



# **BrbLibVc4 V5.04**

## **Dokumentation**

**B&R übernimmt keine Haftung für Folgen, die durch die Implementierung sowie die Benutzung dieser Software entstehen!**

Inhaltliche Änderungen dieses Dokuments behalten wir uns ohne Ankündigung vor. B&R haftet nicht für technische oder drucktechnische Fehler und Mängel in diesem Dokument. Außerdem übernimmt B&R keine Haftung für Schäden, die direkt oder indirekt auf Lieferung, Leistung und Nutzung dieses Materials zurückzuführen sind. Wir weisen darauf hin, dass die in diesem Dokument verwendeten Soft- und Hardwarebezeichnungen und Markennamen der jeweiligen Firmen dem allgemeinen warenzeichen-, marken- oder patentrechtlichen Schutz unterliegen.



## Inhaltsverzeichnis

Inhaltsverzeichnis.....	2
<b>1 Allgemeines .....</b>	<b>6</b>
1.1 Hinweise zum Compiler .....	6
1.2 Abhängigkeiten .....	6
1.3 Hinweise zu StructuredText und anderen IEC-Sprachen.....	6
1.4 Geprüft mit ClangTidy .....	6
1.5 Quellcode und Binär-Variante der Bibliothek.....	6
1.5.1 Quellcode-Variante .....	6
1.5.2 Binär-Variante .....	7
1.6 Neueste Versionen auf GitHub .....	7
<b>2 Revisionsgeschichte .....</b>	<b>8</b>
2.1 BrbLibVc4 V5.04 – 2024-09-09.....	8
2.1.1 Portierung auf neuere Versionen .....	8
2.1.2 Änderung der HW-Konfigurationen .....	8
2.1.3 Entfernung der Binär-Variante aus dem Release.....	8
2.1.4 Vorbereitungen für AS6.00.....	8
2.1.4.1 Include geändert .....	8
<b>3 Pakete .....</b>	<b>9</b>
3.1 General .....	9
3.1.1 BrbVc4General .....	9
3.1.1.1 Struktur.....	9
3.1.1.2 BrbVc4HandleGeneral .....	9
3.1.2 BrbVc4PageHandling.....	10
3.1.2.1 Struktur.....	10
3.1.2.2 BrbVc4HandleChangePage .....	11
3.1.2.3 BrbVc4ChangePage.....	11
3.1.2.4 BrbVc4ChangePageBack.....	12
3.1.2.5 BrbVc4HandleScreenSaver .....	12
3.2 Controls .....	14
3.2.1 BrbVc4ControlStatusHandling .....	14
3.2.1.1 BrbVc4SetControlEnability .....	14
3.2.1.2 BrbVc4IsControlEnabled .....	14
3.2.1.3 BrbVc4SetControlVisibility.....	14
3.2.1.4 BrbVc4IsControlVisible.....	15
3.2.1.5 BrbVc4SetControlFocus.....	15
3.2.1.6 BrbVc4HasControlFocus.....	15
3.2.1.7 BrbVc4IsControlInputActive .....	15
3.2.1.8 BrbVc4OpenTouchpad .....	16
3.2.1.9 BrbVc4CloseTouchpad .....	16
3.2.1.10 BrbVc4IsTouchpadOpen .....	16
3.2.1.11 BrbVc4SetControlColor .....	16
3.2.2 Bitmap-Animation.....	17
3.2.2.1 Struktur.....	17
3.2.2.2 BrbVc4HandleAnimation .....	17
3.2.3 Bargraph .....	17
3.2.3.1 Struktur.....	17
3.2.4 Bitmap .....	17
3.2.4.1 Struktur.....	18
3.2.5 Normaler Button .....	18
3.2.5.1 Struktur.....	18
3.2.5.2 BrbVc4HandleButton.....	18

3.2.6 Checkbox .....	18
3.2.6.1 Struktur.....	19
3.2.6.2 BrbVc4HandleCheckbox .....	19
3.2.7 CheckboxButton.....	19
3.2.7.1 Struktur.....	19
3.2.7.2 BrbVc4HandleCheckboxButton .....	20
3.2.8 DateTime .....	20
3.2.8.1 Struktur.....	20
3.2.9 Drawbox.....	20
3.2.9.1 Struktur.....	20
3.2.10 Dropdown.....	21
3.2.10.1 Struktur.....	21
3.2.10.2 BrbVc4HandleDropdown .....	21
3.2.11 Edit.....	21
3.2.11.1 Struktur.....	22
3.2.12 Gauge .....	22
3.2.12.1 Struktur.....	22
3.2.13 Hotspot.....	22
3.2.13.1 Struktur.....	22
3.2.14 Html.....	22
3.2.14.1 Struktur.....	23
3.2.15 HwPosSwitch2 .....	23
3.2.15.1 Struktur.....	23
3.2.15.2 BrbVc4HandleHwPosSwitch2 .....	23
3.2.16 HwSafetyButton .....	24
3.2.16.1 Struktur.....	24
3.2.16.2 BrbVc4HandleHwSafetyButton .....	24
3.2.17 IncButton .....	25
3.2.17.1 Struktur.....	25
3.2.17.2 BrbVc4HandleIncButton .....	25
3.2.18 JogButton .....	25
3.2.18.1 Struktur.....	26
3.2.18.2 BrbVc4HandleJogButton .....	26
3.2.19 Layer .....	26
3.2.19.1 Struktur.....	27
3.2.20 Listbox.....	27
3.2.20.1 Struktur.....	27
3.2.21 Numeric.....	27
3.2.21.1 Struktur.....	27
3.2.21.2 BrbVc4HandleNumericInput .....	27
3.2.22 NumericEx .....	28
3.2.22.1 Struktur.....	28
3.2.22.2 BrbVc4HandleNumericInputEx .....	28
3.2.23 Optionbox.....	29
3.2.23.1 Struktur.....	29
3.2.23.2 BrbVc4HandleOptionbox .....	29
3.2.24 OptionboxButton .....	30
3.2.24.1 Struktur.....	31
3.2.24.2 BrbVc4HandleOptionboxButton .....	31
3.2.25 Password .....	32
3.2.25.1 Struktur.....	32
3.2.26 PieChart .....	32
3.2.26.1 Struktur.....	32
3.2.27 Scale.....	32
3.2.27.1 Struktur.....	32
3.2.28 Scroll-Listen .....	32
3.2.29 ScrollbarHorizontal.....	33
3.2.29.1 Struktur.....	33
3.2.29.2 BrbVc4HandleScrollbarHorizontal .....	33
3.2.30 ScrollbarVertical .....	34
3.2.30.1 Struktur.....	34
3.2.30.2 BrbVc4HandleScrollbarVertical .....	34

3.2.31 Shape.....	35
3.2.31.1 Struktur.....	35
3.2.32 Slider.....	35
3.2.32.1 Struktur.....	35
3.2.33 String.....	36
3.2.33.1 Struktur.....	36
3.2.34 Tab-Control.....	36
3.2.34.1 Struktur.....	36
3.2.34.2 BrbVc4HandleTabCtrl .....	36
3.2.34.3 BrbVc4SetTabPageInvisible .....	37
3.2.35 Text.....	37
3.2.35.1 Struktur.....	37
3.2.36 ToggleButton.....	37
3.2.36.1 Struktur.....	37
3.2.36.2 BrbVc4HandleToggleButton .....	37
3.2.37 ToggleButton Ext (erweitert) .....	38
3.2.37.1 Struktur.....	38
3.2.37.2 BrbVc4HandleToggleButton .....	38
3.2.37.3 Unterschied zum normalen Toggle-Button .....	39
3.2.38 Touchgrid .....	39
3.2.38.1 Struktur.....	40
3.2.38.2 BrbVc4HandleTouchgrid .....	40
3.3 Draw .....	42
3.3.1 Linie .....	42
3.3.1.1 Struktur.....	42
3.3.1.2 BrbVc4DrawLine .....	42
3.3.1.3 BrbVc4DrawLineCorr .....	42
3.3.1.4 BrbVc4DrawLineClip .....	43
3.3.2 Rechteck .....	43
3.3.2.1 Struktur.....	43
3.3.2.2 BrbVc4DrawRectangle .....	43
3.3.2.3 BrbVc4DrawRectangleCorr .....	43
3.3.2.4 BrbVc4DrawRectangleClip.....	44
3.3.3 Ellipse .....	44
3.3.3.1 Struktur.....	44
3.3.3.2 BrbVc4DrawEllipse.....	44
3.3.4 Arc.....	45
3.3.4.1 Struktur.....	45
3.3.4.2 BrbVc4DrawArc.....	45
3.3.5 Text.....	45
3.3.5.1 Struktur.....	45
3.3.5.2 BrbVc4DrawText .....	46
3.3.6 Font.....	46
3.3.6.1 Struktur.....	46
3.3.7 Hilfsfunktionen .....	46
3.3.7.1 BrbVc4CorrectLine .....	46
3.3.7.2 BrbVc4ClipLine .....	46
3.3.7.3 BrbVc4CorrectRectangle.....	47
3.3.7.4 BrbVc4ClipRectangle .....	47
3.3.7.5 BrbVc4IsPointWithinRectangle .....	47
3.4 DrawExt.....	48
3.4.1 Trend.....	48
3.4.1.1 Struktur.....	48
3.4.1.2 Konfiguration .....	49
3.4.1.2.1 Allgemeines .....	49
3.4.1.2.2 ScaleY – Werte-Skalen .....	49
3.4.1.2.3 SourceBuffer und nSourceArrayIndexMax .....	49
3.4.1.2.4 ScaleX – Zeit-Skala .....	50
3.4.1.2.5 TouchAction – Funktion des Touchs .....	51
3.4.1.2.6 Curve – Kurven .....	52
3.4.1.2.7 Cursor .....	53
3.4.1.2.8 Callbacks.....	53

3.4.1.2.9 pTag .....	54
3.4.1.3 Status .....	55
3.4.1.4 Intern .....	55
3.4.1.5 BrbVc4DrawTrend .....	55
3.4.1.6 BrbVc4GetTrendDisplayCoordinateY .....	55
3.4.1.7 BrbVc4GetTrendDisplayCoordinateX .....	56
3.4.1.8 BrbVc4GetTrendSampleIndexByTime .....	56
3.4.1.9 BrbVc4GetTrendDisplayCoordXByTime .....	56
3.4.1.10 BrbVc4GetTrendTimestampByIndex .....	57
3.4.2 TrendLink .....	58
3.4.2.1 Struktur .....	58
3.4.2.2 Konfiguration .....	58
3.4.2.3 BrbVc4LinkTrends .....	58
3.4.3 XY-Plot .....	59
3.4.3.1 Struktur .....	59
3.4.3.2 Konfiguration .....	60
3.4.3.2.1 Allgemeines .....	60
3.4.3.2.2 ScaleY .....	60
3.4.3.2.3 ScaleX .....	61
3.4.3.2.4 TouchAction – Funktion des Touchs .....	61
3.4.3.2.5 Curve – Kurven .....	61
3.4.3.2.6 Cursor .....	62
3.4.3.2.7 Callbacks .....	63
3.4.3.2.8 pTag .....	65
3.4.3.3 Status .....	65
3.4.3.4 Intern .....	65
3.4.3.5 BrbVc4DrawPlot .....	65
3.4.3.6 BrbVc4GetPlotDisplayCoordinateY .....	66
3.4.3.7 BrbVc4GetPlotDisplayCoordinateX .....	66
3.4.4 Achse linear darstellen .....	67
3.4.4.1 Struktur .....	67
3.4.4.2 BrbVc4DrawAxisLinear .....	67
3.4.5 Achse radial darstellen .....	69
3.4.5.1 Struktur .....	69
3.4.5.2 BrbVc4DrawAxisRadial .....	69
3.4.6 Treeview .....	71
3.4.6.1 Struktur .....	71
3.4.6.2 Konfiguration .....	72
3.4.6.2.1 Allgemeines .....	72
3.4.6.2.2 pSourceNodeList und nSourceArrayIndexMax .....	72
3.4.6.2.3 pInternNodeList .....	73
3.4.6.2.4 Nodes .....	73
3.4.6.2.5 Korrigieren des ScrollOffsetY .....	75
3.4.6.2.6 Scrollbar .....	75
3.4.6.2.7 TouchAction – Funktion des Touchs .....	76
3.4.6.2.8 Callbacks .....	77
3.4.6.2.9 pTag .....	78
3.4.6.3 Steuerung .....	78
3.4.6.4 Status .....	78
3.4.6.5 Intern .....	79
3.4.6.6 BrbVc4DrawTreeview .....	79
3.4.6.7 BrbVc4GetTreeviewInternNodeIndex .....	79

## 1 Allgemeines

Die Bibliothek „BrbLibVc4“ enthält viele nützliche Funktionen für eine Vc4-Visualisierung. Damit können Projekte übersichtlich und transparenter gestaltet werden.

**Diese Bibliothek ist keine offizielle B&R-Software. Es besteht kein Anspruch auf Support, Wartung oder Fehlerbehebung. Die Benutzung geschieht auf eigene Gefahr.**

Die Bibliothek unterliegt der MIT-Lizenz (siehe ‚License.txt‘), welche zwar unbeschränkte Nutzung auf eigene Gefahr gewährt, jedoch alle Haftungsansprüche ausschließt.

### 1.1 Hinweise zum Compiler

Das Entwicklungs- und Demo-Projekt ist auf den Compiler V6.3.0 gesetzt, mit dem das Projekt und damit auch die Bibliothek fehler- und warnungslos kompiliert werden können.

Die Bibliothek ist aber auch unter älteren Compiler-Versionen einsetzbar.

### 1.2 Abhängigkeiten

Es besteht eine Abhängigkeit von folgenden Bibliotheken:

- BrbLib V5.04
- VisApi

### 1.3 Hinweise zu StructuredText und anderen IEC-Sprachen

Die Bibliothek ist in ANSI-C geschrieben, kann aber auch in StructuredText und allen anderen IEC-Sprachen verwendet werden.

Einschränkung:

Bei manchen Funktionsblöcken sind optional über sogenannte Funktionszeiger benutzerdefinierte Erweiterungen implementiert. Beispiel: Beim FB ‚BrbVc4DrawTrend‘ kann der Anwender den gezeichneten Trend um eigene Zeichnungen erweitern.

Da die IEC-Sprachen keine Funktionszeiger unterstützen, sind diese Erweiterungen nur in ANSI-C nutzbar. Die entsprechenden Eingänge des FB's für die Funktionszeiger müssen in IEC-Sprachen auf 0 gesetzt werden. Ansonsten können auch diese FB's ohne Probleme verwendet werden.

### 1.4 Geprüft mit ClangTidy

Das gesamte Entwicklungs- und Demo-Projekt wurde mit dem Code-Analyse-Tool ClangTidy geprüft (Details siehe Dokumentation der Basis-Bibliothek „BrbLib“).

### 1.5 Quellcode und Binär-Variante der Bibliothek

Die Binär-Variante der Bibliothek ist ab V5.04 nicht mehr im Release enthalten, weil es zu viele Kombinationen (Zielsystem SG4 oder SGC, Prozessor Intel oder ARM, eingestellte Compiler-Version usw.) gibt, die Einfluss auf das Kompilat haben.

Außerdem ist es für den Anwender sehr leicht möglich, die benötigte Binär-Variante selbst zu erstellen (siehe AS-Hilfe GUID d750bdd3-0aad-4486-8c0d-4eb43372b325).

Welche Variante der Anwender in seinem Projekt verwende, sollte von diesen Punkten abhängig gemacht werden:

#### 1.5.1 Quellcode-Variante

Sie enthält den kompletten Quellcode aller Funktionen in ANSI-C. Somit kann der Anwender diesen studieren und unter Umständen eine ähnliche/abgewandelte Funktion sehr leicht in einer eigenen Bibliothek implementieren. Auch das Online-Debuggen durch Breakpoints ist möglich.

Beim Rebuild wird allerdings auch diese Bibliothek nochmals kompiliert. Dies kann je nach verwendetem Rechner einige Zeit in Anspruch nehmen.

Hinweis: Von der Änderung der Funktionen in der ausgelieferten Bibliothek wird abgeraten, da dann ein Umstieg auf eine neuere Version schwierig bis unmöglich wird.

### 1.5.2 Binär-Variante

Sie enthält nur vorkompilierte Module der Bibliothek für einen bestimmten Prozessor und Compiler. Es ist also kein Quellcode enthalten. Der Vorteil besteht darin, dass die Bibliothek auch bei einem Rebuild nicht mehr kompiliert werden muss. Dies bedeutet unter Umständen einen großen Zeitvorteil.

**Achtung:** Bei einer für ARM-Prozessoren exportierten Binär-Bibliothek kann nicht in den ArSim-Modus geschaltet werden, da ArSim wiederum die Intel-Version benötigt. Soll ArSim verfügbar sein, muss die Quellcode-Variante der Bibliothek eingefügt werden, denn nur dann kann sie je nach Prozessor kompiliert werden.

## 1.6 Neueste Versionen auf GitHub

GitHub ist eine öffentliche Plattform für kostenlose Software. Der Download ist ohne Anmeldung möglich. Darauf sind verschiedene Pakete des Autors kostenlos erhältlich. Sie unterliegen alle der MIT-Lizenz (siehe oben).

Die Bibliothek BrbLibVc4 und dessen unterlagerte Bibliothek BrbLib ist als eigenes Release-Paket erhältlich. Es enthält die Bibliotheken u.a. in Sourcecode- und Binär-Version:

<https://github.com/br-automation-com/BrbLibs-lib-src/releases>

Auch erhältlich ist das Windows-Tool ‚RnCommTest‘ zum Testen von Kommunikationen. Es enthält u.a. folgende Module:

- Serielle Kommunikation (RS232/485)
- Tcp-Client, Tcp-Server
- Udp
- ModbusTcp-Master, ModbusTcp-Client
- OpcUa-Client, OpcUa-Server, OpcUa-Subscriber

Es ist unter diesem Link erhältlich:

<https://github.com/br-automation-com/RnCommTest-Windows/releases>

Außerdem gibt es ein Beispiel-Projekt für OpcUa inklusive der Bibliothek BrbLibUa:

<https://github.com/br-automation-com/OpcUaSamples-sample-AS/releases>

## 2 Revisionsgeschichte

Ab V5.01 ist hier nur die letzte Version erwähnt. Die gesamte Revisionsgeschichte wurde in die Datei „BrbLib Revisionsgeschichte“ ausgelagert.

### 2.1 BrbLibVc4 V5.04 – 2024-09-09

#### 2.1.1 Portierung auf neuere Versionen

Beim Entwicklungs-Projekt wurden einige Versionen hochgezogen:

	Alte Version	Neue Version
Automation Studio	4.9.5.36	4.12.5.95
Automation Runtime	l4.90	l4.93
VC4	4.72.9	4.73.1

Die Bibliothek sollte trotz der Portierung immer noch unter kleineren und größeren Versionen kompiliert und eingesetzt werden können.

#### 2.1.2 Änderung der HW-Konfigurationen

Im Entwicklungs-Projekt wurde die HW-Konfiguration CP3586 entfernt, dafür wurde die Konfiguration CP3687X eingefügt.

#### 2.1.3 Entfernung der Binär-Variante aus dem Release

Die Binär-Variante der Bibliothek ist ab dieser Version nicht mehr im Release enthalten. Siehe [Quellcode und Binär-Variante der Bibliothek](#).

#### 2.1.4 Vorbereitungen für AS6.00

Die Bibliothek wird es auch für Automation Studio AS6.00 und folgend geben. Als Vorbereitung dazu wurden jetzt schon einige Änderungen gemacht.

##### 2.1.4.1 Include geändert

In den ersten AS-Versionen konnte die Header-Datei für Standard-AnsiC-Funktionen `stdlib.h` nicht original inkludiert werden, weil es damals Kollisionen mit B&R-System-Bibliotheken gab. Dazu wurde sie leicht abgeändert und unter dem Namen `AnsiCFunc.h` inkludiert.

Dies ist jetzt nicht mehr notwendig. Daher wurde die abgeänderte Version entfernt und die Original-Datei inkludiert.

Dies wirkt sich nicht auf die Funktionalitäten aus.



## 3 Pakete

### 3.1 General

In diesem Paket finden sich Struktur-Deklarationen und Funktionen zur schnellen und transparenten Programmierung von Vc4-Visualisierungen. Das Konzept sieht vor, die gesamte Logik einer Visualisierung in einem Task zu implementieren.

#### 3.1.1 BrbVc4General

Hiermit werden allgemein nützliche Datenpunkte und Funktionen zur Visualisierung angeboten.

##### 3.1.1.1 Struktur

BrbVc4General_TYP		
sVisName	STRING[nBRBVC4_VIS_NAME_CHAR_MAX]	<input type="checkbox"/> Eingang: Name des Vc-Objekts
tDoubleClickDelay	TIME	<input type="checkbox"/> Eingang: Zeit zwischen Loslassen und erneutem Drücken in [ms] zum Erkennen eines Doppelklicks
nDoubleClickDrift	UDINT	<input type="checkbox"/> Eingang: Max. Abstand zweier Klicks in [Pixel] zum Erkennen eines Doppelklicks
nRedrawCycles	UINT	<input type="checkbox"/> Eingang: Anzahl der Zyklen für das Zeichnen
bDisableRedraw	BOOL	<input type="checkbox"/> Eingang: 1=Löschen von eigenen Zeichnungen unterdrücken
nLanguageCurrent	UINT	<input type="checkbox"/> Zur Anbindung an den Datenpunkt
nLanguageChange	UINT	<input type="checkbox"/> Zur Anbindung an den Datenpunkt
nLifeSign	UDINT	<input type="checkbox"/> Zur Anbindung an den Datenpunkt
nVcHandle	UDINT	<input type="checkbox"/> Ausgang: VcHandle der Visualisierung
Touch	BrbVc4GeneralTouch_TYP	<input type="checkbox"/> Aktuelle Touch-Daten
nRedrawCounter	UINT	<input type="checkbox"/> Ausgang: Zähler fürs Zeichnen (bei = 0 darf gezeichnet werden)
dtCurrentTime	DATE_AND_TIME	<input type="checkbox"/> Ausgang: Aktueller Zeitstempel

Das Item „sVisName“ muss im Init mit dem Namen des Vc-Objekts belegt werden. Achtung: Es ist der abgekürzte Name zu verwenden.

Die Informationen dieser Struktur werden unter anderem auch von erweiterten Controls (z.B. Trend, siehe unten) verwendet.

##### 3.1.1.2 BrbVc4HandleGeneral

```
plcbit BrbVc4HandleGeneral(struct BrbVc4General_TYP* pVisGeneral)
```

###### Argumente:

`struct BrbVc4General_TYP* pVisGeneral`  
Zeiger auf eine Instanz von "BrbVc4General\_TYP"

###### Rückgabe:

`BOOL`

Immer 0

###### Beschreibung:

Diese Funktion sollte zyklisch vor der Seitenbearbeitung im Visualisierungs-Task aufgerufen werden.

Sie erfüllt mehrere Funktionalitäten:

Sie ermittelt einmalig das „nVcHandle“, welches für den programmseitigen Zugriff auf die Visualisierung benötigt wird.

Sie liefert zyklisch die zuletzt ausgeführte Touch-Aktion des Benutzers, welche unter anderem in eigenen, komplexeren, zusammengestellten Controls verwendet wird.

Der Touch wird in einer Unterstruktur veröffentlicht:

BrbVc4GeneralTouch_TYP		
nX	UDINT	<input type="checkbox"/> Ausgang: X-Koordinate in [Pixel]
nY	UDINT	<input type="checkbox"/> Ausgang: Y-Koordinate in [Pixel]
eState	BrbVc4TouchStates_ENUM	<input type="checkbox"/> Ausgang: Momentaner Status des Touchs
eStateSync	BrbVc4TouchStates_ENUM	<input type="checkbox"/> Ausgang: Mit Redraw-Counter synchronisierter Status des Touchs
dtLastTouchAction	DATE_AND_TIME	<input type="checkbox"/> Ausgang: Zeitstempel der letzten Touchaktion
nTimeSinceLastTouchActi...	UDINT	<input type="checkbox"/> Ausgang: Zeit in [s] seit letzter Touchaktion
fbDoubleClickDelay	TON	<input type="checkbox"/> Interne Variable
TouchAction	TouchAction	<input type="checkbox"/> Interne Variable
TouchActionOld	TouchAction	<input type="checkbox"/> Interne Variable
fbDTGetTime	DTGetTime	<input type="checkbox"/> Interne Variable

Der aktuelle Status eines Klicks wird durch den Ausgang „eState“ bzw. „eStateSync“ aufgeschlüsselt:

BrbVc4TouchStates_ENUM			
12	eBRBVC4_TOUCHSTATE_UNPUSHED	0	0=Nicht gedrückt
12	eBRBVC4_TOUCHSTATE_UNPUSHED_EDGE	1	1=Gerade losgelassen
12	eBRBVC4_TOUCHSTATE_PUSHED_EDGE	2	2=Gerade gedrückt
12	eBRBVC4_TOUCHSTATE_PUSHED	3	3=Gedrückt
12	eBRBVC4_TOUCHSTATE_DOUBLECLICK	4	4=Doppelklick (nur für einen Zyklus)

Die Funktion kann außerdem zum Synchronisieren von Zeichenbefehlen genutzt werden.

Das Zeichnen mithilfe der Funktionen der Bibliothek „VisApi“ muss nicht zwangsweise in jedem Zyklus des Visu-Tasks ausgeführt werden. Die Anzeige der projizierten Elemente wird sowieso nur alle paar 100ms aufgefrischt und es wäre somit eine unnötig große Belastung der CPU.

Ein weiterer Grund zur Synchronisierung ist folgender: Soll ein sich ändernder Inhalt nicht in einer Drawbox, sondern direkt auf einer Seite gezeichnet werden, muss vor jedem neuen Zeichnen die alte Zeichnung gelöscht werden. Das geschieht üblicherweise mit der Funktion „VisApi.VA\_Redraw“, welche aber einige Zeit benötigt, während der der Zugriff für die normale Visualisierung gesperrt ist. Um diesen Umständen Rechnung zu tragen, wird hier ein entsprechender Mechanismus zur Verfügung gestellt:

Über den Eingang „nRedrawCycles“ wird eine Grenze für den Zähler „nRedrawCounter“ parametrisiert (bei einer Zykluszeit von 10ms sollte eine Grenze von 10 Zyklen = 100ms vollends genügen).

Dieser Zähler zählt nun zyklisch von 0 auf diese Grenze. Ist der Zähler 0, wird automatisch ein „VisApi.VA\_Redraw“ ausgeführt. Eigene Zeichenbefehle direkt auf eine Seite sollten ebenfalls nur ausgeführt werden, wenn dieser Zähler 0 ist, und zwar nach dem Aufruf von „BrbVc4HandleGeneral“. Das Ausführen von „VisApi.VA\_Redraw“ kann durch „bDisableRedraw“ unterdrückt werden und spart somit erhebliche CPU-Leistung, wenn nicht direkt auf einer Seite gezeichnet wird.

Bei komplexeren Zeichenfunktionen (z.B. Trend) wird dieser Zähler benutzt, um das Zeichnen mehrerer Anzeigen auf verschiedene Zyklen zu verteilen und so eine gleichmäßigere Verteilung der CPU-Belastung zu erreichen.

Manche Controls dieser Bibliothek, welche Zeichenbefehle benutzen, machen von diesem Mechanismus Gebrauch (z.B. „Touchgrid“).

Um Flanken des Touchs auch dann sicher auszuwerten, wenn der Counter benutzt wird, gibt es den Ausgang „Touch.eStateSync“. Dieser lässt den Flankenzustand (Edge) bis nach einem Zyklus von „nRedrawCounter“ = 0 anstehen.

Über die Ausgänge „dtLastTouchAction“ und „nTimeSinceLastTouchAction“ werden Informationen über den zuletzt erkannten Touch-Klick ausgegeben. So könnte z.B. einfach eine Auto-LogOff-Funktion (automatisches Ausloggen des Benutzers nach einer bestimmten Zeit ohne Bedien-Aktion) realisiert werden.

## 3.1.2 BrbVc4PageHandling

### 3.1.2.1 Struktur

BrbVc4PageHandling_TYP			
nPageDefault	DINT	<input type="checkbox"/>	Eingang: Start-/Standard-Seite
bChangePageDirect	BOOL	<input type="checkbox"/>	Eingang: Auf jeden Fall Seite wechseln
nPageChange	DINT	<input type="checkbox"/>	Zur Anbindung an den Datenpunkt
nPageCurrent	DINT	<input type="checkbox"/>	Zur Anbindung an den Datenpunkt
nPageNext	DINT	<input type="checkbox"/>	Interne Variable
PagesPreviousLifo	BrbMemListManagement_Typ	<input type="checkbox"/>	Interne Variable
nPagesPrevious	DINT[0..nBRBVC4_PAGE_LAST_INDEX_MAX]	<input type="checkbox"/>	Interne Variable
bPageInit	BOOL	<input type="checkbox"/>	Ausgang: Flanke bei Einsprung eine Seite
bPageExit	BOOL	<input type="checkbox"/>	Ausgang: Flanke beim Verlassen einer Seite
bPageChangeInProgress	BOOL	<input type="checkbox"/>	Ausgang: Seitenumschaltung im Gange

Hiermit kann eine komfortable Seitenumschaltung implementiert werden. Z.B. kann erkannt werden, wenn ein Einsprung auf eine Seite erfolgt oder wenn eine Seite verlassen wird. So kann dann einmalig Code für diese Ereignisse ausgeführt werden. Voraussetzung dafür ist, dass alle Seitenwechsel nur über

diese Funktionen ausgeführt werden. Außerdem sollte die Bearbeitung der Seiten-Logik in einer Switch-Anweisung erledigt werden. Das hat den weiteren Vorteil, dass der Code für eine Seite nur dann ausgeführt wird, wenn die Seite auch sichtbar ist. Beispiel:

```
// Seiten -----
switch (Vis.PageHandling.nPageCurrent)
{
    // -----
    case ePAGE_TEMPLATE:
        if (Vis.PageHandling.bPageInit == 1 && Vis.PageHandling.bPageChangeInProgress
== 0)
        {
            Vis.PageHandling.bPageInit = 0;
            // Einsprungs-Code...
        }
        if (Vis.PageHandling.bPageChangeInProgress == 0)
        {
            // Zyklischer Code...
        }
        if (Vis.PageHandling.bPageExit == 1)
        {
            Vis.PageHandling.bPageExit = 0;
            // Verlassen-Code...
        }
        break;
    // Weitere Seiten...
```

Im 1.IF kann Einsprungs-Code ausgeführt werden. Das Flanken-Flag muss unbedingt zurückgesetzt werden.

Im 2.IF kann zyklischer Code ausgeführt werden.

Im 3.IF kann Verlassen-Code ausgeführt werden. Das Flanken-Flag muss unbedingt zurückgesetzt werden.

Weiterhin wird automatisch eine Liste der letzten Seiten geführt, um jederzeit eine Navigation zur vorherigen Seite auszuführen. Dadurch kann einfach eine anspruchsvolle Navigation mit einer Seiten-Hierarchie implementiert werden.

### 3.1.2.2 BrbVc4HandleChangePage

```
unsigned short BrbVc4HandleChangePage(struct BrbVc4PageHandling_TYP* pPageHandling)
```

Argumente:

```
struct BrbVc4PageHandling_TYP* pPageHandling
    Zeiger auf die Instanz von "BrbVc4PageHandling_TYP"
```

Rückgabe:

```
UINT
    Immer 0
```

Beschreibung:

Diese Funktion sollte zyklisch im Visualisierungs-Task aufgerufen werden.

Sie behandelt die Seitenumschaltung und das Setzen der Flags, sowie das Führen der Seiten-Liste.

### 3.1.2.3 BrbVc4ChangePage

```
plcbit BrbVc4ChangePage(struct BrbVc4PageHandling_TYP* pPageHandling, signed long nPageIndex,
plcbit bStoreLastPage)
```

Argumente:

```
struct BrbVc4PageHandling_TYP* pPageHandling
    Zeiger auf die Instanz von "BrbVc4PageHandling_TYP*"
DINT nPageIndex
    Der Index der aufzurufenden Seite
BOOL bStoreLastPage
    Gibt an, ob die aktuelle Seite vor dem Wechsel in die Liste der vorherigen Seiten aufgenommen werden soll.
```

Rückgabe:

BOOL

0=Es wurde versucht, auf die schon aktuelle Seite zu wechseln  
1=Seitenumschaltung wird eingeleitet

Beschreibung:

Diese Funktion initialisiert den Wechsel auf eine Seite und das Setzen der Flags. Wechsel auf die schon aktuelle Seite werden nicht gemacht, es sei denn der Eingang „bChangePage-Direct“ in der Struktur „BrbVc4PageHandling\_TYP“ ist gesetzt. Dann werden auch die Init- und Exit-Flags richtig gesetzt. Der Eingang wird beim Aufruf von „BrbVc4ChangePage“ automatisch zurückgesetzt.

### 3.1.2.4 BrbVc4ChangePageBack

```
plcbrit BrbVc4ChangePageBack(struct BrbVc4PageHandling_TYP* pPageHandling)
```

Argumente:

struct BrbVc4PageHandling\_TYP\* pPageHandling  
Zeiger auf die Instanz von "BrbVc4PageHandling\_TYP"

Rückgabe:

BOOL

0=Da die Liste keine Seiten mehr beinhaltet, wird auf die Standard-Seite gewechselt.  
1=Seitenumschaltung auf die vorherige Seite wird eingeleitet

Beschreibung:

Diese Funktion initialisiert den Wechsel auf die vorherige Seite und das Setzen der Flags. Bei „BrbVc4ChangePage“ kann angegeben werden, welche Seiten in der Liste gespeichert werden. Enthält die Liste keine Seite mehr, wird auf die Standard-Seite gewechselt. Es werden max. die letzten 10 Seiten gespeichert. Das sollte mehr als ausreichend sein.

### 3.1.2.5 BrbVc4HandleScreenSaver

```
plcbrit BrbVc4HandleScreenSaver(struct BrbVc4ScreenSaver_TYP* pScreenSaver, struct BrbVc4General_TYP* pGeneral, struct BrbVc4PageHandling_TYP* pPageHandling)
```

Argumente:

struct BrbVc4ScreenSaver\_TYP\* pScreenSaver  
Zeiger auf die Instanz von "BrbVc4ScreenSaver\_TYP"  
struct BrbVc4General\_TYP\* pGeneral  
Zeiger auf die Instanz von "BrbVc4General\_TYP"  
struct BrbVc4PageHandling\_TYP\* pPageHandling  
Zeiger auf die Instanz von "BrbVc4PageHandling\_TYP"

Rückgabe:

BOOL

0=Keine Umschaltung  
1=Zeit ist abgelaufen und die Seitenumschaltung auf die Bildschirmschoner-Seite wird eingeleitet (nur für einen Zyklus)

Beschreibung:

Die Funktion implementiert einen Bildschirmschoner mit Umschaltung auf eine spezifizierte Seite, da der Bildschirmschoner von VC dafür nicht verwendet werden darf (die Umschaltung der Seite würde dabei nicht über BrbVc4ChangePage geschehen und damit die ganze Navigation und die Init- bzw. Exit-Funktionalität beeinträchtigen).

Die Parametrierung erfolgt dabei über eine Struktur:

BrbVc4ScreenSaver_TYP			
bEnable	BOOL	<input type="checkbox"/>	Eingang: 1=Eingeschaltet
hsScreen	BrbVc4Hotspot_TYP	<input type="checkbox"/>	Control: Optionaler Hotspot für Rückschaltung
nScreenSaverPage	UINT	<input type="checkbox"/>	Eingang: Bildschirmschoner-Seite
fbScreenSaver	TON	<input type="checkbox"/>	Interne Variable
TouchOld	BrbVc4GeneralTouch_TYP	<input type="checkbox"/>	Interne Variable
nPageBeforeScreenSaver	DINT	<input type="checkbox"/>	Interne Variable
BrbVc4PageHandling_TYP			

Die Zeit ohne Mausklick, welche ablaufen muss, damit auf die parametrisierte Seite umgeschaltet wird, muss über `fbScreenSaver.PT` in [ms] angegeben werden. Bei einem erneuten Mausklick wird auf die vorher gezeigte Seite zurückgeschaltet. Init- und Exit-Flags werden dabei richtig gesetzt. Optional kann auch ein Hotspot auf der Bildschirmschoner-Seite verwendet werden, welcher über „`hsScreen`“ angebunden werden kann. Das ist nötig, wenn auf leistungsschwachen Zielsystemen der Zugriff auf den Touch (siehe „`BrbVc4HandleGeneral`“) bei den meisten Aufrufen verweigert wird.

## 3.2 Controls

In diesem Paket finden sich Struktur-Deklarationen und Funktionen zur transparenten Programmierung von Vc4-Controls.

Dabei werden visuelle Komponenten über eine Struktur angebunden. Die dazugehörige Funktion wertet die Eingaben aus, verändert evtl. die Darstellung und liefert dem Task die dazugehörigen Informationen, z.B. den Status eines Buttons.

Es gibt auch Strukturen ohne Funktion, z.B. „BrbVc4Bitmap\_TYP“. Sie dienen lediglich dem strukturierten Datenaustausch zwischen visueller Komponente und Logik.

Es ist nicht immer notwendig, alle Datenpunkte eines Steuerelements zu verbinden. Wenn z.B. der Farbumschlag nicht benötigt wird, muss auch der entsprechende Datenpunkt nicht verbunden werden.

Mit der Bibliothek werden einige Bitmaps mitgeliefert, welche genutzt werden können (z.B. für die Darstellung einer Checkbox). Die transparente Farbe für alle diese Bitmaps ist Lila = 21 (R=255, G=0, B=255). Sollten diese Bitmaps nicht mit dem Aussehen der geplanten Visualisierung harmonisieren, steht es dem Anwender frei, auch eigene Bitmaps zu verwenden.

### 3.2.1 BrbVc4ControlStatusHandling

Hierbei handelt es sich um Funktionen, die den Umgang mit dem Status-Datenpunkt eines Controls einfacher und transparenter machen.

#### 3.2.1.1 BrbVc4SetControlEnableity

```
plcbit BrbVc4SetControlEnableity(unsigned short* pStatus, plcbit bEnable)
```

Argumente:

`UINT*` pStatus  
Zeiger auf den Status-Datenpunkt  
`BOOL` bEnable  
0=Disabled  
1=Enabled

Rückgabe:

`BOOL`  
0=Disabled  
1=Enabled

Beschreibung:

Diese Funktion setzt oder löscht das Enable-Bit des Status-Datenpunktes.

#### 3.2.1.2 BrbVc4IsControlEnabled

```
plcbit BrbVc4IsControlEnabled(unsigned short nStatus)
```

Argumente:

`UINT` nStatus  
Status-Datenpunkt

Rückgabe:

`BOOL`  
0=Disabled  
1=Enabled

Beschreibung:

Diese Funktion gibt das Enable-Bit des Status-Datenpunktes zurück.

#### 3.2.1.3 BrbVc4SetControlVisibility

```
plcbit BrbVc4SetControlVisibility(unsigned short* pStatus, plcbit bVisible)
```

Argumente:

`UINT*` pStatus  
Zeiger auf den Status-Datenpunkt  
`BOOL` bVisible  
0=Unsichtbar

1=Sichtbar

Rückgabe:

BOOL

0=Unsichtbar  
1=Sichtbar

Beschreibung:

Diese Funktion setzt oder löscht das Sichtbarkeits-Bit des Status-Datenpunktes.

### 3.2.1.4 BrbVc4IsControlVisible

```
plcbit BrbVc4IsControlVisible(unsigned short nStatus)
```

Argumente:

UINT nStatus

Status-Datenpunkt

Rückgabe:

BOOL

0=Unsichtbar  
1=Sichtbar

Beschreibung:

Diese Funktion gibt das Sichtbarkeits-Bit des Status-Datenpunktes zurück.

### 3.2.1.5 BrbVc4SetControlFocus

```
plcbit BrbVc4SetControlFocus(unsigned short* pStatus, plcbit bFocus)
```

Argumente:

UINT\* pStatus

Zeiger auf den Status-Datenpunkt

BOOL bVisible

0=Focus setzen  
1=Focus löschen

Rückgabe:

BOOL

0=Focus gesetzt  
1=Focus gelöscht

Beschreibung:

Diese Funktion setzt oder löscht das Focus-Bit des Status-Datenpunktes.

Achtung: Auf einer Seite darf immer nur ein Control den Focus haben. Das Löschen des Focus-Bits aller anderen Controls muss applikativ gemacht werden.

### 3.2.1.6 BrbVc4HasControlFocus

```
plcbit BrbVc4HasControlFocus(unsigned short nStatus)
```

Argumente:

UINT nStatus

Status-Datenpunkt

Rückgabe:

BOOL

0=Control hat den Focus  
1=Control hat den Focus nicht

Beschreibung:

Diese Funktion gibt das Focus-Bit des Status-Datenpunktes zurück.

### 3.2.1.7 BrbVc4IsControlInputActive

```
plcbit BrbVc4IsControlInputActive(unsigned short nStatus)
```

Argumente:

UINT nStatus

Status-Datenpunkt

Rückgabe:

BOOL

0=Eingabe am Control nicht aktiv  
1=Eingabe am Control aktiv

Beschreibung:

Diese Funktion gibt das Edit-Bit des Status-Datenpunktes zurück.

### 3.2.1.8 BrbVc4OpenTouchpad

```
plcbit BrbVc4OpenTouchpad(unsigned short* pStatus)
```

Argumente:

UINT\* pStatus

Zeiger auf den Status-Datenpunkt

Rückgabe:

BOOL

Immer 0

Beschreibung:

Diese Funktion setzt das Touchpad-Bit des Status-Datenpunktes.  
Auf einer Seite kann immer nur ein Control das Touchpad geöffnet haben.

### 3.2.1.9 BrbVc4CloseTouchpad

```
plcbit BrbVc4CloseTouchpad(unsigned short* pStatus)
```

Argumente:

UINT\* pStatus

Zeiger auf den Status-Datenpunkt

Rückgabe:

BOOL

Immer 0

Beschreibung:

Diese Funktion löscht das Touchpad-Bit des Status-Datenpunktes.  
Auf einer Seite kann immer nur ein Control das Touchpad geöffnet haben.

### 3.2.1.10 BrbVc4IsTouchpadOpen

```
plcbit BrbVc4IsTouchpadOpen(unsigned short nStatus)
```

Argumente:

UINT nStatus

Status-Datenpunkt

Rückgabe:

BOOL

0=Touchpad geschlossen  
1= Touchpad geöffnet

Beschreibung:

Diese Funktion gibt das Touchpad-Bit des Status-Datenpunktes zurück.

### 3.2.1.11 BrbVc4SetControlColor

```
unsigned short BrbVc4SetControlColor(unsigned short* pColorDatapoint, plcbit bCondition, unsigned short nColorTrue, unsigned short nColorFalse)
```

Argumente:

UINT\* pColorDatapoint

Zeiger auf den Color-Datenpunkt

BOOL bCondition

Bedingung

UINT nColorTrue



Farbe für Bedingung = 1  
`UINT` `nColorFalse`  
Farbe für Bedingung = 0

**Rückgabe:**








`UINT`  
Immer 0

**Beschreibung:**

Diese Funktion ersetzt eine If-Else-Anweisung zum Setzen einer Farbe in Abhängigkeit einer bool-schen Bedingung.

## 3.2.2 Bitmap-Animation

### 3.2.2.1 Struktur

BrbVc4Animation_TYP			
	<code>bEnable</code>	<code>BOOL</code>	<input type="checkbox"/> Parameter: 1=Enabled
	<code>nIndexMin</code>	<code>UINT</code>	<input type="checkbox"/> Parameter: Kleinster Bitmap-Index der Animation
	<code>nIndexMax</code>	<code>UINT</code>	<input type="checkbox"/> Parameter: Größter Bitmap-Index der Animation
	<code>nIndexStanding</code>	<code>UINT</code>	<input type="checkbox"/> Parameter: Bitmap-Index bei Stillstand
	<code>nIndex</code>	<code>UINT</code>	<input type="checkbox"/> Zur Anbindung an den Datenpunkt
	<code>nStatus</code>	<code>UINT</code>	<input type="checkbox"/> Zur Anbindung an den Datenpunkt
	<code>fbTimer</code>	<code>TON</code>	<input type="checkbox"/> Interne Variable (Zeit PT muss eingestellt werden)

### 3.2.2.2 BrbVc4HandleAnimation

```
unsigned short BrbVc4HandleAnimation(struct BrbVc4Animation_TYP* pAnimation)
```

**Argumente:**

`struct` `BrbVc4Animation_TYP*` `pAnimation`  
Zeiger auf die Instanz

**Rückgabe:**

`UINT`  
Immer 0





**Beschreibung:**

Diese Funktion inkrementiert den Index für ein Bitmap in einem einstellbaren Intervall. Dadurch kann eine Animation durch eine Bitmap-Folge dargestellt werden.

## 3.2.3 Bargraph

Dieses Control besitzt keine dazugehörige Funktion. Der Datentyp dient lediglich der strukturierten Anbindung der Datenpunkte.






### 3.2.3.1 Struktur

BrbVc4Bargraph_TYP			
	<code>rValue</code>	<code>REAL</code>	<input type="checkbox"/> Zur Anbindung an den Datenpunkt
	<code>nValue</code>	<code>DINT</code>	<input type="checkbox"/> Zur Anbindung an den Datenpunkt
	<code>nColor</code>	<code>UINT</code>	<input type="checkbox"/> Zur Anbindung an den Datenpunkt
	<code>nStatus</code>	<code>UINT</code>	<input type="checkbox"/> Zur Anbindung an den Datenpunkt

## 3.2.4 Bitmap






Dieses Control besitzt keine dazugehörige Funktion. Der Datentyp dient lediglich der strukturierten Anbindung der Datenpunkte.

### 3.2.4.1 Struktur

BrbVc4Bitmap_TYP				
	nIndex	UINT	<input type="checkbox"/>	Zur Anbindung an den Datenpunkt
	nColor	UINT	<input type="checkbox"/>	Zur Anbindung an den Datenpunkt
	nFillColor1	UINT	<input type="checkbox"/>	Zur Anbindung an den Datenpunkt
	nFillColor2	UINT	<input type="checkbox"/>	Zur Anbindung an den Datenpunkt
	nStatus	UINT	<input type="checkbox"/>	Zur Anbindung an den Datenpunkt

## 3.2.5 Normaler Button

### 3.2.5.1 Struktur

BrbVc4Button_TYP				
	nBmplIndex	UINT	<input type="checkbox"/>	Zur Anbindung an den Datenpunkt
	nTextIndex	UINT	<input type="checkbox"/>	Zur Anbindung an den Datenpunkt
	nColor	UINT	<input type="checkbox"/>	Zur Anbindung an den Datenpunkt
	bClicked	BOOL	<input type="checkbox"/>	Zur Anbindung an den Datenpunkt
	nStatus	UINT	<input type="checkbox"/>	Zur Anbindung an den Datenpunkt

### 3.2.5.2 BrbVc4HandleButton

```
plcbit BrbVc4HandleButton(struct BrbVc4Button_TYP* pButton)
```

#### Argumente:

```
struct BrbVc4Button_TYP* pButton  
Zeiger auf die Instanz
```

#### Rückgabe:

```
BOOL
```

0=Nicht gedrückt  
1=Button wurde gedrückt

#### Beschreibung:

Diese Funktion behandelt die Logik eines normalen Buttons.

Das Item „bClicked“ muss durch einen virtuellen Key vom Typ „SetDatapoint“ auf 1 gesetzt werden. Das Löschen dieses Items erfolgt dann applikativ. Damit wird gewährleistet, dass der Klick auch wirklich erkannt und bearbeitet wird.

Das Item „nBmplIndex“ wird automatisch gesetzt. Dabei wird folgende Projektierung vorausgesetzt:

0=Disabled (Ausgegrautes Bitmap)  
1=Enabled (Normales Bitmap)

#### Beispiel für den Aufruf:

```
if(BrbVc4HandleButton(&Vis.PageControls.btnNormal) == 1)  
{  
    Vis.PageControls.btnNormal.bClicked = 0;  
    // Code...  
}
```

## 3.2.6 Checkbox

Es gibt kein fertiges Checkbox-Control. Hier wird es durch die Kombination von drei Controls realisiert (Bitmap, Text, Hotspot).

Die benötigten Bitmaps werden mit der Bibliothek mitgeliefert:



### 3.2.6.1 Struktur

BrbVc4Checkbox_TYP				
◆	bClicked	BOOL	<input type="checkbox"/>	Zur Anbindung an den Datenpunkt
◆	nBmplIndex	UINT	<input type="checkbox"/>	Zur Anbindung an den Datenpunkt
◆	nTextColor	UINT	<input type="checkbox"/>	Zur Anbindung an den Datenpunkt
◆	nStatus	UINT	<input type="checkbox"/>	Zur Anbindung an den Datenpunkt
◆	bChecked	BOOL	<input type="checkbox"/>	Ausgang: 1=Angehakt

### 3.2.6.2 BrbVc4HandleCheckbox

```
plcbit BrbVc4HandleCheckbox(struct BrbVc4Checkbox_TYP* pCheckbox)
```

#### Argumente:

`struct BrbVc4Checkbox_TYP* pCheckbox`  
Zeiger auf die Instanz

#### Rückgabe:

`BOOL`  
0=Checkbox wurde nicht geändert  
1=Checkbox wurde geändert

#### Beschreibung:

Diese Funktion behandelt die Logik für eine aus drei Controls zusammen gesetzten Checkbox (Bitmap, Text und Hotspot).

Das Item „bClicked“ muss durch einen virtuellen Key vom Typ „SetDatapoint“ auf 1 gesetzt werden.

Das Item „nBmplIndex“ wird automatisch gesetzt. Dabei wird folgende Projektierung vorausgesetzt:

0=Nicht angehakt und Disabled (Ausgegrautes Bitmap)

1=Angehakt und Disabled (Ausgegrautes Bitmap)

2=Nicht angehakt und Enabled (Normales Bitmap)

3=Angehakt und Enabled (Normales Bitmap)

Das Item „nTextColor“ wird automatisch gesetzt.

#### Beispiel für den Aufruf:

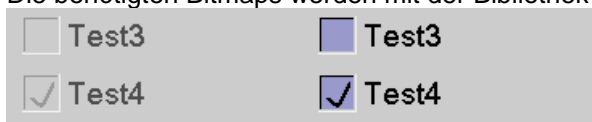
```
if (BrbVc4HandleCheckbox(&Vis.PageControls.chkTest1) == 1)
{
    // Code...
}
```

### 3.2.7 CheckboxButton

Es gibt kein fertiges Checkbox-Control. Hier wird es durch einen Button realisiert.

Die Projektierung ist schneller als mit der oben genannten Checkbox.

Die benötigten Bitmaps werden mit der Bibliothek mitgeliefert:



#### 3.2.7.1 Struktur

BrbVc4CheckboxButton_TYP				
◆	nBmplIndex	UINT	<input type="checkbox"/>	Zur Anbindung an den Datenpunkt
◆	nTextIndex	UINT	<input type="checkbox"/>	Zur Anbindung an den Datenpunkt
◆	nColor	UINT	<input type="checkbox"/>	Zur Anbindung an den Datenpunkt
◆	bVisPushed	BOOL	<input type="checkbox"/>	Zur Anbindung an den Datenpunkt
◆	bChecked	BOOL	<input type="checkbox"/>	Ausgang: 1=Angehakt
◆	nStatus	UINT	<input type="checkbox"/>	Zur Anbindung an den Datenpunkt
◆	eToggleState	BrbVc4CheckboxButtonStates_ENUM	<input type="checkbox"/>	Ausgang: Momentaner Status des Buttons
◆	bVisPushedOld	BOOL	<input type="checkbox"/>	Interne Variable
◆	bCheckedOld	BOOL	<input type="checkbox"/>	Interne Variable

### 3.2.7.2 BrbVc4HandleCheckboxButton

```
plcbrit BrbVc4HandleCheckboxButton(struct BrbVc4CheckboxButton_TYP* pButton)
```

#### Argumente:

```
struct BrbVc4Checkbox_TYP* pButton  
Zeiger auf die Instanz
```

#### Rückgabe:

```
BOOL
```

0=Checkbox wurde nicht geändert  
1=Checkbox wurde geändert

#### Beschreibung:

Diese Funktion behandelt die Logik einer Checkbox als Button.

Das Item „bVisPushed“ muss durch einen virtuellen Key vom Typ „ToggleDatapoint“ gesetzt werden.

Das Item „bChecked“ kann zum Auswerten oder Umschalten des Signals im Programm verwendet werden.

Das Item „eToggleState“ liefert den momentanen Zustand durch eine Enumeration:

BrbVc4CheckboxButtonStates_ENUM	
eBRBVC4_CHKBTNSTATE_UNPUSHED	0=Nicht gedrückt
eBRBVC4_CHKBTNSTATE_UNPUSHED_EDG	1=Gerade losgelassen
eBRBVC4_CHKBTNSTATE_PUSHED_EDGE	2=Gerade gedrückt
eBRBVC4_CHKBTNSTATE_PUSHED	3=Gedrückt
BrbVc4CheckboxButton_TYP	

Das Item „nBmpIndex“ wird automatisch gesetzt. Dabei wird folgende Projektierung vorausgesetzt:

0=Nicht angehakt und Disabled (Ausgegrautes Bitmap)  
1=Angehakt und Disabled (Ausgegrautes Bitmap)  
2=Nicht angehakt und Enabled (Normales Bitmap)  
3=Angehakt und Enabled (Normales Bitmap)

Beispiel für den Aufruf:

```
if (BrbVc4HandleCheckboxButton(&Vis.PageVc4.chkbtnTest3) == 1)  
{  
    // Code...  
}
```

### 3.2.8 DateTime

Dieses Control besitzt keine dazugehörige Funktion. Der Datentyp dient lediglich der strukturierten Anbindung der Datenpunkte.

#### 3.2.8.1 Struktur

BrbVc4DateTime_TYP			
dValue	DATE_AND_TIME	<input type="checkbox"/>	Zur Anbindung an den Datenpunkt
nColor	UINT	<input type="checkbox"/>	Zur Anbindung an den Datenpunkt
nStatus	UINT	<input type="checkbox"/>	Zur Anbindung an den Datenpunkt

### 3.2.9 Drawbox

Dieses Control besitzt keine dazugehörige Funktion. Der Datentyp dient lediglich der strukturierten Anbindung der Datenpunkte.

#### 3.2.9.1 Struktur

BrbVc4Drawbox_TYP			
nLeft	UDINT	<input type="checkbox"/>	Eingang: Linke Koordinate
nTop	UDINT	<input type="checkbox"/>	Eingang: Obere Koordinate
nWidth	UDINT	<input type="checkbox"/>	Eingang: Breite
nHeight	UDINT	<input type="checkbox"/>	Eingang: Höhe
sFullName	STRING[nBRB_FILE_NAME_CHAR_MAX]	<input type="checkbox"/>	Pfad zur Anbindung
nBackgroundColor	UINT	<input type="checkbox"/>	Zur Anbindung an den Datenpunkt
nStatus	UINT	<input type="checkbox"/>	Zur Anbindung an den Datenpunkt

Der Name der Drawbox wird in den erweiterten Zeichenfunktionen zur Referenzierung benötigt. Die Syntax ist folgende: „Seitenname/Layername/Controlname“. Nur wenn diese Bezeichnung korrekt ist, kann in diese Drawbox gezeichnet werden (siehe AS-Hilfe „VisApi.VA\_Attach“).

## 3.2.10 Dropdown

### 3.2.10.1 Struktur

BrbVc4Dropdown_TYP		
nIndex	UINT	Zur Anbindung an den Datenpunkt
nIndexMin	UINT	Zur Anbindung an den Datenpunkt
nIndexMax	UINT	Zur Anbindung an den Datenpunkt
nOptions	USINT[0..nBRBVC4_DROPDOWN_OPTION_MAX]	Zur Anbindung an den Datenpunkt
nColor	UINT	Zur Anbindung an den Datenpunkt
blnInputCompleted	BOOL	Zur Anbindung an den Datenpunkt
nStatus	UINT	Zur Anbindung an den Datenpunkt

### 3.2.10.2 BrbVc4HandleDropdown

```
plcbit BrbVc4HandleDropdown(struct BrbVc4Dropdown_TYP* pDropdown)
```

#### Argumente:

`struct BrbVc4Dropdown_TYP* pDropdown`  
Zeiger auf die Instanz

#### Rückgabe:

`BOOL`  
0=Keine Eingabe  
1=Eingabe abgeschlossen

#### Beschreibung:

Diese Funktion behandelt die Logik eines Dropdowns. Es können sowohl eine String- als auch eine Vc-Text-Liste angeschlossen werden.

Zum Besetzen des Options-Arrays wird eine Enumeration zur Verfügung gestellt:

BrbVc4DropdownOptions_ENUM		
eBRBVC4_DDOPTION_NORMAL	0	0=Normal
eBRBVC4_DDOPTION_DISABLED	1	1=Ausgegraut
eBRBVC4_DDOPTION_INVISIBLE	2	2=Unsichtbar

#### Beispiel für den Aufruf:

```
if(BrbVc4HandleDropdown(&Vis.PageControls.ddTest1) == 1)
{
    // Code...
}
```

## 3.2.11 Edit

Dieses Control besitzt keine dazugehörige Funktion. Der Datentyp dient lediglich der strukturierten Anbindung der Datenpunkte.

### 3.2.11.1 Struktur

BrbVc4EditCtrl_TYP				
nStatus	UINT	<input type="checkbox"/>		Zur Anbindung an den Datenpunkt
nBusy	UINT	<input type="checkbox"/>		Zur Anbindung an den Datenpunkt
sUrl	STRING[nBRB_URL_CHAR_MAX]	<input type="checkbox"/>		Zur Anbindung an den Datenpunkt
sCmdRequest	STRING[nBRBVC4_EDIT_CTRL_CMD_CHAR_MAX]	<input type="checkbox"/>		Zur Anbindung an den Datenpunkt
sCmdResponse	STRING[nBRBVC4_EDIT_CTRL_CMD_CHAR_MAX]	<input type="checkbox"/>		Zur Anbindung an den Datenpunkt
nCmdStatus	UINT	<input type="checkbox"/>		Zur Anbindung an den Datenpunkt
nCompletion	UINT	<input type="checkbox"/>		Zur Anbindung an den Datenpunkt
nCursorLine	UDINT	<input type="checkbox"/>		Zur Anbindung an den Datenpunkt
nCursorColumn	UDINT	<input type="checkbox"/>		Zur Anbindung an den Datenpunkt
nInsertMode	USINT	<input type="checkbox"/>		Zur Anbindung an den Datenpunkt
nModified	USINT	<input type="checkbox"/>		Zur Anbindung an den Datenpunkt
nSelectionMode	USINT	<input type="checkbox"/>		Zur Anbindung an den Datenpunkt
nLineCount	UDINT	<input type="checkbox"/>		Interne Variable
nColumnsMax	UDINT	<input type="checkbox"/>		Interne Variable
sLastCmdRequest	STRING[nBRBVC4_EDIT_CTRL_CMD_CHAR_MAX]	<input type="checkbox"/>		Interne Variable

### 3.2.12 Gauge

Dieses Control besitzt keine dazugehörige Funktion. Der Datentyp dient lediglich der strukturierten Anbindung der Datenpunkte.

#### 3.2.12.1 Struktur

BrbVc4Gauge_TYP				
nValue	DINT	<input type="checkbox"/>		Zur Anbindung an den Datenpunkt
nValueMin	DINT	<input type="checkbox"/>		Zur Anbindung an den Datenpunkt
nValueMax	DINT	<input type="checkbox"/>		Zur Anbindung an den Datenpunkt
nColor	UINT	<input type="checkbox"/>		Zur Anbindung an den Datenpunkt
nStatus	UINT	<input type="checkbox"/>		Zur Anbindung an den Datenpunkt

### 3.2.13 Hotspot

Dieses Control besitzt keine dazugehörige Funktion. Der Datentyp dient lediglich der strukturierten Anbindung der Datenpunkte.

#### 3.2.13.1 Struktur

BrbVc4Hotspot_TYP				
bClicked	BOOL	<input type="checkbox"/>		Zur Anbindung an den Datenpunkt
nStatus	UINT	<input type="checkbox"/>		Zur Anbindung an den Datenpunkt

Das Item „bClicked“ sollte durch einen virtuellen Key vom Typ „SetDatapoint“ auf 1 gesetzt werden. Das Löschen dieses Items erfolgt dann applikativ. Damit wird gewährleistet, dass der Klick auch wirklich erkannt und bearbeitet wurde.

Beispiel für die Verwendung:

```
if(Vis.PageControls.hsTest.bClicked == 1)
{
    Vis.PageControls.hsTest.bClicked = 0;
    // Code...
}
```

### 3.2.14 Html

Dieses Control besitzt keine dazugehörige Funktion. Der Datentyp dient lediglich der strukturierten Anbindung der Datenpunkte.

### 3.2.14.1 Struktur

BrbVc4Html_TYP			
sCurrentUrl	STRING[nBRB_URL_CHAR_MAX]	<input type="checkbox"/>	Zur Anbindung an den Datenpunkt
sChangeUrl	STRING[nBRB_URL_CHAR_MAX]	<input type="checkbox"/>	Zur Anbindung an den Datenpunkt
sCurrentTitle	STRING[nBRB_URL_CHAR_MAX]	<input type="checkbox"/>	Zur Anbindung an den Datenpunkt
bBusy	BOOL	<input type="checkbox"/>	Zur Anbindung an den Datenpunkt
nHttpErrorCode	UDINT	<input type="checkbox"/>	Zur Anbindung an den Datenpunkt
nStatus	UINT	<input type="checkbox"/>	Zur Anbindung an den Datenpunkt
bBusyOld	BOOL	<input type="checkbox"/>	Interne Variable

### 3.2.15 HwPosSwitch2

Dies ist kein klassisches Vc4-Control. Hiermit kann ein Hardware-Schalter ausgewertet werden, welcher 3 Stellungen über 2 Eingänge an die Sps meldet.

#### 3.2.15.1 Struktur

BrbVc4HwPosSwitch2_TYP			
bPos0	BOOL	<input type="checkbox"/>	Hw-Eingang: Schalter auf linker Stellung
bPos2	BOOL	<input type="checkbox"/>	Hw-Eingang: Schalter auf rechter Stellung
nStatus	UINT	<input type="checkbox"/>	Zur Anbindung an den Datenpunkt
nSwitchStatus0	UINT	<input type="checkbox"/>	Zur Anbindung an den Datenpunkt
nSwitchStatus1	UINT	<input type="checkbox"/>	Zur Anbindung an den Datenpunkt
nSwitchStatus2	UINT	<input type="checkbox"/>	Zur Anbindung an den Datenpunkt
ePosOld	BrbVc4HwPosSwitchPositions_ENUM	<input type="checkbox"/>	Interne Variable
ePos	BrbVc4HwPosSwitchPositions_ENUM	<input type="checkbox"/>	Ausgang: Momentaner Status des Schalters

#### 3.2.15.2 BrbVc4HandleHwPosSwitch2

```
enum BrbVc4HwPosSwitchPositions_ENUM BrbVc4HandleHwPosSwitch2 (struct BrbVc4HwPosSwitch2_TYP*  
pPosSwitch)
```

##### Argumente:

```
struct BrbVc4HwPosSwitch2_TYP* pPosSwitch  
    Zeiger auf die Instanz
```

##### Rückgabe:

```
BrbVc4HwPosSwitchPositions_ENUM  
    Siehe unten
```

##### Beschreibung:

Diese Funktion behandelt die Logik eines Hardware-Schalters.

Das Item „bPos0“ muss durch den Eingang „Linke Stellung“, das Item „bPos2“ durch den Eingang „Rechte Stellung“ gesetzt werden.

An den Items „nSwitchStatusX“ wird automatisch das Unsichtbarkeits-Bit entsprechend der Stellung gesetzt.

So kann der Taster durch Bitmap-Controls visualisiert werden.

Das Item „ePos“ liefert den momentanen Zustand durch eine Enumeration:

BrbVc4HwPosSwitchPositions_ENUM	
eBRBVC4_HPSP0S_0_EDGE	0=Gerade auf linke Stellung geschaltet
eBRBVC4_HPSP0S_0	1=Auf linke Stellung gestellt
eBRBVC4_HPSP0S_1_EDGE	2=Gerade auf mittlere Stellung geschaltet
eBRBVC4_HPSP0S_1	3=Auf mittlere Stellung gestellt
eBRBVC4_HPSP0S_2_EDGE	4=Gerade auf rechte Stellung geschaltet
eBRBVC4_HPSP0S_2	5=Auf rechte Stellung gestellt
eBRBVC4_HPSP0S_UNDEF	6=Stellung aufgrund der Eingänge nicht zu ermitteln

##### Beispiel für den Aufruf:

```
BrbVc4HandleHwPosSwitch2 (&Vis.PageControls.btnHwPosSwitch);  
if (Vis.PageControls.btnHwPosSwitch.ePos == eBRBVC4_HPSP0S_0_EDGE)  
{  
    // Code...
```

```
}  
else if (Vis.PageControls.btnHwPosSwitch.ePos == eBRBVC4_HPSPPOS_1_EDGE)  
{  
    // Code...  
}  
else if (Vis.PageControls.btnHwPosSwitch.ePos == eBRBVC4_HPSPPOS_2_EDGE)  
{  
    // Code...  
}
```

### 3.2.16 HwSafetyButton

Dies ist kein klassisches Vc4-Control. Hiermit kann ein Hardware-Taster ausgewertet werden, welcher zwei Kontakte (Schließer und Öffner) gleichzeitig betätigt. Meistens werden damit sicherheitstechnische Benutzer-Freigaben realisiert.

#### 3.2.16.1 Struktur

BrbVc4HwSafetyButton_TYP			
bNormallyOpened	BOOL	<input type="checkbox"/>	Hw-Eingang: Schliesser
bNormallyClosed	BOOL	<input type="checkbox"/>	Hw-Eingang: Öffner
nColor	UINT	<input type="checkbox"/>	Zur Anbindung an den Datenpunkt
nStatus	UINT	<input type="checkbox"/>	Zur Anbindung an den Datenpunkt
bPushedOld	BOOL	<input type="checkbox"/>	Interne Variable
bPushed	BOOL	<input type="checkbox"/>	Ausgang: 0=Nicht gedrückt, 1=Gedrückt
eState	BrbVc4HwSafetyButtonStates_ENUM	<input type="checkbox"/>	Ausgang: Momentaner Status des Tasters

#### 3.2.16.2 BrbVc4HandleHwSafetyButton

```
plcbit BrbVc4HandleHwSafetyButton(struct BrbVc4HwSafetyButton_TYP* pSafetyButton)
```

##### Argumente:

```
struct BrbVc4HwSafetyButton_TYP* pSafetyButton  
    Zeiger auf die Instanz
```

##### Rückgabe:

```
BOOL  
    0=Gedrückt  
    1=Nicht gedrückt
```

##### Beschreibung:

Diese Funktion behandelt die Logik eines Hardware-Tasters mit zwei komplementären Eingängen. Das Item „bNormallyOpened“ muss durch den Eingang „Schließer“, das Item „bNormallyClosed“ durch den Eingang „Öffner“ gesetzt werden.

Das Item „eState“ liefert den momentanen Zustand durch eine Enumeration:

BrbVc4HwSafetyButtonStates_ENUM	
eBRBVC4_HSBSTATE_UNPUSHED	0=Nicht gedrückt
eBRBVC4_HSBSTATE_UNPUSHED_EDGE	1=Gerade losgelassen
eBRBVC4_HSBSTATE_PUSHED_EDGE	2=Gerade gedrückt
eBRBVC4_HSBSTATE_PUSHED	3=Gedrückt

##### Beispiel für den Aufruf:

```
BrbVc4HandleHwSafetyButton(&Vis.PageControls.btnHwSafetyButton);  
if (Vis.PageControls.btnHwSafetyButton.ePos == eBRBVC4_HSBSTATE_PUSHED)  
{  
    // Code...  
}  
else  
{  
    // Code...  
}
```



### 3.2.17 IncButton

Ein Inc-Button bietet die Funktion, bei einem dauerhaften Klick den ersten Impuls sofort zu liefern, dann eine Verzögerungszeit abzuwarten und dann einen Wiederhol-Impuls zu liefern. Diese Funktion ist z.B. bei Scroll-Buttons sehr komfortabel.

#### 3.2.17.1 Struktur

BrbVc4IncButton_TYP			
bEnabled	BOOL	<input type="checkbox"/>	Parameter: 1=Enabled
bSuppressDelay	BOOL	<input type="checkbox"/>	Parameter: 1=Keine Verzögerungszeit
bSuppressRepeat	BOOL	<input type="checkbox"/>	Parameter: 1=Keine Wiederholzeit
nBmpIndex	UINT	<input type="checkbox"/>	Zur Anbindung an den Datenpunkt
nTextIndex	UINT	<input type="checkbox"/>	Zur Anbindung an den Datenpunkt
nColor	UINT	<input type="checkbox"/>	Zur Anbindung an den Datenpunkt
bPushed	BOOL	<input type="checkbox"/>	Zur Anbindung an den Datenpunkt
nStatus	UINT	<input type="checkbox"/>	Zur Anbindung an den Datenpunkt
eIncState	BrbVc4IncButtonStates_ENUM	<input type="checkbox"/>	Ausgang: Momentaner Status des Buttons
bPushedOld	BOOL	<input type="checkbox"/>	Interne Variable
bEnabledOld	BOOL	<input type="checkbox"/>	Interne Variable
fbDelay	TON	<input type="checkbox"/>	Interne Variable (Zeit PT muss eingestellt werden)
fbRepeat	TON	<input type="checkbox"/>	Interne Variable (Zeit PT muss eingestellt werden)

#### 3.2.17.2 BrbVc4HandleIncButton

```
plcbit BrbVc4HandleIncButton(struct BrbVc4IncButton_TYP* pButton)
```

##### Argumente:

`struct BrbVc4IncButton_TYP* pButton`  
Zeiger auf die Instanz

##### Rückgabe:

BOOL  
0=Gedrückt  
1=Nicht gedrückt

##### Beschreibung:

Diese Funktion behandelt die Logik eines Inkremental-Buttons.

Das Item „bPushed“ muss durch einen virtuellen Key vom Typ „SetMomentaryDatapoint“ auf 1 gesetzt werden.

Durch die Items „bSuppressDelay“ und „bSuppressRepeat“ kann festgelegt werden, dass keine Verzögerung bzw. Wiederholung gemacht werden soll.

Das Item „eIncState“ liefert den momentanen Zustand durch eine Enumeration:

BrbVc4IncButtonStates_ENUM		
eBRBVC4_INCBTNSTATE_UNPUSHED		0=Nicht gedrückt
eBRBVC4_INCBTNSTATE_UNPUSHED_EDG		1=Gerade losgelassen
eBRBVC4_INCBTNSTATE_PUSHED_EDGE		2=Gerade gedrückt
eBRBVC4_INCBTNSTATE_PUSHED_DELAY		3=Verzögerungszeit läuft
eBRBVC4_INCBTNSTATE_PUSHED_REPEA		4=Wiederholzeit läuft
eBRBVC4_INCBTNSTATE_PUSHED_INC		5=Wiederholender Impuls

##### Beispiel für den Aufruf:

```
BrbVc4HandleIncButton(&Vis.PageControls.btnInc);  
if(Vis.PageControls.btnInc.eIncState == eBRBVC4_INCBTNSTATE_PUSHED_INC)  
{  
    // Code, z.B. Zähler erhöhen...  
}
```

### 3.2.18 JogButton

Ein Jog-Button bietet eine Tipp-Funktion mit Abschaltung nach einer einstellbaren Zeit.

Tipp-Funktionalität für kritische Module (z.B. Achsen) sollten aus sicherheitstechnischen Gründen nicht über eine Vc4-Visualisierung gemacht werden. Für unkritische Module kann dafür diese Logik verwendet werden.

### 3.2.18.1 Struktur

BrbVc4JogButton_TYP			
bEnabled	BOOL	<input type="checkbox"/>	Parameter: 1=Enabled
bSuppressTimeout	BOOL	<input type="checkbox"/>	Parameter: 1=Keine Verzögerungszeit
nBmpIndex	UINT	<input type="checkbox"/>	Zur Anbindung an den Datenpunkt
nTextIndex	UINT	<input type="checkbox"/>	Zur Anbindung an den Datenpunkt
nColor	UINT	<input type="checkbox"/>	Zur Anbindung an den Datenpunkt
bPushed	BOOL	<input type="checkbox"/>	Zur Anbindung an den Datenpunkt
nStatus	UINT	<input type="checkbox"/>	Zur Anbindung an den Datenpunkt
eJogState	BrbVc4JogButtonStates_ENUM	<input type="checkbox"/>	Ausgang: Momentaner Status des Buttons
bPushedOld	BOOL	<input type="checkbox"/>	Interne Variable
bEnabledOld	BOOL	<input type="checkbox"/>	Interne Variable
fbUnpush	TON	<input type="checkbox"/>	Interne Variable

### 3.2.18.2 BrbVc4HandleJogButton

```
plcbit BrbVc4HandleJogButton(struct BrbVc4JogButton_TYP* pButton)
```

#### Argumente:

```
struct BrbVc4JogButton_TYP* pButton  
    Zeiger auf die Instanz
```

#### Rückgabe:

```
BOOL  
    0=Gedrückt  
    1=Nicht gedrückt
```

#### Beschreibung:

Diese Funktion behandelt die Logik eines Tipp-Buttons mit Abschaltung nach spätestens einer einstellbaren Zeit, d.h. auch bei dauerhaft gedrücktem Button wird eine „UnpushedEdge“ generiert. Der Benutzer muss dann loslassen und erneut drücken.

Das Item „bPushed“ muss durch einen virtuellen Key vom Typ „SetMomentaryDatapoint“ auf 1 gesetzt werden.

Durch das Item „bSuppressTimeout“ kann festgelegt werden, dass keine zeitliche Abschaltung erfolgen soll. Das Item „eJogState“ liefert den momentanen Zustand durch eine Enumeration:

BrbVc4JogButtonStates_ENUM	
eBRBVC4_JOGBTNSTATE_UNPUSHED	0=Nicht gedrückt
eBRBVC4_JOGBTNSTATE_UNPUSHED_EDG	1=Gerade losgelassen
eBRBVC4_JOGBTNSTATE_PUSHED_EDGE	2=Gerade gedrückt
eBRBVC4_JOGBTNSTATE_PUSHED	3=Gedrückt

#### Beispiel für den Aufruf:

```
BrbVc4HandleJogButton(&Vis.PageControls.btnJog);  
if(Vis.PageControls.btnJog.eJogState == eBRBVC4_JOGBTNSTATE_PUSHED_EDGE)  
{  
    // Code...  
}  
else if(Vis.PageControls.btnJog.eJogState == eBRBVC4_JOGBTNSTATE_UNPUSHED_EDG)  
{  
    // Code...  
}
```

## 3.2.19 Layer

Dieses Control besitzt keine dazugehörige Funktion. Der Datentyp dient lediglich der strukturierten Anbindung der Datenpunkte.

### 3.2.19.1 Struktur

BrbVc4Layer_TYP			
nStatus	UINT	<input type="checkbox"/>	Zur Anbindung an den Datenpunkt

### 3.2.20 Listbox

Dieses Control besitzt keine dazugehörige Funktion. Der Datentyp dient lediglich der strukturierten Anbindung der Datenpunkte.

#### 3.2.20.1 Struktur

BrbVc4Listbox_TYP			
nIndex	UINT	<input type="checkbox"/>	Zur Anbindung an den Datenpunkt
nIndexMin	UINT	<input type="checkbox"/>	Zur Anbindung an den Datenpunkt
nIndexMax	UINT	<input type="checkbox"/>	Zur Anbindung an den Datenpunkt
nControlColor	UINT	<input type="checkbox"/>	Zur Anbindung an den Datenpunkt
nDisabledColor	UINT	<input type="checkbox"/>	Zur Anbindung an den Datenpunkt
nSelectedColor	UINT	<input type="checkbox"/>	Zur Anbindung an den Datenpunkt
nSliderColor	UINT	<input type="checkbox"/>	Zur Anbindung an den Datenpunkt
nOptions	USINT[0..nBRBVC4_LISTBOX_OPTION_MAX]	<input type="checkbox"/>	Zur Anbindung an den Datenpunkt
blnInputCompleted	BOOL	<input type="checkbox"/>	Zur Anbindung an den Datenpunkt
nStatus	UINT	<input type="checkbox"/>	Zur Anbindung an den Datenpunkt

### 3.2.21 Numeric

#### 3.2.21.1 Struktur

BrbVc4Numeric_TYP			
nValue	DINT		Zur Anbindung an den Datenpunkt
nMin	DINT		Zur Anbindung an den Datenpunkt
nMax	DINT		Zur Anbindung an den Datenpunkt
rValue	REAL		Zur Anbindung an den Datenpunkt
rMin	REAL		Zur Anbindung an den Datenpunkt
rMax	REAL		Zur Anbindung an den Datenpunkt
nColor	UINT		Zur Anbindung an den Datenpunkt
blnInputCompleted	BOOL		Zur Anbindung an den Datenpunkt
nStatus	UINT		Zur Anbindung an den Datenpunkt

Je nach Datentyp kann „nValue“ oder „rValue“ angebunden werden.

Hinweis: Diese Struktur ist nur noch aus Kompatibilitäts-Gründen vorhanden. Bei neuen Anwendungen sollte die Struktur ‚BrbVc4NumericEx‘ verwendet werden, welche auch die Datentypen UDINT und LREAL unterstützt (siehe unten).

#### 3.2.21.2 BrbVc4HandleNumericInput

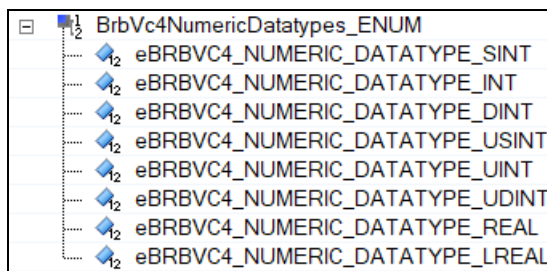
```
plcbit BrbVc4HandleNumericInput(struct BrbVc4Numeric_TYP* pNumeric, unsigned long pSourceValue, enum BrbVc4NumericDatatypes_ENUM eSourceDatatype)
```

##### Argumente:

`struct BrbVc4Numeric_TYP* pNumeric`  
Zeiger auf die Instanz

`UDINT pSourceValue`  
Zeiger auf den Quellwert

`enum BrbVc4NumericDatatypes_ENUM eSourceDatatype`  
Datentyp des Quellwerts



#### Rückgabe:

BOOL

0=Keine Eingabe  
1=Eingabe abgeschlossen

#### Beschreibung:

Diese Funktion behandelt die Eingabe-Logik eines Numeric-Controls. Voraussetzung ist, dass der Datenpunkt „InputCompleted“ bei Abschluss der Eingabe auf „1“ gesetzt wird, sowie der Datenpunkt „nValue“ oder „rValue“ (bei REAL) angeschlossen ist.

Bei Abschluss einer Eingabe wird der eingegebene Wert auf die Quelle kopiert, ansonsten wird die Quelle auf den Datenpunkt kopiert.

**Achtung: Die Datentypen UDINT und LREAL werden nicht bzw. nicht korrekt unterstützt.**

Hinweis: Diese Funktion ist nur noch aus Kompatibilitäts-Gründen vorhanden. Bei neuen Anwendungen sollte die Funktion `BrbVc4HandleNumericInputEx` verwendet werden, welche auch die Datentypen UDINT und LREAL unterstützt (siehe unten).

#### Beispiel für den Aufruf:

```
BrbVc4HandleNumericInput(&Vis.PageControls.numTest2, (UDINT)&gPar.nInterval,  
eBRBVC4_NUMERIC_DATATYPE_UINT);
```

### 3.2.22 NumericEx

Die obige Implementierung ‚Numeric‘ unterstützt die Datentypen UDINT und LREAL **nicht**. Deshalb gibt es diese erweiterte Implementierung ‚NumericEx‘.

#### 3.2.22.1 Struktur

BrbVc4NumericEx_TYP			
nValue	DINT	Zur Anbindung an den Datenpunkt (SINT, INT, DINT)	
nMin	DINT	Zur Anbindung an den Datenpunkt (SINT, INT, DINT)	
nMax	DINT	Zur Anbindung an den Datenpunkt (SINT, INT, DINT)	
nUValue	UDINT	Zur Anbindung an den Datenpunkt (USINT, UINT, UDINT)	
nUMin	UDINT	Zur Anbindung an den Datenpunkt (USINT, UINT, UDINT)	
nUMax	UDINT	Zur Anbindung an den Datenpunkt (USINT, UINT, UDINT)	
rValue	LREAL	Zur Anbindung an den Datenpunkt (REAL, LREAL)	
rMin	LREAL	Zur Anbindung an den Datenpunkt (REAL, LREAL)	
rMax	LREAL	Zur Anbindung an den Datenpunkt (REAL, LREAL)	
nColor	UINT	Zur Anbindung an den Datenpunkt	
blInputCompleted	BOOL	Zur Anbindung an den Datenpunkt	
nStatus	UINT	Zur Anbindung an den Datenpunkt	

Je nach Datentyp kann „nValue“, „nUValue“ oder „rValue“ angebunden werden.

#### 3.2.22.2 BrbVc4HandleNumericInputEx

```
plcbit BrbVc4HandleNumericInputEx(struct BrbVc4NumericEx_TYP* pNumeric, unsigned long pSourceValue,  
enum BrbVc4NumericDatatypes_ENUM eSourceDatatype)
```

#### Argumente:

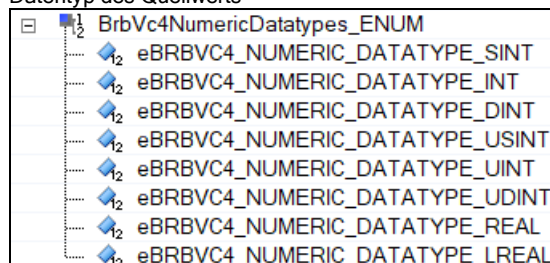
`struct BrbVc4NumericEx_TYP* pNumeric`

Zeiger auf die Instanz

`UDINT` pSourceValue  
Zeiger auf den Quellwert

`enum` BrbVc4NumericDatatypes\_ENUM eSourceDatatype

Datentyp des Quellwerts



### Rückgabe:

`BOOL`

0=Keine Eingabe  
1=Eingabe abgeschlossen

### Beschreibung:

Diese Funktion behandelt die Eingabe-Logik eines Numeric-Controls. Voraussetzung ist, dass der Datenpunkt „bInputCompleted“ bei Abschluss der Eingabe auf „1“ gesetzt wird, sowie der Datenpunkt „nValue“ (bei SINT, INT oder DINT), „nUValue“ (bei USINT, UINT oder UDINT) oder „rValue“ (bei REAL oder LREAL) angeschlossen ist.

Bei Abschluss einer Eingabe wird der eingegebene Wert auf die Quelle kopiert, ansonsten wird die Quelle auf den Datenpunkt kopiert.

Im Gegensatz zur obigen, normalen Implementierung ‚BrbVc4HandleNumericInput‘ werden hier auch die Datentypen UDINT und LREAL korrekt unterstützt.

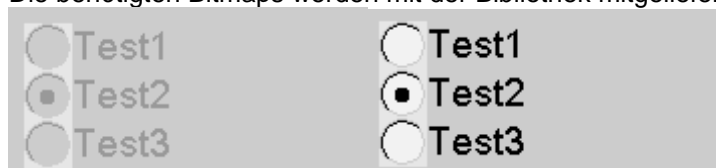
### Beispiel für den Aufruf:

```
BrbVc4HandleNumericInputEx(&Vis.PageControls.numTestEx, (UDINT) &gPar.nSpeed,
eBRBVC4_NUMERIC_DATATYPE_UDINT);
```

## 3.2.23 Optionbox

Es gibt kein fertiges Optionbox-Control. Hier wird es durch die Kombination von drei Controls realisiert (Bitmap, Text, Hotspot).

Die benötigten Bitmaps werden mit der Bibliothek mitgeliefert:



### 3.2.23.1 Struktur

BrbVc4Optionbox_TYP			
bClicked	BOOL	<input type="checkbox"/>	Zur Anbindung an den Datenpunkt
nBmplIndex	UINT	<input type="checkbox"/>	Zur Anbindung an den Datenpunkt
nTextColor	UINT	<input type="checkbox"/>	Zur Anbindung an den Datenpunkt
nStatus	UINT	<input type="checkbox"/>	Zur Anbindung an den Datenpunkt
bChecked	BOOL	<input type="checkbox"/>	Ausgang: 1=Angehakt

### 3.2.23.2 BrbVc4HandleOptionbox

```
plcbit BrbVc4HandleOptionbox(struct BrbVc4Optionbox_TYP* pOptionbox)
```

Argumente:

`struct` BrbVc4Optionbox\_TYP\* pOptionbox  
Zeiger auf die Instanz

Rückgabe:

`BOOL`  
0=Optionsbox wurde gerade nicht angeklickt  
1=Optionsbox wurde gerade angewählt

Beschreibung:

Diese Funktion behandelt die Logik für eine aus drei Controls zusammen gesetzten Optionbox. Das Item „bClicked“ muss durch einen virtuellen Key vom Typ „SetDatapoint“ auf 1 gesetzt werden. Das Item „nBmpIndex“ wird automatisch gesetzt. Dabei wird folgende Projektierung vorausgesetzt:  
0=Nicht angehakt und Disabled (Ausgegrautes Bitmap)  
1=Angehakt und Disabled (Ausgegrautes Bitmap)  
2=Nicht angehakt und Enabled (Normales Bitmap)  
3=Angehakt und Enabled (Normales Bitmap)

Das Item „nTextColor“ wird automatisch gesetzt.

Wenn mehrere Optionsboxen als Gruppe verwendet werden, in der nur eine gesetzt sein darf, muss das Rücksetzen der nicht gewählten applikativ gemacht werden (siehe Beispiel).

Beispiel für den Aufruf:

```
if (BrbVc4HandleOptionbox(&Vis.PageControls.optTest1) == 1)
{
    Vis.PageControls.optTest1.bChecked = 1;
    Vis.PageControls.optTest2.bChecked = 0;
    Vis.PageControls.optTest3.bChecked = 0;
}
if (BrbVc4HandleOptionbox(&Vis.PageControls.optTest2) == 1)
{
    Vis.PageControls.optTest1.bChecked = 0;
    Vis.PageControls.optTest2.bChecked = 1;
    Vis.PageControls.optTest3.bChecked = 0;
}
if (BrbVc4HandleOptionbox(&Vis.PageControls.optTest3) == 1)
{
    Vis.PageControls.optTest1.bChecked = 0;
    Vis.PageControls.optTest2.bChecked = 0;
    Vis.PageControls.optTest3.bChecked = 1;
}
```

### 3.2.24 OptionboxButton

Es gibt kein fertiges Optionbox-Control. Hier wird es durch einen Button realisiert.

Die Projektierung ist schneller als mit der oben genannten Optionbox.

Die benötigten Bitmaps werden mit der Bibliothek mitgeliefert:



### 3.2.24.1 Struktur

BrbVc4OptionboxButton_TYP			
nBmpIndex	UINT	<input type="checkbox"/>	Zur Anbindung an den Datenpunkt
nTextIndex	UINT	<input type="checkbox"/>	Zur Anbindung an den Datenpunkt
nColor	UINT	<input type="checkbox"/>	Zur Anbindung an den Datenpunkt
bVisPushed	BOOL	<input type="checkbox"/>	Zur Anbindung an den Datenpunkt
bChecked	BOOL	<input type="checkbox"/>	Ausgang: 1=Angehakt
nStatus	UINT	<input type="checkbox"/>	Zur Anbindung an den Datenpunkt
eToggleState	BrbVc4CheckboxButtonStates_ENUM	<input type="checkbox"/>	Ausgang: Momentaner Status des Buttons
bVisPushedOld	BOOL	<input type="checkbox"/>	Interne Variable
bCheckedOld	BOOL	<input type="checkbox"/>	Interne Variable

### 3.2.24.2 BrbVc4HandleOptionboxButton

```
plcbit BrbVc4HandleOptionboxButton(struct BrbVc4OptionboxButton_TYP* pButton)
```

#### Argumente:

```
struct BrbVc4OptionboxButton_TYP* pButton  
Zeiger auf die Instanz
```

#### Rückgabe:

```
BOOL  
0=Optionbox wurde nicht geändert  
1=Optionbox wurde geändert
```

#### Beschreibung:

Diese Funktion behandelt die Logik einer Optionbox als Button.

Das Item „bVisPushed“ muss durch einen virtuellen Key vom Typ „ToggleDatapoint“ gesetzt werden.

Das Item „bChecked“ kann zum Auswerten oder Umschalten des Signals im Programm verwendet werden.

Das Item „eToggleState“ liefert den momentanen Zustand durch eine Enumeration:

BrbVc4OptionboxButtonStates_ENUM	
eBRBVC4_OPTBTNSTATE_UNPUSHED	0=Nicht gedrückt
eBRBVC4_OPTBTNSTATE_UNPUSHED_EDG	1=Gerade losgelassen
eBRBVC4_OPTBTNSTATE_PUSHED_EDGE	2=Gerade gedrückt
eBRBVC4_OPTBTNSTATE_PUSHED	3=Gedrückt

Das Item „nBmpIndex“ wird automatisch gesetzt. Dabei wird folgende Projektierung vorausgesetzt:

```
0=Nicht angehakt und Disabled (Ausgegrautes Bitmap)  
1=Angehakt und Disabled (Ausgegrautes Bitmap)  
2=Nicht angehakt und Enabled (Normales Bitmap)  
3=Angehakt und Enabled (Normales Bitmap)
```

Beispiel für den Aufruf:

```
if (BrbVc4HandleOptionboxButton(&Vis.PageVc4.optbtnTest4) == 1)
{
    Vis.PageVc4.optbtnTest4.bChecked = 1;
    Vis.PageVc4.optbtnTest5.bChecked = 0;
    Vis.PageVc4.optbtnTest6.bChecked = 0;
}
if (BrbVc4HandleOptionboxButton(&Vis.PageVc4.optbtnTest5) == 1)
{
    Vis.PageVc4.optbtnTest4.bChecked = 0;
    Vis.PageVc4.optbtnTest5.bChecked = 1;
    Vis.PageVc4.optbtnTest6.bChecked = 0;
}
if (BrbVc4HandleOptionboxButton(&Vis.PageVc4.optbtnTest6) == 1)
{
    Vis.PageVc4.optbtnTest4.bChecked = 0;
    Vis.PageVc4.optbtnTest5.bChecked = 0;
    Vis.PageVc4.optbtnTest6.bChecked = 1;
}
```

### 3.2.25 Password

Dieses Control besitzt keine dazugehörige Funktion. Der Datentyp dient lediglich der strukturierten Anbindung der Datenpunkte.

#### 3.2.25.1 Struktur

BrbVc4Password_TYP			
sPasswords	STRING[nBRBVC4_PASSWORD_CHAR_MAX][0..nBRBVC4_PASSWORD_INDEX_MAX]	<input type="checkbox"/>	Passwörter
nLevel	UINT	<input type="checkbox"/>	Aktuelle Benutzer-Ebene
nColor	UINT	<input type="checkbox"/>	Zur Anbindung an den Datenpunkt
blnInputCompleted	BOOL	<input type="checkbox"/>	Zur Anbindung an den Datenpunkt
nStatus	UINT	<input type="checkbox"/>	Zur Anbindung an den Datenpunkt

### 3.2.26 PieChart

Dieses Control besitzt keine dazugehörige Funktion. Der Datentyp dient lediglich der strukturierten Anbindung der Datenpunkte.

#### 3.2.26.1 Struktur

BrbVc4PieChart_TYP			
nValue	DINT[0..nBRBVC4_PIECHART_VALUE_INDEX_MAX]	<input type="checkbox"/>	Zur Anbindung an den Datenpunkt
nColor	UINT	<input type="checkbox"/>	Zur Anbindung an den Datenpunkt
nStatus	UINT	<input type="checkbox"/>	Zur Anbindung an den Datenpunkt

### 3.2.27 Scale

Dieses Control besitzt keine dazugehörige Funktion. Der Datentyp dient lediglich der strukturierten Anbindung der Datenpunkte.

#### 3.2.27.1 Struktur

BrbVc4Scale_TYP			
nValue	DINT	<input type="checkbox"/>	Zur Anbindung an den Datenpunkt
nValueMin	DINT	<input type="checkbox"/>	Zur Anbindung an den Datenpunkt
nValueMax	DINT	<input type="checkbox"/>	Zur Anbindung an den Datenpunkt
blnInputCompleted	BOOL	<input type="checkbox"/>	Zur Anbindung an den Datenpunkt
nControlColor	UINT	<input type="checkbox"/>	Zur Anbindung an den Datenpunkt
nRangeColor	UINT	<input type="checkbox"/>	Zur Anbindung an den Datenpunkt
nStatus	UINT	<input type="checkbox"/>	Zur Anbindung an den Datenpunkt

### 3.2.28 Scroll-Listen

Es gibt keine fertigen scrollbaren List-Controls, welche über mehrere Spalten in einer Zeile verfügen. Sie müssen über eigens projektierte Controls, welche dann den Ausschnitt der Quelle wiedergeben, realisiert werden.

Optional, aber hilfreich im Umgang mit Scroll-Listen ist diese Struktur:

BrbVc4ScrollList_TYP			
bGetList	BOOL	<input type="checkbox"/>	1=Liste wurde gescrollt und muss neu angezeigt werden
nScrollOffset	DINT	<input type="checkbox"/>	Momentaner Offset des angezeigten Ausschnitts der Quelle
nSelectedIndex	DINT	<input type="checkbox"/>	Index der momentan selektierten Zeile

Für das Realisieren einer vertikalen oder horizontalen Scrollbar gibt es weiter unten auch Strukturen. Normalerweise gibt es auf jeder Seite maximal nur eine scrollbare Liste. Deshalb ist es ausreichend, nur eine Instanz der vertikalen und horizontalen Scrollbar zu haben. Dies verringert den Projektierungsaufwand wesentlich. Der Einfachkeit halber wird auch ein optionaler Typ zur Verfügung gestellt, der beide Scrollbars beinhaltet:



BrbVc4Scrollbar_TYP		
Horizontal	BrbVc4ScrollbarHor_TYP	<input type="checkbox"/>
Vertical	BrbVc4ScrollbarVer_TYP	<input type="checkbox"/>

### 3.2.29 ScrollbarHorizontal

Es gibt keine fertigen Scrollbars. Hier wird sie durch die Kombination mehreren normaler bzw. Inc-Buttons realisiert.

Die benötigten Bitmaps werden mit der Bibliothek mitgeliefert:



#### 3.2.29.1 Struktur

BrbVc4ScrollbarHor_TYP			
bDisabled	BOOL	<input type="checkbox"/>	Parameter: 1=Disabled
nTotalIndexMin	DINT	<input type="checkbox"/>	Parameter: Kleinster Index des Quelle
nTotalIndexMax	DINT	<input type="checkbox"/>	Parameter: Größter Index des Quelle
nCountShow	UDINT	<input type="checkbox"/>	Parameter: Anzahl der angezeigten Zeilen
nEntryCountTotal	UDINT	<input type="checkbox"/>	Kann zur Anzeige der totalen Einträge verwendet werden
nScrollTotal	UDINT	<input type="checkbox"/>	Interne Variable
btnLeft	BrbVc4Button_TYP	<input type="checkbox"/>	Control
btnPageLeft	BrbVc4IncButton_TYP	<input type="checkbox"/>	Control
btnLineLeft	BrbVc4IncButton_TYP	<input type="checkbox"/>	Control
btnLineRight	BrbVc4IncButton_TYP	<input type="checkbox"/>	Control
btnPageRight	BrbVc4IncButton_TYP	<input type="checkbox"/>	Control
btnRight	BrbVc4Button_TYP	<input type="checkbox"/>	Control
bScrollDone	BOOL	<input type="checkbox"/>	Ausgang: 1=Es wurde gescrollt

#### 3.2.29.2 BrbVc4HandleScrollbarHorizontal

```
plcbit BrbVc4HandleScrollbarHorizontal(struct BrbVc4ScrollbarHor_TYP* pScrollbar, signed long* pScrollOffset)
```

##### Argumente:

`struct BrbVc4ScrollbarHor_TYP* pScrollbar`  
Zeiger auf die Instanz  
`DINT* pScrollOffset`  
Zeiger auf die Variable mit dem Scroll-Offset

##### Rückgabe:

`BOOL`

0= Es wurde nicht gescrollt  
1=Es wurde gescrollt

##### Beschreibung:

Diese Funktion behandelt die Logik für eine aus mehreren Buttons zusammen gesetzten Scrollbar.

Die Items der beinhalteten Buttons müssen entsprechend der Deklaration projiziert werden (siehe normaler Button und IncButton).

Durch das Item „bDisabled“ kann die gesamte Leiste ausgegraut werden.

Das Item „nBmpIndex“ der Buttons wird dabei automatisch gesetzt. Dabei wird folgende Projektierung vorausgesetzt:

0=Disabled (Ausgegrautes Bitmap)

1=Enabled (Normales Bitmap)

Die Funktion berechnet den maximalen Scroll-Offset gemäß der übergebenen Parameter der Quelle und des angezeigten Ausschnitts. Außerdem werden die Buttons ausgegraut, welche zu drücken keinen Sinn macht (z.B. wenn sich der Ausschnitt ganz links befindet, kann nicht weiter nach links gescrollt werden).

Weiterhin wird der aktuelle Offset gemäß der Button-Klicks berechnet und begrenzt. Er kann verwendet werden, um den Ausschnitt der Quelle neu anzuzeigen.

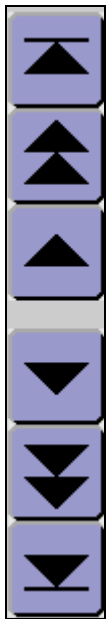
Beispiel für den Aufruf:

```
if(HandleScrollbarHorizontal(&Vis.Scrollbar.Hor, &Vis.PageControls.ScrollHorTest.nOffset) ==  
1)  
{  
    Vis.PageControls.ScrollHorTest.bGetList = 1;  
}
```

### 3.2.30 ScrollbarVertical

Es gibt keine fertigen Scrollbars. Hier wird sie durch die Kombination mehreren normaler bzw. Inc-Buttons realisiert.

Die benötigten Bitmaps werden mit der Bibliothek mitgeliefert:



#### 3.2.30.1 Struktur

BrbVc4ScrollbarVer_TYP			
bDisabled	BOOL	<input type="checkbox"/>	Parameter: 1=Disabled
nTotalIndexMin	DINT	<input type="checkbox"/>	Parameter: Kleinster Index des Quelle
nTotalIndexMax	DINT	<input type="checkbox"/>	Parameter: Größter Index des Quelle
nCountShow	UDINT	<input type="checkbox"/>	Parameter: Anzahl der angezeigten Zeilen
nEntryCountTotal	UDINT	<input type="checkbox"/>	Kann zur Anzeige der totalen Einträge verwendet werden
nScrollTotal	UDINT	<input type="checkbox"/>	Interne Variable
btnTop	BrbVc4Button_TYP	<input type="checkbox"/>	Control
btnPageUp	BrbVc4IncButton_TYP	<input type="checkbox"/>	Control
btnLineUp	BrbVc4IncButton_TYP	<input type="checkbox"/>	Control
btnLineDown	BrbVc4IncButton_TYP	<input type="checkbox"/>	Control
btnPageDown	BrbVc4IncButton_TYP	<input type="checkbox"/>	Control
btnBottom	BrbVc4Button_TYP	<input type="checkbox"/>	Control
bScrollDone	BOOL	<input type="checkbox"/>	Ausgang: 1=Es wurde gescrollt

#### 3.2.30.2 BrbVc4HandleScrollbarVertical

```
plcbit BrbVc4HandleScrollbarVertical(struct BrbVc4ScrollbarVer_TYP* pScrollbar, signed long*  
pScrollOffset)
```

Argumente:

```
struct BrbVc4ScrollbarVer_TYP* pScrollbar
```

Zeiger auf die Instanz  
`DINT*` `pScrollOffset`  
Zeiger auf die Variable mit dem Scroll-Offset

#### Rückgabe:

`BOOL`  
0= Es wurde nicht gescrollt  
1=Es wurde gescrollt

#### Beschreibung:

Diese Funktion behandelt die Logik für eine aus mehreren Buttons zusammen gesetzten Scrollbar. Die Items der beinhalteten Buttons müssen entsprechend der Deklaration projiziert werden (siehe normaler Button und IncButton).

Durch das Item „bDisabled“ kann die gesamte Leiste ausgegraut werden.

Das Item „nBmpIndex“ der Buttons wird dabei automatisch gesetzt. Dabei wird folgende Projektierung vorausgesetzt:

0=Disabled (Ausgegrautes Bitmap)  
1=Enabled (Normales Bitmap)

Die Funktion berechnet den maximalen Scroll-Offset gemäß der übergebenen Parameter der Quelle und des angezeigten Ausschnitts. Außerdem werden die Buttons ausgegraut, welche zu drücken keinen Sinn macht (z.B. wenn sich der Ausschnitt ganz oben befindet, kann nicht weiter nach oben gescrollt werden). Weiterhin wird der aktuelle Offset gemäß der Button-Klicks berechnet und begrenzt. Er kann verwendet werden, um den Ausschnitt der Quelle neu anzuzeigen.

#### Beispiel für den Aufruf:

```
if(HandleScrollbarVertical(&Vis.Scrollbar.Ver, &Vis.PageControls.ScrollVerTest.nOffset) == 1)
{
    Vis.PageControls.ScrollVerTest.bGetList = 1;
}
```

### 3.2.31 Shape

Dieses Control besitzt keine dazugehörige Funktion. Der Datentyp dient lediglich der strukturierten Anbindung der Datenpunkte.

#### 3.2.31.1 Struktur

BrbVc4Shape_TYP			
nColor	UINT	<input type="checkbox"/>	Zur Anbindung an den Datenpunkt
nStatus	UINT	<input type="checkbox"/>	Zur Anbindung an den Datenpunkt

### 3.2.32 Slider

Dieses Control besitzt keine dazugehörige Funktion. Der Datentyp dient lediglich der strukturierten Anbindung der Datenpunkte.

#### 3.2.32.1 Struktur

BrbVc4Slider_TYP			
nValueShow	DINT	<input type="checkbox"/>	Zur Anbindung an den Datenpunkt
nValueInput	DINT	<input type="checkbox"/>	Zur Anbindung an den Datenpunkt
nValueMin	DINT	<input type="checkbox"/>	Zur Anbindung an den Datenpunkt
nValueMax	DINT	<input type="checkbox"/>	Zur Anbindung an den Datenpunkt
bInputCompleted	BOOL	<input type="checkbox"/>	Zur Anbindung an den Datenpunkt
nColor	UINT	<input type="checkbox"/>	Zur Anbindung an den Datenpunkt
nStatus	UINT	<input type="checkbox"/>	Zur Anbindung an den Datenpunkt

### 3.2.33 String

Dieses Control besitzt keine dazugehörige Funktion. Der Datentyp dient lediglich der strukturierten Anbindung der Datenpunkte.

#### 3.2.33.1 Struktur

BrbVc4String_TYP			
sValue	STRING[nBRBVC4_STRING_INPUT_CHAR_MAX]	<input type="checkbox"/>	Zur Anbindung an den Datenpunkt
nColor	UINT	<input type="checkbox"/>	Zur Anbindung an den Datenpunkt
bInputCompleted	BOOL	<input type="checkbox"/>	Zur Anbindung an den Datenpunkt
nStatus	UINT	<input type="checkbox"/>	Zur Anbindung an den Datenpunkt

Diesen Datentypen gibt es auch für Unicode-String „BrbVc4WcString\_TYP“.

### 3.2.34 Tab-Control

Es gibt kein fertiges Tab-Control. Hier wird es durch die Kombination von mehreren Layern, Buttons und Shapes realisiert.

Normalerweise gibt es auf jeder Seite maximal nur ein Tab-Control. Deshalb ist es ausreichend, nur eine Instanz dieses Controls zu projektieren.

Die Anzahl der Reiter wurde auf maximal 8 beschränkt, was ausreichend sein sollte.

Die benötigten Border-Bitmaps werden mit der Bibliothek mitgeliefert:



#### 3.2.34.1 Struktur

BrbVc4TabPage_TYP			
btnPage	BrbVc4Button_TYP	<input type="checkbox"/>	Control
nStatus	UINT	<input type="checkbox"/>	Zur Anbindung an den Datenpunkt
BrbVc4TabCtrl_TYP			
nSelectedTabPageIndex	UINT	<input type="checkbox"/>	Ausgang: Selektierter Tab-Reiter
TabPage	BrbVc4TabPage_TYP[0..nBRBVC4_TAB_PAGE_INDEX_MAX]	<input type="checkbox"/>	Tab-Reiter

#### 3.2.34.2 BrbVc4HandleTabCtrl

```
plcbit BrbVc4HandleTabCtrl(struct BrbVc4TabCtrl_TYP* pTabCtrl)
```

##### Argumente:

```
struct BrbVc4TabCtrl_TYP* pTabCtrl  
Zeiger auf die Instanz
```

##### Rückgabe:

```
BOOL
```

0= Der Reiter wurde nicht gewechselt  
1=Der Reiter wurde gewechselt

##### Beschreibung:

Diese Funktion behandelt die Logik für ein aus mehreren Controls zusammen gesetzten Tab-Control.

Die Items der beinhalteten Buttons müssen entsprechend der Deklaration projektiert werden (siehe normaler Button).

Das Item „nStatus“ jedes Reiters muss an den Status-Datenpunkt eines eigenen, lokalen Layers angebunden werden. Die Schaltung der Sichtbarkeit jedes Reiters wird von der Funktion übernommen. Ebenso wird das Item „nSelectedTabPageIndex“ beim Wechsel durch einen Klick gesetzt.

##### Beispiel für den Aufruf:

```
BrbVc4HandleTabCtrl(&Vis.TabCtrl);  
if(Vis.TabCtrl.nSelectedTabPageIndex == 0)  
{
```

```
        // Code für diesen Reiter...
    }
    else if (Vis.TabCtrl.nSelectedTabPageIndex == 1)
    {
        // Code für diesen Reiter...
    }
    else if (Vis.TabCtrl.nSelectedTabPageIndex == 2)
    {
        // Code für diesen Reiter...
    }
}
```

### 3.2.34.3 BrbVc4SetTabPageInvisible

```
unsigned short BrbVc4SetTabPageInvisible(struct BrbVc4TabCtrl_TYP* pTabCtrl)
```

#### Argumente:

```
struct BrbVc4TabCtrl_TYP* pTabCtrl
    Zeiger auf die Instanz
```

#### Rückgabe:

```
UINT
    Immer 0
```




#### Beschreibung:

Diese Funktion setzt alle Tab-Pages eines Tab-Controls auf unsichtbar. Das ist hilfreich, wenn innerhalb einer Tab-Page ein weiteres Tab-Control sitzt, nämlich zum Ausblenden der unteren Tab-Pages.

### 3.2.35 Text

Dieses Control besitzt keine dazugehörige Funktion. Der Datentyp dient lediglich der strukturierten Anbindung der Datenpunkte.








#### 3.2.35.1 Struktur

BrbVc4Text_TYP			
	nIndex	UINT	<input type="checkbox"/> Zur Anbindung an den Datenpunkt
	nColor	UINT	<input type="checkbox"/> Zur Anbindung an den Datenpunkt
	nStatus	UINT	<input type="checkbox"/> Zur Anbindung an den Datenpunkt

### 3.2.36 ToggleButton

Ein Toggle-Button bietet die Funktion einer Umschaltung. Er ist aus Kompatibilitätsgründen zu alten Versionen enthalten. Bei der Neuerstellung einer Applikation sollte nur noch „ToggleButtonExt“ verwendet werden.

#### 3.2.36.1 Struktur

BrbVc4ToggleButton_TYP			
	nBmpIndex	UINT	<input type="checkbox"/> Zur Anbindung an den Datenpunkt
	nTextIndex	UINT	<input type="checkbox"/> Zur Anbindung an den Datenpunkt
	nColor	UINT	<input type="checkbox"/> Zur Anbindung an den Datenpunkt
	bPushed	BOOL	<input type="checkbox"/> Zur Anbindung an den Datenpunkt
	nStatus	UINT	<input type="checkbox"/> Zur Anbindung an den Datenpunkt
	eToggleState	BrbVc4ToggleButtonStates_ENUM	<input type="checkbox"/> Ausgang: Momentaner Status des Buttons
	bPushedOld	BOOL	<input type="checkbox"/> Interne Variable

#### 3.2.36.2 BrbVc4HandleToggleButton

```
plcbit BrbVc4HandleToggleButton(struct BrbVc4ToggleButton_TYP* pButton)
```

#### Argumente:

```
struct BrbVc4ToggleButton_TYP* pButton
    Zeiger auf die Instanz
```

#### Rückgabe:

BOOL

0=Umschaltung nicht geändert  
1= Umschaltung geändert

**Beschreibung:**

Diese Funktion behandelt die Logik eines Umschalt-Buttons.

Das Item „bPushed“ muss durch einen virtuellen Key vom Typ „ToggleDatapoint“ gesetzt werden.

Das Item „eToggleState“ liefert den momentanen Zustand durch eine Enumeration:

BrbVc4ToggleButtonStates_ENUM	
eBRBVC4_TOGBTNSTATE_UNPUSHED	0=Nicht gedrückt
eBRBVC4_TOGBTNSTATE_UNPUSHED_EDG	1=Gerade losgelassen
eBRBVC4_TOGBTNSTATE_PUSHED_EDGE	2=Gerade gedrückt
eBRBVC4_TOGBTNSTATE_PUSHED	3=Gedrückt

**Beispiel für den Aufruf:**

```
BrbVc4HandleToggleButton(&Vis.PageControls.btnToggle);  
if(Vis.PageControls.btnToggle.eToggleState == eBRBVC4_TOGBTNSTATE_PUSHED_EDGE)  
{  
    // Code...  
}  
else if(Vis.PageControls.btnToggle.eToggleState == eBRBVC4_TOGBTNSTATE_UNPUSHED_EDGE)  
{  
    // Code...  
}
```

### 3.2.37 ToggleButton Ext (erweitert)

Dieser Toggle-Button bietet eine erweiterte Funktion zum normalen Toggle-Button. Bei einer Neuerstellung einer Applikation sollte nur dieser verwendet werden.

**Achtung:** Er behebt auch einen Fehler, welchen beim normalen Toggle-Button nur in ganz bestimmten Fällen auftritt (siehe unten).

#### 3.2.37.1 Struktur

BrbVc4ToggleButtonExt_TYP			
nBmpIndex	UINT	<input type="checkbox"/>	Zur Anbindung an den Datenpunkt
nTextIndex	UINT	<input type="checkbox"/>	Zur Anbindung an den Datenpunkt
nColor	UINT	<input type="checkbox"/>	Zur Anbindung an den Datenpunkt
bVisPushed	BOOL	<input type="checkbox"/>	Zur Anbindung an den Datenpunkt
bPushed	BOOL	<input type="checkbox"/>	Zur Verwendung im Programm
nStatus	UINT	<input type="checkbox"/>	Zur Anbindung an den Datenpunkt
eToggleState	BrbVc4ToggleButtonStates_ENUM	<input type="checkbox"/>	Ausgang: Momentaner Status des Buttons
bVisPushedOld	BOOL	<input type="checkbox"/>	Interne Variable
bPushedOld	BOOL	<input type="checkbox"/>	Interne Variable

#### 3.2.37.2 BrbVc4HandleToggleButton

```
plcbit BrbVc4HandleToggleButtonExt(struct BrbVc4ToggleButtonExt_TYP* pButton)
```

**Argumente:**

struct BrbVc4ToggleButtonExt\_TYP\* pButton  
Zeiger auf die Instanz

**Rückgabe:**

BOOL

0=Umschaltung nicht von Visualisierung geändert  
1= Umschaltung von Visualisierung geändert

**Beschreibung:**

Diese Funktion behandelt die Logik eines Umschalt-Buttons.

Das Item „bVisPushed“ muss durch einen virtuellen Key vom Typ „ToggleDatapoint“ gesetzt werden.

Das Item „bPushed“ kann zum Auswerten oder Umschalten des Signals im Programm verwendet werden.

Das Item „eToggleState“ liefert den momentanen Zustand durch eine Enumeration:

BrbVc4ToggleButtonStates_ENUM		
eBRBVC4_TOGBTNSTATE_UNPUSHED		0=Nicht gedrückt
eBRBVC4_TOGBTNSTATE_UNPUSHED_EDG		1=Gerade losgelassen
eBRBVC4_TOGBTNSTATE_PUSHED_EDGE		2=Gerade gedrückt
eBRBVC4_TOGBTNSTATE_PUSHED		3=Gedrückt

Beispiel für den Aufruf:

```
BrbVc4HandleToggleButtonExt (&Vis.PageControls.btnToggleExt);  
if (Vis.PageControls.btnToggleExt.eToggleState == eBRBVC4_TOGBTNSTATE_PUSHED_EDGE)  
{  
    // Code...  
}  
else if (Vis.PageControls.btnToggleExt.eToggleState == eBRBVC4_TOGBTNSTATE_UNPUSHED_EDGE)  
{  
    // Code...  
}
```

### 3.2.37.3 Unterschied zum normalen Toggle-Button

Die Anbindung an einen virtuellen Key geschieht nun über das Item „bVisPushed“. Umschaltungen durch das Programm können weiterhin am Item „bPushed“ gemacht werden. Dadurch kann die Funktion unterscheiden, ob die Umschaltung durch einen Klick an der Visualisierung oder durch das Programm erfolgt. Der Rückgabewert ist nur noch 1, wenn die Umschaltung über die Visualisierung erfolgt ist. Das Item „eToggleState“ liefert wie bisher den Status der Umschaltung, auch wenn er über das Programm erfolgt.

**Achtung:** Beim normalen Toggle-Button konnte es in sehr seltenen Fällen zu einem Fehlverhalten kommen.

Dieser Code wurde benutzt, um z.B. ein Ventil anzuzeigen und zu schalten, welches ebenfalls von der Automatik geschaltet wird:

```
if (BrbVc4HandleToggleButton (&Vis.PageControls.btnToggle) == 1)  
{  
    // Button geklickt -> Zustand auf Ventil übernehmen  
    gControl.bValveOn = Vis.PageControls.btnToggle.bPushed;  
}  
else  
{  
    // Button nicht geklickt -> Zustand auf Button übernehmen  
    Vis.PageControls.btnToggle.bPushed = gControl.bValveOn;  
}
```

Wenn nun das Ventil „gControl.bValveOn“ in einem schnelleren Task geschaltet wurde, konnte es sporadisch zu einer Rückkopplung führen: Die Logik des Buttons erkannte das von außen geschaltete Signal nicht und setzte es bei dem nächsten Aufruf auf den noch in der Visualisierung bekannten Zustand! Dieses Fehlverhalten tritt mit dem erweiterten Toggle-Button nicht mehr auf, da dieser zwischen Visualisierungs- und Programm-Umschaltung unterscheiden kann.

### 3.2.38 Touchgrid

Ein Touchgrid kann verwendet werden, um die Auswahl und die Anzeige einer selektierten Zelle innerhalb eines Gitters einfach zu realisieren.

### 3.2.38.1 Struktur

BrbVc4Touchgrid_TYP		
bClickEnabled	BOOL	Eingang: 1=Klicks werden erkannt
bUseSyncTouchState	BOOL	Eingang: 1=Auswertung des synchronisierten TouchStates
bDrawGrid	BOOL	Eingang: 1=Gitter wird gezeichnet
bDrawMarker	BOOL	Eingang: 1=Markierung wird gezeichnet
nGridLeft	DINT	Eingang: Linke Koordinate
nGridTop	DINT	Eingang: Obere Koordinate
nCellWidth	UINT	Eingang: Breite einer Zelle
nIndexMaxX	UINT	Eingang: Maximaler Index der Zellen in X-Richtung
nCellHeight	UINT	Eingang: Höhe einer Zelle
nIndexMaxY	UINT	Eingang: Maximaler Index der Zellen in Y-Richtung
nGridColor	UINT	Eingang: Farbe des Gitters
eMarkerFigure	BrbVc4Figures_ENUM	Eingang: Figur der Markierung (z.B. Rechteck)
MarkerLine	BrbVc4Line_TYP	Eingang: Markierung als Linie
MarkerRectangle	BrbVc4Rectangle_TYP	Eingang: Markierung als Rechteck
MarkerEllipse	BrbVc4Ellipse_TYP	Eingang: Markierung als Ellipse
MarkerArc	BrbVc4Arc_TYP	Eingang: Markierung als Bogen
MarkerText	BrbVc4DrawText_TYP	Eingang: Markierung als Text
nSelectedIndexX	UINT	Ausgang: Selektierte Spalte
nSelectedIndexY	UINT	Ausgang: Selektierte Zeile
nSelectedIndex	UINT	Ein-/Ausgang: Selektierter Index
nSelectedIndexMax	UINT	Ausgang: Maximal Selektier-Index
eState	BrbVc4TouchStates_ENUM	Ausgang: Momentaner Status des Touchs

### 3.2.38.2 BrbVc4HandleTouchgrid

```
unsigned short BrbVc4HandleTouchGrid(struct BrbVc4Touchgrid_TYP* pTouchgrid, struct
BrbVc4General_TYP* pGeneral)
```

#### Argumente:

`struct BrbVc4Touchgrid_TYP* pTouchgrid`  
Zeiger auf die Instanz

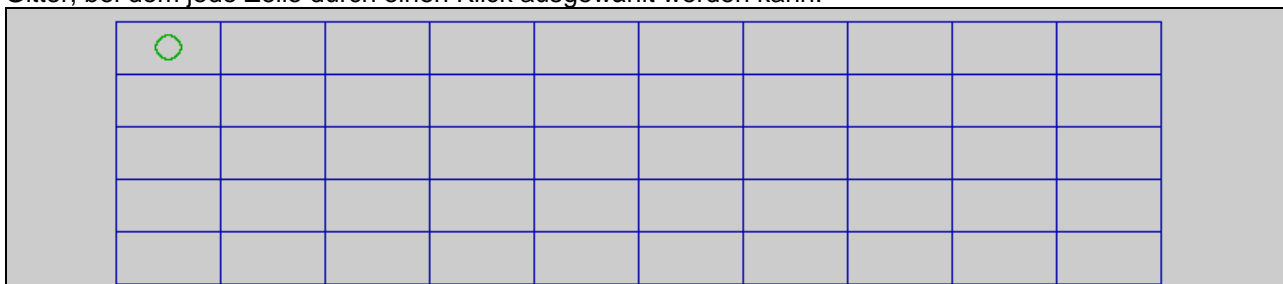
`struct BrbVc4General_TYP* pGeneral`  
Zeiger auf die Instanz von „BrbVc4General\_TYP“

#### Rückgabe:

`UINT`  
Status der intern verwendeten Funktionen der VisApi  
0= Ok

#### Beschreibung:

Diese Funktion behandelt die Logik eines Touchgrids. Es entspricht einem rechteckigen, gleichmäßigen Gitter, bei dem jede Zelle durch einen Klick ausgewählt werden kann:



Wenn man das Touchgrid über projizierte Elemente setzt und das Gitter nicht zeichnen lässt, kann die Selektierung und Markierung einfach und ohne viel zusätzliche Projektierung wie Shapes und Hotspots verwirklicht werden.

Durch die Eingänge kann das Gitter sowie auch die Markierung in der Optik vielfältig angepasst werden, z.B. kann die Figur der Markierung gewählt werden:



BrbVc4Figures_ENUM	
eBRBVC4_FIGURE_LINE	0=Linie
eBRBVC4_FIGURE_RECTANGLE	1=Rechteck
eBRBVC4_FIGURE_ELLIPSE	2=Ellipse
eBRBVC4_FIGURE_ARC	3=Bogen
eBRBVC4_FIGURE_TEXT	4=Text

Es gibt für jeden Typ eine Unterstruktur, welche das Aussehen der Markierung festlegt.

Der aktuelle Status des Touchs wird durch den Ausgang „eState“ wie in „BrbVc4General“ aufgeschlüsselt:

BrbVc4TouchStates_ENUM		
eBRBVC4_TOUCHSTATE_UNPUSHED	0	0=Nicht gedrückt
eBRBVC4_TOUCHSTATE_UNPUSHED_EDGE	1	1=Gerade losgelassen
eBRBVC4_TOUCHSTATE_PUSHED_EDGE	2	2=Gerade gedrückt
eBRBVC4_TOUCHSTATE_PUSHED	3	3=Gedrückt
eBRBVC4_TOUCHSTATE_DOUBLECLICK	4	4=Doppelklick (nur für einen Zyklus)

Das Item „nSelectedIndex“ wird bei einem Klick berechnet, kann aber auch programmseitig geändert werden. Es wird dann automatisch die Spalte und die Zeile berechnet und die Markierung gesetzt.







Durch den Eingang „bUseSyncTouchState“ kann festgelegt werden, dass statt „General.Touch.eState“ der State „General.Touch.eStateSync“ zur Auswertung des Touchs herangezogen wird. Das ist nur nötig, wenn der Aufruf des Touchgrids synchronisiert zum „General.nRedrawCounter“ gemacht wird, z.B. innerhalb eines selbst programmierten Controls.

## 3.3 Draw

In diesem Paket finden sich Funktionen zur einfacheren Behandlung selbstgezeichneter Elemente. Die Funktionen `VA_Saccess` und `VA_Srelease` müssen dabei vorher bzw. nachher selbst aufgerufen werden. So können mehrere Zeichenfunktionen in einem Block aufgerufen werden. Außerdem sollte das Zeichnen synchronisiert sein (siehe `BrbVc4General`).

### 3.3.1 Linie

#### 3.3.1.1 Struktur

BrbVc4Line_TYP			
	nLeft	DINT	<input type="checkbox"/> Eingang: Linke Koordinate
	nTop	DINT	<input type="checkbox"/> Eingang: Obere Koordinate
	nRight	DINT	<input type="checkbox"/> Eingang: Rechte Koordinate
	nBottom	DINT	<input type="checkbox"/> Eingang: Untere Koordinate
	nColor	UINT	<input type="checkbox"/> Eingang: Farbe
	nDashWidth	UINT	<input type="checkbox"/> Eingang: Breite für Strichelung (0=Solide)

#### 3.3.1.2 BrbVc4DrawLine

```
unsigned short BrbVc4DrawLine(struct BrbVc4Line_TYP* pLine, struct BrbVc4General_TYP* pGeneral)
```

##### Argumente:

```
struct BrbVc4Line_TYP* pLine  
    Zeiger auf die Instanz  
struct BrbVc4General_TYP* pGeneral  
    Zeiger auf die Instanz von „BrbVc4General_TYP“
```

##### Rückgabe:

```
UINT  
    Status der intern verwendeten Funktionen der VisApi  
    0= Ok
```

##### Beschreibung:

Diese Funktion zeichnet eine Linie nach den angegebenen Werten. Es werden auch gestrichelte Linien unterstützt.

#### 3.3.1.3 BrbVc4DrawLineCorr

```
unsigned short BrbVc4DrawLineCorr(struct BrbVc4Line_TYP* pLine, struct BrbVc4General_TYP* pGeneral)
```

##### Argumente:

```
struct BrbVc4Line_TYP* pLine  
    Zeiger auf die Instanz  
struct BrbVc4General_TYP* pGeneral  
    Zeiger auf die Instanz von „BrbVc4General_TYP“
```

##### Rückgabe:

```
UINT  
    Status der intern verwendeten Funktionen der VisApi  
    0= Ok
```

##### Beschreibung:

Diese Funktion zeichnet eine Linie nach den angegebenen Werten (siehe `BrbVc4DrawLine`). Allerdings werden die Koordinaten der Linie vorher korrigiert (siehe `BrbVc4CorrectLine`). Die Korrektur wird mit einer Kopie gemacht, das Original bleibt unverändert. Diese Funktion braucht mehr Rechenleistung als die normale Zeichenfunktion, daher sollte sie nur dann angewendet werden, wenn eine Linie wegen dynamischer Koordinaten-Berechnung negative Koordinaten erhalten kann.

### 3.3.1.4 BrbVc4DrawLineClip

```
unsigned short BrbVc4DrawLineClip(struct BrbVc4Line_TYP* pLine, struct BrbVc4Rectangle_TYP* pClip, struct BrbVc4General_TYP* pGeneral)
```

#### Argumente:

```
struct BrbVc4Line_TYP* pLine  
    Zeiger auf die Instanz  
struct BrbVc4Rectangle_TYP* pClip  
    Zeiger auf den Ausschnitt  
struct BrbVc4General_TYP* pGeneral  
    Zeiger auf die Instanz von „BrbVc4General_TYP“
```

#### Rückgabe:

```
UINT  
    Status der intern verwendeten Funktionen der VisApi  
    0= Ok
```

#### Beschreibung:








Diese Funktion zeichnet eine Linie nach den angegebenen Werten (siehe BrbVc4DrawLine).

Allerdings werden die Koordinaten der Linie vorher korrigiert (siehe BrbVc4ClipLine). Die Korrektur wird mit einer Kopie gemacht, das Original bleibt unverändert.

Diese Funktion braucht mehr Rechenleistung als die normale Zeichenfunktion, daher sollte sie nur dann angewendet werden, wenn eine Linie lediglich innerhalb eines Ausschnitts sichtbar sein soll, wegen dynamischer Koordinaten-Berechnung aber Koordinaten außerhalb des Ausschnitts erhalten kann.

## 3.3.2 Rechteck

### 3.3.2.1 Struktur

BrbVc4Rectangle_TYP			
	nLeft	DINT	<input type="checkbox"/> Eingang: Linke Koordinate
	nTop	DINT	<input type="checkbox"/> Eingang: Obere Koordinate
	nWidth	DINT	<input type="checkbox"/> Eingang: Breite
	nHeight	DINT	<input type="checkbox"/> Eingang: Höhe
	nFillColor	UINT	<input type="checkbox"/> Eingang: Füll-Farbe
	nBorderColor	UINT	<input type="checkbox"/> Eingang: Rand-Farbe
	nDashWidth	UINT	<input type="checkbox"/> Eingang: Breite für Strichelung (0=Solide)

### 3.3.2.2 BrbVc4DrawRectangle

```
unsigned short BrbVc4DrawRectangle(struct BrbVc4Rectangle_TYP* pRectangle, struct BrbVc4General_TYP* pGeneral)
```

#### Argumente:

```
struct BrbVc4Rectangle_TYP* pRectangle  
    Zeiger auf die Instanz  
struct BrbVc4General_TYP* pGeneral  
    Zeiger auf die Instanz von „BrbVc4General_TYP“
```

#### Rückgabe:

```
UINT  
    Status der intern verwendeten Funktionen der VisApi  
    0= Ok
```

#### Beschreibung:

Diese Funktion zeichnet ein Rechteck nach den angegebenen Werten. Es werden auch gestrichelte Rechtecke unterstützt. Bei nFillColor = 255 ist die Figur transparent.

### 3.3.2.3 BrbVc4DrawRectangleCorr

```
unsigned short BrbVc4DrawRectangleCorr(struct BrbVc4Rectangle_TYP* pRectangle, struct BrbVc4General_TYP* pGeneral)
```

#### Argumente:

```
struct BrbVc4Rectangle_TYP* pRectangle  
    Zeiger auf die Instanz  
struct BrbVc4General_TYP* pGeneral  
    Zeiger auf die Instanz von „BrbVc4General_TYP“
```

**Rückgabe:**

UINT

Status der intern verwendeten Funktionen der VisApi  
0= Ok

**Beschreibung:**

Diese Funktion zeichnet ein Rechteck nach den angegebenen Werten (siehe BrbVc4DrawRectangle). Allerdings werden die Koordinaten des Rechtecks vorher korrigiert (siehe BrbVc4CorrectRectangle). Die Korrektur wird mit einer Kopie gemacht, das Original bleibt unverändert.

Diese Funktion braucht mehr Rechenleistung als die normale Zeichenfunktion, daher sollte sie nur dann angewendet werden, wenn ein Rechteck wegen dynamischer Koordinaten-Berechnung negative Koordinaten erhalten kann.

### 3.3.2.4 BrbVc4DrawRectangleClip

```
unsigned short BrbVc4DrawRectangleClip(struct BrbVc4Rectangle_TYP* pRectangle, struct  
BrbVc4Rectangle_TYP* pClip, struct BrbVc4General_TYP* pGeneral)
```

**Argumente:**

```
struct BrbVc4Rectangle_TYP* pRectangle  
    Zeiger auf die Instanz  
struct BrbVc4Rectangle_TYP* pClip  
    Zeiger auf den Ausschnitt  
struct BrbVc4General_TYP* pGeneral  
    Zeiger auf die Instanz von „BrbVc4General_TYP“
```

**Rückgabe:**

UINT

Status der intern verwendeten Funktionen der VisApi  
0= Ok








**Beschreibung:**

Diese Funktion zeichnet ein Rechteck nach den angegebenen Werten (siehe BrbVc4DrawRectangle). Allerdings werden die Koordinaten des Rechtecks vorher korrigiert (siehe BrbVc4ClipRectangle). Die Korrektur wird mit einer Kopie gemacht, das Original bleibt unverändert.

Diese Funktion braucht mehr Rechenleistung als die normale Zeichenfunktion, daher sollte sie nur dann angewendet werden, wenn ein Rechteck lediglich innerhalb eines Ausschnitts sichtbar sein soll, wegen dynamischer Koordinaten-Berechnung aber Koordinaten außerhalb des Ausschnitts erhalten kann.

### 3.3.3 Ellipse

#### 3.3.3.1 Struktur

BrbVc4Ellipse_TYP			
	nLeft	DINT	<input type="checkbox"/> Eingang: Linke Koordinate
	nTop	DINT	<input type="checkbox"/> Eingang: Obere Koordinate
	nWidth	DINT	<input type="checkbox"/> Eingang: Breite
	nHeight	DINT	<input type="checkbox"/> Eingang: Höhe
	nFillColor	UINT	<input type="checkbox"/> Eingang: Füll-Farbe
	nBorderColor	UINT	<input type="checkbox"/> Eingang: Füll-Farbe
	nDashWidth	UINT	<input type="checkbox"/> Eingang: Breite für Strichelung (0=Solide)

#### 3.3.3.2 BrbVc4DrawEllipse

```
unsigned short BrbVc4DrawEllipse(struct BrbVc4Ellipse_TYP* pEllipse, struct BrbVc4General_TYP*  
pGeneral)
```

Argumente:

```
struct BrbVc4Ellipse_TYP* pEllipse  
    Zeiger auf die Instanz  
struct BrbVc4General_TYP* pGeneral  
    Zeiger auf die Instanz von „BrbVc4General_TYP“
```

Rückgabe:










```
UINT  
    Status der intern verwendeten Funktionen der VisApi  
    0= Ok
```

Beschreibung:

Diese Funktion zeichnet eine Ellipse nach den angegebenen Werten. Es werden auch gestrichelte Ellipsen unterstützt. Bei `nFillColor = 255` ist die Figur transparent.

### 3.3.4 Arc

#### 3.3.4.1 Struktur

BrbVc4Arc_TYP				
	nLeft	DINT	<input type="checkbox"/>	Eingang: Linke Koordinate
	nTop	DINT	<input type="checkbox"/>	Eingang: Obere Koordinate
	nWidth	DINT	<input type="checkbox"/>	Eingang: Breite
	nHeight	DINT	<input type="checkbox"/>	Eingang: Höhe
	rStartAngle	REAL	<input type="checkbox"/>	Eingang: Start-Winkel (0..360°)
	rEndAngle	REAL	<input type="checkbox"/>	Eingang: End-Winkel (0..360°)
	nFillColor	UINT	<input type="checkbox"/>	Eingang: Füll-Farbe (momentan nicht unterstützt)
	nBorderColor	UINT	<input type="checkbox"/>	Eingang: Rand-Farbe
	nDashWidth	UINT	<input type="checkbox"/>	Eingang: Breite für Strichelung (0=Solide)

#### 3.3.4.2 BrbVc4DrawArc

```
unsigned short BrbVc4DrawArc(struct BrbVc4Arc_TYP* pArc, struct BrbVc4General_TYP* pGeneral)
```

Argumente:

```
struct BrbVc4Arc_TYP* pArc  
    Zeiger auf die Instanz  
struct BrbVc4General_TYP* pGeneral  
    Zeiger auf die Instanz von „BrbVc4General_TYP“
```

Rückgabe:







```
UINT  
    Status der intern verwendeten Funktionen der VisApi  
    0= Ok
```

Beschreibung:

Diese Funktion zeichnet einen Ellipsen-Bogen nach den angegebenen Werten. Es werden auch gestrichelte Bögen unterstützt. Gefüllte Bögen werden momentan nicht unterstützt (Figur ist immer transparent).

### 3.3.5 Text

#### 3.3.5.1 Struktur

BrbVc4DrawText_TYP				
	nLeft	DINT	<input type="checkbox"/>	Eingang: Linke Koordinate
	nTop	DINT	<input type="checkbox"/>	Eingang: Obere Koordinate
	nFontIndex	UINT	<input type="checkbox"/>	Eingang: Index der Schrift
	nTextColor	UINT	<input type="checkbox"/>	Eingang: Text-Farbe
	nBackgroundColor	UINT	<input type="checkbox"/>	Eingang: Hintergrund-Farbe
	sText	STRING[nBRBVC4_DRAW_TEXT_CHAR_MAX]	<input type="checkbox"/>	Eingang: Text

### 3.3.5.2 BrbVc4DrawText

```
unsigned short BrbVc4DrawText(struct BrbVc4DrawText_TYP* pText, struct BrbVc4General_TYP* pGeneral)
```

#### Argumente:

```
struct Text* pText  
    Zeiger auf die Instanz  
struct BrbVc4General_TYP* pGeneral  
    Zeiger auf die Instanz von „BrbVc4General_TYP“
```

#### Rückgabe:

```
UINT  
    Status der intern verwendeten Funktionen der VisApi  
    0= Ok
```

#### Beschreibung:

Diese Funktion zeichnet einen Text nach den angegebenen Werten.

### 3.3.6 Font

#### 3.3.6.1 Struktur

BrbVc4Font_TYP			
nIndex	UINT	<input type="checkbox"/>	Parameter: Index der Schrift
nCharWidth	UINT	<input type="checkbox"/>	Parameter: Durchschnittliche Breite eines Zeichens in [Pixel]
nCharHeight	UINT	<input type="checkbox"/>	Parameter: Höhe eines Zeichens in [Pixel]

Diese Struktur wird benutzt, um Funktionen mit Informationen einer Schrift zur Darstellung von Text zu übergeben.

Da die Pixel-Ausmaße eines Textes nicht ermittelt werden können, können hier die durchschnittliche Breite eines Zeichens sowie die Höhe der Schrift angegeben werden. Dann können Texte auch zentriert oder rechtsbündig gezeichnet werden.

### 3.3.7 Hilfsfunktionen

#### 3.3.7.1 BrbVc4CorrectLine

```
unsigned short BrbVc4CorrectLine(struct BrbVc4Line_TYP* pLine)
```

#### Argumente:

```
struct BrbVc4Line_TYP* pLine  
    Zeiger auf die Linie
```

#### Rückgabe:

```
UINT  
    Immer 0
```

#### Beschreibung:

Die Zeichen-Funktionen der VisApi können keine negativen Koordinaten verarbeiten. Diese Funktion korrigiert die Koordinaten einer Linie ins Positive. Die Steigung der Linie bleibt dabei unverändert.

#### 3.3.7.2 BrbVc4ClipLine

```
unsigned short BrbVc4ClipLine(struct BrbVc4Line_TYP* pLine, struct BrbVc4Rectangle_TYP* pClip)
```

#### Argumente:

```
struct BrbVc4Line_TYP* pLine  
    Zeiger auf die Linie  
struct BrbVc4Rectangle_TYP* pClip  
    Zeiger auf den Ausschnitt
```

#### Rückgabe:

```
UINT  
    Immer 0
```

Beschreibung:

Diese Funktion korrigiert die Koordinaten einer Linie, damit sie nur innerhalb des angegebenen Ausschnitts dargestellt wird. Die Steigung der Linie bleibt dabei unverändert.

### 3.3.7.3 BrbVc4CorrectRectangle

```
unsigned short BrbVc4CorrectRectangle(struct BrbVc4Rectangle_TYP* pRectangle)
```

Argumente:

```
struct BrbVc4Rectangle_TYP* pRectangle  
    Zeiger auf das Rechteck
```

Rückgabe:

```
UINT  
    Immer 0
```

Beschreibung:

Die Zeichen-Funktionen der VisApi können keine negativen Koordinaten verarbeiten. Diese Funktion korrigiert die Koordinaten des Rechtecks ins Positive. Die Maße des Rechtecks bleiben dabei unverändert.

### 3.3.7.4 BrbVc4ClipRectangle

```
unsigned short BrbVc4ClipRectangle(struct BrbVc4Rectangle_TYP* pRectangle, struct  
BrbVc4Rectangle_TYP* pClip)
```

Argumente:

```
struct BrbVc4Rectangle_TYP* pRectangle  
    Zeiger auf das Rechteck  
struct BrbVc4Rectangle_TYP* pClip  
    Zeiger auf den Ausschnitt
```

Rückgabe:

```
UINT  
    Immer 0
```

Beschreibung:

Diese Funktion korrigiert die Koordinaten und die Maße eines Rechtecks, damit es nur innerhalb des angegebenen Ausschnitts dargestellt wird.

### 3.3.7.5 BrbVc4IsPointWithinRectangle

```
plcbit BrbVc4IsPointWithinRectangle(signed long nPointX, signed long nPointY, struct  
BrbVc4Rectangle_TYP* pRectangle)
```

Argumente:

```
DINT nPointX  
    X-Koordinate des Punkts  
DINT nPointY  
    Y-Koordinate des Punkts  
struct BrbVc4Rectangle_TYP* pRectangle  
    Zeiger auf das Rechteck
```

Rückgabe:

```
UINT  
    0= Punkt ist außerhalb des Rechtecks  
    1=Punkt ist innerhalb des Rechtecks
```

Beschreibung:

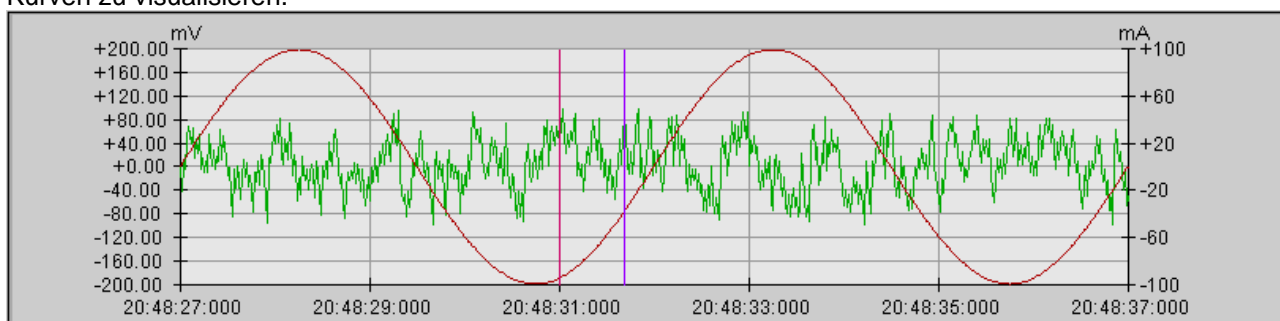
Diese Funktion gibt zurück, ob sich ein Punkt mit den angegebenen Koordinaten innerhalb des Rechtecks inklusive des Rands befindet.

## 3.4 DrawExt

In diesem Paket finden sich Funktionen zum Zeichnen komplexer Controls, wie z.B. ein Trend.

### 3.4.1 Trend

Mit dieser Funktion kann ein Trend dargestellt werden, um aufgenommene Werte in einer oder mehreren Kurven zu visualisieren.



Nachteile gegenüber dem Vc4-Trend-Control:

- Nur 4 Kurven
- Nur 2 Wert-Skalen (links und rechts)
- Werte müssen applikationsseitig aufgenommen werden

Vorteile gegenüber dem Vc4-Trend-Control:

- Sample-Auflösung ab 1µs möglich
- Einfache Parametrierung der Skalen
- Trend-Gitter wird automatisch an die Skalen angepasst
- Quelle kann auch ein Struktur-Array sein
- Quelle kann auch ein Ringpuffer sein
- Verschiedene Darstellungs-Möglichkeiten einer Kurve
- Einfacheres Zooming und Scrolling
- Implementierte Funktionen für Touch-Bedienung (Cursor setzen, Zoom und Scrolling)
- Einfache Erweiterung der visuellen Darstellung durch Callback-Funktionen

#### 3.4.1.1 Struktur

BrbVc4DrawTrend_TYP		
bEnable	BOOL	<input type="checkbox"/> 1=Aktiv
nRedrawCounterMatch	UINT	<input type="checkbox"/> Redraw-Cycle, bei dem gezeichnet wird
Cfg	BrbVc4DrawTrendCfg_TYP	<input type="checkbox"/> Konfiguration des Trends
State	BrbVc4DrawTrendState_TYP	<input type="checkbox"/> Status des Trends
Intern	BrbVc4DrawTrendIntern_TYP	<input type="checkbox"/> Interne berechnete Daten

Der Funktionsblock wird nur ausgeführt, wenn der Eingang „bEnable“ auf 1 ist. Dann wird in jedem Zyklus die Touch-Bedienung ausgewertet. Gezeichnet wird nur, wenn zusätzlich „General.nRedrawCounter“ (siehe „BrbVc4General“) den Wert von „nRedrawCounterMatch“ entspricht. Damit kann das Zeichnen von mehreren Trends auf einer Seite auf verschiedene CPU-Zyklen aufgeteilt werden, um so eine gleichmäßigere Verteilung der CPU-Belastung zu erreichen.



### 3.4.1.2 Konfiguration

BrbVc4DrawTrendCfg_TYP			Konfiguration des Trends
Drawbox	BrbVc4Drawbox_TYP	<input type="checkbox"/>	Angaben zur Drawbox
Padding	BrbVc4DrawPadding_TYP	<input type="checkbox"/>	Einrückung des Kurvenbereichs
nCurveAreaColor	UINT	<input type="checkbox"/>	Farbe des Kurvenbereichs
ScaleFont	BrbVc4Font_TYP	<input type="checkbox"/>	Eingang: Font der Skalierung
ScaleY	BrbVc4DrawTrendCfgScaleY_TYP[0..nBRBVC4_TREND_SCALE_Y_INDEX_MAX]	<input type="checkbox"/>	Konfiguration der Werte-Skalen
nSourceArrayIndexMax	DINT	<input type="checkbox"/>	Eingang: Maximaler Index der Quell-Arrays
ScaleX	BrbVc4DrawTrendCfgScaleX_TYP	<input type="checkbox"/>	Konfiguration der Zeit-Skala
TouchAction	BrbVc4DrawTrendCfgTouchAct_TYP	<input type="checkbox"/>	Konfiguration der Touch-Aktion
Curve	BrbVc4DrawTrendCfgCurve_TYP[0..nBRBVC4_TREND_CURVE_INDEX_MAX]	<input type="checkbox"/>	Konfiguration der Kurven
Cursor	BrbVc4DrawTrendCfgCursor_TYP[0..1]	<input type="checkbox"/>	Konfiguration der Cursor
Callbacks	BrbVc4DrawTrendCfgCallbacks_TYP	<input type="checkbox"/>	Konfiguration der Aufrufe
pTag	UDINT	<input type="checkbox"/>	Eingang: Zeiger auf Benutzer-Daten

Die Konfiguration ist der Übersichtlichkeit wegen in verschiedene Unter-Strukturen aufgeteilt.

#### 3.4.1.2.1 Allgemeines

BrbVc4Drawbox_TYP			
nLeft	UDINT	<input type="checkbox"/>	Eingang: Linke Koordinate
nTop	UDINT	<input type="checkbox"/>	Eingang: Obere Koordinate
nWidth	UDINT	<input type="checkbox"/>	Eingang: Breite
nHeight	UDINT	<input type="checkbox"/>	Eingang: Höhe
sFullName	STRING[nBRB_FILE_NAME_CHAR_MAX]	<input type="checkbox"/>	Pfad zur Anbindung
nBackgroundColor	UINT	<input type="checkbox"/>	Zur Anbindung an den Datenpunkt
nStatus	UINT	<input type="checkbox"/>	Zur Anbindung an den Datenpunkt

Die Angaben zur Drawbox sind korrekt auszufüllen, da nur dann alle Funktionalitäten richtig ausgeführt werden können. So werden z.B. die Koordinaten und Maße für die Touch-Funktionen benötigt. Der Name der Drawbox ist unbedingt auszufüllen, damit diese auch referenziert werden kann. Mit der Background-Color wird die Drawbox vor dem Zeichnen gelöscht.

BrbVc4DrawPadding_TYP			
nTop	DINT	<input type="checkbox"/>	Einrückung
nBottom	DINT	<input type="checkbox"/>	Eingang: Einrückung Oben
nLeft	DINT	<input type="checkbox"/>	Eingang: Einrückung Unten
nRight	DINT	<input type="checkbox"/>	Eingang: Einrückung Links
		<input type="checkbox"/>	Eingang: Einrückung Rechts

Das Padding legt die Einrückung des Kurvenbereichs fest. Es muss so gewählt werden, dass die Skalen genug Platz haben.

#### 3.4.1.2.2 ScaleY – Werte-Skalen

BrbVc4DrawTrendCfgScaleY_TYP				Konfiguration einer Trend-Wert-Skala
bShow	BOOL	<input type="checkbox"/>		Eingang: 1=Anzeigen
nColor	UINT	<input type="checkbox"/>		Eingang: Farbe der Skala
nLinesCount	UINT	<input type="checkbox"/>		Eingang: Anzahl der Skalenstriche
nLineLength	DINT	<input type="checkbox"/>		Eingang: Länge der Skalenstriche
sUnit	STRING[nBRBVC4_DRAW_TEXT_CHAR_MAX]	<input type="checkbox"/>		Eingang: Angezeigter Einheiten-Text
Grid	BrbVc4DrawTrendCfgGrid_TYP	<input type="checkbox"/>		Konfiguration des Gitters
rMin	REAL	<input type="checkbox"/>		Eingang: Unterer Skalenwert
rMax	REAL	<input type="checkbox"/>		Eingang: Oberer Skalenwert
nFractionDigits	UINT	<input type="checkbox"/>		Eingang: Anzahl der anzuzeigenden Nachkommastellen
BrbVc4DrawTrendCfgGrid_TYP				Konfiguration eines Trend-Gitters
bShow	BOOL	<input type="checkbox"/>		Eingang: 1=Gitter anzeigen
nColor	UINT	<input type="checkbox"/>		Eingang: Farbe des Gitters
nDashWidth	UINT	<input type="checkbox"/>		Eingang: Breite für Strichelung (0=Solide)

Der Einheitentext wird mittig oberhalb der Hauptlinie gezeichnet.

Der Zoom bzw. Scroll wird mit „rMin“ und „rMax“ festgelegt.

#### 3.4.1.2.3 SourceBuffer und nSourceArrayIndexMax

Ab V4.00 kann der Typ des Quell-Arrays eingestellt werden. Dazu gibt es folgende Unterstruktur:

BrbVc4DrawTrendCfgBuffer_TYP				Konfiguration des Trends
eBufferType	BrbVc4TrendBufferType_ENUM	<input type="checkbox"/>	<input checked="" type="checkbox"/>	Typ des Quell-Puffers (0=Normal, 1=Ring)
nSourceArrayIndexEnd	DINT	<input type="checkbox"/>	<input checked="" type="checkbox"/>	Eingang: End-Index des Quell-Arrays
bOverflow	BOOL	<input type="checkbox"/>	<input checked="" type="checkbox"/>	Eingang: Ringpuffer ist voll

Es gibt zwei Varianten, aus einem Quell-Array zu lesen:

BrbVc4TrendBufferType_ENUM		Typ des Quell-Puffers
eBRBVC4_TREND_BUFFERTYPE_NORMAL		Normal von 0 beginnend
eBRBVC4_TREND_BUFFERTYPE_RING		Ringpuffer

### 3.4.1.2.3.1 Normal

Diese Einstellung ist kompatibel zu Versionen vor V4.00.

Der Puffer beginnt immer bei 0. In „nSourceArrayIndexMax“ wird der maximale Index angegeben, bis zu dem das Quell-Array ausgewertet wird, also bis zu dem die aufgenommenen Punkte dargestellt werden. Wenn keine Werte vorhanden sind, muss er auf -1 gesetzt werden.

Bei einem laufenden Trend müssen, wenn der Puffer voll ist, alle Einträge um 1 nach oben geschoben werden und der neue Wert an unterster Stelle eingetragen werden.

### 3.4.1.2.3.2 Ring

Mit dieser Einstellung wird das Quell-Array als Ringpuffer definiert.

Der Eingang „nSourceArrayIndexMax“ muss den maximalen Index des Quell-Arrays enthalten, unabhängig davon, ab wo die Daten tatsächlich beginnen.

Der Eingang „nSourceArrayIndexEnd“ muss den Index des momentan letzten Wertes enthalten. Nach dem ersten Überlauf, wenn also der Ringpuffer voll ist, muss der Eingang „bOverflow“ auf 1 gesetzt werden und neue Werte können wieder ab Index 0 eingetragen werden.

Damit entfällt das manchmal performance-verschlingende Hochschieben der Daten um 1 Eintrag.

### 3.4.1.2.4 ScaleX – Zeit-Skala

BrbVc4DrawTrendCfgScaleX_TYP		Konfiguration der Trend-Zeit-Skala
bShow	BOOL	Eingang: 1=Anzeigen
nColor	UINT	Eingang: Farbe
nLinesCount	UINT	Eingang: Anzahl der Skalenstriche
nLineLength	DINT	Eingang: Länge der Skalenstriche
Grid	BrbVc4DrawTrendCfgGrid_TYP	Konfiguration des Gitters
sFormat	STRING[nBRB_TIME_TEXT_CHAR_MAX]	Eingang: Format der Zeit-Angabe (yyyy.mm.dd hh:MM:ss:mil)
dtStartTime	DTStructure	Eingang: Zeitstempel zum Beginn des Trends
nInterval	UDINT	Eingang: Intervall in [ms]
nZoomValueCount	UDINT	Eingang: Anzahl der anzuzeigenden Samplewerte auf die ganze Trendbreite
nScrollOffset	DINT	Eingang: Scroll-Angabe bei Zoom
bLimitScrollOffset	BOOL	Eingang: 1=Scroll-Offset wird begrenzt
BrbVc4DrawTrendCfgTouchAct_TYP		Konfiguration der Trend-Touch-Aktion
BrbVc4DrawTrendCfgGrid_TYP		Konfiguration eines Trend-Gitters
bShow	BOOL	<input type="checkbox"/> Eingang: 1=Gitter anzeigen
nColor	UINT	<input type="checkbox"/> Eingang: Farbe des Gitters
nDashWidth	UINT	<input type="checkbox"/> Eingang: Breite für Strichelung (0=Solide)

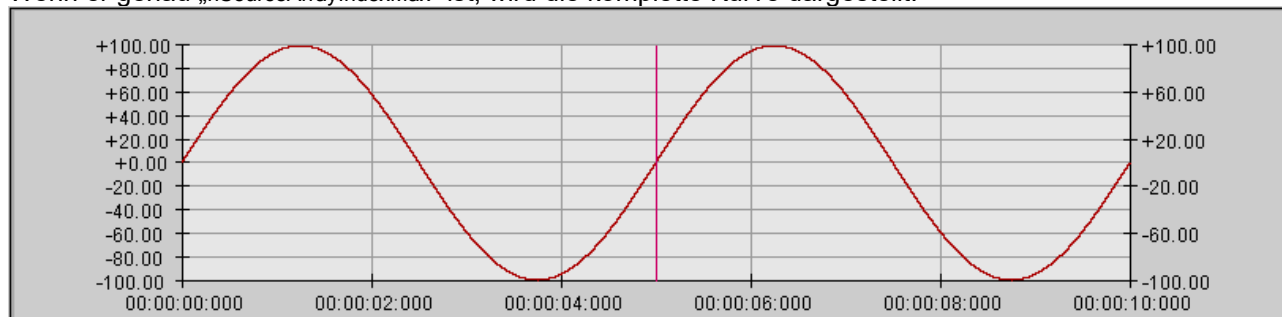
Das Format der Skala-Texte wird wie in der Funktion „BrbGetTimeText“ beschrieben festgelegt.

Zum Konvertieren eines anderen Zeitformats in die benötigte „DTStructure“ gibt es Funktionen in der AS-Bibliothek „AsTime“.

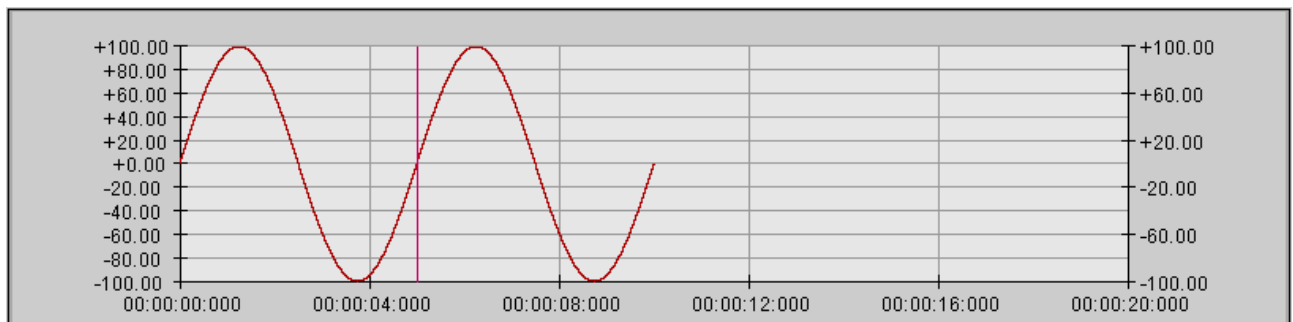
Der Zoom wird durch zwei Werte festgelegt:

Durch „nZoomValueCount“ wird die Anzahl der anzuzeigenden Sample-Werte auf die ganze Trendbreite gesetzt. Er muss  $\geq 1$  sein.

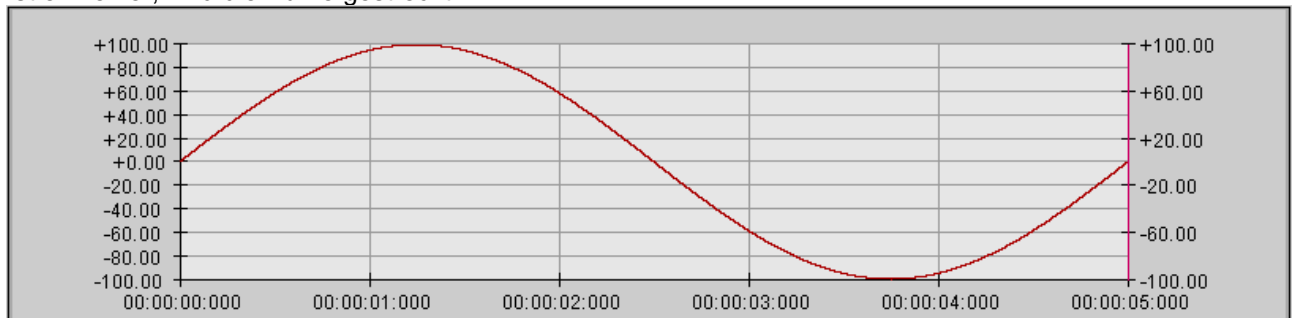
Wenn er genau „nSourceArrayIndexMax“ ist, wird die komplette Kurve dargestellt:



Ist er größer, wird die Kurve gestaucht:

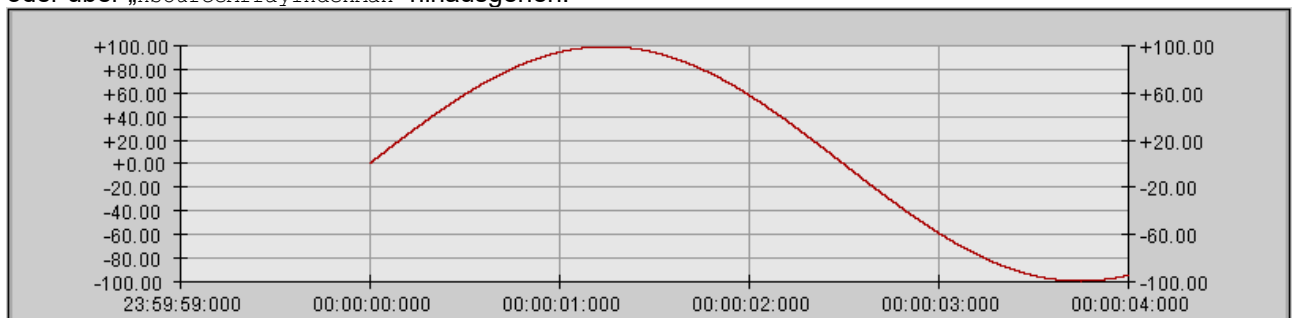


Ist er kleiner, wird die Kurve gestreckt:



Er muss  $\geq 1$ , darf aber auch größer als „nSourceArrayIndexMax“ sein.

Der Eingang „nScrollOffset“ verschiebt den angezeigten Ausschnitt um x Werte. Er darf auch negativ sein oder über „nSourceArrayIndexMax“ hinausgehen:



Mit dem Eingang „bLimitScrollOffset“ kann festgelegt werden, dass „nScrollOffset“ automatisch begrenzt wird, so dass nicht über die Kurve hinaus gescrollt werden kann. Das ist besonders bei Scrolling über den Touch sinnvoll.

### 3.4.1.2.5 TouchAction – Funktion des Touchs

BrbVc4DrawTrendCfgTouchAct_TYP			Konfiguration der Trend-Touch-Aktion	
BorderCorrection	BrbVc4DrawTouchBorderCorr_TYP	<input type="checkbox"/>	Touch-Korrektur des Rahmens	
bSetCursor0	BOOL	<input type="checkbox"/>	Eingang: 1=Cursor0 setzen	
bSetCursor1	BOOL	<input type="checkbox"/>	Eingang: 1=Cursor1 setzen	
bZoomX	BOOL	<input type="checkbox"/>	Eingang: 1=ZoomX aktivieren	
bZoomY	BOOL	<input type="checkbox"/>	Eingang: 1=ZoomY aktivieren	
bScrollX	BOOL	<input type="checkbox"/>	Eingang: 1=ScrollingX aktivieren	
bScrollY	BOOL	<input type="checkbox"/>	Eingang: 1=ScrollingY aktivieren	
BrbVc4DrawTouchBorderCorr_TYP			Touch-Korrektur des Rahmens	
nX	SINT	<input type="checkbox"/>	Eingang: OffsetX für die Touch-Korrektur des Rahmens	
nY	SINT	<input type="checkbox"/>	Eingang: OffsetY für die Touch-Korrektur des Rahmens	

Hier wird festgelegt, ob und welche Funktion durch den Touch bedienbar ist.

Mit der "BorderCorrection" kann ein Offset festgelegt werden, welcher die Koordinaten des Touchs aufgrund des Rahmens der Drawbox korrigiert. Der Rahmen „Flat\_back“ z.B. muss jeweils mit dem Wert 2 korrigiert werden, damit der Cursor auch exakt an dem berührten Punkt gesetzt wird.

Das Setzen der Cursor wird schon beim einmaligen Klicken ausgeführt.

Das Scrollen wird durch Verschiebe-Bewegung ausgelöst.

Beim Zoomen erscheint ein Fenster, welches auf den entsprechenden Ausschnitt gezogen werden kann.

Wenn Scroll und Zoom gemeinsam aktiviert sind, wird zur Unterscheidung ein Timer eingesetzt.

Wenn innerhalb 1 Sekunde der Touch verschoben wird, wird die Scroll-Funktion ausgeführt. Wenn der Touch 1 Sekunde auf Position gehalten wird, wird das Zoom-Fenster eingeblendet.

### 3.4.1.2.6 Curve – Kurven

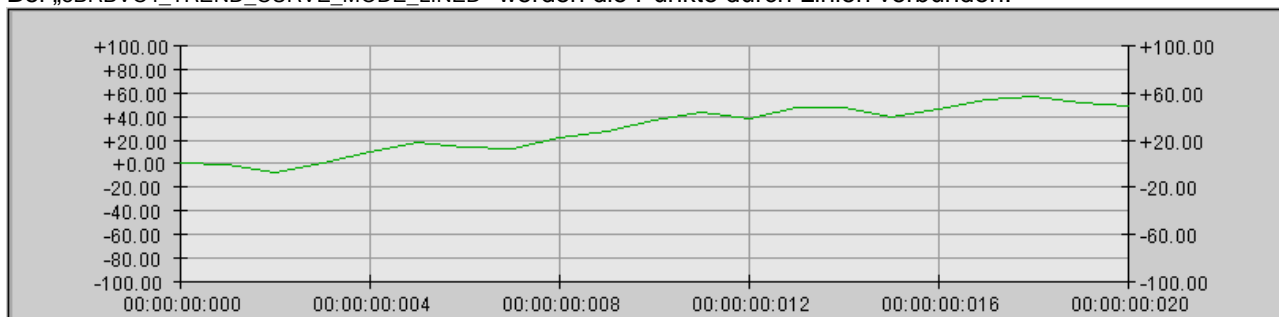
BrbVc4DrawTrendCfgCurve_TYP		Konfiguration einer Trend-Kurve
bShow	BOOL	Eingang: 1=Anzeigen
nColor	UINT	Eingang: Farbe
eScaleY	BrbVc4TrendScaleYIndex_ENUM	Eingang: Skalen-Zugehörigkeit (0=Links, 1=Rechts)
eMode	BrbVc4TrendCurveMode_ENUM	Eingang: Kurven-Zeichen-Modus
eValueSource	BrbVc4TrendSource_ENUM	Eingang: Quelle
eValueDatatype	BrbVc4TrendValueDatatype_ENUM	Eingang: Datentyp
pValueSource	UDINT	Eingang: Zeiger auf den Anfang der Quelle
nStructSize	UDINT	Eingang: Größe der Struktur bei Struktur-Array-Quelle
nStructMemberOffset	UDINT	Eingang: Offset des Wertes in der Struktur bei Struktur-Array-Quelle
rConversionFactor	REAL	Eingang: Faktor zur Umrechnung der Quell-Daten auf die eingestellte Skala
bCalculateStatistics	BOOL	Eingang: 1=Statistik-Werte berechnen

Der Eingang „eMode“ bezeichnet den Modus zum Zeichnen einer Kurve:

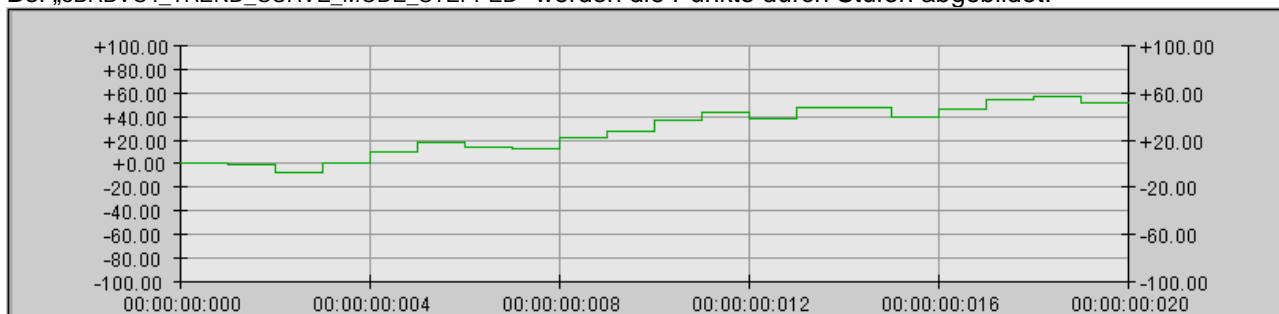
BrbVc4TrendCurveMode_ENUM		Kurven-Zeichen-Modus
eBRBVC4_TREND_CURVE_MODE_LINED		Linear
eBRBVC4_TREND_CURVE_MODE_STEPPED		Gestuft

Auffällig wird dies erst beim Anzeigen sehr weniger Punkte.

Bei „eBRBVC4\_TREND\_CURVE\_MODE\_LINED“ werden die Punkte durch Linien verbunden:



Bei „eBRBVC4\_TREND\_CURVE\_MODE\_STEPPED“ werden die Punkte durch Stufen abgebildet:



Der Eingang „eValueDatatype“ legt den Datentyp der Quell-Kurvenwerte fest:

BrbVc4TrendValueType_ENUM	Datentyp der Kurven-Werte
eBRBVC4_TREND_VALUE_DTYPE_REAL	REAL
eBRBVC4_TREND_VALUE_DTYPE_DINT	DINT
eBRBVC4_TREND_VALUE_DTYPE_INT	INT

Der Eingang „eValueSource“ legt die Strukturierung der Quell-Kurvenwerte fest:

BrbVc4TrendSource_ENUM	Quellen-Angabe 0..1
eBRBVC4_TREND_SOURCE_SINGLE_ARR	Einfaches Array
eBRBVC4_TREND_SOURCE_STRUCT_ARR	Struktur-Array

Wenn die Quelle ein Struktur-Array ist, müssen noch zwei zusätzliche Parameter übergeben werden:

Bei „nStructSize“ muss die Größe eines Eintrags übergeben werden, welcher einfach durch „sizeof(“ ermittelt werden kann.

Bei „nStructMemberOffset“ muss der Byte-Offset auf das Struktur-Item innerhalb der Struktur übergeben werden. Dieser kann bequem mit der Funktion `BrbGetStructMemberOffset` ermittelt werden, welche in der Bibliothek „BrbLib“ beschrieben ist.

Der Eingang „pValueSource“ ist der Zeiger auf den Anfang des Quell-Arrays.

Mit dem Eingang „rConversionFactor“ können die Rohdaten an die parametrisierte Skala angeglichen werden. Im Normalfall ist er „1.0“.

#### 3.4.1.2.7 Cursor

BrbVc4DrawTrendCfgCursor_TYP	Konfiguration eines Trend-Cursors	
bShow	BOOL	Eingang: 1=Anzeigen
nColor	UINT	Eingang: Farbe
nSampleIndex	DINT	Eingang: Position des Cursors

Es gibt zwei voneinander unabhängige Cursor.

Der Eingang „nSampleIndex“ legt die Position des Cursors im gesamten Aufzeichnungs-Bereich fest.

#### 3.4.1.2.8 Callbacks

Hinweis: Diese Funktionalität ist aufgrund von Funktionszeigern nur in ANSI-C nutzbar, aber nicht in IEC-Sprachen (siehe Punkt [Hinweise zu StructuredText und anderen IEC-Sprachen](#))

BrbVc4DrawTrendCfgCallbacks_TYP	Konfiguration der Trend-Aufrufe		
pCallbackAfterClear	UDINT	<input type="checkbox"/>	Eingang: Funktions-Zeiger für Aufruf nach Löschen der Drawbox
pCallbackAfterCurveArea	UDINT	<input type="checkbox"/>	Eingang: Funktions-Zeiger für Aufruf nach Zeichnen des Kurvenbereichs
pCallbackAfterScaleLin...	UDINT	<input type="checkbox"/>	Eingang: Funktions-Zeiger für Aufruf nach Zeichnen eines Werte-Skalierungs-Strichs
pCallbackAfterScaleY	UDINT	<input type="checkbox"/>	Eingang: Funktions-Zeiger für Aufruf nach Zeichnen einer Werte-Skala
pCallbackAfterScaleLin...	UDINT	<input type="checkbox"/>	Eingang: Funktions-Zeiger für Aufruf nach Zeichnen eines Zeit-Skalierungs-Strichs
pCallbackAfterScaleX	UDINT	<input type="checkbox"/>	Eingang: Funktions-Zeiger für Aufruf nach Zeichnen der Zeit-Skala
pCallbackAfterCurve	UDINT	<input type="checkbox"/>	Eingang: Funktions-Zeiger für Aufruf nach Zeichnen einer Kurve
pCallbackAfterCursor	UDINT	<input type="checkbox"/>	Eingang: Funktions-Zeiger für Aufruf nach Zeichnen eines Cursors
pCallbackAfterZoomWin...	UDINT	<input type="checkbox"/>	Eingang: Funktions-Zeiger für Aufruf nach Zeichnen des Zoom-Fensters
BrbVc4DrawTrendCfg_TYP	Konfiguration des Trends		

Ein Callback ist ein Aufruf einer vom Anwender geschriebenen Funktion während des Zeichnens.

Er arbeitet mit sogenannten Funktions-Zeigern. Dabei wird die Adresse einer Funktion übergeben, welche der Anwender selbst schreibt. Lediglich die Signatur, also die Anzahl, Reihenfolge und die Datentypen der Argumente sind dabei vorgeschrieben. Der Inhalt der Funktion bleibt vollkommen dem Anwender überlassen.

Es gibt 9 verschiedene Callbacks (siehe oben), welche nach dem Zeichnen des jeweiligen Elements aufgerufen werden können.

Für jeden Callback gibt es ein Muster der Signatur in der Datei „BrbVc4TrendCallbackTemplates.c“ in der Bibliothek:

```
unsigned short BrbVc4TrendCallbackAfterClear(struct BrbVc4DrawTrend_TYP* pTrend)
```

```
unsigned short BrbVc4TrendCallbackAfterCrveArea(struct BrbVc4DrawTrend_TYP* pTrend)
```

```
unsigned short BrbVc4TrendCallbackAfterScaleLineY(struct BrbVc4DrawTrend_TYP* pTrend,  
BrbVc4TrendScaleYIndex_ENUM eScaleYIndex, UINT nLineIndex, BrbVc4Line_TYP* pScaleLine,  
BrbVc4DrawText_TYP* pScaleText, REAL rScaleValue)
```

```
unsigned short BrbVc4TrendCallbackAfterScaleY(struct BrbVc4DrawTrend_TYP* pTrend,
BrbVc4TrendScaleYIndex_ENUM eScaleYIndex)

unsigned short BrbVc4TrendCallbackAfterScaleLineX(struct BrbVc4DrawTrend_TYP* pTrend, UINT
nLineIndex, BrbVc4Line_TYP* pScaleLine, BrbVc4DrawText_TYP* pScaleText, DINT nSampleIndex)

unsigned short BrbVc4TrendCallbackAfterScaleX(struct BrbVc4DrawTrend_TYP* pTrend)

unsigned short BrbVc4TrendCallbackAfterCurve(struct BrbVc4DrawTrend_TYP* pTrend, UINT nCurveIn-
dex)

unsigned short BrbVc4TrendCallbackAfterCursor(struct BrbVc4DrawTrend_TYP* pTrend, UINT nCursorIn-
dex)

unsigned short BrbVc4TrendCallbackAfterZoomWind(struct BrbVc4DrawTrend_TYP* pTrend)
```

Manche Callbacks werden während des Zeichnens mehrmals aufgerufen, so z.B. nach dem Zeichnen je-  
der Kurve. Über Argumente werden aktuelle Werte übergeben, z.B. die aktuelle Kurve.  
Im Callback kann auf die gesamte Trend-Struktur zugegriffen werden, auch auf die intern berechneten  
Daten. ACHTUNG: Es sollten aber keine Werte verändert werden!

Soll ein Callback aktiviert werden, so ist dessen Adresse in die obige Struktur einzutragen.

Beispiel:

Es wird eine Funktion angelegt, welche dem Muster entspricht:

```
unsigned short TrendCallbackAfterCurveArea(struct BrbVc4DrawTrend_TYP* pTrend)
{
    // Anwender-Code
    return 0;
}
```

Vor dem Aufruf der Trend-Funktion wird die Adresse des Callbacks übergeben

```
Trend.Cfg.Callbacks.pCallbackAfterCurveArea = (UDINT)&TrendCallbackAfterCurveArea;
```

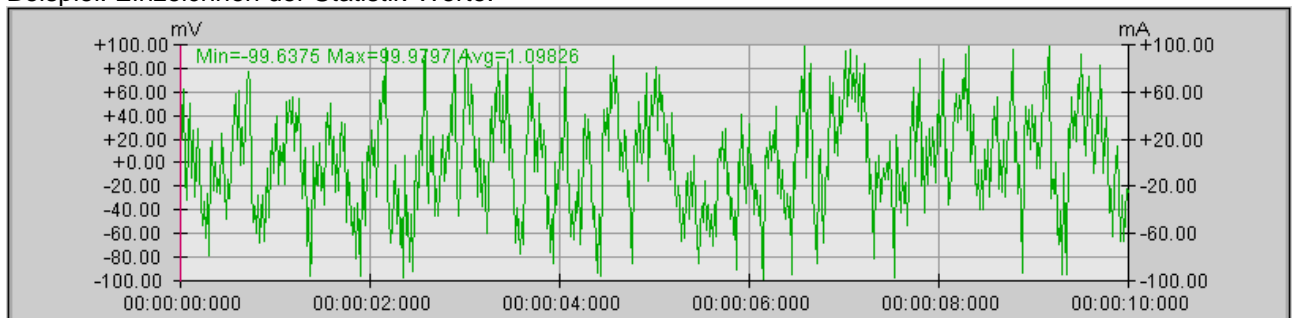
Innerhalb des Callbacks kann dann mit Zeichenfunktionen die visuelle Ausgabe erweitert werden, z.B.  
Texte oder zusätzliche Linien eingezeichnet werden.

Dadurch ergibt sich eine Vielfalt an Möglichkeiten, welche durch das herkömmliche Trend-Control nicht  
gegeben sind, z.B. könnte der Cursor-Kurven-Wert direkt neben dem Cursor ausgegeben oder die Statis-  
tikwerte an geeigneter Stelle eingeblendet werden.

Der Rückgabewert der Funktion ist egal.

ACHTUNG: Das Koordinaten-System bezieht sich auf die Trend-Drawbox, weil diese noch referenziert  
ist.

Beispiel: Einzeichnen der Statistik-Werte:



### 3.4.1.2.9 pTag

Dieser Zeiger wird von der Funktion nicht benutzt. Er kann vom Anwender als Zeiger auf Benutzer-Daten  
gesetzt und dann in den Callbacks verwendet werden.



### 3.4.1.3 Status

Diese Struktur enthält Ausgangs-Daten, welche für den Anwender interessant sein könnten:

BrbVc4DrawTrendState_TYP		Status des Trends
eTouchAction	BrbVc4TrendTouchAction_ENUM	Status einer Touch-Aktion
Curve	BrbVc4DrawTrendStateCurve_TYP[0..nBRBVC4_TREND_CURVE_INDEX_MAX]	Status einer Kurve

Der Ausgang „eTouchAction“ gibt den momentanen Stand der Touch-Bedienung an:

BrbVc4TrendTouchAction_ENUM	Touch-Aktionen des Trends
eBRBVC4_TREND_TOUCHACT_NONE	Keine Touch-Aktion des Trends
eBRBVC4_TREND_TOUCHACT_CURS_SET	Cursor wurde gesetzt
eBRBVC4_TREND_TOUCHACT_SCROLL	Momentan wird gescrollt
eBRBVC4_TREND_TOUCHACT_ZOOM_DRAG	Momentan wird das Zoom-Fenster gezogen
eBRBVC4_TREND_TOUCHACT_ZOOM_SET	Das Zoom-Fenster wurde gesetzt

Abhängig vom Status bleibt er nur einen oder auch mehrere Zyklen anstehen.

Die Statistik-Werte in der folgenden Struktur werden nur berechnet, wenn in der Konfiguration das Berechnen der Statistik-Werte eingeschaltet ist, ansonsten sind sie 0.

BrbVc4DrawTrendStateCurve_TYP		Status der Kurve
rValueMax	REAL	Ausgang: Maximaler Wert der Kurve
rValueMin	REAL	Ausgang: Minimaler Wert der Kurve
rValueAverage	REAL	Ausgang: Wert-Durchschnitt der Kurve
Cursor	BrbVc4DrawTrendStateCurveCur_TYP[0..1]	Status eines Cursors unter der Kurve

Hier werden auch die Kurven-Werte pro Cursor angeboten:

Hier werden auch die Kurven-Werte pro Cursor angeboten:		
	BrbVc4DrawTrendStateCurveCur_TYP	Status des Cursors unter der Kurve
	rValue REAL	Ausgang: Kurven-Wert unter dem Cursor
	dtsTimeStamp DTStructure	Ausgang: Zeitstempel unter dem Cursor

### 3.4.1.4 Intern

Diese Strukturen werden während des Aufrufs mit intern berechneten Daten gefüllt, z.B. Koordinaten und Maße einzelner Zeichen-Elemente. So ist z.B. der berechnete Kurvenbereich oder die Position des Zoom-Fensters enthalten.

Auf sie darf während eines Callbacks lesend zugegriffen werden, um eigene Elemente positionieren zu können. Auf keinen Fall sollten sie verändert werden!

### 3.4.1.5 BrbVc4DrawTrend

```
unsigned short BrbVc4DrawTrend(struct BrbVc4DrawTrend_TYP* pTrend, struct BrbVc4General_TYP* pGeneral)
```

#### Argumente:

```
struct BrbVc4DrawTrend_TYP* pTrend  
    Zeiger auf die Instanz  
struct BrbVc4General_TYP* pGeneral  
    Zeiger auf die Instanz von „BrbVc4General_TYP“
```

#### Rückgabe:

```
UINT  
    Immer 0
```

#### Beschreibung:

Zeichnet einen Trend nach Vorgaben. Der Aufruf sollte in der Restzeit-Task erfolgen. Es empfiehlt sich, die Instanz im Init des Tasks komplett auf 0 zu setzen.

### 3.4.1.6 BrbVc4GetTrendDisplayCoordinateY

```
signed long BrbVc4GetTrendDisplayCoordinateY(struct BrbVc4DrawTrend_TYP* pTrend,  
BrbVc4TrendScaleYIndex_ENUM eScaleY, float rValue)
```

#### Argumente:

```
struct BrbVc4DrawTrend_TYP* pTrend  
    Zeiger auf die Instanz
```

BrbVc4TrendScaleYIndex\_ENUM eScaleY  
Angabe der zugehörigen Skala  
REAL rValue  
Rohwert

**Rückgabe:**

DINT  
Y-Pixel-Koordinate

**Beschreibung:**

Gibt die Y-Pixel-Koordinate eines Trend-Wertes zurück.

### 3.4.1.7 BrbVc4GetTrendDisplayCoordinateX

signed long BrbVc4GetTrendDisplayCoordinateX(struct BrbVc4DrawTrend\_TYP\* pTrend, signed long nSampleIndex)

**Argumente:**

struct BrbVc4DrawTrend\_TYP\* pTrend  
Zeiger auf die Instanz  
DINT nSampleIndex  
Index des Samplewerts

**Rückgabe:**

DINT  
X-Pixel-Koordinate

**Beschreibung:**

Gibt die X-Pixel-Koordinate eines Trends aufgrund des Sample-Index zurück.

### 3.4.1.8 BrbVc4GetTrendSampleIndexByTime

signed long BrbVc4GetTrendSampleIndexByTime(struct BrbVc4DrawTrend\_TYP\* pTrend, struct DTStructure\* pTimeStamp)

**Argumente:**  
struct BrbVc4DrawTrend\_TYP\* pTrend  
Zeiger auf die Instanz  
struct DTStructure\* pTimeStamp  
Zeiger auf den Zeitstempel

**Rückgabe:**

DINT  
Sample-Index

**Beschreibung:**

Gibt den Sample-Index eines Trends aufgrund eines Zeitstempels zurück.

**Achtung:** Wenn der Zeitstempel vor dem Beginn des Trends liegt („pTrend->Cfg.ScaleX.dtsStartTime“), wird ein negativer Wert zurückgegeben. Wenn der Zeitstempel nach dem Ende des Trends liegt, wird ein positiver Wert zurückgegeben. In beiden Fällen darf mit dem Index nicht auf das Quellen-Array zugegriffen werden, weil er außerhalb des gültigen Bereichs ist!

### 3.4.1.9 BrbVc4GetTrendDisplayCoordXByTime

signed long BrbVc4GetTrendDispCoordXByTime(struct BrbVc4DrawTrend\_TYP\* pTrend, struct DTStructure\* pTimeStamp)

**Argumente:**

struct BrbVc4DrawTrend\_TYP\* pTrend  
Zeiger auf die Instanz  
struct DTStructure\* pTimeStamp  
Zeiger auf den Zeitstempel

**Rückgabe:**

DINT  
X-Pixel-Koordinate

**Beschreibung:**



Gibt die X-Pixel-Koordinate eines Trends aufgrund eines Zeitstempels zurück.

**Achtung:** Die Koordinate kann auch außerhalb des Kurvenbereichs liegen!

#### 3.4.1.10 BrbVc4GetTrendTimestampByIndex

```
plcdt BrbVc4GetTrendTimestampByIndex(struct BrbVc4DrawTrend_TYP* pTrend, signed long nSampleIndex, struct DTStructure* pTimeStamp)
```

##### Argumente:

<code>struct</code>	<code>BrbVc4DrawTrend_TYP*</code>	<code>pTrend</code>
	Zeiger auf die Instanz	
<code>DINT</code>	<code>nSampleIndex</code>	
	Index des Samplewerts	
<code>struct</code>	<code>DTStructure*</code>	<code>pTimeStamp</code>
	Zeiger auf den Zeitstempel	

##### Rückgabe:

<code>DATE_AND_TIME</code>	
Ergebnis als DATE_AND_TIME	
0, wenn Fehler (Null-Pointer)	

##### Beschreibung:

Gibt den Zeitstempel eines Trends aufgrund des Sample-Index zurück.

**Achtung:** Der Index kann auch außerhalb des Kurvenbereichs liegen!

### 3.4.2 TrendLink

Mit dieser Funktion können die Touch-Aktionen von bis zu 4 Trends synchronisiert werden. Das heißt, eine an einem Trend ausgeführte Touch-Aktion wird auch an den gelinkten Trends ausgeführt.

#### 3.4.2.1 Struktur

BrbVc4LinkTrends_TYP			
bEnable	BOOL	<input type="checkbox"/>	1=Aktiv
Cfg	BrbVc4DrawTrendLinkCfg_TYP	<input type="checkbox"/>	Konfiguration des TrendLinks

Der Funktionsblock wird nur ausgeführt, wenn der Eingang „bEnable“ auf 1 ist.

#### 3.4.2.2 Konfiguration

BrbVc4DrawTrendLinkCfg_TYP			Konfiguration des TrendLinks
Trend	BrbVc4DrawTrendLinkCfgTrend_TYP[0..nBRBVC4_TRENDLINK_TREND_IDX_MAX]	<input type="checkbox"/>	Zeiger auf zu linkende Trends
bLinkSetCursor0	BOOL	<input type="checkbox"/>	Eingang: 1=Cursor0 setzen ist gelinkt
bLinkSetCursor1	BOOL	<input type="checkbox"/>	Eingang: 1=Cursor1 setzen ist gelinkt
bLinkZoomX	BOOL	<input type="checkbox"/>	Eingang: 1=ZoomX ist gelinkt
bLinkZoomY	BOOL	<input type="checkbox"/>	Eingang: 1=ZoomY ist gelinkt
bLinkScrollX	BOOL	<input type="checkbox"/>	Eingang: 1=ScrollingX ist gelinkt
bLinkScrollY	BOOL	<input type="checkbox"/>	Eingang: 1=ScrollingY ist gelinkt
BrbVc4DrawTrendLinkCfgTrend_TYP			Zeiger auf zu linkende Trends
pTrend	BrbVc4DrawTrend_TYP	<input checked="" type="checkbox"/>	Eingang: Zeiger auf zu linkenden Trend, 0=deaktiviert

Im Array „Trend“ muss für jeden der bis zu 4 zu linkenden Trends der Zeiger auf dessen Struktur übergeben werden.

Mit den Eingängen kann festgelegt werden, welche Touch-Aktionen gelinkt werden. Eine Aktion wird nur zwischen Trends gelinkt, an denen die Aktion ebenfalls aktiviert ist. So können die unterschiedlichsten Kombinationen parametrisiert werden.

Die Zeitleiste (also Start-Zeitstempel und Intervall) der gelinkten Trends sollten gleich sein.

Ebenso sinnvoll, aber nicht notwendig ist es, dass die „rMin“- und „rMax“-Werte der Y-Skalierungen aller gelinkten Trends gleich sind.

#### 3.4.2.3 BrbVc4LinkTrends

```
unsigned short BrbVc4LinkTrends(struct BrbVc4LinkTrends_TYP* pLinkTrends, struct BrbVc4General_TYP* pGeneral)
```

##### Argumente:

```
struct BrbVc4LinkTrends_TYP* pLinkTrends  
    Zeiger auf die Instanz  
struct BrbVc4General_TYP* pGeneral  
    Zeiger auf die Instanz von „BrbVc4General_TYP“
```

##### Rückgabe:

UINT

Immer 0

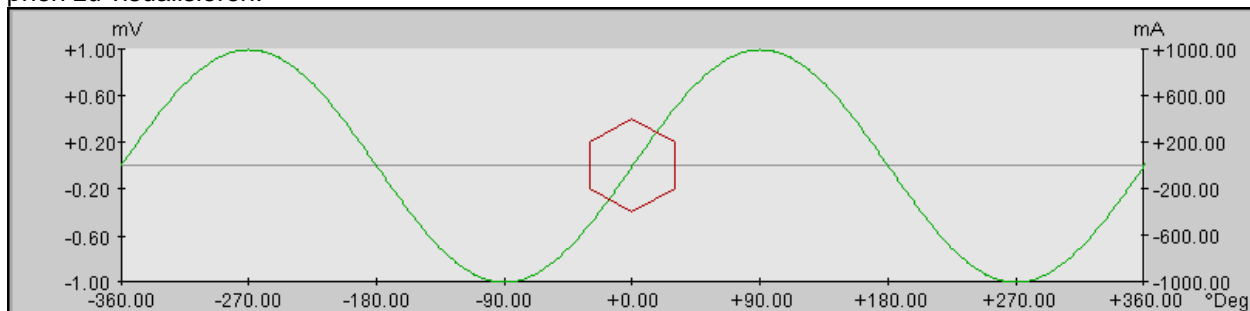
##### Beschreibung:

Synchronisiert die Touch-Aktionen von bis zu 4 Trends. Der Aufruf und das Zeichnen sollten in der Restzeit-Task erfolgen.

Es empfiehlt sich, die Instanz im Init des Tasks komplett auf 0 zu setzen.

### 3.4.3 XY-Plot

Mit dieser Funktion können sehr einfach bis zu vier XY-Plots dargestellt werden, z.B. um Funktionsgraphen zu visualisieren.



Funktionsgraphen könnten auch mit der Trend-Funktion dargestellt werden (durch Ersetzen der Standard-Zeit-Skala durch eine selbstgezeichnete Wert-X-Skala). Dies bedingt aber, dass es zu jeder X-Koordinate nur eine Y-Koordinate gibt (Beispiel Sinus).

Die Plot-Funktion kann auch für Graphen verwendet werden, bei denen es für eine X-Koordinate mehrere Y-Koordinaten gibt (Beispiel Sechseck).

Nachteile gegenüber der Trend-Funktion:

- Standardmäßig nur 1 Cursor, Referenz-Cursor müsste applikativ über Callbacks implementiert werden
- Keine Scroll-Begrenzung möglich, da Kurven unterschiedliche Array-Längen haben können
- Kein Link mehrerer Drawboxen möglich (siehe TrendLink)
- Quell-Werte müssen als REAL vorliegen

Vorteile gegenüber der Trend-Funktion:

- X-Skala ist von Haus aus eine Wert-Skala
- Kurven können unterschiedliche Array-Längen haben
- Cursor ist auf einen Funktionsgraphen optimiert
  - Er kann nicht nur auf einen X-Wert, sondern auf einen X/Y-Wert gestellt werden
  - Er kann kurvenbezogen gesetzt werden
  - Beim Touch-Klick kann automatisch auf den nächstgelegenen Kurvenpunkt gestellt werden
- Parametrierbare Darstellung der Null-Linie einer Skala

#### 3.4.3.1 Struktur

BrbVc4DrawPlot_TYP			
bEnable	BOOL	<input type="checkbox"/>	1=Aktiv
nRedrawCounterMatch	UINT	<input type="checkbox"/>	Redraw-Cycle, bei dem gezeichnet wird
Cfg	BrbVc4DrawPlotCfg_TYP	<input type="checkbox"/>	Konfiguration des Plots
State	BrbVc4DrawPlotState_TYP	<input type="checkbox"/>	Status des Plots
Intern	BrbVc4DrawPlotIntern_TYP	<input type="checkbox"/>	Interne Variablen für den Plot

Der Funktionsblock wird nur ausgeführt, wenn der Eingang „bEnable“ auf 1 ist. Dann wird in jedem Zyklus die Touch-Bedienung ausgewertet. Gezeichnet wird nur, wenn zusätzlich „General.nRedrawCounter“ (siehe „BrbVc4General“) den Wert von „nRedrawCounterMatch“ entspricht. Damit kann das Zeichnen von mehreren Plots auf einer Seite auf verschiedene CPU-Zyklen aufgeteilt werden, um so eine gleichmäßigere Verteilung der CPU-Last zu erreichen.

### 3.4.3.2 Konfiguration

BrbVc4DrawPlotCfg_TYP			Konfiguration des Plots
Drawbox	BrbVc4Drawbox_TYP	<input type="checkbox"/>	Angaben zur Drawbox
Padding	BrbVc4DrawPadding_TYP	<input type="checkbox"/>	Einrückung des Kurvenbereichs
nCurveAreaColor	UINT	<input type="checkbox"/>	Farbe des Kurvenbereichs
ScaleFont	BrbVc4Font_TYP	<input type="checkbox"/>	Eingang: Font der Skalierung
ScaleY	BrbVc4DrawPlotCfgScaleY_TYP[0..nBRBVC4_PLOT_SCALE_Y_INDEX_MAX]	<input type="checkbox"/>	Konfiguration der Y-Skalen
ScaleX	BrbVc4DrawPlotCfgScaleX_TYP	<input type="checkbox"/>	Konfiguration der X-Skala
TouchAction	BrbVc4DrawPlotCfgTouchAct_TYP	<input type="checkbox"/>	Konfiguration der Plot-Touch-Aktion
Curve	BrbVc4DrawPlotCfgCurve_TYP[0..nBRBVC4_PLOT_CURVE_INDEX_MAX]	<input type="checkbox"/>	Konfiguration der Kurven
Cursor	BrbVc4DrawPlotCfgCursor_TYP	<input type="checkbox"/>	Konfiguration des Plot-Cursors
Callbacks	BrbVc4DrawPlotCfgCallbacks_TYP	<input type="checkbox"/>	Konfiguration der Aufrufe
pTag	UDINT	<input type="checkbox"/>	Eingang: Zeiger auf Benutzer-Daten

Die Konfiguration ist der Übersichtlichkeit wegen in verschiedene Unter-Strukturen aufgeteilt.

#### 3.4.3.2.1 Allgemeines

BrbVc4Drawbox_TYP			
nLeft	UDINT	<input type="checkbox"/>	Eingang: Linke Koordinate
nTop	UDINT	<input type="checkbox"/>	Eingang: Obere Koordinate
nWidth	UDINT	<input type="checkbox"/>	Eingang: Breite
nHeight	UDINT	<input type="checkbox"/>	Eingang: Höhe
sFullName	STRING[nBRB_FILE_NAME_CHAR_MAX]	<input type="checkbox"/>	Pfad zur Anbindung
nBackgroundColor	UINT	<input type="checkbox"/>	Zur Anbindung an den Datenpunkt
nStatus	UINT	<input type="checkbox"/>	Zur Anbindung an den Datenpunkt

Die Angaben zur Drawbox sind korrekt auszufüllen, da nur dann alle Funktionalitäten richtig ausgeführt werden können. So werden z.B. die Koordinaten und Maße für die Touch-Funktionen benötigt. Der Name der Drawbox ist unbedingt auszufüllen, damit diese auch referenziert werden kann. Mit der Background-Color wird die Drawbox vor dem Zeichnen gelöscht.

BrbVc4DrawPadding_TYP			
nTop	DINT	<input type="checkbox"/>	Eingang: Einrückung Oben
nBottom	DINT	<input type="checkbox"/>	Eingang: Einrückung Unten
nLeft	DINT	<input type="checkbox"/>	Eingang: Einrückung Links
nRight	DINT	<input type="checkbox"/>	Eingang: Einrückung Rechts

Das Padding legt die Einrückung des Kurvenbereichs fest. Es muss so gewählt werden, dass die Skalen genug Platz haben.

#### 3.4.3.2.2 ScaleY

Diese Struktur gibt es für jede Skala (links und rechts)

BrbVc4DrawPlotCfgScaleY_TYP			Konfiguration der Y-Skala
bShow	BOOL	<input type="checkbox"/>	Eingang: 1=Skalierung der Y-Achse darstellen
nColor	UINT	<input type="checkbox"/>	Eingang: Farbe der Skala
nLinesCount	UINT	<input type="checkbox"/>	Eingang: Anzahl der Skalenstriche
nLineLength	DINT	<input type="checkbox"/>	Eingang: Länge der Skalenstriche
sUnit	STRING[nBRBVC4_DRAW_TEXT_CHAR_MAX]	<input type="checkbox"/>	Eingang: Angezeigter Einheiten-Text
ZeroLine	BrbVc4DrawPlotCfgZeroLine_TYP	<input type="checkbox"/>	Konfiguration der Null-Linie
Grid	BrbVc4DrawPlotCfgGrid_TYP	<input type="checkbox"/>	Konfiguration des Gitters
rMin	REAL	<input type="checkbox"/>	Eingang: Kleinster Wert der Y-Achse
rMax	REAL	<input type="checkbox"/>	Eingang: Größter Wert der Y-Achse
nFractionDigits	UINT	<input type="checkbox"/>	Eingang: Anzahl der anzuzeigenden Nachkommastellen
BrbVc4DrawPlotCfgZeroLine_TYP			Konfiguration einer Null-Linie
bShow	BOOL	<input type="checkbox"/>	Eingang: 1=Null-Linie anzeigen
nColor	UINT	<input type="checkbox"/>	Eingang: Farbe der Null-Linie
nDashWidth	UINT	<input type="checkbox"/>	Eingang: Breite für Strichelung (0=Solide)
BrbVc4DrawPlotCfgGrid_TYP			Konfiguration eines Plot-Gitters
bShow	BOOL	<input type="checkbox"/>	Eingang: 1=Gitter anzeigen
nColor	UINT	<input type="checkbox"/>	Eingang: Farbe des Gitters
nDashWidth	UINT	<input type="checkbox"/>	Eingang: Breite für Strichelung (0=Solide)

Der Einheitentext wird mittig oberhalb der Hauptlinie gezeichnet.

Der Zoom bzw. Scroll wird mit „rMin“ und „rMax“ festgelegt.

### 3.4.3.2.3 ScaleX

BrbVc4DrawPlotCfgScaleX_TYP			Konfiguration der X-Skala
bShow	BOOL	<input type="checkbox"/>	Eingang: 1=Skalierung der X-Achse darstellen
nColor	UINT	<input type="checkbox"/>	Eingang: Farbe der Skala
nLinesCount	UINT	<input type="checkbox"/>	Eingang: Anzahl der Skalenstriche
nLineLength	DINT	<input type="checkbox"/>	Eingang: Länge der Skalenstriche
sUnit	STRING[nBRBVC4_DRAW_TEXT_CHAR_MAX]	<input type="checkbox"/>	Eingang: Angezeigter Einheiten-Text
ZeroLine	BrbVc4DrawPlotCfgZeroLine_TYP	<input type="checkbox"/>	Konfiguration der Null-Linie
Grid	BrbVc4DrawPlotCfgGrid_TYP	<input type="checkbox"/>	Konfiguration des Gitters
rMin	REAL	<input type="checkbox"/>	Eingang: Kleinster Wert der X-Achse
rMax	REAL	<input type="checkbox"/>	Eingang: Größter Wert der X-Achse
nFractionDigits	UINT	<input type="checkbox"/>	Eingang: Anzahl der anzuzeigenden Nachkommastellen
BrbVc4DrawPlotCfgZeroLine_TYP			Konfiguration einer Null-Linie
bShow	BOOL	<input type="checkbox"/>	Eingang: 1=Null-Linie anzeigen
nColor	UINT	<input type="checkbox"/>	Eingang: Farbe der Null-Linie
nDashWidth	UINT	<input type="checkbox"/>	Eingang: Breite für Strichelung (0=Solide)
BrbVc4DrawPlotCfgGrid_TYP			Konfiguration eines Plot-Gitters
bShow	BOOL	<input type="checkbox"/>	Eingang: 1=Gitter anzeigen
nColor	UINT	<input type="checkbox"/>	Eingang: Farbe des Gitters
nDashWidth	UINT	<input type="checkbox"/>	Eingang: Breite für Strichelung (0=Solide)

Der Einheitentext wird rechts der Hauptlinie gezeichnet.  
Der Zoom bzw. Scroll wird mit „rMin“ und „rMax“ festgelegt.

### 3.4.3.2.4 TouchAction – Funktion des Touchs

BrbVc4DrawPlotCfgTouchAct_TYP			Konfiguration der Plot-Touch-Aktion
BorderCorrection	BrbVc4DrawTouchBorderCorr_TYP	<input type="checkbox"/>	Touch-Korrektur des Rahmens
bSetCursor	BOOL	<input type="checkbox"/>	Eingang: 1=Cursor setzen
bZoomX	BOOL	<input type="checkbox"/>	Eingang: 1=ZoomX aktivieren
bZoomY	BOOL	<input type="checkbox"/>	Eingang: 1=ZoomY aktivieren
bScrollX	BOOL	<input type="checkbox"/>	Eingang: 1=ScrollingX aktivieren
bScrollY	BOOL	<input type="checkbox"/>	Eingang: 1=ScrollingY aktivieren
BrbVc4DrawTouchBorderCorr_TYP			Touch-Korrektur des Rahmens
nX	SINT	<input type="checkbox"/>	Eingang: Offset X für die Touch-Korrektur des Rahmens
nY	SINT	<input type="checkbox"/>	Eingang: Offset Y für die Touch-Korrektur des Rahmens

Hier wird festgelegt, ob und welche Funktion durch den Touch bedienbar ist.

Mit der „BorderCorrection“ kann ein Offset festgelegt werden, welcher die Koordinaten des Touchs aufgrund des Rahmens der Drawbox korrigiert. Der Rahmen „Flat\_back“ z.B. muss jeweils mit dem Wert 2 korrigiert werden, damit der Cursor auch exakt an dem berührten Punkt gesetzt wird.

Das Setzen der Cursor wird schon beim einmaligen Klicken ausgeführt. Dabei wird der Cursor je nach Konfiguration (siehe unten) auf den der Klick-Position am nächst gelegenen Kurvenpunkt gesetzt.

Das Scrollen wird durch Verschiebe-Bewegung ausgelöst.

Beim Zoomen erscheint ein Fenster, welches auf den entsprechenden Ausschnitt gezogen werden kann.

Wenn Scroll und Zoom gemeinsam aktiviert sind, wird zur Unterscheidung ein Timer eingesetzt.

Wenn innerhalb 1 Sekunde der Touch verschoben wird, wird die Scroll-Funktion ausgeführt. Wenn der Touch 1 Sekunde auf Position gehalten wird, wird das Zoom-Fenster eingeblendet.

### 3.4.3.2.5 Curve – Kurven

Diese Struktur gibt pro Kurve (max. 4).

BrbVc4DrawPlotCfgCurve_TYP			Konfiguration einer Plot-Kurve
bShow	BOOL	<input type="checkbox"/>	Eingang: 1=Anzeigen
nColor	UINT	<input type="checkbox"/>	Eingang: Farbe
eScaleY	BrbVc4PlotScaleYIndex_ENUM	<input type="checkbox"/>	Eingang: Skalen-Zugehörigkeit (0=Links, 1=Rechts)
nSourceArrayIndexMax	UDINT	<input type="checkbox"/>	Eingang: Max. Index des Werte-Arrays
eValueSource	BrbVc4PlotSource_ENUM	<input type="checkbox"/>	Eingang: Quelle
pArrayX	UDINT	<input type="checkbox"/>	Eingang: Zeiger auf den ersten Eintrag im X-Werte-Array
pArrayY	UDINT	<input type="checkbox"/>	Eingang: Zeiger auf den ersten Eintrag im Y-Werte-Array
pArrayStruct	UDINT	<input type="checkbox"/>	Eingang: Zeiger auf den ersten Eintrag im Struktur-Array
nStructSize	UDINT	<input type="checkbox"/>	Eingang: Größe der Struktur bei Struktur-Array-Quelle
nStructMemberOffsetX	UDINT	<input type="checkbox"/>	Eingang: Offset des X-Wertes in der Struktur bei Struktur-Array-Quelle
nStructMemberOffsetY	UDINT	<input type="checkbox"/>	Eingang: Offset des Y-Wertes in der Struktur bei Struktur-Array-Quelle
rConversionFactorX	REAL	<input type="checkbox"/>	Eingang: Faktor zur Umrechnung der Quell-Daten auf die eingestellte X-Skala
rConversionFactorY	REAL	<input type="checkbox"/>	Eingang: Faktor zur Umrechnung der Quell-Daten auf die eingestellte Y-Skala
bCursorUseSourceValues	BOOL	<input type="checkbox"/>	Eingang: 1=Cursor-Anzeige und Statistik verwenden Quell-Daten
bCalculateStatistics	BOOL	<input type="checkbox"/>	Eingang: 1=Statistik-Werte berechnen

Der Eingang „nSourceArrayIndexMax“ gibt den maximalen Index an, bis zu dem das Quell-Array dieser Kurve ausgewertet wird, also bis zu dem die aufgenommenen Punkte dargestellt werden. Er kann für jede Kurve verschieden sein. Beginn ist immer bei 0. Wenn keine Werte vorhanden sind, muss er auf 0 gesetzt werden.

Der Eingang „eValueSource“ legt die Strukturierung der Quell-Kurvenwerte fest:

BrbVc4PlotSource_ENUM		Quellen-Angabe 0..1
eBRBVC4_PLOT_SOURCE_SINGLE_ARR		Einfaches Array
eBRBVC4_PLOT_SOURCE_STRUCT_ARR		Struktur-Array

Die Werte müssen als REAL vorhanden sein, entweder als einzelne Arrays oder als Struktur-Array.

Bei „eBRBVC4\_PLOT\_SOURCE\_SINGLE\_ARR“ müssen die Eingänge „pArrayX“ und „pArrayY“ auf den Anfang des jeweiligen Arrays gesetzt werden.

Bei „eBRBVC4\_PLOT\_SOURCE\_STRUCT\_ARR“ muss der Eingang „pArrayStruct“ auf den Anfang des Struktur-Arrays gesetzt werden. Zusätzlich müssen noch drei Parameter übergeben werden:

Bei „nStructSize“ muss die Größe eines Eintrags übergeben werden, welcher einfach durch „sizeof(“ ermittelt werden kann.

Bei „nStructMemberOffsetX“ und „nStructMemberOffsetY“ müssen die Byte-Offsets auf die Struktur-Items innerhalb der Struktur übergeben werden. Diese können bequem mit der Funktion „BrbGetStructMemberOffset“ ermittelt werden, welche in der Bibliothek „BrbLib“ beschrieben ist.

Mit dem Eingang „rConversionFactorX“ bzw. „rConversionFactorY“ können die Quell-Werte an die parametrisierten Skalen angeglichen werden. Im Normalfall sollten sie „1.0“ sein.

Der Eingang „bCursorUseSourceValues“ wirkt sich nur aus, wenn einer der Skalierungsfaktoren nicht 1.0 ist, denn dann sind Quell- und konvertierte Werte nicht gleich.

Ist er 0 (=Normal), so werden sowohl bei der Anzeige der Cursor-Koordinaten als auch bei den Statistik-Werten die konvertierten Werte verwendet.

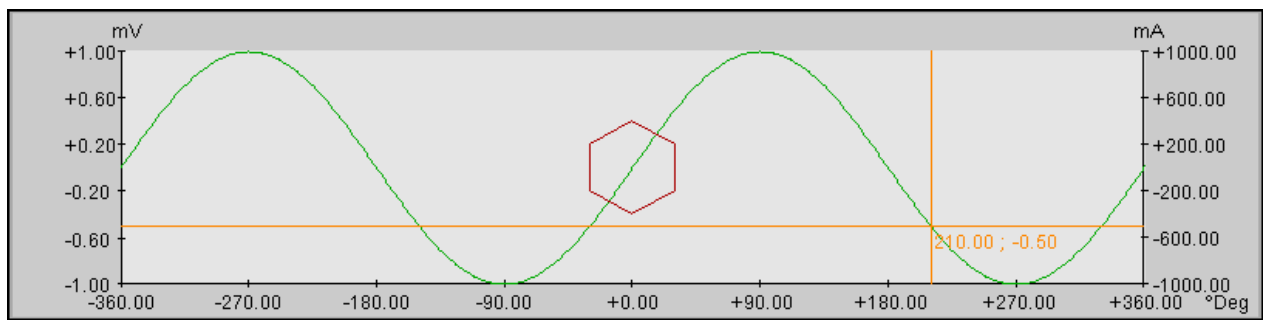
Ist er 1, so werden sowohl bei der Anzeige der Cursor-Koordinaten als auch bei den Statistik-Werten die Quell-Werte (also die nicht konvertierten Werte) verwendet. Dadurch kann eine Kurve durch Skalierung an die anderen Kurven angepasst werden, ohne dass die Werte dieser Kurve zu einer der Y-Skalen passen. Die Anzeige der Cursor-Koordinaten passt dann zwar zu keiner Y-Skala, aber sie entsprechen den tatsächlichen Quell-Werten.

Mit dem Eingang „bCalculateStatistics“ werden beim Zeichnen automatisch die Min/Max-Werte für X und Y ermittelt (siehe ‚Status‘ unten).

### 3.4.3.2.6 Cursor

Es gibt einen Cursor, welcher als Kreuz dargestellt wird.





BrbVc4DrawPlotCfgCursor_TYP		Konfiguration des Plot-Cursors
bShow	USINT	<input type="checkbox"/> Eingang: 1=Cursor darstellen
nColor	UINT	<input type="checkbox"/> Eingang: Farbe des Cursors
eTouchCurve	BrbVc4PlotCursorCurve_ENUM	<input type="checkbox"/> Kurve, auf die der Cursor bei Touch-Klick gesetzt wird
eActCurve	BrbVc4PlotCursorCurve_ENUM	<input type="checkbox"/> Kurve, auf die der Cursor gesetzt ist
rCursorX	REAL	<input type="checkbox"/> Eingang: Aktueller X-Cursor-Wert
rCursorY	REAL	<input type="checkbox"/> Eingang: Aktueller Y-CursorWert

Da es sich um einen Kreuz-Cursor handelt, muss festgelegt werden, ob und welche Kurve dazu verwendet wird.

Dazu wird der Eingang „eTouchCurve“ herangezogen.

BrbVc4PlotCursorCurve_ENUM		Kurve, auf die der Cursor bei Touch-Klick gesetzt wird
eBRBVC4_PLOT_CURSOR_CURVE_NONE		Cursor wird frei gesetzt
eBRBVC4_PLOT_CURSOR_CURVE_0		Cursor wird auf Kurve 0 gesetzt
eBRBVC4_PLOT_CURSOR_CURVE_1		Cursor wird auf Kurve 1 gesetzt
eBRBVC4_PLOT_CURSOR_CURVE_2		Cursor wird auf Kurve 2 gesetzt
eBRBVC4_PLOT_CURSOR_CURVE_3		Cursor wird auf Kurve 3 gesetzt
eBRBVC4_PLOT_CURSOR_CURVE_ALL		Cursor wird auf die am nächsten gelegene Kurve gesetzt

Bei „All“ wird der Cursor auf die Kurve gelegt, welche am nächsten an der Klick-Position liegt. Am Element „eActCurve“ wird dann der dazugehörige Kurven-Index ausgegeben.

Bei einer spezifischen Kurve („0“ bis „3“) wird der Cursor auf diese Kurve gelegt. Am Element „eActCurve“ wird dann der dazugehörige Kurven-Index ausgegeben.

Bei „None“ wird der Cursor unabhängig von einer Kurve auf die geklickte Position gesetzt. Am Element „eActCurve“ wird dann der Index der nächstgelegenen Kurve ausgegeben.

Die Elemente „eCursorX“ und „eCursorY“ enthalten nach einem Klick die Werte der angegebenen Kurve. Der Y-Wert richtet sich dabei nach der Skala, die bei der in „eActCurve“ angegebenen Kurve parametrisiert ist. Über diese Elemente kann der Cursor auch per Programm gesetzt werden. Dazu wird ebenfalls die Y-Skala der in „eActCurve“ angegebenen Kurve verwendet.

### 3.4.3.2.7 Callbacks

Hinweis: Diese Funktionalität ist aufgrund von Funktionszeigern nur in ANSI-C nutzbar, aber nicht in IEC-Sprachen (siehe Punkt [Hinweise zu StructuredText und anderen IEC-Sprachen](#))

BrbVc4DrawPlotCfgCallbacks_TYP		Konfiguration der Plot-Aufrufe
pCallbackAfterClear	UDINT	<input type="checkbox"/> Eingang: Funktions-Zeiger für Aufruf nach Löschen der Drawbox
pCallbackAfterCurveArea	UDINT	<input type="checkbox"/> Eingang: Funktions-Zeiger für Aufruf nach Zeichnen des Kurvenbereichs
pCallbackAfterScaleLineY	UDINT	<input type="checkbox"/> Eingang: Funktions-Zeiger für Aufruf nach Zeichnen eines Y-Skalierungs-Strichs
pCallbackAfterScaleY	UDINT	<input type="checkbox"/> Eingang: Funktions-Zeiger für Aufruf nach Zeichnen einer Y-Skala
pCallbackAfterScaleLineX	UDINT	<input type="checkbox"/> Eingang: Funktions-Zeiger für Aufruf nach Zeichnen eines X-Skalierungs-Strichs
pCallbackAfterScaleX	UDINT	<input type="checkbox"/> Eingang: Funktions-Zeiger für Aufruf nach Zeichnen der X-Skala
pCallbackAfterCurve	UDINT	<input type="checkbox"/> Eingang: Funktions-Zeiger für Aufruf nach Zeichnen der Kurve
pCallbackAfterCursor	UDINT	<input type="checkbox"/> Eingang: Funktions-Zeiger für Aufruf nach Zeichnen des Cursors
pCallbackAfterZoomWindow	UDINT	<input type="checkbox"/> Eingang: Funktions-Zeiger für Aufruf nach Zeichnen des Zoom-Fensters

Ein Callback ist ein Aufruf einer vom Anwender geschriebenen Funktion während des Zeichnens.

Er arbeitet mit sogenannten Funktions-Zeigern. Dabei wird die Adresse einer Funktion übergeben, welche der Anwender selbst schreibt. Lediglich die Signatur, also die Anzahl, Reihenfolge und die Datentypen-

pen der Argumente sind dabei vorgeschrieben. Der Inhalt der Funktion bleibt vollkommen dem Anwender überlassen.

Es gibt 9 verschiedene Callbacks (siehe oben), welche nach dem Zeichnen des jeweiligen Elements aufgerufen werden können.

Für jeden Callback gibt es ein Muster der Signatur in der Datei „BrbVc4PlotCallbackTemplates.c“ in der Bibliothek:

```
unsigned short BrbVc4PlotCallbackAfterClear(struct BrbVc4DrawPlot_TYP* pPlot)

unsigned short BrbVc4PlotCallbackAfterCrveArea(struct BrbVc4DrawPlot_TYP* pPlot)

unsigned short BrbVc4PlotCallbackAfterScaleLineY(struct BrbVc4DrawPlot_TYP* pPlot,
BrbVc4PlotScaleYIndex_ENUM eScaleYIndex, UINT nLineIndex, BrbVc4Line_TYP* pScaleLine,
BrbVc4DrawText_TYP* pScaleText, REAL rScaleValue)

unsigned short BrbVc4PlotCallbackAfterScaleY(struct BrbVc4DrawPlot_TYP* pPlot,
BrbVc4PlotScaleYIndex_ENUM eScaleYIndex)

unsigned short BrbVc4PlotCallbackAfterScaleLineX(struct BrbVc4DrawPlot_TYP* pPlot, UINT nLineIndex,
BrbVc4Line_TYP* pScaleLine, BrbVc4DrawText_TYP* pScaleText, REAL rScaleValue)

unsigned short BrbVc4PlotCallbackAfterScaleX(struct BrbVc4DrawPlot_TYP* pPlot)

unsigned short BrbVc4PlotCallbackAfterCurve(struct BrbVc4DrawPlot_TYP* pPlot, UINT nCurveIndex)

unsigned short BrbVc4PlotCallbackAfterCursor(struct BrbVc4DrawPlot_TYP* pPlot)

unsigned short BrbVc4PlotCallbackAfterZoomWind(struct BrbVc4DrawPlot_TYP* pPlot)
```

Manche Callbacks werden während des Zeichnens mehrmals aufgerufen, so z.B. nach dem Zeichnen jeder Kurve. Über Argumente werden aktuelle Werte übergeben, z.B. die aktuelle Kurve.

Im Callback kann auf die gesamte Plot-Struktur zugegriffen werden, auch auf die intern berechneten Daten. ACHTUNG: Es sollten aber keine Werte verändert werden!

Soll ein Callback aktiviert werden, so ist dessen Adresse in die obige Struktur einzutragen.

Beispiel:

Es wird eine Funktion angelegt, welche dem Muster entspricht:

```
unsigned short PlotCallbackAfterCurveArea(struct BrbVc4DrawPlot_TYP* pPlot)
{
    // Anwender-Code
    return 0;
}
```

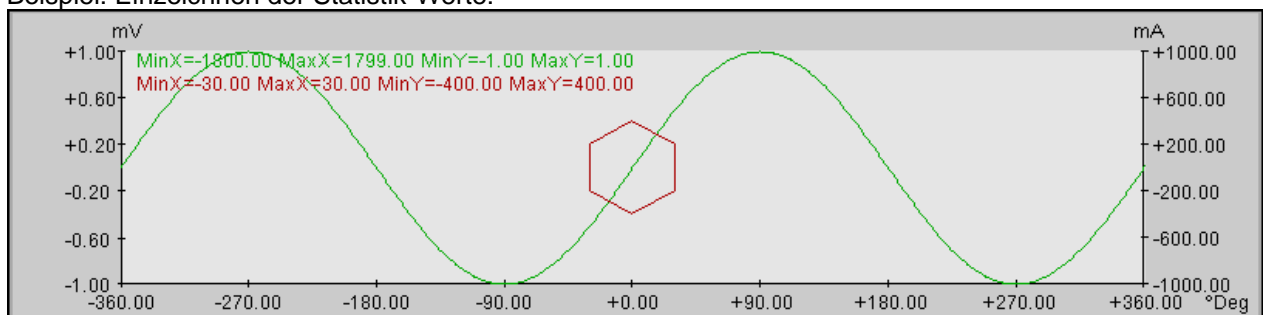
Vor dem Aufruf der Trend-Funktion wird die Adresse des Callbacks übergeben

```
Plot.Cfg.Callbacks.pCallbackAfterCurveArea = (UDINT)&PlotCallbackAfterCurveArea;
```

Innerhalb des Callbacks kann dann mit Zeichenfunktionen die visuelle Ausgabe erweitert werden, z.B. Texte oder zusätzliche Linien eingezeichnet werden. Dadurch ergibt sich eine Vielfalt an Möglichkeiten. Der Rückgabewert der Funktion ist egal.

ACHTUNG: Das Koordinaten-System bezieht sich auf die Plot-Drawbox, weil diese noch referenziert ist.

Beispiel: Einzeichnen der Statistik-Werte:





### 3.4.3.2.8 pTag

Dieser Zeiger wird von der Funktion nicht benutzt. Er kann vom Anwender als Zeiger auf Benutzer-Daten gesetzt und dann in den Callbacks verwendet werden.

### 3.4.3.3 Status

Diese Struktur enthält Ausgangs-Daten, welche für den Anwender interessant sein könnten:

BrbVc4DrawPlotState_TYP		Status des Plots
eTouchAction	BrbVc4PlotTouchAction_ENUM	Status einer Touch-Aktion
Cursor	BrbVc4DrawPlotStateCursor_TYP	Cursor-Daten des Plots
Statistic	BrbVc4DrawPlotStateStatistic_TYP	Statistische Werte

Der Ausgang „eTouchAction“ gibt den momentanen Stand der Touch-Bedienung an:

BrbVc4PlotTouchAction_ENUM	Touch-Aktionen des Plots
eBRBVC4_PLOT_TOUCHACT_NONE	Keine Touch-Aktion
eBRBVC4_PLOT_TOUCHACT_CURS_SET	Cursor wurde gesetzt
eBRBVC4_PLOT_TOUCHACT_SCROLL	Momentan wird gescrollt
eBRBVC4_PLOT_TOUCHACT_ZOOM_DRAG	Momentan wird das Zoom-Fenster gezogen
eBRBVC4_PLOT_TOUCHACT_ZOOM_SET	Das Zoom-Fenster wurde gesetzt

Abhängig vom Status bleibt er nur einen oder auch mehrere Zyklen anstehen.

Unter „Cursor“ stehen die Ausgänge des Cursors:

BrbVc4DrawPlotStateCursor_TYP	Cursor-Daten des Plots
nSampleIndex	Ausgang: Sample-Index des geklickten Cursors

Der Ausgang „nSampleIndex“ ist der Index im Quell-Array. Er bezieht sich immer auf die dem Cursor zugeordnete Kurve (siehe oben). Ist dem Cursor keine Kurve zugeordnet, ist der Ausgang „0“. Liegen auf dem Cursor-Punkt mehrere Punkte derselben Kurve, wird immer der Index des ersten passenden Graph-Punktes ausgegeben.

Die Statistik-Werte gibt es pro Kurve.

BrbVc4DrawPlotStateStatistic_TYP			Statistik einer Kurve
Curve	BrbVc4DrawPlotStateStatCurve_TYP[0..nBRBVC4_PLOT_CURVE_INDEX_MAX]		Statistische Werte pro Kurve
BrbVc4DrawPlotStateStatCurve_TYP			Statistik einer Kurve
rMinX	REAL	<input type="checkbox"/>	Ausgang: Kleinster Wert der X-Achse
rMaxX	REAL	<input type="checkbox"/>	Ausgang: Größter Wert der X-Achse
rMinY	REAL	<input type="checkbox"/>	Ausgang: Kleinster Wert der Y-Achse
rMaxY	REAL	<input type="checkbox"/>	Ausgang: Größter Wert der Y-Achse

Sie werden nur berechnet, wenn in der Konfiguration der Kurve das Berechnen der Statistik-Werte eingeschaltet ist.

### 3.4.3.4 Intern

Diese Strukturen werden während des Aufrufs mit intern berechneten Daten gefüllt, z.B. Koordinaten und Maße einzelner Zeichen-Elemente. So ist z.B. der berechnete Kurvenbereich oder die Position des Zoom-Fensters enthalten.

Auf sie darf während eines Callbacks lesend zugegriffen werden, um eigene Elemente positionieren zu können. Auf keinen Fall sollten sie verändert werden!

### 3.4.3.5 BrbVc4DrawPlot

```
unsigned short BrbVc4DrawPlot(struct BrbVc4DrawPlot_TYP* pPlot, struct BrbVc4General_TYP* pGeneral)
```

Argumente:

```
struct BrbVc4DrawPlot_TYP* pPlot  
    Zeiger auf die Instanz  
struct BrbVc4General_TYP* pGeneral  
    Zeiger auf die Instanz von „BrbVc4General_TYP“
```

Rückgabe:

`UINT`

Immer 0

**Beschreibung:**

Zeichnet einen XY-Plot nach Vorgaben. Der Aufruf sollte in der Restzeit-Task erfolgen.  
Es empfiehlt sich, die Instanz im Init des Tasks komplett auf 0 zu setzen.

**3.4.3.6 BrbVc4GetPlotDisplayCoordinateY**

```
signed long BrbVc4GetPlotDisplayCoordinateY(struct BrbVc4DrawPlot_TYP* pPlot,  
BrbVc4PlotScaleYIndex_ENUM eScaleY, float rValue)
```

**Argumente:**

```
struct BrbVc4DrawPlot_TYP* pPlot  
    Zeiger auf die Instanz  
BrbVc4PlotScaleYIndex_ENUM eScaleY  
    Angabe der zugehörigen Skala  
REAL rValue  
    Rohwert
```

**Rückgabe:**`DINT`

Y-Pixel-Koordinate

**Beschreibung:**

Gibt die Y-Pixel-Koordinate eines Plot-Wertes zurück.

**3.4.3.7 BrbVc4GetPlotDisplayCoordinateX**

```
signed long BrbVc4GetPlotDisplayCoordinateX(struct BrbVc4DrawPlot_TYP* pPlot, float rValue)
```

**Argumente:**

```
struct BrbVc4DrawPlot_TYP* pPlot  
    Zeiger auf die Instanz  
REAL rValue  
    Rohwert
```

**Rückgabe:**`DINT`

X-Pixel-Koordinate

**Beschreibung:**

Gibt die X-Pixel-Koordinate eines Plot-Wertes zurück.

### 3.4.4 Achse linear darstellen

Mit dieser Funktion kann sehr einfach eine Achse linear dargestellt werden.

#### 3.4.4.1 Struktur

BrbVc4DrawAxisLinear_TYP			
bVertical	BOOL	<input type="checkbox"/>	Eingang: 0=Horizontal, 1=Vertikal
bShowDrawArea	BOOL	<input type="checkbox"/>	Eingang: 1=Zeichenbereich darstellen
nDrawAreaLeft	UDINT	<input type="checkbox"/>	Eingang: Linke Koordinate des Zeichenbereichs in [Pixel]
nDrawAreaTop	UDINT	<input type="checkbox"/>	Eingang: Obere Koordinate des Zeichenbereichs in [Pixel]
nDrawAreaWidth	UDINT	<input type="checkbox"/>	Eingang: Breite des Zeichenbereichs in [Pixel]
nDrawAreaHeight	UDINT	<input type="checkbox"/>	Eingang: Höhe des Zeichenbereichs in [Pixel]
nDrawAreaColor	UINT	<input type="checkbox"/>	Eingang: Farbe des Zeichenbereichs
nDrawIndent	UDINT	<input type="checkbox"/>	Eingang: Einzug links + rechts bzw. oben + unten in [Pixel]
nAxisLimitMin	DINT	<input type="checkbox"/>	Eingang: Kleinste Achsposition in [Achseinheiten]
nAxisLimitMax	DINT	<input type="checkbox"/>	Eingang: Größte Achsposition in [Achseinheiten]
bShowAxisScale	BOOL	<input type="checkbox"/>	Eingang: 1=Skalierung darstellen
nAxisScaleCount	UINT	<input type="checkbox"/>	Eingang: Anzahl der Skalierungs-Striche
nAxisScaleColor	UINT	<input type="checkbox"/>	Eingang: Farbe der Skalierung
nAxisScaleFont	BrbVc4Font_TYP	<input type="checkbox"/>	Eingang: Font der Skalierung
bHighlightActPosition	BOOL	<input type="checkbox"/>	Eingang: 1=Skalierung hervorheben, wenn Achse auf dieser Position
nAxisScaleHighlightColor	UINT	<input type="checkbox"/>	Eingang: Farbe der Hervorhebung
blnInverted	BOOL	<input type="checkbox"/>	Eingang: 1=Achs-Richtung umdrehen
bClip	BOOL	<input type="checkbox"/>	Eingang: 1=Ausschnitt anzeigen
nAxisClipRange	UDINT	<input type="checkbox"/>	Eingang: Größe des Ausschnitts in [Achseinheiten]
bShowAxisPosLine	BOOL	<input type="checkbox"/>	Eingang: Achs-Positions-Strich darstellen
bShowAxisBorder	BOOL	<input type="checkbox"/>	Eingang: Achs-Beschriftungs-Umrandung darstellen
nAxisColor	UINT	<input type="checkbox"/>	Eingang: Farbe des Positions-Strichs und der Umrandung
eAxisCaptionOrder	BrbVc4DrawAxisCaptionOrder_ENUM	<input type="checkbox"/>	Eingang: Reihenfolge der Achs-Beschriftung
bShowAxisName	BOOL	<input type="checkbox"/>	Eingang: 1=Achsname darstellen
sAxisName	STRING(nBRBVC4_DRAW_TEXT_CHAR_MAX)	<input type="checkbox"/>	Eingang: Achsname
nAxisNameColor	UINT	<input type="checkbox"/>	Eingang: Farbe des Achsnamens
AxisNameFont	BrbVc4Font_TYP	<input type="checkbox"/>	Eingang: Font des Achsnamens
bShowAxisActPosition	BOOL	<input type="checkbox"/>	Eingang: 1=Aktuelle Achs-Position darstellen
rAxisActPosition	REAL	<input type="checkbox"/>	Eingang: Aktuelle Achs-Position
nAxisActPositionColor	UINT	<input type="checkbox"/>	Eingang: Farbe der Achs-Position
bShowAxisActVelocity	BOOL	<input type="checkbox"/>	Eingang: 1=Aktuelle Achs-Geschwindigkeit darstellen
rAxisActVelocity	REAL	<input type="checkbox"/>	Eingang: Aktuelle Achs-Geschwindigkeit
nAxisActVelocityColor	UINT	<input type="checkbox"/>	Eingang: Farbe der Achs-Geschwindigkeit
AxisValueFont	BrbVc4Font_TYP	<input type="checkbox"/>	Eingang: Font der Achs-Position und Geschwindigkeit
bShowAxisSetPosition	BOOL	<input type="checkbox"/>	Eingang: 1=Soll-Position darstellen
rAxisSetPosition	REAL	<input type="checkbox"/>	Eingang: Soll-Achs-Position
nAxisSetPositionColor	UINT	<input type="checkbox"/>	Eingang: Farbe der Soll-Achs-Position

#### 3.4.4.2 BrbVc4DrawAxisLinear

```
float BrbVc4DrawAxisLinear(struct BrbVc4DrawAxisLinear_TYP* pAxis, struct BrbVc4General_TYP* pGeneral)
```

##### Argumente:

```
struct BrbVc4DrawAxisLinear_TYP* pAxis  
    Zeiger auf die Instanz  
struct BrbVc4General_TYP* pGeneral  
    Zeiger auf die Instanz von „BrbVc4General_TYP“
```

##### Rückgabe:

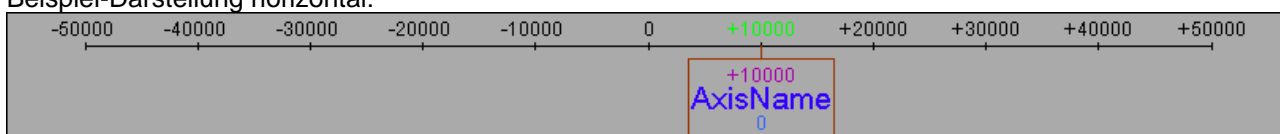
REAL

Der momentane Beginn der Skalierung entsprechend der Parametrierung

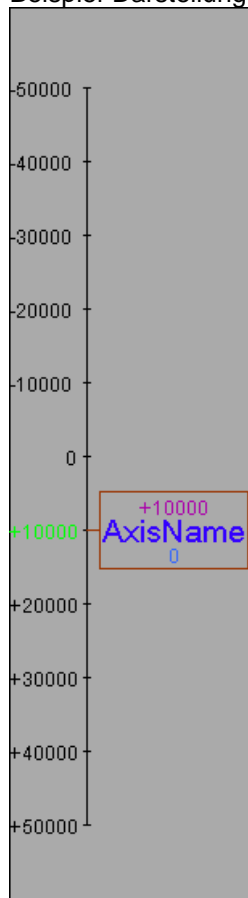
##### Beschreibung:

Mit den Eingängen kann die Darstellung der einzelnen Elemente sehr beeinflusst werden. So können z.B. die einzelnen Elemente ein- oder ausgeblendet und deren Farben gesetzt werden.

Beispiel-Darstellung horizontal:



Beispiel-Darstellung vertikal:



Zum besseren Verständnis werden hier einige Eingänge beschrieben:

- „nDrawAreaXXX“

Hier wird die obere, linke Ecke sowie Breite und Höhe des Zeichen-Bereichs parametrisiert. Wird in eine Drawbox gezeichnet, entspricht 0,0 der linken oberen Ecke.

- „nDrawIndent“

Der Einzug rückt die Skalierung bei horizontaler Darstellung links und rechts ein, bei vertikaler Darstellung oben und unten. Das ist notwendig, weil die Achsbeschriftung sonst bei Endlage aus dem Zeichenbereich rutscht.

- „nAxisLimitMin“ + „nAxisLimitMax“

Hier wird die negative bzw. die positive Endlage der Achse angegeben. Ist der Achsbereich sehr groß, können kleine Positions-Änderungen nur schlecht erkannt werden. In diesem Fall sollte man evtl. die Clip-Funktionalität verwenden (siehe Eingang „bClip“).

Es sollten außerdem keine „schiefen“ Zahlen verwendet werden, da sonst die Rundungsfehler zu groß werden. Also besser „2000000000“ als „2147483648“.

- „bHighlightActPosition“

Hat die Achse die Position eines Skalierungs-Strichs, wird der Skalierungswert in einer anderen Farbe dargestellt.

- „bInverted“

Normalerweise wird die Achse mit positiver Richtung nach rechts bzw. nach unten dargestellt. Mit diesem Eingang kann die Richtung umgedreht werden.

- „bClip“ + „nAxisClipRange“

Mit diesen Eingängen kann erreicht werden, dass nicht der komplette Achs-Bereich dargestellt wird, sondern nur ein Ausschnitt. Der Ausschnitt wird aufgrund der aktuellen Position und der angegebenen Ausschnitts-Größe berechnet.

- „eAxisCaptionOrder“

Hier kann die Reihenfolge der Achs-Beschriftungen von oben nach unten festgelegt werden:

BrbVc4DrawAxisCaptionOrder_ENUM	
eBRBVC4_DRAW_AXIS_NAME_POS_VEL	0=Name-Position-Geschwindigkeit
eBRBVC4_DRAW_AXIS_NAME_VEL_POS	1=Name-Geschwindigkeit-Position
eBRBVC4_DRAW_AXIS_POS_NAME_VEL	2=Position-Name-Geschwindigkeit
eBRBVC4_DRAW_AXIS_POS_VEL_NAME	3=Position-Geschwindigkeit-Name
eBRBVC4_DRAW_AXIS_VEL_NAME_POS	4=Geschwindigkeit-Name-Position
eBRBVC4_DRAW_AXIS_VEL_POS_NAME	5=Geschwindigkeit-Position-Name

### 3.4.5 Achse radial darstellen

Mit dieser Funktion kann sehr einfach eine Achse radial dargestellt werden. Dies eignet sich besonders gut für periodische Achsen.

#### 3.4.5.1 Struktur

BrbVc4DrawAxisRadial_TYP		
bShowDrawArea	BOOL	<input type="checkbox"/> Eingang: 1=Zeichenbereich darstellen
nDrawAreaLeft	UDINT	<input type="checkbox"/> Eingang: Linke Koordinate des Zeichenbereichs in [Pixel]
nDrawAreaTop	UDINT	<input type="checkbox"/> Eingang: Obere Koordinate des Zeichenbereichs in [Pixel]
nDrawAreaWidth	UDINT	<input type="checkbox"/> Eingang: Breite des Zeichenbereichs in [Pixel]
nDrawAreaHeight	UDINT	<input type="checkbox"/> Eingang: Höhe des Zeichenbereichs in [Pixel]
nDrawAreaColor	UINT	<input type="checkbox"/> Eingang: Farbe des Zeichenbereichs
nRadius	UDINT	<input type="checkbox"/> Eingang: Radius der Skalierung in [Pixel]
nAxisLimitMin	DINT	<input type="checkbox"/> Eingang: Kleinste Achsposition in [Achseinheiten]
nAxisLimitMax	DINT	<input type="checkbox"/> Eingang: Größte Achsposition in [Achseinheiten]
bShowAxisScale	BOOL	<input type="checkbox"/> Eingang: 1=Skalierung darstellen
nAxisScaleCount	UINT	<input type="checkbox"/> Eingang: Anzahl der Skalierungs-Striche
nAxisScaleColor	UINT	<input type="checkbox"/> Eingang: Farbe der Skalierung
AxisScaleFont	BrbVc4Font_TYP	<input type="checkbox"/> Eingang: Font der Skalierung
bHighlightActPosition	BOOL	<input type="checkbox"/> Eingang: 1=Skalierung hervorheben, wenn Achse auf dieser Position
nAxisScaleHighlightColor	UINT	<input type="checkbox"/> Eingang: Farbe der Hervorhebung
blnverted	BOOL	<input type="checkbox"/> Eingang: 1=Achs-Richtung umdrehen
nOffset	UINT	<input type="checkbox"/> Eingang: 1=Versatz der Achs-Position von 0 bis 360 in[°]
bClip	BOOL	<input type="checkbox"/> Eingang: 1=Ausschnitt anzeigen
nAxisClipRange	UDINT	<input type="checkbox"/> Eingang: Größe des Ausschnitts in [Achseinheiten]
bShowAxisPosLine	BOOL	<input type="checkbox"/> Eingang: Achs-Positions-Strich darstellen
nAxisColor	UINT	<input type="checkbox"/> Eingang: Farbe des Positions-Strichs und der Umrandung
eAxisCaptionOrder	BrbVc4DrawAxisCaptionOrder_ENUM	<input type="checkbox"/> Eingang: Reihenfolge der Achs-Beschriftung
bShowAxisName	BOOL	<input type="checkbox"/> Eingang: 1=Achsname darstellen
sAxisName	STRING[nBRBVC4_DRAW_TEXT_CHAR_MAX]	<input type="checkbox"/> Eingang: Achsname
nAxisNameColor	UINT	<input type="checkbox"/> Eingang: Farbe des Achsnamens
AxisNameFont	BrbVc4Font_TYP	<input type="checkbox"/> Eingang: Font des Achsnamens
bShowAxisActPosition	BOOL	<input type="checkbox"/> Eingang: 1=Aktuelle Achs-Position darstellen
rAxisActPosition	REAL	<input type="checkbox"/> Eingang: Aktuelle Achs-Position
nAxisActPositionColor	UINT	<input type="checkbox"/> Eingang: Farbe der Achs-Position
bShowAxisActVelocity	BOOL	<input type="checkbox"/> Eingang: 1=Aktuelle Achs-Geschwindigkeit darstellen
rAxisActVelocity	REAL	<input type="checkbox"/> Eingang: Aktuelle Achs-Geschwindigkeit
nAxisActVelocityColor	UINT	<input type="checkbox"/> Eingang: Farbe der Achs-Geschwindigkeit
AxisValueFont	BrbVc4Font_TYP	<input type="checkbox"/> Eingang: Font der Achs-Position und Geschwindigkeit
bShowAxisSetPosition	BOOL	<input type="checkbox"/> Eingang: 1=Soll-Position darstellen
rAxisSetPosition	REAL	<input type="checkbox"/> Eingang: Soll-Achs-Position
nAxisSetPositionColor	UINT	<input type="checkbox"/> Eingang: Farbe der Soll-Achs-Position

#### 3.4.5.2 BrbVc4DrawAxisRadial

```
float BrbVc4DrawAxisRadial(struct BrbVc4DrawAxisRadial_TYP* pAxis, struct BrbVc4General_TYP* pGeneral)
```

##### Argumente:

**struct** BrbVc4DrawAxisRadial\_TYP\* pAxis  
Zeiger auf die Instanz

**struct** BrbVc4General\_TYP\* pGeneral  
Zeiger auf die Instanz von „BrbVc4General\_TYP“

##### Rückgabe:

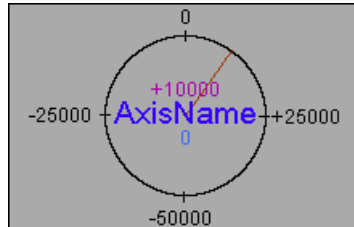
**REAL**

Der momentane Beginn der Skalierung entsprechend der Parametrierung

##### Beschreibung:

Mit den Eingängen kann die Darstellung der einzelnen Elemente sehr beeinflusst werden.  
So können z.B. die einzelnen Elemente ein- oder ausgeblendet und deren Farben gesetzt werden.

Beispiel-Darstellung:



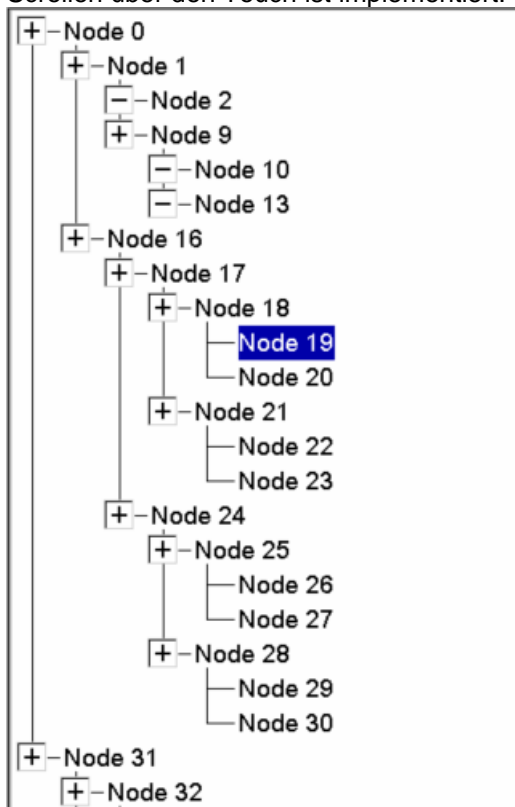
Die meisten Eingänge sind schon bei „BrbVc4DrawAxisLinear\_TYP“ dokumentiert. Hier werden nur die erweiterten Eingänge beschrieben:

– „nOffset“

Normalerweise beginnt die Skalierung am oberen Punkt des Kreises. Hier kann der Beginn der Skalierung um eine Gradanzahl von 0° bis 360° verschoben werden.

### 3.4.6 Treeview

Mit dieser Funktion kann eine Baumansicht dargestellt werden, deren Knoten vom Anwender definiert sind. Das Layout kann auf vielfache Weise auf die Anforderungen angepasst werden. Auch das optionale Scrollen über den Touch ist implementiert.



#### 3.4.6.1 Struktur

BrbVc4DrawTreeview_TYP		
bEnable	BOOL	1=Aktiv
nRedrawCounterMatch	UINT	Redraw-Cycle, bei dem gezeichnet wird
Cfg	BrbVc4DrawTreeviewCfg_TYP	Konfiguration des Treeviews
Ctrl	BrbVc4DrawTreeviewCtrl_TYP	Steuerung des Treeviews
State	BrbVc4DrawTreeviewState_TYP	Status des Treeviews
Intern	BrbVc4DrawTreeviewIntern_TYP	Interne Variablen für den Treeview

Der Funktionsblock wird nur ausgeführt, wenn der Eingang „bEnable“ auf 1 ist. Dann wird in jedem Zyklus die Touch-Bedienung ausgewertet. Gezeichnet wird nur, wenn zusätzlich „General.nRedrawCounter“ (siehe „BrbVc4General“) den Wert von „nRedrawCounterMatch“ entspricht. Damit kann das Zeichnen von mehreren Treeviews auf einer Seite auf verschiedene CPU-Zyklen aufgeteilt werden, um so eine gleichmäßigere Verteilung der CPU-Belastung zu erreichen.

### 3.4.6.2 Konfiguration

BrbVc4DrawTreeviewCfg_TYP		Konfiguration des Treeviews
Drawbox	BrbVc4Drawbox_TYP	Angaben zur Drawbox
Padding	BrbVc4DrawPadding_TYP	Einrückung des Tree-Bereichs
pSourceNodeList	UDINT	Eingang: Zeiger auf den Anfang der Knoten-Liste
nSourceArrayIndexMax	DINT	Eingang: Maximaler Index der Quell-Arrays
pInternNodeList	UDINT	Eingang: Zeiger auf den Anfang der intern benötigten Liste
Nodes	BrbVc4DrawTvCfgNodes_TYP	Konfiguration der Knoten
bCorrectScrollIndexY	BOOL	Eingang: 1=Korrigieren des ScrollIndexY
Scrollbar	BrbVc4DrawTvCfgScrollbar_TYP	Konfiguration der Scroll-Leiste
TouchAction	BrbVc4DrawTvCfgTouchAct_TYP	Konfiguration der Treeview-Touch-Aktion
Callbacks	BrbVc4DrawTvCfgCallbacks_TYP	Konfiguration der Treeview-Aufrufe
pTag	UDINT	Eingang: Zeiger auf Benutzer-Daten

Die Konfiguration ist der Übersichtlichkeit wegen in verschiedene Unter-Strukturen aufgeteilt.

#### 3.4.6.2.1 Allgemeines

BrbVc4Drawbox_TYP				
nLeft	UDINT	<input type="checkbox"/>		Eingang: Linke Koordinate
nTop	UDINT	<input type="checkbox"/>		Eingang: Obere Koordinate
nWidth	UDINT	<input type="checkbox"/>		Eingang: Breite
nHeight	UDINT	<input type="checkbox"/>		Eingang: Höhe
sFullName	STRING[nBRB_FILE_NAME_CHAR_MAX]	<input type="checkbox"/>		Pfad zur Anbindung
nBackgroundColor	UINT	<input type="checkbox"/>		Zur Anbindung an den Datenpunkt
nStatus	UINT	<input type="checkbox"/>		Zur Anbindung an den Datenpunkt

Die Angaben zur Drawbox sind korrekt auszufüllen, da nur dann alle Funktionalitäten richtig ausgeführt werden können. So werden z.B. die Koordinaten und Maße für die Touch-Funktionen benötigt. Der Name der Drawbox ist unbedingt auszufüllen, damit diese auch referenziert werden kann. Mit der Background-Color wird die Drawbox vor dem Zeichnen gelöscht.

BrbVc4DrawPadding_TYP				
nTop	DINT	<input type="checkbox"/>		Einrückung
nBottom	DINT	<input type="checkbox"/>		Eingang: Einrückung Oben
nLeft	DINT	<input type="checkbox"/>		Eingang: Einrückung Unten
nRight	DINT	<input type="checkbox"/>		Eingang: Einrückung Links
		<input type="checkbox"/>		Eingang: Einrückung Rechts

Das Padding legt die Einrückung des Baumbereichs fest. Allerdings werden nur ‚nTop‘ und ‚nLeft‘ berücksichtigt.

#### 3.4.6.2.2 pSourceNodeList und nSourceArrayIndexMax

Anwenderseitig muss ein Array vom Typ ‚BrbVc4TreeviewNode\_TYP‘ angelegt werden. Übergeben wird der Zeiger auf den Anfang des Arrays und der maximale Index des Arrays. Das Array muss groß genug sein, um alle Knoten aufnehmen zu können.

BrbVc4TreeviewNode_TYP				
nIndent	UINT			Eintrag der Knoten-Liste des Treeviews
bExpanded	BOOL			Eingang: Einzug des Knotens (0..n)
bChecked	BOOL			Eingang: 1=Aufgeklappt
nUserIconIndex	UINT			Eingang: 1=Angehakt
sText	STRING[nBRBVC4_TV_NODETEXT_CHAR_MAX]			Eingang: Index des benutzerdefinierten Icon-Bitmaps
pTag	UDINT			Eingang: Text des Knotens
				Eingang: Zeiger auf Benutzer-Daten

Durch diese flache Liste wird der Aufbau des Baumes und seine Knoten beschrieben.

‚nIndent‘ (0..n) gibt die Einzugs-Ebene eines Knotens an. Der erste Knoten **muss** immer den Indent ‚0‘ haben und wird diesbezüglich von der Funktion korrigiert.

Hat der nächste Knoten denselben Indent, so wird er als Knoten derselben Ebene interpretiert.

Hat der nächste Knoten einen um 1 höheren Indent, so wird er als Unterknoten interpretiert. Es ist darauf zu achten, den Indent nicht um mehr als 1 zu inkrementieren, da sonst die Anzeige nicht korrekt erfolgt!

Achtung: Der größtmögliche Indent beträgt **63**. Er wird in der Funktion auf diesen Wert begrenzt.

Über ‚bExpanded‘ kann festgelegt werden, ob der Knoten auf- oder zugeklappt erscheint.

‚bChecked‘ gibt an, ob die (optionale) Checkbox des Knotens an- oder abgehakt ist (siehe unten).

Mit ‚nUserIconIndex‘ kann der Bitmap-Index des (optionalen) Icons angegeben werden (siehe unten).



„sText“ enthält die Beschriftung des Knotens. Sie darf nicht mehr als 200 Zeichen haben.  
Der Zeiger „pTag“ wird von der Funktion nicht benutzt. Er kann vom Anwender als Zeiger auf Benutzer-Daten gesetzt und dann in den Callbacks (siehe unten) verwendet werden.

### 3.4.6.2.3 pInternNodeList

Die Funktion benötigt intern ermittelte Daten für jeden Knoten als Liste. Da die Anzahl der Knoten vom Anwender festgelegt wird, muss er auch diese Liste zur Verfügung stellen.

Daher muss anwenderseitig ein Array vom Typ „BrbVc4TreeviewInternNode\_TYP“ angelegt werden.

Übergeben wird der Zeiger auf den Anfang des Arrays.

Achtung: Das Array **muss** mindestens genauso so groß sein wie das Quell-Array!

BrbVc4TreeviewInternNode_TYP			Eintrag der internen Knoten-Liste des Treeviews
nNodeIndex	UINT		Intern: Index des Knotens in der Quell-Liste
bHasChild	BOOL		Intern: 1=Knoten hat Unterknoten
bHasFollowingNode	BOOL		Intern: 1=Knoten hat folgenden Knoten auf demselben Einzug
nNodeTop	DINT		Intern: Oberer Rand des Knotens
nNodeMiddleV	DINT		Intern: Mitte des Knotens
nNodeBottom	DINT		Intern: Unterer Rand des Knotens
Expandbox	BrbVc4Rectangle_TYP		Intern: Position und Größe der Aufklapp-Box
ExpandboxLine	BrbVc4Line_TYP		Intern: Position und Größe der Linie neben Aufklapp-Box
Checkbox	BrbVc4Rectangle_TYP		Intern: Position und Größe der Checkbox
UserIconBox	BrbVc4Rectangle_TYP		Intern: Position und Größe des benutzerdefinierten Icons
Text	BrbVc4DrawText_TYP		Intern: Angaben zum Zeichnen des Knoten-Textes

Die Einträge werden von der Funktion befüllt. Diese Angaben sollten nicht verändert werden, da sonst die korrekte Anzeige nicht garantiert werden kann. Sie können aber in den Callbacks (siehe unten) verwendet werden.

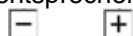
Die Liste enthält nur diejenigen Knoten, welche durch das Aufklappen sichtbar sind. Wird also ein Knoten zugeklappt, enthält sie weniger Knoten als das Quell-Array. Ein Knoten-Index des Quell-Arrays entspricht also nicht zwingend dem Index in diesem Array.

### 3.4.6.2.4 Nodes

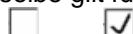
BrbVc4DrawTvCfgNodes_TYP			Konfiguration der Knoten
nLineColor	UINT		Eingang: Farbe der Verbindungs-Linien
nVerticalDistance	INT		Eingang: Zusätzliche vertikale Distanz zwischen den Knoten
nHorizontalDistance	UINT		Eingang: Zusätzliche horizontale Einrückung der Knoten
nExpandboxSize	UINT		Eingang: Größe des Aufklapp-Bitmaps
nBmplIndexUnexpanded	UINT		Eingang: Index des Bitmaps für zugeklappten Knoten
nBmplIndexExpanded	UINT		Eingang: Index des Bitmaps für aufgeklappten Knoten
nExpandBoxLineLength	UINT		Eingang: Länge der horizontalen Linie neben Aufklapp-Box
bShowCheckbox	BOOL		Eingang: 1=Checkboxen anzeigen
nCheckboxSize	UINT		Eingang: Größe des Checkbox-Bitmaps
nBmplIndexCheckboxOff	UINT		Eingang: Index des Bitmaps für nicht angehakte Checkbox
nBmplIndexCheckboxOn	UINT		Eingang: Index des Bitmaps für angehakte Checkbox
bShowUserIcon	BOOL		Eingang: 1=Benutzerdefinierte Icons anzeigen
nUserIconSize	UINT		Eingang: Größe des benutzerdefinierten Icon-Bitmaps
bShowSelection	BOOL		Eingang: 1=Ausgewählten Knoten hervorheben
nTextColorUnselected	UINT		Eingang: Farbe eines nicht ausgewählten Knotens
nTextColorSelected	UINT		Eingang: Farbe eines ausgewählten Knotens
Font	BrbVc4Font_TYP		Eingang: Font der Knoten

Diese Unterstruktur beinhaltet Angaben für das Layout des Baums und der Knoten.

Zur Anzeige der Aufklapp-Box müssen zwei Bitmaps in der Visu angelegt sein (zu- und aufgeklappt) und die entsprechenden Indizes übergeben werden:



Dasselbe gilt für die optionale Checkbox (ab- und angehakt):



Optional kann für jeden Knoten auch ein Icon angezeigt werden, dessen Index in der Quell-Liste übergeben wird. Es können beliebig viele verschiedene Bitmaps angelegt werden, z.B.:



Um die Knoten korrekt zeichnen zu können, müssen die quadratischen Größen der Bitmaps unbedingt richtig angegeben werden. Es wird empfohlen, alle Bitmaps in der gleichen Größe zur Verfügung zu stellen. So sieht das Layout am besten aus.

Beispiele für die benötigten Bitmaps werden im Beispielprojekt der Bibliothek mitgeliefert.

Für die optionale Anzeige des selektierten Knotens kann eine andere Farbe gewählt werden.

Für die verwendete Schrift müssen folgende Parameter übergeben werden:

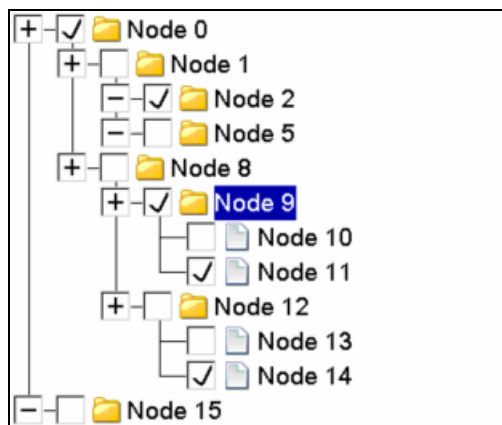
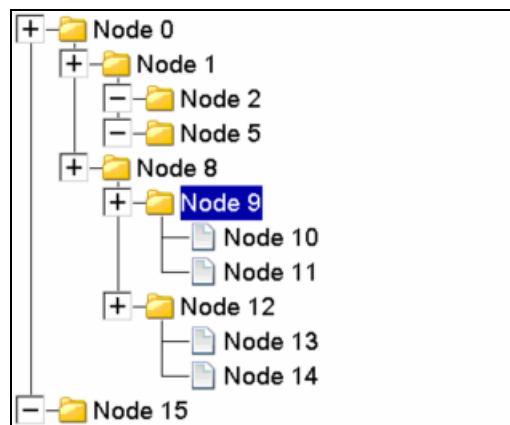
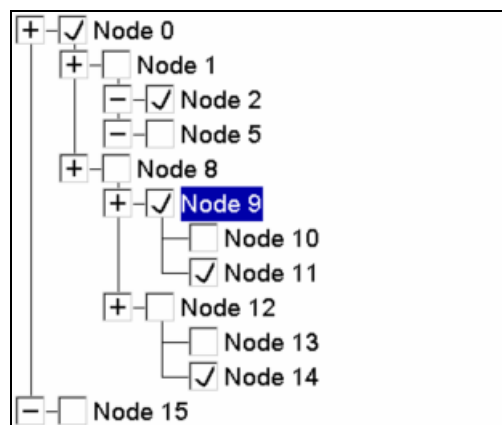
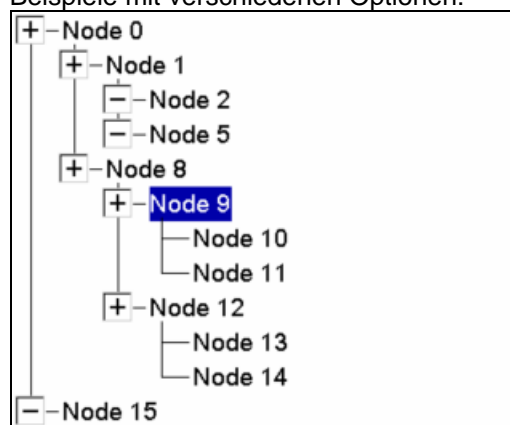
BrbVc4Font_TYP			
nIndex	UINT	<input type="checkbox"/>	Parameter: Index der Schrift
nCharWidth	UINT	<input type="checkbox"/>	Parameter: Durchschnittliche Breite eines Zeichens in [Pixel]
nCharHeight	UINT	<input type="checkbox"/>	Parameter: Höhe eines Zeichens in [Pixel]

Da die Pixel-Ausmaße eines Textes nicht ermittelt werden können, müssen hier die durchschnittliche Breite eines Zeichens sowie die Höhe der Schrift angegeben werden. Dann können Texte auch zentriert gezeichnet werden. Die Werte verändern die Position und Größe der Knoten und müssen daher unbedingt korrekt angegeben werden!

Durch die horizontale und die vertikale Distanz kann der Abstand der Knoten zueinander auf die Größe der Bitmaps und der Schrift abgestimmt werden, um das optimale Layout zu erhalten.

Es empfiehlt sich, das beste Layout empirisch (also durch Probieren) zu ermitteln.

Beispiele mit verschiedenen Optionen:



### 3.4.6.2.5 Korrigieren des ScrollOffsetY

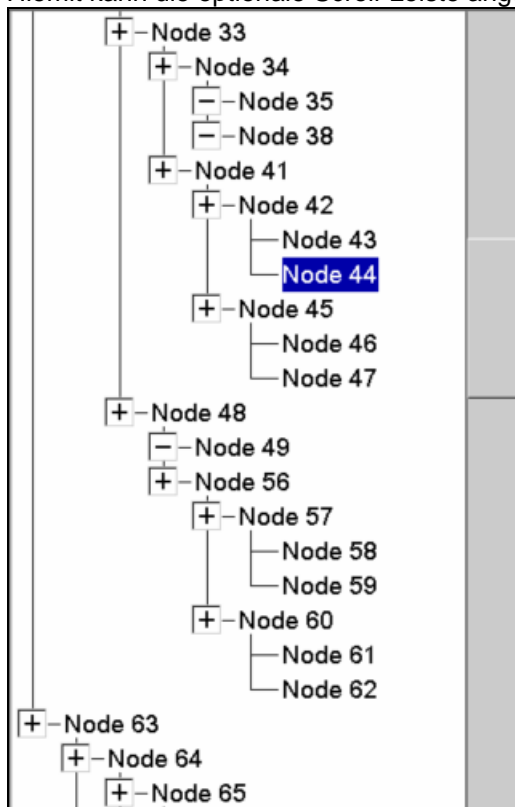
Der ScrollOffset in ‚Ctrl1‘ (siehe unten) gibt den Index des an oberster Position dargestellten Knotens an. Dieser Index bezieht sich auf die Quell-Liste. Ist dieser Knoten nicht sichtbar (weil zugeklappt), so wird der nächste sichtbare Knoten an oberster Position dargestellt.

Ist nun diese Option eingeschaltet, wird der Scroll-Index automatisch auf den richtigen Wert korrigiert. Dies ist nützlich, wenn das Scrollen anwenderseitig gemacht wird, z.B. über eine anwenderseitige Scrollbar.

### 3.4.6.2.6 Scrollbar

BrbVc4DrawTvCfgScrollbar_TYP		Konfiguration der Scroll-Leiste
bShow	BOOL	Eingang: 1=Scroll-Leiste anzeigen
nWidth	UINT	Eingang: Breite der Scroll-Leiste
nScrollbarColor	UINT	Eingang: Farbe der Scroll-Leiste
nMarkerBackColor	UINT	Eingang: Hintergrund-Farbe der Scroll-Marke
nMarkerBorderColorTop	UINT	Eingang: Farbe des oberen Scroll-Marken-Randes
nMarkerBorderColorBottom	UINT	Eingang: Farbe des unteren Scroll-Marken-Randes
nMarkerHeightMin	UINT	Eingang: Mindest-Größe der Scroll-Marke

Hiemit kann die optionale Scroll-Leiste angepasst werden:



Die Farbangaben ‚nMarkerBorderColorTop‘ und ‚nMarkerBorderColorBottom‘ dienen dazu, der Scrollmarke ein 3-dimensionales Aussehen zu verleihen.

Die Größe der Marke wird über die Anzahl der Knoten bestimmt. Damit sie bei sehr vielen Knoten nicht zu klein wird und dadurch nicht mehr mit dem Finger getroffen wird, kann mit dem Parameter ‚nMarkerHeightMin‘ eine Mindestgröße angegeben werden.

Hinweis: Die Touch-Funktion der Scroll-Leiste muss extra aktiviert werden (siehe unten).

### 3.4.6.2.7 TouchAction – Funktion des Touchs

BrbVc4DrawTvCfgTouchAct_TYP		Konfiguration der Treeview-Touch-Aktion
BorderCorrection	BrbVc4DrawTouchBorderCorr_TYP	Touch-Korrektur des Rahmens
bSelect	BOOL	Eingang: 1=Auswahl aktivieren
bExpand	BOOL	Eingang: 1=Auf-/Zuklappen aktivieren
bCheck	BOOL	Eingang: 1=An-/Abhaken aktivieren
bScrollyList	BOOL	Eingang: 1=ScrollingY in der Liste aktivieren
bScrollyBar	BOOL	Eingang: 1=ScrollingY auf der Scroll-Leiste aktivieren

BrbVc4DrawTouchBorderCorr_TYP		Touch-Korrektur des Rahmens
nX	SINT	<input type="checkbox"/> Eingang: OffsetX für die Touch-Korrektur des Rahmens
nY	SINT	<input type="checkbox"/> Eingang: OffsetY für die Touch-Korrektur des Rahmens

Hier wird festgelegt, ob und welche Funktion durch den Touch bedienbar ist.

Mit der "BorderCorrection" kann ein Offset festgelegt werden, welcher die Koordinaten des Touchs aufgrund des Rahmens der Drawbox korrigiert. Der Rahmen „Flat\_back“ z.B. muss jeweils mit dem Wert 2 korrigiert werden, damit ein Klick auch exakt an dem berührten Punkt erkannt wird.

Das Setzen der Auswahl wird schon beim einmaligen Klicken ausgeführt.

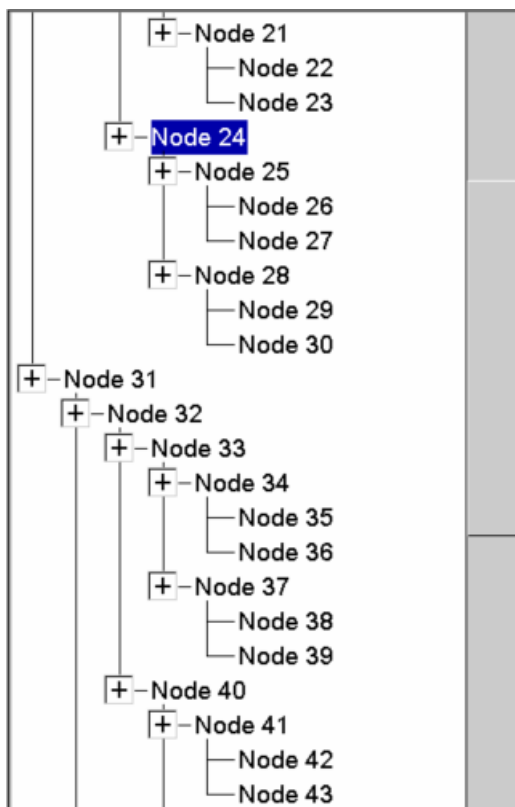
Das Auf-/Zuklappen sowie das An-/Abhaken wird durch Klick auf das entsprechende Bitmap ausgelöst.

Das Scrollen mit dem Touch ist optional über 2 Arten verfügbar, welche auch beide gleichzeitig aktiviert sein können:

#### 3.4.6.2.7.1 Scrollen über die Liste

Das vertikale Ziehen in der Liste löst das Scrolling aus. Es wird dabei in die Richtung gescrollt, in die auch gezogen wird.

#### 3.4.6.2.7.2 Scrollen über die Scroll-Leiste



Die Scroll-Leiste hat mehrere Touchfunktionen:

- Klicken oberhalb der Marke: Seitenweises Scrollen der Liste nach oben
- Klicken unterhalb der Marke: Seitenweises Scrollen der Liste nach unten
- Klicken und Ziehen auf der Marke: Zeilenweises Scrollen in die gezogene Richtung. Dies eignet sich besonders gut zum schnellen Scrollen bei großen Listen.

Die Funktion entspricht somit im Wesentlichen einer Windows-Scrollbar.

Hinweis: Die Anzeige der Scroll-Leiste muss extra aktiviert werden (siehe oben).

### 3.4.6.2.8 Callbacks

Hinweis: Diese Funktionalität ist aufgrund von Funktionszeigern nur in ANSI-C nutzbar, aber nicht in IEC-Sprachen (siehe Punkt [Hinweise zu StructuredText und anderen IEC-Sprachen](#))

BrbVc4DrawTvCfgCallbacks_TYP		Konfiguration der Treeview-Aufrufe
pCallbackAfterClear	UDINT	Eingang: Funktions-Zeiger für Aufruf nach Löschen der Drawbox
pCallbackBeforeNode	UDINT	Eingang: Funktions-Zeiger für Aufruf vor Zeichnen eines Knotens
pCallbackAfterNode	UDINT	Eingang: Funktions-Zeiger für Aufruf nach Zeichnen eines Knotens
pCallbackAfterNodes	UDINT	Eingang: Funktions-Zeiger für Aufruf nach Zeichnen aller Knoten
pCallbackAfterScrollbar	UDINT	Eingang: Funktions-Zeiger für Aufruf nach Zeichnen der Scroll-Leiste

Ein Callback ist ein Aufruf einer vom Anwender geschriebenen Funktion während des Zeichnens.

Er arbeitet mit sogenannten Funktions-Zeigern. Dabei wird die Adresse einer Funktion übergeben, welche der Anwender selbst schreibt. Lediglich die Signatur, also die Anzahl, Reihenfolge und die Datentypen der Argumente sind dabei vorgeschrieben. Der Inhalt der Funktion bleibt vollkommen dem Anwender überlassen.

Es gibt 5 verschiedene Callbacks (siehe oben), welche nach dem Zeichnen des jeweiligen Elements aufgerufen werden können.

Für jeden Callback gibt es ein Muster der Signatur in der Datei „BrbVc4TreeviewCallbackTemplates.c“ in der Bibliothek:

```
unsigned short BrbVc4TreeviewCallbackAfterClear(struct BrbVc4DrawTreeview_TYP* pTreeview)
```

```
unsigned short BrbVc4TreeviewCallbackBeforeNode(struct BrbVc4DrawTreeview_TYP* pTreeview, UINT  
nNodeIndex, UINT nNodeIndexIntern)
```

```
unsigned short BrbVc4TreeviewCallbackAfterNode(struct BrbVc4DrawTreeview_TYP* pTreeview, UINT  
nNodeIndex, UINT nNodeIndexIntern)
```

```
unsigned short BrbVc4TreeviewCallbackAfterNodes(struct BrbVc4DrawTreeview_TYP* pTreeview)
```

```
unsigned short BrbVc4TreeviewCallbackAfterScrollbar(struct BrbVc4DrawTreeview_TYP* pTreeview)
```

Manche Callbacks werden während des Zeichnens mehrmals aufgerufen, so z.B. nach dem Zeichnen jeden Knotens. Über Argumente werden aktuelle Werte übergeben, z.B. die Indizes des aktuellen Knotens für die Quell- und die interne Liste.

Im Callback kann auf die gesamte Trend-Struktur zugegriffen werden, auch auf die intern berechneten Daten. ACHTUNG: Es sollten aber keine Werte verändert werden!

Soll ein Callback aktiviert werden, so ist dessen Adresse in die obige Struktur einzutragen.

Beispiel:

Es wird eine Funktion angelegt, welche dem Muster entspricht:

```
unsigned short BrbVc4TreeviewCallbackAfterClear(struct BrbVc4DrawTreeview_TYP* pTreeview)  
{  
    // Anwender-Code  
    return 0;  
}
```

Vor dem Aufruf der Trend-Funktion wird die Adresse des Callbacks übergeben

```
Treeview.Cfg.Callbacks.pCallbackAfterClear = (UDINT) &BrbVc4TreeviewCallbackAfterClear;
```

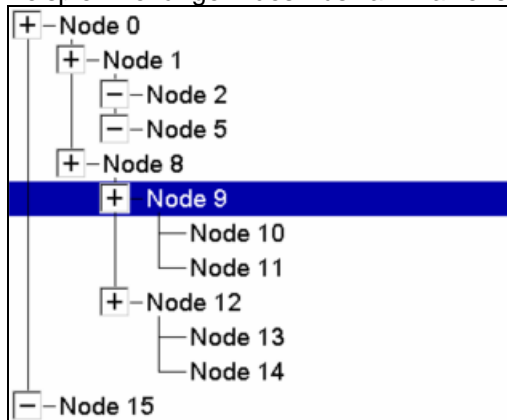
Innerhalb des Callbacks kann dann mit Zeichenfunktionen die visuelle Ausgabe erweitert werden, z.B. Texte oder zusätzliche Linien eingezeichnet werden.

Dadurch ergibt sich eine Vielfalt an Möglichkeiten, z.B. könnten die Knoten mit weiteren Ausgaben ergänzt werden.

Der Rückgabewert der Funktion ist egal.

ACHTUNG: Das Koordinaten-System bezieht sich auf die Treeview-Drawbox, weil diese noch referenziert ist.


Beispiel: Verlängern des Auswahl-Balkens über die komplette Breite:



#### 3.4.6.2.9 pTag

Dieser Zeiger wird von der Funktion nicht benutzt. Er kann vom Anwender als Zeiger auf Benutzer-Daten gesetzt und dann in den Callbacks verwendet werden.

#### 3.4.6.3 Steuerung






 BrbVc4DrawTreeviewCtrl_TYP	Steuerung des Treeviews
 nScrollOffsetY	UINT Eingang: Index des Knotens, welcher als erster gezeichnet wird
 nSelectedIndex	UINT Eingang: Index des Knotens, welcher ausgewählt ist

Über den Eingang ‚nScrollOffsetY‘ kann der Index des Knotens, der an oberster Position dargestellt werden soll, festgelegt oder ausgelesen werden. Ist dieser nicht sichtbar (weil zugeklappt), so wird der nächste sichtbare Knoten an oberster Position dargestellt.

Mit ‚nSelectedIndex‘ kann festgelegt oder ausgelesen werden, welche Knoten gerade selektiert ist. Es kann auch ein Knoten angegeben werden, der gerade nicht sichtbar ist (zugeklappt oder außerhalb des Scroll-Ausschnitts).

#### 3.4.6.4 Status

Diese Struktur enthält Ausgangs-Daten, welche für den Anwender interessant sein könnten:

Diese Struktur enthält Ausgänge Daten, welche für den Anwender interessant sein könnten:			
	BrbVc4DrawTreeviewState_TYP		Status des Treeviews
	 eTouchAction	BrbVc4TreeviewTouchAction_ENUM	Ausgang: Status einer Touch-Aktion
	 nInternNodeListCount	UINT	Ausgang: Anzahl der gültigen Einträge in der internen Liste

Der Ausgang „eTouchAction“ gibt den momentanen Stand der Touch-Bedienung an:

1	BrbVc4TreeviewTouchAction_ENUM	Touch-Aktionen des Treeviews
12	eBRBVC4_TV_TOUCHACT_NONE	Keine Touch-Aktion des Treeviews
12	eBRBVC4_TV_TOUCHACT_SELECT	Knoten wurde selektiert
12	eBRBVC4_TV_TOUCHACT_DOUBLECLICK	Doppelklick
12	eBRBVC4_TV_TOUCHACT_EXPANDED	Knoten wurde aufgeklappt
12	eBRBVC4_TV_TOUCHACT_UNEXPANDED	Knoten wurde zugeklappt
12	eBRBVC4_TV_TOUCHACT_CHECKED	Knoten wurde angehakt
12	eBRBVC4_TV_TOUCHACT_UNCHECKED	Knoten wurde abgehakt
12	eBRBVC4_TV_TOUCHACT_SCROLL_LIST	Momentan wird über die Liste gescrollt
12	eBRBVC4_TV_TOUCHACT_SCROLL_BAR	Momentan wird über die Leiste gescrollt

Abhängig vom Status bleibt er nur einen oder auch mehrere Zyklen anstehen.

Der Ausgang „nInternNodeListCount“ gibt die momentane Anzahl der gültigen Einträge in der internen Knoten-Liste (siehe oben) an.

### 3.4.6.5 Intern

Diese Strukturen werden während des Aufrufs mit intern berechneten Daten gefüllt, z.B. Koordinaten und Maße einzelner Zeichen-Elemente. So sind z.B. berechnete Abstände enthalten.

Auf sie darf während eines Callbacks lesend zugegriffen werden, um eigene Elemente positionieren zu können. Auf keinen Fall sollten sie verändert werden!

### 3.4.6.6 BrbVc4DrawTreeview

```
unsigned short BrbVc4DrawTreeview(struct BrbVc4DrawTreeview_TYP* pTreeview, struct
BrbVc4General_TYP* pGeneral)
```

#### Argumente:

```
struct BrbVc4DrawTreeview_TYP* pTreeview
    Zeiger auf die Instanz
struct BrbVc4General_TYP* pGeneral
    Zeiger auf die Instanz von „BrbVc4General_TYP“
```

#### Rückgabe:

```
UINT
    Immer 0
```

#### Beschreibung:

Zeichnet einen Treeview nach Vorgaben. Der Aufruf sollte in der Restzeit-Task erfolgen. Es empfiehlt sich, die Instanz im Init des Tasks komplett auf 0 zu setzen.

### 3.4.6.7 BrbVc4GetTreeviewInternNodeIndex

```
signed long BrbVc4GetTreeviewInternNodeIndex(struct BrbVc4DrawTreeview_TYP* pTreeview, unsigned
long nIndex)
```

#### Argumente:

```
struct BrbVc4DrawTreeview_TYP* pTreeview
    Zeiger auf die Instanz
UDINT nIndex
    Index des Knotens in der Quell-Liste
```

#### Rückgabe:

```
DINT
    Index des Knotens in der internen Liste
    -1, wenn nicht vorhanden
```

#### Beschreibung:

Wenn ein Knoten zugeklappt ist, sind dessen Unter-Knoten nicht in der internen Liste enthalten. Ein Knoten-Index des Quell-Arrays entspricht also nicht zwingend dem Index im internen Array (siehe oben).



Diese Funktion gibt den Index der internen Knoten-Liste aufgrund eines Indizes der Quell-Knoten-Liste zurück. Ist dieser nicht in der internen Liste vorhanden (z.B. weil ein Oberknoten zugeklappt ist), wird als Ergebnis -1 zurückgeliefert.

Durch den implementierten Algorithmus wird dabei nicht die ganze interne Liste durchsucht. Vielmehr wird der Suchbereich solange halbiert, bis nur noch ein kleiner Bereich durchsucht werden muss.

Trotz dieser Optimierung braucht eine große Liste auch entsprechend länger zum Ermitteln des gesuchten Indizes. Deshalb sollte der Aufruf dieser Funktion nur in der Restzeit-Task erfolgen.