .. 31)

Ergebnis : Wert (0 .. 32767)
```

Die Funktion errechnet eine der 32767 in Profan möglichen Farben aus den Farbanteilen Rot, Grün und Blau:

@RGB(0,0,0) = 0 (Schwarz)
@RGB(31,31,31) = 32767 (Weiß)

Funktionen
Übersicht

@Rnd(N)

```
N : Wert

Ergebnis: Integer bzw. LongInt
```

Zufallszahl zwischen 0 und N-1.

Funktionen
Übersicht

@SaveFile$(S1,S2)

```
S1 : String - Überschrift
S2 : String - Namensvorschlag

Ergebnis: Dateiname (incl. Pfad)
```

Es wird eine Dateiauswahlbox zum Speichern mit der Überschrift S1 geöffnet. Der Dateiname und/oder Pfad kann geändert werden. Das Ergebnis ist der Name (mit Pfad) unter dem gespeichert werden kann.

Funktionen
Übersicht

@SkanKey(N)

```
N : Integer - Scancode

Ergebnis: Integer (1 oder 0)
```

Wenn der letzte Scancode (virtueller Tastencode) mit N (0 bis 254) übereinstimmt, ist das Ergebnis 1, anderenfalls ist es 0. Tabelle der Scancodes siehe unter %SCANKEY.

Funktionen
Übersicht

@Sin(N)

```
N : Wert (Winkel in Bogenmaß)

Ergebnis : Float
```

Der Sinus des Winkels N.

Funktionen
Übersicht

@Sqr(N) - @Sqrt(N)

@Sqr(N)

```
N : Wert

Ergebnis : Float
```

Das Quadrat von N (N^2).

@Sqrt(N)

```
N : Wert

Ergebnis : Float
```

Die Quadratwurzel zu N.

Funktionen
Übersicht

@Str$(S)

```
N : Wert

S : String
```

Der Wert von N wird in einen String ohne folgende Leerzeichen umgewandelt.
Die Länge des Strings und (bei Float) die Anzahl der Nachkommastellen werden durch die Befehle NUMWIDTH und DECIMALS eingestellt.

Funktionen
Übersicht

@Sub(N1,N2)

```
N1 : Wert
N2 : Wert

Ergebnis: Wert
```

N2 wird von N1 abgezogen (subtrahiert).

Funktionen
Übersicht

@TMouse(X1,Y1 - X2,Y2)

```
X1,Y1 : Integer - links oben
X2,Y2 : Integer - rechts unten

Ergebnis: Integer (0 oder 1)
```

War beim letzten Mausklick die Maus innerhalb des Rechteckes, ist das Ergebnis 1. Die Koordinaten sind Textkoordinaten (wie bei LOCATE oder TBOX).

Funktionen
Übersicht

@Trim$(S)

```
S : String

Ergebnis: String
```

Die führenden und endenden Leerzeichen eines Strings werden abgeschnitten.

Funktionen
Übersicht

@Upper$(S)

```
S : String

Ergebnis: String
```

Der String S wird komplett in Großbuchstaben umgewandelt, wobei die deutschen Umlaute korrekt berücksichtigt werden.

Funktionen
Übersicht

@Val(S)

```
S : String

Ergebnis: Wert
```

Der String S wird in einen numerischen Wert umgewandelt. Das Ergebnis ist 0, wenn der String nichtnumerische Zeichen enthält.

Funktionen
Übersicht

PROFAN² 2.5 - Einführung

REFERENZ

INHALT

REFERENZ

1 - EINFÜHRUNG

Gewiß, es gibt schon Batchprogrammiersprachen unter Windows. Die bekanntesten sind "WinBatch" und "Oriel". "WinBatch" ist eine herausragende Sprache, aber leider völlig ohne Grafik.
"WinBatch" von "WilsonWare" (USA) ist sowohl als Shareware oder Vollversion (mit deutschem Handbuch) erhältlich, als auch als Batchsprache (unter anderem Namen) im Norton Desktop für Windows integriert.
"Oriel" hingegen hat umfangreiche Grafikbefehle, aber doch einen sehr eingeschränkten Befehlsumfang. So gibt es da z.B. keine Möglichkeit Eingaben zu machen oder Dateien von Diskette oder Festplatte zu lesen.
Beide Sprachen enthalten keinen Compiler. Anwendungen können daher nur an Leute weitergeben werden, die Ihrerseits diese Sprache gekauft haben.

Ziele der Entwicklung von Profan waren:

* Eine einfache - an BASIC angelehnte - Syntax auch für den Anfänger
* Alle Grafikmöglichlkeiten, die "Oriel" bietet ... und noch mehr
* umfangreiche Datei- und Verwaltungsfunktionen
* der komplette Interpreter kleiner als 100 kB, keine weiteren DLLs oder sonstige Dateien (die 100 kB ließen sich in der vorliegenden zweiten Version nicht mehr halten ... aber unter 200 kB will ich auch künftig bleiben).
* Möglichkeit, Anwendungen ohne weitere Kosten weiterzugeben.

Herausgekommen ist "Profan". Eine komplette Programmiersprache. Der Anfänger wird viele BASIC-Befehle in gewohnter Form wiederfinden, u.a.
PRINT, LOCATE, CLS, INPUT, GOTO, GOSUB, RETURN, IF, WHILE, WEND, SOUND, PRINT #n, INPUT #n, END, LET, ...

Auch die Variablen und Konstanten werden weitestgehend wie in BASIC gehandhabt.

Bei den Dateioperationen wurde allerdings das etwas vielseitigere Konzept von PASCAL übernommen:
ASSIGN, RESET, REWRITE, APPEND, RENAME, ERASE, CLOSE, ...
Mit COPY können Dateien kopiert werden und mit CHDIR, MKDIR und RMDIR können Verzeichnisse verwaltet werden. Spezielle Funktionen geben Auskunft über das aktuelle Laufwerk, den aktuellen Pfad und vieles mehr.
Dazu kamen dann noch WINDOWS-spezifische Dinge, wie Menüs, Listboxen, Inputboxen, Load- und Save-Dialoge, ... um nur ein paar Möglichkeiten zu nennen.

Und die Krönung sind natürlich die umfangreichen Grafikbefehle, die direkt auf den Grafik-Kern von Windows zugreifen: USEBRUSH, USEFONT, USEPEN, USEICON, COPYBMP, RECTANGLE, ROUNDRECT, ELLIPSE, ARC, PIE, DRAWTEXT, ...
Mit LOADBMP und SAVEBMP können Bitmap-Grafiken geladen, verarbeitet und gespeichert werden.

Mit Windows 3.1 wurde dann auch eine weitere Sprachergänzung notwendig: Ansteuerung aller Multimediageräte und Abspielen von WAV-Sounddateien über die Soundkarte. Das Abspielen von Musik-CDs ist ebensowenig ein Problem wie das Untermalen von Spielen mit Geräuschen oder das Abspielen von Microsofts Video für Windows: MCISEND, PLAYSOUND, ...

Der fortgeschrittene Anwender wird vermutlich zur Bildschirmausgabe die textorientierten Ausgabebefehle des BASIC weniger verwenden, als vielmehr die neuen und mächtigen Windows-Befehle.

Strukturierte und übersichtliche Programmierung wird ermöglicht, indem nur ein Befehl pro Zeile erlaubt ist und zahlreiche Kontrollstrukturen verfügbar sind:

IF ... ELSEIF ... ELSE ... ENDIF, CASE, WHILE ... WEND, GOSUB ... RETURN.
Mit PROC ... ENDPROC können - wie in Pascal - Prozeduren definiert werden und mit DEF können neue Funktionen definiert werden. In Prozeduren und Funktionen gibt es lokale Variablen. Außerdem ist beliebiges Einrücken ebenso statthaft, wie komplett leere Zeilen.

Im Laufe der Entwicklung kam dann auch noch ein Compiler dazu, der einen sehr kompakten Zwischencode erzeugt, der mit einem Runtime-Modul gestartet werden kann. Das letzte Glied in der Kette zur kompletten Programmiersprache war dann der Linker, der Runtimemodul und Zwischencode zu einer eigenständigen EXE- Datei linkt. Und da das Runtime-Modul nur wenig mehr als 120 kB groß ist, sind recht aufwendige Windowsapplikationen unter 150 kB möglich. Da die fertige PROFAN-Applikation immer noch als Runtime-Modul für weitere Zwischencode- Dateien dienen kann und sogar eine Übergabe von Parametern möglich ist, steht der modularisierten Programmierung nichts im Wege. Die Sahnehaube aber ist PROFED, die Entwicklungsumgebung: In der vorliegenden zweiten Version wurde sie noch einmal kräftig erweitert: Beliebige Schriftwahl, Projektverwaltung, ...

Ach ja: Das berühmte "Hallo Welt"-Programm in Profan:

```
Print "Hallo Welt"
WaitKey
End
```

So einfach ist das. Das "WaitKey" steht nur deshalb da, damit der Anwender Gelegenheit hat, das Ergebnis zu betrachten: Ein Programm in einem Windows- Fenster mit allem was dazugehört. Sozusagen eine vollständige Windows- Applikation. Mit einem Tastendruck oder über die entsprechende Fensterfunktion wird das Programm beendet.

Die Kombination von Interpreter und Compiler bietet interessante Vorteile. Im Interpretermodus entwickeln Sie das Programm und testen es aus. Und wenn es dann fertig getestet ist, verleihen Sie ihm mit dem Compiler den nötigen Drive! Esw wird dingend geraten erst ein getestetes Programm zu compilieren, da die Laufzeitfehlermeldungen oft weniger aussagekräftig sind, als die des Interpreters.

INHALT
REFERENZ

2 - Installation

PROFAN² benötigt mindestens einen 286er mit 2 MB RAM und VGA-Grafik. Bei weniger RAM kann ein Profan-Programm u.U. kein anderes Programm mehr aufrufen. Optimal sind - wie für Windows 3.1 allgemein - 8 MB RAM auf einem 386er mit mind. 25 MHz. Im Real-Mode läuft PROFAN² nicht. Bei EGA- oder HERCULES-Grafik wird das DEMO nicht ordentlich angezeigt und bei der Programmierung ist darauf zu achten, daß mit WINDOW die richtige Fensterschirmgröße eingestellt wird.

HINWEIS: Wenn beim DEMO nicht alles funktioniert, wie erwartet (also z.B. DRAW nicht startet, etc.), können die einzelnen Programmteile über die Zeile PROFRUN <name>.PRC problemlos aufgerufen werden. Das spart Speicherplatz. (Shareware: SHPRRUN <name>)

PROFAN² ist vom Installationsprogramm auf dem von Ihnen gewählten Datenträger im gewählten Verzeichnis installiert worden.

(Die Sharewareversion ist gemäß Anleitung in der Datei LIESMICH.TXT zu installieren.)

Damit PROFAN²-Programme, auch wenn sie nicht zur EXE-Datei gelinkt sind, durch Doppelklick gestartet werden können, sind folgende Zeilen in die Datei WIN.INI unter dem Abschnitt "Extensions" einzufügen:

```
prf=c:.exe ^.prf
prc=c:.exe ^.prc
```

(Besitzer der Shareware-Version geben statt "profan.exe" und "profrun.exe" bitte "shprofan.exe" und "shprrun.exe" an. Sollten Sie PROFAN² nicht auf Laufwerk C: installiert haben, ändern Sie die Zeilen entsprechend ab.) Nur die wenigsten Dateien werden wirklich gebraucht. Die meisten Dateien gehören zum Demonstrationsprogramm und dienen Ihrer Information.

DEMOPROGRAMM:

Das DEMO-Programm ist komplett in PROFAN² geschrieben. Die Listings sind in der registrierten Vollversion mit auf der Diskette.

DEMO.EXE - Hauptprogramm

*.PRC - kompilierte Programmodule. Beachten Sie die Kompaktheit selbst so komplexer Programme, wie RGH-DRAW.

*.PLT - Beispiel-Farbpaletten für Malprogramm und Farbkasten.

*.BMP - Bilder für das Hypertext-Beispiel und für das Malprogramm

*.WAV - Sounddateien für das Spiel

*.TXT - Texte für den Schnelleinstieg

*.PRF - Listings des DEMO-Programmes und Beispielprogramme

*.INC - Listing der Includedateien zum Malprogramm

PROFAN² (Entwicklung):

Diese Dateien sind unbedingt notwendig: (In Klammern steht bei abweichendem Dateinamen der Name der Datei in der Shareware-Version.)

PROFAN.EXE (SHPROFAN.EXE) - Der Interpreter

PROFED.EXE (SHPROFED.EXE) - Die Entwicklungsumgebung

*.HLP - Hilfstexte für die Entwicklungsumgebung (nur Vollversion)

*.CRD - Hilskartei zur PROFAN-Syntax

DRUCK.EXE - (nur Vollversion) Druck-Programm

ANSI.EXE - (nur Vollversion) ANSI-Tabelle

PROFCOMP.EXE (SHPRCOMP.EXE) - Der Compiler

PROFLINK.EXE (SHPRLINK.EXE) - Der Linker

PROFRUN.EXE (SHPRRUN.EXE) - Das Runtime-Modul

(Wollen Sie PROFAN ausschließlich als Batchsprache nutzen, können Sie auf die letzten drei Programm-Dateien verzichten. Compilieren und Linken ist dann weder notwendig noch sinnvoll.)

PROFAN² (Anwendung als Batchsprache für Windows):

PROFAN.EXE (SHPROFAN.EXE) - Der Interpreter

*.PRF - Das Batchprogramm (+ Module) in Profan

PROFAN² (compilierte Anwendung):

PROFRUN.EXE (SHPRRUN.EXE) - Das Runtime-Modul oder ein zur EXE-Datei gelinktes Profanprogramm

*.PRC - Programm, bzw. Programmmodule

Ein zur EXE-Datei gelinktes Profanprogramm ist ohne weitere Dateien eigenständig lauffähig.

INHALT
REFERENZ

5 - Include-Dateien

`$I <Dateiname>`

Ab PROFAN² 2.0 gibt es auch die Möglichkeiten, Programmodule während des Compilierens bzw. Interpretierens dazuzuladen. Das geschieht mit der (bisher einzigen) Direktive:

```
$I <Dateiname>
```

(Hinter dem großen Buchstaben I muß ein Leerfeld sein.) Für den Interpreter oder Compiler ist es dann so, als stünde der Inhalt der Datei an der entsprechenden Stelle im Programm. Beispiel:

```
Declare Test%, Frage$, Antwort$
$I ZUSATZ.INC
...
```

Auf diese Weise können häufig gebrauchte Module beliebig oft verwandt werden, ohne daß sie jedesmal neu eingetippt oder über die Zwischenablage kopiert werden müßten. Besonders sinnvoll ist diese Methode, wenn man zum Beispiel besonders häufig gebrauchte Prozeduren oder selbstdefinierte Funktionen in Include-Dateien abspeichert. Ich habe mir angewöhnt, Includedateien mit der Endung .INC zu versehen.

INHALT
REFERENZ

6 - Operatoren

Inc, Dec, Add, Sub

Operatoren gibt es in PROFAN nicht. Alles wird mittels Funktionen erledigt. Es gibt für alle gewohnten Operatoren entsprechende Funktionen. Statt "LET A%=A%+3" muß z.B. geschrieben werden:

```
LET A%=@ADD(A%,3)
```

Übrigens: Das "LET" darf nicht weggelassen werden.

Diese Philosophie wird auch konsequent bei Vergleichen und Bedingungen eingesetzt, etwa:

IF A%=B%	wird zu `IF @EQU(A%,B%)`
IF A% AND B%	wird zu `IF @AND(A%,B%)`

Lediglich für schnelle Berechnungen mit Integer-Werten (z.B. in Schleifen, etc.) gibt es in PROFAN² einige neue Befehle, die schneller und übersichtlicher sind:

`INC V%`	erhöht V% um 1
`DEC V%`	erniedrigt V% um 1
`ADD V%,n`	erhöht V% um n
`SUB V%,n`	erniedrigt V% um n

(Hierbei steht V% für eine beliebige Integer-Variable und n für einen beliebigen Ausdruck.)

INHALT
REFERENZ

8 - Arrays

Dim%, Dim&, Dim!, Dim$
List%, List&, List!, List$
@List%, @List&, @List!, @List$

Auch Arrays gibt es in PROFAN² nicht in der z.B. von BASIC gewohnten Form. Es gibt lediglich ein Array mit max. 1000 Elementen (0 .. 999) je Datentyp.

Dimensioniert werden die Arrays mit den Befehlen DIM%, DIM!, DIM& und DIM$. (Wegen der Kompatibilität zu PROFAN 1.x wird auch noch der Befehl DIM ohne Postfix unterstützt. Er hat die gleiche Funktion wie DIM%.) Mit dem Befehl

```
DIM% 499
```

wird ein Integerarray mit 500 Elementen definiert. Jeder DIM-Befehl darf nur einmal während des Programmlaufes vorkommen und zwar zeitlich vor dem ersten Zugriff auf ein Element des Arrays. Es ist sinnvoll, die DIM-Befehle an den Anfang des Programmes zu stellen. Nachträgliche Vergrößerung des Arrays ist ebensowenig möglich, wie ein Löschen und Neufefinieren.

Zum Füllen der Array weden die Befehle LIST%, LIST!, LIST& und LIST$ verwandt. Mit der Zeile

```
LIST& 99 = 120000
```

würde ich dem 99. Element des LongInt-Array den Wert 120000 zuweisen. Ist das Array nicht definiert oder ist 99 außerhalb der definierten Größe, erfolgt eine Fehlermeldung.

Um den Inhalt eines Array-Elementes zu lesen, wird je nach Typ einer der Funktionen @LIST%. @LIST!, @LIST& oder @LIST$ verwandt. Wenn ich den Inhalt des 34. Elementes des Stringarray der Variable Test% zuweisen möchte, verwende ich die Zeile:

```
LET Test$ = @LIST$(34)
```

INHALT
REFERENZ

7 - Variablen, Datentypen und Typen-Umwandlung

Declare

PROFAN² kennt vier Datentypen: Integer, LongInt, Float und String. Wie in PASCAL müssen Variablen vor ihrer Verwendung declariert werden. Das geschieht in PROFAN² mit dem Befehl "Declare". Wenn der Befehl auch überall im Programm stehen darf, so empfiehlt es sich doch, alle Declare-Anweisungen am Anfang des Programmes oder der Prozedur zu stellen. Mehrere Variablen können durch Kommata getrennt mit einem Declare-Befehl declariert werden:

```
Declare Zahl1%, Zahl2!, Text$
Declare Ergebnis&
```

Der Typ der Variable wird durch das Postfix (letztes Zeichen des Variablennamens festgelegt. Der Variablenname darf (inclusive Postfix) 32 Zeichen lang sein und außer dem Leerzeichen so ziemlich alle Zeichen enthalten. Trotzdem ist dringend zu empfehlen, nur Buchstaben, Ziffern, Unterstriche und Punkte zu verwenden. Sonst wirds unleserlich.

In einer Prozedur oder einem Unterprogramm definierte Variablen sind nur in der Prozedur bekannt. Hingegen sind in einer Prozedur oder einem Unterprogramm auch alle außerhalb declarierten Variablen bekannt. Hier verhält sich PROFAN eher wie PASCAL. (Das aus BASIC bekannte SHARED wird nicht benötigt.)

Die Typen im einzelnen:

INHALT
REFERENZ

7.1 Integer

Integervariablen werden, wie in BASIC üblich, durch ein % als Postfix gekennzeichnet.
Integervariablen können Werte von -32768 bis + 32767 annehmen. Wird dieser Wert über- oder unterschritten, erolgt zwar keine Fehlermeldung, aber die Ergebnisse werden ungewöhnlich sein. Nur mit Integervariablen können auch die vier Rechenbefehle INC, DEC, ADD und SUB durchgeführt werden.

7.2 LongInt

LongInt-Variablen haben das & als Postfix. Im übrigen gilt für den Variablennamen das gleiche, wie für Integer. Der Wertebereich geht von -2 Milliarden bis etwas über +2 Milliarden.

7.3 Float

Float heißen jene Variablen, die reelle Zahlen (Fließkomma-Zahlen) aufnehmen können. Die Rechengenauigkeit ist mindestens 10 Stellen. Ach ja: Das Postfix ist, wie in BASIC, das Ausrufezeichen. Im Gegensatz zu BASIC darf das Postfix aber niemals weggelassen werden.Den Versuch, eine Variable ohne Postfix (%, &, !, $) zu declarieren, würde PROFAN² bemängeln! Soll eine ganze Zahl als Float gekennzeichnet werden (z.B., damit beim Ausdruck die Nachkommastellen gedruckt werden), ist sie trotzdem als Dezimalbruch zu schreiben: PRINT 5.0! Wenn PROFAN² auch ein deutsches Produkt ist, so wird trotzdem der Dezimalpunkt verwandt.

7.4 String

Strings werden durch ein $ am Ende des Dateinamens gekennzeichnet und können maximal 255 Zeichen lang sein. Stringkonstanten werden durch Anfügrungszeichen eingeschlossen:

```
LET Text$="Teststring"
```

(Werden diese weggelassen, gäbe es zwar keine Fehlermeldung, aber der String würde in Großbuchstaben umgewandelt werden. Innerhalb einer Funktion könnte das Stringende allerdings nicht erkannt werden und eine Fehlermeldung wäre die Folge.)

7.5 Typen-Umwandlung

@Str$, @Int, @Val

Die Umwandlung der verschiedenen Variablentypen in andere erfolgt zumeist automatisch. Bei einer Rechnung oder Zuweisung wird das Ergebnis automatisch in den Typ der Variable, der das Ergebnis zugewiesen wird, umgewandelt. Wenn Zahl1! den Wert 5.678 hätte, würde bei

```
LET Zahl2% = Zahl1!   der Variablen Zahl2% der Wert 5 zugewiesen.

PRINT Zahl2%
```

würde auf jeden Fall den Wert als Dezimalwert ausgeben, ebenso

```
LET Text$=@STR$(Zahl2%)
```

oder auch

```
LET Text$=Zahl2%
```

(Ja, das geht auch!). Mit der Funktion @INT können Sie jedoch bewußt die Nachkommastellen abschneiden:

```
PRINT @INT(Zahl2%)
```

Die Funktion @INT schneidet bei einer Zahl die Nachkommastellen ab. Das Ergebnis kann jedoch - wie immer - jedem Variablentyp zugewiesen werden, bevorzugt natürlich einer Integer- oder LongInt-Variablen.

Die Funktion @STR$ verwandelt einen numerischen Wert in einen String, wobei bezüglich der Formatierung die Einstellungen mittels DECIMALS und NUMWIDTH berücksichtigt werden. Die Funktion wird also häufig bei formatierter Zahlenausgabe Verwendung finden. Wird keine Formatierung gewünscht, kann das @STR$ auch weggelassen werden.

Die Funktion @VAL ermittelt den numerischen Wert eines Strings. Kann der String nicht komplett umgewandelt werden, wird der Wert 0 angenommen. Eine Fehlermeldung oder Warnung erfolgt nicht. Das @VAL kann oft auch weggelassen werden:

```
LET Zahl% = Text$
```

Selbst eine Zeile

```
LET Zahl% = Maier
```

würde funktionieren. Die Zahl erhielte dann den Wert 0. Wenn die Warnungen mit "SETERRORLEVEL 1" eingeschaltet sind, würde in diesem Fall jedoch eine Warnung darauf hinweisen, daß "Maier" nicht als Zahl zu interpretieren ist. Diese Warnung erscheint immer dann, wenn Interpreter oder Runtime einen Ausdruck nicht interpretieren können, also auch dann, wenn ein String erwartet wird, aber etwas da steht, was weder String noch numerischer Wert ist. (Warum? Nun: Wenn kein String - erkennbar durch Anführungszeichen - gefunden wird, versucht PROFAN² eine Zahl zu lesen, um diese dann intern in einen String umzuwandeln.)

Ich ziehe es vor, die Umwandlungen zwischen Strings und numerischen Werten immer mit @VAL und @STR$ zu programmieren. Das macht die Programme lesbarer.

INHALT
REFERENZ

4 - RGH-DRUCK

RGH-DRUCK ist ein eigenständiges Programm zum Drucken beliebiger Texte. Es enthält weitreichende Formatierungsmöglichkeiten.

Die Menüpunkte:

Datei - Öffnen
Datei - Drucken
Datei - Seitenformat
Datei - Ende
Text - Schrift

Alle diese Funktionen können auch über die entsprechenden Buttons direkt angewählt werden. Die Bilder sind selbsterklärend.

Im System-Menü von RGH-DRUCK finden sich noch zwei weitere Menüpunkte:

Systemsteuerung
Über ...

INHALT
REFERENZ

Datei - Öffen

Öffnet die zu druckende Datei. Diese wird in den Speicher gelesen und kann auch betrachtet werden. Änderungen sind jedoch nicht möglich. Lange Texte werden auch nur teilweise angezeigt, jedoch vollständig gedruckt.

Datei - Drucken

Der Text wird im eingetellten Format mit der eingestellten Schrift gedruckt.

Datei - Seitenformat

Zahlreiche Formatierungen können eingestellt werden: Ränder, Zeilenlineal, Zeilennummern (bei PROFAN-Listings sinnvoll), ANSI-CC-Byte (in der Regel nicht anzukreuzen), Datum, Zeit, Tabulatorgröße. Die Tabulatorgröße des PROFAN-Editors ist 8.

Datei - Ende

Beendet das Druckprogramm.

Text - Schrift

Es kann eine Schriftart und Schriftgröße gewählt werden. Es ist ratsam eine schmale Schrift zu verwenden, da Zeilen, die nicht ganz in eine Druckzeile passen, einfach abgeschnitten werden.

Systemsteuerung

Da kann der Drucker (und anderes) eingestellt werden.

Über ...

Copyright-Hinweis. Hinweis auf das amerikanische Programm, dessen Quellcode Grundlage von RGH-Druck ist.

Amerikanisches Original: PLXpress von Doug Overmyer.
Erweitert und ins Deutsche übertragen von Roland G. Hülsmann.

3 - RGH-PROFAN-EDITOR 2.5

Der Editor ist eigentlich eine komplette Entwicklungsumgebung für RGH- PROFAN.

INHALT
REFERENZ

EDITOR - Übersicht

Der Editor ist ein "Multi-File"-Editor, das heißt: Es können mehrere Dateien gleichzeitig editiert werden. Das ermöglicht einfaches Kopieren und Verschieben auch zwischen verschiedenen Programmen über die Zwischenablage. Um ein anderes Programm oder einen anderen Text einzusehen, muß das aktuelle Programm nicht verlassen werden.

Neu in der Version 2.0 ist neben dem Optionsmenü mit Schriftarten und Projektverwaltung die erweiterte Steuerung über Tastatur. Gerade beim Programmieren hat man schneller eine Taste bzw. Tastenkombination gedrückt, als eine Funktion über das Menü ausgewählt.

In Version 2.1 kam noch der "Drag & Drop"-Mechanismus hinzu: Sie können einfach eine (oder meherer) Datei(en) aus dem Dateimanager auf den Editor oder das Editor-Icon ziehen und sie wird automatisch geladen.

Außerdem gibt es eine INI-Datei (PROFED.INI), in der die aktuellen Einstellungen (Schrift, Projektdatei) festgehalten werde. WICHTIG: Um die aktuellen Einstellungen für den nächten Programmstart zu speichern, muß das Programm über den Menüpunkt DATEI-ENDE verlassen werden. (Über das Systemmenü kann man das Programm verlassen, ohne die aktuelle Einstellung in der PROFED.INI zu ändern.)

Ab Version 2.5 hat der Editor eine Windows-typische Hilfe und 3D-Dialoge, wenn die entsprechende DLL (CTL3D.DLL) im System vorhanden ist. Ist diese DLL nicht vorhanden, werden die Dialoge wie bisher angezeigt. Außerdem wurde der Menüpunkt DATEI-LADEN in DATEI-ÖFFNEN umbenannt. Ach ja ... mehr dem Zeitgeiste folgend, denn der Notwendigkeit: Eine Toolbar mit den wichtigsten Funktionen und eine Statuszeile zur Anzeige von Hilfstexten zu den Menüpunkten wurde auch noch hinzugefügt.

INHALT
REFERENZ

EDITOR - Aufruf

Sie haben mehrere Möglichkeiten, um mit dem Editor zu arbeiten:

Aufruf mit Kommandozeilenparameter

Sie können beim Aufruf des Editors eine oder mehrere Dateien als Kommandozeilenparameter angeben. Die Dateien werden beim Start automatisch geladen. (Der Desktop der letzten Arbeitssitzung mit dem Editor bleibt dabei unberücksichtigt.)

Aufruf ohne Parameter

Wenn Sie den Editor ohne Parameter (zum Beispiel über das Symbol des Programm-Managers aufrufen, wird automatisch der Desktop der letzen Arbeitssitzung wiederhergestellt, das heißt: alle Dateien die damals bearbeitet wurden werden wieder genau so angezeigt, wie bei der letzten Arbeitssitzung.

Drag & Drop

Sie können eine Datei (oder mehrere) mittels Maus aus dem Dateimanager auf den Editor oder den zum Symbol verkleinerten Editor ziehen. Diese Datei wird automatisch geladen.

INHALT
REFERENZ

Editor - Menü

Die Menüpunkte im Einzelnen:

DATEI
- Neu
- Öffnen [F3]
- Speichern [F2]
- Speichern als
- Drucker einrichten
- Druckprogramm
- Datei-Manager
- Programm aufrufen
- Ende

BEARBEITEN
- Rückgängig [Alt-BS]
- Ausschneiden [Shift-Entf]
- Kopie [Strg-Einfg]
- Einfügen [Shift-Einfg]
- Löschen [Strg-Entf]
- Alles löschen [Alt-Entf]
- Ablage

SUCHEN
- Finden [Strg-F]
- Ersetzen [Strg-A]
- Weitersuchen [Strg-L]

PROFAN
- Ausführen (Interpreter)
- Compilieren [F9]
- Starten [F8]
- EXE-Datei erstellen

OPTIONEN
- Schrift ...
- Projekt speichern
- Projekt laden

FENSTER
- Nebeneinander
- Überlappend
- Symbole anordnen
- Alle schließen
- Rechner
- Ansi-Tabelle
- Kalender
- 1 ...

HILFE
- PROFAN-Hilfe [F1]
- Referenz
- Editor
- Druckprogramm
- Über ...

INHALT
REFERENZ

Datei - Neu

Ein neues RGH-PROFAN-Programm schreiben. Es bekommt zunächst den Namen "OHNENAME.PRF". Existiert bereits ein Programm "OHNENAME.PRF", so wird es geladen. Es empfiehlt sich, das Programm sogleich mit "Speichern als" abzuspeichern und ihm einen neuen Namen zu geben.

Datei - Öffnen

Eine bereits bestehende PROFAN-Datei wird zum Bearbeiten geöffnet. (Durch Ändern der Dateimaske können beliebige Texte geladen werden, wie zum Beispiele Include-Dateien oder Datendateien.)

Datei - Speichern

Die aktuell aktive PROFAN-Datei wird unter ihrem Namen gespeichert.

Datei - Speichern als

Die aktuell aktive PROFAN-Datei kann unter einem neuen Namen/Pfad abgespeichert werden.

Datei - Drucker einrichten

Der Drucker kann ausgewählt und eingerichtet werden.

Datei - Druckprogramm

Das Druckprogramm "RGH-DRUCK" wird aufgerufen. Da das Druckprogramm die zu druckende Datei von der Festplatte liest und nicht aus dem Editor, überprüft das Programm zunächst ob geänderte Dateien etwa noch nicht abgespeichert wurden und fragt nach, ob man dies nachholen möchte.

Datei - Datei-Manger

Der Dateimanager wird aufgerufen.

Datei - Programm aufrufen

Ein beliebiges EXE-Programm kann über diesen Menüpunkt aufgerufen werden.

Datei - Ende

Der Editor wird beendet. Sollten Programme nach der Veränderung noch nicht abgespeichert sein, wird jeweils nachgefragt, ob man dies nun möchte.

BEARBEITEN - Rückgängig

Der letzte Vorgang kann evtl. noch rückgängig gemacht werden. Hierzu ist gleichzeitig die ALT- und Rückschritt-Taste zu drücken. Der Umweg über das Menü ist nicht immer empfehlenswert.

BEARBEITEN - Ausschneiden

Der markierte Text wird in die Zwischenablage kopiert und aus dem Programm entfernt.

BEARBEITEN - Kopie

Der markierte Text wird in die Zwischenablage kopiert, bleibt aber im Programm.

BEARBEITEN - Einfügen

Der Text der Zwischenablage wird an der Kursorposition eingefügt.

BEARBEITEN - Löschen

Der markierte Text wird gelöscht.

BEARBEITEN - Alles löschen

Der gesamte Text wird gelöscht.

BEARBEITEN - Ablage

Die Windows-Zwischenablage wird aufgerufen. So kann man nachsehen, was da gerade drin ist.

SUCHEN - Finden

Sucht im aktuellen Programmfenster nach einem Text. Es kann angegeben werden, ob zwischen Groß- und Kleinschreibung zu unterscheiden ist. Mit <Strg-L> kann die Suche wiederholt werden.

SUCHEN - Ersetzen

Sucht im aktuellen Programmfenster nach einem Text und ersetzt ihn durch einen anderen. Es kann angegeben werden, ob Groß- und Kleinschreibung zu unterscheiden ist, ob im ganzen Text gesucht werden soll und ob vor jedem Ersetzen nachgefragt werden soll.

SUCHEN - Weitersuchen

Wiederholt die Suche bzw. das Suchen und Ersetzen.

PROFAN - Ausführen (Interpreter)

Führt ein PROFAN-Programm mit dem Interpreter aus. Das Programm wird von der Festplatte gelesen. Ist ein noch nicht in der aktuellen Version abgespeichertes Programm im Editor, wird nachgefragt, ob man es zuvor abspeichern will.

PROFAN - Compilieren

Es wird ein PROFAN-Programm (.PRF) compiliert. Der Compiler PROFCOMP.EXE muß sich im gleichen Verzeichnis wie der Editor befinden. Das Ergebnis ist eine Datei mit der Endung .PRC, die vom Runtime-Modul PROFRUN oder jeder PROFAN-EXE-Datei ausgeführt werden kann. Die PRC-Datei ist in der Regel nur halb so groß, wie die PRF-Datei, dafür aber deutlich (!) schneller.

PROFAN - Starten

Ein compiliertes PROFAN-Programm wird vom Runtime-Modul gestartet und ausgeführt. PROFRUN.EXE muß sich im gleichen Verzeichnis wie der Editor oder im Windowsverzeichnis befinden.

PROFAN - EXE-Datei erstellen

Eine PRC-Datei wird mit dem Runtime-Modul zu einer EXE-Datei gelinkt. Der Linker PROFLINK.EXE muß sich im gleichen verzeichnis wie der Editor befinden. Die EXE-Datei kann unter Windows selbstständig ausgeführt werden. Da sie das komplette RUNTIME-Modul enthält, kann sie für PRC-Dateien auch als Runtime-Modul dienen.

OPTIONEN - Schrift ...:

Eine der fünf Systemschriften kann eingestellt werden. Zum Programmieren mit PROFAN empfehle ich die nicht-proportionalen Schriften ANSI oder SYSTEM. Für die Bearbeitung von ASCII-Texten aus der DOS-Welt ist ASCII zu wählen. Dann stimmen auch da die Umlaute.

OPTIONEN - Projekt speichern

Der aktuelle Desktop (Anordnung der Editierfenster) wird unter einem frei zu wählenden Namen gespeichert (vorgeschlagene Endung .DSK). Es empfiehlt sich, alle Dateien, die zu einem Programmprojekt gehören zu öffnen, bis auf das Hauptprogramm alles zu Icons zu verkleinern und dann diese Konfiguration als Projekt abzuspeichern. Wenn man dann weiterarbeiten will, kann man das Projekt mit allen Dateien mit einem Aufruf des folgenden Menüpunktes laden.

OPTIONEN - Projekt laden

Ein gespeichertes Projekt wird mit allen seinen Dateien geöffnet. Bereits vorher geöffnete Dateien werden geschlossen.

FENSTER - Nebeneinander

Die verschiedenen Programmfenster im Editor werden nebeneinander, bzw. übereinander angeordnet, sodaß sie sich nicht überschneiden.

FENSTER - Überlappend

Die Programmfenster werden versetzt hintereinander angeordnet, sodaß von allen die Titelseite zu sehen ist.

FENSTER - Symbole anordnen

Die Icons der verkleinerten Programmfenster werden in Reih' und Glied gebracht.

FENSTER - Alle schließen

Alle Programmfenster werden geschlossen.

FENSTER - Rechner

Der Windows-Taschenrechner wird für Berechnungen zwischendurch aufgerufen.

FENSTER - Ansi-Tabelle

Eine Ansi-Tabelle wird aufgerufen. Zeichen können aus der Ansi-Tabelle übernommen werden.

(Siehe Hilfetext der ANSI-Tabelle)

FENSTER - Kalender

Der Windows-Kalender wird aufgerufen.

FENSTER - 1 ...

Hier sind alle geöffneten Programme aufgelistet. Wenn man die Editorfenster auf optimale Größe vergrößert hat, kann durch diese Menüpunkte einfach zwischen den Fenstern umgeschaltet werden.

HILFE - PROFAN-Hilfe

Die komplette Hilfe zu PROFAN wird aufgerufen. Diese Hilfedatei enthält sowohl die komplette Einführung als Hypertext, als auch die Referenz mit allen Systemvariablen, Funktionen und Befehlen

HILFE - Referenz

In der Referenz ist die komplette Programmiersprache RGH-PROFAN² in der aktuellen Version beschrieben. Im Zweifelsfalle ist diese Referenz aktueller als das entsprechende Handbuch. Für jede Systemvariable, jede Funktion und jeden Befehl findet sich eine Syntaxbeschreibung und häufig auch Beispiele.

HILFE - Editor

Hilfe zur Entwicklungsumgebung. Alle Menüpunkte werden erläutert.

HILFE - Druckprogramm

Hilfe zu allen Menüpunkten des Druckprogrammes.

HILFE - Über

Copyright-Vermerk: 1992,1993 - Roland G. Hülsmann, Mannheim

9 - Kontrollstrukturen

```
Syntax:
Case/CaseNot
If/IfNot ... ElseIf ... Else ... If
While/WhileNot ... Wend
@If
```

PROFAN² bietet im wesentlichen die aus anderen Sprachen bekannten Kontrollstrukturen. Die strukturen im Einzelnen:

INHALT
REFERENZ

9.1 CASE / CASENOT

Der CASE-Befehl stellt letztlich ein einzeiliges IF dar:

```
CASE @Equ(A%,3) : PRINT "A hat den Wert 3"
```

Der hinter dem CASE stehende Befehl bzw. Prozeduraufruf wird nur ausgeführt, wenn der Ausdruck vor dem Doppelpunkt einen von Null verschiedenen Wert hat, d.h. wenn die Bedingung erfüllt wird. Ist die Bedingung nicht erfüllt, geht es direkt mit der nächsten Zeile weiter. Eine Sonderform des CASE ist die Form CASENOT:

```
CASENOT @Equ(A%,3) : PRINT "A ist ungleich drei"
```

Dies ist lediglich eine bequemere (und in der Ausführung schnellere) Schreibweise der Zeile

```
CASE @Not(@Equ(A%,3)) : PRINT "A ist ungleich drei"
```

9.2 IF / IFNOT ...

Mit Ausnahme der Sonderform IFNOT, die für IF @Not steht, gilt hier das aus BASIC Bekannte. Lediglich das THEN gibt es in PROFAN² nicht und wird weggelassen:

```
IF @Equ(A%,1)
      PRINT "A ist gleich 1"
ENDIF
```

WICHTIG: Hinter der Bedingung darf (außer einer Anmerkung mit ') kein Befehl stehen. Das in BASIC und PASCAL bekannte einzeilige IF wird in PROFAN² durch CASE ersetzt.

Natürlich gibt es in PROFAN² auch den ELSE-Zweig, der ausgeführt wird, wenn die Bedingung nicht erfüllt wird:

```
IF @Equ(A%,1)
      PRINT "A ist gleich 1"
      PRINT "--------------"
ELSE
      COLOR 5,15
      PRINT "A ist ungleich 1"
ENDIF
```

Auch eine Abfrage auf mehrere Bedingungen (z.B. bei der Auswertung eines Menüs) ist mit ELSEIF möglich. Sobald eine Bedingung erfüllt ist, wird der zugehörige Programmteil abgearbeitet und anschließend mit der Zeile nach dem ENDIF fortgefahren. Die SELECT- und CASE-Strukturen anderer Sprachen lassen sich somit abbilden:

```
IF @Lt(A%,1)
      COLOR 1,15
      PRINT "A ist kleiner als 1"
ELSEIF @Equ(A%,1)
      COLOR 2,15
      PRINT "A ist gleich 1"
ELSEIF @Equ(A%,2)
      COLOR 3,15
      PRINT "A ist gleich 2"
ELSE
      PRINT "A ist größer als 2"
ENDIF
```

Dieser Programmteil wäre identisch mit folgender Struktur:

```
IF @Lt(A%,1)
      COLOR 1,15
      PRINT "A ist kleiner als 1"
ELSE
      IF @Equ(A%,1)
            COLOR 2,15
            PRINT "A ist gleich 1"
      ELSE
            IF @Equ(A%,2)
                  COLOR 3,15
                  PRINT "A ist gleich 2"
            ELSE
                  PRINT "A ist größer als 2"
```

```
            ENDIF
      ENDIF
ENDIF
```

INHALT
REFERENZ

9.3 WHILE / WHILENOT

Selbstverständlich gibt es auch Schleifen in PROFAN². Der zwischen WHILE und WEND stehende Programmteil wird solange ausgeführt, solange die Bedingung erfüllt ist. Auch bei WHILE ist die Zusammenziehung mit @Not zu WHILENOT erlaubt. Ebenso wie bei Unterprogrammen/Prozeduren ist eine Schachtelungstiefe von 10 erlaubt. In der Praxis dürfte dieser Wert kaum erreicht werden. Ein Beispiel für eine WHILE-Schleife:

```
LET A%=0
WHILENOT @Equ(A%,10)
      INC A%
      PRINT A%;" zum Quadrat ist ";@Squ(A%)
WEND
```

9.3 @IF

In Anlehnung an C gibt es das IF auch als Funktion. Die Funktion @IF hat drei Argumente: Das erste Argument ist die Bedingung; das zweite Argument ist der Wert, der zurückgegeben wird, wenn die Bedingung erfüllt ist und das dritte Argument ist schließlich der Wert, der bei unerfüllter Bedingung zurückgegeben wird. Beispiele:

```
LET Text$ = @IF(@Equ(A%,1), "A ist gleich 1", \
                            "A ist ungleich 1")
PRINT Text$

LET A% = @IF(@Gt(A%,1000), @Add(A%,200), @Add(A%,50) )
PRINT A%
```

Im zweiten Beispiel wird zu A% 200 addiert wenn es größer als 1000 ist; im anderen Fall kommen nur 50 dazu.

10 - Prozeduren

Proc ... Endproc, Parameters,
Return, @$(0), @%(0), @!(0), @&(0)

Prozeduren sind Unterprogramme, die wie neue Befehle behandelt werden können. Unterprogramme mit GOSUB werden daher in Zukunft kaum notwendig sind, da Prozeduren deutlich leistungsfähiger sind:

Eine Prozedur wird einmal - vor ihrer ersten Verwendung - definiert und kann dann jederzeit wie ein neuer Befehl im Programm verwandt werden. Es ist sinnvoll, alle Prozedurdefinitionen vor dem Beginn des Programmes (nach den DECLARE-, DIM-, und DEF-Befehlen) durchzuführen. Häufig verwandte Funktionen (s.u.) und Prozeduren kann man in einer Include-Datei (s.o.) zusammenfassen und sich so beliebige Spracherweiterungen selbst programmieren.

Eine Prozedur-Definition beginnt mit dem Befehl PROC, gefolgt vom Namen der Prozedur. Die Definition endet mit der Zeile ENDPROC. Eine Prozedur wird natürlich während Ihrer Definition nicht ausgeführt, sondern erst, wenn Sie aufgerufen wird. Hier ein komplettes Mini-Programm als Beispiel für eine sehr einfache Prozedur:

```
'Testprogramm
'------------
PROC Titelzeile
      PRINT "Das ist der Titel"
ENDPROC

CLS
PRINT "Das ist noch vor der Prozedur."
TITELZEILE
PRINT "Das ist nach der Prozedur."
WAITKEY
END
```

Einer Prozedur können auch Parameter übergeben werden. Diese werden mit dem Befehl PARAMETERS in die Prozedur übernommen: Dem Befehl folgen die Variablen, denen die Parameter zugewiesen werden. Diese Variablen - wie alle in der Prozedur declarierten Variablen - sind "LOKAL", d.h. sie sind nur in der Prozedur dem System bekannt. Alle außerhalb der Prozedur bekannten Variablen sind in der Prozedur auch bekannt (es sei denn, es gibt lokale Variablen mit gleichem Namen). Ein Beispiel, bei dem eine Prozedur definiert wird, die eine Text wiederholt ausgibt:

```
Testprogramm 2
'--------------
PROC Wiederhole
      PARAMETERS Anzahl%, Text$
      DECLARE I%
      LET I% = 0
      WHILE @Lt(I%,Anzahl%)
            PRINT Text$
      WEND
ENDPROC

CLS
PRINT "Das ist noch vor der Prozedur."
WIEDERHOLE 5, "Testtext"
PRINT "Das ist nach der Prozedur."
WAITKEY
END
```

Die Parameter, die der Prozedur übergeben werden, werden durch Kommata voneinander getrennt. Die Typen und die Anzahl der Parameter werden von PROFAN² nicht überprüft. Wegen der automatischen Typenumwandlung ist das auch nicht notwendig. Es sind maximal 8 Parameter erlaubt. Selbstverständlich können in Prozeduren wieder neue Prozeduren definiert werden. Diese sind dann natürlich auch nur lokal bekannt. Hier gilt sinngemäß das Gleiche, wie für Variablen.

Eine Prozedur kann in PROFAN² auch einen Wert zurückgeben: Innerhalb der Prozedur darf der RETURN-Befehl vorkommen, wenn die Prozedur vor dem ENDPROC verlassen werden soll. Dem RETURN kann ein Parameter zurückgegeben werden, der je nach Typ mit der Funktion @$(0), @%(0), @&(0) oder @$(0) gelesen werden kann. (Diese Funktionen sind der "Parameterstack" von PROFAN². Für jeden Datentyp gibt es die Parameter 0 bis 8, wobei der 0. Parameter den Rückgabewert und die übrigen die übergebenen Parameter enthalten. Im obigen Beispiel könnte also auch über @$(2) auf den übergebenen Text zugegriffen werden.) Ein Beispiel für die Rückgabe eines Wertes aus einer Prozedur:

```
'Testprogramm 3
'--------------
PROC Titelzeile
      PRINT "Das ist der Titel"
      RETURN "Ok"
ENDPROC

CLS
PRINT "Das ist noch vor der Prozedur."
TITELZEILE
PRINT "Das ist nach der Prozedur: ";@$(0)
WAITKEY
END
```

Auch wenn eine Prozedur mit RETURN verlassen wird, darf das ENDPROC nicht fehlen!

INHALT
REFERENZ

11 - Definierte Funktionen

Def @<Name>, @$(n), @%(n), @!(n), @&(n)

PROFAN² bietet nun auch die Möglichkeiten, selbst Funktionen zu definieren, die im Programm wie die eingebauten Funktionen verwandt werden können. Eine Funktionsdefinition beginnt mit dem Befehlswort DEF. Diesem folgt der Name der Funktion, der natürlich wie alle PROFAN²-Funktionen mit einem @ beginnen muß. (Das @ erreicht man im Übrigen mit der Tastenkombination <AltGr><Q>.) Nach dem Namen folgt in Klammern die Anzahl der Parameter. Dann kommt der Ausdruck, der die Funktion beschreibt. Die Funktionsparameter werden über den "Parameterstack" (s.o. unter Prozeduren) übergeben. Mit @$(n), @%(n), @&(n) bzw. @!(n) werden die Parameter gelesen, wobei n für die Nummer des Parameters (von 1 - 8) steht. Ein Beispiel für eine Funktion zur Errechnung der dritten Potenz einer Zahl:

```
DEF @Pot3(1) @Mul(@Squ(@!(1)),@!(1))
```

Das Quadrat des ersten Parameters wird mit dem Parameter multipliziert. Aufgerufen wird die Funktion, wie jede andere auch:

```
LET I% = 0
WHILE @Lt(I%,10)
      INC I%
      PRINT I%;"³ = ";@Pot3(I%)
WEND
```

Als zweites Beispiel dient eine Funktion, die die ersten n Zeichen eines Strings zurückgibt. In BASIC heißt diese Funktion LETF$. Da es diese in PROFAN² nicht gibt, definieren wir sie uns:

```
DEF @Letf$(2) @Mid$(@$(1),@%(2))
```

Wenden wir Sie an:

```
DECLARE A$
LET A$ = "Testbild"
PRINT @Left$(A$,4)
```

Auch hier findet bewußt keine Typüberprüfung statt, da alle Typen automatisch ineinander umgewandelt werden. Sehr wohl überprüft wird aber die Anzahl der Parameter! Wie bei allen anderen Funktionen erfolgt eine Fehlermeldung, wenn zuwenig oder zuviel Parameter angegeben werden.

Unter Verwendung der @IF-Funktion und/oder der Dialogbox-Funktionen sind sehr komplexe Funktionsdefinitionen möglich. Funktionsdefinitionen über mehrer Programm-Zeilen werden von PROFAN² nicht unterstützt.

INHALT
REFERENZ

12 - Einige Worte zu GOTO und GOSUB

Label:, Goto,
Gosub, Return

Natürlich kennt PROFAN² auch das GOTO und in PROFAN 1.x war das GOSUB noch die einzige Möglichkeit, Unterprogramme zu erzeugen. Da das GOTO aber der erklärte Feind jeder strukturierten Programmierung ist, sollte man es besser vermeiden. Dies ist auch problemlos möglich.

GOTO hat als Parameter einen String oder Stringausdruck. Mit GOTO springt der Programmlauf zu dem Label (Sprunkmarke), das dem String entspricht:

```
GOTO "Start"
PRINT "Diese Zeile wird nie geschrieben."
Start:
PRINT "Hier fängt das Programm an!"
```

Das Label muß alleine in einer Zeile stehen und wird mit einem Doppelpunkt abgeschlossen.

Auch GOSUB hat einen String oder Stringausdruck als Parameter. Mit GOSUB wird ein Unterprogramm aufgerufen, das beim entsprechenden Label beginnt und mit RETURN endet. Auch hier kann wie in der Prozedur ein Wert an das aufrufende Programm zurückgegeben werden. Im Gegensatz zu BASIC kennt das Unterprogramm auch lokale Variablen: Alle Variablen, die im Unterprogramm mit DECLARE definiert werden. Im Gegensatz zur Prozedur darf das Unterprogramm auch hinter dem Aufruf des Unterprogrammes stehen und außerdem muß der Programmierer selbst darauf achten, daß das Programm nicht versehentlich in das Unterprogramm hineinläuft. Ein Beispiel für ein Unterprogramm:

```
PRINT "Noch im Hauptprogramm .."
GOSUB "Unterprogramm"
PRINT "... und wieder zurück"
WAITKEY
END

Unterprogramm:
      PRINT "..."
      PRINT "... und jetzt im Unterprogramm"
      PRINT
RETURN
```

Eine Berechtigung hat das GOSUB vielleicht, da als Sprungziel ein Ausdruck erlaubt ist, also "berechnete Sprungziele" möglich sind. Bei seltenen Aufgabenstellungen mag man dies nutzen. Übersichtlich und wartbar wird ein solches Programm aber kaum sein.

Wer die Beispielprogramme zu PROFAN² angesehen hat wird jetzt sagen: Da wimmelt es ja nur so von GOTO und GOSUB. Das muß ich zugeben, aber es hat seine Gründe:

Ein frühes Vorbild für PROFAN war ORIEL. PROFAN sollte alles können, was die Batchsprache ORIEL auch konnte. So waren die ersten PROFAN-Programme umgeschriebene ORIEL-Programme. ORIEL kannte aber keine Unterprogramme, sodaß nicht einmal ein GOSUB möglich war. Die Menüprogramme und das Hypertextprogramm stammen aus dieser Zeit. Sie waren aber trotz GOTO klar strukturiert, sodaß ein Umschreiben ohne GOTO und GOSUB sicher eine interessante Übung wäre.

Als PROFAN in der ersten Version fertig war, gab es zwar schon das GOSUB und auch WHILE-Schleifen, aber die Prozeduren sind erst bei PROFAN² 2.0 dazugekommen. Ein großer Teil der Beispielprogramme war schon bei PROFAN 1.x dabei. Aber damit die Anwender nicht allzulange auf die neue Version warten mußten, wurden nur die notwendigen Anpassungen vorgenommen und mehr Wert auf zusätzliche

Features gelegt, als auf Eliminierung aller Labels. Aber es ist bei allen Programmen möglich. Als Beispiel diene das Programm DEMO.PRF, das schon komplett umgestellt ist.

Kurz: Der Leser verzeihe mir meine Faulheit!

HINWEIS: Da ich ständig an der Aktualisierung der Demo-Programme arbeite, kann es sein, daß bei Ihnen das eine oder andere Demo-Programm schon umgestellt wurde und GOTO- und GOSUB-frei ist. Änderungen und Anpassungen bei den DEMO-Programmen erfolgen ohne Änderung der Versionsnummer und ohne weitere Hinweise.

13 - Programm-Aufbau

Alle Variablen in PROFAN müssen vorher (wie etwa in PASCAL) declariert werden. Die DECLARE- und DIM-Befehle sollten am Anfang des Programmes stehen. Lokale Variablen sollten am Anfang einer Prozedur definiert werden. Vor der ersten Bildschirmausgabe sollte immer ein CLS stehen, um das Bildschirmfenster zu öffnen. (Nicht alle Befehle nehmen sich die Zeit, dies zu überprüfen, um dann selbst das Fenster zu öffnen.)

BEISPIEL:

```
'DEKLARATIONSTEIL
'----------------

DECLARE ...
DIM% ...
DIM$ ...

'INCLUDE-DATEIEN MIT FUNKTIONEN UND PROZEDUREN
'--------------------------------------------

$I TOOLBOX.INC

'DEFINITION VON FUNKTIONEN
'-------------------------

DEF @...

'PROZEDUREN
'----------

PROC ...
      PARAMETERS ...
      DECLARE ...
      ...
ENDPROC

'DAS HAUPTPROGRAMM
'-----------------

WINDOWTITLE ...
WINDOWTYPE ...
WINDOW ...
CLS
...
...
...
END
```

INHALT
REFERENZ

14 - Befehle über mehrere Zeilen

Der Übersichtlichkeit wegen können Befehle (Programmzeilen) auch über mehrere Bildschirmzeilen geschrieben werden. Ist das letzte Zeichen einer Zeile ein \ , so wird die nächste Zeile ab dem ersten Zeichen, das kein Leerzeichen ist, angehängt. Beispiele:

```
MESSAGEBOX "Überschrift",\
           "Das ist ein Testtext, der \
            etwas länger ist!",16

PRINT "Das ist ein ziemlich lang\
             er Text!"

COPYSIZEDBMP 34,34-16-16 > \
             12,12-32,32 ; 0
```

Beachten Sie: Auch eine Programmzeile über mehrere Bildschirmzeilen darf nicht länger als 127 Buchstaben sein. Führende und folgende Leerzeichen werden nicht gezählt.

INHALT
REFERENZ

15 - Fehlerbehandlung

%Error, %IOResult, %MCIError,
SeSetErrorLevel

Fehlermeldungen und Warnungen werden in deutsch in einer Messagebox auf den Bildschirm gebracht. Nach einer Fehlermeldung bricht das Programm ab, nach einer Warnung läuft es weiter, wenn Sie "OK" anklicken oder bricht ab, wenn Sie "Abbrechen" wählen. Neben der Fehlerbeschreibung wird beim Interpreter auch die fehlerhafte Zeile angegeben, sodaß der Fehler genau lokalisiert werden kann. Zusätzlich wird (auch wenn der Fehler im compilierten Programm auftritt) die Zeilennummer angegeben. Da aber Includedateien mitgezählt werden und andererseits mit dem Backslash "\" verbundene Zeilen als eine gezählt werden, kann diese Zahl nur als Anhaltspunkt dienen. Auf alle Fälle ist es sehr dringend anzuraten, jegliches Programm im Interpretermodus auf Herz und Nieren zu testen, bevor man es compilert startet.

Das Fehlerverhalten kann über den Befehl SETERRORLEVEL gesteuert werden. Der Befehl hat folgende mögliche Parameter:

Tabelle ErrorLevel

Warnungen treten auf, wenn ein Ausdruck nicht als numerischer Wert zu interpretieren ist oder z.B. eine Bilddatei nicht gefunden wird oder BITMAP- Befehle wegen Speicherplatzmangels nicht ausgeführt werden ... also immer dann, wenn keine ernstzunehmenden Folgen für das restliche Programm zu erwarten sind.

Unabhängig vom gesetzten Errorlevel wird die Systemvariable %ERROR bei einer Warnung bzw. einem Fehler gesetzt. Auf diese Weise kann also der Programmierer die komplette Verantwortung zu Behandlung von Fehlern übernehmen. Wie auch %IORESULT wird %ERROR beim ersten Auslesen wieder auf 0 gesetzt. Die möglichen Error-Werte:

Tabelle %Error

ACHTUNG: Bei Dateioperationen wird nicht %ERROR verwandt, sondern die Systemvariable %IORESULT gesetzt, das Programm aber in keinem Fall abgebrochen. Das Programm muß also diese Variable auswerten, da ansonsten Abstürze möglich sind.

Nach einer Dateioperation oder einer Directory-Suche enthält diese Variable den entsprechenden Wert:

Tabelle %IOResult

Eine dritte Art Fehler kann bei der Ansteuerung von Multimediageräten über die MCI-Schnittstelle erfolgen. Der zuletzt gemeldete Fehler beim Senden eines MCI- Strings zur Ansteuerung eines Multi-Media-Treibers (etwa CD, Soundkarte, MIDI- Gerät, etc.) wird in der Systemvariable %MCIERROR festgehalten. Wie auch %ERROR und %IORESULT wird sie beim ersten Auslesen auf 0 zurückgesetzt. Die Bedeutung der Fehlernummern hängt vom MCI-Gerätetreiber ab und ist für sich daher wenig aussagekräftig. Für den Programmierer bedeutet ein Wert ungleich Null lediglich, daß ein Fehler aufgetreten ist und die Funktion @MCISEND$ nicht einen Ergebnisstring zurückliefert, sondern eine Fehlermeldung (in der Landessprache).

HINWEIS: Sollte bei Ihnen ein "Runtime-Error 202" auftauchen, so bedeutet dies einen Stacküberlauf. Dieser kann dann auftreten, wenn Sie Funktionen zu tief verschachtelt haben. Lösen Sie die Funktionen in mehrere Einzelfunktionen auf. Da ist sowiesio übersichtlicher.
Dieser Fehler kommt nicht von Windows, sondern vom Entwicklungssystem, mit dem ich PROFAN² entwickelt habe. Ihre Windows-Umgebung kann durch diesen Fehler nicht beeinträchtigt werden.

INHALT
REFERENZ

Tabelle: ErrorLevel

2 -Für ganz Vorsichtige: Warnungen werden wie Fehler behandelt und führen zu einer Fehlermeldung mit Programmabbruch.

1 -Für die Programmentwicklung: Auch Warnungen werden ausgeben, aber das Programm läuft auf Wunsch weiter.

0 -Der Normalzustand: Warnungen werden nicht angezeigt. Diesen Errorlevel sollte man bei einem fertigen Programm verwenden.

-1 - Fast schon kriminell: Auch Fehlermeldungen werden übergangen.Das kann unter Umständen zu einem Windowsfehler oder Absturz desSystemes mit Datenverlust führen.

Tabelle: %Error

0 - seit dem letzten Lesen von %ERROR ist kein Fehler aufgetreten
1 - eine Warnung wird gemeldet
2 - ein Fehler wird gemeldet

Tabelle: %IOResult

0 - kein Fehler aufgetreten
2 - Datei nicht gefunden
3 - Pfad nicht gefunden
5 - Zugriff verweigert (ReadOnly?)
12 - Ungültiger Dateimodus
15 - Laufwerksnummer unzuzulässig
16 - Verzeichnis kann nicht gelöscht werden (noch Dateien drin?)
17 - RENAME nicht über Laufwerksgrenzen möglich (siehe Rename)
18 - Kein weiterer Eintrag (bei @FindFirst/@FindNext)
100 - Lesefehler von Diskette/Platte
101 - Schreibfehler auf Diskette/Platte
102 - Dateinummer ist keiner Datei mit ASSIGN zugeordnet (siehe Assign)
103 - Datei nicht offen (Reset, Rewrite oder Append fehlt)
104 - Datei nicht zum Lesen geöffnet (Reset fehlt)
105 - Datei nicht zum Schreiben geöffnet (Rewrite oder Append fehlt)
106 - Falsches Format (bei Input #N,..)

16 - Text- und Grafikmodus

PROFAN kennt (für Windows unüblich) einen Text- und einen Grafikmodus. Da Windows natürlich nur einen Grafikbildschirm hat, wird der Textmodus auf diesem simuliert.

Der Grafikmodus ist der Windowseigene Modus, der nahezu all die Möglichkeiten bietet, die das Windows-Grafik-Interface beeinhaltet.

Der Textmodus simuliert im Hauptfenster einen DOS-Bildschirm, der zeichenorientiert ist.

Es gibt keinen SCREEN-Befehl, um zwischen den Modi umzuschalten, da immer beide gleichzeitig verfügbar sind.

INHALT
REFERENZ

16.1 - Der "Textmodus"

Print, Locate, Input,
Color, Cls, TBox, Font,
Decimals, StrWidth, NumWidth,
WaitKey, WaitInput, @TMouse,
%Key, %MouseKey

Im Textmodus funktionieren die von BASIC her bekannten Befehle, wie z.B. PRINT, LOCATE, INPUT, COLOR und CLS in gewohnter Weise. Zusätzlich sind in diesem Modus Befehle wie TBOX, FONT und @TMOUSE verfügbar. Ein wesentlicher Unterschied ist, daß CLS den BIldschirm nicht in der aktuellen Farbe löscht, sondern grundsätzlich in weiß. Bei INPUT ist kein Prompt-Text erlaubt, sondern nur die Angabe einer Variable. Ein kleines Beispiel:

```
Declare Zahl%, Text$
WindowTitle "Testprogramm"
Cls
Locate 2,2
Print "Das ist die Überschrift"
Font 2
Locate 4,2
Print "Bitte gebe einen Text ein: ";
Input Text$
Print "Bitte gebe eine Zahl ein: ";
Input Zahl%
Font 0
Print
Color 1,15
Print "Du hast eingeben:",Zahl%,Text$
WaitKey
End
```

Das Semikolon nach einem Ausdruck beim Print-Befehl sorgt dafür, daß die nächste Ausgabe in die gleiche Zeile geht und direkt an die vorherige anschließt. Bei einem Komma wird hingegen ein Leerzeichen gelassen. (Die von BASIC her bekannte Tabulatorfunktion gibt es nicht.) Der Befehl WAITKEY wartet auf einen Tastendruck. Möchte man wissen, welche Taste gedrückt wurde, kann man die Systemvariable %KEY abfragen. Ebenso hätte man den Befehl WAITINPUT nutzen können, dann werden zusätzlich die Maustasten abgefragt. Das Ergebnis findet sich in der Systemvariablen %MOUSEKEY, wobei 1 für die linke und 2 für die rechte Maustaste steht. (In diesem Zusammenhang sei auch auf die Befehle WAITSCAN und WAITMOUSE, sowie die Funktionen @KEYIN, @INKEY$ und @GETKEY$ hingewiesen.)

Mit dem Befehl FONT kann der Bildschirmfont im Textmodus eingestellt werden (0 = System, 1 = OEM (ASCII), 2 = ANSI). COLOR stellt die Textfarbe ein, wobei wie in BASIC Farbwerte von 0 bis 15 erlaubt sind.

Oftmals möchte man Zahlen und Texte formatiert ausdrucken. Dies ist in PROFAN² mittels der Befehle NUMWIDTH, STRWIDTH und DECIMALS möglich. Mit DECIMALS wird festgelegt, wieviele Dezimalstellen bei der Ausgabe mit PRINT (und auch bei der Umformung mit @STR$) berücksichtigt werden:

```
Declare Kreiszahl!
Let Kreiszahl! = @PI()
Decimals 0
Print Kreiszahl!
```

```
Decimals 5
Print Kreiszahl!
Decimals 10
Print Kreiszahl!
WaitKey
End
```

Voreingestellt ist ein Wert von 6, d.h. wenn nichts anderes angegeben wird, werden 6 Ziffern hinter dem Komma ausgegeben. Mit NUMWIDTH kann ich festlegen, wieviele Zeichen mindestens gedruckt werden. Ist die Zahl kürzer, wird sie mit führenden Leerzeichen ausgegeben. Das ermöglicht die Ausgabe von Tabellen:

```
Decimals 3
Numwidth 2
Print @PI()
Numwidth 6
Print @PI()
Numwidth 8
Print @PI()
WaitKey
End
```

Gleiches ist mit STRWIDTH für die Ausgabe von Strings möglich. Beide Werte sind mit 0 vorbelegt, sodaß normalerweise alle Zahlen und Texte ohne führende Leerzeichen ausgegeben werden.

Auch im Textmodus kann mit (wenn auch simulierten) "Buttons" gearbeitet werden, wobei abgefragt werden kann, ob ein Maus-Klick in diesem Button erfolgte. Ein kleines Beispielprogramm:

```
Declare Ende%

Cls
' Der Button "A" wird gezeichnet
Color 15,2
TBox 10,10 - 15,15; 3
Locate 13,13
Print "A"
' Der Button "B" wird gezeichnet
Color 15,3
TBox 10,20 - 15,25; 0
Locate 13,23
Print "B"
Color 15,0

Let Ende% = 0
WhileNot Ende%    'Solange Ende den Wert 0 hat ...
      WaitInput     'Auf Tastendruck oder Mausklick warten
      If @TMouse(10,10 - 15,15) 'Mausklick im Button A?
            Locate 20,0
            Print "Du hast 'A' gedrückt!"
            Let Ende% = 1
      ElseIf @TMouse(10,20 - 15,25) 'Mausklick in B?
            Locate 20,0
            Print "Du hast 'B'' gedrückt!"
            Let Ende% = 1
      ElseIf @KeyIn("Aa") 'Taste "A"?
```

```
            Locate 20,0
            Print "Du hast 'A' gedrückt!"
            Let Ende% = 1
      ElseIf @KeyIn("Bb") 'Taste "B"?
            Locate 20,0
            Print "Du hast 'B'' gedrückt!"
            Let Ende% = 1
      Else 'Nichts von alledem ...
            Locate 20,0
            Print "Daneben ..."
      EndIf
Wend
WaitKey
End
```

Der Befehl TBOX zeichnet in der mit COLOR eingestellten Farbe ein Rechteck zwischen den angegebenen Koordinatenpaaren. Wie auch bei LOCATE ist die jeweils erste Koordinate die Zeile und die zweite die Spalte. Der fünfte Parameter gibt die Art des Rahmens an: 0 - einfacher Rahmen; 1 - doppelter Rahmen; 2 - teils einfach, teils doppelt; 3 - kein Rahmen. Der Rahmen wird mit ASCII-Grafikzeichen gemalt. Die Funktion @TMOUSE hat dann den Wert 1, wenn der letzte Mausklick in dem durch die zwei Koordinatenpaare angegebenen Rechteck erfolgte. Die Funktion @KEYIN hat den Wert 1, wenn die zuletzt gedrückte Taste im Funktionsargument (ein String) enthalten ist. Das Beispielprogramm demonstriert, wie man Programme schreibt, die sowohl auf die Maus als auch auf die Tastatur reagieren.

INHALT
REFERENZ

16.2 - Der "Grafikmodus"

```
TextColor, @RGB,
%WinTop, %WinBottom, %WinLeft, %WinRight,
CharSet, Orientation, UseFont,
DrawText, UsePen, UseBrush,
Rectangle, RoundRect, Ellipse,
Chord, Pie, Arc,
SetPixel, @GetPixel
MoveTo, Line, LineTo, Fill
```

Dieses sind die windowseigenen Text- und Grafikbefehle. Der fortgeschrittene Programmierer wird diesen Befehlen den Vorzug vor den textorientierten Befehlen geben. Alle Koordinatenangaben erfolgen hier in Pixel, also im Bereich 0,0 (links oben) bis z.B. bei Standard-VGA und Vollbildmodus 639,479 (rechts unten).

Wenn man mit WINDOW eine andere Größe eingestellt hat, ist der Bereich natürlich entsprechend kleiner bzw. größer. Die augenblickliche Fensterposition und -größe läßt sich mit den Systemvariablen %WINTOP, %WINBOTTOM, %WINLEFT und %WINRIGHT ermitteln, wobei diese die Koordinaten in Bezug zur linken oberen Bildschirmecke angeben. Um z.B. die tatsächliche Breite des Fensters zu ermitten, ist ein wenig Rechnen angesagt: `Let WinWidth% = %WinRight - %WinLeft`

WICHTIG: Vor jedem der Ausgabe-Befehle (dazu gehören auch alle USE-Befehle) muß das Programmfenster geöfnet sein. Hierzu verwenden wir den Befehl CLS. Mit den Befehlen WINDOW, WINDOWSTYLE und WINDOWTITLE können wir zuvor die Attribute unseres Fensters festlegen!

Zunächst einmal wollen wir Schrift auf den Bildschirm bringen. Dazu legen wir die Textfarbe mit dem Befehl TEXTCOLOR fest. Der erste Parameter ist die Vordergrundfarbe und der zweite die Hintergrundfarbe. Beide Werte sind Farbwerte im Bereich von 0 bis 32767. Da man sich unmöglich merken kann, welche Farbe welchen Wert hat, gibt es die Funktion @RGB. Diese hat drei Parameter, die für den ROT-, GRÜN- und BLAU-Anteil der Farbe stehen. Sind alle drei Parameter auf 0, ist die Farbe schwarz, sind alle auf 31 gibt es ein reines Weiß. Eine Besonderheit gilt noch für die Hintergrundfarbe: Ist diese -1, so bedeutet das einen transparenten Hintergrund; d.h. der Hintergrund wird zwischen den Buchstaben sichtbar. Als nächstes müssen wir die Schriftart, den Font festlegen. Unter PROFAN² können wir alle Windows-Fonts einschließlich der True-Type-Fonts anwenden. Zur Einstellung eines Fonts dient der USEFONT- Befehl, der eine Reihe von Parametern hat: Der erste Parameter ist der Name des Fonts, z.B. "ARIAL" oder "COURIER NEW", wie er auch in der Font-Auswahl anderer Programme erscheint. Der zweite und der dritte Parameter geben Zeichenbreite und -höhe in Punkt an. Wird 0 angegeben, wird ein Defaultwert benutzt. der 4. bis 6. Parameter sind Schalter, die angeben, ob der Font unterstrichen, kursiv und/oder fett ausgegeben werden soll. Die Parameter im Überblick:

1. S : String - Fontname
2. N1 : Integer - Zeichenbreite
3. N2 : Integer - Zeichenhöhe
4. N3 : Integer - Unterstrichen? (0 .. 1)
5. N4 : Integer - Kursiv? (0 .. 1)
6. N5 : Integer - Fett? (0 .. 1)

Für TrueType-Fonts ist noch der Befehl ORIENTATION interessant. Hier kann in 10tel Grad eingestellt werden, in welche Richtung der Text ausgegeben wird. Der ORIENTATION-Befehl muß vor (!) dem USEFONT-Befehl kommen, damit er wirkungsvoll wird. Außerdem gibt es noch den CHARSET-Befehl, der dem USEFONT-Befehl mitteilt, welcher Zeichensatz zu wählen ist, wenn der verlangte Font nicht vorhanden ist. In diesem Fall sucht Windows nämlich einen möglichst ähnlichen Font aus ... und der sollte den gleichen Zeichensatz haben. Angezeigt wird der Text schließlich mit dem DRAWTEXT-Befehl, dessen erste beiden Parameter die Koordinaten sind. Ein kleines Beispiel:

```
Cls
TextColor @RGB(31,0,0),-1
CharSet 0
Orientation 300 ' 30°-Winkel
UseFont "ARIAL",30,40,1,0,1
DrawText 30,300,"Dies ist ein Text"
WaitKey
```

Ab Version 2.5 von PROFAN² kann auch eine numerische Variable bzw. Systemvarible mit DRAWTEXT ausgegeben werden, wobei zur Formatierung die Einstellungen mit NUMWIDTH und DECIMALS herangezogen werden.

Selbstverständlich können wir mit PROFAN² auch Punkte und Linien zeichnen. Einen Punkt setzen wir mit dem SETPIXEL-Befehl. Als Parameter übergeben wir die Koordinaten und die Farbe:

```
SetPixel 123,200,@RGB(13,20,3)
```

Wenn Ihr Gafiktreiber nicht alle in PROFAN² möglchen 32768 Farben unterstützt, wird der Punkt in der nächstmöglichen Farbe gesetzt. Ausgelesen werden kann ein Bildpunkt mit der Funktion @GETPIXEL, die den tatsächlichen Farbwert eines Punktes ermittelt:

```
LET A% = @GetPixel(123,200)
```

Vor dem Lienienziehen müssen wir festlegen, welchen Stift wir verwenden. Das erledigt der Befehl USEPEN. Der erste Parameter ist die Linienart, der zweite die Linienstärke und der dritte die Farbe. Die Linienarten 1 bis 5 (gestrichelt und gepunktet) sind nur bei Linienstärke 1 sinnvoll, d.h. dickere Linien werden immer durchgezogen dargestellt. Der eingestellte Stift gilt für alle (!) folgenden Grafikbefehle, mit Ausnahme von SETPIXEL.
Eine Linie ziehen wir mit dem Befehl LINE, gefolgt von zwei Koordinatenpaaren. Zusätzlich gibt es noch den Befehl MOVETO, mit dem der Grafik-Kursor versetzt wird, ohne eine Linie zu zeichnen und den Befehl LINETO, der eine Linie vom Grafik-Kursor zum angegebenen Punkt zeichnet. Probieren Sie folgendes Beispiel aus:

```
Cls
UsePen 2,1,@RGB(0,31,0)
Line 20,20 - 100,100
LineTo 100,20
MoveTo 120,20
LineTo 120,60
WaitInput
```

Zusätzlich gibt es noch Befehle, um Flächen-Objekte (Rechtecke, Kreise, etc.) zu zeichnen. All diesen ist gemeinsam, daß die ersten beiden Koordinatenpaare das Rechteck angebeben, in dem das Objekt Platz findet. Außerdem muß der Pinsel definiert werden, mit dem das Objekt ausgemalt wird. Diese geschieht mit dem Befehl USEBRUSH. Dabei ist der erste Parameter die Art des Pinsels und der zweite die Farbe. Als Art des Pinsels kommen 0 (transparent = nicht ausgefüllt), 1 (voll ausgefüllt) und 2 - 8 (verschiedene Schraffuren) in Betracht. Die Umrandung des Objektes wird durch den Stift mit USEPEN bestimmt. Soll kein Rand gezeichnet werden, muß ein Stift mit Linienstärke 0 definiert werden. Folgende Befehle für Grafikobjekte gibt es: RECTANGLE (Rechteck), ROUNDRECT (mit abgerundeten Ecken), ELLIPSE (auch für Kreise), CHORD (Kreisabschnitt), ARC (Kreisbogen) und PIE (Kreisausschnitt: "Tortenstück"). Ein kleines Beispiel, um ein rotes Rechteck mit blauem Rand zu zeichnen, dazu ein ebensolcher Kreis:

```
Cls
UsePen 1,4,@RGB(0,0,31)
UseBrush 1,@RGB(31,0,0)
```

```
Rectangle 30,30 - 200,200
Ellipse 100,100 - 300,300
WaitInput
```

Der Befehl ROUNDRECT erwartet noch ein Parameterpaar zu Definition der Abrundung und die Befehle ARC, CHORD und PIE erwarten jeweils noch ein Koordinatenpaar für den Beginn und eines für das Ende des betreffenden Ab- bzw. Ausschnittes. Näheres dazu finden Sie in der Referenz. Machen Sie ruhig ein paar Experimente und Sie finden sich schnell in der Windowsgrafik zurecht! Es sei an dieser Stelle auch noch auf den Befehl FILL hingewiesen, mit dem beliebige umrandete Flächen ausgefüllt werden können.

INHALT
REFERENZ

17 - Bitmaps

LoadBmp, LoadSizedBmp,
CopyBmp, CopySizedBmp, SaveBmp,
DrawIcon, DrawSysIcon, UseIcon

Wichtig unter Windows sind natürlich die Bilddateien. PROFAN² unterstützt hier das Windowseigene Bitmap-Format, gekennzeichnet durch die Endung BMP! Die Windows-Hintergrundbilder sind in diesem Format abgelegt und nahezu jedes Grafikprogramm unter Windows versteht dieses Format, so auch das mit Windows mitgelieferte Programm Paintbrush.

Mit dem Befehl LOADBMP wird eine Grafik geladen und in Originalgröße angezeigt. Wir müssen also den Namen des Bildes wissen und wohin auf dem Bildschirm es ausgegeben werden soll. Beispiel:

```
LoadBmp "PAPER.BMP", 10, 10; 0
```

Der letzte Parameter ist der Abbildungsmodus. Die 0 steht für unveränderte Abbildung. Andere Werte bedeuten Verknüpfungen mit dem Hintergrund oder negative Darstellung (z.B. OR, AND oder XOR-Verknüpfung). Soll das Bild nicht in der Origionalgröße angezeigt werden, sondern in einer genau festgelegten Größe, so verwenden wir den Befehl LOADSIZEDBMP, der zusätzlich noch Parameter über Breite und Höhe des Bildes benötigt:

```
LoadSizedBmp "C:\WIN31\CHESS.BMP", 10,10 - 100, 80; 0
```

Es können auch beliebige Bildschirmbereiche von einem Ort zum anderen kopiert werden. Auch hier gibt es die Möglichkeit in der Originalgröße zu kopieren oder aber die Größe zu verändern (Lupenfunktion):

```
CopyBmp 10,10 - 100,80 > 200,200; 0
CopySizedBmp 10,10 - 100,80 > 200,200 - 200,160; 0
```

Eine weitere - in PROFAN² hinzugekommene - Möglichkeit ist das Abspeichern eines beliebigen Bildschirmausschnittes als Bitmap-Datei:

```
SaveBmp "D:\GRAFIK\TEST.BMP", 10,10 - 100,80
```

Desweiteren gibt es natürlich auch in PROFAN² die kleinen Symbolbilder, die Icons. Zum einen stehen die Windows-System-Icons zur Verfügung. Diese werden über eine Nummer angesprochen und an beliebiger Stelle angezeigt:

```
DrawSysIcon 3, 10,50
```

Einige weitere Icons sind in PROFAN² eingebaut, um diese auszugeben, muß man deren Namen wissen:

```
DrawIcon "GESICHT", 30,80
```

Hier sei noch der Hinweis angebracht, daß ein Icon eine feste Größe von 32 * 32 Pixel hat. Die in PROFAN² vorhandenen Icons können mit einem Resourcen-Editor und manchem Icon-Malprogramm verändert und teilweise sogar umbenannt werden. Sie können mit diesen Werkzeugen sogar Icons hinzufügen. WICHTIG: Bei als EXE-Datei gelinkten PROFAN²-Programmen dürfen weder Icons entfernt noch hinzugefügt werden. Das Programm würde dann nicht mehr Funktionieren. Den PROFAN²-Interpreter oder das Runtime-Modul können Sie beliebig um Icons erweitern. Eine EXE-Datei bekommt immer die Icons die das Runtime-Modul zur Zeit des Linkens hat. Ein Hilfsmittel um dieses zu tun, liegt dem PROFAN²-Paket als Shareware bei.

Ein letzter Hinweis zu den Icons: Auch das Icon, das ein PROFAN²-Programm bei Verkleinerung zum Symbol erhält kann eingestellt werden. Hierfür dient der Befehl USEICON:

```
UseIcon "GESICHT"
```

INHALT
REFERENZ

18 - Sounds

Beep, Sound,
Play, Music, PlaySound

Wenn auch die meisten Windowsprogrammiersprachen aus unerfindlichen Gründen keine Befehle zur Tonerzeugung vorsehen (so. z.B. WinBasic, Visual Basic, TurboPascal für Windows, ...), bietet PROFAN² hier einiges an:

Zunächst einmal ist das der aus BASIC bekannte BEEP-Befehl, der einen kurzen Ton erzeugt. Mehr Möglichkeiten bietet da schon der SOUND-Befehl, dem als Parameter die Frequenz (in Hz) und die Dauer (in 18tel Sekunden) übergeben wird:

```
SOUND 440,18
```

Diese Zeile erzeugt eine Sekunde lang den Kammerton A! Musikalisch wird es mit dem PLAY-Befehl, der als Parameter den Notenwert (in Halbtonschritten), die Notenlänge und die Punktierung erwartet. Der Notenwert beginnt beim tiefsten Ton mit 1. Eine Oktave höher ist dann der Ton 13. Ton 0 steht für die Pause. Die Notendauer wird reziprok angegeben, d.h. eine Viertelnote hat die Notendauer 4, eine Achtelnote die Notendauer 8, etc. Schließlich folgt die Anzahl der Punkte. Eine Punktierte Achtelnote würde z.B. mit folgender Zeile gespielt:

```
PLAY 16,8,1
```

Noch musikalischer ist der MUSIC-Befehl. Als Parameter hat er einen String in einer eigenen Makrosprache. Diese Makrosprache ist kompatibel zur Music-Makro-Sprache, wie sie von BASIC her (für den PLAY-Befehl) bekannt ist. Um eine Tonleiter in Viertel-Noten in der 4. Oktave zu spielen, reicht folgender Befehl:

```
MUSIC "O4 C4 D4 E4 F4 G4 A4 B4 > C4"
```

Erhöhungs- und Erniedrigungszeichen sind genauso möglich, wie auch Punktierungen:

```
MUSIC "C#2 F-8 G4."
```

Pausen haben den Notenwert "P".

Wer eine Soundblasterkompatible Karte hat oder auf andere Weise die Windows- Sounddateien mit der Endung WAV hat, wird sich über den PLAYSOUND-Befehl freuen. Jede WAV-Datei kann damit abgespielt werden:

```
PLAYSOUND "SIRENE.WAV", 1
```

Der letzte Paramater bestimmt den Abspiel-Modus. Sinnvolle Parameter sind:

0 - Die Datei wird im Vordergrund gespielt. Das Programm wird solange angehalten.

1 - Die Datei wird im Hintergrund gespielt. Das Programm läuft weiter.

9 - Die Datei wird im Hintergrund gespielt und solange wiederholt, bis ein neuer PLAYSOUND-Befehl kommt.

INHALT
REFERENZ

Tabelle: Abbildungsmodus

0 - normales Kopieren
1 - Quelle und Ziel mit UND verknüpft
2 - Quelle und Ziel mit ODER verknüpft
3 - Quelle und Ziel mit XOR verknüpft
4 - Zielbereich invertieren (Quellbereich wird nicht berücksichtigt)

Tabelle: PROFAN²-Icons

PROFAN	KNOPF1	EIS
BAUM	KNOPF2	COMPUTER
WEG	TEXT	GESICHT
EIMER	MUELL	MUENZE
STEIN	WINDOWS	WASSER
DOS	EDITOR	SAND

Tabelle: System-Icons

0 - leeres Rechteck
1 - STOP-Zeichen
2 - Fragezeichen
3 - Ausrufezeichen
4 - Info-Zeichen

Tabelle: Musik-Makro-Sprache

A..G	entsprechende Note:
	# - Erhöhung
	+ - Erhöhung
	- - Erniedrigung
	. - Punktiert
	n - kennzeichnet die Notenlänge. 4 steht für Viertel, 2 für Halbe, etc. Wird n weggelassen, wird die Standardlänge verwandt.
On	Oktave n (0..6) wird gewählt
>	Eine Oktave höher
<	Eine Oktave niedriger
Ln	Setzt die Standardlänge
Pn	Pause in der Länge n.
Tn	Tempo: Anzahl der Viertelnoten pro Minute. Defaultwert ist 120.
MN	normale Musik. Jede Note wird 7/8 der Zeit gespielt
ML	Legato. Die Töne gehen in einander über.
MS	Staccato. Jede Note wird nur 3/4 der Zeit gespielt.

19 - Multimediaschnittstelle

@MCIsend$, %MCIError

Eine der interessantesten Neuerungen von Windows 3.1 war die Erweiterung um Multimediafähigkeiten. Es wurde eine Multimedia-Schnittstelle geschaffen, mit der alle möglichen (und zukünftigen) Multimediageräte angesprochen werden können. Diese Schnittstelle ist das MCI, das "Media-Control-Interface. Über die Funktion @MCISEND$ kann PROFAN² mit dieser Schnittstelle kommunzieren. Alle angeschlossenen Geräte können über diese Funktion mit einfachen, fast umgangssprachlichen (englischen) Kommandos steuern. Ist das Kommando erfolgreich ausgeführt worden, hat %MCIERROR den Wert 0 und der zurückgemeldete String enthält die angeforderte Information oder einfach "OK". Ist ein Fehler aufgetreten, ist der Rückgabestring die Fehlermeldung.

Was sind Multimediageräte? Multimediageräte sind die Programme zum Abspielen von WAV-Dateien, MIDI-Dateien und neuerdings auch VIDEO-Dateien ebenso, wie z.B. der CD-Spieler (das CD-ROM-Laufwerk als Audiogerät). Um ein solches "Gerät" zu nutzen, muß es erst einmal geöffnet werden. Dazu dient der Befehl OPEN, gefolgt vom Geräte- bzw. Dateinamen. Zur Vereinfachung der weiteren Befehle kann diesem Gerät mittels ALIAS ein Name zugewiesen werden, der bis zum CLOSE-Befehl gilt.

Da es an dieser Stelle nicht möglich ist, alle MCI-Befehle aufzuführen, möchte ich wenigstens die wichtigsten Befehle in Programmzeilen bzw. Bespielprogrammen darstellen:

INHALT
REFERENZ

19.1 - CD-Player

Dieses Programmzeilen spielen die Titel 3 bis 5 der eingelegten CD.

```
@MCISend$(open cdaudio alias cd")
Print "Länge: ";@MCISend$("status cd length")
Print "Titelzahl: ";\
      @MCISend$("status cd number of tracks")
@MCISend$("set cd time format tmsf")
@MCISend$("play cd from 3 to 5")
```

Das Zeitformat "tmsf" bedeutet, daß Start und Ende im PLAY-Befehl im Format "Track:Minute:Sekunde:Frame" angegeben werden, wobei eine Sekunde 75 "Frames" (kleinste Einheit) hat. Es können also beliebige Bruchstücke eines Titels gespielt werden.
Die folgenden Zeilen halten die CD an und schließen das Gerät.

```
@MCISend$("stop cd")
@MCISend$("close cd")
```

Weitere interessante MCI-Befehle im Zusammenhang mit Audio-CDs:

`SET CD DOOR OPEN`
: öffnet das CD-Fach

`SET CD DOOR CLOSED`
: schließt CD-Fach

`STATUS CD CURRENT TRACK`
: Nummer des aktuellen Titels

`STATUS CD LENGTH`
: Länge der CD

`STATUS CD LENGTH TRACK Nr`
: Länge des Titels Nr

`STATUS CD MODE`
: augenblicklicher Modus

`STATUS CD POSITION`
: aktuelle Position im Zeitformat

Wird ein anderer Alias-Name gewählt, ist das Wort "CD" entsprechend abzuändern.

INHALT
REFERENZ

19.2 - WAV-Dateien

Ausgabe einer "WAV"-Datei: Mit folgenden Zeilen wird eine WAV-Datei gespielt:

```
@MCISend$("open gong.wav type waveaudio alias gong")
@MCISend$("play gong")  'Datei spielen
...
@MCISend$("close gong") '... schließen
```

Zwischen PLAY und CLOSE muß natürlich genug Zeit gelassen werden, die Datei zu spielen. Mit folgendem Programm können beliebige WAV-Dateien abgespielt werden. Ein Mausklick bricht die aktuelle WAV-Datei ab und zeigt erneut die Datei-Auswahlbox an. (Gewiß, es ginge auch mit dem PROFAN²-Befehl PLAYSOUND, aber der ist nicht Thema dieses Kapitels!)

```
CLS
Declare A$,B$
WindowTitle "WAV-Player"
Let A$=@LoadFile$("KLANG-DATEI","*.WAV")
While @NEQ$(A$,"")
      Let A$=@ADD$(@ADD$("OPEN ",A$),\
            " TYPE WAVEAUDIO ALIAS SOUND")
      Let B$=@MCISend$(A$)
      Case %MCIError : Print "Fehler: ";B$
      Let B$=@MCISend$("PLAY SOUND")
      WaitMouse
      Let B$=@MCISend$("CLOSE SOUND")
      Let A$=@LoadFile$("KLANG-DATEI","*.WAV")
Wend
End
```

WAV-Datei per Mikro aufnehmen: Folgende Zeilen nehmen eine WAV-Datei über das Mikrofon auf:

```
@MCISend$("open new type waveaudio alias test")
@MCISend$("set test time format milliseconds")
@MCISend$("record test from 0 to 5000 wait")
```

Nach der Aufnahme spielen wir sie ab und können sie auch abspeichern.

```
 @MCISend$("Play test from 0 wait")
 @MCISend$("save test test.wav")
```

Weitere Interessante Befehle für WAV-Dateien:

`PLAY DT FROM x TO y`
spielt von x bis y

`SEEK DT x`
positioniert bei x im Zeiformat

`DELETE DT FROM x TO y`
löscht einen Teil der Datei

`INFO DT FILE`
Dateinamen der Datei

`STATUS DT LENGTH`
Länge der Datei im Zeitformat

`STATUS DT POSITION`
Aktuelle Position in der Datei

`STATUS DT TIME FORMAT`

Aktuelles Zeitformat

```
SET DT TIME FORMAT BYTES
```

stellt Zeitformat "Bytes" ein

```
SET DT TIME FORMAT MILLISECONDS
```

Zeitformat in Millisekunden

"DT" steht hier für den Aliasnamen der entsprechenden Datei. Diese Befehle und Funktionen gelten im übrigen auch für MIDI- und VIDEO-Dateien.

INHALT
REFERENZ

19.3 - MIDI-Dateien

Da MIDI-Dateien den WAV-Dateien recht ähnlich sind, was die Ansteuerung durch MCI betrifft, hier nur ein kurzes Beispielprogramm:

```
Cls
Declare A$,B$
WindowTitle "MIDI-PLAYER"
Let A$=@LOADFILE$("MIDI-DATEI","*.MID")
While @NEQ$(A$,"")
      Let A$=@ADD$(@ADD$("OPEN ",A$),\
            " TYPE SEQUENCER ALIAS MIDI")
      Let B$=@MCISend$(A$)
      Case %MCIError: Print "Fehler: ";B$
      Let B$=@MCISend$("PLAY MIDI")?
      WaitMouse
      Let B$=@MCISend$("CLOSE MIDI")
      Let A$=@LOADFILE$("MIDI-DATEI","*.MID")
Wend
End
```

INHALT
REFERENZ

19.4 - Video für Windows

Das Neueste auf dem Multimedia-Sektor ist Microsofts "Video für Windows". Mit folgendem Programm können dieses Videodateien abgespiel werden, sofern der Video-Treiber (z.B. von der WIN-CD 1/93 aus dem Vogelverlag für 49 DM - Mit auf der CD: interessante Videos aus Werbung und Film!) installiert ist:

```
Cls
Declare A$,B$
WindowTitle "Testprogramm AVI-Video"
Let A$=@LOADFILE$("FILM-DATEI","*.AVI")
While @NEQ$(A$,"")
      Let A$=@ADD$(@ADD$("OPEN ",A$),\
            " TYPE AVIVIDEO ALIAS FILM")
      Let B$=@MCISend$(A$)
      Case %MCIError: Print "Fehler: ";B$
      Let B$=@MCISend$("PLAY FILM")
      WaitMouse
      Let B$=@MCISend$("CLOSE FILM")
      Let A$=@LOADFILE$("FILM-DATEI","*.AVI")
Wend
End
```

Weitere interessante Informationen zum Thema MCI finden sich in der einschlägigen Fachliteratur und im Heft "winTips - Sonderheft 2" aus dem VOGEL-Verlag, Abt. 731, Postfach 6740, 8700 Würzburg. Dort kann das Heft für 19.80 bestellt werden, ebenso die erwähnte CD.

INHALT
REFERENZ

20 - Dateien - Verzeichnisse - Zwischenablage

Assign, Append, Reset,
Rewrite, Print#, Input#,
@Eof, Close,
Rename, Erase, @FindFirst$, @FindNext$,
MkDir, ChDir, RmDir,
$Drive, @GetDir$, $WinPath, $SysPath
$DosVer, $WinVer, @ReadIni$, WriteIni,
ClearClip, PutClip, @GetClip$

Ein BASIC-Programmierer wird sich etwas umstellen müssen, da mir das Dateisystem von PASCAL flexibler erschien. In der aktuellen Version werden lediglich Textdateien unterstützt.

Um auf eine Datei zugreifen zu können, muß einer Dateinummer (Handle) eine physikalische Datei zugewiesen werden. Das geschieht mit dem Befehl ASSIGN:

```
Assign #1,"C:\TEST.DAT"
```

Dateinummern von 1 bis 8 sind erlaubt. Mit dem ASSIGN-Befehl ist die Datei noch nicht geöffnet. Dazu benötigen wir die Befehle RESET, REWRITE bzw APPEND. RESET öffnet die Datei zum Lesen. mit REWRITE wird sie zum Neuschreiben geöffnet und mit APPEND zum Anfügen. Wenn ich eine existierende Datei mit REWRITE öffne, wird sie gelöscht. Möchte ich an eine Datei zusätzlichen Text anhängen, ist APPEND zu verwenden:

```
Assign #1,"C:\AUTOEXEC.BAT"
Append #1
Print #1,"REM Hinzufügen von Share"
Print #1,"SHARE"
Close #1
```

Eine Datei, die ich mit RESET öffne, kann ich mit INPUT# lesen:

```
Print "AUTOEXEC.BAT"
Print
Assign #2,"C:\AUTOEXEC.BAT"
Reset #2
If %IOResult
      Print "Datei kann nicht geöffnet werden."
Else
      WhileNot @EOF(#2)
            Input #2, Zeile$
            Print Zeile$
      Wend
      Close #2
EndIf
WaitKey
```

Geschlossen wird eine Datei mit CLOSE. Mit @EOF frage ich das Ende einer Datei ab, die ich zum Lesen geöffnet habe.

Weitere Befehle sind in diesem Zusammenhang noch interessant: Mit ERASE kann ich eine Datei löschen und mit RENAME umbenennen bzw. innerhalb einer Festplatte verschieben. Mit COPY kann ich Dateien kopieren. Um herauszubekommen, welche Dateien überhaupt vorhanden sind, gibt es die Funktionen @FINDFIRST$ und @FINDNEXT$.

Für Windows spezifisch sind die INI-Dateien. Auch diese kann ich von PROFAN² aus schreiben und lesen. Daszu dienen die Funktion @READINI$ und der Befehl WRITEINI. In diesem Zusammenhang sei auch die Möglichkeit erwähnt, Daten über das Clipboard (Zwischenablage) auszutauschen. PROFAN² kann mit PUTCLIP einen String in die Zwischenblage schreiben und mit @GETCLIP$ Text aus der Zwischenablage in einen String leseb. Mit CLEARCLIP wird die Zwischenablage gelöscht. Das eröffnet interessante Möglichkeiten des Datenaustausches (nur Text) mit anderen Programmen oder der Fernsteuerung eines PROFAN²-Programmes durch ein anderes.

Das aktuelle Laufwerk ist in der Systemvaraiblen $DRIVE abgelegt und den aktuellen Pfad eines Laufwerkes ermittelt die Funktion @GETDIR$. Ebenso gibt es die Befehle MKDIR, CHDIR und RMDIR zum Anlegen, Wechseln und Löschen eines Verzeichnisses.

Für Installationsprogramme ist es wichtig, das Windowsverzeichnis und das Windows-Systemverzeichnis zu erkennen. Das ermittle ich über die Systemvariablen $WINPATH und $SYSPATH. In diesem Zusammenhang interessant sind auch die Systemvariablen $DOSVER und $WINVER.

INHALT
REFERENZ

21 - Drucken mit PROFAN²

StartPrint, EndPrint, ScreenCopy

PROFAN² unterstützt nun auch direkt den Drucker. Unter Windows ist Drucken immer seitenorientiert. Ich teile also Windows mit, wenn eine neue Seite beginnen soll, erstelle diese Seite und gebe dann die Anweisung, diese Seite zu drucken.

Mit dem Befehl STARTPRINT beginne ich eine neue Seite. Alle danach folgenden Ausgaben gehen nicht auf den Bildschirm, sondern auf den Drucker. Einige Befehle sind während des Druckens nicht erlaubt. Das betrifft insbesondere die Befehle, die sich mit Bitmap-Grafiken befassen, wie z.B. LOADBMP oder COPYBMP. Ebenso ist der FILL-Befehl nicht gestattet. Alle anderen Befehle zur Text- oder Grafikausgabe funktionieren wie gewohnt. Der einzige Unterschied: Während der Bildschirm max. 640 Pixel breit und 480 Pixel hoch ist, hat die Druckseite eine Breite von 640 Einheiten und eine Höhe von 960 Einheiten. Ebenso wie auf dem Bildschirm lassen sich die Truetype-Fonts beliebig skalieren und drehen! Ist die Seite fertig, wird sie mit ENDPRINT ausgegeben. Beispiel:

```
StartPrint
      Rectangle 100,100 - 550,850
      DrawText 150,150,"Testtext"
EndPrint
```

Eine weitere Druckmöglichkeit besteht durch den Befehl SCREENCOPY. Wird er ohne Parameter aufgerufen, wird eine Hardcopy des aktuellen Bildschirmes auf den Drucker ausgegeben. Durch Parameter kann aber auch der Bereich eingeschränkt werden, der gedruckt wird:

```
ScreenCopy
ScreenCopy 20,20 - 400,300
```

INHALT
REFERENZ

22 - Fertige Dialoge

Syntax:
@Messagebox, @LoadFile$, @SaveFile$,
@Input$, @ListBox$,
AddFiles, AddString, ClearList,
%GetCount, %GetCurSel, @ListBoxItem$

PROFAN² unterstützt zwar (noch) nicht selbstdefinierte Dialoge, bietet dafür aber eine Reihe von fertigen Standarddialogen an, die manche Programmierarbeit ersparen:

INHALT
REFERENZ

22.1 - MessageBox

Da ist zunächst einmal die Messagebox zur Ausgabe einer einfachen Meldung bzw. Frage auf den Bildschirm. Eine Messagebox hat drei Parameter: Der erste Parameter ist der Text der Meldung oder Frage, der zweite Parameter ist die Überschrift und der dritte Parameter ist eine Zahl und kennzeichnet die Fensterart, das Icon und die Anzahl der Buttons. Das Ergebnis der Messagebox-Funktion ist eine Zahl, die für den gewählten Button steht. Ein kleines Beispiel:

```
Declare Ende%          'Ende-Schalter
Declare Knopf%         'Enthält Button-Wert

Let Ende% = 0
WhileNot Ende%
      Let Knopf% = @MessageBox \
            ("Willst Du schon aufhören?",\
             "Ernstgemeinte Frage",292)
      Case @Equ(Knopf%,6): Let Ende% = 1
Wend
```

Die 292 setzen sich wie folgt zusammen: Das Icon in der Messagebox ist ein Fragezeichen (=32), es soll eine "Ja/Nein" Messagebox sein (=4) und der zweite Button (der "Nein"-Knopf) soll vorgewählt sein (=256). Diese Werte addiert ergeben 292. Die 6 steht schließlich für den "Ja"-Knopf.

Aus Gründen der Kompatibilität zu PROFAN 1.x gibt es auch noch den MessageBox-Befehl. Mit ihm sähe obiges Programm wie folgt aus:

```
Declare Ende%          'Ende-Schalter
Declare Knopf%         'Enthält Button-Wert

Let Ende% = 0
WhileNot Ende%
      MessageBox "Willst Du schon aufhören?",\
                 "Ernstgemeinte Frage",292
      Let Knopf% = %Button
      Case @Equ(Knopf%,6): Let Ende% = 1
Wend
End
```

Diese Form sollte jedoch nicht verwandt werden, da in künftigen Versionen von PROFAN dieser Befehl nicht mehr zur Verfügung stehen wird.

INHALT
REFERENZ

22.2 - InputBox

Ein weiterer Dialog ist die InputBox. Sie ermöglicht die Eingabe eines beliebigen Wertes. Die Funktion hat drei Parameter: Der erste Parameter ist der "Prompt", also die Frage bzw. Hilfe zur Eingabe. Der zweite Parameter ist die Überschrift des Dialoges und der dritte Parameter ein Vorgabewert. Dieser Vorgabewert steht schon beim Aufruf in der Inputbox und kann übernommen oder überschrieben werden. Ein Beispiel:

```
Declare Text$

Let Text$ = "Mannheim"
Let Text$ = @Input$("Wohin möchten Sie reisen?",\
                    "REISEZIEL",Text$)
Print Text$
WaitKey
End
```

Ab PROFAN² 2.5 kann sowohl die Vorgabe als auch das Ergebnis numerisch sein. Die Funktion kann so zur Eingabe von Strings und numerischen Werten verwandt werden. Beispiel:

```
Let Z&=@Input$("Neuer Wert:","Test",Z&)
```

INHALT
REFERENZ

22.3 - ListBox

Der mächtigste Dialog von PROFAN² ist aber die ListBox. Dazu gibt es ein eigenes String-Feld, daß bis zu 1000 Einträge haben kann. Diese Strings werden dann in der ListBox angezeigt und einer kann dann vom Anwender ausgewählt werden. Um diese ListBox-Liste zu leeren gibt es den Befehl CLEARLIST. Gefüllt werden kann diese Liste mit den Befehlen ADDSTRING und/oder ADDFILES. ADDSTRING fügt einen Eintrag, der als Parameter angegeben wird, zur Liste hinzu. Mit ADDFILES werden die Dateinamen eines Verzeichnisses der Liste hinzugefügt, wobei Verzeichnisnamen in eckige Klammern gestellt werden. ADDFILES verlangt als Parameter eine Dateimaske und den Pfad. Wird der Pfad weggelassen, wird das aktuelle Verzeichnis gelesen. Angezeigt wird der Dialog mit der @LISTBOX$-Funktion, die zwei Parameter hat: Der erste Parameter ist die Überschrift und der zweite Parameter gibt an, ob eine kleine (alphabetisch sortierte) oder große (unsortierte) ListBox angezeigt werden soll. Die große ListBox eignet sich insbesondere zur Darstellung von Texten. Diese werden eingelesen und dann angezeigt. Beispiele für ListBoxen:

```
Declare Text$, Wahl$

'Eine kleine ListBox mit verschiedenen Strings

ClearList
AddString "Rot"
AddString "Blau"
AddString "Grün"
AddString "Schwarz"
Wahl$ = @ListBox$("Wähle ein Farbe:",0)
Print "Du hast ";Wahl$;" gewählt."


'Eine kleine ListBox mit Dateiliste

ClearList
AddFiles "C:\WIN31\*.*"
Wahl$ = @ListBox$("Wähle eine Datei:",0)
Print "Du hast aus ";%GetCount,\
      " Einträgen ";Wahl$;" ausgewählt."

'Eine große Box zur Anzeige eines Textes

ClearList
Assign #1,"C:\WIN31\WIN.INI"
Reset #1
WhileNot @Eof(#1)
      Input #1,Text$
      AddString Text$
Wend
Close #1
@ListBox$("WIN.INI",1)

End
```

Mit %GETCOUNT wird die Anzahl der Einträge in der ListBox-Liste ermittelt. Zusätzlich besteht die Möglichkeit, mit der Funktion @LISTBOXITEM$ einen speziellen Eintrag in der ListBox zu ermitteln. Mit dem Befehl LISTBOXITEM$ kann ein spezieller Eintrag ersetzt werden. %GETCURSEL gibt die Position des ausgewählten Eintrages (immer von der unsortierten Liste ausgehend) an.

INHALT

REFERENZ

22.4 - LOAD- und SAVE-Dialoge

Am häufigsten gebraucht werden aber Dialoge zum Laden und Speichern von Dateien. Daher gibt es diese in PROFAN² schon fix und fertig. Gewiß, es wäre auch möglich, solche Dialoge vermittels der Listbox abzuwicklen, aber so ist es einfacher:

```
Declare Name$

Let Name$ = @LoadFile$("Lade ein Bild","*.BMP")
LoadSizedBmp Name$,0,0-100,100

Let Name$ = @SaveFile$("Speichere das Bild","NEU.BMP")
If NEqu$(Name$,"")
      SaveBmp Name$,0,0-100,100
EndIf
WaitInput
End
```

Der Name "NEU.BMP" wird für das Speichern vorgeschlagen, kann aber jederzeit im Dialog geändert werden. Wenn das Ergebnis eines SaveFile-Dialoges der Leerstring ist wurde der Dialog mit "Abbruch" beendet. Probieren Sie bitte - wie immer - alle Beispiele aus.

INHALT
REFERENZ

Tabelle: Ergebnis von @MessageBox

1	- OK
2	- Abbrechen (Cancel)
3	- Abbrechen (Abort)
4	- Wiederholen
5	- Ignorieren
6	- Ja
7	- Nein

Tabelle: MessageBox-Stile

Der Wert setzt sich zusammen aus BUTTONS + ICON + DEFAULT + FENSTERART.

Werte für BUTTON:

0 - OK

1 - OK Abbrechen

2 - Abbrechen Wiederholen Ignorieren

3 - Ja Nein Abbrechen

4 - Ja Nein

5 - Wiederholen Abbrechen

Werte für ICON:

0 - Kein Icon

16 - STOP

32 - "?"

49 - "!"

64 - "i"

Werte für DEFAULT-Knopf:

0 - 1. Knopf

256 - 2. Knopf

512 - 3. Knopf

Werte für FENSTERART:

0 - "normales" Fenster.

4096 - nicht verschiebbares "Fehler"-Fenster.

23 - Menüs und Mäuse

Syntax:
AppendMenuBar, AppendMenu, PopUp,
SubPopUp, EndSub, CreateMenu, TrackMenu,
WaitMouse, WaitInput, WaitKey, WaitScan, %Key, %MouseKey,
%ScanKey, %MouseX, %MouseY,
@MenuItem, @Inkey$, @Getkey$

Windows ist ein ereignisgesteuertes System. Bedeutende Ereignise werden durch Menüs und PopUp-Menüs ausgelöst.

INHALT
REFERENZ

23.1 - Fenster-Menüs

PROFAN² kennt zwei Sorten Menüs. Da ist zunächst das ganz normale Anwendungsmenü, das Fenster-Menü. Jedes PROFAN²-Programm läßt sich mit einem eigenen Menü ausstatten. Ein Fenster-Menü besteht aus mehreren Menüpunkten in der Menüzeile. In der Regel öffnen sich beim Anwählen dieser Menüpunkte PopUp-Menüs mit weiteren Punkten. Mit dem POPUP-Befehl wird der Menüzeile ein weiterer Menüpunkt hinzugefügt und mit APPENDMENU werden die einzelnen Punkte dieses PopUp-Menüs angehängt. Die Zahl bei AppendMenü ist die Nummer dieses Menüpunktes. Über diese Nummer kann festgestellt werden, welcher Menüpunkt gewählt wurde. Werte zwischen 1 und 253 sind erlaubt. Mit dem Befehl SEPARATOR kann auch eine Trennlinie zwischen Menüpunkten eingeführt werden.
Mit SUBPOPUP wird anstelle eines Menüpunktes ein weiteres Untermenü angehängt, das mit ENDSUB endet. Um einen Buchstaben unterstrichen darzustellen, damit man auch die Tastatur benutzen kann, setzt man das "&" davor.
Mit @MENUITEM wird abgefragt, ob der entsprechende Menüpunkt angewählt wurde. Der gewählte Menüpunkt ist auch in der Systemvariablen %MENUITEM gespeichert.
Probieren Sie folgendes Beispiel aus, und Ihnen wird klar, was ich so umständlich zu erklären suche:

```
Declare Ende%  PopUp "&Datei"
      AppendMenu 101,"&Laden"
      AppendMenu 102,"&Speichern"
      AppendMenu 103,"Speichern &als"
      Separator
      AppendMenu 109,"&Ende"
PopUp "&Bearbeiten"
      AppendMenu 111,"&Ausschneiden"
      AppendMenu 112,"&Kopieren"
      SubPopUp "&Schrift"
            AppendMenu 113,"&Arial"
            AppendMenu 114,"&Courier"
      EndSub
PopUp "&Suchen"
      AppendMenu 121,"&Ersetzen"
      AppendMenu 122,"&Suchen"
AppendMenuBar 130,"&Info"

PROC Info
      @MessageBox("Testprogramm","Info",64)
ENDPROC

PROC Ueber
      @MessageBox("Ihr Copyright","Über ...",48)
ENDPROC

PROC WasAnderes
      Print "Gewählter Menüpunkt ";%MenuItem
ENDPROC

Let Ende% = 0
WhileNot Ende%
      WaitInput
      If @MenuItem(109)
            Let Ende% = 1
      ElseIf @MenuItem(130)
            Info
      ElseIf @MenuItem(254)
```

```
            Ueber
      Else
            WasAnderes
      EndIf
Wend
End
```

Dieses Beispiel zeigt, wie die Grundstruktur eines PROFAN²-Programmes mit Menü sein kann. Zwei Besonderheiten: Mit APPENDMENUBAR wird der Menüzeile ein Menüpunkt hinzugefügt, der keine weiteren Menüpunkte hat. Beim Anwählen dieses Punktes wird kein PopUp-Menü geöffnet, sondern sofort eine Aktion ausgeführt. Die zweite Besonderheit ist der Wert 254. Hier handelt es sich um die Anwahl des Copyright-Zeichens, dem vordersten Menüpunkt. Dieser Menüpunkt existiert immer! (In der Shareware-Version wird zusätzlich noch ein Sharewarehinweis eingeblendet, wenn dieser Menüpunkt angewählt wird.)

INHALT
REFERENZ

23.2 - Unabhängige PopUp-Menüs

Die zweite Sorte Menüs sind die unabhängigen PopUp-Menüs. Diese können an jedem Ort erzeugt werden und verschwinden nach Auswahl eines Menüpunktes wieder. Mit CREATEMENU beginnt ein solches Menü. Die Definition des Menüs erfolgt ebenso wie bei den PopUp-Menüs des Fenster-Menüs. Angezeigt wird das Menü mit dem Befehl TRACKMENU, dem als Parameter Koordinaten der oberen linken Ecke des Menüs folgen. Der Befehl TRACKMENU wartet, bis ein Menupunkt ausgewählt wurde. Häufig werden bei kommerziellen Programmen solche Menüs dort angezeigt, wo die Maus gerade steht. Diese Menüs werden meist mit der rechten Maustaste geöffnet. Hierzu ein kleines Beispiel:

```
Declare Ende%

Let Ende% = 0
WhileNot Ende%
      If @Equ(%MouseKey,2)
            CreateMenu
                  AppendMenu 101,"&blau"
                  AppendMenu 102,"&rot"
                  AppendMenu 103,"&grün"
                  Separator
                  AppendMenu 109,"&Ende"
            TrackMenu %MouseX,%MouseY
            If @MenuItem(109)
                  Let Ende% = 1
            Else
                  Print "Ausgewählt:",%MenuItem
            EndIf
      EndIf
Wend
End
```

INHALT
REFERENZ

23.3 - Warten auf Ereignisse

Im Zusammenhang mit Windows spricht man häufig von "ereignisorientierter Programmierung". Was ist darunter zu verstehen? Bei den meisten Programmen ist es doch so, daß immer nur dann etwas geschieht, wenn ich mit der Maus etwas anklicke oder irgendeine Taste drücke. Ansonsten wartet Windows darauf, daß etwas passiert.Windows wartet, bis ein Ereignis eintritt, um dann auf das Ereignis zu reagieren. Nach der Reaktion auf das Ereignis oder schon während der Reaktion wartet Windows auf das nächste Ereignis. In einem Windowsprogramm bedeutet dies, daß die meisten Routinen aufgrund eines Ereignisses aufgerufen werden. Unser obiges Menüprogramm ist dafür ein Beispiel: Windows wartet (bei WAITINPUT) auf ein Ereignis. Wenn dieses eintritt, wird überprüft, was für ein Ereignis es war und je nach Ereignis wird die entsprechende Routine aufgerufen.

In PROFAN² kennen wir mehrere Befehle, um auf Ereignisse zu warten:

WAITKEY wartet auf einen Tastendruck. Dabei sind nur die 'normalen' alphanumerischen Tasten erlaubt. Auf Sondertasten und Mausklick wird nicht reagiert. Welches Ereignis (= welche Taste) es war, steht in der Systemvariablen %KEY. Wollen wir auch die Funktionstasten abfragen, müssen wir WAITSCAN einsetzen. Das Ergebnis finden wir in %SCANKEY.

Sobald wir aber ein Menü mit im Programm haben, wird man immer auch die Maus mit abfragen wollen. WAITINPUT wartet, bis entweder eine Taste gedrückt wurde oder ein Mausklick oder eine Menüauswahl erfolgte. (Wollen wir auf die Abfrage der Tastatur verzichten, reicht auch WAITMOUSE.) Das Ergebnis von WAITINPUT finden wir in &MENUITEM, &KEY, &SCANKEY, &MOUSEKEY, &MOUSEX und &MOUSEY.

Hinweis: Auch ohne WAITINPUT oder WAITMOUSE ist die aktuelle Mausposition in %MOUSEX und %MOUSEY vorhanden und die aktuell gedrückte Maustaste in %MOUSEKEY. Das Beispielprogramm für das unabhängige PopUp-Menü macht davon Gebrauch. Daher kommt es ohne einen WAIT-Befehl aus. Das Hauptmodul selbst (zwischen WHILE und WEND) ist die Warteschleife, die auf ein Ereignis wartet.

Wenn auch die Abfrage der Systemvariablen ausreichen würde, um das Ereignis näher zu beschreiben, so gibt es in PROFAN² einige Funktionen, die dies erleichtern. Wollten Sie mit den Systemvaraiblen herausbekommen, ob sich die Maus in einem bestimmten Bereich befindet, ergäbe das eine recht lange verschachtelte Funktion mit mehreren @AND, @LT unf @GT. Das erspart uns die Funktion @MOUSE, die zwei Koordinatenpaare als Parameter erwartet und den Wert 0 hat, wenn die Maus außerhalb dieses Rechteches ist:

```
If @Mouse(10,10-60,60)
      Print "Die Maus ist im Rechteck 10,10 - 60,60 !"
Else
      Print "Daneben!"
EndIf
```

Die Abfrage der Systemvariablen %MENUITEM wird durch die Funktion @MENUITEM vereinfacht, die Abfrage von %SCANKEY durch die Funktion @SKANKEY und die Abfrage von %KEY durch die Funktion @KEYIN.

In Anlehnung an BASIC gibt es noch zwei weitere Funktionen, um die Tastatur abzufragen: Das ist einmal die Funktion @INKEY$(), die die augenblicklich gedrückte Taste zurückgibt und die Funktion @GETKEY$(), die auf einen Tastedruck wartet und die Taste als Zeichen zurückgibt.

INHALT
REFERENZ

24 - Kompatibilität zu PROFAN 1.x

PROFAN² ist weitestgehend Quelltext-kompatibel zu PROFAN 1.x. Das Format des Zwischencodes mußte jedoch (zur Temposteigerung etc.) verändert werden, sodaß es NICHT möglich ist, eine alte compilierte PRC-Datei mit der neuen RUNTIME zu starten. In den meisten Fällen reicht es aus, den bisherigen Quelltext neu zu compilieren. Um sicherzugehen, das alle Zahlen als Integer ausgegeben werden und auch Divisionen gleich behandelt werden, sollten Sie die Zeile "DECIMALS 0" an den Anfang setzen und jedes "@DIV" durch "@DIV&" ersetzen.

Die Befehle LISTBOX und INPUTBOX und die damit verbundenen Befehle wurden durch die leistungsfähigeren Funktionen @LISTBOX$ und @INPUT$ ersetzt. Wenn Sie diese Befehle verwandten, müssen Sie die entsprechenden Programmteile umschreiben, wobei diese einfacher werden, da die Funktionen die komplette Dialogsteuerung beinhalten.

Ein Fehler bei TRACKMENU wurde behoben: Auch TRACKMENU benutzt jetzt zum Fenster relative Koordinaten und nicht die absoluten Koordinaten, wie in PROFAN 1.x! Im Übrigen müssen in PROFAN² die PopUp-Menüs des Fenstermenüs nicht mehr mit TRACKMENU simuliert werden. Es gibt jetzt echte PopUp-Menüs und sogar Untermenüs.

Künftige PROFAN-Versionen werden 100%ig Quelltext-kompatibel zu PROFAN² 2.x sein. Lediglich die Befehle und Funktionen, die nur aus Kompatibilitätsgründen zu 1.x enthalten sind, können für die Zukunft nicht garantiert werden. Dazu zählen insbesondere die DIM- und LIST-Anweisung ohne Datentypkennung, ebenso die dazugehörige Version @LIST. Außerdem ist die Funktion @MESSAGEBOX dem gleichnamigen Befehl vorzuziehen.

INHALT
REFERENZ

Anhang: Icon-Manager "Phish"

Dieser Icon-Manager ist Shareware und muß bei regelmäßigem Gebrauch beim Autoren registriert werden! Ich habe ihn dem PROFAN²-Paket beigepackt, weil er sehr gut geeignet ist, die Icons in PROFAN.EXE bzw. PROFRUN.EXE zu verändern. Beachten Sie bitte:

- Sie können auch Icons in der fertigen EXE-Datei verändern, dürfen diese aber nicht um ein Byte vergrößern oder verkleinern!

- Es ist sinnvoll, die Icons im Runtime-Modul PROFRUN.EXE vor dem Linken zu ändern. Vor dem Linken dürfen dieser Datei mittels entsprechender Werkzeuge (z.B. "Resource-Workshop" von Borland) auch weitere Icons zugefügt werden. Diese können dann auch im Programm verwandt werden.

Hier die Originaldokumentation (gekürzt):

PhishEdit version 1.3

(C) Copyright 1992 Philip B. Eskelin, Jr. All rights reserved. Windows is a trademark of Microsoft Corp. (blah blah blah...<g>)

...
I tested PhishEdit to the best of my ability, in all resolutions. I fully developed it under debug Windows. It should be very solid, robust, and machine independent. Notify me of any bugs over compuserve mail [76216,1410]. Please beware when changing icons in any Windows executable or library. I suggest you back up the executable you change until you get use to using the feature of saving to executables/libraries. Have Fun!!

Usage:

File Menu

New - Opens a new untitled icon. It is not possible to create a new icon library or Windows executable with this option. A maximum of eight icons can be opened at one time. The icon opended by this command must be saved to a file name by way of the "Save As..." command from the File Menu.

Open - A dialog box is displayed which allows you to find an icon, library, or Windows executable. If it is just an icon, then the image is displayed in a new window for editing. If the file is a Windows executable or library, another dialog box is shown, so that you can pick which icon inside the executable or library to edit. From there, the icon image in the file is displayed, and can be edited and saved back into the same executable. I recommend that only advanced users who are accustomed to using Windows open/manipulate executables and libaries, as it can mean disaster if not used correctly.

Close - closes the currently active icon, prompting to save changes if any changes have been made. The currently active icon is the icon window that is up front, or has the focus.

Save - saves an icon that has been opened under the same file name. If the icon image was opened from a Windows executable or library, then the "Save" command writes over the existing image. ...

Save As - Displays a dialog box allowing you to specify which file name to save the image into. BEWARE: if you create a new icon image, then save to a Windows executable or library file, the icon image will be written over the first icon in the executable or library.

Exit - Exits PhishEdit, and asks you whether or not to save changes over unsaved icon images.

Edit Menu

Undo - Cancels the previous drawing operation.

Clear - Clears the drawing area of the currently active icon image.

Copy - Copies the currently active icon image to the clipboard in bitmap format. The image can then be pasted into other icon drawing programs with clipboard support, or pasted into PaintBrush.

Cut - Equivalent to choosing "Copy" then "Clear" from the edit menu.

Paste - Takes a bitmap image from the clipboard, and pastes it into the drawing area of the currently active icon in PhishEdit.

Tools Menu

Pixel - draws one pixel at a time in the currently active icon image with the current color.

Line - draws a straight line (click and drag with the mouse to draw a line) with the current color.

Hollow Rectangle - draws a hollow rectangle with the border being of the current drawing color.

Solid Rectangle - draws a solid rectangle with the current drawing color.

Hollow Ellipse - draws a hollow ellipse or circle with the border being of the current drawing color.

Solid Ellipse - draws a solid ellipse or circle with the current drawing color.

Rotate 90 Degrees - Rotates the currently active icon image 90 degrees in the clockwise direction. Undo can be used to cancel this.

Shift Right - Shifts the icon image to the right by one pixel column.

Shift Left - Shifts the icon image to the left by one pixel column.

Shift Up - Shifts the icon image up by one pixel row.

Shift Down - Shifts the icon image down by one pixel row.

32x32 Capture - PhishEdit minimizes, and the cursor changes to a hollow box. When you click down with the left mouse button, everything inside the box cursor is copied into the drawing area of the current icon. If you make a mistake, Undo can be used. This is especially useful if you want to capture text.

Variable Capture - PhishEdit minimizes. You then click and drag with the left mouse button. When you up-click with the left mouse button, the image inside of the rectangle is copied into the currently active icon window inside PhishEdit.

Window Menu

Cascade - Cascades all opened drawing windows so that each one is offset down and to the right of the previous window.

Tile - tiles all opened drawing windows so that all of them can be seen at once.

Arrange Icons - Arranges all of the minimized drawing windows within PhishEdit at the lower-left area of

the PhishEdit window.

File Listings - Below the "Cascade", "Tile", and "Arrange Icons" menu options, you will see a list of the drawing windows. Picking one of them brings the drawing window up to the front of the other windows and makes it the active drawing window.

Color Menu

Colors - You can change the current color in any of the three drawing modes. Choosing a color in here is equivalent to choosing a color in the drop-down box in the tool bar.

Color Mode - The drawing mode where drawing in a color will result in the color being displayed normally when the icon is displayed.

Screen Mode - The same thing as "Transparent" mode. Whatever color you choose for this, will be the "invisible" part of the icon.

Inverse Mode - Whatever color you choose for this, will be the inverse part of the icon, so that when the icon is displayed, the inverse part is just the colors behind the icon inverted. The Inverse color will always be inverse of the "Transparent" or "Screen Mode" color.

Window Grid - Displays a grid in the currently active drawing area. Especially useful when you are trying to be accurate.

Window Coordinates - displays the logical coordinates of the mouse cursor when the mouse cursor is inside of the drawing window. This also aids in accuracy. The Window coordinates are displayed in the status bar at the bottom of PhishEdit's window.

Help Menu

Help on PhishEdit - Runs notepad and displays the README.TXT file you are reading right now.

About PhishEdit - displays about information and shows where to send the shareware registration (...).

...

Registration:

If you find this useful, send 25 dollars (suggested) as a donation to my work, in either a check or money form. If you are outside of the U.S., send either cash or a money order in US dollars.

Please mail along with your checks any suggestions or comments about its interface and its abilities.

Please send to:

Philip B. Eskelin, Jr.
P.O. Box 4010
Silverdale, WA 98383-4010

Kommentare

' ... (Kommentarzeichen)

Kommentarzeichen. Im Gegensatz zu REM kann (sollte) es auch hinter einem Befehl in einer Zeile stehen:

```
' Unterprogramm

Let X%=125  'X-Position zuweisen
```

Rem ...

Alles was dahinter steht, ist Kommentar. REM steht immer am Anfang der Kommentarzeile. Für Kommentare hinter anderen Befehlen ist das ' zu benutzen.

Befehle
Übersicht

Add VAR,N

```
VAR - Variablenname (Integer)
N   - Integer
```

N wird zur Variablen VAR addiert. (Ist schneller und einfacher als die @ADD-Funktion.) Beispiel:

```
ADD Punkte%,10
```

Befehle
Übersicht

AddFiles S

```
S : String - Dateimaske
```

Mit diesem Befehl wird eine Dateiliste der Listbox-Liste hinzugefügt. Beispiel:

```
ClearList
AddFiles "*.RGH"
```

Befehle
Übersicht

AddString S

```
S : String
```

Mit diesem Befehl wird der String der Listbox-Liste hinzugefügt. Beispiel:

```
ClearList
AddString "Item 1"
AddString "Item 2"
```

Befehle
Übersicht

Append #N

N: Wert - Dateinummer (1 .. 8)

Die Datei #N wird geöffnet, um Daten anzufügen. Tritt ein Fehler auf, ist %IoResult größer als 0. Beispiel:

```
Assign #1,"TEST.DAT"
Append #1
Case %IoResult:MessageBox "Fehler:",\
          "Kann Datei nicht öffnen!",16
Print #1,"Testdaten",56,Hugo$
Close #1
```

Befehle
Übersicht

AppendMenu N,S

```
N : Wert - Menünummer (1 .. 253)
S : String - Eintrag
```

Dem PopUp-Menü wird der Eintrag S unter der Nummer N hinzugefügt. Ein vorangestelltes & sorgt für eine Unterstreichung. Ein Menüeintrag mit der Nummer 254 bleibt wirkungslos, 255 erzeugt immer die PROFAN-Copyright- Meldung. Beispiel:

```
CreateMenu
AppendMenu 100,"&Laden"
AppendMenu 101,"&Speichern"
AppendMenu 102,"Speichern &als"
TrackMenu 20,130
Case @MenuItem(100):Goto "Laden"
Case @MenuItem(101):Goto "Speichern"
...
```

Weiteres Beispiel:

```
PopUp "&Datei"
   AppendMenu 100,"&Laden"
   AppendMenu 101,"&Speichern"
   AppendMenu 102,"Speichern &als"
   Separator
   SubPopUp "Datei&typ"
      AppendMenu 110,"Exe"
      AppendMenu 120,"Com"
   EndSub
   AppendMenu 103,"&Ende"
PopUp .....
```

Befehle
Übersicht

AppendMenuBar N,S

```
N : Wert - Menünummer (1 .. 253)
S : String - Eintrag
```

Dem Haupt-Menü wird der Eintrag S unter der Nummer N hinzugefügt. Ein vorangestelltes & sorgt für eine Unterstreichung. Beispiel:

```
AppendMenuBar 10,"&Laden"
AppendMenuBar 11,"S&peichern"
WaitMouse
Case %MenuItem(10):Goto "Laden"
```

Befehle
Übersicht

Arc X1,Y1-X2,Y2; X3,Y3; X4,Y4

```
X1,Y1 : Wert - Links oben
X2,Y2 : Wert - Rechts unten
X3,Y3 : Start
X4,X5 : Ende
```

Es wird ein Kreisbogen gezeichnet. X1,Y1 und X2,Y2 bezeichnen das Rechteck, indem der Kreis sich befindet, die anderen Koordinaten Start und Ende des Kreisbogens.

Befehle
Übersicht

Assign #N,S

```
N : Wert - Dateinummer (1 .. 8)
S : String - Dateiname (+ Pfad)
```

Der Datei Nummer N wird die Datei S zugewiesen. Beispiel:

```
Assign #1,"DEMO.PRF"
Reset #1
WhileNot @Eof(#1)
   Input #1,Zeile$
   Print Zeile$
Wend
Close #1
```

Befehle
Übersicht

Beep

Gibt einen Signalton aus. Zur Tonausgabe siehe auch unter SOUND und PLAY!

Befehle
Übersicht

Case - CaseNot

Case N:BEF

```
N   : Wert - Bedingungsausdruck
BEF : Profan Befehlszeile
```

Hat der Ausdruck N einen Wert <> 0, wird BEF ausgeführt, ansonsten geht die Programmausführung in die nächste Zeile. CASE entspricht einem einzeiligem IF-Befehl.

CaseNot N:BEF

```
N   : Wert - Bedingungsausdruck
BEF : Profan Befehlszeile
```

Hat der Ausdruck N den Wert 0, wird BEF ausgeführt, ansonsten geht die Programmausführung in die nächste Zeile. CASENOT entspricht einem einzeiligem IFNOT-Befehl.

Befehle
Übersicht

CharSet N

`N: Integer - Zeichenatz`

Der für den nächsten USEFONT-Befehl benutzte Zeichensatz wird festgelegt.

0 - ANSI
1 - ASCII
2 - Symbol

Voreinstellung ist ANSI.

Befehle
Übersicht

ChDir S

`S : String - Pfadangabe`

Es wird zum Verzeichnis S gewechselt. Enthält S eine Laufwerksangabe, so wird auch das aktuelle Laufwerk gewechselt!
Beispiel:

```
ChDir "D:\TABELLEN\GRAFIK"
LoadSizedBmp "Umsatz",10,30-400,600;1
```

Befehle
Übersicht

Chord X1,Y1-X2,Y2; X3,Y3; X4,Y4

```
X1,Y1 : Wert - Links oben
X2,Y2 : Wert - Rechts unten
X3,Y3 : Start
X4,X5 : Ende
```

Es wird ein Kreisabschnitt gezeichnet. X1,Y1 und X2,Y2 bezeichnen das Rechteck, indem der Kreis sich befindet, die anderen Koordinaten Start und Ende des Kreisabschnittes.

Befehle
Übersicht

ClearList

Die ListBox-Liste wird gelöscht, d.h. der Zeiger wird auf den Anfang gesetzt. Mit AddFiles und AddString wird die Liste gefüllt.

Befehle
Übersicht

Close #N

```
N : Wert - Dateinummer (1 .. 8)
```

Die Datei mit der Dateinummer N wird geschlossen. Beispiel:

```
Close #Nr%
```

Befehle
Übersicht

Cls

Löscht den Bildschirm. Ist noch kein Fenster offen, so wird eins geöffnet.

Befehle
Übersicht

Color C1,C2

```
C1 : Wert - Textfarbe (0 .. 15)
C2 : Wert - Hintegrund (0 .. 15)
```

Die Farben des Textmodus (PRINT, TBOX, INPUT) werden festgelegt. Die Farben entsprechen den 16 Farben von MS-DOS im Textmodus. Beispiel:

```
Color 15,0
Print "Hugo was here!"
```

Befehle
Übersicht

Copy S1 > S2

```
S1 : String - Quelldatei
S2 : String - Zieldatei
```

Die Quelldatei wird kopiert und erhält Zieldatei als Namen. Beide Strings können auch Pfadangaben enthalten. Wildcards sind jedoch nicht gestattet. Beispiel:

```
COPY "C:\START.BAT" > "D:\BAK\START.BAK"
```

Befehle
Übersicht

CopyBmp - CopySizedBmp

CopyBmp X1,Y1-X2,Y2 > X3,Y3;N

```
X1,Y1 : Wert : Quelle rechts oben
X2,Y2 : Wert : Breite, Höhe
X3,Y3 : Wert : Ziel rechts oben
N     : Wert : Kopiermodus
```

Der Bildschirmausschnitt der beginnend bei X1,Y1 eine Breite von X2 Pixel und eine Höhe von Y2 Pixel hat, wird an die Position X3,Y3 kopiert. ACHTUNG: X2,Y2 sind keine absoluten Koordinaten sondern Breite und Höhe!
Der letzte Parameter bestimmt den Kopiermodus.

CopySizedBmp X1,Y1-X2,Y2>X3,Y3-X4,Y4;N

```
X1,Y1 : Wert : Quelle rechts oben
X2,Y2 : Wert : Breite, Höhe
X3,Y3 : Wert : Ziel rechts oben
X4,Y4 : Wert : Breite, Höhe
N     : Wert : Kopiermodus
```

Der Bildschirmausschnitt an X1,Y1 mit der Größe X2,Y2 wird an die Position X3,Y3 in der Größe X4,Y4 kopiert. ACHTUNG: X2,Y2,X4,Y4 sind keine absoluten Koordinaten sondern die Größe!

Befehle
Übersicht

CreateMenu

Erzeugt ein PopUp-Menü. Das Menü wird mit APPENDMENU gefüllt und mit TRACKMENU abgefragt. Das Ergebnis steht in der Systemvariablen %MENUITEM.

Befehle
Übersicht

Dec VAR

```
VAR : Variablenname (Integer)
```

VAR wird um 1 erniedrigt. (Ist identisch mit SUB VAR,1)

Befehle
Übersicht

Decimals N

`N: Integer - Stellenzahl`

Die Anzahl der auszugebenden Dezimalstellen wird bestimmt. Vorgabewert ist 6. Dieser Wert wird auch bei der @Str$-Funktion berücksichtigt. (Bei in PROFAN 1.x geschriebenen Programmen sollte grundsätzlich die Zeile "DECIMALS 0" eingefügt werden, bevor sie unter PROFAN² 2.x neu compiliert werden.)

Befehle
Übersicht

Declare VAR [,VAR[,VAR]...]

```
VAR : Variablen-Name
```

In PROFAN müssen alle Variablen (zeitlich) vor der ersten Benutzung declariert werden.
String-Variablen werden durch ein $, LongInt-Variablen durch ein &, Float-Variablen durch ein ! und Integer-Variablen durch ein % gekennzeichnet.
Beispiel:

```
Decalare XPos%,YPos%,Text$
```

Befehle
Übersicht

Def @<Name>(N) <Ausdruck>

```
<Name>     - Funktionsname
N          - Anzahl der Parameter
<Ausdruck> - Funktion
```

Eine neue Funktion wird definiert. <Name> ist der Name der neuen Funktion, N kennzeichnet die Anzahl der übergebenen Parameter und <Ausdruck> beschreibt die neue Funktion. Die Parameter sind in @$(n), @%(n), @!(n) und @&(n) enthalten.

```
Declare A%,B%
Def @Max(2) @If(@Gt(@!(1),@!(2)),\
                    @!(1),@!(2))
Let A%=34
Let B%=12
Print @Max(A%,B%)
```

Diese Funktion ermittelt den größeren der beiden Werte.

Befehle
Übersicht

Dim! - Dim% - Dim& - Dim$

Dim! N

```
N : Integer - Anzahl 0 .. 999
```

Festlegung der Größe des Float-Arrays. Der Befehl darf nur einmal im Programm vorkommen und muß vor dem ersten Gebrauch einer Float-Array- Variablen erfolgen.
Beispiel:

```
DIM! 56
LIST! 56 = 2345.67
```

Dim$ N

```
N : Integer - Anzahl 0 .. 999
```

Festlegung der Größe des Stringarrays. Der Befehl darf nur einmal im Programm vorkommen und muß vor dem ersten Gebrauch einer String- Array-Variablen erfolgen.

Dim% N

```
N : Integer - Anzahl 0 .. 999
```

Festlegung der Größe des Integerarrays. Der Befehl darf nur einmal im Programm vorkommen und muß vor dem ersten Gebrauch einer Integer- Array-Variablen erfolgen.

Dim& N

```
N : Integer - Anzahl 0 .. 999
```

Festlegung der Größe des LongInt-Arrays. Der Befehl darf nur einmal im Programm vorkommen und muß vor dem ersten Gebrauch einer LongInt- Array-Variablen erfolgen.

Befehle
Übersicht

DrawIcon - DrawSysIcon

DrawIcon S,X,Y

```
S   : String : Icon-Name
X,Y : Wert : Koordinaten
```

Das Icon S wird an Position X,Y gezeichnet. Beispiel:

```
DrawIcon "Knopf1",12,12
```

18 Icons sind in PROFAN.EXE enthalten, können aber z.B. mit einem Resourcen-Toolkit verändert werden.

Die Icons, die in Profan vorhanden sind: Liste

Es gibt auch Icon-Editoren, mit denen sie verändert werden können.

DrawSysIcon N,X,Y

```
N   : Wert - Icon-Nummer (0 .. 4)
X,Y : Wert - Koordionaten
```

Das Windowseigene Icon mit der Nummer N wird an Position X,Y gezeichnet.

0 : Windows-Symbol
1 : STOP-Zeichen
2 : Fragezeichen
3 : Ausrufezeichen
4 : Info-Zeichen

Befehle
Übersicht

DrawText X,Y,S

```
X,Y : Wert - Koordinaten
S   : String oder numerischer Wert
```

Der Text S wird an der Bildschirmposition X,Y mit dem durch USEFONT eingestellten Font in der durch TEXTCOLOR eingestellten Farbe dargestellt.

```
DrawText 10,50,"Highscore:"
```

Ab Profan² 2.5 darf S auch ein numerischer Literal oder eine numerische Variable sein. Vor der Ausgabe wird sie unter Berücksichtigung von Numwidth und Decimals in einen String umgewandelt:

```
DrawText 10,50,HighScore&
```

Befehle
Übersicht

Ellipse X1,Y1-X2,Y2

```
X1,Y1 : Wert - Links oben
X2,Y2 : Wert - Rechts unten
```

Es wird ein Kreis, bzw. eine Ellipse gezeichnet. X1,Y1 und X2,Y2 bezeichnen das Rechteck, indem die Ellipse sich befindet.

Befehle
Übersicht

End

Programmende. Ohne weitere Abfrage wird das Profan-Programm und der Profan-Interpreter verlassen.

Befehle
Übersicht

Erase #N S

N : Wert - Dateinummer (1 .. 8)

Die Datei mit dem Dateikennzeichen N wird gelöscht. Beispiel:

```
ASSIGN #1,"TEST.BAK"
ERASE #1
```

Befehle
Übersicht

Fill X,Y,C

```
X,Y : Wert - Koordinaten
C   : Wert - RGB-Farbwert der Grezfarbe
```

Ausgehend von Punkt X,Y wird alles bis zu einem Rahmen, der die Farbe C hat, mit dem aktuellen Pinsel (USEBRUSH) gefüllt. Beispiel:

```
Fill 20,50,@RGB(0,0,31)
```

Befehle
Übersicht

Font N

`N : Wert - Font - Nummer (0 .. 2)`

Der Font N für den Textmodus (Textausgabe mit PRINT) wird gewählt:

0 : System-Font (ANSI-Zeichensatz)
1 : OEM-Font (ASCII-Zeichensatz)
2 : ANSI-Font

Befehle
Übersicht

Gosub S ... Return [<Wert>]

```
S : String - Unterprogramm-Label
Wert - Rückgabewert
```

Mit GOSUB wird zum Label S verzweigt. Bei RETURN gehts in der Zeile nach dem GOSUB weiter. Beispiel:

```
Gosub "Unterprogramm"
End

Unterprogramm:
      Print "Unterprogramm" Return
```

Es ist auch möglich einen Wert zurückzugeben. Dieser steht dann in @$(0), @%(0), @&(0) und/oder @!(0):

```
Gosub "Unterprogramm"
Print @$(0)
End

Unterprogramm:
Print "Unterprogramm"
Return "OK"
```

Mit RETURN kann ebenso eine Prozedur (siehe unter PROC) mit einem Rückgabewert verlassen werden.

Alle innerhalb des Unterprogrammes deklarierten Variablen sind nur in diesem bekannt. Hat eine solche Variable den gleichen Namen wie eine des aufrufenden Programmes, so ist im Unterprogramm nur die im Unterprogramm definierte bekannt.

Befehle
Übersicht

Goto S

```
S : String - Label
```

Das Programm verzweigt zum Label S. Ein Label wird durch einen Doppelpunkt gekennzeichnet (der in S nicht enthalten ist). Beispiel:

```
Nochmal:
Input S%
IfNot S%
      Goto "Nochmal"
EndIf
```

Befehle
Übersicht

If | IfNot ... ElseIf ... Else ... EndIf

If N ... [ElseIf N ...] [Else ...] EndIf

```
N : Wert . Bedingungsausdruck
```

Die zwischen IF und ENDIF stehenden Zeilen werden nur dann ausgeführt, wenn N ungleich 0 ist.
Beispiel:

```
If @Equ(Test%,78)
      Print "Test% ist 78!"
      Print "-------------"
EndIf
```

Auch eine Abrage auf mehrere Bedingungen (z.B. bei der Auswertung eines Menüs) ist mit ELSEIF möglich. Sobald eine Bedingung erfüllt ist, wird der zugehörige Programmteil abgearbeitet und anschließend mit der Zeile nach dem ENDIF fortgefahren:

```
IF @Lt(A%,1)
      COLOR 1,15
      PRINT "A ist kleiner als 1"
ELSEIF @Equ(A%,1)
      COLOR 2,15
      PRINT "A ist gleich 1"
ELSEIF @Equ(A%,2)
      COLOR 3,15
      PRINT "A ist gleich 2"
ELSE
      PRINT "A ist größer als 2"
ENDIF
```

IfNot N ... [ElseIf N ...] [Else ...] EndIf

Als Kuruform für If @Not(N) kann auch das kürzere (und schnellere) IfNot verwandt werden:

```
IfNot @Equ(Test%,78)
      Print "Test% ist nicht 78!"
      Print "-------------------"
EndIf
```

Befehle
Übersicht

Inc VAR

```
VAR - Variablenname (Integer)
```

VAR wird um 1 erhöht. (Ist identisch mit ADD VAR%,1)

Befehle
Übersicht

Input #N,VAR

```
N : Wert - Dateikennzeichen (1..8)
VAR : Variable
```

Aus der Datei #N wird ein Wert in die Variable gelesen. Beispiel:

```
Assign #1,"TEST.DAT"
Reset #1
Input #1,Titel$
Input #1,Anzahl%
Close #1
```

Befehle
Übersicht

Input VAR

```
VAR : Variable
```

Von der Tastatur wird ein Wert in die Variable gelesen (Textmodus). Beispiel:

```
Print "Geben Sie den Titel ein: ";
Input Titel$
Print "Geben Sie die Anzahl ein: ";
Input Anzahl%
```

Windowstypischer ist jedoch die Verwendung des Eingabedialoges mit der Funktion @Input$.

Befehle
Übersicht

Let VAR=N|S

```
VAR : Variablenname
N   : Wert
S   : String
```

Einer zuvor mit DECLARE declarierten Variablen wird ein Wert zugewiesen. Das LET muß (!) da stehen!

Befehle
Übersicht

Line - LineTo

Line X1,Y1-X2,Y2

```
X1,Y1 : Wert - Startkoordinaten
X2,Y2 : Wert - Endkoordinaten
```

Es wird eine Linie zwischen den angegebenen Punkten in der mit USEPEN eingestellten Linienart und Strichstärke gezeichnet.

LineTo X1,Y1

```
X1,Y1 : Wert - Endkoordinaten
```

Es wird eine Linie vom zuletzt gezeichneten (oder mit MOVETO gesetzten) Punkt zum angegebenen Punkt in der mit USEPEN eingestellten Linienart und Strichstärke gezeichnet.

Befehle
Übersicht

ListBoxItem$ N = S

```
N : Integer - Index (0 .. 999)
S : String
```

Der String S wird wird an Position N in die ListBox-Liste übernommen.

Befehle
Übersicht

List! - List% - List& - List$

List! N1 = N2

```
N1 : Integer - Index (0 .. 999)
N2 : Float - Wert
```

Der Wert N2 wird an Position N1 in die Float-Liste übernommen. Beispiel:

```
List& 1 = 5.67
List& 2 = 1045.324
Print @List!(1),@List!(2)
```

List$ N = S

```
N : Integer - Index (0 .. 999)
S : String
```

Der String S wird wird an Position N in die Stringliste übernommen.

List% N1 = N2

```
N1 : Integer - Index (0 .. 999)
N2 : Integer - Wert
```

Der Wert N2 wird an Position N1 in die Integerliste übernommen.

List& N1 = N2

```
N1 : Integer - Index (0 .. 999)
N2 : LongInt - Wert
```

Der Wert N2 wird an Position N1 in die LongInt-Liste übernommen.

Befehle
Übersicht

LoadBmp - LoadSizedBmp

```
LoadBmp S, X,Y ;N

S   : String - Name der Bilddatei
X,Y : Wert - Koordinaten
N   : Wert - Kopiermodus
```

Eine Windows-Bild-Datei (BMP-Format) wird geladen und unter Berücksichtigung des Kopiermodus angezeigt. S kann auch Pfadangaben enthalten. Das Bild muß im BMP-Format vorliegen. Es wird an der Position X,Y in Originalgröße angezeigt. Beispiel:

```
LoadBmp "C:\WINDOWS\PAPER.BMP",10,10;0
```

LoadSizedBmp S, X1,Y1-X2,Y2;N

```
S     : String - Name der Bilddatei
X1,Y1 : Wert - Koordinaten
X2,Y2 : Wert - Größe
N     : Wert - Kopiermodus
```

Eine Windows-Bild-Datei (BMP-Format) wird geladen und unter Berücksichtigung des Kopiermodus angezeigt. Das Bild wird an der Position X1,Y1 in der angegebenen Größe angezeigt.Durch negative Werte für X2 bzw. Y2 sind auch Spiegelbilder realisierbar.

Befehle
Übersicht

Locate X,Y

```
X : Wert - Zeile
Y : Wert - Spalte
```

Der nächste PRINT- oder INPUT-Befehl positioniert auf Zeile X, Spalte Y. Wirkt nur für Textmodus-Befehle.

Befehle
Übersicht

MessageBox S1,S2,N

```
S1 : String - Meldungstext
S2 : String - Überschrift
N  : Wert - Art der MessageBox
```

Eine MessageBox wird auf dem Bildschirm gebracht. N setzt sich zusammen aus BUTTONS + ICON + DEFAULT + FENSTERART. Der gedrückte Knopf ist in der Systemvariablen %BUTTON. Der Befehl ist nur wegen der Kompatibilität zu PROFAN 1.x vorhanden und kann für künftige Versionen nicht garantiert werden. Die @MESSAGEBOX-Funktion ist unbedingt vorzuziehen.
Beispiel:

```
MessageBox "Abbruch!","FEHLER",32
```

Befehle
Übersicht

MkDir S

```
S : String - Pfadangabe
```

Das Verzeichnis S wird angelegt. Im Falle eines Fehlers hat %IORESULT einen von 0 verschiedenen Wert. Beispiel:

```
MkDir "C:\WINDOWS\TEST"
```

Siehe auch: ChDir, RmDir

Befehle
Übersicht

MoveTo X1,Y1

`X1,Y2 : Wert - Koordinaten`

Der Grafik-Kursor wird zum angegebenen Punkt bewegt, ohne eine Linie zu zeichnen. Praktisch ist dieser Befehl nur im Zusammenhang mit nachfolgendem LINETO-Befehl sinnvoll.

Befehle
Übersicht

Music S

S: String (Noten)

Der Parameter S ist ein String in einer eigenen Makrosprache. Diese Makrosprache ist kompatibel zur Music-Makro-Sprache, wie sie von BASIC her (für den PLAY-Befehl) bekannt ist. Um eine Tonleiter in Viertel-Noten in der 3. Oktave zu spielen, reicht folgender Befehl:

```
MUSIC "O3 C4 D4 E4 F4 G4 A4 B4 > C4"
```

Erhöhungs- und Erniedrigungszeichen sind genauso möglich, wie auch Punktierungen:

```
MUSIC "C#2 F-8 G4."
```

Pausen haben den Notenwert "P".

PLAY "O3 C#4 D2 > D2 E4 E2."

Befehle
Übersicht

NumWidth N

```
N: Integer - Mindestweite
```

Mindestweite der Ausgabe numerischer Werte. Dieser Wert wird auch von der @STR$-Funktion und dem Befehl DrawText verwandt. Ist der Ausgabestring kürzer wird er mit führenden Leerzeichen aufgefüllt. Standarteinstellung ist 0.

Siehe auch: Decimals

Befehle
Übersicht

Orientation N

```
N : Integer - Winkel (in Zehntel-Grad!)
```

Bestimmt den Winkel der Textausgabe; wird durch den nächsten USEFONT- Befehl aktiviert. (Sinnvoll nur bei TRUE-Type-Fonsts.) Beliebige Textausrichtungen auf dem Bildschirm und Drucker sind somit möglich!

Befehle
Übersicht

Parameters VAR [,VAR[,VAR] ...]

VAR - Variablennamen

Die Aufrufparameter einer Prozedur werden Variablen zugewiesen, die innerhalb der Prozedur bekannt sind. Ein Beispiel findet sich unter PROC. Die Anzahl der Variablen sollte gleich der Anzahl der Übergabeparameter sein; überprüft wird dieses aber ebensowenig, wie die Datentypen. Gegebenenfalls erfolgt eine automatische Umwandlung.

Siehe aus: Proc

Befehle
Übersicht

Pie X1,Y1-X2,Y2; X3,Y3; X4,Y4

```
X1,Y1 : Wert - Links oben
X2,Y2 : Wert - Rechts unten
X3,Y3 : Start
X4,X5 : Ende
```

Es wird ein Kreisteil (Tortenstück) gezeichnet. X1,Y1 und X2,Y2 bezeichnen das Rechteck, indem der Kreis sich befindet, die anderen Koordinaten Start und Ende des Kreisteiles.

Befehle
Übersicht

Play N1,N2,N3

```
N1 : Wert - Notenwert (0 = Pause)
N2 : Wert - Dauer
N3 : Wert - Punkte
```

Die Notenwerte werden in Halbtonschritten hochgezählt. Die Dauer wird reziprok angegeben: 4 bedeutet z.B. 1/4 Note; 2 = 1/2 Note. Im Beispiel wird eine halbe punktierte Note gespielt:

```
Play 24,2,1
```

Hinweis: Dieser Befehl funktioniert auch ohne Soundkarte und benutzt den internen Lautsprecher. Siehe auch: Beep Sound

Befehle
Übersicht

PlaySound S,N

```
S : String - Dateiname (mit Pfad)
N : Integer - Spielmodus
```

Eine WAV-Datei wird über die Soundkarte (oder den SPEAKER-Treiber) abgespielt. Ist kein Soundtreiber vorhanden, passiert nichts.

Beispiel:

```
PlaySound "C:\MMDATA\GONG.WAV",1
```

Mögliche Werte für N:

- 0 - Die Sound wird im Vordergrund gespielt. Das Programm wird solange angehalten.
- 1 - Die Sound wird im Hintergrund gespielt. Das Programm läuft weiter.
- 8 - Ein Sound wird solange wiederholt, bis ein neuer gestartet wird.
- 16 - Der Sound wird nur gestartet, wenn kein anderer z.Zt. gespielt wird.

Kombinationen sind durch Addition möglich. Bei 8+0 gibts keine Wiederkehr!

Befehle
Übersicht

PopUp S

```
S : String - Name des Menüs
```

Ein PopUp-Menü wird an das FensterMenü angehängt:

```
PopUp "&Datei"
      AppendMenu "&Laden"
      AppendMenu "&Speichern"
PopUp "&Bearbeiten"
      ...
```

Befehle
Übersicht

Print #N,f;f,f ...

```
N : Wert - Dateikennzeichen (1..8)
f : druckbarer Ausdruck
```

Der Ausdruck, bzw. die Ausdrücke werden in die mit N bezeichnete Datei geschrieben. Steht zwischen den Ausdrücken ein Semikolon, so werden sie direkt aneinander gehängt, bei einem Komma wird ein Leerzeichen eingefügt.

Siehe auch unter ASSIGN, REWRITE, APPEND und CLOSE.

Befehle
Übersicht

Print f;f,f...

```
f : druckbarer Ausdruck
```

Der Ausdruck, bzw. die Ausdrücke werden auf den Bildschirm geschrieben. Steht zwischen den Ausdrücken ein Semikolon, so werden sie direkt aneinander gehängt, bei einem Komma wird ein Leerzeichen eingefügt. Siehe auch unter LOCATE. Beispiel:

```
Print "Dies ist der",N%;". Test"
```

Befehle
Übersicht

Proc <Name> ... EndProc

<Name> : Name der Prozedur

Eine Prozedur wird definiert, die im nachfolgenden Programm wie ein neuer Befehl verwandt werden kann. Über den PARAMETERS-Befehl werden die übergebenen Parameter eingelesen.

Beispiel:

```
PROC Wiederhole
      PARAMETERS Text$, Anzahl%
      DECLARE I%
      LET I%=0
      WHILE @LT(I%,Anzahl%)
            PRINT Text$
            INC I%
      WEND
ENDPROC

Wiederhole "Testtext",10
```

Die in der Prozedur über PARAMETERS oder DECLARE deklarierten Variablen sind nur innerhalb der Prozedur bekannt (lokal). Alle Variablen, die außerhalb (oberhalb) der Prozedur definiert wurden sind in der Prozedur auch bekannt (wie in PASCAL). Die Prozedur kann auch schon vor Erreichen der Zeile ENDPROC über RETURN verlassen werden, wobei dann die Übergabe eines Wertes an das aufrufende Programm möglich ist.

Befehle
Übersicht

Randomize

Sorgt dafür, daß die Zufallszahlen (@RND) auch wirklich zufälligen Charakter haben.

Befehle
Übersicht

Rectangle X1,Y1-X2,Y2

```
X1,Y1 : Wert - Links oben
X2,Y2 : Wert - Rechts unten
```

Es wird ein Rechteck innerhalb der angegebenen Koordinaten gezeichnet.

Befehle
Übersicht

Rename #N,S

```
N : Wert - Dateikennzeichen (1..8)
S : String  - neuer Name (mit Pfad)
```

Die Datei mit dem Kennzeichen N wird in S umbenannt. Enthält S eine Pfadangabe (mit oder ohne Laufwerksbezeichnung), so wird sie verschoben. Verschieben ist nur innerhalb eines Laufwerks möglich. Beispiel:

```
Assign #3,"C:\TEST.DAT"
Rename #3,"C:\UTILS\TOAST.DAT"
```

Befehle
Übersicht

RePaint

Malt den Bildschirm neu. Ist nur dann zu benutzen wenn bestimmte Befehlskombinationen (u.U. in Zusammenhang mit PopUp-Windows) den Bildschirm etwas durcheinander bringen. Dies sollte in der aktuellen PROFAN²-Version nicht mehr vorkommen.

Befehle
Übersicht

Reset #N

N : Wert - Dateikennzeichen (1..8)

Die Datei N wird zum Lesen geöffnet. Beispiel:

```
Declare Zeile$  Assign #2,"TEST.DAT"
Reset #2
WhileNot @Eof(#2)
      Input #2,Zeile$
      Print Zeile$
Wend
```

Befehle
Übersicht

Return [<Wert>]

Beenden eines Unterprogrammes oder einer Prozedur. Es ist möglich einen Wert zurückzugeben. Dieser steht dann in @$(0), @%(0), @&(0) und/oder @!(0):

```
PROC Test
      Print "Das ist ein Test"
      Return "OK!"
ENDPROC

Print "Im Hauptprogramm"
Test
Print "Wieder zurück! Ergebnis:",@$(0)
```

WICHTIG: Auch wenn eine Prozedur mit RETURN verlassen wird, muß die Definition der Prozedur mit ENDPROC abgeschlossen werden.

Befehle
Übersicht

Rewrite #N

N : Wert - Dateikennzeichen (1..8)

Die Datei N wird zum Schreiben geöffnet. Der bisherige Dateiinhalt geht verloren. Existiert sie nicht, wird sie erzeugt. Beispiel:

```
Assign #2,"C:\UTILS\TEST.DAT"
Rewrite #2
Print #2,Zeile$
Close #2
```

Um eine bestehende Datei zu erweitern, ist der Befehl APPEND zu verwenden.

Befehle
Übersicht

RmDir S

```
S : String - Verzeichnisnamen
```

Das Verzeichnis S wird gelöscht. Enthält es noch Dateien oder Unterverzeichnisse, kann es nicht gelöscht werden. Bei erfolgreicher Operation hat %IORESULT den Wert 0.

Sie auch: ChDir MkDir

Befehle
Übersicht

RoundRect X1,Y1-X2,Y2; X3,Y3

```
X1,Y1 : Wert - Links oben
X2,Y2 : Wert - Rechts unten
X3,Y3 : Wert - Rundung
```

Es wird ein abgerundetes Rechteck gezeichnet. X1,Y1 und X2,Y2 bezeichnen das Rechteck, die anderen Parameter den Grad der Rundung in waagrechter und senkrechter Richtung.

Befehle
Übersicht

Run S

```
S : String - Programmname
```

Das Programm S wird aufgerufen und das aufrufende Profanprogramm beendet. Es können alle Windowsprogramme - auch mit Parametern - aufgerufen werden. Beispiele:

```
Run "NOTEPAD HILFE.TXT"

Run "E:\PROFAN\PROFAN DEMO.RGH"

Run "WRITE"
```

Befehle
Übersicht

SaveBmp S,X1,Y1-X2,Y2

```
S:     String - Dateinahme (mit Pfad)
X1,Y1: Integer - linke obere Ecke
X2,Y2: Integer - rechte untere Ecke
```

Der angegebene Teil des Bildschirmes wird in eine BMP-Datei gespeichert. Die Dateiendung ".BMP" ist mit anzugeben. Beispiel:

```
SaveBmp "TESTBILD.BMP",10,10-200,140
```

Befehle
Übersicht

ScreenCopy [X1,Y1-X2,Y2]

```
X1,Y1: Integer - linke obere Ecke
X2,Y2: Integer - rechte untere Ecke
```

Der angegebene Bereich wird auf den Drucker kopiert und die Seite ausgedruckt. Ist kein Bereich angegeben, wird der gesamte Bildschirm ausgedruckt. Beispiele:

```
ScreenCopy 10,10 - 600,100
```

```
ScreenCopy
```

Siehe auch das Befehlspaar STARTPRINT ... ENDPRINT.

Befehle
Übersicht

Separator

Fügt dem PopUp-Menü eine Trennungslinie hinzu. Beispiel:

```
PopUp "&Datei"
      AppendMenu 100,"&Neu"
      AppendMenu 101,"&Laden"
      AppendMenu 102,"&Speichern"
      AppendMenu 103,"Speichern &als"
      Separator
      AppendMenu 104,"&Ende"
```

Befehle
Übersicht

SetErrorLevel N

`N: Integer - Errorlevel`

Je nach Wunsch kann der Errorlevel auf einen der möglichen Werte gesetzt werden. Damit wird das Verhalten des Systems bei Auftreten eines Fehlers oder einer Warnung definiert.

Warnungen treten auf, wenn ein Ausdruck nicht als numerischer Wert zu interpretieren ist oder z.B. eine Bilddatei nicht gefunden wird ...

Befehle
Übersicht

SetPixel X,Y,C

```
X,Y : Wert - Koordinaten
C   : Wert - RGB-Farbewert
```

Setzt an die Position X,Y ein Pixel in der Farbe C (bzw. der C am nächsten liegenden darstellbaren Farbe). Beispiel:

```
SetPixel 10,10,@RGB(13,20,3)
SetPixel 11,11,0
```

Befehle
Übersicht

Shell S

```
S : String - Programmname
```

Das Programm S wird aufgerufen und das aufrufende Profanprogramm läuft weiter. Es können alle Windowsprogramme - auch mit Parametern - aufgerufen werden. Beispiel:

```
Shell "NOTEPAD HILFE.TXT"  Vergleiche auch: Run
```

Befehle
Übersicht

ShowMax - ShowMin - ShowNormal

ShowMax

Das Bildschirmfenster wird zu seiner Maximalgröße erweitert. Der Befehl ist identisch mit dem Anklicken des Pfeiles nach oben an der rechten oberen Ecke des Rahmens.

ShowMin

Das Profanprogramm wird zum Icon verkleinert und läuft allerdings weiter. Vergrößert wird es entwededer durch SHOWNORMAL oder SHOWMAX oder aber durch entsprechendes Auswahl im Systemmenü [-] des Programmes.

ShowNormal

Das Bildschirmfenster des Profanprogrammes wird in normaler Größe angezeigt, das heißt z.B. in der Größe, die es vor dem Verkleinern (zum Icon) oder Vergrößern hatte.

Befehle
Übersicht

Sound N1,N2

```
N1 : Wert - Frequenz in Hertz
N2 : Wert - Dauer in 18tel Sekunden
```

Es wird ein Ton mit angegebener Frequenz und Dauer ezeugt. Während der Tonerzeugung wird das Programm angehalten.

Befehle
Übersicht

StartPrint ... EndPrint

Zwischen diesen beiden Befehlen werden alle grafischen Bildschirmausgaben (DRAWTEXT, RECTANGLE, ...) auf eine Druckeseite ausgegeben. "Textmodus"- und Bitmap-Befehle sind nicht gestattet. Mit ENDPRINT wird der Druck der Seite veranlaßt. Die Druckseite hat horizontal eine Weite von 640 Punkten und senkrecht eine Höhe von 960 Punkten. (Der Drucker sollte auf DIN A4 - Hochformat eingestellt sein.)

Befehle
Übersicht

StrWidth N

`N: Integer (Mindestweite)`

Die Mindest-Ausgabeweite für Strings (mittels des PRINT-Befehles) wird festgelegt. Strings, die kürzer als N sind, werden entsprechend mit vorangestellten Leerzeichen verlängert. Voreingestellt ist der Wert 0 (= "normales" Verhalten der Strings).

Befehle
Übersicht

Sub VAR,N

```
VAR - Variablenname (Integer)
N   - Integer
```

N wird von Variablen VAR subtrahiert. (Ist schneller und einfacher als die @SUB-Funktion.) Beispiel:

```
SUB Punkte%,10
```

Befehle
Übersicht

SubPopUp ... EndSub

Ein Untermenü zu einem Popup-Menü wird erzeugt. Eine Verschachtelung ist nicht möglich:

```
...
AppendMenu 210,"&Auswahl 3"
SubPopUp "&Schriften ..."
      AppendMenu 211,"Schrift &1"
      AppendMenu 212,"Schrift &2"
      ...
EndSub
AppendMenu ...
```

Befehle
Übersicht

TBox X1,Y1 - X2,Y2; N

```
X1,Y1 : Wert - linke obere Ecke
X2,Y2 : Wert - rechte untere Ecke
N     : Rahmentyp
```

Es wird im Textmodus eine Box gezeichnet. Die Koordinaten werden wie bei LOCATE in Zeichen eingegeben. Benutzt werden die Grafikzeichen des OEM-Zeichensatzes.

0 - Einfacher Rahmen
1 - Doppelter Rahmen
2 - Senkrecht doppelt/waagrecht einf.
3 - Ohne Rahmen

Befehle
Übersicht

TextColor C1,C2

```
C1 : Wert - Vordergrundfarbe (RGB)
C2 : Wert - Hintergrundfarbe (RGB)
```

Die Farben für die Ausgabe mittels DRAWTEXT werden festgelegt. Bei -1 als Hintergrundfarbe, ist diese transparant, d.h. der vorhandene Hintergrund wird genommen. Beispiel:

```
TextColor @RGB(15,15,31),-1
DrawText 23,67,"Hugo was here!"
```

Befehle
Übersicht

TrackMenu X,Y

`X,Y : Wert - Position`

Das Programm zeigt das mit CREATEMENU und APPENDMENU erzeugte Menü an der gewünschten Position an und erwartet die Auswahl. Das Ergebnis steht in %MENUITEM. Beispiel:

```
TrackMenu 20,130
Case %MenuItem(100):Gosub "Speichern"
Case %MenuItem(120):End
```

Befehle
Übersicht

UseBrush N,C

```
N : Wert - Art des Pinsels
C : Wert - Farbe des Pinsels (RGB)
```

Der Pinsel wird benutzt, um die Grafiken (RECTANGLE, ROUNDRECT, etc.) auszufüllen. C ist die benutzte RGB-Farbe (0 ... 32767), N bezeichnet die Art des Pinsels:

0 : Transparant (nicht ausgefüllt)
1 : Solid (voll gefüllt)
2 .. 8 : verschiedene Schraffuren

Schraffuren:

2 - waagrecht =====
3 - senkrecht |||||
4 - diagonal \\\\\
5 - diagonal /////
6 - kariert #####
7 - diag.kar. XXXXX

Die Schraffur wird in der Farbe C ausgeführt.

Befehle
Übersicht

UseCursor N

```
N : Wert - Cursortyp (0 .. 11)
```

Ändert das Aussehen des Mauszeigers. Es sind alle 11 von Windows vorgegebenen Mauszeiger verwendbar und zusätzlich ein Druckersymbol (während des Druckens.). Der Sanduhr-Zeiger sollte immer dann eingesetzt werden, wenn der Anwender warten muß und eh' keine Eingaben machen kann, z.B. beim Speichern und Laden.

Die Bedeutung von N:

0 - Zeiger (Standard)
1 - Texteingabe (senkr. Strich)
2 - Sanduhr (Bitte warten ...)
3 - Fadenkreuz
4 - Großer senkrechter Pfeil
5 - Vierfach-Pfeil (+)
6 - Icon (kleines Quadrat in Größerem
7 - Doppel-Pfeil (\)
8 - Doppel-Pfeil (/)
9 - Doppel-Pfeil (-)
10 - Doppel-Pfeil (|)
11 - Drucker

Befehle
Übersicht

UseFont S,N1,N2,N3,N4,N5

```
S  : String - Fontname
N1 : Wert - Zeichenbreite
N2 : Wert - Zeichenhöhe
N3 : Wert - Unterstrichen? (0 .. 1)
N4 : Wert - Kursiv? (0 .. 1)
N5 : Wert - Fett? (0 .. 1)
```

Der Font für die Ausgabe von Text mittels DRAWTEXT wird festgelegt.

N1: Zeichenhöhe. Steht hier 0, wird ein Defaultwert für Breite und Höhe genommen.
N2: Zeichenbreite. Steht hier 0, wird ein Defaultwert genommen
N3: 1 = fett , 0 = normal
N4: 1 = kursiv, 0 = normal
N5: 1 = unterstrichen, 0 = normal

Befehle
Übersicht

UseIcon S

```
S : String - Name des Icons
```

Es wird das Icon ausgewählt, das für das Programm verwandt wird, wenn es verkleinert wird.
Standardmäßig ist es das Profan-Icon. Es kann aber jedes der 18 "eingebauten" Icons verwandt werden.

Befehle
Übersicht

UsePen N1,N2,C

```
N1 : Wert - Linienart (0 .. 5)
N2 : Wert - Linienstärke
C  : Wert - RGB-Farbe
```

Die hier eingestellten Werte werden von den Zeichenbefehlen für die Rahmen bzw. Linien benutzt.
Beispiel:

```
UsePen 1,5,@RGB(0,0,31)
Rectangle 10,10 - 150,100
```

Linienart:

0 - durchgezogen _____
1 - gestrichelt _ _ _ _ _ _
2 - gepunktet .....
3 - Strich-Punkt _._._._._.
4 - Strich-Punkt-Punkt _.._.._..
5 - keine Linie

Die Werte 1 bis 5 sind nur bei einer Strichstärke von 1 sinnvoll.

Befehle
Übersicht

WaitInput - WaitKey - WaitMouse - WaitScan

WaitInput

Wartet auf einen Mausklick oder Tastendruck im Profan-Fenster. Das Ergebnis steht in den Systemvariablen %MOUSEKEY, %MOUSEX, %MOUSEY, %SCANKEY und %KEY.

WaitKey

Wartet auf einen Tastendruck. Das Ergebnis steht in der Systemvariablen %KEY. (Funktionstasten, sog. "virtuelle Tastencodes" werden nicht erkannt. Wird dies benötigt, ist WAITSCAN zu verwenden.)

WaitMouse

Wartet, bis eine Maustaste (im Profan-Fenster) gedrückt wird. Welche Taste es war, und die Position der Maus steht in den Systemvariablen %MOUSEKEY, %MOUSEX und %MOUSEY.

WaitScan

Wartet auf einen beliebigen Tastendruck. Auch Funktionstasten, Kusortasten, Shift- und Ctrltasten werden erkannt. Das Ergebnis steht in als virtueller Tastencode in %SCANKEY und - im Falle einer "normalen" Taste - als ANSI- Code in %KEY.

Wichtige Scancodes:

16 - Shift

17 - Strg

27 - Esc

33 - BildHoch

34 - BildRunter

35 - Ende

36 - Pos1

37 - Links

38 - Hoch

39 - Rechts

40 - Runter

45 - Einfg

46 - Entf

112 - F1

... - ...

123 - F12

Die übrigen lassen sich durch Ausprobieren mit folgendem Programm leicht austesten:

```
Cls
While 1    'Endlosschleife!
      WaitScan
      Print %ScanKey
Wend
```

```
End
```

Befehle
Übersicht

While | WhileNot N ... Wend

While N ... Wend

N : Wert - Bedingungsausduck

Die zwischen WHILE und WEND stehenden Befehle werden solange wiederholt, solange N ungleich 0 ist. Beispiel:

```
While N%
      Print "Gebe eine Zahl ein: ";
     Input N%
Wend
```

WhileNot N ... Wend

N : Wert - Bedingungsausduck

Die zwischen WHILENOT und WEND stehenden Befehle werden solange wiederholt, solange N gleich 0 ist. Beispiel:

```
WhileNot @Eof(#2)
      Input #2,Zeile$
      Print Zeile$
Wend
```

Befehle
Übersicht

Window X1,Y1-X2,Y2

```
X1,Y1 : Wert - linke obere Ecke
X2,Y2 : Wert - Größe in Pixeln
```

Das Fenster wird in der angegebenen Größe geöffnet. X1 und Y1 sind absolute Bildschirmkoordinaten.

Wenn nichts anderes angeben wird, benutzt Profan den ganzen (Standard-VGA-)Bildschirm (640 * 480 Pixel).

Befehle
Übersicht

WindowTitle S

```
S : String
```

Die Überschrift des Programmfensters wird festgelegt. Beispiel:

```
WindowTitle "Mein erstes Programm"
```

Wenn im Programm keine Überschrift festgelegt wird, wird eine voreingestellte Überschrift benutzt. (Der Befehl existiert nur in der Vollversion!)

Befehle
Übersicht

WriteIni S1,S2,S3=S4

```
S1: String - Name der INI-Datei
S2: String - Anwendung/Rubrik/Abschnitt
S3: String - Eintrag
S4: String - Wert des Eintrages
```

Ein Eintrag in einer INI-Datei wird geändert bzw. erstellt. Existiert die INI-Datei nicht, wird sie im angegebenen bzw. im Windowsverzeichnis erstellt.

Beispiel:

In der Datei SPIEL.INI soll unter der Rubrik [MeinProgramm] der Eintrag: "HIGHSCORE=25000" stehen:

```
LET HighScore& = 25000
WRITEINI "SPIEL.INI","MeinProgramm",\
         "HIGHSCORE"=@Str$(HighScore&)
```

Um die WIN.INI nicht zu überlasten, sollte jedes Programm eine eigene INI-Datei anlegen.

Befehle
Übersicht

$DosVer - $WinVer

$DOSVER

ermittelt die DOS-Version (z.B. "5.0")

$WinVer

ermittelt die Windows-Version (z.B. "3.10")

SystemVariablen
Übersicht

%GetCurSel

Nummer des in einer Listbox gewählten Eintrages.

Systemvariablen
Übersicht

$SysPath - $WinPath

$SysPath

Windows-System-Pfad (z.B. "C:\WINDOWS\SYSTEM")

$WinPath

Windows-Pfad (z.B. "C:\WINDOWS")

SystemVariablen
Übersicht

%WinTop - %WinBottom - %WinRight - %WinLeft

%WinTop

Die y-Koordinate des oberen Fensterrandes

%WinBottom

Die y-Koordinate des unteren Fensterrandes

%WinLeft

Die x-Koordinate des linken Fensterrandes

%WinRight

die x Koordinate des rechten Fensterrandes

SystemVariablen
Übersicht

Date$(N)

```
N : Integer - Ausgabeform

ERgebnis: String - Datum
```

Die Funktion ermittelt das aktuelle Datum in verschieden Formaten:

0 : Kurform (z.B. "09.04.1993")
1 : Langform (z.B. "9. April 1993")
2 : Langform mit Tag (z.B. "Freitag, der 9. April 1993")

Funktionen
Übersicht

GetClip$()

```
Ergebnis : String
```

Die Funktion liest Text aus dem Clipboard (max. 255 Zeichen). Die Zwischenablage wird bei Lesen nicht gelöscht. Dazu ist der Befehl ClearClip zu verwenden.

Sie auch: PutClip

Funktionen
Übersicht

@GetPixel(X,Y)

```
X : Integer - X-Koordinate
Y : Integer - Y-Koordinate

Ergebnis : Integer - Farbwert (RGB)
```

Farbwert des Bildpunktes an Position x,y.

ACHTUNG: Je nach Grafikmodus weicht dieser Wert von dem in SetPixel übergebenen Farbwert ab, da z.B. im 16-Farbmodus ein Punkt nur eine von 16 Farben haben kann.

Funktionen
Übersicht

@Time$(N)

```
N : Integer (0 .. 1) - Unterfunktion

Ergebnis: String
```

Die Funktion ermittelt die aktuelle Uhrzeit:

N = 0 : Sunden und Minuten (z.B. "23:45")
N = 1 : Sekunden und 100stel Sekunden (z.B. "39.67")

Funktionen
Übersicht

ClearClip

Die Zwischenablage wird gelöscht.

Befehle
Übersicht

PutClip S

```
S : String
```

Schreibt die Zeichenkette S (es kann auch ein numerischer Wert sein) als Text in die Zwischenablage. Beispiele:

```
PutClip "Das ist ein Test!"

PutClip Text$
```

Die Zwischenablage wird vor dem Schreiben nicht gelöscht.

Mittels der Zwischenablage können somit Daten zwischen verschiedenen Programmen ausgetauscht werden. Es ist sogar denkbar, daß ein Programm ein anderes über die Zwischenablage fernsteuert.

Siehe auch: ClearClip @GetClip$()

Befehle
Übersicht

WindowStyle N

```
N : Integer - Stil
```

Dieser Befehl muß vor der ersten Bildschirmausgabe verwandt werden, ansonsten bleibt er wirkungslos. Mit ihm wird der Fensterstil es Programmfensters bestimmt.

Werte für N:

1 - Fenster hat Vergrößerungsbox (Pfeil nach oben) und kann vergrößert werden

2 - Fenster hat Verkleinerungsbox (Pfeil nach unten) und kann zum Icon werden

4 - Dicker Rahmen (Größe des Fensters ist veränderbar)

8 - Fenster hat System-Menü

Kombinationen werden durch Addition erreicht. Die Standardeinstellung ist 15: Ein Fenster, das alles hat.

Befehle
Übersicht