Visualisierung mit comFORTH

- Windows-Dialoge
- Interaktive Programmierung von Fensterfunktionen
- Visual Basic Extensions

Windows-Dialogboxen

- Bedienerkommunikation über standardisierte Steuerelemente
  - vordefiniert: Druckknöpfe, Radioknöpfe, Kontrollkästchen, Roll-Leisten, Editierfenster, Auswahlboxen...
  - selbst definiert (Custom Controls): Schieberegler, Drehknöpfe, Registrierschreiber...

- Gestaltung durch Zusammensetzen von Bausteinen in speziellen Editoren:
  - SDK-Dialogeditor
  - MS Visual C Application Studio
  - Borland Resource Workshop

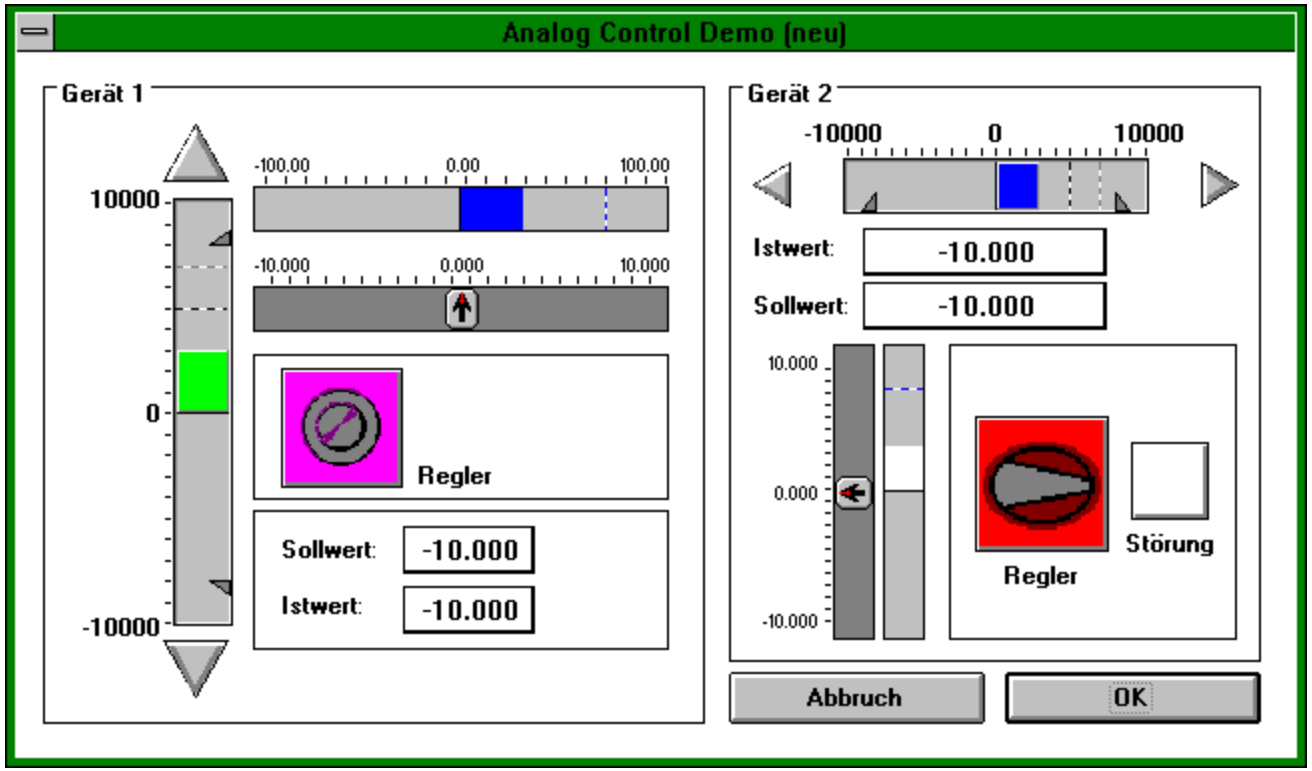
Interaktive Programmierung von Dialogboxen

```
DLGPROC: DEMODP

IDOK DEMODP ONCMD :WHEN
        MB_ICONASTERISK MessageBeep      \ Laut geben
        hWnd 0 EndDialog                 \ Dialog beenden
        ;WHEN

WM_CLOSE DEMODP ONMSG :WHEN
        hWnd 0 EndDialog
        ;WHEN
```


```
: DEMO  ( ps: ===> )( öffnet Aboutbox )
        hInstance 0 300     \ About-Dialog-Ressource
        hWndFrame           \ Parent-Window
        DEMODP lpDlgProc    \ FAR-Pointer der DialogProc
        DialogBox DROP ;    \ Dialog modal öffnen
```

Beispiel: Dialog mit Custom Controls

Testlauf im SDK-Dialogeditor:



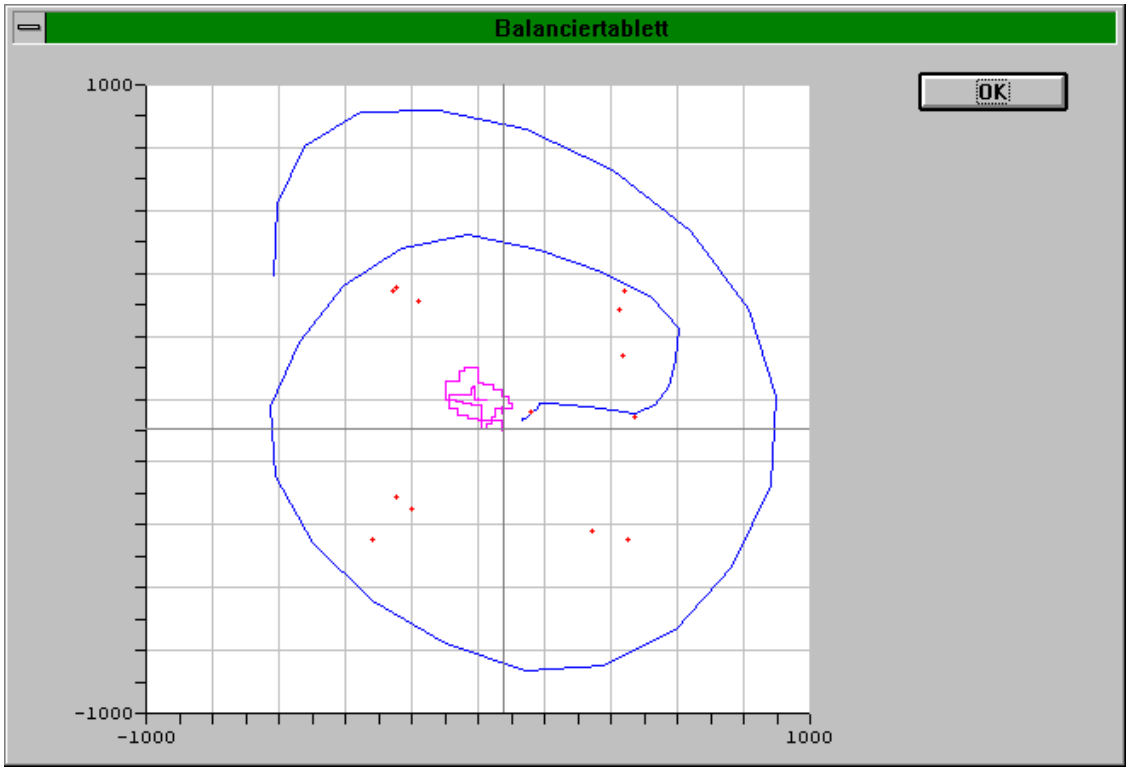
Visual Basic Extensions

- Nachteile von Custom Controls:
  - Interface über Messages
  - kein standardisierter Weg zur Einstellung von Parametern (nur über eigene Dialogbox)
  - stark eingeschränkte Einstellbarkeit von Laufzeitparameter zur Entwurfszeit

- Vorteile von Visual Basic Extensions:
  - Interface über spezielle Lese- und Schreibfunktionen
  - kein Zusatzaufwand für Parametereinstellung
  - Initialisierung mit nahezu beliebigen während des Entwurfs eingestellter Parameter

Beispiel: Dialog mit Visual Basic Extensions

Dialogbox unter comFORTH:



Einbindung von VBX-Controls

Beschreibung von VBX-Controls

```
VBXMODEL: vbmodelname
PRROPERTY: propname
...
;VBXMODEL
```

Definition von Controls:

```
id modelname VBX: vbxname
```

Benutzung von Controls:

```
vbxname propname VBX!
vbxname propname VBX@

vbmodelname propname VBX!
vbmodelname propname VBX@
```

Beispiel: Kurven zeichnen und Cursor setzen

```
3000 XY VBX: Diagram


: .XY ( ps: x y curve ctrl ===> )( xy-Paar anzeigen )
       VAL: Chart
       Chart XY CurveNr VBX! DROP
       Chart XY ValueY VBX! DROP
       Chart XY ValueX VBX! DROP ;


: @XY ( ps: ===> x y )( aktuelle Kursorposition )
       Diagram CursorX VBX@ DROP
       Diagram CursorY VBX@ DROP ;
```