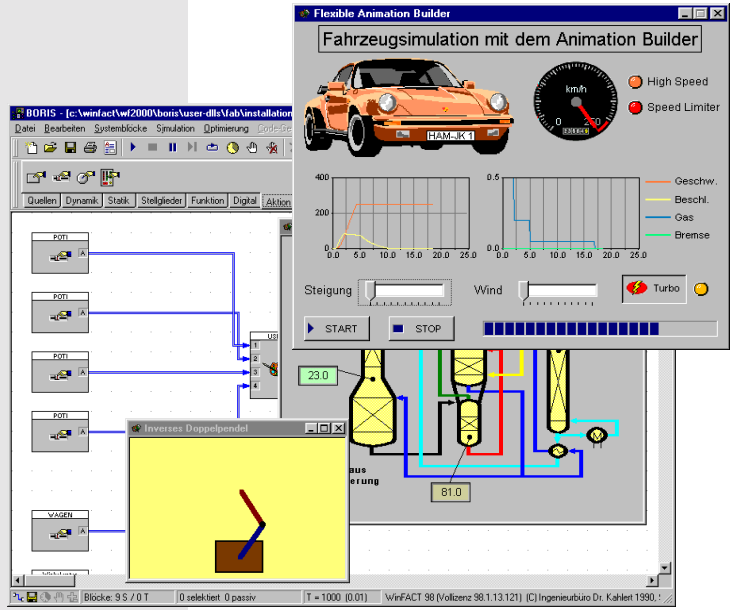
WinFACT 98



WinFACT 98

Benutzerhandbuch

Flexible Animation Builder

Ingenieurbüro Dr. Kahlert

Ludwig-Erhard-Str. 45
D-59065 Hamm

WinFACT 98

WINDOWS FUZZY AND CONTROL TOOLS

BENUTZERHANDBUCH

FLEXIBLE ANIMATION BUILDER

Release 5

Ingenieurbüro Dr. Kahlert
Ludwig-Erhard-Str. 45 D-59065 Hamm
Tel. 0 23 81/926 996 Fax 0 23 81/926 997

Inhalt

Einführung

Der *Flexible Animation Builder* (kurz *FAB*) für WinFACT 98 erlaubt erstmals die komfortable Erstellung einfacher bis komplexer Prozeßvisualisierungen, Animationen und Bedienoberflächen für das blockorientierte Simulationssystem BORIS ohne jegliche Programmierung. War die Umsetzung solcher Vorhaben bisher an die Realisierung mittels selbstprogrammierter User-DLLs gebunden, steht nunmehr ein leistungsfähiges Werkzeug zum interaktiven, direkten Entwurf zur Verfügung. Dazu stellt der FAB folgende Grundelemente zur Verfügung:

- Linien und Polylinien
- Kreise, Ellipsen, Rechtecke und abgerundete bzw. schattierte Rechtecke
- Dreiecke und Parallelogramme (drehbar)
- y-t- und x-y-Grafiken
- Mehrspaltige Tabellen
- Bitmaps und Windows-Metafiles
- Bitmap-Statusanzeigen und Bitmap-Sequenzen (animierte Bitmaps)
- *Video für Windows*-Dateien (AVI-Dateien)
- Sound-Unterstützung (WAV-Dateien)
- Statische Texte und Textmeldungen
- Zeit- und Datumsfelder
- Numerische Ausgabefelder
- Horizontale und vertikale Balkenanzeigen, Ablaufanzeigen, Thermometer
- LED-Anzeigen (rund oder rechteckförmig) und LCD-Panels
- Analoginstrumente (skalierbar)
- Schalter und Taster (wahlweise mit Text und/oder Grafik) sowie Bitmap-Schaltflächen und Schaltergruppen
- Schaltflächen mit Sonderfunktionen (z. B. zur Simulationssteuerung)
- Editierfelder

- Schaltfelder (Checkboxen)
- Horizontale und vertikale Schieberegler
- Horizontale und vertikale Leitungen (animiert)
- Tanksysteme
- Ventile, Hydraulikzylinder, Pumpen und Motoren (animiert)

Alle grafischen Elemente und Bedienelemente können beliebig miteinander verknüpft werden. Durch die Möglichkeit, die meisten Elementeigenschaften (z. B. Position oder Größe) an einzelne Blockein- oder -ausgänge anzukoppeln, lassen sich statische und dynamische Visualisierungen jeglicher Art realisieren. Weiterhin besteht die Möglichkeit, einzelne Elemente unabhängig voneinander zu- oder abzuschalten. Hinweise zu den Neuerungen in dem vorliegenden Release 5 finden Sie im Abschnitt *Neu in Release 5*.

Da es sich beim FAB-Kernel um eine DLL nach dem BORIS-User-DLL-Standard handelt, kann der FAB nicht nur als Tool innerhalb von BORIS, sondern auch völlig unabhängig davon als eigenständige Prozessvisualisierung für andere Applikationen - z. B. vom Benutzer programmierte Anwendungen - benutzt werden. Dazu muss der Anwender lediglich die entsprechenden Schnittstellenfunktionen der FAB-DLL in sein Programm einbinden und kann dann auf einfache und komfortable Weise Ausgaben seines Programms visualisieren bzw. Eingaben für sein Programm erzeugen. Einzelheiten dazu finden Sie später im Abschnitt *Einbindung des FAB in andere Applikationen*.

Auf den FAB-Installationsdisketten bzw. CD befindet sich eine ganze Reihe von Beispielen (BORIS-BSY-Dateien), die in das Unterverzeichnis *FAB-DEMOS* installiert werden und die unterschiedlichen Einsatzmöglichkeiten des *Flexible Animation Builder* demonstrieren. Dies sind im einzelnen:

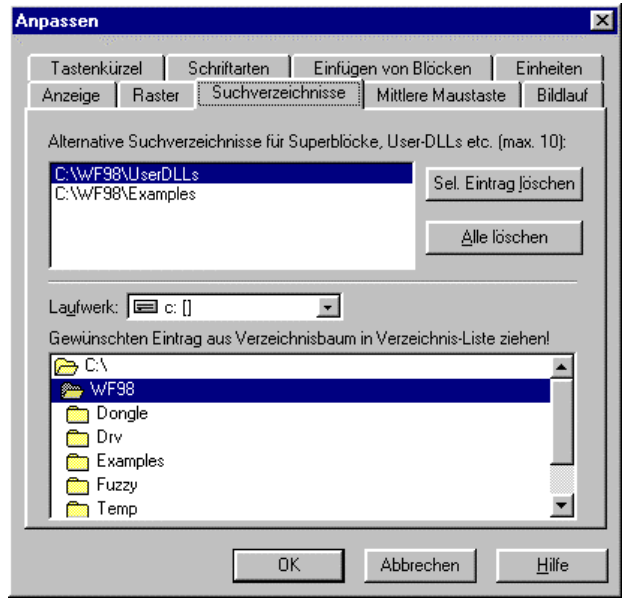
Datei	**Bedeutung**
DOUBLEPENDULUM.BSY	Inverses Doppelpendel
DOGANDBALL.BSY	Demo für die Nutzung transparenter Bitmaps
PROCESSVISU1.BSY	Einfache Prozeßvisualisierung
TRAFFICLIGHTS.BSY	Demo für die Nutzung des Anzeigestatus-Steuereingangs
SNOWMAN.BSY	Fenstergrößen-unabhängige Ausgabe
SEESAWANDBALL.BSY	Wippe mit Ball
PROCESSVISU2.BSY	Weitere Prozeßvisualisierung

COUNTERDEMO1.BSY	Demo zum Einsatz von Zählern
COUNTERDEMO2.BSY	Zähler mit eingangsabhängigem Inkrement
AVIDEMO.BSY	Demo zum Einsatz von AVI-Dateien
BARDEMO.BSY	Demo für *VBAR*/*HBAR*-Elemente
EXTBARGRAPH.BSY	Demo für *VBARGRAPH*-Element
CONTROLS.BSY	Bedienelemente-Demo
SOUNDDEMO.BSY	Sound-Demo
FYLINGWITCH.BSY	Demo für transparente bewegte Bitmaps
SLOTMACHINE.BSY	Einarmiger Bandit
LEDCUBE.BSY	LED-Würfel
CLOCK.BSY	Analoguhr
TIMEANDDATE.BSY	*TIME*- und *DATE*-Elementdemo
YTPLOTDEMO.BSY	Demo für den Einsatz des *YTPLOT*-Elements
XYPLOTDEMO.BSY	Demo für den Einsatz des *XYPLOT*-Elements
SIMCONTROL.BSY	Demo für den Einsatz des *BUTTON*-Elements zur Simulationssteuerung
SPINEDITDEMO.BSY	*SPINEDIT*-Elementdemo
UPDOWNDEMO.BSY	*UPDOWN*-Elementdemo
CARSIM.BSY	Bedienoberfläche für Fahrzeugsimulation
SWITCHGROUP.BSY	Demo für den Aufbau von Schaltergruppen.
ELECTRICCIRCUIT.BSY	*HPIPE*, *DYNBMP* und *STATEBMP*-Demo
TANKS.BSY	*HPIPE*, *VPIPE* und *TANK*-Demo
MOTOR.BSY	*MOTOR*-Demo
ASSEMBLYLINE.BSY	*MOTOR*-, *VPIPE* und *TANK*-Demo
VARSIZEDLG.BSY	Demo für aufklappbare Fenster
NAILPENDULUM.BSY	Fadenpendel mit Anschlag
PIDMOTOR.BSY	Drehzahlregelung mit PID-Regler

Installation

Die Installation des *Flexible Animation Builder* erfolgt vollständig automatisch. Legen Sie dazu die mit dieser Dokumentation gelieferte Diskette in Ihr Diskettenlaufwerk ein und starten Sie von der Diskette das Programm SETUP.EXE. Bei der nachfolgenden Installation müssen Sie dann lediglich das Programmverzeichnis angeben, in dem sich Ihre WinFACT 98-Installation befindet.

Sollten Sie das BORIS-User-DLL-Verzeichnis nicht als Suchverzeichnis definiert haben (in der Regel geschieht dies bei der Installation von WinFACT 98 automatisch), so müssen Sie dies beim nächsten Aufruf von BORIS zunächst nachholen, damit Sie später auf einfache Weise auf das FAB-Modul zugreifen können. Dazu rufen Sie über OPTIONEN | ANPASSEN... den Konfigurierungsdialog für die Suchverzeichnisse auf und fügen den entsprechen Pfad (z. B. C:\WF98\USERDLLS) hinzu.



Hinzufügen des User-DLL-Verzeichnisses zu den Suchverzeichnissen von BORIS

Arbeiten mit dem Flexible Animation Builder

Neu in Release 5

Das vorliegende Release 5 weist gegenüber der Version 4 im wesentlichen folgende Verbesserungen und Erweiterungen auf:

- Optionaler Fenstermanager zur komfortablen Verwaltung mehrerer FAB-Visualisierungsfenster zur Laufzeit
- Verarbeitung von Tastaturbotschaften (ermöglicht Auslösung von Ereignissen durch Tastendruck)
- Eingangsgesteuertes Anzeigen und Verbergen des Visualisierungsfensters
- Selektion und Bearbeitung von Elementgruppen (über <Strg>- bzw. <Shift>-Taste oder durch Aufziehen eines Rechtecks mit der Maus)
- Optimiertes Fenstermanagement im Entwurfsmodus (über Kontextmenü des Visualisierungsfensters)
- Wesentlich erweiterte Palette an Grafik- und Bedienelementen
- Modifizierter Konfigurierungsdialog mit getrennten Element- und Eigenschaftstabellen
- Steuerbare Elementfarben und Füllmuster
- Ändern der Elementreihenfolge per Drag & Drop
- Einfügen neuer Elemente per Drag & Drop an beliebiger Position
- Kopieren und Einfügen von Elementen
- Optionales Entwurfsraster im Visualisierungsfenster
- Selektieren und Verschieben von Elementen mit der Maus (Drag & Drop) in beiden Betriebsmodi
- Spezielle Schaltflächen zur Simulationssteuerung, zum Drucken etc.

- Auf Knopfdruck aufklappbare Visualisierungs- und Bedienfenster
- Integrierbare Online-Hilfe
- Verwendung von Konstanten
- I/O-Kontrollfenster
- Bitmap-Bibliotheken (standard und benutzerdefiniert)

Betriebsmodi des Flexible Animation Builder

Der *Flexible Animation Builder* kann in zwei unterschiedlichen Betriebsarten benutzt werden:

- Als eigenständiges Programm (FAB.EXE), das aus der WinFACT 98-Programmgruppe heraus aufgerufen wird.
- Als in BORIS integriertes Werkzeug, das automatisch beim Einfügen eines FAB-Moduls in eine BORIS-Struktur aktiviert wird.

Beide Betriebsarten sind bezüglich der Bedienung praktisch identisch. In den nachfolgenden Abschnitten wird zunächst die zweite Betriebsart beschrieben, da diese die gebräuchlichere Art der Nutzung darstellt. Die Arbeit mit dem FAB-Hauptprogramm wird am Ende dieser Dokumentation näher betrachtet.

Einfügen eines FAB-Moduls

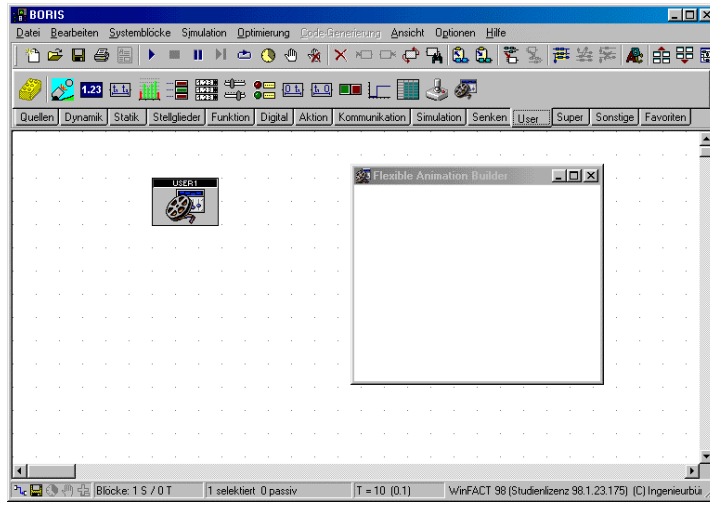
Der *Flexible Animation Builder* ist eine spezielle User-DLL für BORIS, die zunächst wie alle anderen User-DLLs gehandhabt wird. Es gibt daher drei verschiedene Möglichkeiten, ein FAB-Modul in eine Systemstruktur einzufügen:

- Durch Einfügen eines leeren User-DLL-Blocks und Angabe des Dateinamens für das FAB-Modul. Befindet sich Ihre WinFACT 98-Installation z. B. im Verzeichnis C:\WF98, so lautet der Dateiname für das FAB-Modul *C:\WF98\USERDLLS\FAB.DLL*. Dieser Weg ist der umständlichste von allen.
- Durch Aufruf des Moduls über das BORIS-Hauptmenü. Sie finden den Block dann unter SYSTEMBLÖCKE | USER-DLL-BLÖCKE | FLEXIBLE ANIMATION BUILDER.

- Über die Schaltfläche  der Palette *User* der Systemblock-Toolbar. Dies ist der einfachste Weg.

Die beiden letzten Möglichkeiten setzen voraus, dass das Unterverzeichnis *UserDLLs* zuvor als BORIS-Suchverzeichnis konfiguriert wurde (siehe Kapitel *Installation*).

Die nachfolgende Bildschirmgrafik zeigt das BORIS-Hauptfenster nach Einfügen eines FAB-Moduls.



BORIS-Hauptfenster mit eingefügtem FAB-Modul

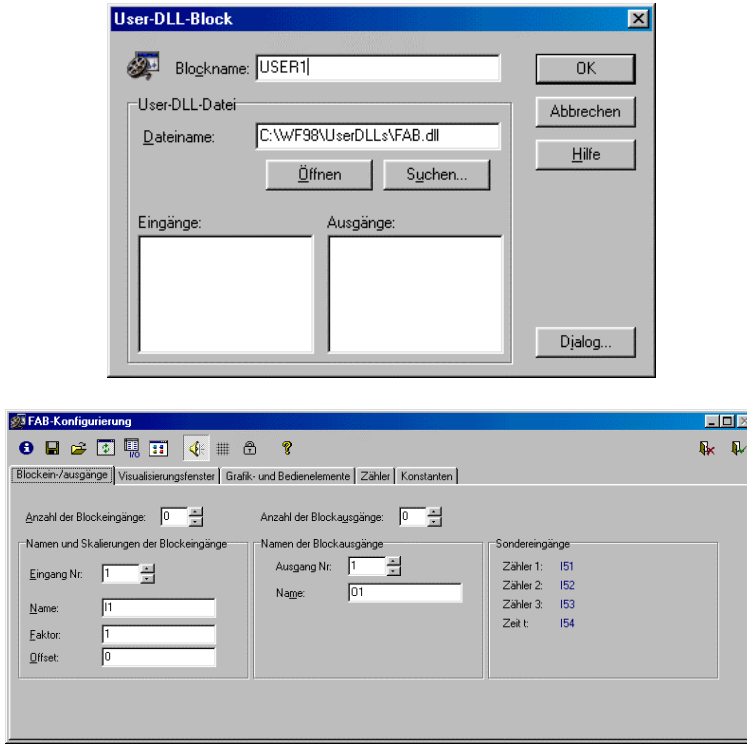
Neben dem FAB-Block – der zunächst noch keine Ein- oder Ausgänge aufweist – erscheint auch das FAB-Visualisierungsfenster in seiner voreingestellten Größe. Dieses Fenster ist zunächst noch leer; es wird später die benutzerdefinierte Visualisierung oder Animation sowie die Bedienelemente enthalten.

Konfigurierung der Modulein- und -ausgänge

Zur Konfigurierung des FAB-Blocks steht eine komfortable Benutzeroberfläche zur Verfügung, die über die Schaltfläche *Dialog...* des Standard-Parameterdialogs des User-DLLs-Blocks von BORIS aufgerufen wird. Die Be-

nutzeroberfläche besteht aus drei unabhängig voneinander verschiebbaren Fenstern:

- Dem eigentlichen Konfigurierungsdialog (siehe unten),
- dem Fenster mit den verschiedenen Grafik- und Bedienelementen (*Elementfenster*),
- dem *I/O*-Kontrollfenster zur Vorgabe von Blockeingangswerten während des Entwurfs bzw. zur Kontrolle der aktuellen Blockausgangswerte.



Aufruf des Konfigurierungsdialogs (unten) über den Standard-Parameterdialog des User-DLL-Blocks (oben)

Die Anzahl der Blockeingänge wird über das Feld *Anzahl der Blockeingänge* vorgegeben. Jeder Blockeingang kann mit einem eigenen Namen versehen wer-

Skalierung der Eingänge

den, der dann später in den entsprechenden Listboxen des User-DLL-Standarddialogs erscheint (Eingabefeld *Name*). Zusätzlich kann jeder Eingang über die Eingabefelder *Faktor* und *Offset* umskaliert werden, um z. B. physikalische Eingangsgrößen des Blocks an das spätere Grafik-Koordinatensystem anzupassen; eine externe Beschaltung des Blocks mit entsprechenden Skalierungsblöcken wird damit überflüssig. Ist *IU1* z. B. der aktuelle Eingangswert für Eingang 1, so ergibt sich der skalierte Wert *I1* zu

$$I1 = Faktor1 * IU1 + Offset1.$$

Blockausgänge

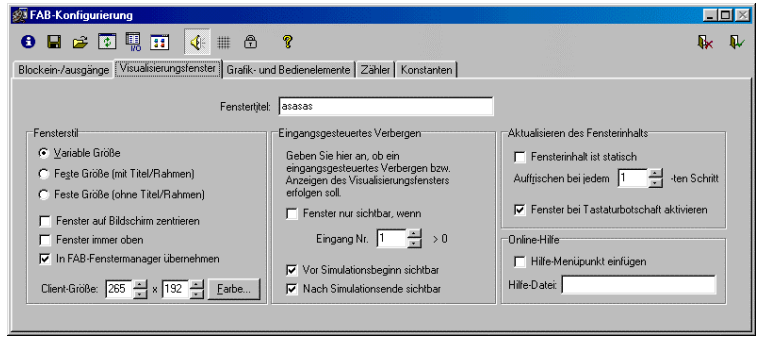
Die Anzahl der Blockausgänge (die später z. B. über Bedienelemente beeinflußt werden sollen) wird über das Feld *Anzahl der Blockausgänge* festgelegt. Die entsprechenden Namen für die einzelnen Ausgänge können - analog zur Festlegung der Blockeingangsnamen - innerhalb der Gruppenbox *Namen der Blockausgänge* spezifiziert werden.

Einstellungen des Visualisierungsfensters

Fensterstil

Die Palette *Visualisierungfenster* enthält einige für das Visualisierungsfenster wichtige Optionen. Über das Feld *Fenstertitel* kann der Titel des Visualisierungsfensters festgelegt werden. Das Fenster selbst kann eine feste oder einstellbare Größe besitzen; bei Fenstern mit fester Größe kann zudem Fenstertitel und -rahmen entfallen (Optionen *Feste Größe*, *Variable Größe (mit Titel/Rahmen)* sowie *Variable Größe (ohne Titel/Rahmen)*). Über *Client-Größe* kann dem Fenster eine exakte Größe für seinen Client-Bereich (Zeichenfläche) gegeben werden. Ist die *Option Fenster auf Bildschirm zentrieren* aktiviert, erscheint das Fenster unabhängig von der Bildschirmauflösung immer in der Bildschirmmitte. Eine Aktivierung der Option *Fenster immer oben* bewirkt, dass das Fenster später über allen anderen Anzeigefenstern (z. B. von BORIS-Standardinstrumenten) erscheint (Hinweis: Bei mehreren FAB-Blöcken innerhalb einer BORIS-Struktur sollte diese Option grundsätzlich immer nur für *maximal eins* der Fenster aktiviert werden, da es sonst zu unerwünschtem "Flakkern" kommen kann!). Weiterhin kann das Visualisierungsfenster mit einer beliebigen Hintergrundfarbe versehen werden (Schaltfläche *Farbe...*).

Wird die Option *In Fenstermanager übernehmen* aktiviert, so kann das Fenster später zur Laufzeit über den FAB-Fenstermanager (siehe Abschnitt *Der FAB-Fenstermanager*) verwaltet werden. Eine Anwahl dieser Option ist nur bei mehreren FAB-Blöcken innerhalb einer BORIS-Struktur sinnvoll.



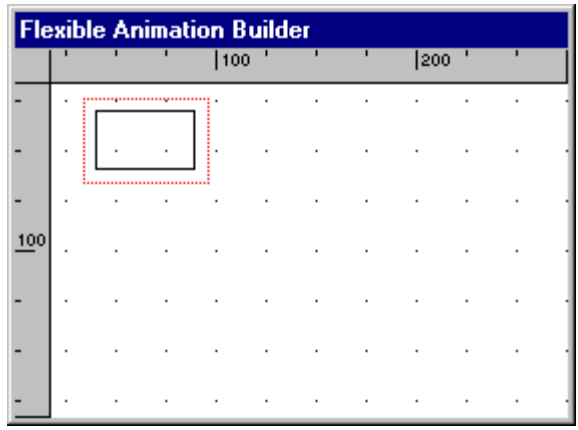
Palette Visualisierungsfenster *des Konfigurierungsdialogs*

Online-Hilfe

Über die Gruppenbox *Online-Hilfe zum Visualisierungsfenster* kann ein Hilfe-Menüpunkt in das Fenster integriert werden. Ist die Option *Hilfe-Menüpunkt einfügen* aktiviert, wird bei Anwahl des *Hilfe*-Menüpunktes die unter *Hilfe-Datei* angegebene Datei angezeigt. Dies kann entweder eine ASCII-Textdatei (Endung *TXT*) oder eine Windows-Hilfedatei (Endung *HLP*) sein. Alternativ kann der Aufruf dieser Hilfedatei auch über Bedienelement des Typs BUTTON erfolgen (siehe später).

Entwurfsraster

Während der Entwurfsphase kann als Hilfestellung ein Entwurfsraster mit horizontalem und vertikalem Lineal in das Visualisierungsfenster eingeblendet werden. Hierzu dient die Schaltfläche  der Toolbar des Konfigurierungsdialogs.



Visualisierungsfenster mit Entwurfsraster

Bei komplexen Visualisierungen kann es aus Geschwindigkeitsgründen sinnvoll sein, das Visualisierungsfenster nicht bei jedem Simulationsschritt aufzufrischen, sondern z. B. nur bei jedem zehnten Schritt. Die entsprechende Einstellung kann über das Editierfeld *Auffrischen bei jedem x-ten Schritt* vorgenommen werden. Sofern das Visualisierungsfenster lediglich Bedienelemente und statische (d. h. während der Simulation unveränderte) Grafikelemente enthält, kann das Auffrischen auch ganz entfallen. Hierzu dient das Optionsfeld *Fensterinhalt ist statisch*. Ist dieses aktiviert, ist der im darüberliegenden Editierfeld vorgegebene Wert ohne Bedeutung.

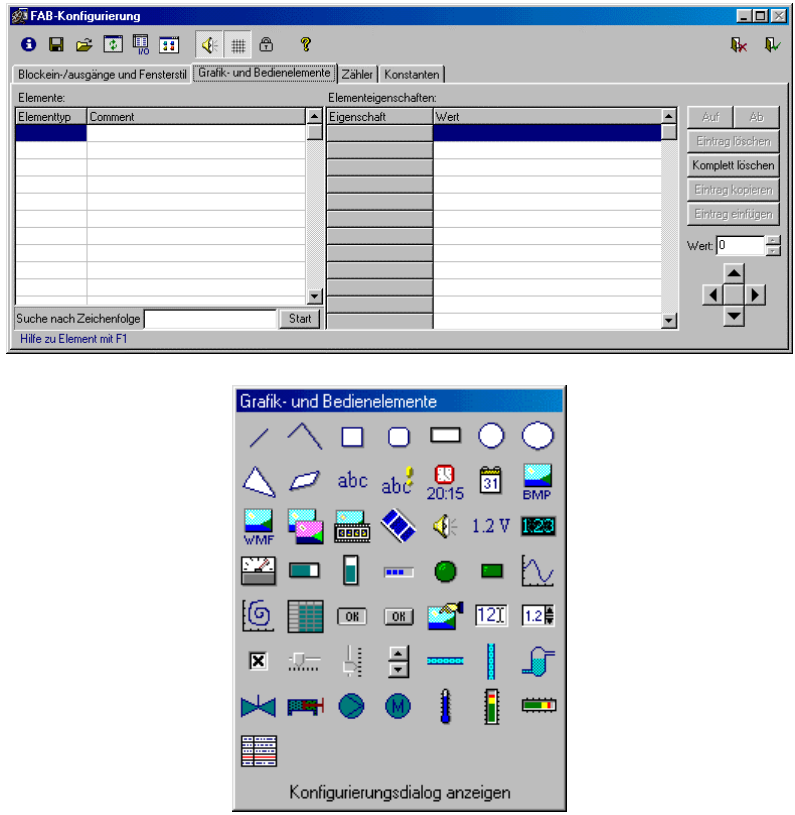
Über die Gruppenbox *Eingangsgesteuertes Verbergen* können Einstellungen vorgenommen werden, die zur Laufzeit ein über einen beliebigen Blockeingang gesteuertes Anzeigen und Verbergen des Visualisierungsfensters ermöglichen. Einzelheiten dazu finden Sie an späterer Stelle im Abschnitt *Eingangsgesteuertes Verbergen des Visualisierungsfensters*.

Die FAB-Grafik- und Bedienelemente

Konfigurierung der Elemente

Zur Konfigurierung der Grafik- und Bedienelemente dient die Palette *Grafik- und Bedienelemente* des FAB-Konfigurierungsdialogs. Sie ist nach Anlegen eines neuen FAB-Moduls zunächst leer (siehe nachfolgende Bildschirmgrafik). Die einzelnen Elementtypen befinden sich im frei schwebenden Fenster *Grafik- und Bedienelemente*, dem *Elementfenster*.

Der Dialog enthält zwei Tabellen: Die *Elementtabelle* (links) sowie die *Eigenschaftstabelle* (rechts). Alle aktuell vorhandenen Elemente mit einer benutzerdefinierten Beschreibung (Spalte *Comment*) werden in der Elementtabelle dargestellt. Die jeweiligen Eigenschaften des in der Elementtabelle aktuell selektierten Elements zeigt die Eigenschaftstabelle an. Die Breite der einzelnen Spalten beider Tabellen läßt sich mit der Maus beliebig verändern. Befindet sich die Maus über einem Eintrag, der nicht in voller Breite zu sehen ist, erscheint als Hilfestellung automatisch ein Hintfenster mit dem vollen Text. Über *Suche nach Zeichenfolge* kann die *Comment*-Spalte der Elementtabelle nach einer beliebigen Zeichenfolge durchsucht werden.



Palette Grafik- und Bedienelemente *des FAB-Konfigurierungsdialogs (oben) und Elementfenster (unten)*

Einfügen eines neuen Elements

Alle verfügbaren Elementtypen stehen über das Elementfenster zur Verfügung. Um ein neues Element einzufügen, gibt es zwei verschiedene Möglichkeiten:

- Durch einen *Einfachklick* mit der linken Maustaste auf die entsprechende Schaltfläche des Elementfensters fügen Sie das Element *an das Ende* der aktuellen Elementliste an.
- Durch Anklicken der Schaltfläche mit *festgehaltener* linker Maustaste gelangen Sie in den Drag&Drop-Modus und können das Element an eine beliebige Position innerhalb der aktuellen Elementliste ziehen. Das Element wird vor demjenigen Element der Liste eingefügt, über dem es "fallengelassen" wurde.

Für das Element der jeweils selektierten Zeile der Tabelle kann jederzeit über die Taste F1 eine spezifische Hilfe angefordert werden.

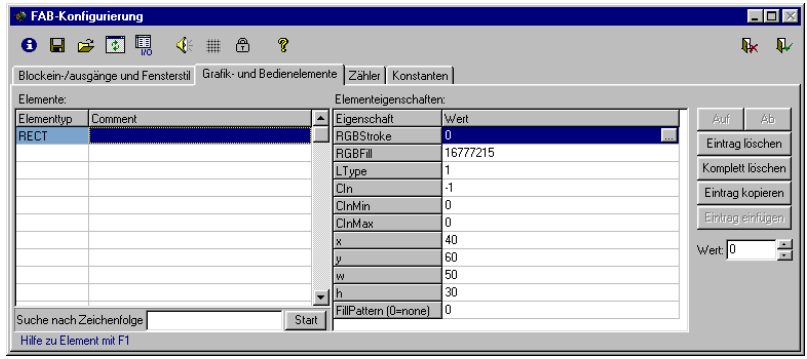
Erstellungsreihenfolge

Die Elemente werden später im Visualisierungsfenster in der Reihenfolge erstellt, in der sie sich in der Elementtabelle befinden (von oben nach unten). Diese Reihenfolge kann über die Schalter *Auf* und *Ab* jederzeit beliebig geändert werden. Alternativ dazu können einzelne Einträge der Elementtabelle auch per Drag & Drop verschoben werden. Dazu wird der zu verschiebende Eintrag der Elementtabelle angeklickt und dann bei festgehaltener Maustaste an die gewünschte Zielposition verschoben. Das verschobene Element wird vor demjenigen Eintrag eingefügt, auf dem es fallengelassen wurde.

Ändern der Reihenfolge

Kopieren und Einfügen

Über den Schalter *Eintrag kopieren* wird das aktuell selektierte Element mit sämtlichen Elementeigenschaften in eine temporäre Datei kopiert, von wo aus es über den Schalter *Eintrag einfügen* jederzeit wieder hinter das letzte Element der Elementtabelle eingefügt werden kann. Auf diese Weise kann eine einfache Duplizierung eines konfigurierten Elementes erfolgen.



Dialog nach Einfügen eines neuen Elements vom Typ RECT

Farbkennzeichnung von Eigenschaften

Die aktuellen Werte der Elementeigenschaften (rechte Spalte der Eigenschaftstabelle) werden zur besseren Übersichtlichkeit in verschiedenen Farben dargestellt. Alle Einträge, die als konstante Zahlenwerte (Ganzzahl- oder Fließkommazahl) interpretiert werden können, werden in schwarz dargestellt. Einträge, die als Formeln (siehe später) interpretiert werden können, erscheinen in grüner Schrift. Alle anderen Einträge schließlich werden in blau ausgegeben.

Löschen von Einträgen

Um ein einzelnes Element zu löschen, klicken Sie zunächst auf eine beliebige Zelle innerhalb der entsprechenden Elementtabellenzeile und betätigen dann die *Eintrag löschen*-Schaltfläche. Um sämtliche Elemente zu löschen, klicken

Sie auf die *Komplett löschen*-Schaltfläche. In diesem Fall erscheint vor dem Löschen zunächst eine Sicherheitsabfrage.

Ein einzelnes Grafik- oder Bedienelement kann auch direkt innerhalb des Visualisierungsfensters durch Anklicken mit der Maus selektiert werden. Bei festgehaltener linker Maustaste kann das Element dann mit der Maus beliebig innerhalb des Visualisierungsfensters verschoben werden. Die Koordinaten des Elements werden während des Verschiebevorgangs automatisch in der Eigenschaftstabelle aktualisiert. Weist allerdings eine der Elementkoordinaten einen *Formelausdruck* statt eines festen Zahlenwertes auf (siehe später), so ist ein Verschieben mit der Maus *nicht* möglich, da ansonsten beim Verschiebevorgang der Formelausdruck durch den aktuellen Koordinatenwert überschrieben würde. Um versehentliches Verschieben von Elementen mit der Maus zu verhindern, kann die Verschiebefunktion über die Schaltfläche  der Toolbar des Konfigurierungsdialogs auf Wunsch deaktiviert werden.

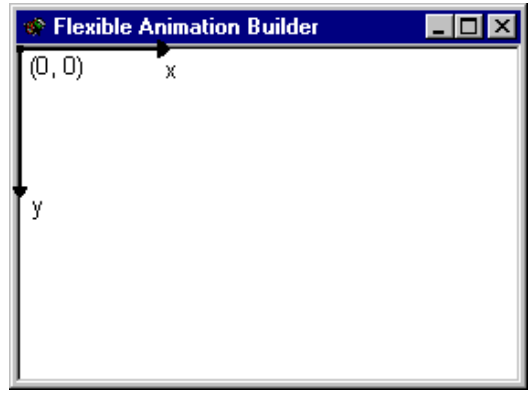
Selektieren mehrerer Elemente

Eine gleichzeitige Selektierung (und z. B. nachfolgende Verschiebung) mehrerer Elemente ist ebenfalls möglich - und zwar sowohl innerhalb der Elementtabelle als auch direkt im Visualisierungfenster. Halten Sie dazu während der Selektierung einfach die <Strg>- oder <Shift>-Taste gedrückt oder ziehen Sie im Visualisierungsfenster mit der Maus ein Selektionsrechteck auf. Statt das bzw. die selektierten Elemente mit Hilfe der Maus zu verschieben (was in der Regel nur relativ grob möglich ist), können auch die *Navigationstasten* in der rechten unteren Dialogecke benutzt werden. Diese erlauben ein pixelgenaues Verschieben der aktuell selektierten Elemente in sämtliche Richtungen.

Navigationstasten

Koordinatensystem des FAB-Visualisierungsfensters

Sämtliche Koordinaten von Grafik- und Bedienelementen sind in Pixeln anzugeben. Dabei wird die x-Koordinate von links nach rechts, die y-Koordinate von oben nach unten angesetzt. Der Punkt (0, 0) liegt dabei in der linken oberen Ecke des Visualisierungsfensters (siehe nachfolgende Grafik).

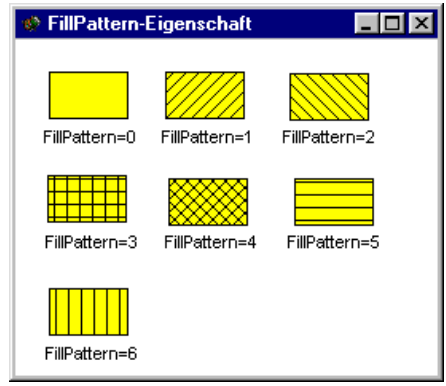


Koordinatensystem des FAB-Visualisierungsfensters

Grundlegende Elementeigenschaften

Das Erscheinungsbild der unterschiedlichen Elemente, die Ihnen der *Flexible Animation Builder* zur Verfügung stellt, kann über die jeweiligen Elementeigenschaften gesteuert werden. Einige dieser Elementeigenschaften existieren für sämtliche (oder fast sämtliche) Elementtypen; diese sollen hier zunächst erläutert werden.

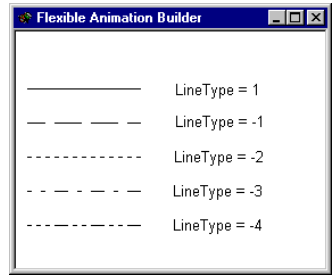
Eigenschaft	Bedeutung
RGBStroke	Stiftfarbe des Elements bzw. Textfarbe bei den Elementtypen *LABEL*, *NUMBER*, *SWITCH*, *BUTTON* und *EDIT*
RGBFill	Füllfarbe des Elements. Wird *RGBFill* auf -1 gesetzt, erhält das Element keine Füllung, erscheint später also transparent.
FillPattern	Füllmuster des Elements (nur von Bedeutung bei geschlossenen Formen wie z. B. *RECT*- oder *CIRCLE*-Element). Ist *FillPattern = 0*, wird kein Füllmuster ausgegeben.



Bedeutung von FillPattern

LType

Falls *LType* > 0 ist, gibt diese Eigenschaft die Linienbreite in Pixeln an; für *LType* = 0 wird keine Linie gezeichnet. Wird *LType* < 0 gewählt, lassen sich dadurch verschiedene Linientypen darstellen (siehe untenstehende Grafik). Letztere können allerdings nur in 1-Pixel-Breite gezeichnet werden.



Unterschiedliche Linientypen

CIn

Steuereingang für den Anzeigestatus (sichtbar/unsichtbar) des Elements. Ist dieser Wert -1 (Voreinstellung), so ist das Grafikelement immer sichtbar. Ist der Wert z. B. 2, wird der Anzeigestatus über Blockeingang 2 gesteuert. Das Element ist dann sichtbar, wenn der Wert der Eingangsgröße 2 im Bereich [*CInMin*, *CInMax*] (s. u.) liegt.

Bei Bedarf kann der Anzeigestatus auch über einen Block*ausgang* gesteuert werden. Dazu ist für *CIn* die Nummer des Blockausgangs + 100 anzugeben. Beispiel: Ist der Wert z. B.

	103, wird der Anzeigestatus über Blockausgang 3 gesteuert. Das Element ist dann sichtbar, wenn der Wert der Ausgangsgröße 3 im Bereich [*CInMin*, *CInMax*] (s. u.) liegt.
CInMin	Legt die untere Grenze des Bereichs für den über *CIn* spezifizierten Steuerein- bzw. -ausgang fest, bei dem das Element sichtbar ist.
CInMax	Legt die obere Grenze des Bereichs für den über *CIn* spezifizierten Steuerein- bzw. -ausgang fest, bei dem das Element sichtbar ist.
Comment	Kommentartext zum Grafikelement (kann vom Anwender frei vergeben werden)

Nutzung von Bitmaps

Eine Reihe der in den folgenden Abschnitten beschriebenen Grafik- und Bedienelemente basiert auf Windows-Bitmaps, die für verschiedene Anzeigezwecke benutzt werden können (z. B. als Grafik auf Schaltflächen). Diese Bitmaps können auf zwei unterschiedliche Arten spezifiziert werden:

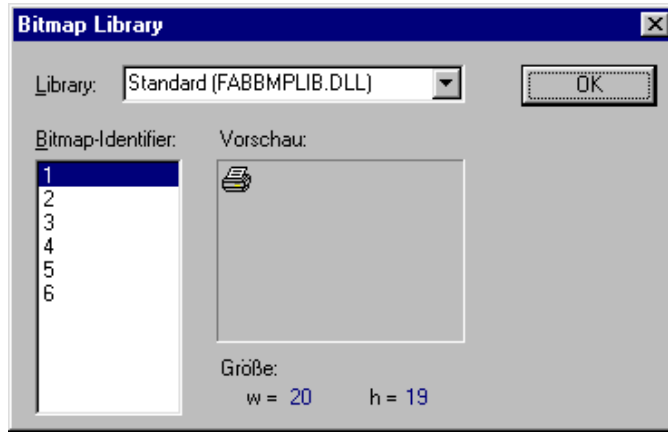
- Als eigenständige Bitmap-Dateien im Windows-BMP-Format. In diesem Fall müssen *alle* BMP-Dateien, die für eine entsprechende FAB-Visualisierung oder -Bedienoberfläche benötigt werden, bei einer Installation auf einem anderen Zielrechner zusammen mit der BORIS-Datei (BSY-Datei) kopiert werden, damit sie dort auch angezeigt werden können.
- Als Windows-Bitmap-Ressourcen innerhalb einer Windows-DLL (*Dynamic Link Library*). Diese DLL braucht außer den Bitmaps selbst keinerlei Funktion o. ä. zu enthalten.

Der *Flexible Animation Builder* erlaubt den Zugriff auf Bitmaps aus zwei unterschiedlichen Libraries: einer mit dem FAB gelieferten DLL mit Standard-Bitmaps für die verschiedensten Zwecke (FABBMPLIB.DLL) sowie einer benutzerdefinierten DLL (FABUSERBMPLIB.DLL), die vom Anwender mit eigenen Bitmaps gefüllt werden kann. Zur Erstellung einer solchen, lediglich Bitmap-Ressourcen enthaltenden DLL eignet sich praktisch jede Windows-Entwicklungsumgebung wie z. B. DELPHI, Visual C++ o. ä.

Die Identifikation eines Bitmaps innerhalb der DLL geschieht über eine eindeutige, ganzzahlige Kennziffer (*Identifier*). Zugelassen sind alle Identifier zwischen 1 und 9999. Da die Standard-Bitmaps aus FABBMPLIB.DLL Identifier besitzen, die bei 0 beginnen und dann steigend durchnummeriert sind,

sollten die Identifier der benutzerdefinierten Bitmaps aus FABUSERBMPLIB.DLL bei 9999 beginnen und dann rückwärts laufen. Auf diese Weise wird eine Überschneidung der Identifier vermieden.

Um auf schnelle und komfortable Weise das gewünschte Bitmap zu finden, können die beiden Bibliotheken durchsucht werden. Dazu dient der *Bitmap Library Viewer*, der aus dem Konfigurierungsdialog über die Schaltfläche auf gerufen wird (siehe nachfolgende Bildschirmgrafik).



Bitmap Library Viewer

Das Listenfeld *Library* ermöglicht hier zunächst die Auswahl zwischen den beiden DLLs. In der Liste *Bitmap-Identifier* werden daraufhin die zur Verfügung stehenden Bitmaps angezeigt und können durch Anklicken im *Vorschau*-Bereich des Dialogs betrachtet werden. Unterhalb dieses Bereichs werden Breite *w* und Höhe *h* des Bitmaps in Pixeln angezeigt.

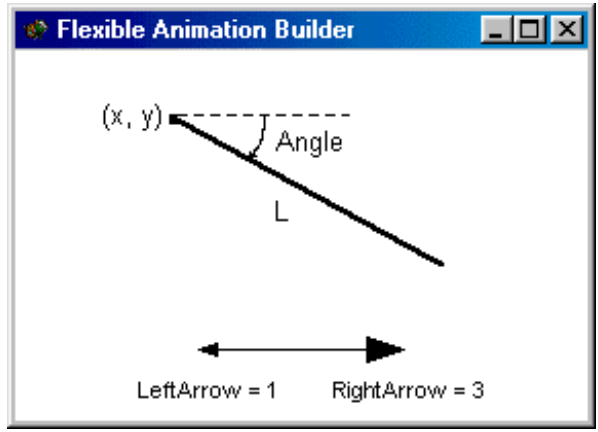
Hinweis: Bei der Installation des FAB wird automatisch eine Datei mit Namen FABUSERBMPLIB.DLL mitinstalliert. Diese ist jedoch leer, enthält also noch keine Bitmaps. Sie kann also ohne Bedenken durch eine anwendereigene DLL überschrieben werden.

Spezielle Elementeigenschaften

Elementtyp *LINE*

Der Elementtyp *LINE* stellt eine Linie dar, die über die Elementeigenschaften *x*, *y*, *L* und *Angle* spezifiziert wird. Bei Bedarf kann die Linie an einem Ende oder an beiden Enden mit einer Bepfeilung (Eigenschaften *LeftArrow* bzw. *RightArrow*) versehen werden.

Eigenschaft	Bedeutung
x	x-Koordinate des Linien-Anfangspunktes
y	y-Koordinate des Linien-Anfangspunktes
L	Länge der Linie in Pixeln
Angle	Winkel der Linie in Radiant
LeftArrow	Größe der Bepfeilung am linken Linienende. Für *LeftArrow = 0* wird keine Bepfeilung gezeichnet.
RightArrow	Größe der Bepfeilung am rechten Linienende. Für *RightArrow = 0* wird keine Bepfeilung gezeichnet.

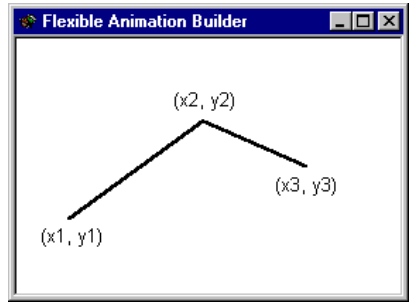


Grafikelement LINE *(unten mit Bepfeilung am linken und rechten Linienende)*

Elementtyp *POLYLINE*

Das Grafikelement POLYLINE stellt einen Linienzug aus drei Punkten mit den Eigenschaften *x1*, *y1*, *x2*, *y2*, *x3*, *y3* dar.

Eigenschaft	Bedeutung
x1	x-Koordinate des ersten Punktes
y1	y-Koordinate des ersten Punktes
x2	x-Koordinate des zweiten Punktes
y2	y-Koordinate des zweiten Punktes
x3	x-Koordinate des dritten Punktes
y3	y-Koordinate des dritten Punktes

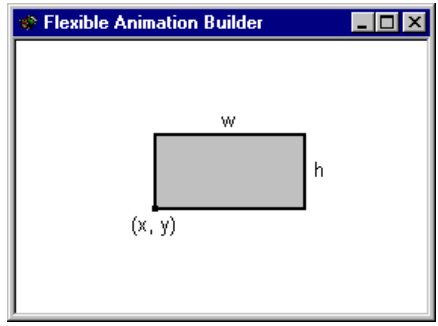


Grafikelement POLYLINE

Elementtyp *RECT*

Das Grafikelement *RECT* stellt ein Rechteck mit den Eigenschaften *x*, *y*, *w* und *h* dar.

Eigenschaft	Bedeutung
x	x-Koordinate des linken unteren Eckpunktes
y	y-Koordinate des linken unteren Eckpunktes
w	Breite des Rechtecks in Pixeln
h	Höhe des Rechtecks in Pixeln

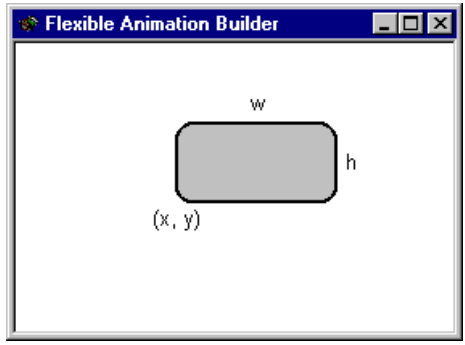


Grafikelement RECT

Elementtyp *RRECT*

Das Grafikelement *RRECT* stellt ein abgerundetes Rechteck mit den Eigenschaften *x*, *y*, *w* und *h* dar.

Eigenschaft	Bedeutung
x	x-Koordinate des linken unteren Eckpunktes
y	y-Koordinate des linken unteren Eckpunktes
w	Breite des Rechtecks in Pixeln
h	Höhe des Rechtecks in Pixeln

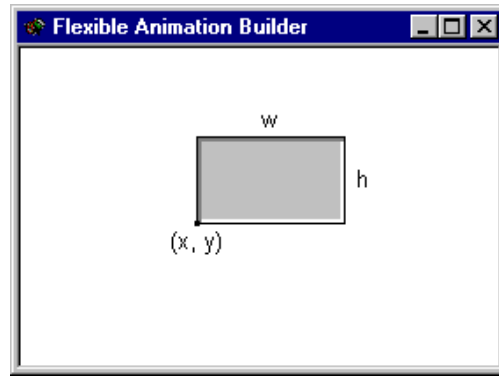


Grafikelement RRECT

Elementtyp *SHADRECT*

Das Grafikelement *SHADRECT* stellt ein innen schattiertes Rechteck mit den Eigenschaften *x*, *y*, *w* und *h* dar.

Eigenschaft	Bedeutung
x	x-Koordinate des linken unteren Eckpunktes
y	y-Koordinate des linken unteren Eckpunktes
w	Breite des Rechtecks in Pixeln
h	Höhe des Rechtecks in Pixeln

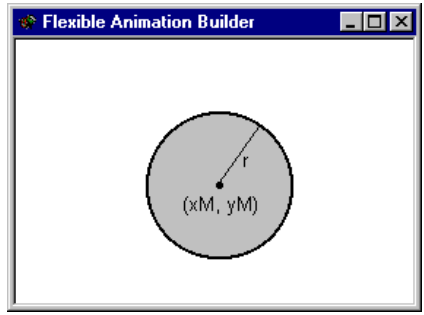


Grafikelement SHADRECT

Elementtyp *CIRCLE*

Der Elementtyp *CIRCLE* stellt einen Kreis dar und wird über die drei Elementeigenschaften *xM*, *yM* und *r* spezifiziert.

Eigenschaft	Bedeutung
xM	x-Koordinate des Kreis-Mittelpunktes
yM	y-Koordinate des Kreis-Mittelpunktes
r	Kreisradius in Pixeln

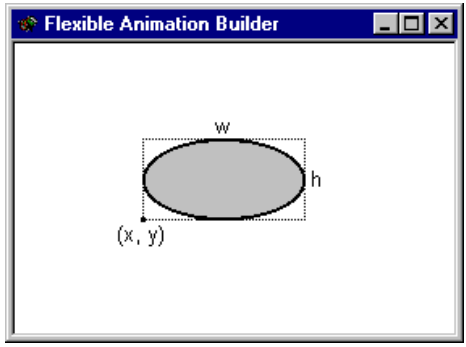


Grafikelement CIRCLE

Elementtyp *ELLIPSE*

Das Grafikelement ELLIPSE stellt eine Ellipse mit den Eigenschaften *x*, *y*, *w* und *h* dar.

Eigenschaft	Bedeutung
x	x-Koordinate des linken unteren Eckpunktes des umgebenden Rechtecks
y	y-Koordinate des linken unteren Eckpunktes des umgebenden Rechtecks
w	Breite des umgebenden Rechtecks in Pixeln
h	Höhe des umgebenden Rechtecks in Pixeln

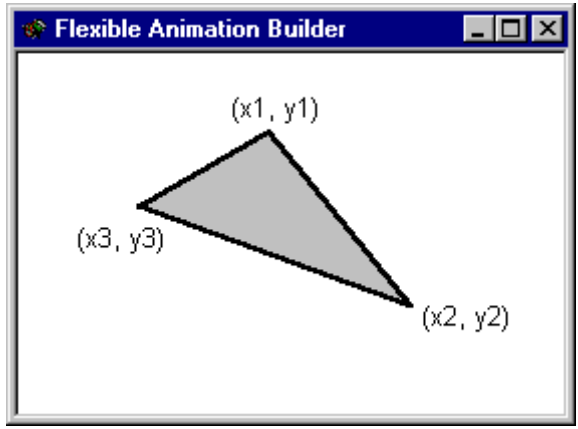


Grafikelement ELLIPSE

Elementtyp *TRIANG*

Das Grafikelement TRIANG stellt ein Dreieck mit den Eigenschaften *x1*, *y1*, *x2*, *y2*, *x3*, *y3* dar.

Eigenschaft	Bedeutung
x1	x-Koordinate des ersten Punktes
y1	y-Koordinate des ersten Punktes
x2	x-Koordinate des zweiten Punktes
y2	y-Koordinate des zweiten Punktes
x3	x-Koordinate des dritten Punktes
y3	y-Koordinate des dritten Punktes



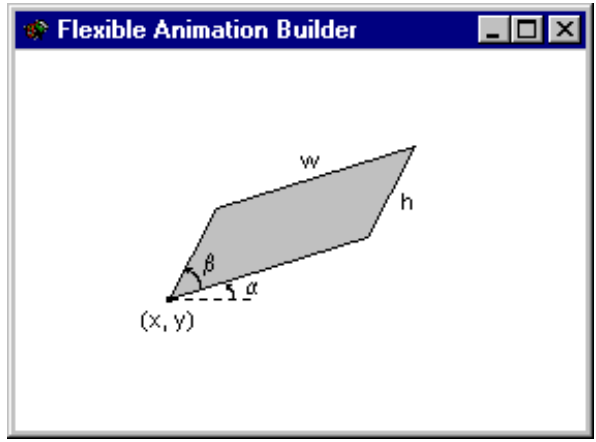
Grafikelement TRIANG

Elementtyp *PARALL*

Das Grafikelement PARALL stellt ein drehbares Parallelogramm mit den Eigenschaften *x*, *y*, *w*, *h*, α und β dar.

Eigenschaft	Bedeutung
x	x-Koordinate des linken unteren Eckpunktes

y	y-Koordinate des linken unteren Eckpunktes
w	Länge der Längsseite des Parallelogramms in Pixeln
h	Länge der Querseite des Parallelogramms in Pixeln
α	Drehwinkel des Parallelogramms in Radiant
β	Spreizwinkel des Parallelogramms in Radiant



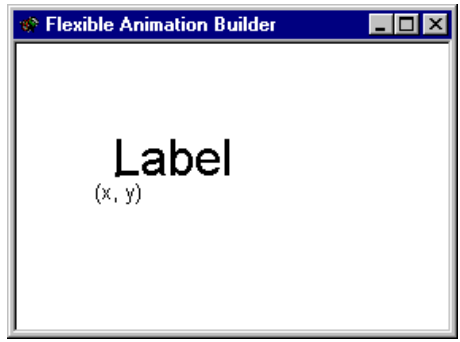
Grafikelement PARALL

Elementtyp *LABEL*

Der Elementtyp *LABEL* dient zur Ausgabe einzeiliger statischer Texte (optional mit Rahmen) und besitzt die nachfolgend aufgelisteten Eigenschaften.

Eigenschaft	Bedeutung
RGBStroke	Textfarbe
x	x-Koordinate des linken unteren Eckpunkts des Textes
y	y-Koordinate des linken unteren Eckpunkts des Textes
Size	Textgröße
Font	Schriftart

Caption	Auszugebender Text
BorderType	Typ des Rahmens um den ausgegebenen Text. Folgende Typen sind verfügbar: 0 kein Rahmen 1 Rahmen in Textfarbe, keine Füllung 2 Rahmen in Textfarbe, Füllung in *RGBFill* 3 Schattierter Rahmen in schwarz, keine Füllung 4 Schattierter Rahmen in schwarz, Füllung in *RGBFill*
BorderWidth	Breite des Rahmens in Pixeln



Grafikelement LABEL

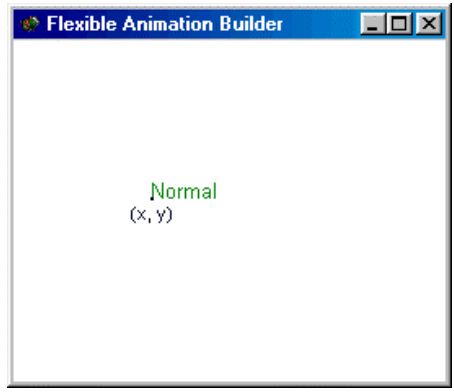
Elementtyp *MESSAGE*

Der Elementtyp *MESSAGE* dient zur Ausgabe einzeiliger dynamischer Textmeldungen (optional mit Rahmen) in Abhängigkeit von einem Blockeingang. Text und Farbe der Meldung können für drei verschiedene Bereiche der Eingangsgröße spezifiziert werden. Der Elementtyp besitzt die nachfolgend aufgelisteten Eigenschaften.

Eigenschaft	Bedeutung
RGBStroke	Textfarbe
x	x-Koordinate des linken unteren Eckpunkts des Textes

y	y-Koordinate des linken unteren Eckpunkts des Textes
Input	Auszugebende Ein- bzw. Ausgangsgröße. Nummern von Blockeingängen werden direkt angegeben, bei Blockausgängen muß der Wert 100 addiert werden. Beispiel: Ist *Input* = 4, wird der vierte Blockeingang ausgegeben, ist *Input* = 102, so wird der zweite Blockausgang angezeigt.
Size	Textgröße
Font	Schriftart
LowText	Auszugebender Text, falls die durch *Input* spezifizierte Größe unterhalb von *LowThreshold* liegt.
MidText	Auszugebender Text, falls die durch *Input* spezifizierte Größe unterhalb zwischen *LowThreshold* und *HighThreshold* liegt.
HighText	Auszugebender Text, falls die durch *Input* spezifizierte Größe oberhalb von *HighThreshold* liegt.
LowThreshold	Unterer Grenzwert
HighThreshold	Oberer Grenzwert
LowColor	Textfarbe, falls die durch *Input* spezifizierte Größe unterhalb von *LowThreshold* liegt.
MidColor	Textfarbe, falls die durch *Input* spezifizierte Größe unterhalb zwischen *LowThreshold* und *HighThreshold* liegt.
HighColor	Textfarbe, falls die durch *Input* spezifizierte Größe oberhalb von *HighThreshold* liegt.
BorderType	Typ des Rahmens um den ausgegebenen Text. Die Ausdehnung des Rahmens orientiert sich jeweils an der längsten der drei Textmeldungen. Folgende Typen sind verfügbar: 0 kein Rahmen 1 Rahmen in Textfarbe, keine Füllung 2 Rahmen in Textfarbe, Füllung in *RGBFill* 3 Schattierter Rahmen in schwarz, keine Füllung 4 Schattierter Rahmen in schwarz, Füllung in *RGBFill*

BorderWidth	Breite des Rahmens in Pixeln
Centered (0/1)	Gibt an, ob der Text zentriert innerhalb des Rahmens ausgegeben werden soll.

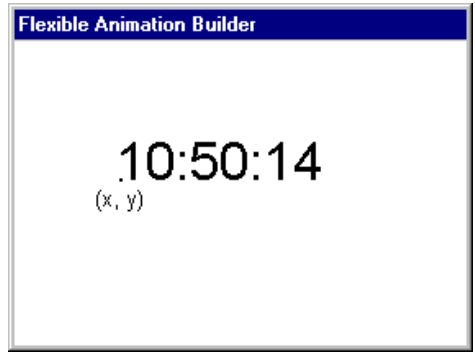


Grafikelement MESSAGE

Elementtyp *TIME*

Der Elementtyp *TIME* stellt ein frei formatierbares Ausgabefeld für die aktuelle Systemzeit des Rechners dar und besitzt die nachfolgend aufgelisteten Eigenschaften.

Eigenschaft	Bedeutung
RGBStroke	Textfarbe
x	x-Koordinate des linken unteren Eckpunkts des Textes
y	y-Koordinate des linken unteren Eckpunkts des Textes
Size	Textgröße
Font	Schriftart
TimeFormat	Ausgabeformat für die Systemzeit



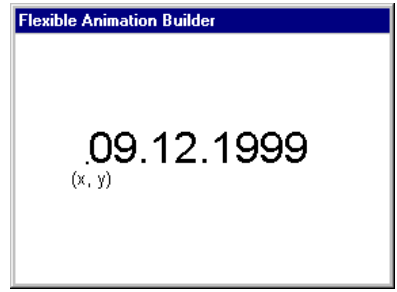
Grafikelement TIME

Die Beispieldatei *TIMEANDDATE.BSY* demonstriert den Einsatz dieses Elementtyps.

Elementtyp *DATE*

Der Elementtyp *DATE* stellt ein frei formatierbares Ausgabefeld für das aktuelle Systemdatum des Rechners dar und besitzt die nachfolgend aufgelisteten Eigenschaften.

Eigenschaft	Bedeutung
RGBStroke	Textfarbe
x	x-Koordinate des linken unteren Eckpunkts des Textes
y	y-Koordinate des linken unteren Eckpunkts des Textes
Size	Textgröße
Font	Schriftart
DateFormat	Ausgabeformat für das Systemdatum



Grafikelement DATE

Die Beispieldatei *TIMEANDDATE.BSY* demonstriert den Einsatz dieses Elementtyps.

Elementtyp *BITMAP*

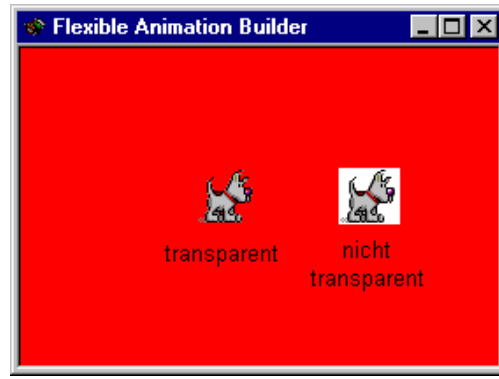
Über den Elementtyp *BITMAP* lassen sich beliebige Bitmaps in das FAB-Visualisierungsfenster einfügen, die in einer externen BMP-Datei vorliegen müssen. Der Elementtyp weist nachfolgend dargestellte Eigenschaften auf.

Eigenschaft	Bedeutung
Transp	Flag, das angibt, ob das Bitmap transparent (1) oder nicht transparent (0) dargestellt werden soll (siehe untenstehende Bildschirmgrafik). Bei transparenter Darstellung wird die Farbe, die das Pixel in der linken unteren Ecke des Bitmaps besitzt, als transparente Farbe gewählt.
x	x-Koordinate des linken unteren Eckpunktes des umgebenden Rechtecks (vgl. *ELLIPSE*)
y	y-Koordinate des linken unteren Eckpunktes des umgebenden Rechtecks (vgl. *ELLIPSE*)
w	Breite des umgebenden Rechtecks in Pixeln (vgl. *ELLIPSE*). Entspricht dieser Wert nicht der Originalbreite des Bitmaps, wird es entsprechend gestaucht bzw. gestreckt. Nach Einlesen eines neuen Bitmaps wird der Wert automatisch an die Originalbreite des Bitmaps angepaßt.
h	Höhe des umgebenden Rechtecks in Pixeln (vgl. *ELLIPSE*). Entspricht dieser Wert nicht der Originalhöhe des Bitmaps,

wird es entsprechend gestaucht bzw. gestreckt. Nach Einlesen eines neuen Bitmaps wird der Wert automatisch an die Originalhöhe des Bitmaps angepaßt.

File or Id Name der BMP-Datei incl. Pfad. Wird die Datei nicht im angegebenen Pfad gefunden, wird sie in demjenigen Verzeichnis gesucht, in dem die Datei FAB.DLL liegt. Wird sie dort ebenfalls nicht gefunden, erscheint eine entsprechende Warnung und das Bitmap kann nicht dargestellt werden.

Befindet sich das Bitmap in einer der beiden Bitmap-Libraries (siehe Kapitel *Nutzung von Bitmaps*), so kann es auch über seinen *Identifier* spezifiziert werden. Achtung: Breite *w* und Höhe *h* des Bitmaps (s. o.) müssen in diesem Fall ggf. von Hand angepasst werden!



Zwei identische BITMAP-*Elemente, einmal transparent und einmal nicht transparent dargestellt*

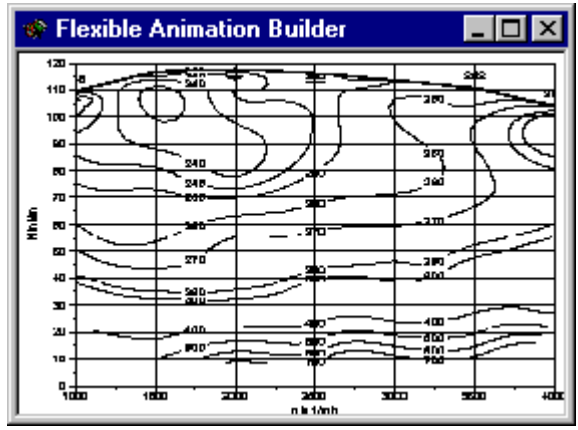
Elementtyp *WMF*

Über den Elementtyp *WMF* lassen sich beliebige Windows-Metafiles in das FAB-Visualisierungsfenster einfügen, die in einer externen WMF-Datei vorliegen müssen. Der Elementtyp weist nachfolgend dargestellte Eigenschaften auf.

Eigenschaft	Bedeutung
x	x-Koordinate des linken unteren Eckpunktes des umgebenden Rechtecks (vgl. *ELLIPSE*)

y	y-Koordinate des linken unteren Eckpunktes des umgebenden Rechtecks (vgl. *ELLIPSE*)
w	Breite des umgebenden Rechtecks in Pixeln (vgl. *ELLIPSE*).
h	Höhe des umgebenden Rechtecks in Pixeln (vgl. *ELLIPSE*).
File	Name der WMF-Datei incl. Pfad. Wird die Datei nicht im angegebenen Pfad gefunden, wird sie in demjenigen Verzeichnis gesucht, in dem die Datei FAB.DLL liegt. Wird sie dort ebenfalls nicht gefunden, erscheint eine entsprechende Warnung und das Metafile kann nicht dargestellt werden.

Da WMF-Dateien in der Regel als formatfüllende Hintergrundgrafiken eingesetzt werden, werden die Werte für *x*, *y*, *w* und *h* nach dem Einlesen einer neuen WMF-Datei automatisch so gesetzt, dass die Grafik das gesamte Visualisierungsfenster ausfüllt.



Visualisierungsfenster mit Anzeige einer WMF-Datei

Elementtyp *STATEBMP*

Der Elementtyp *STATEBMP* stellt ein dynamisches Bitmap (Bitmap-Statusanzeige) zur Verfügung, das sein Aussehen in Abhängigkeit von einem Blockein- oder -ausgangssignal zwischen zwei Zuständen wechselt, die über die beiden Bitmap-Dateien *OnBMPFile* und *OffBMPFile* festgelegt sind. Das Element weist nachfolgend dargestellte Eigenschaften auf.

Eigenschaft	Bedeutung
Input	Maßgebliche Ein- bzw. Ausgangsgröße. Nummern von Blockeingängen werden direkt angegeben, bei Blockausgängen muß der Wert 100 addiert werden. Beispiel: Ist *Input* = 4, wird der vierte Blockeingang ausgegeben, ist *Input* = 102, so wird der zweite Blockausgang angezeigt.
x	x-Koordinate des linken unteren Eckpunktes des umgebenden Rechtecks
y	y-Koordinate des linken unteren Eckpunktes des umgebenden Rechtecks
w	Breite des umgebenden Rechtecks in Pixeln.
h	Höhe des umgebenden Rechtecks in Pixeln.
OnValue	Umschaltwert. Wird dieser Wert von dem über *Input* spezifizierten Signal überschritten, wird das Bitmap *OnBMPFile* angeizeigt, sonst das Bitmap *OffBMPFile*.
OnBMPFile or Id	Bitmap-Datei für den LOW-Zustand . Befindet sich das Bitmap in einer der beiden Bitmap-Libraries (siehe Kapitel *Nutzung von Bitmaps*), so kann es auch über seinen *Identifier* spezifiziert werden. Achtung: Breite *w* und Höhe *h* des Bitmaps (s. o.) müssen in diesem Fall ggf. von Hand angepasst werden!
OffBMPFile or Id	Bitmap-Datei für den HIGH-Zustand . Befindet sich das Bitmap in einer der beiden Bitmap-Libraries (siehe Kapitel *Nutzung von Bitmaps*), so kann es auch über seinen *Identifier* spezifiziert werden. Achtung: Breite *w* und Höhe *h* des Bitmaps (s. o.) müssen in diesem Fall ggf. von Hand angepasst werden!
Transparent (0/1)	Legt fest, ob die Bitmaps transparent dargestellt werden sollen.

Elementtyp *BMPSEQ*

Der Elementtyp *BMPSEQ* stellt eine Bitmap-Sequenz dar, die eingangsgesteuert oder automatisch (zyklisch zeitgesteuert mit eingangsgesteuerter

Start/Stopp-Funktion) ablaufen und aus bis zu zehn Einzelbitmaps bestehen kann. Mit Hilfe dieses Elementtyps lassen sich auf einfache Weise Animationen erstellen. Das Element weist nachfolgend dargestellte Eigenschaften auf.

Eigenschaft	**Bedeutung**
Input	Maßgebliche Ein- bzw. Ausgangsgröße. Nummern von Blockeingängen werden direkt angegeben, bei Blockausgängen muß der Wert 100 addiert werden. Beispiel: Ist *Input* = 4, ist der vierte Blockeingang maßgebend, ist *Input* = 102, so ist der zweite Blockausgang maßgebend. Ist *Input* ≤ *0*, so läuft die Sequenz automatisch, d. h. zyklisch zeitgesteuert ab. Für *Input = 0* (Voreinstellung) ist die Animation ständig aktiv; ist *Input* < *0*, so dient der entsprechende *positive* Blockeingang als Start/Stopp-Eingang mit TTL-Pegel. Ist also z. B. *Input = -1*, so ist die Animation aktiv, wenn Blockeingang 1 einen Wert von 2.5 überschreitet.
x	x-Koordinate des linken unteren Eckpunktes des umgebenden Rechtecks
y	y-Koordinate des linken unteren Eckpunktes des umgebenden Rechtecks
w	Breite des umgebenden Rechtecks in Pixeln.
h	Höhe des umgebenden Rechtecks in Pixeln.
BMPCount	Anzahl der in der Sequenz enthaltenen Bitmaps
BMPFile[x] or Id	Bitmap-Datei für das *x*-te Bitmap der Sequenz . Befindet sich das Bitmap in einer der beiden Bitmap-Libraries (siehe Kapitel *Nutzung von Bitmaps*), so kann es auch über seinen *Identifier* spezifiziert werden. Achtung: Breite *w* und Höhe *h* des Bitmaps (s. o.) müssen in diesem Fall ggf. von Hand angepasst werden!
Transparent (0/1)	Legt fest, ob die Bitmaps transparent dargestellt werden sollen.
RangeMin	Untere Bereichsgrenze der durch *Input* spezifizierten Eingangsgröße für den Fall *Input* > *0* (eingangsgesteu-

erte Sequenz).

RangeMax — Obere Bereichsgrenze der durch *Input* spezifizierten Eingangsgröße für den Fall *Input > 0* (eingangsgesteuerte Sequenz). Der gesamte Bereich von *RangeMax - RangeMin* wird linear auf die insgesamt *BMPCount* Bitmaps der Sequenz aufgeteilt. Ist z. B. *BMPCount = 2* (Sequenz besteht nur aus zwei Bitmaps), so wird bei einer Eingangsgröße unterhalb von *(RangeMax-RangeMin)/2* das erste Bitmap angezeigt, bei einem darüber liegenden Eingangsgwert das zweite Bitmap.

Delay — Legt die Verzögerung zwischen zwei Bitmapwechseln im Fall *Input = -1* (zeitgesteuerte Sequenz) fest, d. h. die Anzahl an Simulationsschritten, die jeweils verstreichen muss, bis auf das nächste Bitmap der Sequenz gewechselt wird. Nach dem letzten Bitmap der Sequenz beginnt die Sequenz wieder von vorn (zyklischer Ablauf).

Elementtyp *AVI*

Über den Elementtyp *AVI* lassen sich *Video für Windows*-Grafiken (AVI-Dateien) darstellen. Dieser Dateityp enthält eine Folge von Einzelgrafiken (*Frames* genannt) gleicher Größe, die zu einer bewegten Grafik verknüpft sind. AVI-Dateien können prinzipiell auch Ton enthalten; dieser wird allerdings vom FAB-Modul nicht unterstützt. Die Grafiken können in zwei Modi dargestellt werden:

- Als zyklisch ablaufende Grafik mit externer Start/Stopp-Steuerung (Betriebsart *Zyklus*)
- Als Folge von Einzelgrafiken, wobei die bei einem Simulationsschritt jeweils anzuzeigende Grafik über einen Moduleingang gesteuert wird (Betriebsart *Einzelgrafik*)

Die Betriebsart wird über die Elementeigenschaft *FrIn* (Frame-Input) gesteuert. Weist *FrIn* die Nummer eines gültigen Modulein- oder -ausgangs auf (z. B. *FrIn* = 1 für Eingang 1 oder *FrIn* = 103 für Ausgang 3; bei Ausgängen ist zur Ausgangsnummer jeweils der Wert 100 zu addieren), so gibt bei jedem Simulationsschritt der Wert an diesem Eingang die auszugebende Einzelgrafik der AVI-Datei an (ist also *FrIn* = 1 und hat das Signal an Moduleingang 1 den

Wert 5, so wird die fünfte Grafik der AVI-Datei ausgegeben). Ist *FrIn* = -1 (Voreinstellung), so ist die Betriebsart *Zyklus* aktiv.

Der Elementtyp weist nachfolgend dargestellte Eigenschaften auf.

Eigenschaft	Bedeutung
x	x-Koordinate des linken unteren Eckpunktes des umgebenden Rechtecks
y	y-Koordinate des linken unteren Eckpunktes des umgebenden Rechtecks
File	Name der AVI-Datei incl. Pfad. Wird die Datei nicht im angegebenen Pfad gefunden, wird sie in demjenigen Verzeichnis gesucht, in dem die Datei FAB.DLL liegt. Wird sie dort ebenfalls nicht gefunden, erscheint eine entsprechende Warnung und die Datei kann nicht dargestellt werden.
Act	Diese Eigenschaft startet bzw. stoppt den zyklischen Ablauf der Grafik in der Betriebsart *Zyklus* (*FrIn* = -1). Ist der Wert von *Act* größer null, läuft die Bewegtgrafik, ist er kleiner oder gleich null, so wird sie gestoppt. Für *Act* kann ein konstanter Zahlenwert angegeben werden (dann läuft bzw. steht die Grafik immer) oder aber ein gültiger Moduleingang (z. B. *I1*) oder Modulausgang (z. B. *O3*); in diesem Fall wird die Grafik über das Signal an diesem Modulein- bzw. -ausgang gesteuert.
Rep	Gibt die Anzahl der Zyklus-Wiederholungen in der Betriebsart *Zyklus* an, nachdem die Grafik einmal gestartet wurde. Ist *Rep* = 0, läuft die Grafik solange weiter, bis sie gestoppt wird.
FrIn	Gibt die Nummer des Steuerein- bzw. -ausgangs in der Betriebsart *Einzelgrafik* an (s. o.). Diese Betriebsart wird automatisch aktiviert, sobald *FrIn* eine gültige Ein- bzw. Ausgangsnummer enthält.

Nachfolgender Screenshot zeigt das Visualisierungsfenster beim Abspielen der mitgelieferten Datei CLOCK.AVI.

Visualisierungsfenster mit Anzeige einer AVI-Datei

Elementtyp *SOUND*

Der Elementtyp *SOUND* bietet eine Soundunterstützung auf der Basis von WAV-Dateien oder einfachen akustischen Signalen. Die Steuerung erfolgt über die Eigenschaften *CIn*, *CInMax* und *CInMin* (siehe auch Abschnitt *Grundlegende Elementeigenschaften*). Ist *CIn* = -1, ist der Sound ständig aktiv. Weist *CIn* hingegen die Nummer eines Blockeingangs auf, so ist der Sound dann aktiv, wenn das Signal an diesem Blockeingang zwischen *CInMin* und *CInMax* liegt.

Darüber hinaus besitzt dieser Elementtyp nachfolgend dargestellte Eigenschaften.

Eigenschaft	Bedeutung
x	Während der Konfigurierung wird ein *SOUND*-Element im Visualisierungsfenster durch ein kleines Bitmap (symbolisierter Lautsprecher) dargestellt. Diese Eigenschaft legt die x-Koordinate des linken unteren Eckpunktes des umgebenden Rechtecks fest.
y	y-Koordinate des linken unteren Eckpunktes des umgebenden Rechtecks
File	Name der WAV-Datei incl. Pfad. Wird die Datei nicht im angegebenen Pfad gefunden, wird sie in demjenigen Verzeichnis gesucht, in dem die Datei FAB.DLL liegt. Wird kein Name spezifiziert, wird bei aktiviertem Sound ein Standard-Ton auf dem Lautsprecher ausgegeben. Diese Betriebsart ist daher auch ohne Soundkarte möglich.

Nachfolgender Screenshot zeigt das Visualisierungsfenster mit einem *SOUND*-Element im Konfigurierungsmodus.

Visualisierungsfenster mit Anzeige eines SOUND-*Elements im Entwurfsmodus*

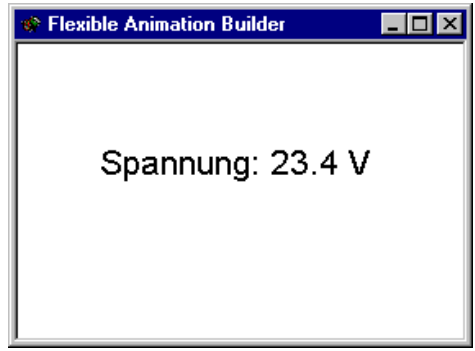
Die Beispieldatei SOUNDDEMO.BSY demonstriert die Anwendung des *SOUND*-Elementtyps.

Alle eingefügten SOUND-Elemente können während der Konfigurierung über die Schaltfläche der Toolbar des Konfigurierungsdialogs temporär deaktiviert werden; damit wird verhindert, dass sie bei jedem Auffrischen des Visualisierungsfensters abgespielt werden.

Elementtyp *NUMBER*

Der Elementtyp *NUMBER* dient zur Ausgabe einzeiliger Texte (optional mit Rahmen), die einen numerischen Wert enthalten und besitzt die nachfolgend aufgelisteten Eigenschaften.

Eigenschaft	Bedeutung
RGBStroke	Textfarbe
x	x-Koordinate des linken unteren Eckpunkts des Textes (vgl. *LABEL*)
y	y-Koordinate des linken unteren Eckpunkts des Textes (vgl. *LABEL*)
Size	Textgröße
Font	Schriftart
Format	Auszugebender Text und Formatierung (s. u.)
Input	Auszugebende Ein- bzw. Ausgangsgröße. Nummern von Blockeingängen werden direkt angegeben, bei Blockausgängen muß der Wert 100 addiert werden. Beispiel: Ist *Input* = 4, wird der vierte Blockeingang ausgegeben, ist *Input* = 102, so wird der zweite Blockausgang angezeigt.
BorderType	Typ des Rahmens um den ausgegebenen Text. Folgende Typen sind verfügbar: 0 kein Rahmen 1 Rahmen in Textfarbe, keine Füllung 2 Rahmen in Textfarbe, Füllung in *RGBFill* 3 Schattierter Rahmen in schwarz, keine Füllung 4 Schattierter Rahmen in schwarz, Füllung in *RGBFill*
BorderWidth	Breite des Rahmens in Pixeln



Kombinierte Text-/Zahlenausgabe mit dem NUMBER-*Grafikelement*

Über die *Format*-Eigenschaft kann die auszugebende Größe in umfassender Weise formatiert werden. Dazu muß das Format-Feld in jedem Falle den sog. *Formatbezeichner* enthalten, der das Zahlenformat festlegt und immer mit einem Prozentzeichen beginnt. Vor und hinter diesem Formatierungsstring kann dann bei Bedarf zusätzlich statischer Text angegeben werden. So wurde die obenstehende Bildschirmgrafik erzeugt durch die Formatangabe

Spannung: %4.1f V

Dabei spezifiziert der eigentliche Formatbezeichner

%4.1f

eine Fließkommaausgabe des (ggf. skalierten) Ein- bzw. Ausgangswertes mit einer Breite von 4 Stellen und einer Nachkommastelle.

Allgemein muß der Formatbezeichner in der folgenden Form angegeben werden:

"%" [index ":"] ["-"] [width] ["." prec] type

Jeder Formatbezeichner beginnt mit dem Zeichen %. Auf das Prozentzeichen folgt eine der nachstehenden Angaben (in der aufgeführten Reihenfolge):

- Ein optionaler Argumentindex-Bezeichner: [index ":"]
- Eine optionale Angabe für die linksbündige Ausrichtung: ["-"]
- Eine optionale Breitenangabe: [width]
- Eine optionale Genauigkeitsangabe: ["." prec]

- Das Zeichen für den Konvertierungstyp: type

In der folgenden Tabelle sind die verschiedenen Werte aufgeführt:

d *Dezimal*. Das Argument muß ein Integerwert sein. Der Wert wird in einen String umgewandelt, der aus Dezimalzahlen besteht. Wenn der Format-String einen Bezeichner für die Genauigkeit enthält, muß der resultierende String mindestens die angegebene Anzahl von Stellen aufweisen. Enthält er weniger Stellen, wird der String auf der linken Seite mit Nullen aufgefüllt.

e *Wissenschaftliche Notation*. Das Argument muß ein Gleitkommawert sein. Der Wert wird in einen String mit der folgenden Form umgewandelt: "-d,ddd...E+ddd". Wenn es sich um eine negative Zahl handelt, beginnt der String mit einem Minuszeichen. Vor dem Dezimaltrennzeichen steht immer eine Ziffer. Die Gesamtzahl der Stellen im Ergebnis-String (einschließlich der Ziffer vor dem Dezimalkomma) wird durch den Genauigkeitsbezeichner im Format-String festgelegt. Ist dieser nicht vorhanden, wird eine vorgegebene Genauigkeit von 15 Stellen angenommen. Auf den Exponenten "E" im String folgen immer ein Plus- oder Minuszeichen und mindestens drei Stellen.

f *Fest*. Das Argument muß ein Gleitkommawert sein. Der Wert wird in einen String mit der folgenden Form umgewandelt: "-ddd,ddd...". Wenn es sich um eine negative Zahl handelt, beginnt der String mit einem Minuszeichen. Die Anzahl der Stellen nach dem Dezimalkomma wird durch den Genauigkeitsbezeichner im Format-String festgelegt. Ist dieser nicht vorhanden, wird eine vorgegebene Genauigkeit von zwei Dezimalstellen verwendet.

g *Allgemein*. Das Argument muß ein Gleitkommawert sein. Der Wert wird unter Verwendung des Formats *Fest* oder *wissenschaftliche Notation* in den kürzestmöglichen Dezimal-String umgewandelt. Die Anzahl der signifikanten Stellen im resultierenden String wird durch den Genauigkeitsbezeichner im Format-String festgelegt. Ist dieser nicht vorhanden, wird eine vorgegebene Genauigkeit von 15 Stellen angenommen.

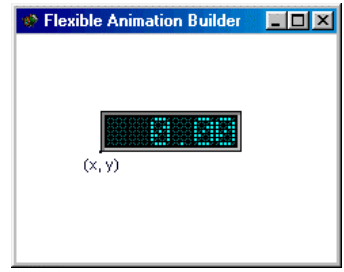
n *Zahl*. Das Argument muß ein Gleitkommawert sein. Der Wert wird in einen String mit der folgenden Form umgewandelt: "-d.ddd.ddd,ddd...". Das Format "n" entspricht dem Format "f", allerdings enthält der resultierende String Tausendertrennzeichen.

Konvertierungszeichen können beliebig in Klein- oder Großschreibung angegeben werden.

Elementtyp *LCD*

Der Elementtyp *LCD* stellt ein numerisches Ausgabefeld im "LCD-Look" dar und besitzt die nachfolgend aufgelisteten Eigenschaften.

Eigenschaft	Bedeutung
RGBStroke	Textfarbe
x	x-Koordinate des linken unteren Eckpunkts des Textes (vgl. *LABEL*)
y	y-Koordinate des linken unteren Eckpunkts des Textes (vgl. *LABEL*)
FormatWidth	Gesamt-Stellenzahl (incl. Dezimalpunkt)
FormatPrecision	Anzahl der Nachkommastellen
Input	Auszugebende Ein- bzw. Ausgangsgröße. Nummern von Blockeingängen werden direkt angegeben, bei Blockausgängen muß der Wert 100 addiert werden. Beispiel: Ist *Input* = 4, wird der vierte Blockeingang ausgegeben, ist *Input* = 102, so wird der zweite Blockausgang angezeigt.
BorderType	Typ des Rahmens um den ausgegebenen Text. Folgende Typen sind verfügbar: 0 kein Rahmen 1 Einfachrahmen in schwarz 2 Schattierter Rahmen
BorderWidth	Breite des Rahmens in Pixeln



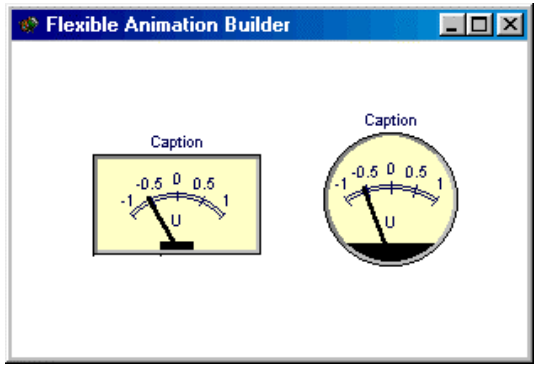
Beispiel für LCD-*Grafikelement*

Elementtyp *ANADISP*

Der Elementtyp *ANADISP* stellt ein frei skalierbares Analoginstrument dar und besitzt die nachfolgend aufgelisteten Eigenschaften.

Eigenschaft	Bedeutung
RGBStroke	Text- und Skalenfarbe
RGBFill	Hintergrundfarbe des Instruments
LType	Zeigerbreite in Pixeln
x	x-Koordinate des linken unteren Eckpunkts der Anzeige
y	y-Koordinate des linken unteren Eckpunkts der Anzeige
w	Breite der Anzeige in Pixeln
h	Höhe der Anzeige in Pixeln
Input	Auszugebende Ein- bzw. Ausgangsgröße. Nummern von Blockeingängen werden direkt angegeben, bei Blockausgängen muß der Wert 100 addiert werden. Beispiel: Ist *Input* = 4, wird der vierte Blockeingang ausgegeben, ist *Input* = 102, so wird der zweite Blockausgang angezeigt.
MinValue	Unterer Skalenwert
MaxValue	Oberer Skalenwert
RedValue	Liegt *RedValue* zwischen *MinValue* und *MaxValue*, so wird derjenige Skalenteil, der zwischen *RedValue* und *MaxValue* liegt, in rot gezeichnet.
Caption	Überschrift der Anzeige
Unit	Skaleneinheit
BorderType	Typ des Rahmens um das Instrument. Folgende Typen sind verfügbar: 0 kein Rahmen 1 Rechteck 2 Schattiertes Rechteck 3 Kreis bzw. Ellipse 4 Schattierter Kreis bzw. Ellipse

BorderWidth	Breite des Rahmens in Pixeln
ScaleHeight [%]	Vertikale Position der Skala innerhalb des Instruments in Prozent der Gesamthöhe *w*
ShowScale (0/1)	Gibt an, ob die Skala gezeichnet werden soll.



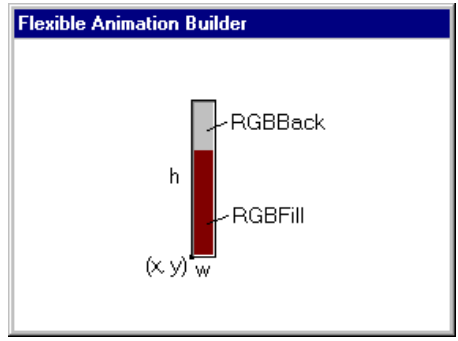
Beispiele für ANADISP-*Grafikelemente*

Elementtyp *VBAR*

Der Elementtyp *VBAR* stellt eine vertikale Balkenanzeige zur Verfügung, die über einen Blockein- bzw. -ausgang angesteuert wird. Er weist nachfolgend dargestellte Eigenschaften auf.

Eigenschaft	Bedeutung
RGBBack	Hintergrundfarbe
RGBFill	Farbe des Balkens
Input	Auszugebende Ein- bzw. Ausgangsgröße. Nummern von Blockeingängen werden direkt angegeben, bei Blockausgängen muß der Wert 100 addiert werden. Beispiel: Ist *Input* = 4, wird der vierte Blockeingang ausgegeben, ist *Input* = 102, so wird der zweite Blockausgang angezeigt.
x	x-Koordinate des linken unteren Eckpunktes des umgebenden Rechtecks

y	y-Koordinate des linken unteren Eckpunktes des umgebenden Rechtecks
w	Breite des umgebenden Rechtecks in Pixeln.
h	Höhe des umgebenden Rechtecks in Pixeln.
Min	Kleinster dargestellter Wert von *Input* (Nullausschlag der Anzeige).
Max	Größter dargestellter Wert von *Input* (Vollausschlag der Anzeige).



Grafikelement VBAR

Die Beispieldatei BARDEMO.BSY demonstriert die Anwendung des *VBAR*-Elementtyps.

Elementtyp *HBAR*

Der Elementtyp *HBAR* stellt eine horizontale Balkenanzeige zur Verfügung, die über einen Blockein- bzw. -ausgang angesteuert wird. Er weist nachfolgend dargestellte Eigenschaften auf.

Eigenschaft	Bedeutung
RGBBack	Hintergrundfarbe
RGBFill	Farbe des Balkens
Input	Auszugebende Ein- bzw. Ausgangsgröße. Nummern von Blockeingängen werden direkt angegeben, bei Blockausgän-

gen muß der Wert 100 addiert werden. Beispiel: Ist *Input* = 4, wird der vierte Blockeingang ausgegeben, ist *Input* = 102, so wird der zweite Blockausgang angezeigt.

x x-Koordinate des linken unteren Eckpunktes des umgebenden Rechtecks

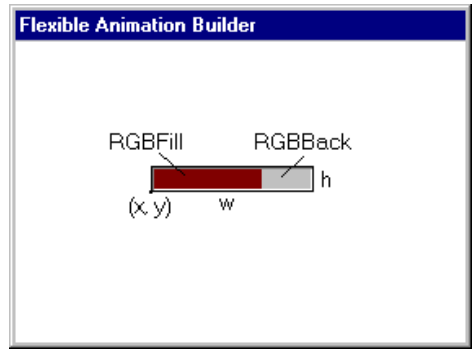
y y-Koordinate des linken unteren Eckpunktes des umgebenden Rechtecks

w Breite des umgebenden Rechtecks in Pixeln.

h Höhe des umgebenden Rechtecks in Pixeln.

Min Kleinster dargestellter Wert von *Input* (Nullausschlag der Anzeige).

Max Größter dargestellter Wert von *Input* (Vollausschlag der Anzeige).



Grafikelement HBAR

Die Beispieldatei BARDEMO.BSY demonstriert die Anwendung des *HBAR*-Elementtyps.

Elementtyp *VBARGRAPH*

Der Elementtyp *VBARGRAPH* stellt eine vertikale Balkenanzeige mit erweiterten Möglichkeiten (z. B. bezüglich Beschriftung und Farbgebung) zur Verfügung, die über einen Blockein- bzw. -ausgang angesteuert wird. Er weist nachfolgend dargestellte Eigenschaften auf.

Eigenschaft	Bedeutung
RGBBack	Hintergrundfarbe
RGBFill	Standard-Balkenfarbe (s. u.)
Input	Auszugebende Ein- bzw. Ausgangsgröße. Nummern von Blockeingängen werden direkt angegeben, bei Blockausgängen muß der Wert 100 addiert werden. Beispiel: Ist *Input* = 4, wird der vierte Blockeingang ausgegeben, ist *Input* = 102, so wird der zweite Blockausgang angezeigt.
x	x-Koordinate des linken unteren Eckpunktes des umgebenden Rechtecks
y	y-Koordinate des linken unteren Eckpunktes des umgebenden Rechtecks
w	Breite des umgebenden Rechtecks in Pixeln.
h	Höhe des umgebenden Rechtecks in Pixeln.
Min	Kleinster dargestellter Wert von *Input* (Nullausschlag der Anzeige).
Max	Größter dargestellter Wert von *Input* (Vollausschlag der Anzeige).
SymmMode (0-2)	Legt die Farbgebung des Balkens für den Fall fest, dass *Min < 0* und *Max > 0* gewählt wurde (Nullpunkt der Anzeige liegt innerhalb des Skalenbereichs). Liegt der Nullpunkt außerhalb des Anzeigebereichs, ist diese Eigenschaft ohne Bedeutung (siehe auch untenstehende Bildschirmgrafik). Für *SymmMode = 0* wird der Balken vom unteren Skalenendwert (*Min*) bis zum aktuellen Eingangswert gezeichnet. Für *SymmMode = 1* erfolgt der Ausschlag bei positiven Eingangsgwerten vom Nullpunkt nach oben, bei negativen Eingangswerten vom Nullpunkt nach unten; je nach Wahl von *Threshold2* und *Threshold3* (s. u.) wird der Balken dabei ggf. mehrfarbig dargestellt. Für *SymmMode = 3* erfolgt der Ausschlag nach oben bzw. unten unabhängig von *Threshold2* und *Threshold3* immer einfarbig, wobei für den Ausschlag

nach oben *RGBFill* und für den Ausschlag nach unten *RGBFill2* (s. u.) benutzt wird.

RGBFill2

Die Bedeutung dieser Farbeigenschaft hängt vom Wert der Eigenschaft *SymmMode* ab:

SymmMode = 0:

Farbe des Balkens für den Bereich des Eingangswertes zwischen *Threshold2* und *Threshold3*. Liegt *Threshold2* außerhalb des Anzeigebereichs, ist der Wert ohne Bedeutung.

SymmMode = 1:

Farbe des Balkens für den Bereich des Eingangswertes zwischen *Threshold2* und *Threshold3* bzw. zwischen -*Threshold2* und -*Threshold3*. Liegt *Threshold2* außerhalb des Anzeigebereichs, ist der Wert ohne Bedeutung.

SymmMode = 2:

Farbe des Balkens für negative Eingangswerte (Ausschlag nach unten).

RGBFill3

Die Bedeutung dieser Farbeigenschaft hängt vom Wert der Eigenschaft *SymmMode* ab:

SymmMode = 0:

Farbe des Balkens für den Bereich des Eingangswertes oberhalb von *Threshold3*. Liegt *Threshold3* außerhalb des Anzeigebereichs, ist der Wert ohne Bedeutung.

SymmMode = 1:

Farbe des Balkens für den Bereich des Eingangswertes oberhalb von *Threshold3* bzw. unterhalb von -*Threshold3*. Liegt *Threshold3* außerhalb des Anzeigebereichs, ist der Wert ohne Bedeutung.

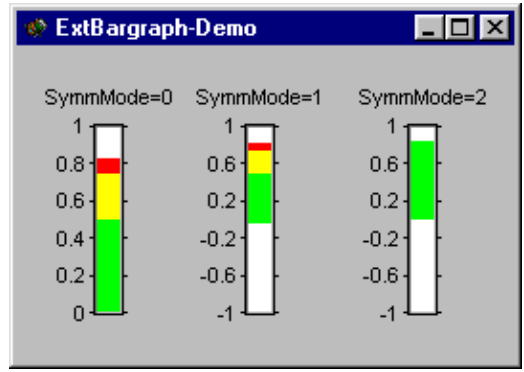
SymmMode = 2:

Wert ist ohne Bedeutung

Threshold2

Eingangswert für Farbwechsel zwischen *RGBFill* und *RGBFill2*

Threshold3	Eingangswert für Farbwechsel zwischen *RGBFill2* und *RGBFill3*
Ticks	Anzahl der Skalenmarkierungen.
FontSize	Schriftgröße für die Skalenbeschriftung bzw. Überschrift in Pixeln.
ScaleAlign (0=none)	Ausrichtung der Skalenbeschriftung (links bzw. rechts). Für *ScaleAlign = 0* wird keine Skalenbeschriftung (wohl aber die Skala selbst) ausgegeben.
Caption	Überschrift des Bargraphen. In die Überschrift kann auch durch Angabe einer entsprechenden Formatanweisung (siehe Elementtyp *NUMBER*) der aktuelle Eingangswert aufgenommen werden. Beispiel: Wird für *Caption* die Zeichenfolge *'Temp = %5.2f Grad'* angegeben, so lautet bei einem aktuellen Eingangswert von 25.5 Grad die Überschrift *'Temp = 25.50 Grad'*.



VBARGRAPH-*Elemente für verschiedene Werte der Eigenschaft* SymmMode *und einen aktuellen Eingangswert von 0.8*

Die Beispieldatei EXTBARGRAPH.BSY demonstriert die Anwendung des Elementtyps.

Elementtyp *HBARGRAPH*

Der Elementtyp *HBARGRAPH* stellt eine horizontale Balkenanzeige mit erweiterten Möglichkeiten (z. B. bezüglich Beschriftung und Farbgebung) zur Verfügung, die über einen Blockein- bzw. -ausgang angesteuert wird. Er weist nachfolgend dargestellte Eigenschaften auf (siehe auch Elementtyp *VBARGRAPH*).

Eigenschaft	Bedeutung
RGBBack	Hintergrundfarbe
RGBFill	Standard-Balkenfarbe (s. u.)
Input	Auszugebende Ein- bzw. Ausgangsgröße. Nummern von Blockeingängen werden direkt angegeben, bei Blockausgängen muß der Wert 100 addiert werden. Beispiel: Ist *Input* = 4, wird der vierte Blockeingang ausgegeben, ist *Input* = 102, so wird der zweite Blockausgang angezeigt.
x	x-Koordinate des linken unteren Eckpunktes des umgebenden Rechtecks
y	y-Koordinate des linken unteren Eckpunktes des umgebenden Rechtecks
w	Breite des umgebenden Rechtecks in Pixeln.
h	Höhe des umgebenden Rechtecks in Pixeln.
Min	Kleinster dargestellter Wert von *Input* (Nullausschlag der Anzeige).
Max	Größter dargestellter Wert von *Input* (Vollausschlag der Anzeige).
SymmMode (0-2)	Legt die Farbgebung des Balkens für den Fall fest, dass *Min < 0* und *Max > 0* gewählt wurde (Nullpunkt der Anzeige liegt innerhalb des Skalenbereichs). Liegt der Nullpunkt außerhalb des Anzeigebereichs, ist diese Eigenschaft ohne Bedeutung . Für *SymmMode = 0* wird der Balken vom unteren Skalenendwert (*Min*) bis zum aktuellen Eingangswert gezeichnet. Für *SymmMode = 1* erfolgt der Ausschlag

bei positiven Eingangsgwerten vom Nullpunkt nach rechts, bei negativen Eingangswerten vom Nullpunkt nach links; je nach Wahl von *Threshold2* und *Threshold3* (s. u.) wird der Balken dabei ggf. mehrfarbig dargestellt. Für *SymmMode = 3* erfolgt der Ausschlag nach rechts bzw. links unabhängig von *Threshold2* und *Threshold3* immer einfarbig, wobei für den Ausschlag nach rechts *RGBFill* und für den Ausschlag nach links *RGBFill2* (s. u.) benutzt wird.

RGBFill2

Die Bedeutung dieser Farbeigenschaft hängt vom Wert der Eigenschaft *SymmMode* ab:

SymmMode = 0:

Farbe des Balkens für den Bereich des Eingangswertes zwischen *Threshold2* und *Threshold3*. Liegt *Threshold2* außerhalb des Anzeigebereichs, ist der Wert ohne Bedeutung.

SymmMode = 1:

Farbe des Balkens für den Bereich des Eingangswertes zwischen *Threshold2* und *Threshold3* bzw. zwischen -*Threshold2* und -*Threshold3*. Liegt *Threshold2* außerhalb des Anzeigebereichs, ist der Wert ohne Bedeutung.

SymmMode = 2:

Farbe des Balkens für negative Eingangswerte (Ausschlag nach links).

RGBFill3

Die Bedeutung dieser Farbeigenschaft hängt vom Wert der Eigenschaft *SymmMode* ab:

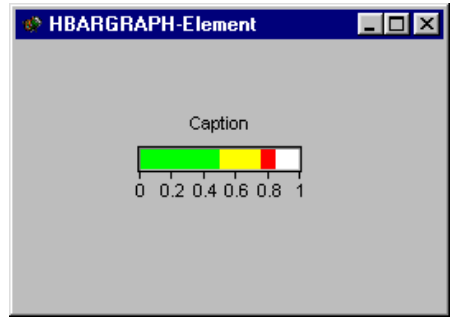
SymmMode = 0:

Farbe des Balkens für den Bereich des Eingangswertes oberhalb von *Threshold3*. Liegt *Threshold3* außerhalb des Anzeigebereichs, ist der Wert ohne Bedeutung.

SymmMode = 1:

Farbe des Balkens für den Bereich des Eingangswertes oberhalb von *Threshold3* bzw. unterhalb von -*Threshold3*. Liegt *Threshold3* außerhalb des Anzeige-

	bereichs, ist der Wert ohne Bedeutung. *SymmMode = 2:* Wert ist ohne Bedeutung
Threshold2	Eingangswert für Farbwechsel zwischen *RGBFill* und *RGBFill2*
Threshold3	Eingangswert für Farbwechsel zwischen *RGBFill2* und *RGBFill3*
Ticks	Anzahl der Skalenmarkierungen.
FontSize	Schriftgröße für die Skalenbeschriftung bzw. Überschrift in Pixeln.
ShowScale (0/1))	Legt fest, ob eine Skalenbeschriftung ausgegeben werden soll. Für *ShowScale = 0* wird keine Skalenbeschriftung (wohl aber die Skala selbst) ausgegeben.
Caption	Überschrift des Bargraphen. In die Überschrift kann auch durch Angabe einer entsprechenden Formatanweisung (siehe Elementtyp *NUMBER*) der aktuelle Eingangswert aufgenommen werden. Beispiel: Wird für *Caption* die Zeichenfolge *'Temp = %5.2f Grad'* angegeben, so lautet bei einem aktuellen Eingangswert von 25.5 Grad die Überschrift *'Temp = 25.50 Grad'*.



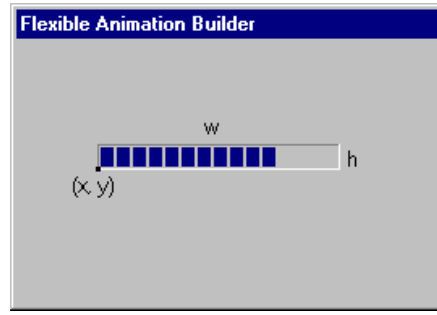
HBARGRAPH-*Element für* SymmMode = 1 *und einen aktuellen Eingangswert von etwas über 0.8*

Die Beispieldatei EXTBARGRAPH.BSY demonstriert die Anwendung des analogen Elementtyps *VBARGRAPH* (s. o.).

Elementtyp *PROGRESSBAR*

Der Elementtyp *PROGRESSBAR* stellt eine Windows-Standardverlaufsanzeige zur Verfügung, die über einen Blockein- bzw. -ausgang angesteuert wird. Er weist nachfolgend dargestellte Eigenschaften auf.

Eigenschaft	Bedeutung
Input	Auszugebende Ein- bzw. Ausgangsgröße. Nummern von Blockeingängen werden direkt angegeben, bei Blockausgängen muß der Wert 100 addiert werden. Beispiel: Ist *Input* = 4, wird der vierte Blockeingang ausgegeben, ist *Input* = 102, so wird der zweite Blockausgang angezeigt.
x	x-Koordinate des linken unteren Eckpunktes des umgebenden Rechtecks
y	y-Koordinate des linken unteren Eckpunktes des umgebenden Rechtecks
w	Breite des umgebenden Rechtecks in Pixeln.
h	Höhe des umgebenden Rechtecks in Pixeln.
Min	Kleinster dargestellter Wert von *Input* (Nullausschlag der Anzeige).
Max	Größter dargestellter Wert von *Input* (Vollausschlag der Anzeige).

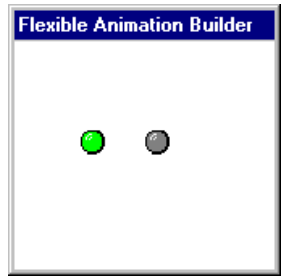


Grafikelement PROGRESSBAR

Elementtyp *LED*

Der Elementtyp *LED* stellt eine runde LED-Anzeige fester Größe zur Verfügung, die über einen Blockein- bzw. -ausgang angesteuert wird. Er weist nachfolgend dargestellte Eigenschaften auf.

Eigenschaft	Bedeutung
RGBFill	Farbe der LED im EIN-Zustand
Input	Anzuzeigende Ein- bzw. Ausgangsgröße. Nummern von Blockeingängen werden direkt angegeben, bei Blockausgängen muß der Wert 100 addiert werden. Beispiel: Ist *Input* = 4, wird der vierte Blockeingang ausgegeben, ist *Input* = 102, so wird der zweite Blockausgang angezeigt.
x	x-Koordinate des linken unteren Eckpunktes des umgebenden Rechtecks
y	y-Koordinate des linken unteren Eckpunktes des umgebenden Rechtecks
w	Breite des umgebenden Rechtecks in Pixeln.
h	Höhe des umgebenden Rechtecks in Pixeln.
OnValue	Schaltwert für EIN-Zustand. Wird *OnValue* überschritten, "leuchtet" die LED auf.



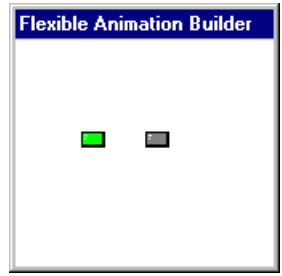
Grafikelement LED *im EIN- (links) bzw. AUS-Zustand (rechts)*

Die Beispieldatei CONTROLS.BSY erläutert den Einsatz des Elementtyps.

Elementtyp *RECTLED*

Der Elementtyp *RECTLED* stellt eine rechteckige LED-Anzeige fester Größe zur Verfügung, die über einen Blockein- bzw. -ausgang angesteuert wird. Er weist nachfolgend dargestellte Eigenschaften auf.

Eigenschaft	Bedeutung
RGBFill	Farbe der LED im EIN-Zustand
Input	Anzuzeigende Ein- bzw. Ausgangsgröße. Nummern von Blockeingängen werden direkt angegeben, bei Blockausgängen muß der Wert 100 addiert werden. Beispiel: Ist *Input* = 4, wird der vierte Blockeingang ausgegeben, ist *Input* = 102, so wird der zweite Blockausgang angezeigt.
x	x-Koordinate des linken unteren Eckpunktes des umgebenden Rechtecks
y	y-Koordinate des linken unteren Eckpunktes des umgebenden Rechtecks
w	Breite des umgebenden Rechtecks in Pixeln.
h	Höhe des umgebenden Rechtecks in Pixeln.
OnValue	Schaltwert für EIN-Zustand. Wird *OnValue* überschritten, "leuchtet" die LED auf.



Grafikelement RECTLED *im EIN- (links) bzw. AUS-Zustand (rechts)*

Die Beispieldatei CONTROLS.BSY erläutert den Einsatz des Elementtyps.

Elementtyp *YTPLOT*

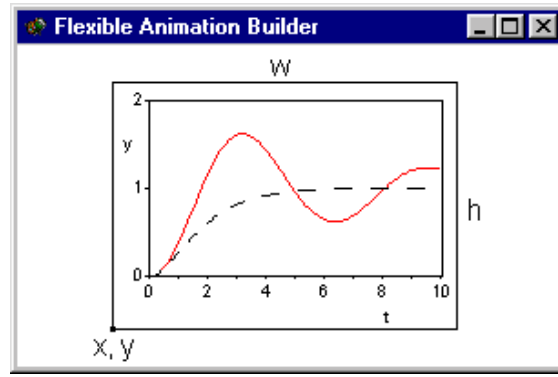
Der Elementtyp *YTPLOT* ermöglicht die grafische Darstellung des Zeitverlaufs einer oder mehrerer Eingangsgrößen des Blocks in einem y-t-Diagramm. Sollen mehrere Kurven in ein Diagramm gezeichnet werden (d. h. mehrere Eingangsgrößen über einer gemeinsamen Zeitachse dargestellt werden), so ist für jeden Eingang ein *YTPLOT*-Element einzufügen und die Eigenschaft *Group* (Gruppenindex) aller Elemente auf den gleichen positiven Wert (z. B. 1) zu setzen. Weisen mehrere *YTPLOT*-Elemente den gleichen positiven Gruppenindex auf, so ist das in der Elementtabelle zuerst auftretende Element der *Group Master*; die Eigenschaften dieses *Group Masters* (z. B. Skalierungseinstellungen etc.) legen dann das Aussehen des Diagramms fest, in das alle Kurven von *YTPLOT*-Elementen mit gleichem Gruppenidex gezeichnet werden. Auf diese weise können auch mehrere Kurvengruppen gebildet werden, die sich dann jeweils durch ihren Gruppenindex unterscheiden müssen. Wahlweise kann das Anzeigeelement auch im *Recorder*-Modus (d. h. mit mitlaufendem Zeitfenster) betrieben werden.

Insgesamt weist der Elementtyp *YTPLOT* die folgenden Eigenschaften auf:

Eigenschaft	Bedeutung
RGBStroke	Kurvenfarbe
LType	Bei positiven Werten wird die Kurve als durchgezogene Linie gezeichnet, der Wert von *LType* legt in diesem Fall die Liniendicke in Pixeln fest. Für negative Werte von *LType* werden unterschiedliche Linientypen (z. B. gestrichelt, strichpunktiert etc.) gezeichnet; die Liniendicke beträgt in diesem Fall immer ein Pixel.
x	x-Koordinate des linken unteren Eckpunktes des umgebenden Rechtecks des Diagramms in Pixeln
y	y-Koordinate des linken unteren Eckpunktes des umgebenden Rechtecks des Diagramms in Pixeln
w	Breite des umgebenden Rechtecks des Diagramms in Pixeln
h	Höhe des umgebenden Rechtecks des Diagramms in Pixeln
Input	Nummer des Blockeingangs, dessen Zeitverlauf dar-

	gestellt werden soll.
Group	Gruppenindex der Kurve. Ist der Gruppenindex -1 (Voreinstellung), wird die Kurve in ein eigenes Diagramm gezeichnet. Ist der Gruppenindex positiv, gehört die Kurve in eine Gruppe von Kurven, die alle den gleichen Gruppenindex besitzen. Diese Kurven werden alle in ein gemeinsames Diagramm gezeichnet, dessen Eigenschaften durch die Eigenschaften des *Group Masters* (s. o.) festgelegt werden.
DrawAxes (0/1)	Legt fest, ob die Koordinatenachsen gezeichnet werden sollen (*DrawAxes* = 1). Diese Einstellung ist nur von Belang, wenn die Kurve in ein eigenes Diagramm gezeichnet werden soll oder das Element der *Group Master* einer Gruppe von Elementen mit gleichem Gruppenindex ist.
AutoTimeScale (0/1)	Legt fest, ob die Skalierung der Zeitachse automatisch erfolgen soll (*AutoTimeScale* = 1). Diese Einstellung ist nur von Belang, wenn die Kurve in ein eigenes Diagramm gezeichnet werden soll oder das Element der *Group Master* einer Gruppe von Elementen mit gleichem Gruppenindex ist.
AutoAmpScale (0/1)	Legt fest, ob die Skalierung der Amplitudenachse automatisch erfolgen soll (*AutoAmpScale* = 1). Werden mehrere Kurven in ein Diagramm gezeichnet, erfolgt die autmatische Skalierung derart, dass alle Kurven vollständig im Darstellungsbereich liegen. Diese Einstellung ist nur von Belang, wenn die Kurve in ein eigenes Diagramm gezeichnet werden soll oder das Element der *Group Master* einer Gruppe von Elementen mit gleichem Gruppenindex ist.
LogAmpScale (0/1)	Legt fest, ob die Skalierung der Amplitudenachse logarithmisch erfolgen soll (*LogAmpScale* = 1) oder aber linear (*LogAmpScale* = 0). Diese Einstellung ist nur von Belang, wenn die Kurve in ein eigenes Diagramm gezeichnet werden soll oder das Element der *Group Master* einer Gruppe von Elementen mit gleichem Gruppenindex ist.
TimeMin, TimeMax	Legt den Anfangs- bzw. Endwert für Skalierung der

	Zeitachse fest, falls diese nicht automatisch erfolgt (s. o.). Diese Einstellung ist nur von Belang, wenn die Kurve in ein eigenes Diagramm gezeichnet werden soll oder das Element *Group Master* einer Gruppe von Elementen mit gleichem Gruppenindex ist.
AmpMin, AmpMax	Legt den Anfangs- bzw. Endwert für Skalierung der Amplitudenachse fest, falls diese nicht automatisch erfolgt (s. o.). Diese Einstellung ist nur von Belang, wenn die Kurve in ein eigenes Diagramm gezeichnet werden soll oder das Element der *Group Master* einer Gruppe von Elementen mit gleichem Gruppenindex ist.
FrameWidth	Legt die Breite des Diagrammrahmens in Pixeln fest. Für *FrameWidth* = 0 wird kein Rahmen gezeichnet. Diese Einstellung ist nur von Belang, wenn die Kurve in ein eigenes Diagramm gezeichnet werden soll oder das Element der *Group Master* einer Gruppe von Elementen mit gleichem Gruppenindex ist.
TimeText	Beschriftungstext für die Zeitachse
AmpText	Beschriftungstext für die Amplitudenachse
DrawGrid (0/1)	Gibt an, ob ein Raster gezeichnet werden soll
RecorderMode (0/1)	Ist *RecorderMode* = 1, wird das Element im Recorder-Modus, d. h. mit einem mitlaufenden Zeitfenster betrieben. In dieser Betriebsart legt die Eigenschaft *TimeMax* (s. o.) die Breite des Zeitfensters fest. Sobald der aktuelle Zeitwert den rechten Rand des Zeitfensters überschreitet, scrollt das Zeitfenster automatisch um einen Wert von *TimeMax*/2 weiter.



Grafikelement YTPLOT *(hier mit zwei Kurven unterschiedlichen Linientyps in einem Diagramm)*

Die Beispieldatei YTPLOTDEMO.BSY erläutert den Einsatz des Elementtyps.

Elementtyp *XYPLOT*

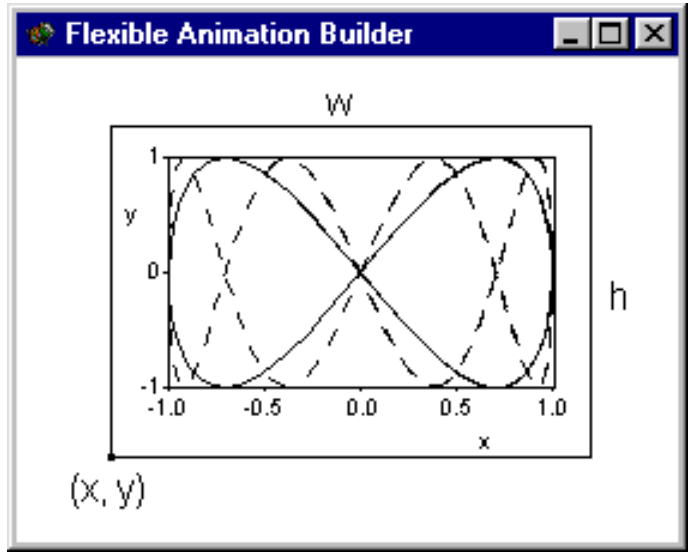
Der Elementtyp *XYPLOT* ermöglicht die grafische Darstellung einer oder mehrerer Trajektorien von Eingangsgrößen des Blocks in einem x-y-Diagramm. Sollen mehrere Trajektorien in ein Diagramm gezeichnet werden (d. h. mehrere Trajektorien über einem gemeinsamen Koordinatensystem dargestellt werden), so ist für jede Trajektorie ein *XYPLOT*-Element einzufügen und die Eigenschaft *Group* (Gruppenindex) aller Elemente auf den gleichen positiven Wert (z. B. 1) zu setzen. Weisen mehrere *XYPLOT*-Elemente den gleichen positiven Gruppenindex auf, so ist das in der Elementtabelle zuerst auftretende Element der *Group Master*; die Eigenschaften dieses *Group Masters* (z. B. Skalierungseinstellungen etc.) legen dann das Aussehen des Diagramms fest, in das alle Kurven von *XYPLOT*-Elementen mit gleichem Gruppenidex gezeichnet werden. Auf diese Weise können auch mehrere Kurvengruppen gebildet werden, die sich dann jeweils durch ihren Gruppenindex unterscheiden müssen.

Insgesamt weist der Elementtyp *XYPLOT* die folgenden Eigenschaften auf:

Eigenschaft	Bedeutung
RGBStroke	Kurvenfarbe
LType	Bei positiven Werten wird die Kurve als durchgezo-

	gene Linie gezeichnet, der Wert von *LType* legt in diesem Fall die Liniendicke in Pixeln fest. Für negative Werte von *LType* werden unterschiedliche Linientypen (z. B. gestrichelt, strichpunktiert etc.) gezeichnet; die Liniendicke beträgt in diesem Fall immer ein Pixel.
x	x-Koordinate des linken unteren Eckpunktes des umgebenden Rechtecks des Diagramms in Pixeln
y	y-Koordinate des linken unteren Eckpunktes des umgebenden Rechtecks des Diagramms in Pixeln
w	Breite des umgebenden Rechtecks des Diagramms in Pixeln
h	Höhe des umgebenden Rechtecks des Diagramms in Pixeln
x-Input	Nummer des ersten darzustellenden Blockeingangs (Abszisse des Diagramms)
y-Input	Nummer des zweiten darzustellenden Blockeingangs (Ordinate des Diagramms)
Group	Gruppenindex der Kurve. Ist der Gruppenindex -1 (Voreinstellung), wird die Kurve in ein eigenes Diagramm gezeichnet. Ist der Gruppenindex positiv, gehört die Kurve in eine Gruppe von Kurven, die alle den gleichen Gruppenindex besitzen. Diese Kurven werden alle in ein gemeinsames Diagramm gezeichnet, dessen Eigenschaften durch die Eigenschaften des *Group Masters* (s. o.) festgelegt werden.
DrawAxes (0/1)	Legt fest, ob die Koordinatenachsen gezeichnet werden sollen (*DrawAxes* = 1). Diese Einstellung ist nur von Belang, wenn die Kurve in ein eigenes Diagramm gezeichnet werden soll oder das Element der *Group Master* einer Gruppe von Elementen mit gleichem Gruppenindex ist.
AutoScale (0/1)	Legt fest, ob die Skalierung der Amplitudenachsen automatisch erfolgen soll (*AutoScale* = 1). Werden mehrere Kurven in ein Diagramm gezeichnet, erfolgt die autmatische Skalierung derart, dass alle Kurven

	vollständig im Darstellungsbereich liegen. Diese Einstellung ist nur von Belang, wenn die Kurve in ein eigenes Diagramm gezeichnet werden soll oder das Element der *Group Master* einer Gruppe von Elementen mit gleichem Gruppenindex ist.
xMin, xMax	Legt den Anfangs- bzw. Endwert für Skalierung der x-Achse (Abszisse) fest, falls diese nicht automatisch erfolgt (s. o.). Diese Einstellung ist nur von Belang, wenn die Kurve in ein eigenes Diagramm gezeichnet werden soll oder das Element der *Group Master* einer Gruppe von Elementen mit gleichem Gruppenindex ist.
yMin, yMax	Legt den Anfangs- bzw. Endwert für Skalierung der y-Achse (Ordinate) fest, falls diese nicht automatisch erfolgt (s. o.). Diese Einstellung ist nur von Belang, wenn die Kurve in ein eigenes Diagramm gezeichnet werden soll oder das Element der *Group Master* einer Gruppe von Elementen mit gleichem Gruppenindex ist.
FrameWidth	Legt die Breite des Diagrammrahmens in Pixeln fest. Für *FrameWidth* = 0 wird kein Rahmen gezeichnet. Diese Einstellung ist nur von Belang, wenn die Kurve in ein eigenes Diagramm gezeichnet werden soll oder das Element der *Group Master* einer Gruppe von Elementen mit gleichem Gruppenindex ist.
xText	Beschriftungstext für die x-Achse (Abszisse)
yText	Beschriftungstext für die y-Achse (Ordinate)
DrawGrid (0/1)	Gibt an, ob ein Raster gezeichnet werden soll



Grafikelement XYPLOT *(hier mit zwei Trajektorien unterschiedlichen Linientyps in einem Diagramm)*

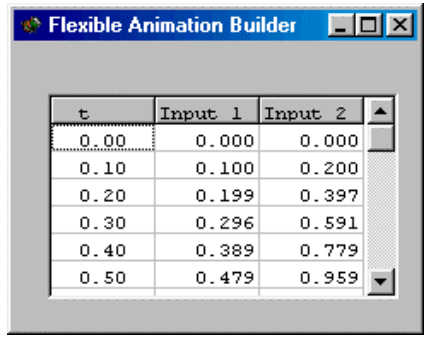
Die Beispieldatei XYPLOTDEMO.BSY erläutert den Einsatz des Elementtyps.

Elementtyp *TABLE*

Der Elementtyp *TABLE* ermöglicht die tabellarischer Darstellung einer oder mehrerer Eingangsgrößen des Blocks. Sollen mehrere Eingangsgrößen in eine gemeinsame Tabelle übernommen werden, so ist für jede Eingangsgröße ein *TABLE*-Element einzufügen und die Eigenschaft *Group* (Gruppenindex) aller Elemente auf den gleichen positiven Wert (z. B. 1) zu setzen. Weisen mehrere *TABLE*-Elemente den gleichen positiven Gruppenindex auf, so ist das in der Elementtabelle zuerst auftretende Element der *Group Master*; die Grundeigenschaften dieses *Group Masters* (z. B. Zeilenhöhe der Tabelle.) legen dann das Aussehen der Tabelle fest, in die alle Verläufe von *TABLE*-Elementen mit gleichem Gruppenidex gezeichnet werden. Auf diese Weise können auch mehrere Gruppen gebildet werden, die sich dann jeweils durch ihren Gruppenindex unterscheiden müssen. Insgesamt weist der Elementtyp *TABLE* die folgenden Eigenschaften auf:

Eigenschaft	Bedeutung
RGBFill	Hintergrundfarbe der Tabelle
x	x-Koordinate des linken unteren Eckpunktes des umgebenden Rechtecks der Tabelle in Pixeln
y	y-Koordinate des linken unteren Eckpunktes des umgebenden Rechtecks der Tabelle in Pixeln
w	Breite der Tabelle in Pixeln
h	Höhe der Tabelle in Pixeln
Input	Nummer des auszugebenden Blockeingangs
Group	Gruppenindex der Eingangsggröße. Ist der Gruppenindex -1 (Voreinstellung), wird die Größe in eine eigene Tabelle gezeichnet. Ist der Gruppenindex positiv, gehört die Größe in eine Gruppe von Größen, die alle den gleichen Gruppenindex besitzen. Diese Größen werden alle in eine gemeinsame Tabelle gezeichnet, deren Eigenschaften im wesentlichen durch die Eigenschaften des *Group Masters* (s. o.) festgelegt werden.
Caption	Spaltenüberschrift
ColumnWidth	Spaltenbreite in Pixels (Hinweis: Die Spaltenbreite kann für jede Eingangsgröße getrennt spezifiziert werden!)
RowHeight	Zeilenhöhe in Pixeln
Format	Ausgabeformat für die entsprechende Eingangsgröße (siehe Element NUMBER)
Font	Schriftart
FontSize	Schriftgröße
TimeMode	Legt fest, ob und in welcher Form die Zeit ausgegeben werden soll. Folgende Modi sind möglich: 0 Keine Zeitspalte 1 Zeitspalte mit Simulationszeit 2 Zeitspalte mit Echtzeit

	3 Zeitspalte mit Echtzeit und Datum
OutputInterval	Legt das Ausgabeintervall unabhängig von der Simulationsschrittweite fest. Ist *OutputInterval = 0*, erfolgt zu jedem Simulationsschritt auch eine Ausgabe in die Tabelle.
TimeColumnWidth	Spaltenbreite für Zeitausgabe
TimeFormat	Ausgabeformat für Simulationszeit falls *TimeMode = 1* (siehe Element NUMBER)



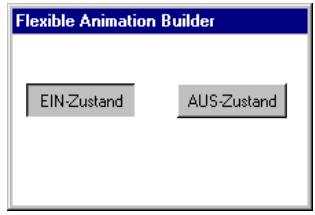
Grafikelement TABLE *(hier mit zwei Spalten für* TimeMode = 1*)*

Elementtyp *SWITCH*

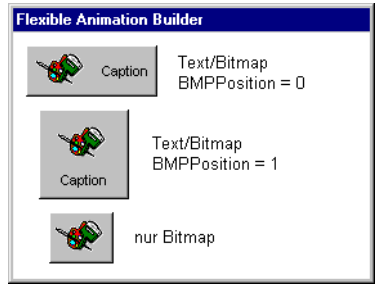
Der Elementtyp *SWITCH* stellt einen rechteckigen Schalter (wahlweise mit Text und/oder Grafik) zur Verfügung, der einen beliebigen Blockausgang ansteuert. Er weist nachfolgend dargestellte Eigenschaften auf.

Eigenschaft	Bedeutung
RGBStroke	Farbe der Schalter-Beschriftung
RGBFill	Schalterfarbe
Output	Nummer des Blockausgangs, auf den der Schalter wirkt
x	x-Koordinate des linken unteren Eckpunktes des umgebenden Rechtecks
y	y-Koordinate des linken unteren Eckpunktes des umgebenden Rechtecks

w	Breite des umgebenden Rechtecks in Pixeln.
h	Höhe des umgebenden Rechtecks in Pixeln.
OnValue	Ausgangswert des Schalters im gedrückten Zustand (EIN-Zustand). Im AUS-Zustand wird immer der Wert 0 ausgegeben.
Caption	Schalterbeschriftung. Auf Wunsch kann auch eine zustandsabhängige Beschriftung realisiert werden. Dazu sind die für den nicht gedrückten bzw. gedrückten Zustand auszugebenden Texte durch ein Doppelkreuz (#) zu trennen. Beispiel: Soll im nicht gedrückten Zustand die Beschriftung *Ein* und im gedrückten Zustand die Beschriftung *Aus* ausgegeben werden, so muss für *Caption* der Text *Ein#Aus* vorgegeben werden.
FontSize	Textgröße
BMPFile or Id	Bitmap-Datei, falls das Element mit einer Grafik versehen werden soll. Befindet sich das Bitmap in einer der beiden Bitmap-Libraries (siehe Kapitel *Nutzung von Bitmaps*), so kann es auch über seinen *Identifier* spezifiziert werden.
BMPPosition	Position der Grafik. Ist *BMPPosition* = 0, erscheint die Grafik links von der Beschriftung, sonst oberhalb des Textes.
Group	Dient zur Bildung von automatische Schaltergruppen, bei denen das Drücken eines Schalters den bisher gedrückten Schalter der Gruppe automatisch zurücksetzt. Ist *Group* = -1 (Voreinstellung), so fungiert der Schalter als Einzelschalter. Weist *Group* einen positiven Wert auf, so bildet der Schalter mit allen anderen Schaltern, die den gleichen *Group*-Wert besitzen, eine Schalter-Gruppe. Alle Schalter einer Gruppe sollten normalerweise auf den gleichen Blockausgang (Eigenschaft *Output*) wirken. Der jeweils gedrückte Schalter bestimmt dann den jeweils zugehörigen Blockausgangswert (Eigenschaften *OnValue*), alle anderen Schalter der Gruppe sind passiv. Die Beispieldatei SWITCHGROUP.BSY demonstriert den Einsatz von Schaltergruppen.



Bedienelement SWITCH *im EIN- (links) bzw. AUS-Zustand (rechts)*



Schaltflächen mit Bitmap-Grafik

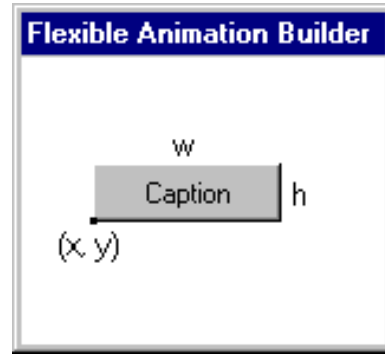
Die Beispieldatei CONTROLS.BSY erläutert den Einsatz dieses Elementtyps.

Elementtyp *BUTTON*

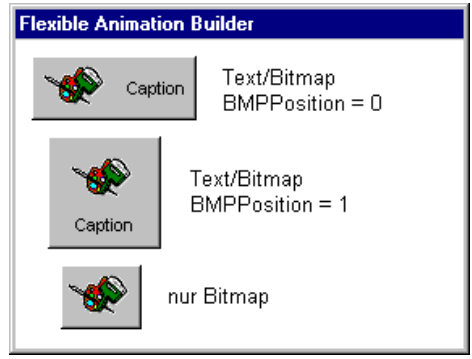
Der Elementtyp *BUTTON* stellt einen rechteckigen Taster (wahlweise mit Text und/oder Grafik) zur Verfügung, der einen beliebigen Blockausgang ansteuert. Er weist nachfolgend dargestellte Eigenschaften auf.

Eigenschaft	Bedeutung
RGBStroke	Farbe der Taster-Beschriftung
RGBFill	Tasterfarbe
Output	Nummer des Blockausgangs, auf den der Taster wirkt. Der Taster kann neben der normalen Betriebsart auch zur Simulationssteuerung bzw. für einige Sonderfunktionen benutzt werden, indem ein negativer Wert bzw. bestimmte positive Werte für *Output* angegeben werden (siehe Kapi-

	tel *Schaltflächen mit Sonderfunktion*)
x	x-Koordinate des linken unteren Eckpunktes des umgebenden Rechtecks
y	y-Koordinate des linken unteren Eckpunktes des umgebenden Rechtecks
w	Breite des umgebenden Rechtecks in Pixeln.
h	Höhe des umgebenden Rechtecks in Pixeln.
OnValue	Ausgangswert des Tasters im gedrückten Zustand (EIN-Zustand). Im AUS-Zustand wird immer der Wert 0 ausgegeben.
Caption	Tasterbeschriftung
FontSize	Textgröße
BMPFile or Id	Bitmap-Datei, falls das Element mit einer Grafik versehen werden soll. Befindet sich das Bitmap in einer der beiden Bitmap-Libraries (siehe Kapitel *Nutzung von Bitmaps*), so kann es auch über seinen *Identifier* spezifiziert werden
BMPPosition	Position der Grafik. Ist *BMPPosition* = 0, erscheint die Grafik links von der Beschriftung, sonst oberhalb des Textes.



Bedienelement BUTTON



Schaltflächen mit Bitmap-Grafik

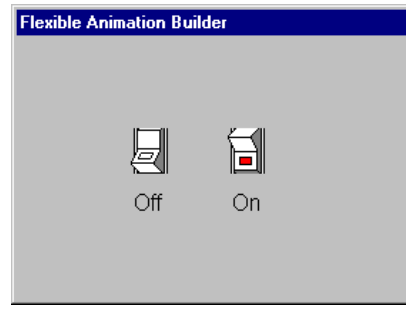
Die Beispieldatei CONTROLS.BSY erläutert den Einsatz dieses Elementtyps.

Elementtyp *DYNBMP*

Der Elementtyp *DYNBMP* stellt ein dynamisches Bitmap (Bitmap-Schaltfläche) zur Verfügung, das sein Aussehen bei jedem Anklicken zwischen zwei Zuständen wechselt, die über die beiden Bitmap-Dateien *OnBMPFile* und *OffBMPFile* festgelegt sind. Die Schaltfläche steuert einen beliebigen Blockausgang an und weist nachfolgend dargestellte Eigenschaften auf.

Eigenschaft	Bedeutung
Output	Nummer des Blockausgangs, auf den das Element wirkt
x	x-Koordinate des linken unteren Eckpunktes des umgebenden Rechtecks
y	y-Koordinate des linken unteren Eckpunktes des umgebenden Rechtecks
w	Breite des umgebenden Rechtecks in Pixeln.
h	Höhe des umgebenden Rechtecks in Pixeln.
OnValue	Ausgangswert der Schaltfläche im gedrückten (EIN-) Zustand. Im AUS-Zustand wird immer der Wert 0 ausgegeben.
OnBMPFile or Id	Bitmap-Datei für den EIN-Zustand. Befindet sich das Bitmap in einer der beiden Bitmap-

	Libraries (siehe Kapitel *Nutzung von Bitmaps*), so kann es auch über seinen *Identifier* spezifiziert werden. Achtung: Breite *w* und Höhe *h* des Bitmaps (s. o.) müssen in diesem Fall ggf. von Hand angepasst werden!
OffBMPFile or Id	Bitmap-Datei für den AUS-Zustand. Befindet sich das Bitmap in einer der beiden Bitmap-Libraries (siehe Kapitel *Nutzung von Bitmaps*), so kann es auch über seinen *Identifier* spezifiziert werden. Achtung: Breite *w* und Höhe *h* des Bitmaps (s. o.) müssen in diesem Fall ggf. von Hand angepasst werden!
Mode (0/1)	Legt fest, ob die Schaltfläche als Schalter (d. h. einrastend) oder als Taster arbeitet.



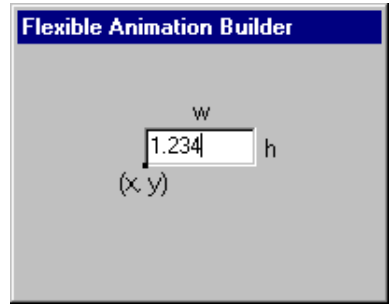
Beispiel für ein DYNBMP-*Bedienelement im AUS- (links) bzw. EIN-Zustand (rechts)*

Elementtyp *EDIT*

Der Elementtyp *EDIT* stellt ein Windows-Standard-Editierfeld zur Verfügung, das einen beliebigen Blockausgang ansteuert. Sämtliche Änderungen innerhalb des Editierfeldes machen sich sofort am Blockausgang bemerkbar. Das Editierfeld weist nachfolgend dargestellte Eigenschaften auf.

Eigenschaft	Bedeutung
RGBStroke	Textfarbe des Editierfeld-Textes
RGBFill	Hintergrundfarbe des Editierfeldes

Output	Nummer des Blockausgangs, an dem der im Editierfeld befindliche Wert ausgegeben wird. Enthält das Editierfeld keine gültigen (d. h. als Zahlenwert zu interpretierenden) Daten, wird der Wert 0 ausgegeben..
x	x-Koordinate des linken unteren Eckpunktes des umgebenden Rechtecks
y	y-Koordinate des linken unteren Eckpunktes des umgebenden Rechtecks
w	Breite des umgebenden Rechtecks in Pixeln.
h	Höhe des umgebenden Rechtecks in Pixeln.



Bedienelement EDIT

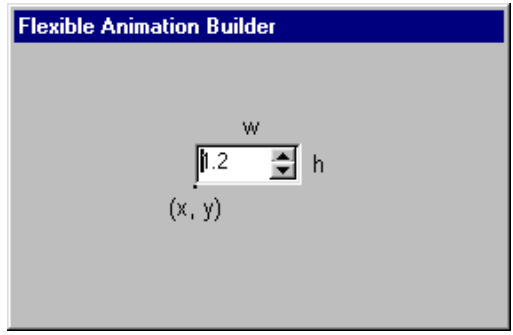
Die Beispieldatei CONTROLS.BSY erläutert den Einsatz dieses Elementtyps.

Elementtyp *SPINEDIT*

Der Elementtyp *SPINEDIT* stellt ein Editierfeld mit seitlichem Scroller (Spinelement) zur Verfügung, das einen beliebigen Blockausgang ansteuert. Sämtliche Änderungen innerhalb des Editierfeldes machen sich sofort am Blockausgang bemerkbar. Das Editierfeld weist nachfolgend dargestellte Eigenschaften auf.

Eigenschaft	Bedeutung
RGBStroke	Textfarbe des Editierfeld-Textes

RGBFill	Hintergrundfarbe des Editierfeldes
Output	Nummer des Blockausgangs, an dem der im Editierfeld befindliche Wert ausgegeben wird. Enthält das Editierfeld keine gültigen (d. h. als Zahlenwert zu interpretierenden) Daten, wird der Wert 0 ausgegeben..
x	x-Koordinate des linken unteren Eckpunktes des umgebenden Rechtecks
y	y-Koordinate des linken unteren Eckpunktes des umgebenden Rechtecks
w	Breite des umgebenden Rechtecks in Pixeln.
h	Höhe des umgebenden Rechtecks in Pixeln.
Increment	Wert, um den die Anzeige bei Betätigung der Spinelemente erhöht bzw. erniedrigt wird.
MinValue	Kleinster darstellbarer Zahlenwert
MaxValue	Größter darstellbarer Zahlenwert
Format	Zahlen-Ausgabeformat (siehe *NUMBER*-Grafikelement)
NoBlanks (0/1)	Gibt an, ob führende Leerzeichen bei der Ausgabe unterdrückt werden (*NoBlanks* = 1) oder nicht (*NoBlanks* = 0)



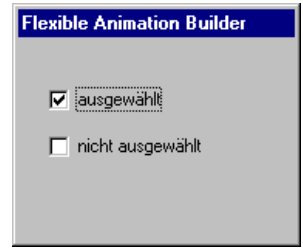
Bedienelement SPINEDIT

Die Datei SPINEDITDEMO.BSY erläutert den Einsatz dieses Elementtyps.

Elementtyp *CHECKBOX*

Der Elementtyp *CHECKBOX* stellt ein Windows-Standard-Schaltfeld zur Verfügung, das einen beliebigen Blockausgang ansteuert. Er weist nachfolgend dargestellte Eigenschaften auf.

Eigenschaft	Bedeutung
RGBStroke	Textfarbe des Schaltfeldes
RGBFill	Hintergrundfarbe des Schaltfeldes. Diese wird beim Einfügen eines neuen Schaltfeldes automatisch auf die aktuelle Hintergrundfarbe des Visualisierungsfensters voreingestellt.
Output	Nummer des Blockausgangs, auf den das Schaltfeld wirkt.
x	x-Koordinate des linken unteren Eckpunktes des umgebenden Rechtecks
y	y-Koordinate des linken unteren Eckpunktes des umgebenden Rechtecks
w	Breite des umgebenden Rechtecks in Pixeln.
h	Höhe des umgebenden Rechtecks in Pixeln.
OnValue	Ausgangswert des Schaltfeldes im ausgewählten Zustand (EIN-Zustand). Im AUS-Zustand wird immer der Wert 0 ausgegeben.
Caption	Schaltfeldbeschriftung



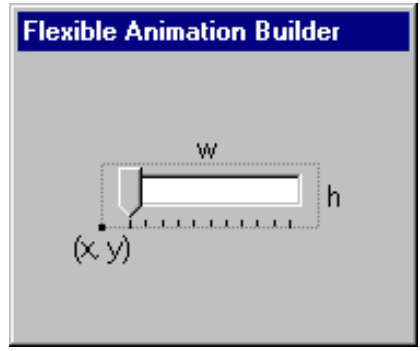
Bedienelement CHECKBOX im ausgewählten (oben) bzw. nicht ausgewählten Zustand (unten)

Die Beispieldatei CONTROLS.BSY erläutert den Einsatz dieses Elementtyps.

Elementtyp *TRACKBAR*

Der Elementtyp *TRACKBAR* stellt einen horizontalen Windows-Standard-Schieberegler zur Verfügung, der einen beliebigen Blockausgang ansteuert. Er weist nachfolgend dargestellte Eigenschaften auf.

Eigenschaft	Bedeutung
Output	Nummer des Blockausgangs, auf den der Schieberegler wirkt.
x	x-Koordinate des linken unteren Eckpunktes des umgebenden Rechtecks
y	y-Koordinate des linken unteren Eckpunktes des umgebenden Rechtecks
w	Breite des umgebenden Rechtecks in Pixeln.
h	Höhe des umgebenden Rechtecks in Pixeln.
Min	Ausgangswert bei Linksanschlag des Reglers
Max	Ausgangswert bei Rechtsanschlag des Reglers



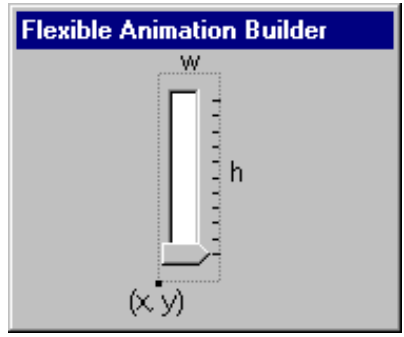
Bedienelement TRACKBAR

Die Beispieldatei CONTROLS.BSY erläutert den Einsatz dieses Elementtyps.

Elementtyp *VTRACKBAR*

Der Elementtyp *VTRACKBAR* stellt einen vertikalen Windows-Standard-Schieberegler zur Verfügung, der einen beliebigen Blockausgang ansteuert. Er weist nachfolgend dargestellte Eigenschaften auf.

Eigenschaft	Bedeutung
Output	Nummer des Blockausgangs, auf den der Schieberegler wirkt.
x	x-Koordinate des linken unteren Eckpunktes des umgebenden Rechtecks
y	y-Koordinate des linken unteren Eckpunktes des umgebenden Rechtecks
w	Breite des umgebenden Rechtecks in Pixeln.
h	Höhe des umgebenden Rechtecks in Pixeln.
Min	Ausgangswert am unteren Anschlag des Reglers
Max	Ausgangswert am oberen Anschlag des Reglers



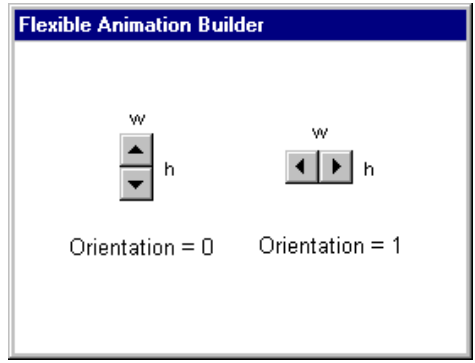
Bedienelement VTRACKBAR

Elementtyp *UPDOWN*

Der Elementtyp *UPDOWN* stellt einen vertikalen oder horizontalen Windows-Standard-Wippregler (Spin-Element) zur Eingabe von inkrementellen Werten

zur Verfügung, der einen beliebigen Blockausgang ansteuert. Er weist nachfolgend dargestellte Eigenschaften auf.

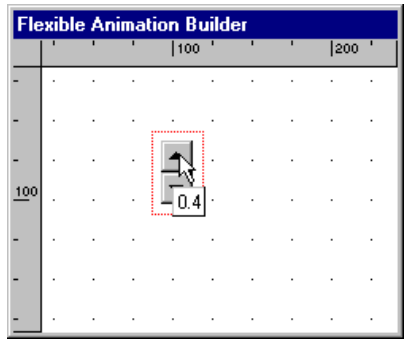
Eigenschaft	Bedeutung
Output	Nummer des Blockausgangs, auf den der Wippregler wirkt.
x	x-Koordinate des linken unteren Eckpunktes des umgebenden Rechtecks
y	y-Koordinate des linken unteren Eckpunktes des umgebenden Rechtecks
w	Breite des umgebenden Rechtecks in Pixeln.
h	Höhe des umgebenden Rechtecks in Pixeln.
Min	Ausgangswert am unteren Anschlag des Reglers
Max	Ausgangswert am oberen Anschlag des Reglers
Increment	Inkrement des Wippreglers
Orientation (0/1)	Legt fest, ob ein vertikaler oder horizontaler Wippregler gezeichnet wird



Bedienelement UPDOWN

Da die aktuelle Position (aktueller Ausgangswert) des Wippreglers während des Entwurfs ohne Zuhilfenahme eines zusätzlichen Anzeigeelements nicht ohne weiteres erkennbar ist, wird in der Entwurfsphase bzw. immer dann, wenn BO-

RIS sich nicht in der Simulation befindet, zur Hilfestellung der aktuelle Ausgangswert des Elements in einem kleinen Anzeigefenster (Hint-Fenster) angezeigt (siehe nachfolgende Bildschirmgrafik).



Anzeige des aktuellen Ausgangswertes (hier 0.4) während der Entwurfsphase bzw. ausserhalb der Simulation über ein Hint-Fenster

Die Datei UPDOWNDEMO.BSY erläutert den Einsatz dieses Elementtyps.

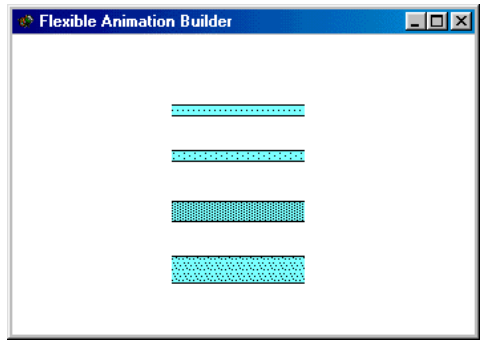
Elementtyp *HPIPE*

Der Elementtyp *HPIPE* beinhaltet ein horizontales Leitungsstück, das animiert arbeitet und z. B. zum Aufbau von Strömungssystemen, zur Visualisierung fließender Ströme und für ähnliche Anwendungszwecke benutzt werden kann. Er weist nachfolgend dargestellte Eigenschaften auf.

Eigenschaft	Bedeutung
RGBStroke	Randfarbe der Leitung
RGBFill	Füllfarbe der Leitung
LType	Randdicke der Leitung in Pixeln. Für *LType = 0* wird die Leitung randlos dargestellt. Dadurch lässt sich dieser Elementtyp z. B. auch für gemusterte Füllungen verwenden.
x	x-Koordinate des linken unteren Eckpunktes des umgebenden Rechtecks
y	y-Koordinate des linken unteren Eckpunktes des umgebenden Rechtecks

w	Länge der Leitung in Pixeln
h	Dicke der Leitung in Pixeln
Input	Ein- bzw. Ausgangsgröße zur Steuerung der Animation. Nummern von Blockeingängen werden direkt angegeben, bei Blockausgängen muß der Wert 100 addiert werden. Beispiel: Ist *Input* = 4, wird der vierte Blockeingang ausgegeben, ist *Input* = 102, so wird der zweite Blockausgang angezeigt. Die Richtung des Flusses wird über das Vorzeichen des Signals gesteuert und kann zudem über die Eigenschaft *Direction* (s. u.) beeinflusst werden. Ist das Signal 0, findet keine Bewegung statt. Zusätzlich sind über spezielle Werte von *Input* folgende Sonderfunktionen verfügbar: 0 keine Animation -1 ständige Animation -2 zufällige Bewegung in x-Richtung -3 zufällige Bewegung in y-Richtung -4 zufällige Bewegung in x- und y-Richtung
Direction (0/1)	Für *Direction = 0* ist die Flussrichtung bei positiver Steuergröße am *Input*-Eingang von links nach rechts, für *Direction = 1* in umgekehrter Richtung.
FillPattern	Füllmuster der Leitung. Erlaubt sind Werte zwischen 0 und 4. Bei Wahl eines anderen Wertes wird die Leitung ohne Füllmuster (und damit auch ohne Animation!) dargestellt.
LeftEnd	Legt das Aussehen des linken Leitungsendes fest. Erlaubt sind Werte zwischen 0 und 5. Hierüber kann z. B. eine Leitungsöffnung vorgegeben werden, die eine Verbindung mit einem vertikalen Leitungsstück ermöglicht.
RightEnd	Legt das Aussehen des rechten Leitungsendes fest. Erlaubt sind Werte zwischen 0 und 5. Hierüber kann z. B. eine Leitungsöffnung vorgegeben werden, die eine Verbindung mit einem vertikalen Leitungsstück ermöglicht.
Delay	Verzögerung der Animation in Millisekunden. Durch Variation dieses Wertes kann die Geschwindigkeit der Animation (Fließgeschwindigkeit) an die speziellen Erforder-

nisse angepasst werden. Auch eine eingangsgesteuerte Fließgeschwindigkeit ist damit möglich.



Beispiele für HPIPE-*Leitungselemente*

Die Beispieldateien ELECTRICCIRCUIT.BSY und TANKS.BSY erläutern den Einsatz dieses Elementtyps.

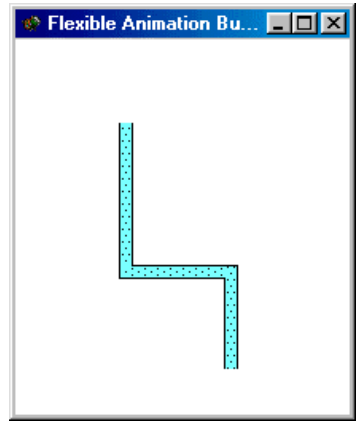
Elementtyp *VPIPE*

Der Elementtyp *VPIPE* beinhaltet ein vertikales Leitungsstück, das animiert arbeitet und z. B. zum Aufbau von Strömungssystemen, zur Visualisierung fließender Ströme und für ähnliche Anwendungszwecke benutzt werden kann. Er weist nachfolgend dargestellte Eigenschaften auf.

Eigenschaft	Bedeutung
RGBStroke	Randfarbe der Leitung
RGBFill	Füllfarbe der Leitung
LType	Randdicke der Leitung in Pixeln. Für *LType = 0* wird die Leitung randlos dargestellt. Dadurch lässt sich dieser Elementtyp z. B. auch für gemusterte Füllungen verwenden.
x	x-Koordinate des linken unteren Eckpunktes des umgebenden Rechtecks
y	y-Koordinate des linken unteren Eckpunktes des umgebenden Rechtecks

w	Dicke der Leitung in Pixeln
h	Länge der Leitung in Pixeln
Input	Ein- bzw. Ausgangsgröße zur Steuerung der Animation. Nummern von Blockeingängen werden direkt angegeben, bei Blockausgängen muß der Wert 100 addiert werden. Beispiel: Ist *Input* = 4, wird der vierte Blockeingang ausgegeben, ist *Input* = 102, so wird der zweite Blockausgang angezeigt. Die Richtung des Flusses wird über das Vorzeichen des Signals gesteuert und kann zudem über die Eigenschaft *Direction* (s. u.) beeinflusst werden. Ist das Signal 0, findet keine Bewegung statt. Zusätzlich sind über spezielle Werte von *Input* folgende Sonderfunktionen verfügbar: 0 keine Animation -1 ständige Animation -2 zufällige Bewegung in x-Richtung -3 zufällige Bewegung in y-Richtung -4 zufällige Bewegung in x- und y-Richtung
Direction (0/1)	Für *Direction = 0* ist die Flussrichtung bei positiver Steuergröße am *Input*-Eingang von oben nach unten, für *Direction = 1* in umgekehrter Richtung.
FillPattern	Füllmuster der Leitung. Erlaubt sind Werte zwischen 0 und 4. Bei Wahl eines anderen Wertes wird die Leitung ohne Füllmuster (und damit auch ohne Animation!) dargestellt.
TopEnd	Legt das Aussehen des oberen Leitungsendes fest. Erlaubt sind Werte zwischen 0 und 5. Hierüber kann z. B. eine Leitungsöffnung vorgegeben werden, die eine Verbindung mit einem horizontalen Leitungsstück ermöglicht.
BottomEnd	Legt das Aussehen des unteren Leitungsendes fest. Erlaubt sind Werte zwischen 0 und 5. Hierüber kann z. B. eine Leitungsöffnung vorgegeben werden, die eine Verbindung mit einem horizontalen Leitungsstück ermöglicht.
Delay	Verzögerung der Animation in Millisekunden. Durch Variation dieses Wertes kann die Geschwindigkeit der Animation (Flussgeschwindigkeit) an die speziellen Erforder-

nisse angepasst werden.



Verbindung aus zwei VPIPE-*Elementen und einem* HPIPE-*Element*

Die Beispieldateien ELECTRICCIRCUIT.BSY, ASSEMBLYLINE.BSY und TANKS.BSY erläutern den Einsatz dieses Elementtyps.

Elementtyp *TANK*

Der Elementtyp *TANK* dient zur Darstellung von Tanksystemen und kann z. B. zum Aufbau von Strömungssystemen und für ähnliche Anwendungszwecke benutzt werden kann. Er weist nachfolgend dargestellte Eigenschaften auf.

Eigenschaft	Bedeutung
RGBStroke	Randfarbe des Tanks
RGBFill	Farbe des Tankinhalts
LType	Randdicke des Tanks in Pixeln
x	x-Koordinate des linken unteren Eckpunktes des umgebenden Rechtecks
y	y-Koordinate des linken unteren Eckpunktes des umgebenden Rechtecks
w	Breite des Tanks in Pixeln
h	Höhe des Tanks in Pixeln

FillHeight (%) Füllhöhe des Tanks in Prozent. Diese Eigenschaft kann z. B. an einen Blockeingang gekoppelt werden, um die Füllhöhe dynamisch zu ändern.

FillPattern Füllmuster des Tankinhalts. Erlaubt sind Werte zwischen 0 und 4. Bei Wahl eines anderen Wertes wird der Tank ohne Füllmuster (und damit auch ohne Animation!) dargestellt.

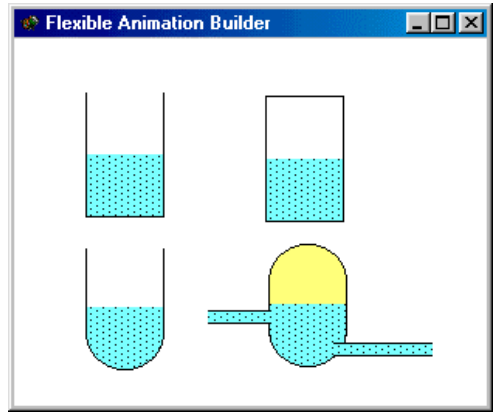
Delay Verzögerung der Animation in Millisekunden. Durch Variation dieses Wertes kann die Geschwindigkeit der Animation (Flussgeschwindigkeit) an die speziellen Erfordernisse angepasst werden.

Shape Legt die Tankform (rechteckig, rund, dreieckig) fest. Erlaubt sind Werte zwischen 0 und 5.

RGBBack Hintergrundfarbe des Tanks

Move Legt fest, ob der Tankinhalt bewegt (animiert) dargestellt wird. Folgende Werte sind erlaubt:

0	Keine Animation
1	Fließende Bewegung in x-Richtung
2	Zufällige Bewegung in x-Richtung



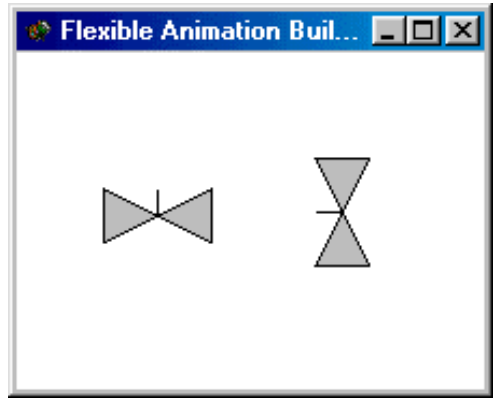
Einige TANK-*Elemente (teilweise mit* HPIPE-*Anschlüssen)*

Die Beispieldateien TANKS.BSY und ASSEMBLYLINE.BSY erläutern den Einsatz dieses Elementtyps.

Elementtyp *VALVE*

Der Elementtyp *VALVE* dient zur Darstellung von Ventilen und kann beim Aufbau von Strömungssystemen und für ähnliche Anwendungszwecke benutzt werden kann. Die Ventilfarbe kann optional in Abhängigkeit vom Ventilzustand (offen oder geschlossen) wechseln. Dieser Elementtyp weist nachfolgend dargestellte Eigenschaften auf.

Eigenschaft	Bedeutung
RGBStroke	Randfarbe des Ventils
RGBFill	Füllfarbe des Ventils für den Fall *Input = -1*
LType	Randdicke des Ventils in Pixeln
x	x-Koordinate des linken unteren Eckpunktes des umgebenden Rechtecks
y	y-Koordinate des linken unteren Eckpunktes des umgebenden Rechtecks
w	Breite des Ventils in Pixeln
h	Höhe des Ventils in Pixeln
Input	Ein- bzw. Ausgangsgröße zur Steuerung der Ventilfarbe (Schaltzustand, s. u.). Nummern von Blockeingängen werden direkt angegeben, bei Blockausgängen muß der Wert 100 addiert werden. Beispiel: Ist *Input* = 4, wird der vierte Blockeingang ausgegeben, ist *Input* = 102, so wird der zweite Blockausgang angezeigt. Für *Input = -1* hat das Ventil immer die unter *RGBFill* eingestellte Farbe.
OnValue	Gibt im Falle *Input <> -1* den Wert des an *Input* anliegenden Signals an, bei dem das Ventil seine Farbe von *RGBClosed* nach *RGBOpen* wechselt.
RGBOpen	Füllfarbe des Ventils im geöffneten Zustand für *Input <> -1*
RGBClosed	Füllfarbe des Ventils im geschlossenen Zustand für *Input <> -1*
Orientation	Ausrichtung des Ventils



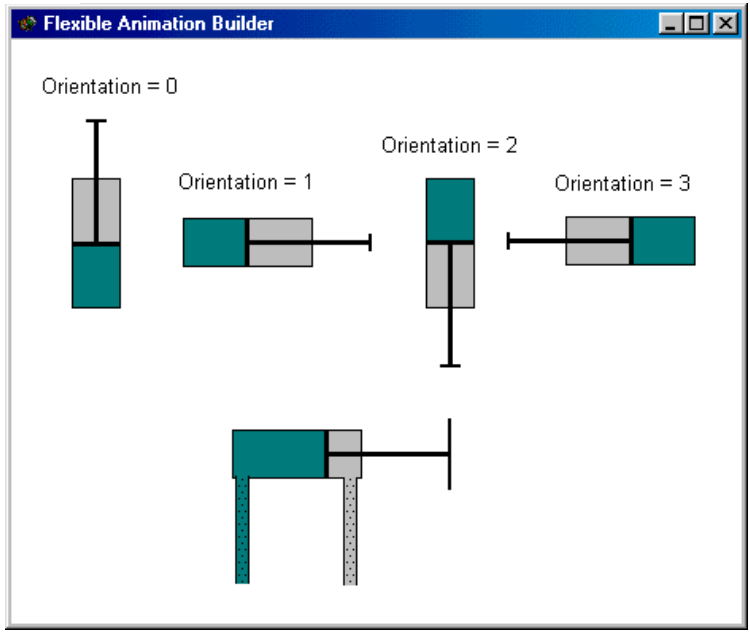
Zwei VALVE-*Elemente mit unterschiedlicher Ausrichtung*

Elementtyp *HYDCYL*

Der Elementtyp *HYDCYL* ermöglicht die Darstellung von (Hydraulik-) Zylindern und kann beispielsweise beim Aufbau pneumatischer Systeme und für ähnliche Anwendungszwecke benutzt werden. Dieser Elementtyp weist nachfolgend dargestellte Eigenschaften auf.

Eigenschaft	Bedeutung
RGBStroke	Randfarbe des Zylinders
RGBFill	Füllfarbe des Zylinders unterhalb des Kolbens
LType	Randdicke des Zylinders in Pixeln
x	x-Koordinate des linken unteren Eckpunktes
y	y-Koordinate des linken unteren Eckpunktes
w	Breite des Zylinders in Pixeln
h	Länge des Zylinders in Pixeln
SliderPos [%]	Position des Kolbens. 0% bedeutet, dass sich der Kolben in seiner untersten Position befindet, bei 100% ist er maximal ausgefahren.
RGBBack	Füllfarbe des Zylinders oberhalb des Kolbens

Orientation	Ausrichtung des Zylinders
RGBSlider	Farbe von Kolben, Stange und Schieber
SliderSize	Breite des Schiebers am Stangenende in Pixeln



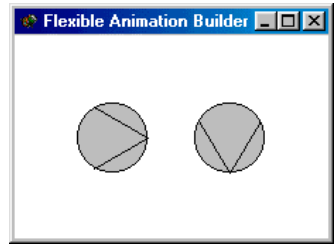
HYDCYL-*Elemente mit unterschiedlicher Ausrichtung und mit angeschlossenen* VPIPE-*Elementen (unten)*

Elementtyp *PUMP*

Der Elementtyp *PUMP* dient zur Darstellung von Pumpen und kann beim Aufbau von Strömungssystemen und für ähnliche Anwendungszwecke benutzt werden kann. Die Pumpenfarbe kann optional in Abhängigkeit vom Pumpenzustand (laufend oder stehend) wechseln. Dieser Elementtyp weist nachfolgend dargestellte Eigenschaften auf.

Eigenschaft	**Bedeutung**
RGBStroke	Randfarbe der Pumpe

RGBFill	Füllfarbe der Pumpe für den Fall *Input = -1*
LType	Randdicke der Pumpe in Pixeln
x	x-Koordinate des linken unteren Eckpunktes des umgebenden Rechtecks
y	y-Koordinate des linken unteren Eckpunktes des umgebenden Rechtecks
w	Durchmesser der Pumpe in Pixeln
Input	Ein- bzw. Ausgangsgröße zur Steuerung der Pumpenfarbe (Pumpenzustand, s. u.). Nummern von Blockeingängen werden direkt angegeben, bei Blockausgängen muß der Wert 100 addiert werden. Beispiel: Ist *Input* = 4, wird der vierte Blockeingang ausgegeben, ist *Input* = 102, so wird der zweite Blockausgang angezeigt. Für *Input = -1* hat die Pumpe immer die unter *RGBFill* eingestellte Farbe.
OnValue	Gibt im Falle *Input <> -1* den Wert des an *Input* anliegenden Signals an, bei dem die Pumpe ihre Farbe von *RGBStopped* nach *RGBRunning* wechselt.
RGBStopped	Füllfarbe der Pumpe im stehenden Zustand für *Input <> -1*
RGBRunning	Füllfarbe der Pumpe im laufenden Zustand für *Input <> -1*
Orientation	Ausrichtung der Pumpe



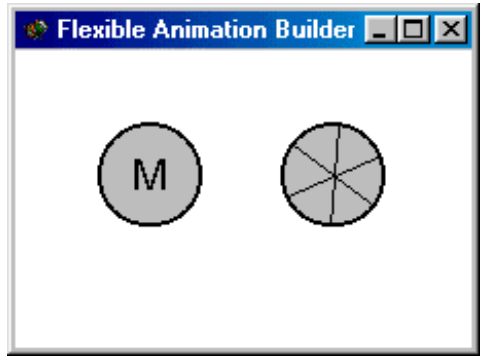
Zwei PUMP-*Elemente mit unterschiedlicher Ausrichtung*

Elementtyp *MOTOR*

Der Elementtyp *MOTOR* stellt einen Motor dar, der auf Wunsch animiert werden kann (Drehrichtung und Drehgeschwindigkeit eingangsgesteuert). Die Motorfarbe kann optional in Abhängigkeit vom Motorzustand (laufend oder

stehend) wechseln. Dieser Elementtyp weist nachfolgend dargestellte Eigenschaften auf.

Eigenschaft	Bedeutung
RGBStroke	Randfarbe des Motors
RGBFill	Füllfarbe des Motors für den Fall *Input = -1*
LType	Randdicke des Motors in Pixeln
x	x-Koordinate des linken unteren Eckpunktes des umgebenden Rechtecks
y	y-Koordinate des linken unteren Eckpunktes des umgebenden Rechtecks
w	Durchmesser des Motors in Pixeln
Input	Ein- bzw. Ausgangsgröße zur Steuerung der Motorfarbe (Motorzustand, s. u.) sowie der Drehrichtung im animierten Fall. Nummern von Blockeingängen werden direkt angegeben, bei Blockausgängen muß der Wert 100 addiert werden. Beispiel: Ist *Input* = 4, wird der vierte Blockeingang ausgegeben, ist *Input* = 102, so wird der zweite Blockausgang angezeigt. Für *Input = -1* hat der Motor immer die unter *RGBFill* eingestellte Farbe. Weist *Input* hingegen eine gültige Blockein- bzw. Ausgangsnummer auf, so wird der Motor bei einem Signalwert von 0 als stehend in der Farbe *RGBStopped* dargestellt, bei einem negativen Signalwert als linkslaufend und bei einem positiven Signalwert als rechtslaufend, jeweils mit der Farbe *RGBRunning*.
RGBStopped	Füllfarbe des Motors im stehenden Zustand für *Input <> -1*
RGBRunning	Füllfarbe des Motors im laufenden Zustand für *Input <> -1*
Animated (0/1)	Legt fest, ob der Motor animiert werden soll.
Delay	Legt die Drehgeschwindigkeit des Motors im animierten Fall fest. Diese Eigenschaft kann an einen Blockein- bzw. -ausgang angekoppelt werden, so dass der Motor auch mit variabler Drehzahl dargestellt werden kann.



Zwei MOTOR-*Elemente (rechts animierte Darstellung)*

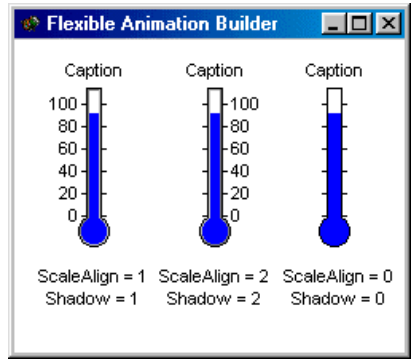
Die Beispieldateien MOTOR.BSY und ASSEMBLYLINE.BSY erläutern den Einsatz dieses Elementtyps.

Elementtyp *THMETER*

Der Elementtyp *THMETER* stellt ein stilisiertes Thermometer dar, das über einen Blockein- bzw. -ausgang angesteuert wird. Das Thermometer kann stufenlos skaliert werden und weist nachfolgend dargestellte Eigenschaften auf.

Eigenschaft	Bedeutung
RGBBack	Hintergrundfarbe
RGBFill	Füllfarbe des Thermometers
Input	Auszugebende Ein- bzw. Ausgangsgröße. Nummern von Blockeingängen werden direkt angegeben, bei Blockausgängen muß der Wert 100 addiert werden. Beispiel: Ist *Input* = 4, wird der vierte Blockeingang ausgegeben, ist *Input* = 102, so wird der zweite Blockausgang angezeigt.
x	x-Koordinate des linken unteren Eckpunktes des umgebenden Rechtecks
y	y-Koordinate des linken unteren Eckpunktes des umgebenden Rechtecks

w	Breite des umgebenden Rechtecks in Pixeln.
h	Höhe des umgebenden Rechtecks in Pixeln.
Min	Kleinster dargestellter Wert von *Input* (Nullausschlag der Anzeige).
Max	Größter dargestellter Wert von *Input* (Vollausschlag der Anzeige).
Shadow (0=none)	Typ des dargestellten Schattens. Erlaubt sind die Werte 0, 1, und 2. Beim Wert 0 wird kein Schatten dargestellt.
Ticks	Anzahl der Skalenmarkierungen.
FontSize	Schriftgröße für die Skalenbeschriftung bzw. Überschrift in Pixeln.
ScaleAlign (0=none)	Ausrichtung der Skalenbeschriftung (links bzw. rechts). Für *ScaleAlign = 0* wird keine Skalenbeschriftung (wohl aber die Skala selbst) ausgegeben.
Caption	Überschrift des Thermometers. In die Überschrift kann auch durch Angabe einer entsprechenden Formatanweisung (siehe Elementtyp *NUMBER*) der aktuelle Eingangswert (Temperaturwert) aufgenommen werden. Beispiel: Wird für *Caption* die Zeichenfolge *'Temp = %5.2f Grad'* angegeben, so lautet bei einem aktuellen Temperaturwert von 25.5 Grad die Überschrift *'Temp = 25.50 Grad'*.



Grafikelement THMETER *für verschiedene Werte der Eigenschaften* ScaleAlign *und* Shadow

Elementtyp *KEY*

Über den Elementtyp *KEY* können innerhalb eines FAB-Visualisierungsfensters Tastaturbotschaften abgefangen und zur Steuerung eines beliebigen Blockausgangs benutzt werden.. Das Element weist nachfolgend dargestellte Eigenschaften auf.

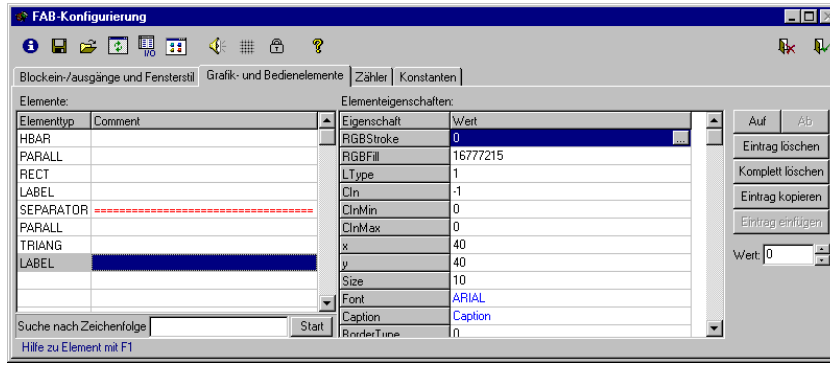
Eigenschaft	Bedeutung
Output	Nummer des Blockausgangs, auf den die Taste wirkt.
OnValue	Wert, den der Blockausgang bei gedrückter Taste erhält. Im nicht gedrückten Zustand wird immer der Wert 0 ausgegeben.
KeyCode	Tastaturcode der gewählten Taste. Um an diesen Code zu gelangen, aktivieren Sie zunächst im Entwurfsmodus durch Anklicken mit der Maus das Visualisierungsfenster. Drücken Sie nachfolgend die gewünschte Taste, so erscheint im Titelbalken des Fenster kurzzeitig der zugehörige Tastencode.
ToggleMode (0/1)	Betriebsart des Elements. Ist *ToggleMode = 0*, so ist der Ausgang nur aktiv, solange die Taste gedrückt ist (Taste wirkt wie ein Taster). Ist *ToggleMode = 1*, wird bei jedem Tastendruck zwischen aktivem und inaktivem Ausgang umgeschaltet (Taste wirkt als Schalter).

Elementtyp KEY *zur Entwurfszeit. Im Titelbalken des Visualisierungsfensters wird beim Drücken der Taste der zugehörige Tastencode (hier 18) angezeigt.*

Das Visualisierungsfenster kann den Tastendruck zur Laufzeit gewöhnlich nur dann verarbeiten, wenn es aktiv ist (erkennbar am farbigen Titelbalken). Der FAB bietet aber die Möglichkeit, während der Simulation bei Auftreten eines Tastendrucks das zugehörige Fenster *automatisch* zu aktivieren. Dazu ist auf der Palette *Visualisierungsfenster* des FAB-Konfigurierungsdialogs die Option *Fenster bei Tastaturbotschaft aktivieren* zu aktivieren. Beachten Sie dabei aber, dass diese Option nur dann korrekt funktionieren kann, wenn nicht für zwei FAB-Module Tastaturbotschaften mit *gleichen* Tastencodes definiert wurden!

Elementtyp *SEPARATOR*

Der Elementtyp *SEPARATOR* weist keinerlei Funktionalität innerhalb des Visualisierungsfensters auf, sondern dient lediglich zur optischen Gruppierung zusammengehöriger Elemente innerhalb der Elementtabelle des Konfigurierungsdialogs in Form eines farbigen waagrechten Trennstrichs. Die Farbe des Trennstrichs lässt sich über die Eigenschaft *RGBStroke* spezifizieren.



Konfigurierungsdialog mit eingefügtem SEPARATOR-*Element in der Elementtabelle*

Spezifizierung der Elementeigenschaften

Um eine Elementeigenschaft zu modifizieren, muß diese zunächst durch Anklicken der entsprechenden Zelle der Elementtabelle selektiert werden. Im Anschluß daran kann dann bei den meisten Eigenschaften unmittelbar der gewünschte Wert eingegeben werden. Bei einigen Elementeigenschaften (z. B.

BITMAP-Dateien, Stift- oder Füllfarbe, Schriftarten) erfolgt die Änderung über einen entsprechenden Auswahldialog.

Verknüpfen von Elementeigenschaften mit Blockein- oder -ausgängen

Um überhaupt eine dynamische Visualisierung erstellen zu können, muß es möglich sein, bestimmte Elementeigenschaften (z. B. die Position oder Größe eines Elements oder seine Füllfarbe) mit Blockeingängen (u. U. aber auch mit Blockausgängen) zu verknüpfen. Der *Flexible Animation Builder* bietet dazu weitreichende Möglichkeiten.

Input/Output-Eigenschaft

Zunächst besitzen einige Elementtypen (z. B. *NUMBER* oder *HBAR*) eine *Input*-Eigenschaft, die die Nummer des Blockeingangs angibt, der vom Element angezeigt werden soll. Wird für ein *NUMBER*-Element z. B. *Input* = 2 gesetzt, so zeigt dieses den Wert des zweiten Blockeingangs an. Analog dazu weisen alle Bedienelemente (z. B. *SWITCH* oder *CHECKBOX*) eine *Output*-Eigenschaft auf, die die Nummer des Blockausgangs angibt, der vom Element angesteuert wird.

Die Besonderheit liegt nun darin, dass für die *Input*-Eigenschaft statt eines Blockeingangs auch ein Blockausgang angegeben werden kann, indem die Nummer des Blockausgangs erhöht um 100 angegeben wird (also z. B. *Input* = 103 für den dritten Blockausgang). Ein *NUMBER*-Element wäre dann also z. B. mit dem dritten Blockausgang verbunden. Dadurch ist es möglich, mit einem Bedienelement (z. B. einem Schieberegler, der auf den dritten Blockausgang wirkt) direkt, d. h. auch ohne äußere Rückführung innerhalb der BORIS-Struktur, ein Anzeigeelement anzusteuern. Dies funktioniert im übrigen unabhängig davon, ob der entsprechende Blockausgang überhaupt aus dem Block "herausgeführt" wird. Die Beispieldateien BARDEMO.BSY und CONTROLS.BSY (siehe nachfolgende Bildschirmgrafik) demonstrieren diesen Effekt.

Visualisierungsfenster von CONTROLS.BSY. Der obere linke Schalter wirkt auf den ersten Blockausgang (Output = 1). Der Zustand des Schalters wird direkt im Visualisierungsfenster durch das LED*-Element rechts neben dem Schalter angezeigt, welches demnach auf Blockausgang 1 gelegt ist (*Input *= 101). Auch Taster, Editierfeld und Schaltfeld besitzen eine entsprechende Kontrollausgabe rechts neben dem Element.*

Darüber hinaus erlaubt es das FAB-Modul, auch für (fast) alle anderen Elementeigenschaften eine nahezu beliebige Abhängigkeit zwischen Elementeigenschaft und Blockein- bzw. -ausgängen über einen Formelparser zu spezifizieren. Dabei werden die (ggf. skalierten, siehe Abschnitt *Konfigurierung der Modulein- und -ausgänge*) Blockeingänge mit *I1*, *I2*, *I3* usw., die Blockausgänge mit *O1*, *O2*, *O3*... bezeichnet. Soll sich z. B. ein Element in Abhängigkeit des ersten Blockeingangs horizontal bewegen, so verknüpft man die x-Position des Elements mit dem Eingang *I1*, beispielsweise in der Form

$$x = 100*I1+50$$

Dazu wird der rechte Teil des obenstehenden Ausdrucks einfach in die entsprechende Zelle der Tabelle eingegeben. Die in diesem Beispiel erfolgte Skalierung des ersten Eingangs (Multiplikation mit 100, Addition von 50) kann selbstverständlich auch bereits bei der Eingangsskalierung *(*Palette *Blockein/ausgänge und Fensterstil* des Dialogs) erfolgen.

Der integrierte Formelparser erlaubt die nachfolgend aufgelisteten Operationen.

Syntax	**Funktion**
+	Plus (Addition)
-	Minus und monadisches Minus
*	Multiplikation

/	Division
^	Potenzoperator
( )	Klammern (max. Verschachtelungstiefe 20)
Pi	Zahl π
ln	natürlicher Logarithmus
log	Zehnerlogarithmus
sqr	Quadrat
sqrt	Wurzel
exp	Exponentialfunktion
sin	Sinus
cos	Cosinus
tan	Tangens
asin	Arcussinus
acos	Arcuscosinus
atan	Arcustangens
sinh	Sinus hyperbolicus
cosh	Cosinus hyperbolicus
tanh	Tangens hyperbolicus
abs	Absolutwert
int	ganzzahliger Anteil
frac	gebrochener Anteil
round	rundet auf ganze Zahl
sign	Signumfunktion
step	Sprungfunktion (Heavisidefunktion)

Der Funktionsstring darf maximal 255 Zeichen aufweisen. Groß- und Kleinschreibung werden nicht unterschieden.

Jeder Blockein- oder -ausgang kann selbstverständlich auf mehrere Elementeigenschaften (auch unterschiedlicher Elemente) wirken; ebenso kann eine Elementeigenschaft von mehreren Blockein- oder -ausgängen gleichzeitig abhängig sein.

Festlegung von Element-Farbeigenschaften

Bei der Verknüpfung von Element-Farbeigenschaften (z. B. Elementeigenschaften *RGBStroke* oder *RGBFill*) mit Blockeingängen ist zu beachten, dass die Farbe im sogenannten *RGB-Modus* definiert werden muß, d. h. die Farbzahl sich aus (R)ot-, (G)rün- und (B)lauanteil zusammensetzt. Intern ist dies eine 3-Byte-Zahl, wobei das niederwertige Byte den Rotanteil, das mittlere Byte den Grünanteil und das höchstwertige Byte den Blauanteil festlegt. Wird der Ro-

tanteil mit *r*, der Grünanteil mit *g* und der Blauanteil mit *b* bezeichnet, so ergibt sich die Farbzahl *F* also zu

$$F = r + 256*g + 65536*b$$

Für reines Rot ergibt sich also z. B. ein Farbwert von *F* = 255, für reines Blau ein Farbwert von *F* = 65536*256 = 16711680. Soll nun z. B. die Füllfarbe eines Elements in Abhängigkeit vom ersten Blockeingang fließend von blau nach rot übergehen, so kann dies durch einen Ausdruck der Form

*RGBFill = ROUND(I1)*65536+(255-ROUND(I1))*

realisiert werden (siehe auch Beispieldatei VARIABLECOLOR.BSY). Die *ROUND()*-Funktion ist dabei erforderlich, um auch bei nicht-ganzzahligen Eingangsgrößenwerten des Blockeingangs korrekte Farbwerte zu generieren.

Dynamische Anpassung der Ausgabe an Fenstergröße

Manchmal soll die Ausgabe innerhalb des Visualisierungsfensters nicht absolut, sondern in Abhängigkeit von der aktuellen Fenstergröße erfolgen. Zu diesem Zweck bietet FAB die Möglichkeit, die aktuelle Breite und Höhe des Client-Bereichs (d. h. des Innenbereichs) des Visualisierungsfensters in die Koordinatenberechnung einzubeziehen. Dies geschieht innerhalb des Formelparsers über die Variablen CW (für *client width*) für die Fensterbreite bzw. CH (für *client height*) für die Fensterhöhe. Soll also z. B. ein Rechteck der Breite *w* = 60 und der Höhe *h* = 40 unabhängig von der aktuellen Fenstergröße immer zentriert im Fenster erscheinen, so gelingt Ihnen dies über die Beziehungen

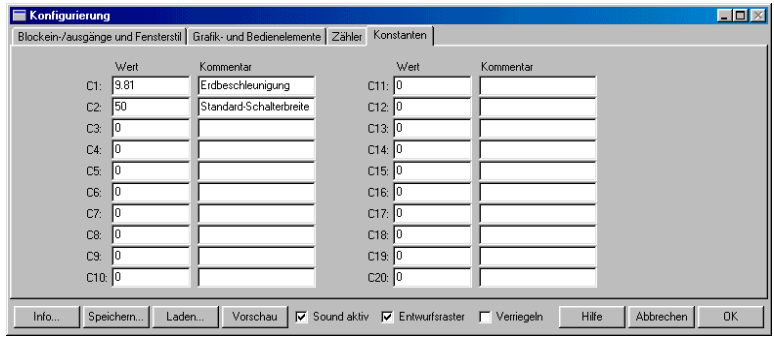
$$x = CW/2 - 30$$
$$y = CH/2 + 20$$

Der Wert 30 im ersten Ausdruck entspricht der halben Breite des Rechtecks, der Wert 20 im zweiten Ausdruck der halben Höhe. Da die y-Koordinate von oben nach unten verläuft, ist der Wert hier zu addieren, während er im ersten Ausdruck subtrahiert werden muß (siehe hierzu auch SNOWMAN.BSY).

Verwendung von Konstanten

Bei der Spezifizierung der meisten Elementeigenschaften können neben festen Zahlenwerten und Blockein- bzw. -ausgängen auch benutzerdefinierte Konstanten verwendet werden. Dabei sind bis zu 20 Konstanten verfügbar, die mit

C1, *C2*, ..., *C20* bezeichnet werden und über die Palette *Konstanten* des Konfigurierungsdialogs spezifiziert werden können.

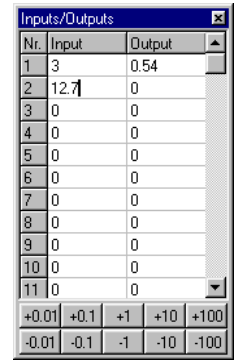


Palette Konstanten *des Konfigurierungsdialogs (hier nach Definition zweier Konstanten)*

Durch die Verwendung von Konstanten ist es beispielsweise möglich, die Eigenschaften einer Reihe von Elementen *gleichzeitig* zu ändern, indem diese Eigenschaft einfach über eine Konstante definiert wird und diese Konstante geändert wird. Beispiel: Sollen innerhalb des Visualisierungsfensters 20 Rechtecke gleicher Breite angezeigt werden, legt man zunächst die Eigenschaft *w* aller Rechtecke auf die Konstante *C1* und legt den Wert von *C1* z. B. mit 50 fest. Stellt man später fest, dass eine Breite von z. B. *w* = 40 angebrachter wäre, so braucht man lediglich den Wert von *C1* entsprechend zu modifizieren.

Das I/O-Kontrollfenster

In der Regel ist es wünschenswert, die erstellte Visualisierung oder Bedienoberfläche testen zu können, ohne dazu die FAB-Konfigurierung verlassen und zu BORIS zurückkehren zu müssen. Dazu dient das *I/O-Kontrollfenster*, über das sich zu Testzwecken sämtliche Blockeingangswerte modifizieren und sämtliche Blockausgangswerte kontrollieren lassen. Das Fenster lässt sich über die Schaltfläche  der Toolbar des Konfigurierungsdialogs ein- bzw. ausblenden.



Das I/O-Kontrollfenster

Ändert sich z. B. durch Betätigung eines Bedienelements innerhalb des Visualisierungsfensters eine Ausgangsgröße, so wird der entsprechende Wert in der *Output*-Spalte des I/O-Kontrollfensters automatisch aktualisiert. In der *Input*-Spalte können beliebige Eingaangswerte vorgegeben werden. Alternativ dazu kann der Eingangswert in der selektierten Zelle auch über die Schaltflächen am unteren Rand des Fensters schrittweise inkrementiert oder dekrementiert werden.

Schaltflächen mit Sonderfunktion

Simulationssteuerung

Das *BUTTON*-Bedienelement kann neben seiner "normalen" Funktion auch zur Simulationssteuerung von BORIS eingesetzt werden (z. B. Starten oder Stoppen der Simulation). Es besitzt dann die gleiche Funktion wie die entsprechenden Menübefehle des Untermenüs SIMULATION bzw. die entsprechenden Schaltflächen der BORIS-Systemtoolbar. Dazu wird die *Output*-Eigenschaft des Bedienelements auf einen *negativen* Wert gesetzt. Nachfolgende Tabelle gibt einen Überblick über die zur Verfügung stehenden Funktionen.

Wert von *Output*	**entspricht Menüoption SIMULATION |...**	**entspricht Systemtoolbar-Schaltfläche..**
-1	START	

-2	START ENDLOSSIMULATION	
-3	STOPP	
-4	PAUSE/EINZELSCHRITTMODUS	
-5	EINZELSCHRITT AUSFÜHREN	

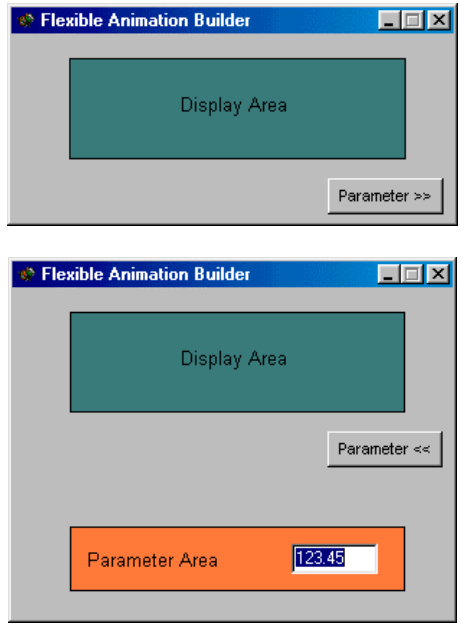
Die Beispieldatei SIMCONTROL.BSY demonstriert den Einsatz des BUTTON-Bedienelmentes zur Simulationssteuerung.

Aufklappbare Visualisierungs- und Bedienfenster

In manchen Fällen ist es wünschenswert, bestimmte Bedien- oder Anzeigeelemente eines Fensters nur bei Bedarf sichtbar und damit zugänglich zu machen. In der Praxis wird dies meistens durch sogenannte *aufklappbare* (Dialog-) Fenster realisiert, die bestimmte Bereiche auf Knopfdruck ein- und auch wieder ausblenden.

Auch mit dem *Animation Builder* sind derartige Fenster realisierbar, und zwar sowohl in Form von nach unten aufklappbaren als auch nach rechts aufklappbaren Visualisierungs- und Bedienfenstern. Dazu wird die *Output*-Eigenschaft des BUTTON-Bedienlementes, welches das Auf- bzw. Zuklappen des Fensters bewirken soll, auf einen Wert von 1000+*dy* (Aufklappen nach unten) bzw. 2000+*dx* (Aufklappen nach rechts) gesetzt. Dabei ist *dx* bzw. *dy* der Wert in Pixeln, um den sich das Fenster bei Betätigung der Schaltfläche vergrößern bzw. wieder verkleinern soll. Wird also für *Output* z. B. ein Wert von 1100 vorgegeben, so vergrößert sich das Fenster bei Betätigung der Schaltfläche um 100 Pixel und verkleinert sich bei der erneuten Betätigung wieder um den entsprechenden Wert. Die Beispieldatei VARSIZEDLG.BSY (siehe nachfolgende Bildschirmgrafik) demonstriert die Realisierung dieser Funktion.

Weiterhin ist es möglich, dem BUTTON-Bedienelement, welches für das Auf- und Zuklappen zuständig ist, eine zustandsabhängige Beschriftung zu verleihen. Dazu sind die für den zu- bzw. aufgeklappten Zustand gewünschten Beschriftungtexte durch ein Doppelkreuz (#) zu trennen und der Eigenschaft *Caption* des Bedienelements zuzuweisen. Beispiel: Soll die Beschriftung im zugeklappten Zustand *Öffnen* und im aufgeklappten Zustand *Schließen* lauten, so muss *Caption* zu *Öffnen#Schließen* gesetzt werden.



Aufklappbares Fenster im geschlossenen Zustand (oben) und nach Betätigung der Parameter>>*-Schaltfläche im geöffneten Zustand (unten)*

Weitere Sonderfunktionen

Neben den bereits aufgeführten Optionen stehen einige weitere Sonderfunktionen zur Verfügung, die in nachfolgender Tabelle aufgeführt sind.

Wert von *Output*	**Funktion**
-6	BORIS beenden (eine noch laufende Simulation wird ggf. zuvor automatisch beendet)
-7	Inhalt des Visualisierungsfensters auf dem Standarddrucker ausgeben
-8	Inhalt des Visualisierungsfensters als Bitmap in die Zwischenablage kopieren
-99	Online-Hilfe zum Visualisierungsfenster aufrufen.

Dazu wird die im Konfigurierungsdialog unter *Hilfe-Datei* spezifizierte Datei geladen. Dies kann entweder eine ASCII-Textdatei (Endung TXT) oder eine Windows-Hilfedatei (Endunh HLP) sein.

Hinweis: Die beschriebenen Funktionen sind nur dann verfügbar, wenn Ihre BORIS-Version mindestens die Build-Nummer 162 aufweist. Sollten Sie eine ältere BORIS-Version besitzen, können Sie eine aktuelle Version von unserer Web-Site unter *www.winfact.de* beziehen.

Sondereingänge

Ein FAB-Modul besitzt neben den bis zu 50 Blockeingängen einige weitere sog. "Sondereingänge", die keine echten Blockeingänge darstellen, aber bei der Spezifizierung von Grafikelement-Eigenschaften wie gewöhnliche Eingänge behandelt werden können. Die Sondereingänge besitzen die Bezeichnungen I51-I54 und haben folgende Funktionen:

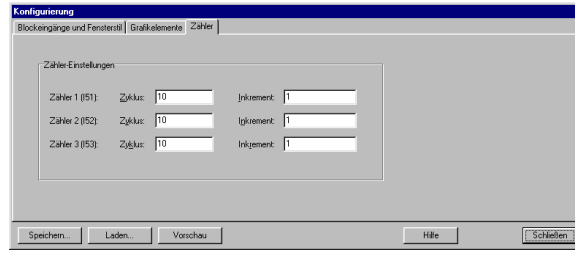
Bezeichnung	Funktion
I51	Zähler 1
I52	Zähler 2
I53	Zähler 3
I54	Aktueller Zeitwert

Zähler

Die ersten drei Sondereingänge sind also mit Zählern belegt, die zu Beginn der Simulation auf 0 gesetzt und dann bei jedem Simulationsschritt um ein benutzerdefiniertes Inkrement erhöht werden. Für jeden Zähler kann ein Zyklus vorgegeben werden, nach dem der Zähler auf 0 zurückgesetzt wird. Hat Zähler 1 beispielsweise einen Zyklus von 10, wird er beim 11. Simulationsschritt wieder auf 1 gesetzt und zählt dann entsprechend weiter. Die Zykluswerte und Inkremente für die drei Zähler können über die Palette *Zähler* des Konfigurationsdialogs spezifiziert werden. Voreingestellt ist jeweils ein Zyklus von 10 und ein Inkrement von 1.

Die Zähler können benutzt werden, um sich zyklisch wiederholende Abläufe zu realisieren, z. B. indem mehrere übereinanderliegende Bitmaps zyklisch wechselweise aktiviert bzw. deaktiviert werden, so dass eine fließende Bewegung

entsteht. Dazu wird der Anzeigestatus eines jeden Bitmaps (Eigenschaft *CIn*) auf den entsprechenden Zählereingang (z. B. 51 für Zähler 1) gelegt und der Zyklusbereich des Zählers gleichmäßig auf die Sichtbarkeitsgrenzen [*CInMIn*, *CInMax*] aufgeteilt. Eine einfache Beispiel-Animation nach diesem Schema befindet sich in der Datei COUNTERDEMO1.BSY im *FAB-DEMOS*-Verzeichnis.



Zähler-Einstellungen

Eingangsabhängiger Inkrement

Zählerzykluswerte und Inkremente sind Fließkommawerte, so dass nicht zwangsläufig ganzzahlige Werte angegeben werden müssen. Die Zählerinkremente können außerdem *eingangsabhängig* gesteuert werden, so dass Animationen mit variabler Geschwindigkeit (z. B. ein laufender Motor mit eingangsgesteuerter Drehzahl) möglich sind. Soll z. B. das Inkrement für Zähler 1 über den ersten Blockeingang gesteuert werden, so ist für das Inkrement *I1* anzugeben. Die Datei COUNTERDEMO2.BSY im *FAB-DEMOS*-Verzeichnis enthält eine entsprechende Beispielanwendung.

Aktueller Zeitwert

Der vierte Sondereingang (I54) enthält bei jedem Simulationsschritt den aktuellen Zeitwert. Er kann vom Anwender für verschiedene Anwendungszwecke benutzt werden.

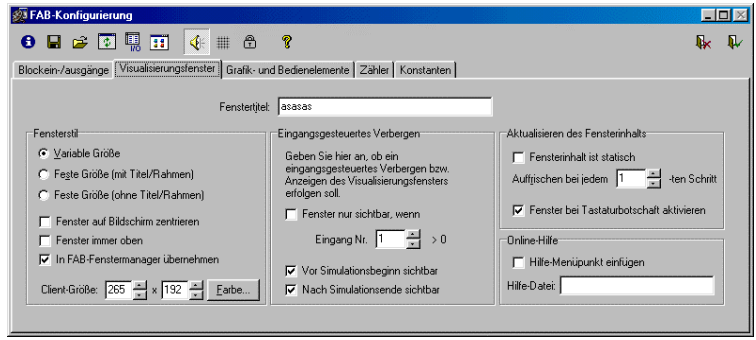
Die Vorschaufunktion des FAB

Um die Auswirkung einzelner Änderungen direkt und ohne jeweiliges Verlassen des Konfigurationsdialogs sichtbar werden zu lassen, besitzt dieser eine Vorschaufunktion. Der Dialog selbst wird beim Aufruf automatisch in der linken unteren Bildschirmecke positioniert, das Visualisierungsfenster in der rechten oberen Ecke, so dass es möglichst vollständig sichtbar ist (nach Verlassen des Dialogs wird es wieder an seine Position zurückgesetzt). Durch Betäti-

gung der Schaltfläche der Toolbar können dann alle aktuellen Änderungen in das Visualisierungsfenster übernommen werden. Bei der Eingabe von Ganzzahlparametern (z. B. Position oder Größe eines Elements) kann die Vorschau auch automatisiert werden, um z. B. ein *LABEL*-Element direkt an die gewünschte Position zu schieben. Ist nämlich eine entsprechende Zelle der Elementtabelle selektiert, erscheint der Zelleninhalt automatisch in dem Eingabefeld *Wert* und kann dann dort über das entsprechende Spin-Element inkrementiert oder dekrementiert werden, wobei nach jeder Änderung automatisch ein Auffrischen des Visualisierungsfensters erfolgt. Außerdem erfolgt eine automatische Auffrischung jedesmal dann, wenn innerhalb der Elementtabelle ein anderes Element angewählt wird (also bei jedem Zeilenwechsel!).

Eingangsgesteuertes Verbergen des Visualisierungsfensters

In einigen Fällen kann es erwünscht sein, ein FAB-Visualisierungsfenster zur Laufzeit (d. h. während einer Simulation) unter bestimmten Bedingungen zu verbergen (d. h. unsichtbar zu machen) und später ggf. wieder anzuzeigen. Dies ist über die Optionen möglich, die sich auf der Palette *Visualisierungsfenster* des Konfigurierungsdialogs in der Gruppenbox *Eingangsgesteuertes Verbergen* befinden.



Optionen zum eingangsgesteuerten Verbergen des Visualisierungsfensters

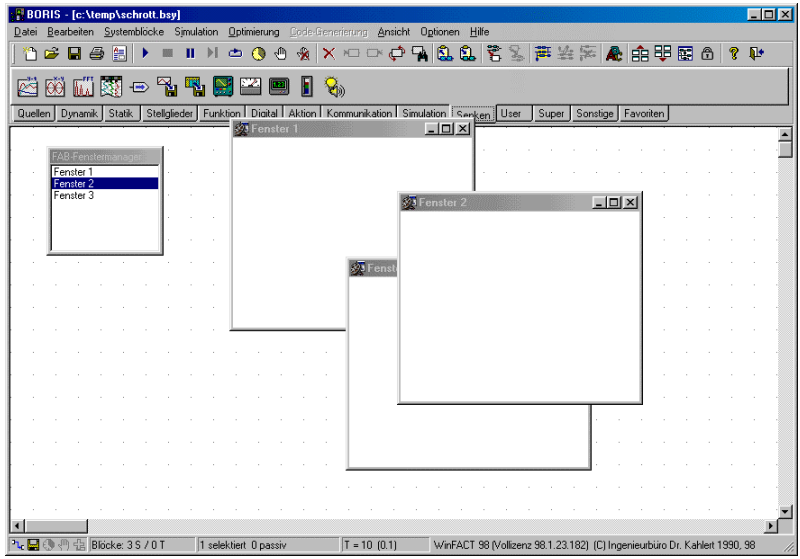
Zur Aktivierung dieser Programmoption muss zunächst die Option *Fenster nur sichtbar, wenn* angewählt werden. Unter *Eingang Nr.* wird dann der Blockein-

gang angegeben, über den später zur Laufzeit das Anzeigen bzw. Verbergen des Visualisierungsfensters gesteuert werden soll. Weist das Signal am entsprechenden Blockeingang dann einen Wert größer 0 auf, ist das Visualisierungsfenster sichtbar, ansonsten unsichtbar.

Über die Option *Vor Simulationsbeginn sichtbar* kann der Status des Fensters nach dem Laden der entsprechenden BORIS-Struktur (d. h. vor Simulationsbeginn) festgelegt werden. Entsprechendes gilt für die Option *Nach Simulationsende sichtbar*. Ist letztere deaktiviert, so behält das Visualisierungsfenster nach Ende der Simulation seinen aktuellen Status bei (ist also unter Umständen unsichtbar!).

Der FAB-Fenstermanager

Enthält eine BORIS-Struktur mehrere FAB-Blöcke, so können die zugehörigen Visualisierungsfenster bei Bedarf über ein zentrales Auswahlfenster (den *FAB-Fenstermanager*) verwaltet werden. Dadurch wird zur Laufzeit ein schnelles Umschalten zwischen den Fensters ohne umständliche "Fensterschieberei" möglich.



BORIS-Hauptfenster mit drei (hier leeren) FAB-Visualisierungsfenstern und dem aktivierten FAB-Fenstermanager

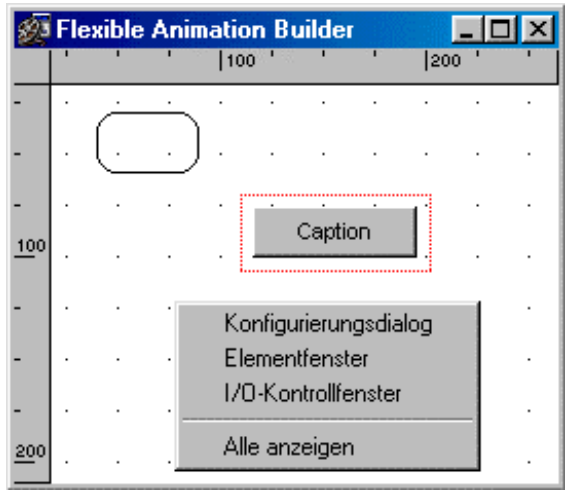
Um die Verwaltung eines Fensters über den Fenstermanager zu ermöglichen, muss lediglich die Option *In FAB-Fenstermanager übernehmen* auf der Palette *Visualisierungsfenster* des FAB-Konfigurierungsdialogs aktiviert werden. Nach dem Verlassen des Dialogs erscheint dann automatisch der Fenstermanager mit dem neuen Eintrag in Form des zugehörigen Fenstertitels. Durch Anklicken eines Titels im Fenstermanager wird später das zugehörige Visualisierungsfenster automatisch in den Vordergrund geholt (unabhängig davon, ob es zuvor ggf. zum Symbol verkleinert worden war).

Der FAB-Fenstermanager kann bei Bedarf immer über allen anderen BORIS-Fensters gehalten werden. Klicken Sie dazu mit der rechten Maustaste in den Fenstermanager. Es erscheint ein Kontextmenü, das lediglich den Eintrag *Immer im Vordergrund* aufweist. Nach dem Aktivieren dieser Option verbleibt das Fenster immer in der obersten Ebene.

Fenstermanagement im Entwurfsmodus

Im Entwurfsmodus des FAB ist - unabhängig davon, ob er aus BORIS heraus oder als eigenständiges Hauptprogramm benutzt wird - eine ganze Reihe von Fenstern gleichzeitig sichtbar. Dies sind das eigentliche Visualisierungsfenster, der Konfigurierungsdialog, das Elementfenster und eventuell auch das I/O-Kontrollfenster. Beim Schließen des FAB wird die aktuelle Konfiguration (Größe und Position der Fenster) jeweils in der Windows-Registry gespeichert und steht dann beim nächsten Aufruf automatisch wieder zur Verfügung.

Je nach benutzter Bildschirmauflösung und aktueller Größe des FAB-Visualisierungsfensters kann es sinnvoll sein, zeitweise einige der Fenster zu schließen oder zum Symbol zu verkleinern. Um jederzeit schnell wieder auf das jeweilige Fenster zugreifen zu können, besitzt das Visualisierungsfenster im Entwurfsmodus ein Kontextmenü (Popup-Menü), das jederzeit über die rechte Maustaste aufgerufen werden kann. Über dieses Menü lassen sich einzelne Fenster oder auch sämtliche Fenster unabhängig von ihrem aktuellen Status unmittelbar wieder in den Vordergrund holen. Alternativ dazu kann ein zum Symbol verkleinerter Konfigurierungsdialog auch über die Schaltfläche *Konfigurierungsdialog anzeigen* des Elementfensters "wiederbelebt" werden.



Kontextmenü des Visualisierungsfensters

Laden und Speichern von Konfigurationen

Der gesamte Inhalt des Konfigurationsdialogs kann in einer FAB-Konfigurationsdatei (Extension .FAB) gespeichert und später bei Bedarf wieder von dort geladen werden. Zu diesem Zweck dienen die beiden Schaltflächen und der Toolbar des Dialogs.

Einfaches Anwendungsbeispiel: Inverses Doppelpendel

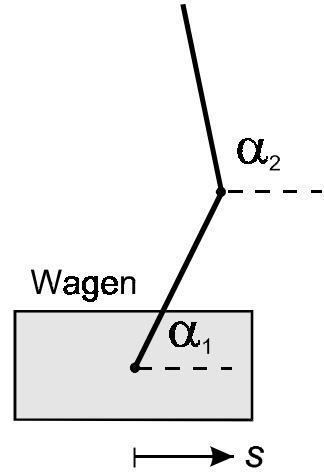
Nachfolgend soll anhand eines einfachen Anwendungsbeispiels die Vorgehensweise bei der Erstellung einer Systemvisualisierung mit dem *Flexible Ani-*

mation Builder erläutert werden. Das Beispiel befindet sich unter dem Namen DOUBLEPENDULUM.BSY im Lieferumfang.

Aufgabenstellung

Es soll eine Visualisierung für ein inverses (d. h. stehendes) Doppelpendel auf einem beweglichen Wagen erstellt werden (siehe nachfolgende Grafik).

Eingangsgrößen der Visualisierung sollen die Wagenposition *s*, der Ausschlag des unteren Stabs α_1 und der Ausschlag des oberen Stabs α_2 sein. Die Wagenposition soll sich im Bereich $0 < s < 10$m bewegen.



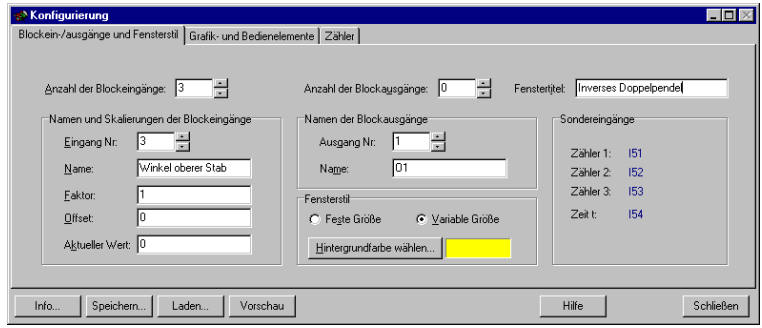
Inverses Doppelpendel

Spezifizierung der Blockeingänge

Nach dem Einfügen eines leeren FAB-Moduls werden zunächst die Blockeingänge konfiguriert. Dazu aktivieren Sie im FAB-Konfigurationsdialog die Palette *Blockeingänge und Fensterstil*. Nunmehr sind folgende Schritte zu bewerkstelligen:

1. Festlegung des Fenstertitels für das Visualisierungsfenster. Hier wird *Inverses Doppelpendel* gewählt.
2. Festlegung der Anzahl der Blockeingänge auf 3.
3. Benennung der drei Blockeingänge mit *Wagenposition*, *Winkel unterer Stab* und *Winkel oberer Stab*.
4. Wahl einer geeigneten Hintergrundfarbe für die Visualisierung.

Auf eine Skalierung der Eingänge wird hier verzichtet; sie wird später direkt in der Elementtabelle mit Hilfe des Formelparsers vorgenommen. Der Fensterstil bleibt zunächst auf variable Größe gesetzt; nach Fertigstellung der Visualisierung kann er dann später auf eine passende, feste Größe gesetzt werden. Die nachfolgende Bildschirmgrafik zeigt den Konfigurationsdialog nach Eingabe aller Daten.



Konfigurationsdialog nach Spezifizierung der Blockeingänge

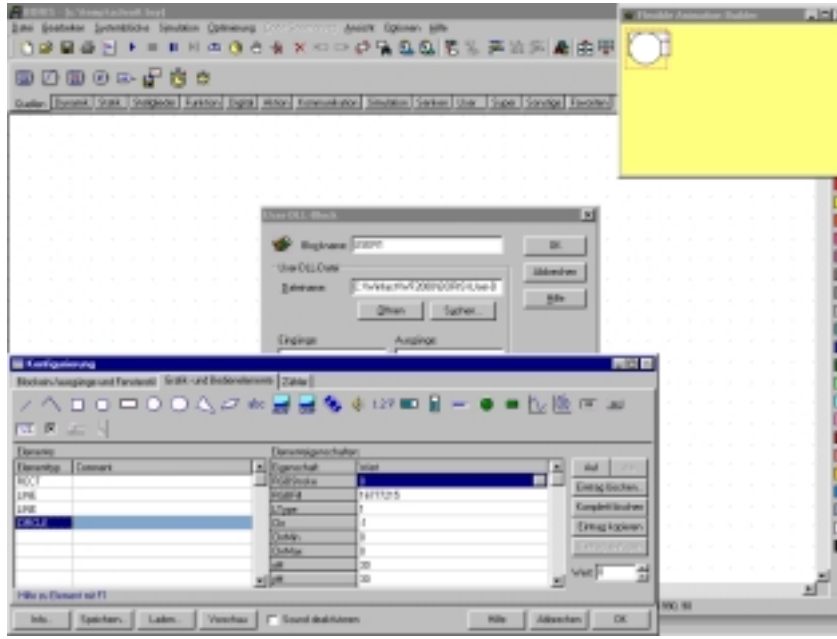
Einfügen der Grafikelemente

Nach Konfigurierung der Blockeingänge können die Grafikelemente eingefügt und konfiguriert werden. Es sind vier Grafikelemente erforderlich:

- Ein *RECT*-Element für den Wagen
- Zwei *LINE*-Elemente für den unteren und oberen Stab
- Ein *CIRCLE*-Element für das Gelenk zwischen den beiden Stäben

Diese Elemente werden zunächst mit ihren Vorgabewerten eingefügt. Es ergibt sich anschließend nach Betätigung der *Vorschau*-Taste der nachfolgend dargestellte Bildschirm.

In den nachfolgenden Schritten werden nun die Eigenschaften der einzelnen Elemente in der erforderlichen Weise gesetzt.



Bildschirm nach Einfügen der erforderlichen Grafikelemente mit ihren Vorgabewerten

Wagen

Der Wagen (*RECT*-Element) soll zunächst eine zwei Pixel breite Umrandung (Eigenschaft *LType* = 2) sowie eine braune Füllung (Eigenschaft *RGBFill*) erhalten. Die Größe soll 60x40 Pixel betragen (Eigenschaft $w = 60$, $h = 40$). Die y-Position (Eigenschaft *y*) legen wir auf 170 fest.

Die x-Position des Wagens (Eigenschaft *x*) muß nun mit Eingang *I1* (Wagenposition *s* in m) verknüpft werden. Dabei ist zu beachten, dass die Wagenposition *s* gemäß obiger Grafik über die Wagenmitte definiert ist, *x* aber über den linken Rand des Rechtecks. Ferner soll angenommen werden, dass die maximale

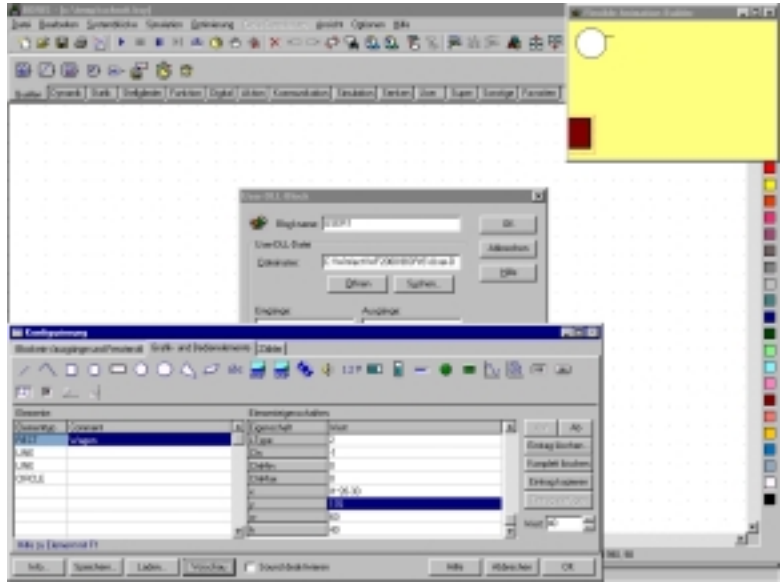
Auslenkung von s = 10m im Visualisierungsfenster einer Auslenkung von 200 Pixeln entspricht. Es gilt dann also für die Umrechnung

$$x = I1/10*200 - 30 = I1*20 - 30$$

Der Ausdruck "-30" entspricht dabei gerade der halben Wagenbreite. Für die Eigenschaft *x* des Wagens ist also der Ausdruck

$$I1*20-30$$

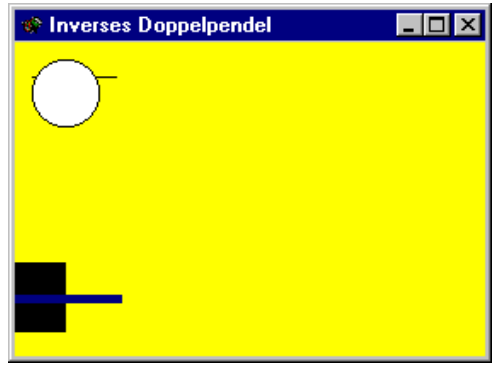
einzugeben. Die Vorschau nach Eingabe aller Daten ergibt nachfolgende Grafik:



Da der aktuelle Eingangswert *I1* noch 0 ist (es wurde noch keine Simulation gestartet), befindet sich die Wagenmitte genau auf Höhe des linken Fensterrandes. Um z. B. das Verhalten des Wagens für einen konkreten Wert von *I1* zu testen, kann man *I1* in obigem Ausdruck einfach kurzzeitig durch den entsprechenden Zahlenwert ersetzen und dann über die Vorschau-Funktion überprüfen. Alternativ dazu kann man den aktuellen Eingangswert auch über das Eingabefeld *Aktueller Wert* des Dialogregisters *Blockeingänge und Fensterstil* modifizieren.

Unterer Stab

Der untere Stab soll zunächst eine blaue Farbe (Eigenschaft *RGBStroke*) und eine Dicke von 5 Pixeln erhalten (Eigenschaft *LType* = 5). Die y-Koordinate muß auf der halben Höhe des Wagens liegen; es gilt also $y = 150$. Die x-Koordinate liegt in der Wagenmitte, entspricht also der umgerechneten Wagenposition *s*. Somit gilt $x = I1*20$ (s. o.). Die Länge des Stabs legen wir willkürlich auf 60 Pixel fest ($L = 60$). Der Winkel (Eigenschaft *Angle*) schließlich entspricht direkt dem zweiten Blockeingang; es gilt also *Angle* = *I2*. Nachfolgende Grafik zeigt das Visualisierungsfenster nach Eingabe aller Daten und Aktualisierung.



Visualisierungsfenster nach Parametrierung des unteren Stabs

Oberer Stab

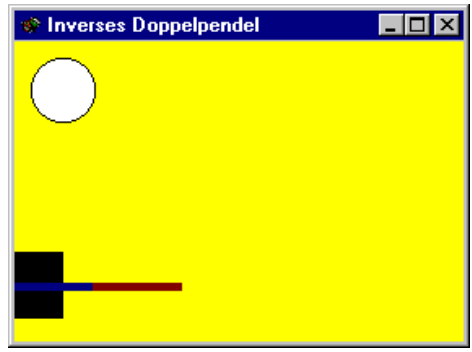
Der obere Stab soll eine rote Farbe (Eigenschaft *RGBStroke*), eine Dicke von 5 Pixeln (Eigenschaft *LType* = 5) und ebenfalls eine Länge von 50 Pixeln erhalten ($L = 50$). Die x-Koordinate ergibt sich aus der x-Koordinate des unteren Stabs, der Stablänge und dem Cosinus des Stabwinkels. Da für die x-Koordinate des unteren Stabs die Beziehung $x = I1*20$ gilt (s. o.), ergibt sich für die x-Koordinate des oberen Stabs

$$x = I1*20 + 50*\cos(I2)$$

Analog ergibt sich die y-Koordinate des oberen Stabs aus der y-Koordinate des unteren Stabs, der Stablänge und dem Sinus des Stabwinkels. Man erhält

$$y = 150 + 50*\sin(I2)$$

Der Winkel (Eigenschaft *Angle*) schließlich entspricht direkt dem dritten Blokkeingang; es gilt also *Angle = I3*. Nachfolgende Grafik zeigt das Visualisierungsfenster nach Eingabe aller Daten und Aktualisierung.



Visualisierungsfenster nach Parametrierung des oberen Stabs

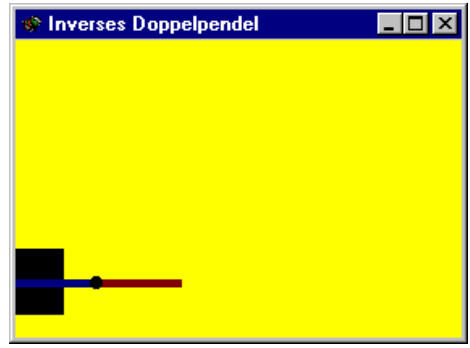
Gelenk

Abschließend bleibt noch das Gelenk (*CIRCLE*-Element) zu parametrieren. Es soll eine schwarze Füllung (Eigenschaft *RGBFill*) erhalten sowie einen Radius von 4 Pixeln aufweisen (Eigenschaft *r* = 4). Das Gelenk soll genau zwischen den beiden Stäben sitzen; seine x- und y-Koordinaten entsprechen also gerade denen des oberen Stabs:

$$x = I1*20 + 50*\cos(I2)$$

$$y = 150 + 50*\sin(I2)$$

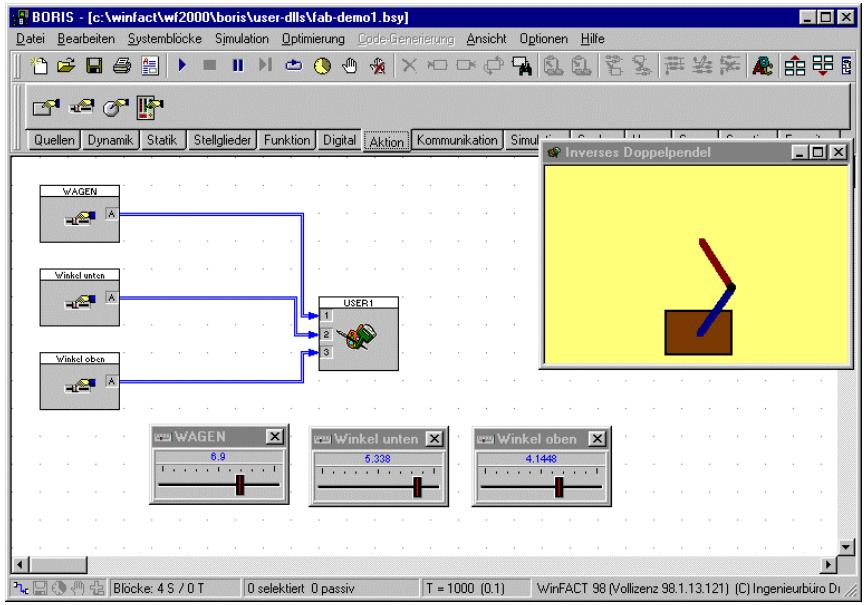
Nachfolgende Grafik zeigt das Visualisierungsfenster nach Eingabe aller Daten und Aktualisierung.



Visualisierungsfenster nach Parametrierung des Gelenks

Aufbau der Teststruktur

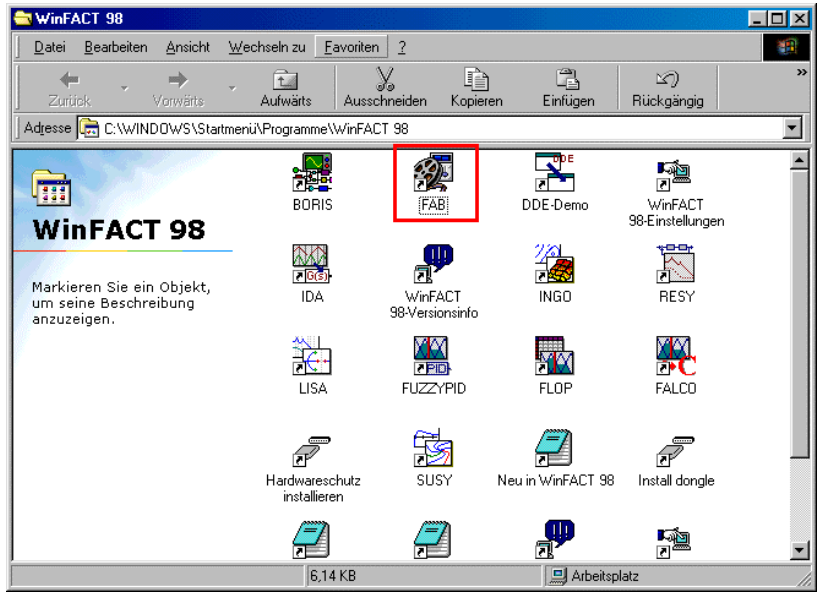
Zum Testen des gerade erstellten Visualisierungsmoduls speist man am besten alle Blockeingänge mit Potentiometer-Blöcken, um die einzelnen Funktionen dann interaktiv austesten zu können. Dabei ist die Verstärkung des Potis für Eingang 1 (Wagenposition) auf 10 (maximale Auslenkung des Wagens), die für die Eingänge 2 und 3 (Stabwinkel) jeweils auf 6.28 (entsprechend 2π) festzusetzen. Nachfolgende Bildschirmgrafik zeigt die komplette Teststruktur.



Inverses Doppelpendel mit Teststruktur

Arbeiten mit dem FAB-Hauptprogramm

Neben der integrierten Nutzung innerhalb von BORIS kann der *Flexible Animation Builder* auch als eigenständiges Hauptprogramm aus der WinFACT 98-Programmgruppe aufgerufen werden (siehe nachfolgende Bildschirmgrafik).

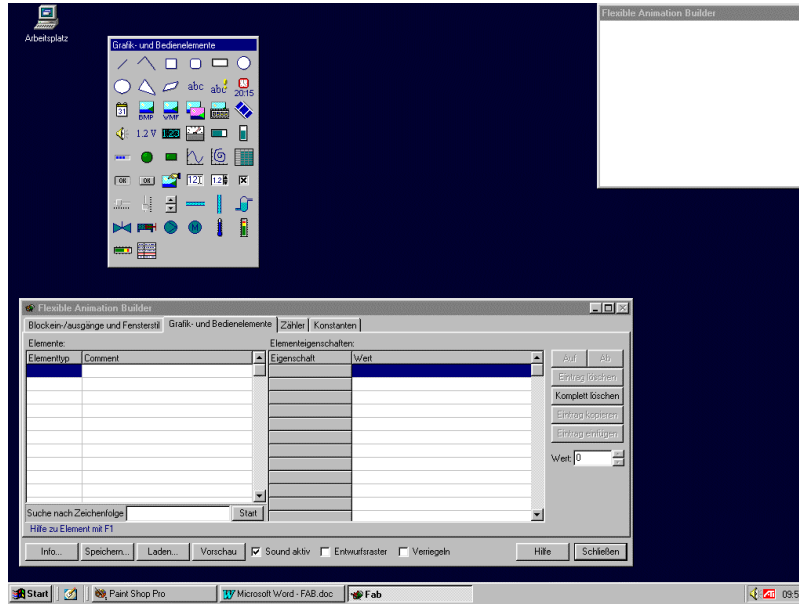


Der Flexible Animation Builder *in der WF98-Programmgruppe*

Nach dem Aufruf präsentiert sich FAB mit dem bekannten Visualisierungsfenster und dem Konfigurierungsdialog.

Die Bedienung läuft völlig analog zur in BORIS integrierten Version ab. Lediglich folgende Punkte sind zu beachten:

- Das Visualisierungsfenster hat in dieser Betriebsart kein Systemmenü. Zum Beenden des FAB dient die *Schließen*-Taste des Konfigurierungsdialogs. Die *Abbruch*-Taste fehlt in dieser Betriebsart.
- Da in dieser Betriebsart naturgemäß keine BSY-Datei mit den eingegebenen Daten verknüpft ist, müssen die Daten zwangsläufig vor dem Beenden in einer FAB-Datei gespeichert werden (Schaltfläche *Speichern*). Wird dies unterlassen, erfolgt vor dem Beenden eine automatische Sicherheitsabfrage.
- Der Eintrag im Feld *Anzahl der Blockeingänge* bzw. *Anzahl der Blockausgänge* ist ohne Bedeutung.



Bildschirmansicht nach Aufruf des FAB-Hauptprogramms

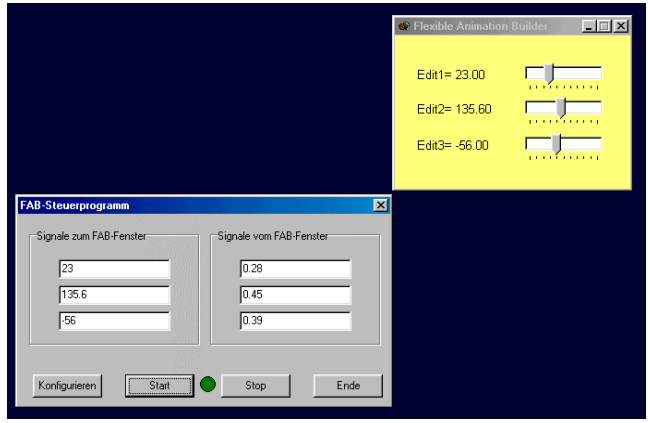
Nutzung des FAB aus anderen Applikationen

Den Kernel des *Flexible Animation Builders* bildet die Datei FAB.DLL. Diese DLL entspricht dem BORIS-User-DLL-Standard (siehe *WinFACT 98-Benutzerhandbuch*), d. h. sie weist sämtliche Schnittstellenfunktionen auf, die erforderlich sind, um einen Datenaustausch mit dem FAB (genauer gesagt, mit dem zugehörigen Visualisierungsfenster) zu ermöglichen. Daher kann der FAB auch völlig unabhängig von BORIS zur Erstellung von Prozessvisualisierungen, Animationen und Bedienoberflächen für andere Applikationen (z. B. anwenderprogrammierte Anwendungen) benutzt werden. Dazu müssen lediglich die entsprechenden Daten- und Funktionsschnittstellen gemäß BORIS-User-DLL-

Standard innerhalb des Anwenderprogramms definiert werden. Der Datenaustausch zwischen Anwendung und FAB-Visualisierungsfenster kann dann einfach durch Aufruf der Schnittstellenfunktion *SimulateDLL2* erfolgen.

Das Unterverzeichnis *\FAB-Control-Demo* enthält ein einfaches Beispielprogramm namens FABCONTROL.EXE, welches die Nutzung des FAB innerhalb anderer Applikationen demonstriert. Um das Programm zu testen, gehen Sie wie folgt vor:

1. Starten Sie das Programm FABCONTROL.EXE. Es erscheinen das Anwendungs-Hauptfenster und das (zunächst noch leere) FAB-Visualisierungsfenster.
2. Betätigen Sie die Schaltfläche *Konfigurieren*. Es erscheint der bekannte Konfigurierungsdialog des FAB.
3. Laden Sie über die Schaltfläche *Laden...* die ebenfalls im Unterverzeichnis *\FAB-Control-Demo* befindliche Demo-Datei DEMO.FAB und verlassen Sie den Konfigurierungsdialog. Im Visualisierungsfenster erscheinen nun die eingelesenen Elemente.
4. Starten Sie den Datenaustausch über die *Start*-Schaltfläche. Die über die Schieberegler im Visualisierungsfenster vorgegebenen Werte erscheinen nun im rechten Teil des Hauptfensters, während die im linken Teil des Hauptfensters eingegebenen Werte im linken Teil des Visualisierungsfensters angezeigt werden. Die Datenübergabe geschieht in diesem Demo-Programm timergesteuert zyklisch alle 100 ms.



Das FAB-Control-Demoprogramm nach Einlesen von DEMO.FAB.

Das komplette DELPHI 3-Projekt (incl Quelltexte) befindet sich ebenfalls im Unterverzeichnis *\FAB-Control-Demo.*Nachfolgendes Listing zeigt den Quelltext des Programm-Hauptfensters. Auf völlig analoge Weise kann der FAB auch unter anderen Entwicklungsumgebungen (z. B. Visual C++ oder Visual Basic) eingebunden werden.

```
unit FabControlForm;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, ExtCtrls;

type

  // Datentypen zur Bedienung der BORIS-User-DLL-Schnittstelle
  // (werden hier nicht alle benötigt!)

  PParameterStruct=^TParameterStruct;
  TParameterStruct=packed record
    NuE : Byte;
    NuI : Byte;
    NuB : Byte;
    E:Array[0..31] of Extended;
    I:Array[0..31] of Integer;
    B:Array[0..31] of Byte;
    D:Array[0..255] of char;
    EMin:Array[0..31] of Extended;
    EMax:Array[0..31] of Extended;
    IMin:Array[0..31] of Integer;
    IMax:Array[0..31] of Integer;
    NaE : Array[0..31,0..40] of char;
    NaI : Array[0..31,0..40] of char;
    NaB : Array[0..31,0..40] of char;
    UserDataPtr: TForm;
    ParentPtr: Pointer;
    ParentHWnd: HWnd; {Handle des Elternfensters}
    ParentName: PChar;
    UserHWindow: HWnd;
    DataFile: text;
  end;

  PDialogEnableStruct=^TDialogEnableStruct;
  TDialogEnablestruct=packed record
    AllowE: Longint;
    AllowI: Longint;
    AllowB: Longint;
    AllowD: Byte;
  end;

  PNumberOfInputsOutputs=^TNumberOfInputsOutputs;
  TNumberOfInputsOutputs=packed record
    Inputs :Byte;                          {Anzahl Eingänge}
    Outputs:Byte;                          {Anzahl Ausgänge}
```

```
  NameI : Array[0..49,0..40] of char;
  NameO : Array[0..49,0..40] of char;
end;

PInputArray = ^TInputArray;
TInputArray = packed array[1..50] of extended;
POutputArray = ^TOutputArray;
TOutputArray = packed array[1..50] of extended;


// Hauptformular der Anwendung

TForm1 = class(TForm)
  Button1: TButton;
  GroupBox1: TGroupBox;
  GroupBox2: TGroupBox;
  Edit1: TEdit;
  Edit2: TEdit;
  Edit3: TEdit;
  Edit4: TEdit;
  Edit5: TEdit;
  Edit6: TEdit;
  Timer1: TTimer;
  Button2: TButton;
  Button3: TButton;
  Button4: TButton;
  Shape1: TShape;
  procedure Button1Click(Sender: TObject);
  procedure FormCreate(Sender: TObject);
  procedure FormDestroy(Sender: TObject);
  procedure Button2Click(Sender: TObject);
  procedure Button3Click(Sender: TObject);
  procedure Button4Click(Sender: TObject);
  procedure Timer1Timer(Sender: TObject);
private
  { Private-Deklarationen }
  ParameterStruct: TParameterStruct;
  NumberOfInputsOutputs: TNumberOfInputsOutputs;
  Inputs: TInputArray;
  Outputs: TOutputArray;
public
  { Public-Deklarationen }
end;

// Einbinden der Schnittstellenfunktionen zur FAB.DLL
// (werden in dieser Demo nicht alle wirklich benötigt!)

function  CanSimulateDLL(D: PParameterStruct):integer; stdcall;
  external 'FAB.DLL';
procedure GetParameterStruct(D:PParameterStruct);export stdcall;
  external 'FAB.DLL';
procedure GetDialogEnableStruct(D:PDialogEnableStruct;
  D2:PParameterStruct); export stdcall; external 'FAB.DLL';
procedure GetNumberOfInputsOutputs2(D:PNumberofInputsOutputs;
  Parameterfilename: PChar; UserDataPtr: Pointer;
  UserHWindow: Pointer);export stdcall; external 'FAB.DLL';
```

```
  procedure CallParameterDialogDLL(D1: PParameterStruct;
    D2: PNumberOfInputsOutputs); export stdcall;
    external 'FAB.DLL';
  procedure SimulateDLL(T:Extended;D:PParameterStruct;
    Inputs:PInputArray;Outputs:POutputArray);export stdcall;
    external 'FAB.DLL';
  procedure InitSimulationDLL(D:PParameterStruct;
    Inputs:PInputArray; Outputs:POutputArray);
    export stdcall; external 'FAB.DLL';
  procedure EndSimulationDLL2(D:PParameterStruct);export stdcall;
    external 'FAB.DLL';
  procedure InitUserDLL(D: PParameterStruct); export stdcall;
    external 'FAB.DLL';
  procedure DisposeUserDLL(D: PParameterStruct); export stdcall;
    external 'FAB.DLL';
  function  GetDLLName: PChar; export stdcall; external 'FAB.DLL';
  procedure ShowWindowDLL(D: PParameterstruct); export stdcall;
    external 'FAB.DLL';
  procedure HideWindowDLL(D: PParameterstruct); export stdcall;
    external 'FAB.DLL';
  procedure WriteToFile(AFileHandle:word; D: PParameterStruct);
    export stdcall; external 'FAB.DLL';
  procedure ReadFromFile(AFileHandle:word; D: PParameterStruct);
    export stdcall; external 'FAB.DLL';

var
  Form1: TForm1;

implementation

{$R *.DFM}

procedure TForm1.Button1Click(Sender: TObject);
begin
  Close;
end;

procedure TForm1.FormCreate(Sender: TObject);
begin
  // Initialisierung der notwendigen Records von ParameterStruct
  with ParameterStruct do begin
    ParentHWnd := Handle;  // Fensterhandle des Hauptformulars
  end;
  InitUserDLL(@ParameterStruct);
  DecimalSeparator := '.'; // Punkt als Dezimaltrenner
end;

procedure TForm1.FormDestroy(Sender: TObject);
begin
  DisposeUserDLL(@ParameterStruct);// FAB-Fenster wieder freigeben
end;

procedure TForm1.Button2Click(Sender: TObject);
begin
  // Konfigurierungsdialog des FAB aufrufen
  CallParameterDialogDLL(@ParameterStruct,@NumberOfInputsOutputs);
```

```
end;

procedure TForm1.Button3Click(Sender: TObject);
begin
  Timer1.Enabled := true;
end;

procedure TForm1.Button4Click(Sender: TObject);
begin
  Timer1.Enabled := false;
  Shape1.Brush.Color := clGreen;
end;

procedure TForm1.Timer1Timer(Sender: TObject);
begin
  if Shape1.Brush.Color = clGreen then
    Shape1.Brush.Color := clLime
  else
    Shape1.Brush.Color := clGreen;
  // Input-Vektor mit aktuellen Werten füllen
  try
    Inputs[1] := StrToFloat(Edit1.Text);
  except
  end;
  try
    Inputs[2] := StrToFloat(Edit2.Text);
  except
  end;
  try
    Inputs[3] := StrToFloat(Edit3.Text);
  except
  end;
  // Simulationsroutine des FAB aufrufen
  SimulateDLL(0, @ParameterStruct, @Inputs, @Outputs);
  // Output-Vektor auslesen
  Edit4.Text := FloatToStr(Outputs[1]);
  Edit5.Text := FloatToStr(Outputs[2]);
  Edit6.Text := FloatToStr(Outputs[3]);
end;

end.
```

Listing von FABCONTROLFORM.PAS