bmc messsysteme gmbh

NextView®4 Script

Event-orientierte Skriptsprache
für NextView®4

Programmierhandbuch

Version 4.6

www.bmcm.de
bavarian measurement company munich

Inhaltsverzeichnis

1 Überblick

1.1 Einleitung

NextView®4 Script ist eine einfache Skriptsprache zur Erstellung spezieller messtechnischer Anwendungen in Kombination mit der professionellen Messdatenerfassungs- und Verarbeitungssoftware **NextView®4**. Im wesentlichen basierend auf der Programmiersprache BASIC wurde **NextView®4 Script** jedoch um spezielle Befehle und Funktionen erweitert, um die Standardbefehle und Features von **NextView®4** auf ideale Weise zu ergänzen.

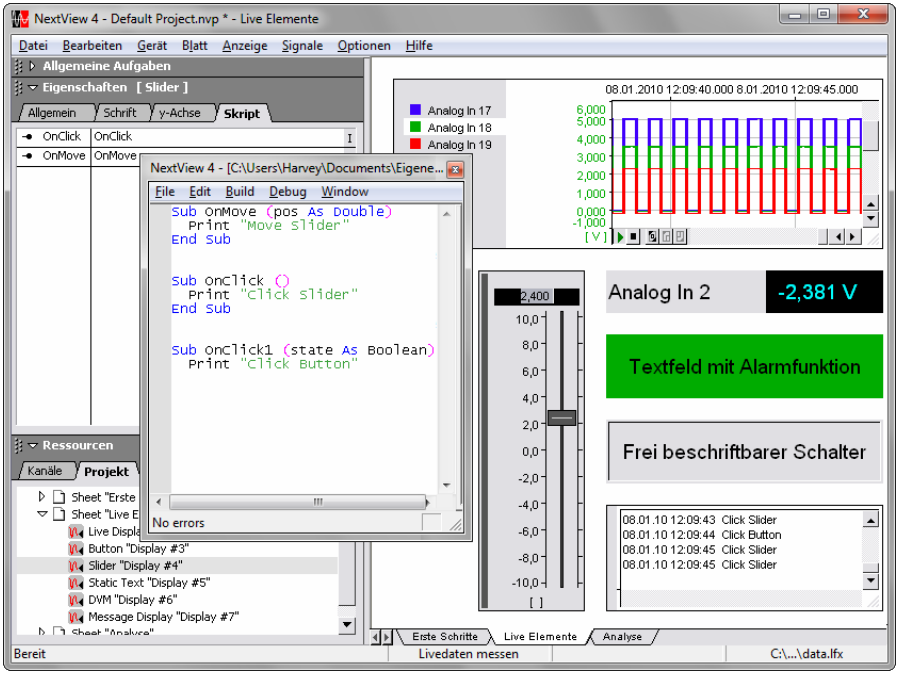

Abbildung 1

Mit **NextView®4 Script** lassen sich kundenspezifische Messaufgaben durch einfache Programmierung direkt verwirklichen, Prozesse automatisieren und steuern.

Als optional erhältliches Zusatzmodul (kostenpflichtig) leistet **NextView®4 Script** einen wichtigen Beitrag die Software **NextView®4** noch vielseitiger und leistungsfähiger zu gestalten ohne dabei einer wesentlichen Eigenschaft von **NextView®4** untreu zu werden: der Benutzerfreundlichkeit. Durch die Verwendung der weit verbreiteten und leicht zu erlernenden Programmiersprache BASIC als Grundlage für **NextView®4 Script** können Anwendungen ohne große Einarbeitungszeiten problemlos realisiert werden.

Dies Handbuch beschreibt den Aufbau eines **NextView®4 Script** Programms und erklärt die wichtigsten Befehle mit Betonung auf die speziellen Eigenschaften und Funktionen, die die Skriptsprache **NextView®4 Script** bietet, stellt jedoch keine Referenz für die Programmiersprache BASIC dar. Um die Programmierung mit BASIC zu erlernen, sollte auf alle Fälle ein Programmierhandbuch für BASIC zu Rate gezogen werden.

Einzelne Beispielskripte für die bmcm Messhardware stehen im Downloadbereich der bmcm Website zur Verfügung.

1.2 BMC Messsysteme GmbH

BMC Messsysteme GmbH steht für innovative Messtechnik "made in Germany". Vom Sensor bis zur Software bieten wir alle für die Messkette benötigten Komponenten an.

Unsere Hard- und Software ist aufeinander abgestimmt und dadurch besonders anwenderfreundlich. Darüber hinaus legen wir größten Wert auf die Einhaltung gängiger Industriestandards, die das Zusammenspiel vieler Komponenten erleichtern.

BMC Messsysteme Produkte finden Sie im industriellen Großeinsatz ebenso wie in Forschung und Entwicklung oder im privaten Anwenderbereich. Wir fertigen unter Einhaltung der ISO-9000-Vorschriften, denn Standards und Zuverlässigkeit sind uns wichtig - für Sie und für uns!

Neueste Informationen finden Sie im Internet auf unserer Homepage unter http://www.bmcm.de.

1.3 Urheberrechte

Die Programmiersprache **NextView®4 Script** mit allen Erweiterungen wurde mit größtmöglicher Sorgfalt erstellt und geprüft. Die BMC Messsysteme GmbH gibt keine Garantien, weder in Bezug auf dieses Handbuch noch in Bezug auf die in diesem Buch beschriebene Hard- und Software, ihre Qualität, Durchführbarkeit oder Verwendbarkeit für einen bestimmten Zweck. Die BMC Messsysteme GmbH haftet in keinem Fall für direkt oder indirekt verursachte oder erfolgte Schäden, die entweder aus unsachgemäßer Bedienung oder aus irgendwelchen Fehlern am System resultieren. Änderungen, die dem technischen Fortschritt dienen, bleiben uns vorbehalten.

Stand: 20. September 2014

BMC Messsysteme GmbH

Hauptstraße 21
82216 Maisach
DEUTSCHLAND

Tel.: +49 8141/404180-1
Fax: +49 8141/404180-9
E-Mail: info@bmcm.de

2 Programmiergrundlagen

2.1 Allgemeines

2.1.1 Dateityp *.nvs

Ein mit **NextView®4 Script** erstelltes Programm wird als Skriptdatei im Format **`*.nvs`** gespeichert. Es enthält sowohl einen Deklarationsteil, der alle verwendeten Datentypen auflistet und benennt, als auch einen Anwendungsteil mit den verwendeten Befehlen, Prozeduren und Funktionen.

Dabei muss sich der Code, der in ein Skript geschrieben wird, nicht zwingend auf eine einzige Applikation beziehen, sondern kann, wenn er allgemein gehalten ist, in verschiedenen Anwendungen verwendet werden.

2.1.2 Nicht zu verwendende Zeichen

Folgende Zeichen sind im Programmcode zu vermeiden:

- Umlaute
- Sonderzeichen (z. B. @, §, ...)
- landesspezifische Buchstaben (z. B. ß)

2.1.3 Kommentare

Um Programmcode besser nachvollziehen zu können, verwendet man Kommentare. Alles was nach dem Zeichen `'` steht, wird beim Programmdurchlauf nicht berücksichtigt.

Kommentare können hinter einem Programmbefehl (jedoch nicht hinter einem Zeilenumbruch!) stehen oder eine ganze Zeile einnehmen. Gehen sie über mehrere Zeilen, muss jedes Mal das Kommentarzeichen `'` erneut gesetzt werden.

```
' Beispiel eines Kommentars über
' mehrere Zeilen

Print "Hello World!" ' Kommentarbeispiel nach Programmbefehl
```

2.1.4 Zeilenumbruch

Um die Übersichtlichkeit eines Programms zu erhöhen, empfiehlt sich bei langen Programmzeilen unter Umständen ein Zeilenumbruch. Dazu wird an die Befehlszeile ein Unterstrich, dem ein Leerzeichen vorangeht, angehängt: (_)

```
Print "Der Flächeninhalt ist Radius * Radius * Pi " _
      & radius * radius * 3.141592653
```

Kommentare sind nach einem Zeilenumbruch nicht zugelassen!

2.1.5 Darstellung nicht dezimaler Zahlensysteme

Im Unterschied zu Dezimalzahlen werden Hexadezimalzahlen mit der Präfix **&H** geschrieben. Zahlen aus dem Oktalsystem erhalten den Vorsatz **&O**, aus dem Binärsystem **&B**. Im Dezimalsystem ist der Vorsatz **&D** möglich aber nicht zwingend erforderlich.

```
Dim i As Integer
i = &H10       ' setze i auf 10 Hex (Dezimalsystem: 16)
```

2.1.6 Namenskonventionen

Bei der Benennung von Variablen, Konstanten, Programmen etc. ist auf folgende Namenskonventionen zu achten:

- Eindeutigkeit
- 1. Zeichen muss ein Buchstabe sein
- max. 256 Zeichen
- keine Verwendung von Leerzeichen, Punkten oder anderen Sonderzeichen (Ausnahme: Unterstrich _), insbesondere Kurzformen von Variablentypen

2.2 Datentyp

Eine wesentliche Aufgabe von Programmen ist die Verarbeitung von Daten verschiedenster Art. Datentypen bestimmen die Eigenschaft der Werte und deren Wertebereich. Folgende Datentypen stehen in **NextView®4 Script** zur Verfügung:

Datentyp	Wertebereich	Anmerkungen
Byte	0..255	nur ganze Zahlen im Wertebereich
Integer	-2147483648 .. 2147483647	nur ganze Zahlen im Wertebereich
Double	$-1{,}79769313486232*10^{308}$ .. $1{,}79769313486232*10^{308}$	Fließkommazahl mit zehnstelliger Genauigkeit
String	alle Zeichen des ASCII-Codes	Zeichenkette (Groß-/ Kleinbuchstaben, Ziffern, Sonderzeichen etc.)
Boolean	True, False	signalisiert den Zustand Boolescher Ausdrücke oder Variablen
Klassen		Klasse (z. B. Channel)
Array		Werte gleichen Datentyps mit Indizierung

Werte vom Typ *Double* werden exponentiell folgendermaßen geschrieben:

<Mantisse>D<Exponent> bzw. **<Mantisse>d<Exponent>** oder
<Mantisse>E<Exponent> bzw. **<Mantisse>e<Exponent**

Der Exponent bezieht sich immer auf die Basis 10. Dabei ist beim Exponenten auf das Vorzeichen zu achten:

45D+3 = $45*10^{3}$ = 45000 bzw. **45D-3 = $45*10^{-3}$ = 0,045**

Bei Datentypen wird zwischen einfachen Datentypen, Klassen und Arrays unterschieden. Einfache Datentypen sind *Byte*, *Integer*, *Double*, *String* und *Boolean*.

Windows® Datum/Uhrzeit Format:

Eine Windows® Datum-Uhrzeit-Angabe wird als Double-Floatingpoint-Zahl gespeichert. Vor dem Dezimalpunkt steht die Anzahl an Tagen seit dem 30.12.1899. Der Nachkommawert gibt den fraktionalen Teil des Tages an (z. B. 0.5 entspricht 12:00Uhr Mittag).

2.3 Variablen

Variablen bestehen aus folgenden Elementen: dem Variablennamen, ihrem Eigenwert und dem Datentyp. Bei der Vergabe des Variablennamens muss auf die Einhaltung der Namenskonventionen geachtet werden. Im Gegensatz zu Konstanten kann sich bei Variablen der Eigenwert im Verlauf des Programms ändern, so dass sich mit Hilfe von Variablen Zwischenwerte speichern lassen.

2.3.1 Globale und lokale Variablen

Zwischen globalen und lokalen Variablen muss unterschieden werden: globale Variablen werden im Deklarationsteil des Programms angegeben und gelten damit immer, lokale Variablen werden innerhalb einer Prozedur definiert und existieren nur während die Prozedur durchlaufen wird.

Wird für eine globale und eine lokale Variable der gleiche Name verwendet, gilt in der Prozedur, für die die lokale Variable definiert wurde, immer die lokale Variable (s. "Gültigkeitsbereiche", S. 50).

2.3.2 Variablendeklaration

Variablen müssen immer explizit deklariert werden. Dies erfolgt mit dem Schlüsselwort **`Dim`** am Anfang jeder Zeile.

```
Dim Variablenname As Datentyp
```

Bei Deklaration mehrerer Variablen in einer Zeile müssen diese durch Komma voneinander getrennt werden.


```
Dim Var1[, Var2, ...] As Datentyp
```

```
Dim Zahl As Double
Dim i, j As Integer
```

2.3.3 Wertzuweisung an Variablen

Folgender Befehl weist einer Variablen einen Wert zu. Dieser muss sich im Wertebereich seines deklarierten Variablenbereichs befinden.

```
Variablenname = Variablenwert
```

Dabei kann der Ausdruck auf der rechten Seite selbst eine Variable enthalten.

```
Zahl = 23
Zahl = Zahl + 1
```

Man kann einer numerischen Variablen manchmal auch eine Zeichenkette zuweisen, wenn diese selbst ein numerischer Wert ist. Die Zeichenkette wird automatisch in eine Zahl umgewandelt (s. "Ausdruckskonvertierung", S. 31), jedoch sollten Zuweisungen dieser Art die Ausnahme bleiben, da sie sehr fehlerträchtig sind.

```
Dim s As String
Dim i As Integer

s = 123
i = s + 1
```

2.4 Konstanten

Konstanten besitzen einen Namen, einen Datentyp und einen festen Eigenwert, der zu Beginn des Programms bereits bekannt ist. Im Gegensatz zu Variablen ist der Wert von Konstanten zur Laufzeit des Programms nicht veränderbar. Bei der Ver-

gabe des Namens muss auf die Einhaltung der Namenskonventionen geachtet werden.

Konstanten werden folgendermaßen deklariert. Dabei muss sich der auf der rechten Seite zugewiesene Wert im Wertebereich des angegebenen Datentyps befinden.

```
Const Konstantenname As Datentyp = Wert
```

An Konstanten können nur einfache Datentypen (*Byte*, *Integer*, *Double*, *String*, *Boolean*) übergeben werden. Konstante *Arrays* oder *Klassen* sind nicht möglich.

2.5 Felder (Arrays)

In Feldern werden im Gegensatz zu einfachen Datentypen, wie *Integer* oder *String* mehrere Werte gleichen Typs abgespeichert.

Der Zugriff erfolgt über einen Feldnamen und einen Index, durch den die Elemente eines Feldes voneinander unterscheidbar sind.

Felder ermöglichen durch die Wiederholung von Vorgängen eine erhebliche Vereinfachung im Programmcode. Mit Hilfe von Schleifen können beispielsweise nur durch Veränderung der Indexnummer Felder mit großen Datenmengen durchlaufen werden.

Felder besitzen eine obere und eine untere Grenze und die Elemente eines Feldes besitzen eine definierte Position. Der Index beginnt immer an der unteren Grenze und beträgt 0, falls nicht anders deklariert, und wird dann fortlaufend um eins hochgezählt.

2.5.1 Arraydeklaration

Mit dem Schlüsselwort **`Dim`** wird bei der Deklaration von Feldern immer der Feldname, direkt gefolgt vom höchsten Indexwert in runden Klammern und der verwendete Datentyp angegeben:

```
Dim Arrayname(oberer Index) As Datentyp
```
oder

```
Dim Arrayname(unterer_Index to oberer_Index) As Datentyp
```

Der Array Datentyp kann nur ein einfacher Datentyp (*Byte*, *Integer*, *Double*, *String*, *Boolean*) sein.

2.5.2 Zugriff auf Elemente eines Arrays

Über den Feldnamen und den spezifischen Index wird einem Feldelement sein Wert zugewiesen. Es muss dabei sichergestellt werden, dass sich der zugewiesene Wert im Wertebereich des verwendeten Datentyps befindet.

```
Arrayname(Index) = Wert
```

Wird anstelle des Index eine Variable verwendet, lässt sich die Wertzuweisung für Arrays bei Verwendung einer Zählschleife meist erheblich vereinfachen.

Ebenso kann umgekehrt durch Angabe des Arraynamens direkt gefolgt vom Index in runden Klammern der Inhalt eines Feldelements ausgelesen werden:

```
Arrayname(Index)
```

2.5.3 Mehrdimensionale Arrays

Manchmal haben die Elemente eines (eindimensionalen) Arrays auch Bezug zu Elementen anderer Felder (Arrays). Um diese zusätzliche Information darstellen zu können, empfiehlt sich die Verwendung eines mehrdimensionalen Arrays.

Diese werden mit Hilfe von mehreren Indizes definiert. Dabei ist zu beachten, dass je höher die Dimension des Arrays, desto komplexer und unübersichtlicher die Struktur und desto größer auch der Speicherplatzbedarf.

```
Dim Arrayname(Index 1, Index 2, .. , Index n) As Datentyp
```

Der Index eines Feldes startet immer bei 0 und geht bis zum in der **`Dim`** Definition angegebenen Wert. Zum Beispiel legt **`Dim Feld(3) as Integer`** ein Feld an mit 4 Integer Elementen.

Mit Hilfe von ineinander geschachtelten Zählschleifen lassen sich mehrdimensionale Felder effizient verarbeiten. Das folgende Beispiel initialisiert ein zweidimensionales Feld der Größe 5x4 mit den Produkten der Indexwerte:

```
Dim i, j As Integer
Dim Feld(4, 3) As Integer

For i  = 0 to 4
    For j = 0 to 3
        Feld(i, j) = i * j
    Next
Next
```

i j	0	1	2	3	4
0	(0, 0) 0	(1, 0) 0	(2, 0) 0	(3, 0) 0	(4, 0) 0
1	(0, 1) 0	(1, 1) 1	(2, 1) 2	(3, 1) 3	(4, 1) 4
2	(0, 2) 0	(1, 2) 2	(2, 2) 4	(3, 2) 6	(4, 2) 8
3	(0, 3) 0	(1, 3) 3	(2, 3) 6	(3, 3) 9	(4, 3) 12

2.5.4 Dynamische Arrays

Die von **NextView®4 Script** zur Verfügung gestellten Felder (Arrays) sind immer dynamisch, das heißt ihre Größe kann im Verlauf des Programms jederzeit geändert werden. Dies schafft hohe Flexibilität. Beispielsweise lässt sich durch ständige Anpassung der Feldgröße kurzzeitig ein großes Feld verwenden und, sobald Daten nicht mehr benötigt werden, das Feld verkleinern und dadurch viel Speicherplatz sparen.

Die Deklaration eines dynamischen Arrays erfolgt mit Hilfe des Schlüsselwortes **`Dim`** ohne Angabe von Grenzen. Erst später werden die Grenzen mit dem Befehl **`ReDim`** festgelegt jedoch ohne Angabe des Datentyps.

```
Dim Arrayname() As Datentyp
...
ReDim Arrayname(Index 1, Index 2, .. , Index n)
```

Bei Anpassung der Feldgröße gehen alle im Feld gespeicherten Werte verloren.

Nach Ausführung der **`ReDim`** Anweisung werden alle Feldelemente auf die in der Tabelle abgebildeten Werte in Abhängigkeit ihres Datentyps zurückgesetzt. Dies ist von Vorteil, wenn einem Feld neue Daten zugewiesen werden sollen oder um Speicherplatz zu sparen.

Datentyp	Neuer Wert
Byte	0
Integer	0
Double	0
String	""
Boolean	False
Klassen	Nothing

2.5.5 Befehle im Zusammenhang mit Arrays

Die folgenden Befehle finden häufig in Verbindung mit Feldern ihre Anwendung:

Befehl	Beschreibung
`LBound(Arrayname)`	gibt den Index der unteren Grenze zurück
`LBound(Arrayname, n)`	gibt bei mehrdimensionalen Arrays den Index der unteren Grenze der n-ten Dimension zurück
`UBound(Arrayname)`	gibt den Index der oberen Grenze zurück
`UBound(Arrayname, n)`	gibt bei mehrdimensionalen Arrays den Index der oberen Grenze der n-ten Dimension zurück

2.6 Ausdrücke und Operatoren

Ein wichtiger Baustein einer Skriptsprache sind Ausdrücke. Die einfachste Möglichkeit einen Ausdruck zu beschreiben wäre: "alles was einen Wert besitzt". Beispiele für Ausdrücke wären die Zahl **5**, die Addition **5+37** oder der Aufruf einer Funktion wie **Rnd** um eine Zufallszahl zu erhalten. Der Wert eines Ausdrucks kann Variablen zugewiesen werden oder an Prozeduren bzw. Funktionen übergeben werden.

```
Dim i As Integer

i = 5
i = i + 37
Print "Die Antwort ist " & i
Print "Eine zufällige Zahl ist " & Rnd
```

Operatoren verknüpfen Ausdrücke zu einem neuen Ausdruck. So ist zum Beispiel das **+** Symbol in **5 + 37** ein Operator, der die beiden ganzzahligen Ausdrücke **5** und **37** zu einem neuen ganzzahligen Ausdruck verknüpft.

2.6.1 Operatorreihenfolge

Die folgende Tabelle listet die verfügbaren Operatoren auf, beginnend mit dem am stärksten bindenden Operator. Ausdrücke, die mit einem stärker bindenden Operator verknüpft sind, werden vorrangig ausgeführt. Die Operatoren in der gleichen Zeile haben dieselbe Priorität. Operationen mit gleichrangiger Priorität werden in der Reihenfolge von links nach rechts ausgeführt.

Symbol	Bedeutung
^	Exponentiation
-	arithmetische Negation
* /	Multiplikation, Division
\	ganzzahlige Division
mod	modulo Division
+ -	Addition, Subtraktion

Symbol	Bedeutung
&	Zeichenkettenverbindung
= <> < > <= >=	Vergleich
Not	logische und binäre Negation
And	logisches und binäres Und
Or	logisches und binäres Oder
Xor	logisches und binäres Xor

Um in die Rangfolge der Operatoren zusätzlich einzugreifen, kann man Ausdrücke klammern. Die Klammer bindet stärker als jeder Operator. So setzt die Klammer im Ausdruck **(2+3)*4** die "Punkt vor Strich"-Regel außer Kraft und das Ergebnis ist **20**.

2.6.2 Arithmetische Operatoren

Arithmetische Operatoren sind die Exponentiation (**^**), arithmetische Negation (-), Addition (**+**), Subtraktion (-), Multiplikation (*****), Division(**/**), ganzzahlige Division (\) und modulo Division (**mod**). Sie verknüpfen numerische Werte (*Byte*, *Integer*, *Double*) und liefern diese zurück.

```
Dim i As Integer
Dim d As Double

i = 5 mod 3         ' das Ergebnis ist 2
Print i * i         ' gibt 4 aus
d = 1.0 / 2.0       ' das Ergebnis ist 0.5
Print - d ^ 2       ' gibt -0.25 aus
```

2.6.3 Logische Operatoren

Logische Operatoren sind die Negation (**Not**), Konjunktion (**And**), Disjunktion (**Or**) und die exklusive Disjunktion (**Xor**). Sie verwenden Werte des Typs *Boolean* und liefern einen *Boolean* Wert zurück. Die so genannten "Wahrheitstafeln" geben das Ergebnis einer logischen Verknüpfung an:

x	y	Not x	Not y	x And y	x Or y	x Xor y
True	True	False	False	True	True	True
True	False	False	True	False	True	False
False	True	True	False	False	True	False
False	False	True	True	False	False	True

```
Dim b As Boolean

b = True Xor False ' das Ergebnis ist True
```

2.6.4 Vergleichsoperatoren

Vergleichsoperatoren verbinden Werte desselben Datentyps miteinander und liefern einen *Boolean* Wert zurück. Falls im Zusammenhang sinnvoll, können alle Datentypen verwendet werden, wobei *Arrays* die gleiche Dimension haben müssen. Beim Vergleich von numerischen Werten (*Byte*, *Integer*, *Double*) dürfen diese auch verschiedenen Typs sein (s. "Ausdruckskonvertierung", S. 31).

Symbol	Bedeutung
=	gleich
<>	ungleich
<	kleiner
>	größer
<=	kleiner gleich
>=	größer gleich

2.6.5 Bitweise Operatoren

Bitweise Operatoren sind die Negation (**`Not`**), Und (**`And`**), Oder (**`Or`**) und die Exklusion (**`Xor`**). Sie nehmen Werte des Typs *Integer* (0, 1) und liefern einen *Integer* Wert zurück. Die so genannten "Bittafeln" geben das Ergebnis einer bitweisen Verknüpfung an.

x	y	Not x	Not y	x And y	x Or y	x Xor y
1	1	0	0	1	1	1
1	0	0	1	0	1	0
0	1	1	0	0	1	0
0	0	1	1	0	0	1

```
Dim i As Integer

i = &B01111111 And &B11110111 ' das Ergebnis ist &B01110111
i = &H7f And &Hf7             ' das Ergebnis ist &H77
```

2.6.6 Zeichenkettenoperatoren

Der Operator **&** verbindet Zeichenketten miteinander und liefert einen zusammenhängenden *String*-Wert zurück. Zeichenketten, die nicht zuvor einer Variablen zugewiesen wurden, müssen dabei in Anführungszeichen stehen.

```
Dim s As String

s = "Next"
s = s & "View"   ' das Ergebnis ist "NextView"
```

2.6.7 Ausdruckskonvertierung

NextView®4 Script verfügt über eine automatische Ausdruckskonvertierung von einfachen Datentypen. Das bedeutet, eine Zeichenkette, die nur aus Ziffern besteht, wird als Zahlenwert erkannt und umgekehrt, ohne dass dazu ein zusätzlicher Konvertierungsbefehl im Programm erforderlich ist.

Im folgendem Beispiel wird der Wert der Integervariablen **i** automatisch in eine Zeichenkette umgewandelt und dann die beiden Zeichenketten mit **&** verbunden:

```
Dim i As Integer

i = 13
Print "i ist " & i      ' ausgegeben wird: i ist 13
```

Umgekehrt werden nun zwei Zeichenketten in numerische Werte umgewandelt und dann miteinander multipliziert:

```
Print "14" * "1.5"      ' ausgegeben wird: 21
```

2.7 Unterprogramme

Funktionen und Prozeduren verringern den Programmieraufwand und tragen wesentlich zur Verbesserung der Übersichtlichkeit und Strukturierung bei. So kann ein mehrfach verwendeter Vorgang einfach durch einen Funktions- bzw. Prozeduraufruf ablaufen und muss nicht jedes Mal als Programmcode eingegeben werden.

NextView®4 Script stellt vordefinierte Unterprogramme wie zum Beispiel die schon häufig in den Beispielen verwendete Prozedur **`Print`** oder die Funktion **`Rnd`** zur Verfügung, die ohne vorherige Deklarationen direkt im Programm integriert werden können (s. "Standard Funktionen und Prozeduren", S. 150).

Man unterscheidet bei Unterprogrammen zwischen Prozeduren (**`Sub`**), die Befehlsfolgen ausführen, oder Funktionen (**`Function`**), die einen Rückgabewert liefern.

```
Print "Asterix"     ' Beispiel einer Prozedur
                    ' Print hat keinen Rückgabewert

Dim i, j As Integer ' Beispiel einer Funktion
i = Rnd             ' Rnd liefert eine Zufallszahl als
                    ' Rückgabewert
```

2.7.1 Argumente

Informationen an ein Unterprogramm werden mit Hilfe von Argumenten übergeben. An die Prozedur **Print** gibt man beispielsweise eine Zeichenkette, die **Print** dann an der Standardausgabe ausgibt. Unterprogramme können auch optionale Argumente besitzen, d. h. Argumente können, müssen nicht angegeben werden. So kann man der Prozedur **UBound** den Indexwert übergeben. Wird keine Indexposition angegeben, wird immer der erste Wert im Array verwendet.

```
Print "Asterix"          ' ausgegeben wird: Asterix

Dim i(5,15) As Integer

Print UBound (i)         ' ausgegeben wird: 5
Print UBound (i,1)       ' ausgegeben wird: 5
Print UBound (i,2)       ' ausgegeben wird: 15
```

2.7.2 Prozeduren

Eine Prozedur wird folgendermaßen definiert:

```
Sub Prozedurname ([arg1 As Datentyp1, arg2 As Datentyp2, ...])
  [Anweisungen]
End Sub
```

Benötigt die Prozedur keine Argumente, steht hinter dem Prozedurnamen bei der Definition eine leere (runde) Klammer.

Beim Aufruf der Prozedur im Skript gibt man den Namen der Prozedur und, falls erforderlich, anschließend die zu verwendenden Argumente an. Mehrere Argumente werden voneinander durch Kommata getrennt.

```
Prozedurname
```

bzw.

```
Prozedurname [Ausdruck1, Ausdruck2, ...]
```

Hierzu einige Beispiele:

```
Sub ShowName ()
  Print "Asterix"
End Sub

Sub Mult (a As Double, b As Double)
  Print a * b
End Sub
...
Sub Test ()
  ShowName           ' ausgegeben wird: Asterix
  Mult 3, 1.5        ' ausgegeben wird: 4.5
End Sub
```

Um aus einer Prozedur frühzeitig herausgehen zu können, verwendet man den Befehl **`Exit Sub`**.

```
Sub Div (a As Double, b As Double)
  If b = 0 Then Exit Sub Else Print a / b
End Sub
```

2.7.3 Funktionen

Eine Funktion wird folgendermaßen definiert:

```
Function Funktionsname ([arg1 As Datentyp1, _
                         arg2 As Datentyp2, ...]) As Datentyp
  [Anweisungen]
End Function
```

Die Anweisungen der Funktionsdefinition müssen immer das Ergebnis der Funktion, den Rückgabewert, dem Funktionsnamen zuweisen.

```
Funktionsname = Funktionsergebnis
```

Um eine Funktion im Skript aufzurufen, verwendet man den Funktionsnamen. Die zu übergebenden Argumente stehen in runden Klammern **( )** hinter dem Funktionsnamen. Der Rückgabewert einer Funktion, der durch Aufruf des

Funktionsnamens geliefert wird, wird entweder einer Variable zugewiesen oder als Ausdruck weiterverarbeitet.

```
Variable = Funktionsname
```
bzw.
```
Variable = Funktionsname (Ausdruck1, Ausdruck2, ...)
```

```
Function Pi () As Double  ' Funktionsdeklaration
  Pi = 3.1415926535       ' Rückgabewert zuweisen
End Function

Function Mult (a As Integer, b As Integer) As Integer
  Mult = a * b            ' Rückgabewert zuweisen
End Function
...
Sub Test ()
Dim i as Double
  i = Pi                  ' i erhält den Wert: 3.1415926535
  Print Mult(3, 2)        ' ausgegeben wird: 6
  Print 1 + Mult(3, i)    ' ausgegeben wird: 10.4247779605
End Sub
```

Um aus einer Funktion frühzeitig herausgehen zu können, verwendet man den Befehl **`Exit Function`**.

```
Function Div (a As Integer, b As Integer) As Integer
  Div = 1
  If b = 0 Then Exit Function Else Div = a / b
End Function
```

2.7.4 Optionale Argumente

Soll ein Unterprogramm optionale Argumente haben, so geschieht dies durch das Schlüsselwert **`Optional`** vor dem Argument und der Zuweisung eines Defaultwertes bei der Deklaration des Unterprogramms. Diese werden verwendet, wenn beim Aufruf der Funktion/Prozedur kein anderes Argument verwendet wurde. Alle Argumente nach dem ersten optionalen Argument sind ebenfalls optional.

```
Optional argx As Datentypx = Defaultwert
```

```
Sub Name (Optional s As String = "Asterix")
  Print s
End Sub

Function Mult (a As Integer, Optional _
               b As Integer = 1) As Integer
  Mult = a * b
End Function

Sub Test ()
  Name                ' ausgegeben wird: Asterix
  Name "Obelix"       ' ausgegeben wird: Obelix
  Print Mult(3)       ' ausgegeben wird: 3
  Print Mult(3, 2)    ' ausgegeben wird: 6
End Sub
```

2.7.5 ByVal und ByRef

Die Schlüsselwörter **ByVal** und **ByRef** bestimmen die Methode, wie an ein Unterprogramm ein Argument übergeben werden soll.

ByVal argx As Datentypx bzw.

ByRef argy As Datentypy

Wird die Standardmethode **ByVal** verwendet, so wird eine Kopie des Ausdrucks an das Unterprogramm übergeben. Ändert das Unterprogramm den Wert, so ändert es den Wert in der Kopie und das Original bleibt erhalten.

Bei der **ByRef** Methode hingegen wird bei einer Änderung des Wertes das Original verändert. Aus diesem Grund können bei der **ByRef** Methode nur Variablen an ein Unterprogramm übergeben werden, und beispielsweise keine Konstanten.

```
Sub Name1 (ByVal s As String)
  s = "Asterix"
End Sub

Sub Name2 (ByRef s As String)
  s = "Asterix"
End Sub

Sub Test ()
  Dim s As String

  s = "Obelix"
  Name1 "Obelix"       ' erlaubt
  Name1 s
  Print s              ' ausgegeben wird: Obelix

  s = "Obelix"
' Name2 "Obelix"         WÄRE NICHT ERLAUBT!
  Name2 s
  Print s              ' ausgegeben wird: Asterix
End Sub
```

2.8 Kontrollstrukturen

Neben einem möglichen Deklarationsteil besteht jedes **NextView®4 Script** aus einer Reihe von Anweisungen. Dies kann beispielsweise eine Zuweisung, ein Prozeduraufruf oder eine Kontrollstruktur wie eine Schleife oder eine bedingte Anweisung sein.

2.8.1 Bedingte Anweisungen

Bedingte Anweisungen erlauben nach Überprüfung eines Ausdrucks von Typ *Boolean*, die Ausführung von klar gekennzeichneten Teilen des Skripts, für die eine gestellte Bedingung wahr (**`TRUE`**) ist.

2.8.1.1 If ... Then ... Else

Man verwendet eine **`If ... Then`** Struktur um eine oder mehrere Anweisungen auszuführen, falls sie eine hinter **`If`** gestellte Bedingung erfüllen. Es ist möglich

die einzeilige Syntax oder die mehrzeilige Blocksyntax zu benutzen. Die einzeilige Form wird nur verwendet, wenn nur ein einziger Befehl ausgeführt werden soll.

- Einzeilige Syntax:
```
If Bedingung Then Anweisung
```
- Mehrzeilige Syntax:
```
If Bedingung1 Then
  [Anweisungen]

  ...

End If
```

Man beachte, dass man bei der mehrzeiligen Syntax **`End If`** schreibt, um das Ende der bedingt auszuführenden Anweisungen zu markieren.

```
' einzeilige Syntax
If value < 10 Then Print "Less then 10"

' mehrzeilige Syntax
If value < 10 Then
  Print "Less then 10 "
  value = value + 1        ' erhöhe value um eins
End If
```

Um weitere Bedingungen abzufragen, falls der **`if`** Ausdruck nicht zutrifft, verwendet man **`Else If`** für jede weitere Abfrage. Alternative Anweisungen, falls keine **`If`** oder **`Else If`** Bedingung gilt, gibt man mit dem Schlüsselwort **`Else`** an.

- Einzeilige Syntax:
```
If Bedingung Then Anweisung Else Anweisung
```
- Mehrzeilige Syntax:
```
If Bedingung1 Then
  [Anweisungen1]

[Else If Bedingung2 Then
  [Anweisungen2]]

  ...
[Else
  [AnweisungenN]]

End If
```

Das heißt, zunächst wird Bedingung1 getestet und ggf. die entsprechenden Anweisungen1 ausgeführt oder Bedingung2 getestet usw. Sind keine Bedingungen

erfüllt, werden die AnweisungenN des **`Else`** Blocks ausgeführt, falls angegeben.

```
' einzeilige Syntax
If value = 0 Then Print "Zero" Else Print "Non-zero"'
mehrzeilige Syntax
If value = 0 Then
  Print "Zero"
Else If value = 10 Then
  Print " 10 "
Else
  Print "Not Zero or 10"
End If
```

Die Ineinanderschachtelung mehrerer **`if`** Anweisungen ist möglich, kann jedoch auch die Übersichtlichkeit beeinträchtigen und die Fehleranfälligkeit erhöhen.

```
' mehrzeilige Syntax
If value > 0 Then
  If value > 0 und value <10 then
    Print "positiver Wert ist einstellig"
  Else
    Print "positiver Wert ist mehrstellig"
  End If
Else
  Print "ungültiger Wert"
End If
```

2.8.1.2 Select Case

`Select Case` bietet sich an, wenn mehrere Ausdrücke verglichen und unterschiedliche Anweisungen ausgeführt werden sollen. Hier ist **`Select Case`** im Vergleich zur **`If ... Then ... Else`** Anweisung oft übersichtlicher.

```
Select Case Ausdruck
Case Ausdrucksliste1
  [Anweisungen1]
[Case Ausdrucksliste2
  [Anweisungen2]]
...
[Case Else
  [AnweisungenN]]
End Select
```

Das **If ... Then ... Else** Beispiel könnte man also auch so schreiben:

```
Select Case value
Case 0, 1
  Print "Zero oder 1"
Case 10
  Print "10"
Case Else
  Print "Non-zero"
End Select
```

Mit dem Schlüsselwort **Case** wird überprüft, ob der hier angegebene Wert übereinstimmt mit dem unter **Select Case** angegebenen Ausdruck.

2.8.2 Schleifen

Man verwendet Schleifenstrukturen um Anweisungen und Programmteile zu wiederholen.

2.8.2.1 Do ... Loop

Die **Do ... Loop** Schleife wiederholt die eingeschlossenen Anweisungen solange oder bis ein Ausdruck erfüllt ist (**While** bzw. **Until**). Die Bedingung steht dabei entweder vor (nach dem Schlüsselwort **Do**) oder nach (nach dem Schlüsselwort **Loop**) den zu wiederholenden Programmbefehlen, woraus sich feine aber ggf. folgenschwere Unterschiede bei der Ausführung des Programms ergeben.

Vor der Schleife:

- mit **While**:

```
Do While Ausdruck
  [Anweisungen]

Loop
```

- mit **Until**:

```
Do Until Ausdruck
  [Anweisungen]

Loop
```

Nach der Schleife:

- mit **`While`**:

```
Do
  [Anweisungen]

Loop While Ausdruck
```

- mit **`Until`**:

```
Do
  [Anweisungen]

Loop Until Ausdruck
```

Steht die Bedingung am Anfang, wird zuerst überprüft, ob die Bedingung erfüllt ist. Trifft sie bereits das erste Mal nicht zu, werden die in der Schleife stehenden Anweisungen übersprungen. Steht die Bedingung am Ende der Schleife, wird diese immer mindestens einmal ausgeführt. Um eine Endlosschleife zu vermeiden, sollte bei der Formulierung des Ausdrucks darauf geachtet werden, dass er irgendwann nicht mehr zutrifft (**`While`**) bzw. er irgendwann eintritt (**`Until`**).

Das folgende Beispiel zeigt, wie wichtig es ist auf die Auswahl der passenden Schreibweise zu achten.

```
Do
  j = 1 / i
  Print j
  i = i - 1
Loop While i <> 0
```

Ist die Variable **`i`** bei Erreichen der **`Do ... Loop`** Schleife **`0`**, würde hier durch **`0`** geteilt werden. Bei negativem **`i`** würde man außerdem in eine Endlosschleife kommen. Richtig wäre die folgende Schleife:

```
Do While i > 0
  j = 1 / i
  Print j
  i = i - 1
Loop
```

2.8.2.2 While ... Wend

Die **`While ... Wend`** Anweisung entspricht der zuvor aufgeführten **`Do ... Loop`** Variante.

- **`While ... Wend`**:

```
While Ausdruck
  [Anweisungen]

Wend
```

2.8.2.3 For ... Next

Ist bekannt, wie oft eine Schleife durchlaufen werden soll, bietet sich die **`For ... Next`** Anweisung an. Damit lassen sich beispielsweise Endlosschleifen vermeiden. Diese benutzt eine Integervariable, **`Zaehler`** genannt, die während der Wiederholung automatisch um das angegebene Inkrement erhöht oder vermindert (bei negativem Inkrement) wird. Ist kein Inkrement angegeben, wird der Startwert bei jedem Durchlauf kontinuierlich um eins erhöht, bis der Endwert erreicht ist.

```
For Zaehler = Start To Ende [Step Inkrement]
[Ausdrücke]
Next
```

Dabei ist **`Zaehler`** eine globale oder lokale Integervariable, **`Start`**, **`Ende`** und **`Inkrement`** Integerkonstanten.

```
Dim i As Integer

For i = 0 To 42
  Print i                    ' Ausgabe: 0 1 2 3 usw. bis 42
Next

For i = 100 To 0 Step -2
  Print i                    ' Ausgabe: 100 98 96 usw. bis 0
Next
```

2.8.2.4 Repeat, Exit

Um aus einer Schleife herauszuspringen oder eine Schleife schon zu einem früheren Zeitpunkt wiederholen zu lassen, gibt es die Schlüsselwörter **`Exit`** und **`Repeat`**.

- **`Do ... Loop`**:

```
Do [While | Until] Ausdruck
  [Repeat]

  [Exit Do]

Loop
```

- **`Do ... Loop`**:

```
Do
  [Repeat]

  [Exit Do]

Loop [While | Until] Ausdruck
```

- **`For ... Next`**:

```
For Zaehler = Start To Ende [Step Inkrement]
  [Exit For]

Next
```

Bei einer **`While ... Wend`** Schleife gibt es diese Möglichkeit nicht. Für die **`For ... Next`** Schleife steht nur die **`Exit For`** Anweisung zur Verfügung.

```
Dim i as Integer
Do while True
  i = InputBox ("Bereich 0..10, -1 beendet Schleife", _
                "Zahl eingeben", 0)
  If i = -1 Then Exit Do
  ...
  If i > 10 then Repeat
  ...
Loop
```

2.9 Klassen und Objekte

NextView®4 Script arbeitet Objekt orientiert. Objekte werden aus Klassen erzeugt. Der Datentyp Klasse ist eine Sammlung von Variablen, Prozeduren und Funktionen, die mit diesen Variablen arbeiten. Durch die Prozeduren und

Funktionen, die in der jeweiligen Klasse definiert sind, wird somit festgelegt, welche Operationen mit einem erzeugten Objekt dieser Klasse durchgeführt werden können. Der Zugriff auf Objekte erfolgt mit Hilfe von Referenzvariablen, die auf ein Objekt oder auf nichts (**`Nothing`**) weisen.

NextView®4 Script selbst stellt bereits vordefinierte Klassen zur Verfügung. Beispielsweise enthält die Klasse "NvSheet" (repräsentiert ein Blatt in **NextView®4**) Funktionen, die sich nur auf ein Blatt beziehen, wie die Funktion **`NvSheet.DisplayCount`**.

Ein Objekt der Klasse "NvSheet", auf das die Referenzvariable **`mySheet`** weist, ist ein real vorhandenes Blatt in **NextView®4**, in dem man z. B. mit dem Befehl **`mySheet.DisplayCount`** die Anzahl der Anzeigen auf dem Blatt erhält.

```
Dim mySheet as NvSheet
Set mySheet = NvCurrentProject.Sheet(1)
print mySheet.DisplayCount
```

2.9.1 Klassendeklaration

Neben den von **NextView®4 Script** vordefinierten Klassen lassen sich auch eigene Klassen definieren. Die Deklaration einer Klasse erfolgt mit dem Schlüsselwort **`Class`**.

```
Class MyClass
  ...
End Class
```

In dieser Klassendeklaration werden die Variablen, Prozeduren und Funktionen, die einem Objekt zur Verfügung stehen sollen, definiert.

2.9.1.1 Variablen

Die Deklaration von Klassenvariablen wird mit dem Schlüsselwort **`Dim`** wie die allgemeine Variablendeklaration durchgeführt, jedoch muss diese innerhalb der Klassendeklaration stehen.

```
Class MyClass
  Dim i As Integer
End Class
```

2.9.1.2 Prozeduren und Funktionen

Die in einer Klasse verfügbaren Prozeduren und Funktionen werden mit den Schlüsselwörtern **Sub** bzw. **Function** innerhalb der Klassendeklaration nach dem allgemein üblichen Regeln definiert.

```
Class MyClass
  Sub MySub ()
    ...
  End Sub
  Function MyFunction () As Integer
    ....
  End Function
End Class
```

2.9.1.3 Me

In einer Klasse gibt es stets eine Referenzvariable (s. "Referenzvariablen", S. 46) auf die eigene Klasse. Diese muss nicht extra definiert werden und verbirgt sich hinter dem Schlüsselwort **Me**.

```
Class MyClass
  Dim i As Integer

  Sub MyOtherSub ()
    Me.i = 42
  End Sub

  Sub MySub ()
    Me.MyOtherSub
  End Sub
End Class
```

Bei Verwendung der **Me** Referenzvariablen hat man die Möglichkeit in einer Klasse auf Klassenvariablen, -prozeduren und -funktionen zuzugreifen. Diese Vorgehensweise sieht man an folgendem Beispiel.

```
Class MyClass
  Dim i As Integer                ' i als Klassenvariable
  ...
  Sub MySub (i As Integer)        ' i als lokale Variable
    Print "Your answer is " & i    ' verwendet lokales i
    Print "My answer is " & Me.i   ' verwendet
  End Sub                         ' Klassenvariable i
End Class
```

2.9.2 Arbeiten mit Objekten

2.9.2.1 Referenzvariablen

Eine Referenzvariable zeigt entweder auf ein neu erzeugtes Objekt (s. "Anlegen von Objekten", S. 46), ein von einer Funktion zurückgegebenes Objekt (s. "Objekte als Rückgabewert von Funktionen", S. 47) oder auf kein Objekt (**Nothing**, s. S. 48). Diese wird mit dem Schlüsselwort **Dim** deklariert.

```
Dim Referenzvariable As Klasse
```

Nach der Deklaration steht der Wert einer Referenzvariablen immer auf **Nothing**. Um einer Referenzvariablen ein Objekt zuzuweisen, verwendet man den Befehl **Set**.

```
Dim cha As NvChannel       ' AnalogIn ist eine in NextView
Set cha = AnalogIn (5)     ' Script vordefinierte Funktion
```

2.9.2.2 Anlegen von Objekten

Das Anlegen von Objekten aus selbst definierten Klassen geschieht mit dem Schlüsselwort **New**.

```
Class MyClass
  Dim i As Integer

  Sub MySub ()
    Print "The answer is " & i & "."
  End Sub
End Class
...
Sub Test ()
  Dim my As MyClass
  Set my = New MyClass      ' Objekt wird angelegt und Refe-
                            ' renzvariable darauf gesetzt.
  my.i = 42
  my.MySub                  ' Ausgabe: The answer is 42.
End Sub
```

Für Klassen, die von **NextView®4 Script** bereits vordefiniert sind, wie zum Beispiel die Klasse "NvDisplay", stellt **NextView®4 Script** Funktionen zum Anlegen eines neuen Objekts zur Verfügung. **`New`** darf in diesem Fall nicht verwendet werden.

```
Dim g As NvDisplay
Set g = NvCurrentProject.Sheet(1).Display(1)
' holt das erste Display des ersten Sheets im Projekt
' und setzt eine Referenzvariable darauf
```

2.9.2.3 Objekte als Rückgabewert von Funktionen

Ein Objekt kann Rückgabewert einer Funktion sein. Zum Beispiel liefert die vordefinierte Funktion **`NvProject.FindSheet (name As String)`** der Klasse "NvProject" das **NextView®4** Blatt mit dem Namen **`name`** im Projekt zurück, wobei **`NvCurrentProject`** das aktuelle Projekt als eine Referenz auf ein "NvProject" zurückgibt.

```
Dim s As NvSheet
Set s = NvCurrentProject.FindSheet ("FFT")
' Finde Blatt mit dem Namen FFT und setze
' Referenzvariable darauf
```

2.9.2.4 Nothing

Zeigt eine Referenzvariable auf kein Objekt, steht sie auf **`Nothing`**. Ebenso kann man eine Referenzvariable mit diesem Schlüsselwort auf kein Objekt setzen oder eine Referenzvariable mit **`Is Nothing`** abfragen, ob sie auf ein gültiges Objekt zeigt.

```
Dim s As NvSheet

Set s = Nothing
If s Is Nothing Then Print "Nothing!"
```

2.9.2.5 Löschen von Objekten

Objekte werden in **NextView®4 Script** automatisch gelöscht, sobald auf ein Objekt keine Referenzvariable zeigt oder eine sonstige Referenz existiert.

```
Dim my As MyClass
Set my = New MyClass
Set my = Nothing
```

Der Code im obigen Beispiel würde ein Objekt der Klasse "MyClass**"** anlegen, das in der nächsten Zeile wieder freigegeben wird, da die einzige Referenz auf das Objekt gelöscht wird.

Setzt man auf Objekt **`my1`** jedoch eine weitere Referenzvariable, so wird **`my1`** durch die **`Nothing`** Zuweisung nicht gelöscht, da die Referenz von **`my2`** auf **`my1`** weiterhin besteht.

```
Dim my1, my2 As MyClass
Set my1 = New MyClass
Set my2 = my1
Set my1 = Nothing
```

2.9.3 Die Programmierung mit Klassen

Um auf die von einer Klasse dem Objekt zur Verfügung gestellten Eigenschaften, Prozeduren und Funktionen zugreifen zu können, wird immer der Objektname, gefolgt von einem Punkt vorangestellt.

```
Print NvCurrentProject.SheetCount

Print NvCurrentProject.Sheet(1).Name
```

Zeigt eine Referenzvariable auf kein Objekt, also auf **`Nothing`**, erzeugt **NextView®4 Script** zur Laufzeit einen so genannten Laufzeitfehler (Runtime Error 13, Nothing Referenced). Dieser wird in der Statuszeile von **NextView®4 Script** angezeigt.

Überprüfen Sie in diesem Fall die Gültigkeit der Referenzvariablen. Referenzvariablen auf neu angelegte Objekte sind immer gültig. Sollten hier Fehler auftreten, erzeugt dies schon während dem Anlegen des Objekts einen Laufzeitfehler.

Objekte, die von Funktionen zurückgegeben werden, können jedoch auch **`Nothing`** sein. Beispielsweise liefert die Funktion **`NvProject.FindSheet`**, falls es das gesuchte Blatt nicht gibt, **`Nothing`** zurück.

```
Dim s As NvSheet

Set s = NvCurrentProject.FindSheet ("something funny")

If s Is Nothing Then
  Print "Sheet not found"
  Exit Sub
End If
```

Ein spezieller Fall tritt auf, wenn eine Klasse von einer "Parent"-Klasse abgeleitet ist. So ist zum Beispiel die Klasse "NvGraphDisplay", als auch die Klasse "NvButton" und die Klasse "NvSlider" spezielle Klassen der Klasse "NvDisplay". Um festzustellen, dass ein zurück geliefertes Anzeigeobjekt eine spezielles Objekt ist, gibt es den "?=" Zuweisungsoperator. Ist das Objekt nicht vom Typ der Referenzvariablenklasse, wird **`Nothing`** zugewiesen.

```
Dim g as NvGraphDisplay
set g ?= NvCurrentProject.Sheet(1).Display(1)
' versucht die erste Anzeige im ersten Blatt des Projekts
' der NvGraphDisplay Referenzvariablen zuzuweisen
' Ist die Anzeige kein NvGraphDisplay wird Nothing
' der Referenzvariablen zugewiesen
if g Is Nothing then print "kein NvGraphDisplay"
```

Würde man hier den normalen "=" Zuweisungsoperator verwenden, kommt es schon bei der Skript-Kompilierung zu einem Runtime Error.

2.10 Gültigkeitsbereiche

Verwendet man gleiche Namen für globale Variablen oder lokal (s. "Globale und lokale Variablen", S. 22) in Prozeduren, Funktionen oder Klassen, ergibt sich das Problem der Gültigkeitsbereiche. Die folgende Tabelle listet in absteigender Priorität die Rangfolge von Variablen bzw. Prozeduren oder Funktionen auf.

Variablen
Rückgabewert einer Funktion
lokale Variablen, Prozedur bzw. Funktionsargumente
Klassenvariablen
globale Variablen

Prozeduren / Funktionen
Klassenprozeduren und –funktionen
globale Prozeduren und Funktionen

Wie man sieht, haben beispielsweise lokale Variablen innerhalb von Funktionen, Prozeduren oder Klassen Vorrang vor globalen Variablen, das heißt, bei gleichen Variablennamen ist die lokale Variable gültig.

```
Dim i As Integer          ' i als globale Variable (auf 0)

Sub Test ()
  Dim i As Integer        ' i als lokale Variable
  i = 42
  Print "Ergebnis: " & i  ' ausgegeben wird: Ergebnis: 42
End Sub
```

Wird innerhalb einer Prozedur- bzw. Funktionsdeklaration eine andere Prozedur / Funktion aufgerufen, gelten die lokalen Variablen der definierten Prozedur / Funktion nicht für die aufgerufene. Für diese gelten deren eigenen lokalen Variablen und ferner die globalen Variablen, wie im folgenden Beispiel gezeigt wird.

```
Dim i As Integer         ' i als globale Variable (auf 0)

Sub Answer ()
  Print "Ergebnis: " & i ' verwendet die globale Variable i
End Sub
Sub Test ()
  Dim i As Integer       ' i als lokale Variable
  i = 42
  Answer                 ' ausgegeben wird: Ergebnis: 0
End Class
```

Eine Klassenprozedur /-funktion hat Priorität vor globalen Prozeduren / Funktionen. Diese veranschaulicht das nächste Beispiel:

```
Sub MyOtherSub ()         ' globale Prozedur
  Print "Hello World!"
End Sub

Class MyClass
  Sub MyOtherSub ()       ' Klassenprozedur
    Print "Ergebnis: 42"
  End Sub
  Sub MySub ()
    MyOtherSub            ' ausgegeben wird: Ergebnis: 42
  End Sub
End Class
```

3 NV4 Script Klassen/Routinen

3.1 NextView® Funktionen und Prozeduren

3.1.1 NV4 Funktionen und Prozeduren: Überblick

Funktion	Beschreibung
`NvCurrentProject`	gibt das aktuelle Projekt zurück
`NvStartScan`	startet einen Scan
`NvStopScan`	stoppt einen Scan
`NvScanState`	gibt den aktuellen Zustand des Scans zurück
`NvSetTimerInterval`	setzt das Aufrufintervall für das `OnTimer` Event
`NvExitProgram`	beendet **NextView®4**
`NvAnalogIn`	stellt die Schnittstelle zu einem Analogeingang zur Verfügung
`NvAnalogOut`	stellt die Schnittstelle zu einem Analogausgang zur Verfügung
`NvDigital`	stellt die Schnittstelle zu einem Digitalkanal zur Verfügung
`NvDigitalLine`	stellt die Schnittstelle zu einer Digitalleitung zur Verfügung
`NvCounter`	stellt die Schnittstelle zu einem Zählerkanal zur Verfügung
`NvFormula`	stellt die Schnittstelle zu einem Formelkanal zur Verfügung

3.1.2 NvCurrentProject

```
Function NvCurrentProject () As NvProject auf highfff
```

Gibt das aktuelle Projekt der Klasse "NvProject" zurück.

```
' gibt die Anzahl der Sheets im Projekt aus
Print NvCurrentProject.SheetCount
```

Eine Auflistung aller verfügbaren NextView® Funktionen und Prozeduren befindet sich im Kapitel "NV4 Funktionen und Prozeduren: Überblick", S. 52.

3.1.3 NvStartScan

```
Sub NvStartScan (Optional checkExisting As Boolean = True)
```

Startet einen Scan. Dies entspricht dem Befehl "Messung starten" (Menüpunkt "Gerät") in **NextView®4**. Ist **`checkExisting`** auf "False", werden eventuell vorhandene gleichnamige Messdateien ohne Nachfrage überschrieben.

```
' startet einen Scan mit Nachfrage
NvStartScan

' startet einen Scan ohne Nachfrage
NvStartScan False
```

Eine Auflistung aller verfügbaren NextView® Funktionen und Prozeduren befindet sich im Kapitel "NV4 Funktionen und Prozeduren: Überblick", S. 52.

3.1.4 NvStopScan

```
Sub NvStopScan ()
```

Stoppt einen Scan. Dies entspricht dem Befehl "Messung stoppen" (Menüpunkt "Gerät") in **NextView®4**.

Eine Auflistung aller verfügbaren NextView® Funktionen und Prozeduren befindet sich im Kapitel "NV4 Funktionen und Prozeduren: Überblick", S. 52.

3.1.5 NvScanState

```
Function NvScanState () as Integer
```

Gibt den Scanstatus von **NextView®4** zurück. Folgende vordefinierte Konstanten werden zurückgeliefert:

Scanstatus	Beschreibung
scanStatePreparing	Datenspeicherung wird vorbereitet
scanStateBeforeTrigger	Datenspeicherung läuft, Triggerkondition noch nicht erfüllt
scanStateAfterTrigger	Datenspeicherung läuft, Triggerkondition wurde erfüllt
scanStateBusy	Scansystem ist beschäftigt nach der letzten Datenspeicherung
scanStateRunning	Livedaten werden gemessen
scanStateReady	Scansystem ist bereit
scanStateUnknown	Scanstatus unbekannt

Beispiel: Ausgabe des Scanstatus in einem Textfeld mit Namen "Text Scanstatus":

```
...
Dim T as NVTextField
Set T ?= NvCurrentProject.FindDisplay("Text Scanstatus")

if Not(T is Nothing) then
  Select case NvScanState
    case scanStateBeforeTrigger
      T.Title = "Trigger...?"     ' Warten auf den Trigger
    case scanStateAfterTrigger
      if T.Title = "Scanning..." then  ' Blink "Scanning..."
        T.Title = ""
      Else
        T.Title = "Scanning..."
      End IF
    case scanStatePreparing
      T.Title = "Busy..."
    case scanStateBusy
      T.Title = "Busy..."
    case scanStateRunning
      T.Title = "Live Daten..."
    case scanStateRunning
      T.Title = "Live Daten..."
    case scanStateReady
      T.Title = "Ready..."
    case else
      T.Title = "Status..."
  end Select
End IF
...
```

Eine Auflistung aller verfügbaren NextView® Funktionen und Prozeduren befindet sich im Kapitel "NV4 Funktionen und Prozeduren: Überblick", S. 52.

3.1.6 NvSetTimerInterval

```
Sub NvSetTimerInterval (ms As Integer)
```

Setzt das Zeitintervall in Millisekunden, in dem das Event **`OnTimer`** aufgerufen wird. Dieses Event kann nicht schneller als alle 15ms aufgerufen werden. Dabei kann es, bedingt durch das Betriebssystem Windows®, zu zeitlichen Verschiebungen kommen, die in der Regel im 15ms Bereich liegen.

Eine Auflistung aller verfügbaren NextView® Funktionen und Prozeduren befindet sich im Kapitel "NV4 Funktionen und Prozeduren: Überblick", S. 52.

3.1.7 NvExitProgram

```
Sub NvExitProgram (checkModified As Boolean)
```

Dieser Befehl beendet **NextView®4**. Ist **`checkModified`** auf **`True`**, führen eventuell nicht gespeicherte Änderungen an dem Projekt zu einer Abfrage, ob das Projekt gespeichert werden soll. Bei **`checkModified`** gleich **`False`** wird **NextView®4** ohne Speicherung etwaiger Änderungen sofort beendet.

Eine Auflistung aller verfügbaren NextView® Funktionen und Prozeduren befindet sich im Kapitel "NV4 Funktionen und Prozeduren: Überblick", S. 52.

3.1.8 NvAnalogIn

```
Function NvAnalogIn (index As Integer) As NvChannel
```

Gibt den analogen Eingangskanal der Nummer **index** zurück oder **Nothing**, falls dieser nicht vorhanden ist. Die Kanäle sind fortlaufend durchnummeriert beginnend bei **index**=1 mit dem ersten Analogeingang des zuerst installierten Geräts (s. "Klasse "NvChannel"", S. 59).

In NextView®4 Script stehen während einem laufenden Scan nur die im Scan definierten Messkanäle zur Verfügung.

Eine Auflistung aller verfügbaren NextView® Funktionen und Prozeduren befindet sich im Kapitel "NV4 Funktionen und Prozeduren: Überblick", S. 52.

3.1.9 NvAnalogOut

```
Function NvAnalogOut (index As Integer) As NvChannel
```

Gibt den analogen Ausgangskanal der Nummer **index** zurück oder **Nothing**, falls dieser nicht vorhanden ist. Die Kanäle sind fortlaufend durchnummeriert beginnend bei **index**=1 mit dem ersten Analogausgang des zuerst installierten Geräts (s. "Klasse "NvChannel"", S. 59).

Eine Auflistung aller verfügbaren NextView® Funktionen und Prozeduren befindet sich im Kapitel "NV4 Funktionen und Prozeduren: Überblick", S. 52.

3.1.10 NvDigital

```
Function NvDigital (index As Integer) As NvChannel
```

Gibt den Digitalkanal der Nummer **index** zurück oder **Nothing**, falls dieser nicht vorhanden ist. Die Kanäle sind fortlaufend durchnummeriert beginnend bei **index**=1 mit dem ersten Digitalkanal des zuerst installierten Geräts (s. "Klasse "NvChannel"", S. 59).

```
' gibt den aktuellen Wert des 1. Digitalkanals
' in der Nachrichtenanzeige aus

Print NvDigital(1).Value
```

In NextView®4 Script stehen während einem laufenden Scan nur die im Scan definierten Messkanäle zur Verfügung.

Eine Auflistung aller verfügbaren NextView® Funktionen und Prozeduren befindet sich im Kapitel "NV4 Funktionen und Prozeduren: Überblick", S. 52.

3.1.11 NvDigitalLine

```
Function NvDigitalLine (index As Integer) As NvChannel
```

Gibt die Digitalleitung der Nummer **index** zurück oder **Nothing**, falls diese nicht vorhanden ist. Die Kanäle sind fortlaufend durchnummeriert beginnend bei **index**=1 mit der ersten Digitalleitung des zuerst installierten Geräts (s. "Klasse "NvChannel"", S. 59).

```
' gibt den aktuellen Wert der 1. Digitalleitung
' in der Nachrichtenanzeige aus und setzt diese auf high

Print NvDigitalLine(1).Value
Nv.DigitalLine(1).Value = 1
```

In NextView®4 Script stehen während einem laufenden Scan nur die im Scan definierten Messkanäle zur Verfügung.

Eine Auflistung aller verfügbaren NextView® Funktionen und Prozeduren befindet sich im Kapitel "NV4 Funktionen und Prozeduren: Überblick", S. 52.

3.1.12 NvCounter

```
Function NvCounter (index As Integer) As NvChannel
```

Gibt den Zählerkanal der Nummer **index** zurück oder **Nothing**, falls dieser nicht vorhanden ist. Die Kanäle sind fortlaufend durchnummeriert beginnend bei **index**=1 mit dem ersten Zähler des zuerst installierten Geräts (s. "Klasse "NvChannel"", S. 59).

In NextView®4 Script stehen während einem laufenden Scan nur die im Scan definierten Messkanäle zur Verfügung.

Eine Auflistung aller verfügbaren NextView® Funktionen und Prozeduren befindet sich im Kapitel "NV4 Funktionen und Prozeduren: Überblick", S. 52.

3.1.13 NvFormula

```
Function NvFormula (index As Integer) As NvChannel
```

Gibt den Formelkanal der Nummer **index** zurück oder **Nothing**, falls diese nicht vorhanden ist. Die Kanäle sind fortlaufend durchnummeriert beginnend bei **index**=1 für den ersten Formelkanal (s. "Klasse "NvChannel"", S. 59).

In NextView®4 Script stehen während einem laufenden Scan nur die im Scan definierten Messkanäle zur Verfügung.

Eine Auflistung aller verfügbaren NextView® Funktionen und Prozeduren befindet sich im Kapitel "NV4 Funktionen und Prozeduren: Überblick", S. 52.

3.2 Klasse "NvChannel"

Die Klasse "NvChannel" beschreibt die Eigenschaften und Funktionen der vorhandenen Eingabe- und Ausgabekanäle in **NextView®4**. Ein Objekt der **NextView®4** Klasse "NvChannel" kann nicht über den **`New`** Befehl angelegt werden. Es wird stattdessen von **`NvSetTimerInterval`**, **`NvAnalogOut`**, **`NvDigital`**, **`NvDigitalLine`**, **`NvFormula`** zurückgegeben.

```
' holt den 1. analogen Eingangskanal als NvChannel Objekt

Dim sig As NvChannel
Set sig = NvAnalogIn(1)

If sig Is Nothing Then
  Print "Kanal nicht vorhanden"
Else
  Print "aktueller Wert: " & sig.Value & " " & sig.Unit
End If
```

3.2.1 NvChannel: Überblick

Elementfunktion	Beschreibung
`NvChannel.Name`	Name des Kanals
`NvChannel.Group`	Gruppe des Kanals
`NvChannel.Comment`	Kommentar des Kanals
`NvChannel.Unit`	Einheit des Kanals
`NvChannel.Value`	aktueller Wert des Kanals

3.2.2 NvChannel.Name

```
Function Name () As String
```

Liefert den Namen des Kanals.

```
' gibt den Namen des 1. analogen Eingangskanals
' in der Nachrichtenanzeige aus
Print NvAnalogIn(1).Name
```

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Klasse "NvChannel" befindet sich im Kapitel "NvChannel: Überblick", S. 59.

3.2.3 NvChannel.Group

```
Function Group () As String
```

Liefert die Gruppe des Kanals.

```
' gibt die Gruppe des 1. analogen Eingangskanals
' in der Nachrichtenanzeige aus
Print NvAnalogIn(1).Group
```

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Klasse "NvChannel" befindet sich im Kapitel "NvChannel: Überblick", S. 59.

3.2.4 NvChannel.Comment

```
Function Comment () As String
```

Liefert den Kommentar für den Kanal.

```
' gibt den Kommentar des 1. analogen Eingangskanals
' in der Nachrichtenanzeige aus
Print NvAnalogIn(1).Comment
```

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Klasse "NvChannel" befindet sich im Kapitel "NvChannel: Überblick", S. 59.

3.2.5 NvChannel.Unit

```
Function Unit () As String
```

Liefert die Einheit des Kanals.

```
' gibt die Einheit des 1. analogen Eingangskanals
' in der Nachrichtenanzeige aus
Print NvAnalogIn(1).Unit
```

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Klasse "NvChannel" befindet sich im Kapitel "NvChannel: Überblick", S. 59.

3.2.6 NvChannel.Value

```
' Wert lesen:
  Function Value () As Double
' Wert setzen:
  Value = newValue      'newValue As Double
```

Setzt den Ausgabewert eines Kanals oder gibt diesen zurück. Bei Eingabekanälen wird immer der letzte gemessene Livedatenwert zurückgegeben. Wie oft dieser Wert aktualisiert wird, hängt ab von den Scaneinstellungen der verwendeten Messhardware.

```
' setzt den Wert des 1. analogen Ausgangskanals
' und gibt diesen in der Nachrichtenanzeige aus

Dim sig As NvChannel

Set sig = NvAnalogOut(1)

sig.Value = -3.14
Print sig.Value
```

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Klasse "NvChannel" befindet sich im Kapitel "NvChannel: Überblick", S. 59.

3.3 Klasse "NvProject"

Ein Objekt der **NextView®4** Klasse "NvProject" kann nicht über den **`New`** Befehl angelegt werden. Es wird stattdessen von der **NextView®4** Funktion **`NvCurrentProject`** zurückgegeben.

```
' findet ein Blatt

Const SN As String = "Live Elemente"

Sub OnClick (State As Boolean)
  Dim p As NvProject
  Set p = NvCurrentProject
  If p.FindSheet(SN) Is Nothing Then
    Print "Sheet " & SN & " nicht gefunden"
    Exit Sub
  End If
End Sub
```

3.3.1 NvProject: Überblick

Elementfunktion	Beschreibung
`NvProject.SheetCount`	liefert die Anzahl der Blätter im Projekt
`NvProject.Sheet`	liefert ein Blatt aus dem Projekt
`NvProject.FindSheet`	findet ein Blatt im Projekt
`NvProject.FindDisplay`	findet eine Anzeige im Projekt
`NvProject.Name`	liefert den Namen des Projekts
`NvProject.SetPrintInfo`	Angaben für die Kopfzeile im Ausdruck machen
`NvProject.SetPrintOptions`	Druckoptionen einstellen
`NvProject.ActiveSheet`	liefert das aktive Blatt
`NvProject.ActiveDisplay`	liefert die aktive Anzeige

3.3.2 NvProject.SheetCount

```
Function SheetCount () As Integer
```

Liefert die Anzahl der Blätter im Projekt.

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Klasse "NvProject" befindet sich im Kapitel "NvProject: Überblick", S. 62.

3.3.3 NvProject.Sheet

```
Function Sheet (index As Integer) As NvSheet
```

Liefert das Blatt Nummer **index** oder **Nothing**, falls dieses nicht vorhanden ist (s. "Klasse "NvSheet"", S. 67).

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Klasse "NvProject" befindet sich im Kapitel "NvProject: Überblick", S. 62.

3.3.4 NvProject.FindSheet

```
Function FindSheet (name As String) As NvSheet
```

Liefert das Blatt mit dem Namen **name** oder **Nothing**, falls dieses nicht vorhanden ist (s. "Klasse "NvSheet"", S. 67).

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Klasse "NvProject" befindet sich im Kapitel "NvProject: Überblick", S. 62.

3.3.5 NvProject.FindDisplay

```
Function FindDisplay (name As String) As NvDisplay
```

Liefert die Anzeige mit dem Namen **`name`** oder **`Nothing`**, falls diese nicht vorhanden ist (s. "Klasse "NvDisplay"", S. 69).

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Klasse "NvProject" befindet sich im Kapitel "NvProject: Überblick", S. 62.

3.3.6 NvProject.Name

```
Function Name () as String
```

Liefert den vollständigen Namen des **NextView®4** Projekts zurück.

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Klasse "NvProject" befindet sich im Kapitel "NvProject: Überblick", S. 62.

3.3.7 NvProject.SetPrintInfo

```
Sub SetPrintInfo (header As String, comment As String,
  responsible as String, company As String)
```

Dieser Befehl definiert die folgenden Textfelder für die Druckausgabe: Kopfzeile ("Header"), Kommentar ("Comment"), Bearbeiter ("Responsable"), und Firmenname ("Company"). Das Kommentarfeld kann mehrzeilig sein, wobei einzelne Zeilen durch Zeilenvorschub (**`chr(10)`**) im übergebenen String getrennt werden.

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Klasse "NvProject" befindet sich im Kapitel "NvProject: Überblick", S. 62.

3.3.8 NvProject.SetPrintOptions

```
Sub SetPrintOptions (noFrame as Boolean, printHeading As
  Boolean, useColors as Boolean, usePatterns as Boolean,
  singleaxis as Boolean, legend as Boolean, pages as
  Integer)
```

Mit diesem Befehl werden die folgenden Layout Optionen für die Druckausgabe gesetzt:

Parameter	Beschreibung
noFrame	Seite wird ohne Rahmen gedruckt
printHeading	Titelleiste im Ausdruck enthalten
legend	Legende mit Informationen über die angezeigten Signale (z. B. Speicherort, Abtastzeit, Anzahl der Messwerte, Signalinformationen, ggf. Cursorwerte)
useColors	farbiger Ausdruck
usePatterns	Signallinien werden als Muster und nicht als durchgezogene Linie gedruckt (bessere Unterscheidung bei Schwarzweißdruck)
singleAxis	gemeinsame y-Achse für alle Signale verwenden (0..100%)
pages	Anzahl der Seiten, auf die der Signalausdruck verteilt wird

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Klasse "NvProject" befindet sich im Kapitel "NvProject: Überblick", S. 62.

3.3.9 NvProject.ActiveSheet

```
Function ActiveSheet () As NvSheet
```

Liefert das momentan aktive Blatt zurück oder **Nothing**, falls kein Blatt vorhanden ist (s. "Klasse "NvSheet"", S. 67).

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Klasse "NvProject" befindet sich im Kapitel "NvProject: Überblick", S. 62.

3.3.10 NvProject.ActiveDisplay

```
Function ActiveDisplay () As NvDisplay
```

Liefert die momentan aktive Anzeige zurück oder **`Nothing`**, falls keine Anzeige selektiert oder vorhanden ist (s. "Klasse "NvDisplay"", S. 69).

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Klasse "NvProject" befindet sich im Kapitel "NvProject: Überblick", S. 62.

3.4 Klasse "NvSheet"

Ein Objekt der **NextView®4** Klasse "NvSheet" kann nicht über den **`New`** Befehl angelegt werden. Es wird stattdessen von den Funktionen **`NvProject.Sheet`** und **`NvProject.FindSheet`** der Klasse "NvProject" erzeugt.

```
' findet ein Blatt und aktiviert es

Const SN As String = "Live Elemente"

Sub OnClick (State As Boolean)
  Dim s As NvSheet
  Set s = NvCurrentProject.FindSheet(SN)
  If s Is Nothing Then
    Print "Sheet " & SN & " nicht gefunden"
    Exit Sub
  End If
  s.Activate
End Sub
```

3.4.1 NvSheet: Überblick

Elementfunktion	Beschreibung
`NvSheet.Name`	Blattnamen setzen oder auslesen
`NvSheet.DisplayCount`	liefert die Anzahl der Anzeigen im Blatt
`NvSheet.Display`	liefert Anzeige aus dem Blatt
`NvSheet.Activate`	aktiviert dieses Blatt
`NvSheet.Print`	druckt das Blatt aus

3.4.2 NvSheet.Name

```
' Name lesen:
  Function Name () As String
' Name setzen:
  Name = newName      ' newName As String
```

Setzt den Namen des Blattes oder gibt diesen zurück.

```
' setzt den Namen des ersten Sheets
If NvCurrentProject.SheetCount < 1 Then Exit Sub
NvCurrentProject.Sheet(1).Name = "Bierschaumzerfall"
```

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Klasse "NvSheet" befindet sich im Kapitel "NvSheet: Überblick", S. 67.

3.4.3 NvSheet.DisplayCount

```
Function DisplayCount () As Integer
```

Liefert die Anzahl der Anzeigen auf dem Blatt.

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Klasse "NvSheet" befindet sich im Kapitel "NvSheet: Überblick", S. 67.

3.4.4 NvSheet.Display

```
Function Display (index As Integer) As NvDisplay
```

Liefert die Anzeige mit der Nummer **`index`** oder **`Nothing`**, falls diese nicht vorhanden ist (s. "Klasse "NvDisplay"", S. 69).

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Klasse "NvSheet" befindet sich im Kapitel "NvSheet: Überblick", S. 67.

3.4.5 NvSheet.Activate

```
Sub Activate ()
```

Aktiviert das Blatt, macht es also sichtbar.

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Klasse "NvSheet" befindet sich im Kapitel "NvSheet: Überblick", S. 67.

3.4.6 NvSheet.Print

```
' Blatt drucken:
  Sub Print (useDefaults as Boolean)
```

Druckt das angezeigte Blatt auf dem eingestellten Drucker aus. Wird die Variable **`useDefaults`** auf **`True`** gesetzt, werden die eingestellten Druckerparameter benutzt.

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Klasse "NvSheet" befindet sich im Kapitel "NvSheet: Überblick", S. 67.

3.5 Klasse "NvDisplay"

Ein Objekt der **NextView®4** Klasse "NvDisplay" kann nicht über den **`New`** Befehl angelegt werden. Es wird stattdessen von der Funktion **`NvSheet.Display`** der Klasse "NvSheet" erzeugt.

```
' gibt die Positionen der Displays im Blatt aus
Sub OnClick (State As Boolean)
  Dim d As NvDisplay
  Dim s As NvSheet

  If NvCurrentProject.SheetCount < 1 Then Exit Sub
  Set s = NvCurrentProject.Sheet (1)
  Print "Anzahl der Displays: " & s.DisplayCount
  For i = 1 To s.DisplayCount
    Set d = s.Display (i)
    Print "Display " & i & ": " & d.Left & ", " & d.Top & ", " _
                  & d.Width & ", " & d.Height
  Next
End Sub
```

3.5.1 NvDisplay: Überblick

Elementfunktion	Beschreibung
`NvDisplay.Name`	Anzeigenname setzen oder auslesen
`NvDisplay.Sheet`	zugehöriges Blatt ausgeben
`NvDisplay.Left`	linke Position der Anzeige setzen oder auslesen
`NvDisplay.Top`	obere Position der Anzeige setzen oder auslesen
`NvDisplay.Width`	Anzeigenbreite der Anzeige setzen oder auslesen
`NvDisplay.Height`	Anzeigenhöhe setzen oder auslesen
`NvDisplay.Bounds`	Anzeigengröße und Position setzen
`NvDisplay.Print`	druckt die Anzeige aus
`NvDisplay.SetFont`	Schriftparameter für die Anzeige setzen

3.5.2 NvDisplay.Name

```
' Name lesen:
  Function Name () As String
' Name setzen:
  Name = newName      ' newName As String
```

Setzt den Namen einer einzelnen Anzeige oder gibt diesen zurück.

```
' setzt den Namen der ersten Anzeige auf dem ersten Blatt

If NvCurrentProject.Sheet(1).DisplayCount < 1 Then Exit Sub
NvCurrentProject.Sheet(1).Display(1).Name = "Test Display"
```

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Klasse "NvDisplay" befindet sich im Kapitel "NvDisplay: Überblick", S. 70.

3.5.3 NvDisplay.Sheet

```
Sub Sheet () as NvSheet
```

Gibt das Blatt, auf dem sich das Display befindet, zurück (s. "Klasse "NvSheet"", S. 67).

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Klasse "NvDisplay" befindet sich im Kapitel "NvDisplay: Überblick", S. 70.

3.5.4 NvDisplay.Left

```
' Position links lesen:
  Function Left () As Integer
' Position links setzen:
  Left = newLeft      ' newLeft As Integer
```

Setzt die Position des Displays links oder gibt diese zurück.

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Klasse "NvDisplay" befindet sich im Kapitel "NvDisplay: Überblick", S. 70.

3.5.5 NvDisplay.Top

```
' Position oben lesen:
  Function Top () As Integer
' Position Top setzen:
  Top = newTop      ' newTop As Integer
```

Setzt die Position des Displays oben oder gibt diese zurück.

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Klasse "NvDisplay" befindet sich im Kapitel "NvDisplay: Überblick", S. 70.

3.5.6 NvDisplay.Width

```
' Breite lesen:
  Function Width () As Integer
' Breite setzen:
  Width = newWidth      ' newWidth As Integer
```

Setzt die Breite des Displays oder gibt diese zurück.

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Klasse "NvDisplay" befindet sich im Kapitel "NvDisplay: Überblick", S. 70.

3.5.7 NvDisplay.Height

```
' Höhe lesen:
  Function Height () As Integer
' Höhe setzen:
  Height = newHeight      ' newHeight As Integer
```

Setzt die Höhe des Displays oder gibt diese zurück.

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Klasse "NvDisplay" befindet sich im Kapitel "NvDisplay: Überblick", S. 70.

3.5.8 NvDisplay.Bounds

```
Sub Bounds (left as Integer, top as Integer,
            right as Integer, bottom as Integer)
```

Definiert die Grösse der rechteckigen Anzeige durch Angabe der Eckpositionen links oben (**`left`**, **`top`**) und rechts unten (**`right`**, **`bottom`**) in Pixeln. Dabei handelt es sich bei der Position (0,0) um die Ecke links oben auf dem Blatt.

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Klasse "NvDisplay" befindet sich im Kapitel "NvDisplay: Überblick", S. 70.

3.5.9 NvDisplay.Print

```
' Display drucken:
  Sub Print (useDefaults as Boolean)
```

Druckt die Anzeige auf dem eingestellten Drucker aus. Wird die Variable **`useDefaults`** auf **`True`** gesetzt, werden die eingestellten Druckerparameter benutzt.

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Klasse "NvDisplay" befindet sich im Kapitel "NvDisplay: Überblick", S. 70.

3.5.10 NvDisplay.SetFont

```
Sub SetFont (name as String, size as Integer,
             bold as Integer, italic as Integer)
```

Setzt die Schriftparameter für die Anzeige.

Parameter	Beschreibung
`name`	Name der Schriftart
`size`	Schriftgröße in Punkt
`bold`	Schriftschnitt: 1 = fett; 0 = normal
`italic`	Schriftstärke: 1 = kursiv; 0 = normal

```
...
Dim d As NvDisplay
Set d = NvCurrentProject.Sheet(1).Display(1)
' set font to Arial, 16 point, bold, not italic.
d.SetFont "Arial", 16, 1, 0
...
```

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Klasse "NvDisplay" befindet sich im Kapitel "NvDisplay: Überblick", S. 70.

3.6 Klasse "NvButton"

Die Klasse "NvButton" beschreibt die Eigenschaften eines auf einem Blatt liegenden Schalters. Ein Objekt der **NextView®4** Klasse "NvButton" kann nicht über den **`New`** Befehl angelegt werden. Es wird stattdessen von der Routine **`NvProject.FindDisplay`** zurückgegeben.

```
' Beispiel für NvButton

Sub ToggleState ()  ' Schalter fortlaufend ein- und ausschalten
   Dim B as NvButton
   Set B ?= NvCurrentProject.FindDisplay("Button Test")
   if Not(B is Nothing) then
      B.State = Not B.State
   End If
End Sub
```

3.6.1 NvButton: Überblick

Elementfunktion	Beschreibung
`NvButton.Title`	Beschriftung des Schalters lesen/setzen
`NvButton.State`	Zustand des Schalters lesen/setzen
`NvButton.SetColor`	Farbeinstellungen des Schalters setzen
`NvButton.GetColor`	Farbeinstellungen des Schalters lesen
`NvButton.SetActiveColor`	Farbeinstellungen des Schalters im aktiven Zustand setzen
`NvButton.GetActiveColor`	Farbeinstellungen des Schalters im aktiven Zustand lesen
`NvButton.SetInactiveColor`	Farbeinstellungen des Schalters im inaktiven Zustand setzen
`NvButton.GetInactiveColor`	Farbeinstellungen des Schalters im inaktiven Zustand lesen
`NvButton.ActiveTitle`	Text des Schalters im aktiven Zustand lesen/setzen
`NvButton.InactiveTitle`	Text des Schalters im inaktiven Zustand lesen/setzen

3.6.2 NvButton.Title

```
' Beschriftung lesen:
  Function Title () As String
' Beschriftung setzen:
  Title = newTitle        ' newTitle As String
```

Mit diesem Befehl kann die Beschriftung des Schalters eingegeben bzw. ausgelesen werden.

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Klasse "NvButton" befindet sich im Kapitel "NvButton: Überblick", S. 74.

3.6.3 NvButton.State

```
' Zustand lesen:
  Function State () As String
' Zustand setzen:
  State = newState        ' newState As String
```

Ein Schalter besitzt zwei Zustände: eingeschaltet (**`True`**) oder ausgeschaltet (**`False`**). Dieser Befehl setzt bzw. liest den Zustand des Schalters.

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Klasse "NvButton" befindet sich im Kapitel "NvButton: Überblick", S. 74.

3.6.4 NvButton.SetColor

```
Sub SetColor (textColor as Integer, bgColor as Integer)
```

Setzt die Farbeinstellungen des Schalters. Der Übergabewert wird mit der Standardfunktion RGBColor berechnet.

Parameter	Beschreibung
textColor	Textfarbe (RGB-Wert)
bgColor	Hintergrundfarbe (RGB-Wert)

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Klasse "NvButton" befindet sich im Kapitel "NvButton: Überblick", S. 74.

3.6.5 NvButton.GetColor

```
Sub GetColor (Byref textColor as Integer,
  Byref bgColor as Integer)
```

Liefert die Farbeinstellungen des Schalters. Bei dem ausgegebenen Wert handelt es sich um einen RGB-Wert.

Parameter	Beschreibung
textColor	Textfarbe (RGB-Wert)
bgColor	Hintergrundfarbe (RGB-Wert)

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Klasse "NvButton" befindet sich im Kapitel "NvButton: Überblick", S. 74.

3.6.6 NvButton.SetActiveColor

```
Sub SetActiveColor (textColor as Integer,
  bgColor as Integer)
```

Setzt die Farbeinstellungen für den aktiven Zustand (Alarmzustand) des Schalters. Der Übergabewert wird mit der Standardfunktion RGBColor berechnet.

Parameter	Beschreibung
textColor	Textfarbe (RGB-Wert)
bgColor	Hintergrundfarbe (RGB-Wert)

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Klasse "NvButton" befindet sich im Kapitel "NvButton: Überblick", S. 74.

3.6.7 NvButton.GetActiveColor

```
Sub GetActiveColor (Byref textColor as Integer,
  Byref bgColor as Integer)
```

Liest die Farbeinstellungen für den aktiven Zustand (Alarmzustand) des Schalters. Bei dem ausgegebenen Wert handelt es sich um einen RGB-Wert.

Parameter	Beschreibung
textColor	Textfarbe (RGB-Wert)
bgColor	Hintergrundfarbe (RGB-Wert)

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Klasse "NvButton" befindet sich im Kapitel "NvButton: Überblick", S. 74.

3.6.8 NvButton.SetInactiveColor

```
Sub SetInactiveColor (textColor as Integer,
  bgColor as Integer)
```

Setzt die Farbeinstellungen für den inaktiven Zustand (Normalzustand) des Schalters. Der Übergabewert wird mit der Standardfunktion RGBColor berechnet.

Parameter	Beschreibung
textColor	Textfarbe (RGB-Wert)
bgColor	Hintergrundfarbe (RGB-Wert)

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Klasse "NvButton" befindet sich im Kapitel "NvButton: Überblick", S. 74.

3.6.9 NvButton.GetInactiveColor

```
Sub GetInactiveColor (Byref textColor as Integer,
  Byref bgColor as Integer)
```

Liest die Farbeinstellungen für den inaktiven Zustand (Normalzustand) des Schalters. Bei dem ausgegebenen Wert handelt es sich um einen RGB-Wert.

Parameter	Beschreibung
textColor	Textfarbe (RGB-Wert)
bgColor	Hintergrundfarbe (RGB-Wert)

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Klasse "NvButton" befindet sich im Kapitel "NvButton: Überblick", S. 74.

3.6.10 NvButton.ActiveTitle

```
' Beschriftung lesen:
  Function ActiveTitle () As String
' Beschriftung setzen:
  ActiveTitle = newTitle      ' newTitle As String
```

Setzt oder liefert den Text für den aktiven Zustand des Schalters.

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Klasse "NvButton" befindet sich im Kapitel "NvButton: Überblick", S. 74.

3.6.11 NvButton.InactiveTitle

```
' Beschriftung lesen:
  Function InactiveTitle () As String
' Beschriftung setzen:
  InactiveTitle = newTitle      ' newTitle As String
```

Setzt oder liefert den Text für den inaktiven Zustand des Schalters.

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Klasse "NvButton" befindet sich im Kapitel "NvButton: Überblick", S. 74.

3.7 Klasse "NvSlider"

Die Klasse "NvSlider" beschreibt die Eigenschaften eines auf einem Blatt liegenden Schiebereglers. Ein Objekt der **NextView®4** Klasse "NvSlider**"** kann nicht über den **`New`** Befehl angelegt werden. Es wird stattdessen von der Routine **`NvProject.FindDisplay`** zurückgegeben.

```
' Beispiel für NvSlider

Dim Ampl As Double
Sub OnSliderAmplMove (pos As Double)
  Dim S as NvSlider
  Set S ?= NvCurrentProject.FindDisplay("Slider Amplitude")
  if S is Nothing then Exit Sub
  Ampl = pos * (S.Maximum - S.Minimum) + S.Minimum
End Sub
```

3.7.1 NvSlider: Überblick

Elementfunktion	Beschreibung
`NvSlider.Title`	liefert oder setzt die Beschriftung des Schiebereglers
`NvSlider.Value`	Wert an der aktuellen Position des Schiebereglers
`NvSlider.Minimum`	unterster Wert der Schiebereglerskala
`NvSlider.Maximum`	oberster Wert der Schiebereglerskala

3.7.2 NvSlider.Title

```
' Beschriftung lesen:
  Function Title () As String
' Beschriftung setzen:
  Title = newTitle        ' newTitle As String
```

Mit diesem Befehl kann die Beschriftung des Schiebereglers eingegeben bzw. ausgelesen werden.

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Klasse "NvSlider" befindet sich im Kapitel "NvSlider: Überblick", S. 79.

3.7.3 NvSlider.Value

```
' Wert lesen:
  Function Value () As Double
' Wert setzen:
  Value = newValue      ' newValue As Double
```

Liefert oder setzt den Wert an der aktuellen Position des Schiebereglers.

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Klasse "NvSlider" befindet sich im Kapitel "NvSlider: Überblick", S. 79.

3.7.4 NvSlider.Minimum

```
' Minimum lesen:
  Function Minimum () As Double
' Minimum setzen:
  Minimum = newMinimum    ' newMinimum As Double
```

Liefert oder setzt den untersten Wert der Schiebereglerskala.

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Klasse "NvSlider" befindet sich im Kapitel "NvSlider: Überblick", S. 79.

3.7.5 NvSlider.Maximum

```
' Maximum lesen:
  Function Maximum () As Double
' Maximum setzen:
  Maximum = newMaximum    ' newMaximum As Double
```

Liefert den obersten Wert der Schiebereglerskala zurück oder setzt diesen auf einen neuen Wert.

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Klasse "NvSlider" befindet sich im Kapitel "NvSlider: Überblick", S. 79.

3.8 Klasse "NvTextField"

Die Klasse "NvTextField" beschreibt die Eigenschaften eines auf einem Blatt liegenden Textfeldes. Ein Objekt der **NextView®4** Klasse "NvTextField**"** kann nicht über den **`New`** Befehl angelegt werden. Es wird stattdessen von der Routine **`NvProject.FindDisplay`** zurückgegeben.

```
' Beispiel für NvTextField

Sub Test ()
  Dim T as NvTextField
  Set T ?= NvCurrentProject.FindDisplay("Text Test")
  if Not(T is Nothing) then
    Print T.Title
  End If
End Sub
```

3.8.1 NvTextField: Überblick

Elementfunktion	Beschreibung
`NvTextField.Title`	Beschriftung des Textfelds lesen/setzen
`NvTextField.SetColor`	Farbeinstellungen des Textfeldes setzen
`NvTextField.GetColor`	Farbeinstellungen des Textfeldes lesen
`NvTextField.SetActiveColor`	Farbeinstellungen des Textfeldes im aktiven Zustand setzen
`NvTextField.GetActiveColor`	Farbeinstellungen des Textfeldes im aktiven Zustand lesen
`NvTextField.SetInactiveColor`	Farbeinstellungen des Textfeldes im inaktiven Zustand setzen
`NvTextField.GetInactiveColor`	Farbeinstellungen des Textfeldes im inaktiven Zustand lesen
`NvTextField.ActiveTitle`	Beschriftung des Textfeldes im aktiven Zustand lesen/setzen
`NvTextField.InactiveTitle`	Beschriftung des Textfeldes im inaktiven Zustand lesen/setzen

3.8.2 NvTextField.Title

```
' Text lesen:
  Function Title () As String
' Text setzen:
  Title = newTitle        ' newTitle As String
```

Mit diesem Befehl kann der Inhalt des Textfeldes eingegeben bzw. ausgelesen werden.

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Klasse "NvTextField" befindet sich im Kapitel "NvTextField: Überblick", S. 82.

3.8.3 NvTextField.SetColor

```
Sub SetColor (textColor as Integer, bgColor as Integer)
```

Setzt die Farbeinstellungen des Textfeldes. Der Übergabewert wird mit der Standardfunktion **RGBColor** berechnet.

Parameter	Beschreibung
textColor	Textfarbe (RGB-Wert)
bgColor	Hintergrundfarbe (RGB-Wert)

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Klasse "NvTextField" befindet sich im Kapitel "NvTextField: Überblick", S. 82.

3.8.4 NvTextField.GetColor

```
Sub GetColor (Byref textColor as Integer,
  Byref bgColor as Integer)
```

Liefert die Farbeinstellungen des Textfeldes. Bei dem ausgegebenen Wert handelt es sich um einen RGB-Wert.

Parameter	Beschreibung
textColor	Textfarbe (RGB-Wert)
bgColor	Hintergrundfarbe (RGB-Wert)

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Klasse "NvTextField" befindet sich im Kapitel "NvTextField: Überblick", S. 82.

3.8.5 NvTextField.SetActiveColor

```
Sub SetActiveColor (textColor as Integer,
  bgColor as Integer)
```

Setzt die Farbeinstellungen für den aktiven Zustand (Alarmzustand) des Textfeldes. Der Übergabewert wird mit der Standardfunktion **RGBColor** berechnet.

Parameter	Beschreibung
textColor	Textfarbe (RGB-Wert)
bgColor	Hintergrundfarbe (RGB-Wert)

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Klasse "NvTextField" befindet sich im Kapitel "NvTextField: Überblick", S. 82.

3.8.6 NvTextField.GetActiveColor

```
Sub GetActiveColor (Byref textColor as Integer,
  Byref bgColor as Integer)
```

Liest die Farbeinstellungen für den aktiven Zustand (Alarmzustand) des Textfeldes. Bei dem ausgegebenen Wert handelt es sich um einen RGB-Wert.

Parameter	Beschreibung
textColor	Textfarbe (RGB-Wert)
bgColor	Hintergrundfarbe (RGB-Wert)

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Klasse "NvTextField" befindet sich im Kapitel "NvTextField: Überblick", S. 82.

3.8.7 NvTextField.SetInactiveColor

```
Sub SetInactiveColor (textColor as Integer,
  bgColor as Integer)
```

Setzt die Farbeinstellungen für den inaktiven Zustand (Normalzustand) des Textfeldes. Der Übergabewert wird mit der Standardfunktion **RGBColor** berechnet.

Parameter	Beschreibung
textColor	Textfarbe (RGB-Wert)
bgColor	Hintergrundfarbe (RGB-Wert)

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Klasse "NvTextField" befindet sich im Kapitel "NvTextField: Überblick", S. 82.

3.8.8 NvTextField.GetInactiveColor

```
Sub GetInactiveColor (Byref textColor as Integer,
  Byref bgColor as Integer)
```

Liest die Farbeinstellungen für den inaktiven Zustand (Normalzustand) des Textfeldes. Bei dem ausgegebenen Wert handelt es sich um einen RGB-Wert.

Parameter	Beschreibung
textColor	Textfarbe (RGB-Wert)
bgColor	Hintergrundfarbe (RGB-Wert)

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Klasse "NvTextField" befindet sich im Kapitel "NvTextField: Überblick", S. 82.

3.8.9 NvTextField.ActiveTitle

```
' Beschriftung lesen:
  Function ActiveTitle () As String
' Beschriftung setzen:
  ActiveTitle = newTitle      ' newTitle As String
```

Setzt oder liefert die Beschriftung für den aktiven Zustand des Textfeldes.

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Klasse "NvTextField" befindet sich im Kapitel "NvTextField: Überblick", S. 82.

3.8.10 NvTextField.InactiveTitle

```
' Beschriftung lesen:
  Function InactiveTitle () As String
' Beschriftung setzen:
  InactiveTitle = newTitle      ' newTitle As String
```

Setzt oder liefert die Beschriftung für den inaktiven Zustand des Textfeldes.

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Klasse "NvTextField" befindet sich im Kapitel "NvTextField: Überblick", S. 82.

3.9 Klasse "NvGraphDisplay"

Die **NextView®4** Klasse "NvGraphDisplay" ist von der Klasse "NvDisplay" abgeleitet. Ein Objekt der **NextView®4** Klasse "NvGraphDisplay**"** kann nicht über den **New** Befehl angelegt werden. Stattdessen wird von der Funktion **NvSheet.Display** der Klasse "NvSheet" ein Objekt der Klasse "NvDisplay" erzeugt. Handelt es sich dabei um ein "NvGraphDisplay", kann man dieses mit dem "?=" Operator zuweisen.

```
' gibt die Positionen der Displays im Blatt aus
Sub OnClick (State As Boolean)
  Dim graph As NvGraphDisplay
  Dim s As NvSheet

  If NvCurrentProject.SheetCount < 1 Then Exit Sub
  Set s = NvCurrentProject.Sheet (1)
  Print "Anzahl der Displays: " & s.DisplayCount

  For i = 1 To s.DisplayCount
    Set graph ?= s.Display (i)

    If graph Is Nothing Then
      Print "Display " & i & " ist kein GraphDisplay"
    Else
      Print "GraphDisplay " & i & ": " & graph.Left&  ", " _
            & graph.Top
      Print "  Anzahl der Signale: " & graph.SignalCount
    End If
  Next
End Sub
```

3.9.1 NvGraphDisplay: Überblick

Elementfunktion	Beschreibung
NvGraphDisplay.SignalCount	liefert die Anzahl der Signale in der Signalanzeige
NvGraphDisplay.Signal	liefert Signal aus der Signalanzeige
NvGraphDisplay.WhiteCursor	liefert den weißen (1.) Cursor der Signalanzeige
NvGraphDisplay.BlackCursor	liefert den schwarzen (2.) Cursor der Signalanzeige
NvGraphDisplay.xAxis	liefert die x-Achse einer Signalanzeige
NvGraphDisplay.yAxis	liefert die y-Achse eines Signals der Signalanzeige

3.9.2 NvGraphDisplay.SignalCount

```
Function SignalCount () As Integer
```

Liefert die Anzahl der Signale in der Signalanzeige.

Eine Auflistung aller verfügbaren Elementfunktionen der Klasse "NvGraphDisplay" befindet sich im Kapitel "NvGraphDisplay: Überblick", S. 87.

3.9.3 NvGraphDisplay.Signal

```
Function Signal (index As Integer) As NvSignal
```

Liefert das Signal Nummer **`index`** oder **`Nothing`**, falls dieses nicht vorhanden ist (s. "Klasse "NvSignal"", S. 118).

Signale in einer Anzeige sind schreibgeschützt. Bei Änderungen an einem Signal muss zuerst die zugehörige Messdatei mit den Befehlen der Klasse "NvOpenDataFile" mit Schreib-/Lesezugriff geöffnet werden, bevor die gewünschten Änderungen in der Datei durchgeführt werden können.

Eine Auflistung aller verfügbaren Elementfunktionen der Klasse "NvGraphDisplay" befindet sich im Kapitel "NvGraphDisplay: Überblick", S. 87.

3.9.4 NvGraphDisplay.WhiteCursor

```
Function WhiteCursor () As NvCursor
```

Liefert den weißen (1.) Cursor der Signalanzeige (s. "Klasse "NvCursor"", S. 101).

Eine Auflistung aller verfügbaren Elementfunktionen der Klasse "NvGraphDisplay" befindet sich im Kapitel "NvGraphDisplay: Überblick", S. 87.

3.9.5 NvGraphDisplay.BlackCursor

```
Function BlackCursor () As NvCursor
```

Liefert den schwarzen (2.) Cursor der Signalanzeige (s. "Klasse "NvCursor"", S. 101).

Eine Auflistung aller verfügbaren Elementfunktionen der Klasse "NvGraphDisplay" befindet sich im Kapitel "NvGraphDisplay: Überblick", S. 87.

3.9.6 NvGraphDisplay.xAxis

```
Function xAxis () As NvAxis
```

Liefert die x-Achse als Element der Klasse "NvAxis" der Signalanzeige (s. S. 103).

Eine Auflistung aller verfügbaren Elementfunktionen der Klasse "NvGraphDisplay" befindet sich im Kapitel "NvGraphDisplay: Überblick", S. 87.

3.9.7 NvGraphDisplay.yAxis

```
Function yAxis (index as Integer) As NvAxis
```

Liefert die y-Achse als Element der Klasse "NvAxis" des Signals mit der Nummer **`index`** in der Signalanzeige (s. S. 103).

Eine Auflistung aller verfügbaren Elementfunktionen der Klasse "NvGraphDisplay" befindet sich im Kapitel "NvGraphDisplay: Überblick", S. 87.

3.10 Klasse "NvLevelIndicator"

Die Klasse "NvLevelIndicator" beschreibt die Eigenschaften einer auf einem Blatt liegenden Füllstandsanzeige. Ein Objekt der **NextView®4** Klasse "NvLevelIndicator**"** kann nicht über den **`New`** Befehl angelegt werden. Es wird stattdessen von der Routine **`NvProject.FindDisplay`** zurückgegeben.

```
' Beispiel für NvLevelIndicator

Sub Test ()
  Dim L as NvLevelIndicator
  Set L ?= NvCurrentProject.FindDisplay("Level Test")
  if Not(L is Nothing) then
' nur sinnvoll, wenn kein Analogkanal mit der Anzeige verbunden ist
    L.Title = "Analog*2"
    L.Value = 2 * NvAnalogIn(1).Value
  End If
End Sub
```

3.10.1 NvLevelIndicator: Überblick

Elementfunktion	Beschreibung
`NvLevelIndicator.Title`	Kanalname der Füllstandsanzeige lesen / setzen
`NvLevelIndicator.SetColor`	Farbeinstellungen der Füllstandsanzeige setzen
`NvLevelIndicator.GetColor`	Farbeinstellungen der Füllstandsanzeige lesen
`NvLevelIndicator.SetActiveColor`	Farbe im aktiven Zustand setzen
`NvLevelIndicator.GetActiveColor`	Farbe im aktiven Zustand lesen
`NvLevelIndicator.SetInactiveColor`	Farbe im inaktiven Zustand setzen
`NvLevelIndicator.GetInactiveColor`	Farbe im inaktiven Zustand lesen
`NvLevelIndicator.Value`	Wert lesen/setzen
`NvLevelIndicator.Minimum`	kleinsten Wert lesen/setzen
`NvLevelIndicator.Maximum`	größten Wert lesen/setzen

3.10.2 NvLevelIndicator.Title

```
' Beschriftung lesen:
  Function Title () As String
' Beschriftung setzen:
  Title = newTitle        ' newTitle As String
```

Mit diesem Befehl kann der Kanalname der Füllstandsanzeige eingegeben bzw. ausgelesen werden.

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Klasse "NvLevelIndicator" befindet sich im Kapitel "NvLevelIndicator: Überblick", S. 90.

3.10.3 NvLevelIndicator.SetColor

```
Sub SetColor (fgColor as Integer, bgColor as Integer)
```

Setzt die Farbeinstellungen des inaktiven und aktiven Zustands einer Füllstandsanzeige auf den gleichen Farbwert. Der Übergabewert wird mit der Standardfunktion RGBColor berechnet.

Parameter	Beschreibung
fgColor	Vordergrundfarbe (Füllung, RGB-Wert)
bgColor	Hintergrundfarbe (RGB-Wert)

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Klasse "NvLevelIndicator" befindet sich im Kapitel "NvLevelIndicator: Überblick", S. 90.

3.10.4 NvLevelIndicator.GetColor

```
Sub GetColor (Byref fgColor as Integer,
  Byref bgColor as Integer)
```

Liefert die Farbeinstellungen der Füllstandsanzeige. Bei dem ausgegebenen Wert handelt es sich um einen RGB-Wert.

Parameter	Beschreibung
fgColor	Vordergrundfarbe (Füllung, RGB-Wert)
bgColor	Hintergrundfarbe (RGB-Wert)

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Klasse "NvLevelIndicator" befindet sich im Kapitel "NvLevelIndicator: Überblick", S. 90.

3.10.5 NvLevelIndicator.SetActiveColor

```
Sub SetActiveColor (fgColor as Integer, bgColor as Integer)
```

Setzt die Farbeinstellungen für den aktiven Zustand (Alarmzustand) der Füllstandsanzeige. Der Übergabewert wird mit der Standardfunktion RGBColor berechnet.

Parameter	Beschreibung
fgColor	Vordergrundfarbe (Füllung, RGB-Wert)
bgColor	Hintergrundfarbe (RGB-Wert)

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Klasse "NvLevelIndicator" befindet sich im Kapitel "NvLevelIndicator: Überblick", S. 90.

3.10.6 NvLevelIndicator.GetActiveColor

```
Sub GetActiveColor (Byref fgColor as Integer,
  Byref bgColor as Integer)
```

Liest die Farbeinstellungen für den aktiven Zustand (Alarmzustand) der Füllstandsanzeige. Bei dem ausgegebenen Wert handelt es sich um einen RGB-Wert.

Parameter	Beschreibung
fgColor	Vordergrundfarbe (Füllung, RGB-Wert)
bgColor	Hintergrundfarbe (RGB-Wert)

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Klasse "NvLevelIndicator" befindet sich im Kapitel "NvLevelIndicator: Überblick", S. 90.

3.10.7 NvLevelIndicator.SetInactiveColor

```
Sub SetInactiveColor (fgColor as Integer,
  bgColor as Integer)
```

Setzt die Farbeinstellungen für den inaktiven Zustand (Normalzustand) der Füllstandsanzeige. Der Übergabewert wird mit der Standardfunktion RGBColor berechnet.

Parameter	Beschreibung
fgColor	Vordergrundfarbe (Füllung, RGB-Wert)
bgColor	Hintergrundfarbe (RGB-Wert)

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Klasse "NvLevelIndicator" befindet sich im Kapitel "NvLevelIndicator: Überblick", S. 90.

3.10.8 NvLevelIndicator.GetInactiveColor

```
Sub GetInactiveColor (Byref fgColor as Integer,
  Byref bgColor as Integer)
```

Liest die Farbeinstellungen für den inaktiven Zustand (Normalzustand) der Füllstandsanzeige. Bei dem ausgegebenen Wert handelt es sich um einen RGB-Wert.

Parameter	Beschreibung
fgColor	Vordergrundfarbe (Füllung, RGB-Wert)
bgColor	Hintergrundfarbe (RGB-Wert)

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Klasse "NvLevelIndicator" befindet sich im Kapitel "NvLevelIndicator: Überblick", S. 90.

3.10.9 NvLevelIndicator.Value

```
' Wert lesen:
  Function Value () As Double
' Wert setzen:
  Value = newValue      ' newValue As Double
```

Liefert oder setzt den Wert an der aktuellen Position der Füllstandsanzeige.

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Klasse "NvLevelIndicator" befindet sich im Kapitel "NvLevelIndicator: Überblick", S. 90.

3.10.10 NvLevelIndicator.Minimum

```
' Minimum lesen:
  Function Minimum () As Double
' Minimum setzen:
  Minimum = newMinimum    ' newMinimum As Double
```

Liefert oder setzt den untersten Wert der Füllstandsanzeige.

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Klasse "NvLevelIndicator" befindet sich im Kapitel "NvLevelIndicator: Überblick", S. 90.

3.10.11 NvLevelIndicator.Maximum

```
' Maximum lesen:
  Function Maximum () As Double
' Maximum setzen:
  Maximum = newMaximum    ' newMaximum As Double
```

Liefert den obersten Wert der Füllstandsanzeige zurück oder setzt diesen auf einen neuen Wert.

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Klasse "NvLevelIndicator" befindet sich im Kapitel "NvLevelIndicator: Überblick", S. 90.

3.11 Klasse "NvDVM"

Die Klasse "NvDVM" beschreibt die Eigenschaften eines Digitalmultimeters. Ein Objekt der **NextView®4** Klasse "NvDVM" kann nicht über den **`New`** Befehl angelegt werden. Es wird stattdessen von der Routine **`NvProject.FindDisplay`** zurückgegeben.

```
' Beispiel für NvDVM

Sub Test ()
  Dim D as NvDVM
  Set D ?= NvCurrentProject.FindDisplay("DVM Test")
  if Not(D is Nothing) then
' Nur sinnvoll, wenn kein Analogkanal mit der Anzeige verbunden ist
    D.Title = "Analog*2"
    D.Value = 2 * NvAnalogIn(1).Value
  End If
End Sub
```

3.11.1 NvDVM: Überblick

Elementfunktion	Beschreibung
`NvDVM.Title`	Kanalname des Digitalmultimeters lesen/setzen
`NvDVM.SetColor`	Farbeinstellungen des Digitalmultimeters setzen
`NvDVM.GetColor`	Farbeinstellungen des Digitalmultimeters lesen
`NvDVM.SetActiveColor`	Farbe im aktiven Zustand setzen
`NvDVM.GetActiveColor`	Farbe im aktiven Zustand lesen
`NvDVM.SetInactiveColor`	Farbe im inaktiven Zustand setzen
`NvDVM.GetInactiveColor`	Farbe im inaktiven Zustand lesen
`NvDVM.Value`	Wert lesen/setzen

3.11.2 NvDVM.Title

```
' Beschriftung lesen:
  Function Title () As String
' Beschriftung setzen:
  Title = newTitle        ' newTitle As String
```

Mit diesem Befehl kann der Kanalname eingegeben bzw. ausgelesen werden, den das Digitalmultimeter anzeigt.

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Klasse "NvDVM" befindet sich im Kapitel "NvDVM: Überblick", S. 96.

3.11.3 NvDVM.SetColor

```
Sub SetColor (textColor as Integer, bgColor as Integer)
```

Setzt die Farbeinstellungen des inaktiven und aktiven Zustands eines Digitalmultimeters auf den gleichen Farbwert. Der Übergabewert wird mit der Standardfunktion **`RGBColor`** berechnet.

Parameter	Beschreibung
`textColor`	Textfarbe (RGB-Wert)
`bgColor`	Hintergrundfarbe (RGB-Wert)

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Klasse "NvDVM" befindet sich im Kapitel "NvDVM: Überblick", S. 96.

3.11.4 NvDVM.GetColor

```
Sub GetColor (Byref textColor as Integer,
  Byref bgColor as Integer)
```

Liefert die Farbeinstellungen des Digitalmultimeters. Bei dem ausgegebenen Wert handelt es sich um einen RGB-Wert.

Parameter	Beschreibung
textColor	Textfarbe (RGB-Wert)
bgColor	Hintergrundfarbe (RGB-Wert)

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Klasse "NvDVM" befindet sich im Kapitel "NvDVM: Überblick", S. 96.

3.11.5 NvDVM.SetActiveColor

```
Sub SetActiveColor (textColor as Integer,
  bgColor as Integer)
```

Setzt die Farbeinstellungen für den aktiven Zustand (Alarmzustand) des Digitalmultimeters. Der Übergabewert wird mit der Standardfunktion **RGBColor** berechnet.

Parameter	Beschreibung
textColor	Textfarbe (RGB-Wert)
bgColor	Hintergrundfarbe (RGB-Wert)

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Klasse "NvDVM" befindet sich im Kapitel "NvDVM: Überblick", S. 96.

3.11.6 NvDVM.GetActiveColor

```
Sub GetActiveColor (Byref textColor as Integer,
  Byref bgColor as Integer)
```

Liest die Farbeinstellungen für den aktiven Zustand (Alarmzustand) des Digitalmultimeters. Bei dem ausgegebenen Wert handelt es sich um einen RGB-Wert.

Parameter	Beschreibung
textColor	Textfarbe (RGB-Wert)
bgColor	Hintergrundfarbe (RGB-Wert)

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Klasse "NvDVM" befindet sich im Kapitel "NvDVM: Überblick", S. 96.

3.11.7 NvDVM.SetInactiveColor

```
Sub SetInactiveColor (textColor as Integer,
  bgColor as Integer)
```

Setzt die Farbeinstellungen für den inaktiven Zustand (Normalzustand) des Digitalmultimeters. Der Übergabewert wird mit der Standardfunktion **RGBColor** berechnet.

Parameter	Beschreibung
textColor	Textfarbe (RGB-Wert)
bgColor	Hintergrundfarbe (RGB-Wert)

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Klasse "NvDVM" befindet sich im Kapitel "NvDVM: Überblick", S. 96.

3.11.8 NvDVM.GetInactiveColor

```
Sub GetInactiveColor (Byref textColor as Integer,
  Byref bgColor as Integer)
```

Liest die Farbeinstellungen für den inaktiven Zustand (Normalzustand) des Digitalmultimeters. Bei dem ausgegebenen Wert handelt es sich um einen RGB-Wert.

Parameter	Beschreibung
textColor	Textfarbe (RGB-Wert)
bgColor	Hintergrundfarbe (RGB-Wert)

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Klasse "NvDVM" befindet sich im Kapitel "NvDVM: Überblick", S. 96.

3.11.9 NvDVM.Value

```
' Wert lesen:
  Function Value () As Double
' Wert setzen:
  Value = newValue      ' newValue As Double
```

Liefert oder setzt den aktuellen Wert des Digitalmultimeters.

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Klasse "NvDVM" befindet sich im Kapitel "NvDVM: Überblick", S. 96.

3.12 Klasse "NvCursor"

Ein Objekt der **NextView®4** Klasse "NvCursor" kann nicht über den **`New`** Befehl angelegt werden. Stattdessen wird von den Funktionen **`NvGraphDisplay.BlackCursor`** und **`NvGraphDisplay.WhiteCursor`** der Klasse "NvGraphDisplay" ein Objekt der **NextView®4** Klasse "NvCursor**"** erzeugt.

```
Dim g As NvGraphDisplay

' check ob Sheet und Display existieren
If NvCurrentProject.SheetCount < 1 Then Exit Sub
If NvCurrentProject.Sheet(1).DisplayCount < 1 Then Exit Sub

Set g ?= NvCurrentProject.Sheet(1).Display(1)
' ist das Display eine Signalanzeige?
If g Is Nothing Then
  Print "Display ist keine Signalanzeige"
  Exit Sub
End If

Dim c As NvCursor
Set c = g.WhiteCursor

If c.Enabled Then
  Dim ax As NvAxis
  Set ax = g.xAxis
  If ax.IsDateMode Then
    print "WhiteCursor: " & TimeStampStr(c.Value)
  Else
    Print "WhiteCursor: " & c.Value
  End If
Else
  Print "WhiteCursor: nicht aktiv"
End If
```

3.12.1 NvCursor: Überblick

Elementfunktion	Beschreibung
`NvCursor.Value`	Cursorwert setzen oder auslesen
`NvCursor.Enabled`	Abfrage oder Setzen des Cursorstatus'

3.12.2 NvCursor.Value

```
' Wert lesen:
  Function Value () As Double
' Wert setzen:
  Value = newValue      ' newValue As Double
```

Setzt den Wert des Cursors oder gibt diesen zurück. Der Wert ist abhängig vom Anzeigeformat der Signalanzeige. Im Datumsformat wird der Wert in Absolutzeit dargestellt, in Relativzeit in Sekunden.

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Klasse "NvCursor" befindet sich im Kapitel "NvCursor: Überblick", S. 101.

3.12.3 NvCursor.Enabled

```
' Cursor Status abfragen:
  Function Enabled () As Boolean
' Cursor Status setzen:
  Enabled = newState   ' newState As Boolean
```

Liefert oder setzt den Cursorstatus in der Anzeige. Der Cursor ist entweder eingeschaltet (**`True`**) oder ausgeschaltet (**`False`**).

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Klasse "NvCursor" befindet sich im Kapitel "NvCursor: Überblick", S. 101.

3.13 Klasse "NvAxis"

Ein Objekt der **NextView®4** Klasse "NvAxis" kann nicht über den **`New`** Befehl angelegt werden. Stattdessen wird von den Funktionen **`NvGraphDisplay.xAxis`** und **`NvGraphDisplay.yAxis`** der Klasse "NvGraphDisplay" ein Objekt der **NextView®4** Klasse "NvAxis" erzeugt.

```
Dim g As NvGraphDisplay
Set g ?= NvCurrentProject.Sheet(1).Display(1)
' Ist das Display eine Signalanzeige?
If g Is Nothing Then
  Print "Anzeige ist keine Signalanzeige"
  Exit Sub
End If

Dim ax As NvAxis
Set ax = g.xAxis
If ax.IsDateMode Then
  print "x-Achse Minimum: " & TimeStampStr(ax.Min)
  print "x-Achse Maximum: " & TimeStampStr(ax.Max)
Else
  print "x-Achse Minimum: " & ax.Min
  print "x-Achse Maximum: " & ax.Max
End If
Dim ay As NvAxis
Set ay = g.yAxis(1)
ay.Min = 0.1
ay.Max = 1.1

print "y-Achse gesetzt auf: " & ay.Min & " .. " & ay.Max
```

3.13.1 NvAxis: Überblick

Elementfunktion	Beschreibung
`NvAxis.Min`	liefert den unteren Wert der Achse
`NvAxis.Max`	liefert den oberen Wert der Achse
`NvAxis.SetDateMode`	legt den Zeitmodus der x-Achse fest auf Absolut- oder Relativzeit
`NvAxis.IsDateMode`	liefert den Zeitmodus der x-Achse (Absolut- oder Relativzeit)

3.13.2 NvAxis.Min

```
' Wert lesen:
  Function Min () As Double
' Wert setzen:
  Min = newMin      ' newMin As Double
```

Setzt den unteren Wert der Achse oder gibt diesen zurück. Bei der x-Achse ist der Wert abhängig vom Anzeigeformat. Im Datumsformat wird der Wert in Absolutzeit (PC-Lokalzeit) dargestellt, in Relativzeit in Sekunden.

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Klasse "NvAxis" befindet sich im Kapitel "NvAxis: Überblick", S. 103.

3.13.3 NvAxis.Max

```
' Wert lesen:
  Function Max () As Double
' Wert setzen:
  Max = newMax      ' newMax As Double
```

Setzt den oberen Wert der Achse oder gibt diesen zurück. Bei der x-Achse ist der Wert abhängig vom Anzeigeformat. Im Datumsformat wird der Wert in Absolutzeit (PC-Lokalzeit) dargestellt, in Relativzeit in Sekunden.

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Klasse "NvAxis" befindet sich im Kapitel "NvAxis: Überblick", S. 103.

3.13.4 NvAxis.SetDateMode

```
' x-Axis only
Sub SetDateMode (mode as Boolean)
```

Legt das Anzeigeformat einer x-Achse fest. Bei **`True`** wird die x-Achse im Datumsformat angezeigt, bei **`False`** in Relativzeit. Auf die y-Achse hat die Funktion keine Auswirkung.

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Klasse "NvAxis" befindet sich im Kapitel "NvAxis: Überblick", S. 103.

3.13.5 NvAxis.IsDateMode

```
' x-Axis only
Function IsDateMode () as Boolean
```

Gibt den eingestellten Anzeigemodus einer x-Achse zurück. Bei **`True`** wird die Anzeige im Datumsformat dargestellt, bei **`False`** in Relativzeit.

```
Dim g As NvGraphDisplay
set g ?= NvCurrentProject.Sheet(1).Display(1)
If Not(g is Nothing) Then
  Dim lTime as Double
  lTime = g.Signal(1).TimeStamp
  print "Start (Localtime) " & TimeStampStr(lTime)
  ' dependent on x-axis mode the position values on
  ' the x-axis has to be set differently
  If g.xAxis.IsDateMode Then
    g.xAxis.Min = lTime + TimeSerial(0,0,30)
    g.xAxis.Max = lTime + TimeSerial(0,0,90)
    g.WhiteCursor.Enabled
    g.WhiteCursor.Value = lTime + TimeSerial(0,0,60)
  Else
    g.xAxis.Min = 30 ' s
    g.xAxis.Max = 90 ' s
    g.WhiteCursor.Enabled
    g.WhiteCursor.Value = 60 ' s
  End If
End If
```

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Klasse "NvAxis" befindet sich im Kapitel "NvAxis: Überblick", S. 103.

3.14 Klasse "NvProgress"

Ein Objekt der **NextView®4** Klasse "NvProgress" zeigt eine Fortschrittsanzeige in **NextView®4** an. Ein Objekt der **NextView®4** Klasse "NvProgress" wird über den **New** Befehl angelegt.

```
Dim prog as NvProgress
set prog = new NvProgress
prog.Init (200)
Dim i as Integer
i = 0

Do While (i < 200)
  prog.setstep "Step " & i & " " & rnd, I
  sleep 10
  if prog.aborted then exit do
  i = i + 1
loop

prog.done
```

3.14.1 NvProgress: Überblick

Elementfunktion	Beschreibung
`NvProgress.Init`	initialisiert die Fortschrittsanzeige
`NvProgress.SetStep`	setzt die Fortschrittsanzeige auf angegebene Stufe
`NvProgress.Aborted`	meldet vorzeitigen Abbruch der Fortschrittsanzeige
`NvProgress.Done`	schließt die Fortschrittsanzeige

3.14.2 NvProgress.Init

```
' Fortschrittsanzeige einrichten:
  Sub Init (stufen as Integer)
```

Dieser Befehl richtet eine Fortschrittsanzeige mit der angegebenen Stufenanzahl ein.

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Klasse "NvProgress" befindet sich im Kapitel "NvProgress: Überblick", S. 106.

3.14.3 NvProgress.SetStep

```
' Progress Stufe setzen:
  Sub SetStep (stufe as Integer)
```

Dieser Befehl setzt die Fortschrittsanzeige auf die angegebene Stufe.

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Klasse "NvProgress" befindet sich im Kapitel "NvProgress: Überblick", S. 106.

3.14.4 NvProgress.Aborted

```
' Abbruch der Fortschrittsanzeige abfragen:
  Function Aborted () as Boolean
```

Gibt **`True`** zurück, wenn die "Abbruch"-Taste der Fortschrittsanzeige gedrückt wurde.

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Klasse "NvProgress" befindet sich im Kapitel "NvProgress: Überblick", S. 106.

3.14.5 NvProgress.Done

```
' Fortschrittsanzeige schliessen:
  Sub Done ()
```

Mit diesem Befehl wird die Fortschrittsanzeige geschlossen.

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Klasse "NvProgress" befindet sich im Kapitel "NvProgress: Überblick", S. 106.

3.15 Klasse "NvOpenDataFile"

Die Klasse "NvOpenDataFile" dient zum Öffnen von Datendateien zur Auswertung in **NextView®4 Script**. Ein Objekt der **NextView®4** Klasse "NvOpenDataFile" wird über den **New** Befehl angelegt.

```
Dim o As NvOpenDataFile
Dim f As NvDataFile

Set o = New NvOpenDataFile
If Not o.Browse Then Exit Sub
o.Flags = nvSfReadOnly

Set f = o.Open
If f Is Nothing Then
  Print "Datei " & o.Path & " nicht gefunden."
  Exit Sub
End If

Print "Datei " & o.Path & " enthält " & f.SignalCount & " Signale."
```

3.15.1 NvOpenDataFile: Überblick

Elementfunktion	Beschreibung
`NvOpenDataFile.Browse`	zeigt den Dialog "Datei öffnen"
`NvOpenDataFile.Path`	bestimmt den Dateinamen oder gibt diesen zurück
`NvOpenDataFile.Flags`	setzt Optionen oder gibt diese zurück
`NvOpenDataFile.Open`	öffnet die Datei

3.15.2 NvOpenDataFile.Browse

```
Function Browse () As Boolean
```

Zeigt den Dialog "Datei öffnen". Wählt der Benutzer eine Datei, so wird der Dateiname gesetzt und **Browse** liefert **True** zurück. Der selektierte Dateiname kann über das Property `NvOpenDataFile.Path` abgefragt und verändert

werden. Über die Funktion **Open** (s. "**NvOpenDataFile.Open**", S. 110) kann die selektierte Datei geöffnet werden. Bricht der Benutzer bei **Browse** den Dialog "Datei öffnen" ab, so wird **False** zurückgegeben.

Eine Auflistung aller verfügbaren Elementfunktionen der Klasse "NvOpenDataFile" befindet sich im Kapitel "NvOpenDataFile: Überblick", S. 108.

3.15.3 NvOpenDataFile.Path

```
' Pfad lesen:
  Function Path () As String
' Pfad setzen:
  Path = newPath          ' newPath As String
```

Liefert oder setzt den Dateinamen der zu öffnenden Datei.

```
...
Dim o As NvOpenDataFile
Dim f As NvDataFile
Set o = New NvOpenDataFile
o.Path = "c:\data.lfx"
o.Flags = nvSfReadOnly
Set f = o.Open
If f Is Nothing Then
  Print "Datei " & o.Path & " nicht gefunden."
  Exit Sub
End If
...
```

Eine Auflistung aller verfügbaren Elementfunktionen der Klasse "NvOpenDataFile" befindet sich im Kapitel "NvOpenDataFile: Überblick", S. 108.

3.15.4 NvOpenDataFile.Flags

```
' Flags lesen:
  Function Flags () As Integer
' Flags setzen:
  Flags = newFlags        'newFlags as Integer
```

Liefert oder setzt die Optionen, wie die Datei zu öffnen ist.

flags	Beschreibung
nvSfReadOnly	öffnet die Datei mit Schreibschutz
nvSfReadWrite	öffnet die Datei zum Lesen und Schreiben

Eine Auflistung aller verfügbaren Elementfunktionen der Klasse "NvOpenDataFile" befindet sich im Kapitel "NvOpenDataFile: Überblick", S. 108.

3.15.5 NvOpenDataFile.Open

```
Function Open (Optional path as String, Optional flags as
Integer) As NvDataFile
```

Öffnet die in **NvOpenDataFile.Path** eingestellte Messdatei mit den in **NvOpenDataFile.Flags** eingestellten Optionen und gibt diese als **NvDataFile** Objekt zurück. Falls die Datei nicht existiert oder nicht geöffnet werden konnte, wird **Nothing** zurückgeliefert (s. "Klasse "NvDataFile"", S. 116).

```
Dim o As NvOpenDataFile
Dim f As NvDataFile
...
Set o = New NvOpenDataFile
Set f = o.Open ("c:\data.lfx", nvSfReadOnly)
...
```

Alternative Syntax:

```
Dim o As NvOpenDataFile
Dim f As NvDataFile
...
Set o = New NvOpenDataFile
o.Path = "c:\data.lfx"
o.Flags = nvSfReadOnly
Set f = o.Open
...
```

Eine Auflistung aller verfügbaren Elementfunktionen der Klasse "NvOpenDataFile" befindet sich im Kapitel "NvOpenDataFile: Überblick", S. 108.

3.16 Klasse "NvCreateDataFile"

Zum Erzeugen von Datendateien gibt es in **NextView®4 Script** die Klasse "NvCreateDataFile". Ein Objekt der **NextView®4** Klasse "NvCreateDataFile" wird über den **New** Befehl angelegt.

```
Const pi As Double = 3.14159265358979323846264338327 95
Const samples As Integer = 1000

Sub Test()
  Dim c As NvCreateDataFile
  Set c = New NvCreateDataFile

  c.Add nvSfAnalog, samples
  c.Path = "c:\sinus.lfx"

  Dim f As NvDataFile
  Set f = c.Create

  Dim s As NvSignal
  Set s = f.Signal (1)

  s.xResolution = 0.1    ' Messfrequenz 10Hz angeben

 ' Definitionen der y-Signalparameter
  s.yUnit = "mm"
  s.yRangeMin = -5
  s.yRangeMax = 5
  s.yMin = -5
  s.yMax = 5

  Dim i As Integer
  For i = 1 To samples
    s.Value(i) = 4 * Sin (2 * pi * (i-1)/samples)
  Next
End Sub
```

3.16.1 NvCreateDataFile: Überblick

Elementfunktion	Beschreibung
NvCreateDataFile.Add	fügt ein Signal der Liste hinzu, die für **Create** verwendet wird
NvCreateDataFile.Reset	entfernt alle Signale aus der **Create**-Liste
NvCreateDataFile.Browse	zeigt den "Datei speichern" Dialog
NvCreateDataFile.Path	setzt den Dateinamen oder gibt diesen zurück
NvCreateDataFile.FileType	setzt das Dateiformat oder gibt dieses zurück
NvCreateDataFile.AskOverwrite	setzt die Option der "Überschreiben" Abfrage oder gibt diese zurück
NvCreateDataFile.Create	erzeugt die Datei

3.16.2 NvCreateDataFile.Add

```
Sub Add (type As Integer, samples As Integer)
```

Fügt ein Signal mit **sample** Messwerten der Liste der zu erzeugenden Signale hinzu.

Die folgende Tabelle gibt die Konstanten für **type** an. Bitte benutzen Sie nur diese Konstanten!

type	Beschreibung
nvSfAnalog	analoges Signal
nvSfDigital	digitales Signal

Eine Auflistung aller verfügbaren Elementfunktionen der Klasse "NvCreateDataFile" findet sich im Kapitel "NvCreateDataFile: Überblick", S. 112.

3.16.3 NvCreateDataFile.Reset

```
Sub Reset ()
```

Setzt die Liste der zu erzeugenden Signale zurück, löscht also alle Einträge.

Eine Auflistung aller verfügbaren Elementfunktionen der Klasse "NvCreateDataFile" findet sich im Kapitel "NvCreateDataFile: Überblick", S. 112.

3.16.4 NvCreateDataFile.Browse

```
Function Browse () As Boolean
```

Zeigt den Dialog "Datei öffnen". Bricht der Benutzer den Dialog ab, so wird **`False`** zurückgegeben. Wählt der Benutzer eine Datei, so wird der Dateiname gesetzt und **`Browse`** liefert **`True`** zurück. Der Dateiname kann über das Property **`NvCreateDataFile.Path`** abgefragt und verändert werden.

Eine Auflistung aller verfügbaren Elementfunktionen der Klasse "NvCreateDataFile" findet sich im Kapitel "NvCreateDataFile: Überblick", S. 112.

3.16.5 NvCreateDataFile.Path

```
' Pfad lesen:
  Function Path () As String
' Pfad setzen:
  Path = newPath          ' newPath As String
```

Liefert oder setzt den Dateinamen der zu erzeugenden Datei.

Eine Auflistung aller verfügbaren Elementfunktionen der Klasse "NvCreateDataFile" findet sich im Kapitel "NvCreateDataFile: Überblick", S. 112.

3.16.6 NvCreateDataFile.FileType

```
' Dateiformat lesen:
  Function FileType () As String
' Dateiformat setzen:
  FileType = newType          ' newType As String
```

Liefert oder setzt das Dateiformat der zu erzeugenden Datei.

Folgende Formate werden dabei unterstützt:

Name	Beschreibung
LFX	**NextView®4** Signaldatei
DAFF	TurboLab Signaldatei
DIADEM	DIAdem Signaldatei
ASCII	ASCII-Text Signaldatei

Eine Auflistung aller verfügbaren Elementfunktionen der Klasse "NvCreateDataFile" findet sich im Kapitel "NvCreateDataFile: Überblick", S. 112.

3.16.7 NvCreateDataFile.AskOverwrite

```
' Wert lesen:
  Function AskOverwrite () As Boolean
' Wert setzen:
  AskOverwrite = newMode   ' newMode As Boolean
```

Liefert oder setzt die Option für die vorherige Abfrage eine bereits vorhandene Datei zu überschreiben oder nicht. Steht der Wert auf **`False`**, wird ohne Nachfrage überschrieben.

Eine Auflistung aller verfügbaren Elementfunktionen der Klasse "NvCreateDataFile" findet sich im Kapitel "NvCreateDataFile: Überblick", S. 112.

3.16.8 NvCreateDataFile.Create

```
Function Create () As NvDataFile
```

Erzeugt die in **`NvCreateDataFile.Path`** eingestellte Messdatei mit der durch **`NvCreateDataFile.Add`** erzeugten Signalliste und gibt diese als NvDataFile Objekt zurück, oder **`Nothing`** falls die Datei nicht existiert oder nicht geöffnet werden konnte (s. "Klasse "NvDataFile"", S. 116).

Eine Auflistung aller verfügbaren Elementfunktionen der Klasse "NvCreateDataFile" findet sich im Kapitel "NvCreateDataFile: Überblick", S. 112.

3.17 Klasse "NvDataFile"

Ein Objekt der **NextView®4** Klasse "NvDataFile" kann nicht über den **`New`** Befehl angelegt werden. Es wird stattdessen von der Funktion **`NvOpenDataFile.Open`** (Klasse "NvOpenDataFile") oder **`NvCreateDataFile.Create`** (Klasse "NvCreateDataFile") zurückgegeben.

```
Dim o As NvOpenDataFile
Dim f As NvDataFile

Set o = New NvOpenDataFile
If Not o.Browse Then Exit Sub

Set f = o.Open
If f Is Nothing Then
  Print "Datei " & o.Path & " nicht gefunden."
  Exit Sub
End If
Print "Datei " & o.Path & " enthält " & f.SignalCount & " Signale."
```

3.17.1 NvDataFile: Überblick

Elementfunktion	Beschreibung
`NvDataFile.Name`	liefert den Namen der Messdatei
`NvDataFile.SignalCount`	liefert die Anzahl der Signale der Messdatei
`NvDataFile.Signal`	liefert ein Signal aus der Messdatei
`NvDataFile.CreateTrain`	macht aus der vorhandenen Messdatei einen Messdatei-Train und liefert ein Objekt der Klasse "NvDataTrain" zurück

3.17.2 NvDataFile.Name

```
Function Name () As String
```

Gibt den Namen der Messdatei zurück.

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Klasse "NvDataFile" befindet sich im Kapitel "NvDataFile: Überblick", S. 116.

3.17.3 NvDataFile.SignalCount

```
Function SignalCount () As Integer
```

Liefert die Anzahl der Signale in der Messdatei.

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Klasse "NvDataFile" befindet sich im Kapitel "NvDataFile: Überblick", S. 116.

3.17.4 NvDataFile.Signal

```
Function Signal (index As Integer) As NvSignal
```

Liefert das Signal Nummer **`index`** oder **`Nothing`**, falls dieses nicht vorhanden ist (s. "Klasse "NvSignal"", S. 118").

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Klasse "NvDataFile" befindet sich im Kapitel "NvDataFile: Überblick", S. 116.

3.17.5 NvDataFile.CreateTrain

```
Function CreateTrain () As NvDataTrain
```

Mit diesem Befehl wird eine Messdatei als Traindatei angelegt. Weitere passende Messdateien können mit dem Befehl **`NvDataTrain.Dock`** an die Traindatei angehängt werden (s. "Klasse "NvDataTrain"", S. 134).

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Klasse "NvDataFile" befindet sich im Kapitel "NvDataFile: Überblick", S. 116.

3.18 Klasse "NvSignal"

Ein Objekt der **NextView®4** Klasse "NvSignal" kann nicht über den **`New`** Befehl angelegt werden. Es wird stattdessen von der Funktion **`NvDataFile.Signal`** der Klasse "NvDataFile" oder **`NvGraphDisplay.Signal`** der Klasse "NvGraphDisplay" erzeugt.

```
' öffnet die Messdatei "c:\data.lfx", holt das erste Signal
' und gibt dessen Namen in der Nachrichtenanzeige aus

Dim o As NvOpenDataFile
Set o = New NvOpenDataFile
o.Path = "c:\data.lfx"
Dim file As NvDataFile
Set file = o.Open

If file Is Nothing Then
  Print "Konnte data.lfx nicht öffnen"
Else
  Dim sig As NvSignal
  Set sig = file.Signal(1)

  If sig Is Nothing Then
    Print "Konnte Signal 1 nicht öffnen"
  Else
    Print "Signal 1 heißt " & sig.Name
  End If
End If
```

3.18.1 NvSignal: Überblick

Elementfunktion	Beschreibung
NvSignal.Name	Setzen oder Auslesen des Signalnamens
NvSignal.File	liefert das zugehörige Datenfile als ein Objekt der Klasse "NvDataFile" zurück
NvSignal.Group	Setzen oder Auslesen der Signalgruppe
NvSignal.Comment	Setzen oder Auslesen des Signalkommentars
NvSignal.Type	gibt aus, ob Signal ist analog oder digital
NvSignal.Samples	gibt die Anzahl der Messwerte eines Signals aus
NvSignal.Prehist	gibt die Anzahl der Messwerte der Vorgeschichte eines Signals aus
NvSignal.Posthist	gibt die Anzahl der Messwerte der Nachgeschichte eines Signals aus
NvSignal.Timestamp	Start der Aufzeichnung in PC Ortszeit im Windows® Datum-Uhrzeit-Format
NvSignal.xStart	Setzen oder Auslesen des Signalanfangs
NvSignal.xResolution	Setzen oder Auslesen der x-Auflösung (Zeit zwischen Abtastungen)
NvSignal.xUnit	Setzen oder Auslesen der x-Achseneinheit des Signals
NvSignal.yMin	Setzen oder Auslesen der unteren Grenze des y-Anzeigen-bereichs
NvSignal.yMax	Setzen oder Auslesen der oberen Grenze des y-Anzeigen-bereichs
NvSignal.yRangeMin	Setzen oder Auslesen der unteren Messbereichsgrenze
NvSignal.yRangeMax	Setzen oder Auslesen der oberen Messbereichsgrenze
NvSignal.yUnit	Setzen oder Auslesen der y-Achseneinheit des Signals
NvSignal.yUsing	Setzen oder Auslesen des Ausgabeformats der y-Werte des Signals
NvSignal.Value	Setzen oder Auslesen eines Signalwerts
NvSignal.ValueAt	Setzen oder Auslesen eines Signalwerts zum Zeitpunkt x

3.18.2 NvSignal.Name

```
' Name lesen:
  Function Name () As String
' Name setzen:
  Name = newName      ' newName As String
```

Setzt den Namen des Signals oder gibt diesen zurück.

```
' hole Signal
Dim sig As NvSignal
Set sig = file.Signal(1)
...

' gibt den alten Namen aus und setzt dann den neuen des 1. Signals
Print sig.Name
sig.Name = "Mein Signal"
```

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Klasse "NvSignal" befindet sich im Kapitel "NvSignal: Überblick", S. 119.

3.18.3 NvSignal.File

```
Function File () As NvDataFile
```

Die Messdatei des Signals wird als Objekt der Klasse "NvDataFile" zurückgeliefert.

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Klasse "NvSignal" befindet sich im Kapitel "NvSignal: Überblick", S. 119.

3.18.4 NvSignal.Group

```
' Gruppe lesen:
  Function Group () As String
' Gruppe setzen:
  Group = newGroup      ' newGroup As String
```

Setzt die Gruppe des Signals oder gibt diese zurück.

```
' hole Signal
Dim sig As NvSignal
Set sig = file.Signal(1)
...

' gibt den alten Namen aus und setzt dann den neuen des 1. Signals
Print sig.Group
sig.Group = "Meine Gruppe"
```

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Klasse "NvSignal" befindet sich im Kapitel "NvSignal: Überblick", S. 119.

3.18.5 NvSignal.Comment

```
' Kommentar lesen:
  Function Comment () As String
' Kommentar setzen:
  Comment = newComment     ' newComment As String
```

Setzt den Kommentar des Signals oder gibt diesen zurück.

```
' hole Signal
Dim sig As NvSignal
Set sig = file.Signal(1)
...

' gibt den alten Kommentar aus und setzt den neuen des 1. Signals
Print sig.Comment
sig.Comment = "Mein Kommentar zum Signal"
```

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Klasse "NvSignal" befindet sich im Kapitel "NvSignal: Überblick", S. 119.

3.18.6 NvSignal.Type

```
Function Type () As Integer
```

Gibt den Typ des Signals zurück. Dabei werden folgende Werte zurückgegeben:

flags	Bedeutung
nvSfAnalog	Signal ist analog
nvSfDigital	Signal ist digital

```
' hole Signal
Dim sig As NvSignal
Set sig = file.Signal(1)
...

If sig.Type = sfAnalog Then
  Print "Signal 1 ist analog"
Else
  Print "Signal 1 ist digital"
End If
```

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Klasse "NvSignal" befindet sich im Kapitel "NvSignal: Überblick", S. 119.

3.18.7 NvSignal.Samples

```
Function Samples () As Integer
```

Gibt die Anzahl der Messwerte des Signals zurück.

```
' hole Signal
Dim sig As NvSignal
Set sig = file.Signal(1)
...

Print "Signal 1 enthält " & sig.Samples & " Messwerte"
```

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Klasse "NvSignal" befindet sich im Kapitel "NvSignal: Überblick", S. 119.

3.18.8 NvSignal.Prehist

```
Function Prehist () As Integer
```

Gibt die Anzahl der Messwerte der Vorgeschichte des Signals zurück.

```
' hole Signal
Dim sig As NvSignal
Set sig = file.Signal(1)
...

Print "Signal 1 enthält " & sig.Prehist
Print " Messwerte als Vorgeschichte"
```

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Klasse "NvSignal" befindet sich im Kapitel "NvSignal: Überblick", S. 119.

3.18.9 NvSignal.Posthist

```
Function Posthist () As Integer
```

Gibt die Anzahl der Messwerte der Nachgeschichte des Signals zurück.

```
' hole Signal
Dim sig As NvSignal
Set sig = file.Signal(1)
...

Print "Signal 1 enthält " & sig.Posthist
Print " Messwerte als Nachgeschichte"
```

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Klasse "NvSignal" befindet sich im Kapitel "NvSignal: Überblick", S. 119.

3.18.10 NvSignal.Timestamp

```
Function Timestamp () As Double
```

Gibt die Startzeit der Aufzeichnung zurück. Der Wert wird in der Ortszeit des PCs dargestellt und kann mit der Funktion **LocalToSystemTime** in die UTC-Zeit umgewandelt werden.

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Klasse "NvSignal" befindet sich im Kapitel "NvSignal: Überblick", S. 119.

3.18.11 NvSignal.xStart

```
' Startzeit lesen:
  Function xStart () As Double
' Startzeit setzen:
  xStart = newStart       ' newStart As Double
```

Setzt den Anfangspunkt des Signals oder gibt diesen zurück. Der Wert wird in

Sekunden angegeben (z. B. -1.2s)

```
' hole Signal
Dim sig As NvSignal
Set sig = file.Signal(1)
...

' erst alten Anfangspunkt lesen, dann den neuen des 1. Signals setzen
Print sig.xStart
sig.xStart = 0.0 ' s
```

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Klasse "NvSignal" befindet sich im Kapitel "NvSignal: Überblick", S. 119.

3.18.12 NvSignal.xResolution

```
' x-Auflösung lesen:
  Function xResolution () As Double
' x-Auflösung setzen:
  xResolution = newResolution   ' newResolution As Double
```

Setzt die x-Auflösung des Signals, d. h. in der Regel die Zeit in Sekunden zwischen den Messwerten oder gibt diese zurück.

```
' hole Signal
Dim sig As NvSignal
Set sig = file.Signal(1)
...

' gibt zuerst die alte x-Auflösung aus und setzt dann
' die neue des 1. Signals
Print sig.xResolution
sig.xResolution = 0.01 ' setze auf 10 ms
```

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Klasse "NvSignal" befindet sich im Kapitel "NvSignal: Überblick", S. 119.

3.18.13 NvSignal.xUnit

```
' Einheit der x-Werte lesen:
  Function xUnit () As String
' Einheit der x-Werte setzen:
  xUnit = newUnit         ' newUnit As String
```

Setzt die x-Einheit des Signals oder gibt diese zurück.

```
' hole Signal
Dim sig As NvSignal
Set sig = file.Signal(1)
...

' gibt zuerst die alte x-Einheit aus und setzt dann die neue
' des 1. Signals
Print sig.xUnit
sig.xUnit = "sec"
```

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Klasse "NvSignal" befindet sich im Kapitel "NvSignal: Überblick", S. 119.

3.18.14 NvSignal.yMin

```
' untere Grenze des y-Anzeigebereichs lesen:
  Function yMin () As Double
' untere Grenze des y-Anzeigebereichs setzen:
  yMin = newMin           ' newMin As Double
```

Legt die untere Grenze des Standardanzeigebereichs der y-Achse für die Signalanzeige in **NextView®4** fest oder gibt diese zurück.

```
' hole Signal
Dim sig As NvSignal
Set sig = file.Signal(1)
...

' gibt die alte Grenze aus und setzt dann die neue des 1. Signals
Print sig.yMin
sig.yMin = -1.0
```

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Klasse "NvSignal" befindet sich im Kapitel "NvSignal: Überblick", S. 119.

3.18.15 NvSignal.yMax

```
' obere Grenze des y-Anzeigebereichs lesen:
  Function yMax () As Double
' obere Grenze des y-Anzeigebereichs setzen:
  yMax = newMax           ' newMax As Double
```

Legt die obere Grenze des Standardanzeigebereichs der y-Achse für die Signalanzeige in **NextView®4** fest oder gibt diese zurück.

```
' hole Signal
Dim sig As NvSignal
Set sig = file.Signal(1)
...

' gibt die alte Grenze aus und setzt dann die neue des 1. Signals
Print sig.yMax
sig.yMax = 1.0
```

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Klasse "NvSignal" befindet sich im Kapitel "NvSignal: Überblick", S. 119.

3.18.16 NvSignal.yRangeMin

```
' untere Grenze des Messbereichs lesen:
  Function yRangeMin () As Double
' untere Grenze des Messbereichs setzen:
  yRangeMin = newRangeMin   ' newRangeMin As Double
```

Setzt die untere Messbereichsgrenze des Messsystems oder gibt diese zurück.

```
' hole Signal
Dim sig As NvSignal
Set sig = file.Signal(1)
...

' gibt die alte Grenze aus und setzt dann die neue des 1. Signals
Print sig.yRangeMin
sig.yRangeMin = -10.0
```

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Klasse "NvSignal" befindet sich im Kapitel "NvSignal: Überblick", S. 119.

3.18.17 NvSignal.yRangeMax

```
' obere Grenze des Messbereichs lesen:
  Function yRangeMax () As Double
' obere Grenze des Messbereichs setzen:
  yRangeMax = newRangeMax   ' newRangeMax As Double
```

Setzt die obere Messbereichsgrenze des Messsystems oder gibt diese zurück.

```
' hole Signal
Dim sig As NvSignal
Set sig = file.Signal(1)
...

' gibt die alte Grenze aus und setzt dann die neue des 1. Signals
Print sig.yRangeMax
sig.yRangeMax = 10.0
```

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Klasse "NvSignal" befindet sich im Kapitel "NvSignal: Überblick", S. 119.

3.18.18 NvSignal.yUnit

```
' Einheit der y-Werte lesen:
  Function yUnit () As String
' Einheit der y-Werte setzen:
  yUnit = newUnit         ' newUnit As String
```

Setzt die y-Einheit des Signals oder gibt diese zurück.

```
' hole Signal
Dim sig As NvSignal
Set sig = file.Signal(1)
...

' gibt zuerst die alte Einheit aus und setzt dann die neue
' des 1. Signals
Print sig.yUnit
sig.yUnit = "°C"
```

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Klasse "NvSignal" befindet sich im Kapitel "NvSignal: Überblick", S. 119.

3.18.19 NvSignal.yUsing

```
Function yUsing As NvUsing
```

Bestimmt das Ausgabeformat der y-Werte (z. B. exponentielle Schreibweise, wissenschaftliche Darstellung, etc.) oder gibt es zurück (s. "Klasse "NvUsing"", S. 131).

```
' hole Signal
Dim sig As NvSignal
Set sig = file.Signal(1)
...

' gibt zuerst das alte Ausgabeformats aus und verändert es dann
' für das 1. Signal
Print sig.yUsing
sig.yUsing.Width = 9
sig.yUsing.Fraction = 5
sig.yUsing.Type = nvUsingFixed
```

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Klasse "NvSignal" befindet sich im Kapitel "NvSignal: Überblick", S. 119.

3.18.20 NvSignal.Value

```
' Wert lesen:
  Function Value (pos As Integer) As Double
' Wert setzen:
  Value (pos) = newValue   'newValue As Double
```

Setzt den Wert eines Signals an der Stelle **pos** oder gibt diesen zurück. Die Variable **pos** liegt zwischen 1 und der Anzahl der Samples.

```
' hole Signal
Dim sig As NvSignal
Set sig = file.Signal(1)
...

' setzt Sample Nummer 3
sig.Value(3) = 0

' gibt die ersten 5 Samples aus
Dim i As Integer

For i = 1 To 5
  Print sig.Value(i)
Next
```

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Klasse "NvSignal" befindet sich im Kapitel "NvSignal: Überblick", S. 119.

3.18.21 NvSignal.ValueAt

```
' Wert lesen:
  Function ValueAt (xPos As Double, Optional isDate as
                    Boolean = False) As Double
' Wert setzen:
  ValueAt (xPos, isDate) = newValue   'newValue As Double
```

Setzt den Wert eines Signals zum Zeitpunkt **xPos** oder gibt diesen zurück. Der Positionswert **xPos** kann im Datum-Uhrzeit-Format (Absolutzeit; **isDate** = **True**) oder in Sekunden (Relativzeit; **isDate** = **False**) angegeben werden.

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Klasse "NvSignal" befindet sich im Kapitel "NvSignal: Überblick", S. 119.

3.19 Klasse "NvUsing"

Ein Objekt der **NextView®4** Klasse "NvUsing" kann nicht über den **`New`** Befehl angelegt werden. Es wird stattdessen von der Funktion **`NvSignal.yUsing`** der Klasse "NvSignal" erzeugt.

```
Dim o As NvOpenDataFile
set o = New NvOpenDataFile

Dim file As NvDataFile
Dim i As Integer
Set file = o.Open ("c:\data\test-1-1.lfx", nvSFReadWrite)

If file Is Nothing Then
  Print "Konnte lfx nicht öffnen"
Else
  Dim sig As NvSignal
  Set sig = file.Signal(1)

  If sig Is Nothing Then
    Print "Konnte Signal 1 nicht öffnen"
  Else
    Print "Signal 1 heißt " & sig.Name
    Print "old yUsing: " & sig.yUsing

    ' neues Ausgabeformat in Exponentialschreibweise
    sig.yUsing.Width = 12
    sig.yUsing.Fraction = 3
    sig.yUsing.Type = nvUsingEngineering

    Print "new yUsing: " & sig.yUsing
 End If

End If
```

3.19.1 NvUsing: Überblick

Elementfunktion	Beschreibung
`NvUsing.Width`	Feldbreite: Anzahl der Zeichen pro Zahlenwert
`NvUsing.Type`	Ausgabeformat
`NvUsing.Fraction`	Anzahl der Nachkommastellen
`NvUsing.Print`	gibt eine Zeichenkette zurück, die das Ausgabeformat beschreibt

3.19.2 NvUsing.Width

```
' Feldbreite lesen:
  Function Width () As Integer
' Feldbreite setzen:
  Width = newWidth        'newWidth as Integer
```

Setzt bzw. liest die Anzahl der Zeichen pro Zahlenwert der als y-Wert für ein Signal ausgegeben wird. Dabei ist ein Zeichen vorbehalten für das Vorzeichen und eines für den Dezimalpunkt.

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Klasse "NvUsing" befindet sich im Kapitel "NvUsing: Überblick", S. 131.

3.19.3 NvUsing.Type

```
' Ausgabeformat lesen:
  Function Type () As Integer
' Ausgabeformat setzen:
  Type = newType        'newType as Integer
```

Setzt bzw. fragt die Darstellung der y-Werte eines Signals ab. Folgende Werte stehen zur Verfügung. Dabei wird bei dem Zahlenbeispiel von einer Feldbreite (**`NvUsing.Width`**) von 9 Stellen inklusive 4 Nachkommastellen (**`NvUsing.Fraction`**) ausgegangen:

val	Bedeutung	Beispiel 123,456 mm
`nvUsingDecimal`	dezimale Darstellung	123.456 mm
`nvUsingHex`	hexadezimale Darstellung	7B mm
`nvUsingFixed`	feste Darstellung	123.4560 mm
`nvUsingEngineering`	Exponentialschreibweise	1.2E+002 mm
`nvUsingScientific`	wissenschaftliche Darstellung	12.3456cm
`nvUsingFixedScientific`	wissenschaftliche Darstellung mit fester Einheit	123.456 mm

val	Bedeutung	Beispiel 123,456 s
nvUsingTime	Uhrzeitformat	z. B. 2min 3.5s (Basiseinheit Sekunden)
nvUsingDate	Datumsformat	<Tag>.<Monat>.<Jahr>

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Klasse "NvUsing" befindet sich im Kapitel "NvUsing: Überblick", S. 131.

3.19.4 NvUsing.Fraction

```
' Anzahl Nachkommastellen lesen:
  Function Fraction () As Integer
' Anzahl Nachkommastellen setzen:
  Fraction = newFraction    'newFraction as Integer
```

Legt die Anzahl der Nachkommastellen fest.

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Klasse "NvUsing" befindet sich im Kapitel "NvUsing: Überblick", S. 131.

3.19.5 NvUsing.Print

```
Function Print () As String
```

Gibt die eingestellten Parameter des Ausgabeformats (**NvUsing.Type**, **NvUsing.Width**, **NvUsing.Fraction**) aus.

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Klasse "NvUsing" befindet sich im Kapitel "NvUsing: Überblick", S. 131.

3.20 Klasse "NvDataTrain"

Ein Objekt der **NextView®4** Klasse "NvDataTrain" kann nicht über den **`New`** Befehl angelegt werden. Es wird stattdessen von der Funktion **`NvDataFile.CreateTrain`** der Klasse "NvDataFile" erzeugt. Die Klasse "NvDataTrain" ist abgeleitet von der Klasse "NvDataFile", daher sind auch alle Aufrufe der Klasse NvDataFile (s. "NvDataFile: Überblick", S. 116) für ein Objekt der Klasse "NvDataTrain" zulässig.

3.20.1 NvDataTrain: Überblick

Elementfunktion	Beschreibung
`NvDataTrain.Dock`	fügt eine Messdatei zur Traindatei hinzu
`NvDataTrain.Undock`	entfernt eine Messdatei aus der Traindatei

3.20.2 NvDataTrain.Dock

```
Function Dock (wagon as NvDataFile) As Boolean
```

Fügt einer existierenden Traindatei die übergebene Messdatei hinzu. War der Aufruf erfolgreich, wird **`True`** zurückgeliefert.

```
Dim f, ftmp as NvDataFile
Dim t as NvDataTrain
Dim i as Integer
Dim str as String
' erste Datei öffnen
str = "c:\data\Test-1-1.lfx"
Set f = New NvOpenDataFile.Open (str, nvSfReadOnly)
If f Is Nothing Then
  Print "Fehler beim Öffnen von " & str
  Exit Sub
End If

' erste Datei zieht den Zug (t)
set t = f.CreateTrain

' test-1-2.lfx und test-1-3.lfx andocken
for i = 2 to 3
  str = "c:\data\Test-1-" & i & ".lfx"
  Set ftmp = New NvOpenDataFile.Open (str, nvSfReadOnly)
  if Not(ftmp is Nothing) then
    if Not t.Dock(ftmp) then
      Print "Fehler: Bei Docken von " & str
    end if
  else
    Print "Fehler: Bei Öffnen von " & str
  end if
next

print "Anzahl Samples test-1-1.lfx: " & f.Signal(1).Samples
print "Anzahl Sample Train (Test-1-1+2+3.lfx): " & _
       t.Signal(1).Samples
```

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Klasse "NvDataTrain" befindet sich im Kapitel "NvDataTrain: Überblick", S. 134.

3.20.3 NvDataTrain.Undock

```
Function Undock (idx as Integer) as Boolean
```

Entfernt aus einer existierenden Traindatei die Messdatei mit Nummer **`idx`**. Der übergebene Indexbereich liegt zwischen 1 und der Anzahl der Messdateien in der Traindatei. War der Aufruf erfolgreich, wird **`True`** zurückgeliefert.

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Klasse "NvDataTrain" befindet sich im Kapitel "NvDataTrain: Überblick", S. 134.

3.21 Klasse "NvFFT"

Ein Objekt der **NextView®4** Klasse "NvFFT" wird über den **New** Befehl angelegt.

```
Dim s As NvSheet
Dim d As NvGraphDisplay

' kurzes Überprüfen, ob Script funktioniert
If NvCurrentProject.SheetCount < 1 Then Exit Sub

Set s = NvCurrentProject.Sheet (1)
If s.DisplayCount < 1 Then Exit Sub

Set d ?= s.Display (1)
If d.SignalCount < 3 Then Exit Sub

Dim fft As NvFFT
Set fft = New NvFFT

' FFT vorbereiten

fft.Add d.Signal(1), 0, nvFftWindowNone, _
        nvFftLines128, nvFftResultsMag, False
fft.Add d.Signal(2), 0, nvFftWindowHanning, _
        nvFftLines512, nvFftResultsComplex, False
fft.Add d.Signal(3), 0, nvFftWindowHamming, _
        nvFftLines128, nvFftResultsMagPhase, True

Dim f As NvDataFile
' FFT ausführen
Set f = fft.Run ("C:\fft.lfx")
```

3.21.1 NvFFT: Überblick

Elementfunktion	Beschreibung
NvFFT.Add	fügt ein Signal der Liste hinzu, die für die FFT verwendet wird
NvFFT.Run	führt eine FFT-Analyse aus
NvFFT.Reset	entfernt alle Signale aus der FFT-Liste

3.21.2 NvFFT.Add

```
Sub Add (sig As NvSignal, xStart As Double,
         fftWindow As Integer, fftLines As Integer,
         fftResults As Integer, fftLog As Boolean)
```

Fügt das Signal **`sig`** der Liste der zu analysierenden Signale für eine FFT hinzu. **`NvSignal.xStart`** beschreibt die Anfangszeit, **`fftLog`**, ob das Ergebnis logarithmisch ist.

Folgende Tabellen geben die Konstanten für **`fftWindow`**, **`fftLines`** und **`fftResults`** an. Bitte benutzen Sie nur diese Konstanten!

FFT - Fenster	Bedeutung
`nvFftWindowNone`	ohne Fenster
`nvFftWindowRoundLast`	letzten Punkt runden
`nvFftWindowHanning`	Hanning-Fenster
`nvFftWindowHamming`	Hamming-Fenster
`nvFftWindowBlackman`	Blackman-Fenster
`nvFftWindowBartlet`	Bartlet-Fenster

FFT - Linien	Bedeutung
`nvFftLines64`	64 Linien
`nvFftLines128`	128 Linien
`nvFftLines256`	256 Linien
`nvFftLines512`	512 Linien
`nvFftLines1024`	1024 Linien
`nvFftLines2048`	2048 Linien
`nvFftLines4096`	4096 Linien
`nvFftLines8192`	8192 Linien

FFT - Ergebnis	Bedeutung
nvFftResultsMag	Betrag
nvFftResultsMagPhase	Betrag & Phase
nvFftResultsPower	Powerspektrum
nvFftResultsComplex	komplexes Ergebnis

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Klasse "NvFFT" befindet sich im Kapitel "NvFFT: Überblick", S. 136.

3.21.3 NvFFT.Run

```
Function Run (path As String,
              Optional askOverwrite As Boolean = True)
              As NvDataFile
```

Führt die FFT für die mit **NvFFT.Add** der Liste hinzugefügten Signale durch und speichert das Ergebnis in der Datei **path**. Ist die Liste leer, wird ein Runtime Error ausgelöst. Stimmen der angegebene Dateiname **path** und der Name einer zu verwendenden Signaldatei überein, wird ebenfalls ein Runtime Error ausgelöst. Die Option **askOverwrite** gibt an, ob vor dem Überschreiben einer bestehenden Datei nachgefragt werden soll. Steht **askOverwrite** auf **False** wird ohne Nachfrage überschrieben.

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Klasse "NvFFT" befindet sich im Kapitel "NvFFT: Überblick", S. 136.

3.21.4 NvFFT.Reset

```
Sub Reset ()
```

Setzt die FFT-Liste der Signale zurück, löscht also alle Einträge.

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Klasse "NvFFT" befindet sich im Kapitel "NvFFT: Überblick", S. 136.

3.22 Klasse "NvIntegration"

Ein Objekt der **NextView®4** Klasse "NvIntegration" wird über den **`New`** Befehl angelegt.

```
Dim s As NvSheet
Dim d As NvGraphDisplay

' kurzes Überprüfen ob Script funktioniert
If NvCurrentProject.SheetCount < 1 Then Exit Sub

Set s = NvCurrentProject.Sheet (1)
If s.DisplayCount < 1 Then Exit Sub

Set d = s.Display (1)
If d.SignalCount < 3 Then Exit Sub

Dim int As NvIntegration
Set int = New NvIntegration

' Integration vorbereiten
int.Add d.Signal(1), 0, 1
int.Add d.Signal(2), 0, 1, False, 1, True, False, 1
int.Add d.Signal(3), 0, 1, True, 0, True

Dim f As NvDataFile
' Integration ausführen
Set f = int.Run ("C:\integration.lfx")
```

3.22.1 NvIntegration: Überblick

Elementfunktion	Beschreibung
`NvIntegration.Add`	fügt ein Signal der Liste hinzu, die für die Integration verwendet wird
`NvIntegration.Run`	führt die Integration aus
`NvIntegration.Reset`	entfernt alle Signale aus der Integrations-Liste

3.22.2 NvIntegration.Add

```
Sub Add (sig As NvSignal, xStart As Double,
         xEnd As Double,
         Optional intAutoOffset As Boolean = True,
         Optional intOffset As Double = 0.0,
         Optional int2nd As Boolean = False,
         Optional intAutoOffset2 As Boolean = True,
         Optional intOffset2 As Double = 0.0)
```

Fügt das Signal **sig** der Liste der zu analysierenden Signale hinzu. **xStart** beschreibt die Anfangszeit, **xEnd** die Endzeit. Ist **int2nd** auf **True** gesetzt, wird ein 2. Integral berechnet (d. h. Integral vom Integral). **intAutoOffset** bzw. **intAutoOffset2** geben an, ob der Offset des Signals automatisch bestimmt wird, oder **intOffset** bzw. **intOffset2** benutzt wird.

Eine Auflistung aller verfügbaren Elementfunktionen der Klasse "NvIntegration" befindet sich im Kapitel "NvIntegration: Überblick", S. 139.

3.22.3 NvIntegration.Run

```
Function Run (path As String) As NvDataFile
```

Führt die Integration für die mit **NvIntegration.Add** der Liste hinzugefügten Signale durch und speichert das Ergebnis in der Datei **path** (s. "Klasse "NvDataFile"", S. 116).

Ist die Liste leer, wird ein Runtime Error ausgelöst. Stimmen der angegebene Dateiname **path** und der Name einer zu verwendenden Signaldatei überein, wird ebenfalls ein Runtime Error ausgelöst.

Die Option **askOverwrite** gibt an, ob vor dem Überschreiben einer bestehenden Datei nachgefragt werden soll. Steht **askOverwrite** auf **False** wird ohne Nachfrage überschrieben.

Eine Auflistung aller verfügbaren Elementfunktionen der Klasse "NvIntegration" befindet sich im Kapitel "NvIntegration: Überblick", S. 139.

3.22.4 NvIntegration.Reset

```
Sub Reset ()
```

Setzt die Liste der zu integrierenden Signale zurück, löscht also alle Einträge.

Eine Auflistung aller verfügbaren Elementfunktionen der Klasse "NvIntegration" befindet sich im Kapitel "NvIntegration: Überblick", S. 139.

3.23 Klasse "NvDifferentiation"

Ein Objekt der **NextView®4** Klasse "NvDifferentiation" wird über den **New** Befehl angelegt.

```
Dim s As NvSheet
Dim d As NvGraphDisplay

' kurzes Überprüfen ob Script funktioniert
If NvCurrentProject.SheetCount < 1 Then Exit Sub

Set s = NvCurrentProject.Sheet (1)
If s.DisplayCount < 1 Then Exit Sub

Set d ?= s.Display (1)
If d.SignalCount < 3 Then Exit Sub

Dim diff As NvDifferentiation
Set diff = New NvDifferentiation

' Differentiation vorbereiten

diff.Add d.Signal(1), 0, 1, 5
diff.Add d.Signal(2), 0, 1, 3
diff.Add d.Signal(3), 0, 1, 7

Dim f As NvDataFile
' Differentiation ausführen
Set f = diff.Run ("C:\diff.lfx")
```

3.23.1 NvDifferentiation: Überblick

Elementfunktion	Beschreibung
`NvDifferentiation.Add`	fügt ein Signal zur Liste der zu differenzierenden Signale hinzu
`NvDifferentiation.Run`	führt die Differentiation aus
`NvDifferentiation.Reset`	entfernt alle Signale aus der Differentiations-Liste

3.23.2 NvDifferentiation.Add

```
Sub Add (sig As NvSignal, xStart As Double,
         xEnd As Double, int diffArea)
```

Fügt das Signal **`sig`** der Liste der zu analysierenden Signale hinzu. **`xStart`** beschreibt die Anfangszeit, **`xEnd`** die Endzeit. **`diffArea`** gibt die Anzahl der Samples der Umgebung an, die in die Berechnung einfließen.

Eine Auflistung aller verfügbaren Elementfunktionen der Klasse "NvDifferentiation" findet sich im Kapitel "NvDifferentiation: Überblick", S. 142.

3.23.3 NvDifferentiation.Run

```
Function Run (path As String,
              Optional askOverwrite As Boolean = True)
              As NvDataFile
```

Führt die Differentiation für die mit **`NvDifferentiation.Add`** der Liste hinzugefügten Signale durch und speichert das Ergebnis in der Datei **`path`** (s. "Klasse "NvDataFile"", S. 116).

Ist die Liste leer, wird ein Runtime Error ausgelöst. Stimmen der angegebene Dateiname **`path`** und der Name einer zu verwendenden Signaldatei überein, wird ebenfalls ein Runtime Error ausgelöst.

Die Option **`askOverwrite`** gibt an, ob vor dem Überschreiben einer bestehenden Datei nachgefragt werden soll. Steht **`askOverwrite`** auf **`False`** wird ohne Nachfrage überschrieben.

Eine Auflistung aller verfügbaren Elementfunktionen der Klasse "NvDifferentiation" findet sich im Kapitel "NvDifferentiation: Überblick", S. 142.

3.23.4 NvDifferentiation.Reset

```
Sub Reset ()
```

Setzt die Liste der zu differenzierenden Signale zurück, löscht also alle Einträge.

Eine Auflistung aller verfügbaren Elementfunktionen der Klasse "NvDifferentiation" findet sich im Kapitel "NvDifferentiation: Überblick", S. 142.

3.24 Klasse "NvFilter"

Ein Objekt der **NextView®4** Klasse "NvFilter" wird über den **New** Befehl angelegt.

```
Dim s As NvSheet
Dim d As NvGraphDisplay

' kurzes Überprüfen ob Script funktioniert
If NvCurrentProject.SheetCount < 1 Then Exit Sub

Set s = NvCurrentProject.Sheet (1)
If s.DisplayCount < 1 Then Exit Sub

Set d ?= s.Display (1)
If d.SignalCount < 3 Then Exit Sub

Dim filter As NvFilter
Set filter = New NvFilter

' Filter vorbereiten

filter.Add d.Signal(1), 0, 1, nvFilterCriticalDamp, _
           nvFilterOrder2, nvFilterLowPass, 10
filter.Add d.Signal(2), 0, 1, nvFilterChebycheff30, _
           nvFilterOrder6, nvFilterBandPass, 5
filter.Add d.Signal(3), 0, 1, nvFilterRunningMean, 0, 0, 20

Dim f As NvDataFile
' Filter ausführen
Set f = filter.Run ("C:\filter.lfx")
```

3.24.1 NvFilter: Überblick

Elementfunktion	Beschreibung
NvFilter.Add	fügt ein Signal der Liste hinzu, auf die der Filter angewendet wird
NvFilter.Run	führt den Filter durch
NvFilter.Reset	entfernt alle Signale aus der Filterliste

3.24.2 NvFilter.Add

```
Sub Add (sig As NvSignal, xStart As Double,
         xEnd As Double, filterType As Integer,
         filterOrder As Integer, filterPass As Integer,
         filterCutoff As Double)
```

Fügt das Signal **sig** der Liste der zu analysierenden Signale hinzu. **xStart** beschreibt die Anfangszeit, **xEnd** die Endzeit.

Der Parameter **filterCutoff** definiert bei Bandpass-Filtern die Filtereckfrequenz. Bei einem laufendem Mittelwert- oder Median-Filter ist **filterCutoff** die Anzahl der Messwerte, die um den betreffenden Messwert herum für die Berechnung verwendet werden.

filterType	Bedeutung
nvFilterCriticalDamp	Kritische Dämpfung
nvFilterBessel	Bessel-Filter
nvFilterButterworth	Butterworth-Filter
nvFilterChebycheff05	Tschebyscheff-Filter, 0.5db
nvFilterChebycheff10	Tschebyscheff-Filter, 1.0db
nvFilterChebycheff20	Tschebyscheff-Filter, 2.0db
nvFilterChebycheff30	Tschebyscheff-Filter, 3.0db
nvFilterRunningMean	laufender Mittelwert-Filter
nvFilterRunningMedian	laufender Median-Filter
nvFilterNothing	Signaldaten werden ohne Filterung in die Ausgabedatei übernommen

filterOrder	Bedeutung
nvFilterOrder2	2. Ordnung
nvFilterOrder4	4. Ordnung
nvFilterOrder6	6. Ordnung
nvFilterOrder8	8. Ordnung

filterPass	Bedeutung
nvFilterLowPass	Tiefpass
nvFilterHighPass	Hochpass
nvFilterBandPass	Bandpass
nvFilterBandElimination	Bandsperre
nvFilterByPass	führt eine Herzoperation durch

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Klasse "NvFilter" befindet sich im Kapitel "NvFilter: Überblick", S. 144.

3.24.3 NvFilter.Run

```
Function Run (path As String,
              Optional askOverwrite As Boolean = True)
              As NvDataFile
```

Wendet den Filter auf die mit **NvFilter.Add** der Liste hinzugefügten Signale durch und speichert das Ergebnis in der Datei **path** (s. "Klasse "NvDataFile"", S. 116).

Ist die Liste leer, wird ein Runtime Error ausgelöst. Stimmen der angegebene Dateiname **path** und der Name einer zu verwendenden Signaldatei überein, wird ebenfalls ein Runtime Error ausgelöst.

Die Option **askOverwrite** gibt an, ob vor dem Überschreiben einer bestehenden Datei nachgefragt werden soll. Steht **askOverwrite** auf **False** wird ohne Nachfrage überschrieben.

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Klasse "NvFilter" befindet sich im Kapitel "NvFilter: Überblick", S. 144.

3.24.4 NvFilter.Reset

```
Sub Reset ()
```

Setzt die Liste der zu filternden Signale zurück, löscht also alle Einträge.

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Klasse "NvFilter" befindet sich im Kapitel "NvFilter: Überblick", S. 144.

3.25 Klasse "NvReduction"

Ein Objekt der **NextView®4** Klasse "NvReduction" wird über den **New** Befehl angelegt.

```
Dim s As NvSheet
Dim d As NvGraphDisplay

' kurzes Überprüfen ob Script funktioniert
If NvCurrentProject.SheetCount < 1 Then Exit Sub

Set s = NvCurrentProject.Sheet (1)
If s.DisplayCount < 1 Then Exit Sub

Set d ?= s.Display (1)
If d.SignalCount < 3 Then Exit Sub

Dim reduce As NvReduction
Set reduce = New NvReduction

' Reduktion vorbereiten

reduce.Add d.Signal(1), 0, 1, nvReductionMin, 10
reduce.Add d.Signal(2), 0, 1, nvReductionAverage
reduce.Add d.Signal(3), 0, 1, nvReductionRMS, 3

Dim f As NvDataFile
' Reduktion ausführen
Set f = reduce.Run ("C:\reduction.lfx")
```

3.25.1 NvReduction: Überblick

Elementfunktion	Beschreibung
NvReduction.Add	fügt ein Signal der Liste hinzu, auf die die Reduktion erfolgt
NvReduction.Run	führt die Reduktion durch
NvReduction.Reset	entfernt alle Signale aus der Reduktionsliste

3.25.2 NvReduction.Add

```
Sub Add (sig As NvSignal, xStart As Double,
         xEnd As Double, reduceType As Integer,
         Optional reduceRatio As Integer = 1)
```

Fügt das Signal **sig** der Liste der Signale hinzu, für die eine Datenreduktion durchgeführt werden soll. **xStart** beschreibt die Anfangszeit, **xEnd** die Endzeit. **reduceRatio** gibt die Anzahl der Messwerte an, auf die sich die Reduktion bezieht.

reduceType	Bedeutung
nvReductionMin	speichert intervallweise das Minimum der Signalwerte
nvReductionMax	speichert intervallweise das Maximum der Signalwerte
nvReductionAverage	speichert intervallweise den Mittelwert der Signalwerte
nvReductionRMS	speichert intervallweise den Effektivwert der Signalwerte

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Klasse "NvReduction" befindet sich im Kapitel "NvReduction: Überblick", S. 148.

3.25.3 NvReduction.Run

```
Function Run (path As String,
              Optional askOverwrite As Boolean = True)
              As NvDataFile
```

Führt die Reduktion auf die mit **`NvFilter.Add`** der Liste hinzugefügten Signale durch und speichert das Ergebnis in der Datei **`path`** (s. "Klasse "NvDataFile"", S. 116).

Ist die Liste leer, wird ein Runtime Error ausgelöst. Stimmen der angegebene Dateiname **`path`** und der Name einer zu verwendenden Signaldatei überein, wird ebenfalls ein Runtime Error ausgelöst.

Die Option **`askOverwrite`** gibt an, ob vor dem Überschreiben einer bestehenden Datei nachgefragt werden soll. Steht **`askOverwrite`** auf **`False`** wird ohne Nachfrage überschrieben.

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Klasse "NvReduction" befindet sich im Kapitel "NvReduction: Überblick", S. 148.

3.25.4 NvReduction.Reset

```
Sub Reset ()
```

Setzt die Liste der zu reduzierenden Signale zurück, löscht also alle Einträge.

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Klasse "NvReduction" befindet sich im Kapitel "NvReduction: Überblick", S. 148.

4 Standard Skript-Befehle

4.1 Standard Funktionen und Prozeduren

4.1.1 Standardfunktionen und Befehle: Überblick

Funktion	Beschreibung
`Print`	gibt angegebene Zeichenkette in der Nachrichtenanzeige von **NextView®4** aus
`Timer`	liefert die Anzahl der Sekunden seit Mitternacht
`TickCount`	liefert die Anzahl der Millisekunden seit dem Start vom Windows®
`TimestampStr`	konvertiert den Start der Aufzeichnung eines Signals (Timestamp) in eine Zeichenkette)
`Format`	liefert formatierte Zeichenkette aus Zahl
`Hex`	liefert Zahl als Zeichenkette im Hexformat
`LCase`	liefert Zeichenkette in Kleinbuchstaben
`UCase`	liefert Zeichenkette in Grossbuchstaben
`Asc`	liefert ASCII-Code vom ersten Buchstaben einer Zeichenkette
`Chr`	liefert das Zeichen des angegebenen ASCII-Codes
`Tab`	liefert Zeichenkette mit Tabs
`Spc`	liefert Zeichenkette mit Leerzeichen
`Len`	liefert die Länge einer Zeichenkette
`Left`	liefert eine bestimmte Anzahl von Zeichen einer Zeichenkette von links
`Right`	liefert eine bestimmte Anzahl von Zeichen einer Zeichenkette von rechts
`Mid`	liefert eine bestimmte Anzahl von Zeichen einer Zeichenkette aus der Mitte
`InStr`	sucht eine Zeichenkette innerhalb einer Zeichenkette

Funktion	Beschreibung
`Date`	liefert Datum als Zeichenkette
`Time`	liefert Zeit als Zeichenkette
`Now`	liefert aktuelles Datum und Uhrzeit als Zeichenkette
`LBound`	liefert untersten Indexwert eines Arrays
`UBound`	liefert obersten Indexwert eines Arrays
`GetDim`	liefert die Dimension eines Arrays
`Sleep`	wartet bestimmte Zeit
`Randomize`	initialisiert Zufallsfunktion
`Rnd`	liefert Zufallswert
`Int`	liefert ganzzahligen Anteil einer Zahl
`MsgBox`	zeigt Messagebox
`InputBox`	liefert Zeichenkette aus Eingabedialog
`Import`	importiert Standardklassen Module ins **NextView®4 Script**
`Include`	fügt beim Kompilieren das angegebene Skript ins **NextView®4 Script** ein
`System`	führt einen Systembefehl aus und liefert den Status der Befehlsausführung
`Quote`	setzt eine Zeichenkette in Anführungszeichen
`RGBColor`	gibt einen RGB-Farbwert zurück
`IsNumber`	testet, ob eine Zeichenkette eine Zahl ist und konvertiert diese ggf. in einen Double-Typ
`LocalTime`	liefert die Ortszeit im Windows® Datum/Uhrzeit Format
`SystemTime`	liefert die Systemzeit im Windows® Datum/Uhrzeit Format
`SystemToLocalTime`	konvertiert die Systemzeit in Ortszeit
`LocalToSystemTime`	konvertiert die Ortszeit in Systemzeit
`GetDate`	Datum lesen
`GetTime`	Uhrzeit lesen
`DateValue`	konvertiert ein Datum als Zeichenkette ins Windows® Datumsformat
`TimeValue`	konvertiert eine Uhrzeit als Zeichenkette ins Windows® Uhrzeitformat

Funktion	Beschreibung
DateTimeValue	konvertiert ein Datum mit Uhrzeitangabe als Zeichenkette ins Windows® Datum/Uhrzeit Format
GetDateFormat	konvertiert eine Windows® Datumsangabe in ein Datum als Zeichenkette optional mit Angabe bestimmter Datumsformate
GetTimeFormat	konvertiert eine Windows® Uhrzeitangabe in eine Uhrzeit als Zeichenkette optional mit Angabe bestimmter Uhrzeitformate
DateSerial	konvertiert das angegebene Jahr, Monat, Tag in eine Windows® Datumsangabe
TimeSerial	konvertiert die angegebene Stunde, Minute, Sekunde in eine Windows® Datumsangabe
ExtractDate	extrahiert Jahr, Monat, Tag aus einer Windows® Datumsangabe
ExtractTime	extrahiert Stunde, Minute, Sekunde, Millisekunde aus einer Windows® Datumsangabe
GetYear	liefert das Jahr aus einer Datum-Uhrzeit-Angabe
GetMonth	liefert den Monat aus einer Datum-Uhrzeit-Angabe
GetDay	liefert den Tag aus einer Datum-Uhrzeit-Angabe
GetHour	liefert die Stunde aus einer Datum-Uhrzeit-Angabe
GetMinute	liefert die Minute aus einer Datum-Uhrzeit-Angabe
GetSecond	liefert die Sekunde aus einer Datum-Uhrzeit-Angabe
GetMillisecond	liefert die Millisekunde aus einer Datum-Uhrzeit-Angabe
GDN	konvertiert Tag, Monat, Jahr in die Anzahl von Tagen im Gregorianischen Kalender
IsNaN	überprüft die Gültigkeit des Messwerts

4.1.2 Print

```
Sub Print (str As String)
```

Gibt die Zeichenkette **`str`** an der Standardausgabe aus. Die Standardausgabe in **NextView®4 Script** ist die Nachrichtenanzeige, die in **NextView®4** über den Menüpunkt "Anzeige" auf einem Blatt eingefügt werden kann.

Druckanweisungen eines Skriptes sind erst nach Beendigung des Skriptes in der Nachrichtenanzeige sichtbar.

Eine Auflistung aller verfügbaren Standard Funktionen und Prozeduren befindet sich im Kapitel "Standardfunktionen und Befehle: Überblick", S. 150.

4.1.3 Timer

```
Function Timer () As Integer
```

Liefert die Anzahl der seit Mitternacht in Coordinated Universal Time (UTC) vergangenen Sekunden.

Eine Auflistung aller verfügbaren Standard Funktionen und Prozeduren befindet sich im Kapitel "Standardfunktionen und Befehle: Überblick", S. 150.

4.1.4 TickCount

```
Function TickCount () As Integer
```

Liefert die Anzahl der seit dem Start von Windows® vergangenen Millisekunden.

Eine Auflistung aller verfügbaren Standard Funktionen und Prozeduren befindet sich im Kapitel "Standardfunktionen und Befehle: Überblick", S. 150.

4.1.5 TimestampStr

```
Function TimeStampStr (time as Double) As String
```

Konvertiert die übergebene Zeit beim Start der Aufzeichnung eines Signals **`time`** in eine Zeichenkette mit folgendem Format: **`DD.MM.YYYY hh:mm:ss.sss`**.

Eine Auflistung aller verfügbaren Standard Funktionen und Prozeduren befindet sich im Kapitel "Standardfunktionen und Befehle: Überblick", S. 150.

4.1.6 Format

```
Function Format (x As Double, fmt As String) As String
```

Liefert die Zahl **`x`** als formatierte Zeichenkette entsprechend den Anweisungen in **`fmt`**.

Dabei sind in **`fmt`** folgende Zeichen speziell:

Zeichen	Beschreibung
`#`	Platzhalter, der durch ein Leerzeichen ersetzt wird, falls die Zahl nicht groß genug ist
`0`	Platzhalter, der durch eine 0 ersetzt wird, falls die Zahl nicht groß genug ist
`,`	Platzhalter, der durch das landesspezifische Tausendertrennzeichen ersetzt wird
`.`	gibt den Start des Nachkommateils an und wird durch das landesspezifische Trennzeichen ersetzt
`+` oder **`-`**	ein vorangestelltes + oder - erzwingt die Ausgabe des Vorzeichens

```
Dim d As Double

  d = 1234.1234

  Print Format (d, "0.00")     ' gibt "1234.12" aus
  Print Format (d, "0.0#")     ' gibt "1234.12" aus
  Print Format (d, "0.##")     ' gibt "1234.12" aus

  Print Format (d, "+000000")  ' gibt "+001234" aus
  Print Format (d, "X######")  ' gibt "X  1234" aus

  Print Format (d, "0,000.00") ' gibt "1,234.00" aus
  Print Format (d, "##test##") ' gibt "12test34" aus

  Print Format (1000, "#,###") ' gibt "1,000" aus
  Print Format (1, "#,###")    ' gibt "    1" aus

  Print Format (d, "-000000")  ' gibt "+001234" aus
  Print Format (d, "X######")  ' gibt "X  1234" aus
```

Eine Auflistung aller verfügbaren Standard Funktionen und Prozeduren befindet sich im Kapitel "Standardfunktionen und Befehle: Überblick", S. 150.

4.1.7 Hex

```
Function Hex (x As Double) As String
```

Liefert die Zahl **x** als Zeichenkette im Hexformat zurück.

Eine Auflistung aller verfügbaren Standard Funktionen und Prozeduren befindet sich im Kapitel "Standardfunktionen und Befehle: Überblick", S. 150.

4.1.8 LCase

```
Function LCase (str As String) As String
```

Liefert die Zeichenkette **str** in Kleinbuchstaben.

Eine Auflistung aller verfügbaren Standard Funktionen und Prozeduren befindet sich im Kapitel "Standardfunktionen und Befehle: Überblick", S. 150.

4.1.9 UCase

```
Function UCase (str As String) As String
```

Liefert die Zeichenkette **str** in Großbuchstaben.

Eine Auflistung aller verfügbaren Standard Funktionen und Prozeduren befindet sich im Kapitel "Standardfunktionen und Befehle: Überblick", S. 150.

4.1.10 Asc

```
Function Asc (str As String) As Integer
```

Liefert den ASCII-Code des ersten Zeichens in der Zeichenkette **str**.

Eine Auflistung aller verfügbaren Standard Funktionen und Prozeduren befindet sich im Kapitel "Standardfunktionen und Befehle: Überblick", S. 150.

4.1.11 Chr

```
Function Chr (asc As Integer) As String
```

Liefert den ASCII-Code **asc** als Zeichen.

Eine Auflistung aller verfügbaren Standard Funktionen und Prozeduren befindet sich im Kapitel "Standardfunktionen und Befehle: Überblick", S. 150.

4.1.12 Tab

```
Function Tab (n As Integer) As String
```

Liefert eine Zeichenkette mit **n** Tabulatorzeichen.

Eine Auflistung aller verfügbaren Standard Funktionen und Prozeduren befindet sich im Kapitel "Standardfunktionen und Befehle: Überblick", S. 150.

4.1.13 Spc

```
Function Spc (n As Integer) As String
```

Liefert eine Zeichenkette mit **n** Leerzeichen.

Eine Auflistung aller verfügbaren Standard Funktionen und Prozeduren befindet sich im Kapitel "Standardfunktionen und Befehle: Überblick", S. 150.

4.1.14 Len

```
Function Len (str As String) As Integer
```

Liefert die Anzahl (Länge) der Zeichen in der Zeichenkette **`str`**.

Eine Auflistung aller verfügbaren Standard Funktionen und Prozeduren befindet sich im Kapitel "Standardfunktionen und Befehle: Überblick", S. 150.

4.1.15 Left

```
Function Left (str As String, n As Integer) As String
```

Liefert die ersten **n** Zeichen in der Zeichenkette **str**.

Eine Auflistung aller verfügbaren Standard Funktionen und Prozeduren befindet sich im Kapitel "Standardfunktionen und Befehle: Überblick", S. 150.

4.1.16 Right

```
Function Right (str As String, n As Integer) As String
```

Liefert die letzten **n** Zeichen in der Zeichenkette **str**.

Eine Auflistung aller verfügbaren Standard Funktionen und Prozeduren befindet sich im Kapitel "Standardfunktionen und Befehle: Überblick", S. 150.

4.1.17 Mid

```
Function Mid (str As String, pos As Integer, _
              n As Integer) As String
```

Liefert die folgenden **n** Zeichen ab der Stelle **pos** in der Zeichenkette **str**.

Eine Auflistung aller verfügbaren Standard Funktionen und Prozeduren befindet sich im Kapitel "Standardfunktionen und Befehle: Überblick", S. 150.

4.1.18 InStr

```
Function InStr (str as String, search as String) as Integer
```

Die Funktion **InStr** sucht in der Zeichenkette **str** nach dem ersten Auftreten des Zeichenkette **search**. Gross- und Kleinschreibung wird beachtet. Wurde **search** gefunden, wird die Position des ersten Zeichens in **str** zurückgegeben. Ist **search** nicht enthalten, wird 0 zurückgeliefert.

Eine Auflistung aller verfügbaren Standard Funktionen und Prozeduren befindet sich im Kapitel "Standardfunktionen und Befehle: Überblick", S. 150.

4.1.19 Date

```
Function Date () As String
```

Liefert das aktuelle Datum als Zeichenkette im landesspezifischen Format.

Eine Auflistung aller verfügbaren Standard Funktionen und Prozeduren befindet sich im Kapitel "Standardfunktionen und Befehle: Überblick", S. 150.

4.1.20 Time

```
Function Time () As String
```

Liefert die aktuelle Zeit als Zeichenkette im landesspezifischen Format.

Eine Auflistung aller verfügbaren Standard Funktionen und Prozeduren befindet sich im Kapitel "Standardfunktionen und Befehle: Überblick", S. 150.

4.1.21 Now

```
Function Now () As String
```

Liefert das aktuelle Datum und die aktuelle Zeit als Zeichenkette im landesspezifischen Format.

Eine Auflistung aller verfügbaren Standard Funktionen und Prozeduren befindet sich im Kapitel "Standardfunktionen und Befehle: Überblick", S. 150.

4.1.22 LBound

```
Function LBound (Array[, dim As Integer]) As Integer
```

Liefert die untere Indexgrenze der **dim**-ten Dimension eines Arrays.

```
Dim a(1 To 10) As Integer
Dim b(2 To 11,3 To 12) As Integer

Print LBound(b, 2)           ' gibt 3 aus
Print LBound(a)              ' gibt 1 aus
```

Eine Auflistung aller verfügbaren Standard Funktionen und Prozeduren befindet sich im Kapitel "Standardfunktionen und Befehle: Überblick", S. 150.

4.1.23 UBound

```
Function UBound (Array[, dim As Integer]) As Integer
```

Liefert die obere Indexgrenze der **dim**-ten Dimension eines Arrays.

```
Dim a(1 To 10) As Integer
Dim b(2 To 11,3 To 12) As Integer

Print UBound(b, 2)           ' gibt 12 aus
Print UBound(a)              ' gibt 10 aus
```

Eine Auflistung aller verfügbaren Standard Funktionen und Prozeduren befindet sich im Kapitel "Standardfunktionen und Befehle: Überblick", S. 150.

4.1.24 GetDim

```
Function GetDim (Array) As Integer
```

Liefert die Dimension eines Arrays.

```
Dim a(1 To 10) As Integer
Dim b(2 To 11,3 To 12) As Integer

Print GetDim(a)           ' gibt 1 aus
Print GetDim(b)           ' gibt 2 aus
```

Eine Auflistung aller verfügbaren Standard Funktionen und Prozeduren befindet sich im Kapitel "Standardfunktionen und Befehle: Überblick", S. 150.

4.1.25 Sleep

```
Sub Sleep (ms As Integer)
```

Lässt das Programm ungefähr **ms** Millisekunden warten.

Eine Auflistung aller verfügbaren Standard Funktionen und Prozeduren befindet sich im Kapitel "Standardfunktionen und Befehle: Überblick", S. 150.

4.1.26 Randomize

```
Sub Randomize (seed As Integer)
```

Initialisiert den Zufallsgenerator mit dem Startwert **seed**.

Eine Auflistung aller verfügbaren Standard Funktionen und Prozeduren befindet sich im Kapitel "Standardfunktionen und Befehle: Überblick", S. 150.

4.1.27 Rnd

```
Function Rnd () As Integer
```

Liefert den nächsten Wert des Zufallsgenerators (Wertebereich 0 .. 32767).

Eine Auflistung aller verfügbaren Standard Funktionen und Prozeduren befindet sich im Kapitel "Standardfunktionen und Befehle: Überblick", S. 150.

4.1.28 Int

```
Function Int (x As Double) As Integer
```

Liefert den ganzzahligen Anteil von **x** (s. "Floor", S. 180).

Eine Auflistung aller verfügbaren Standard Funktionen und Prozeduren befindet sich im Kapitel "Standardfunktionen und Befehle: Überblick", S. 150.

4.1.29 MsgBox

```
Function MsgBox (msg As String[, buttons As Integer, _
                 title As String]) As Integer
```

Öffnet eine Dialogbox mit Nachricht **msg**. Optional kann der Stil über **buttons** und der Titel über **title** gesetzt werden.

Buttons	Beschreibung
mbOkOnly	zeige nur "OK" Knopf
mbOKCancel	zeige "OK" und "Abbrechen" Knöpfe
mbAbortRetryIgnore	zeige "Abbrechen", "Wiederholen" und "Ignorieren" Knöpfe
mbYesNoCancel	zeige "Ja", "Nein" und "Abbrechen" Knöpfe
mbYesNo	zeige "Ja" und "Nein" Knöpfe
mbRetryCancel	zeige "Wiederholen" und "Abbrechen" Knöpfe

Als Rückgabewert liefert **`MsgBox`**:

MsgBox Rückgabe	Beschreibung
`mbOk`	Benutzer wählte "OK"
`mbCancel`	Benutzer wählte "Abbrechen"
`mbAbort`	Benutzer wählte "Abbrechen"
`mbRetry`	Benutzer wählte "Wiederholen"
`mbIgnore`	Benutzer wählte "Ignorieren"
`mbYes`	Benutzer wählte "Ja"
`mbNo`	Benutzer wählte "Nein"

Eine Auflistung aller verfügbaren Standard Funktionen und Prozeduren befindet sich im Kapitel "Standardfunktionen und Befehle: Überblick", S. 150.

4.1.30 InputBox

```
Function InputBox (prompt As String,  title As String, _
                   def As String) As String
```

Öffnet eine Eingabedialogbox mit der Aufforderung **`prompt`** und dem Title **`title`**. Als Standardwert wird **`def`** gesetzt.

Eine Auflistung aller verfügbaren Standard Funktionen und Prozeduren befindet sich im Kapitel "Standardfunktionen und Befehle: Überblick", S. 150.

4.1.31 Import

```
Import "Modul"
```

Einige Standardklassen müssen erst ins **NextView®4 Script** importiert werden, bevor ihre Prozeduren und Funktionen im Skript benutzt werden können.

Modul	Beschreibung	Import der Standardklassen
db	Database Modul	Standardklasse "Database", Standardklasse "Recordset"
os	Operating System Modul	Standardklasse "Stream", Standardklasse "Directory", Standardklasse "Clipboard"
xnf	Windows® INI File Modul	Standardklasse "XNF"

Eine Auflistung aller verfügbaren Standard Funktionen und Prozeduren befindet sich im Kapitel "Standardfunktionen und Befehle: Überblick", S. 150.

4.1.32 Include

```
Include "filename.nvs"
```

Mit diesem Befehl wird das Skript **filename.nvs** beim Kompilieren ins **NextView®4 Script** eingefügt.

Eine Auflistung aller verfügbaren Standard Funktionen und Prozeduren befindet sich im Kapitel "Standardfunktionen und Befehle: Überblick", S. 150.

4.1.33 System

```
Function System(cmd as String) as Integer
```

Führt den Systembefehl **cmd** aus und liefert den Status der Befehlsausführung zurück. Einige Befehlsparameter müssen eventuell mit der Funktion **Quote** extra mit Anführungszeichen ("") angegeben werden.

```
Dim rc as Integer
rc = system(quote("c:\program files\alarm.exe") & " on")
```

Eine Auflistung aller verfügbaren Standard Funktionen und Prozeduren befindet sich im Kapitel "Standardfunktionen und Befehle: Überblick", S. 150.

4.1.34 Quote

```
Function Quote (str as String) As String
```

Setzt eine Zeichenkette **str** in Anführungszeichen ("").

Eine Auflistung aller verfügbaren Standard Funktionen und Prozeduren befindet sich im Kapitel "Standardfunktionen und Befehle: Überblick", S. 150.

4.1.35 RGBColor

```
Function RGBColor (red as Integer, green as Integer, blue as
Integer) as Integer
```

Gibt den RGB-Farbwert mit den Farbbestandteilen rot (**red**), grün (**green**) und blau (**blue**) zurück. Diese haben jeweils einen Wertebereich von 0 bis 255, wobei 0 einem Farbanteil von 0% und 255 einem Farbanteil von 100% entspricht. Beispielsweise wird Weiss durch **RGBColor(255,255,255)** definiert und Schwarz durch **RGBColor(0,0,0)**.

Eine Auflistung aller verfügbaren Standard Funktionen und Prozeduren befindet sich im Kapitel "Standardfunktionen und Befehle: Überblick", S. 150.

4.1.36 IsNumber

```
Function IsNumber(number as String, dbl as Double) as
Boolean
```

Testet, ob der String **number** ein Zahl darstellt und konvertiert diese in die Variable **dbl**. Wird **False** zurückgegeben, stellt **number** keine Zahl dar.

Eine Auflistung aller verfügbaren Standard Funktionen und Prozeduren befindet sich im Kapitel "Standardfunktionen und Befehle: Überblick", S. 150.

4.1.37 LocalTime

```
Function LocalTime () as Double
```

Liefert die Ortszeit als Windows® Datum-Uhrzeit-Angabe.

Eine Auflistung aller verfügbaren Standard Funktionen und Prozeduren befindet sich im Kapitel "Standardfunktionen und Befehle: Überblick", S. 150.

4.1.38 SystemTime

```
Function SystemTime () as Double
```

Liefert die Systemzeit des PCs als Windows® Datum-Uhrzeit-Angabe.

Eine Auflistung aller verfügbaren Standard Funktionen und Prozeduren befindet sich im Kapitel "Standardfunktionen und Befehle: Überblick", S. 150.

4.1.39 SystemToLocalTime

```
Function SystemToLocalTime (vDate as Double) as Double
```

Konvertiert einen Datumswert von Systemzeit zu Ortszeit.

Eine Auflistung aller verfügbaren Standard Funktionen und Prozeduren befindet sich im Kapitel "Standardfunktionen und Befehle: Überblick", S. 150.

4.1.40 LocalToSystemTime

```
Function LocalToSystemTime (vDate as Double) as Double
```

Konvertiert einen Datumswert von Ortszeit zu Systemzeit.

Eine Auflistung aller verfügbaren Standard Funktionen und Prozeduren befindet sich im Kapitel "Standardfunktionen und Befehle: Überblick", S. 150.

4.1.41 GetDate

```
Function GetDate (datetime as Double) as Double
```

Liefert das Datum des **`datetime`**-Werts.

Eine Auflistung aller verfügbaren Standard Funktionen und Prozeduren befindet sich im Kapitel "Standardfunktionen und Befehle: Überblick", S. 150.

4.1.42 GetTime

```
Function GetTime (datetime as Double) as Double
```

Liefert die Uhrzeit des **`datetime`**-Werts.

Eine Auflistung aller verfügbaren Standard Funktionen und Prozeduren befindet sich im Kapitel "Standardfunktionen und Befehle: Überblick", S. 150.

4.1.43 DateValue

```
Function DateValue (date as String) as Double
```

Konvertiert eine Datumsangabe als Zeichenkette im eingestellten Benutzerformat in einen Windows® Datumswert (z. B. **`DateValue ("20.1.2008")`**). Der übergebene String muss im eingestellten Benutzerformat des PCs angegeben werden.

Eine Auflistung aller verfügbaren Standard Funktionen und Prozeduren befindet sich im Kapitel "Standardfunktionen und Befehle: Überblick", S. 150.

4.1.44 TimeValue

```
Function TimeValue (time as String) as Double
```

Konvertiert eine Uhrzeitangabe als Zeichenkette im eingestellten Benutzerformat in einen Windows® Zeitwert (z. B. **`TimeValue ("12:00:01")`**). Der übergebene String muss im eingestellten Benutzerformat des PCs angegeben werden.

Eine Auflistung aller verfügbaren Standard Funktionen und Prozeduren befindet sich im Kapitel "Standardfunktionen und Befehle: Überblick", S. 150.

4.1.45 DateTimeValue

```
Function DateTimeValue (date as String) as Double
```

Konvertiert eine Datum-Uhrzeit-Angabe als Zeichenkette im eingestellten Benutzerformat in einen Windows® Datumswert (z. B. **`DateTimeValue ("20.1.2008 12:00:01")`**). Der übergebene String muss im eingestellten Benutzerformat des PCs angegeben werden.

Eine Auflistung aller verfügbaren Standard Funktionen und Prozeduren befindet sich im Kapitel "Standardfunktionen und Befehle: Überblick", S. 150.

4.1.46 GetDateFormat

```
Function GetDateFormat (vDate as Double,
  Optional format as String) as String
```

Konvertiert eine Windows® Datumsangabe in ein Datum als Zeichenkette. Optional können weitere Zeichen mit den folgenden Format-Spezifikationen angegeben werden.

Parameter	Beschreibung
d	Tag des Monats als Ziffer ohne vorangestellte Null bei einstelligen Tagen
dd	Tag des Monats als Ziffer mit vorangestellter Null bei einstelligen Tagen
ddd	Tag der Woche abgekürzt; die Funktion verwendet den LOCALE_SABBREVDAYNAME Wert, der zum angegebenen Gebietsschema gehört
dddd	Tag der Woche voll ausgeschrieben; die Funktion verwendet den LOCALE_SDAYNAME Wert, der zum angegebenen Gebietsschema gehört
M	Monat als Ziffer ohne vorangestellte Null bei einstelligen Monaten
MM	Monat als Ziffer mit vorangestellter Null bei einstelligen Monaten
MMM	Monat abgekürzt; die Funktion verwendet den LOCALE_SABBREVMONTHNAME Wert, der zum angegebenen Gebietsschema gehört
MMMM	Monat voll ausgeschrieben; die Funktion verwendet den LOCALE_SMONTHNAME Wert, der zum angegebenen Gebietsschema gehört
y	zweistellige Jahreszahl ohne vorangestellte Null bei einstelligen Jahren
yy	zweistellige Jahreszahl mit vorangestellter Null bei einstelligen Jahren
yyyy	vierstellige Angabe des Jahres

Das folgende Beispiel weist der Variablen **str** das Datum "Do 31.12.2009" zu:

```
Dim str as String
str = GetDateFormat(40178,"ddd d.MM.yyyy")
```

Eine Auflistung aller verfügbaren Standard Funktionen und Prozeduren befindet sich im Kapitel "Standardfunktionen und Befehle: Überblick", S. 150.

4.1.47 GetTimeFormat

```
Function GetTimeFormat (vTime as Double,
  Optional format as String) as String
```

Konvertiert eine Windows® Uhrzeitangabe in eine Uhrzeitangabe als Zeichenkette. Optional können weitere Zeichen mit den folgenden Format-Spezifikationen angegeben werden.

Parameter	Beschreibung
h	Stunde ohne vorangestellte Null bei einstelligen Stunden, 12-Stunden-Anzeige
hh	Stunde mit vorangestellter Null bei einstelligen Stunden, 12-Stunden-Anzeige
H	Stunde ohne vorangestellte Null bei einstelligen Stunden, 24-Stunden-Anzeige
HH	Stunde mit vorangestellter Null bei einstelligen Stunden, 24-Stunden-Anzeige
m	Minute ohne vorangestellte Null bei einstelligen Minuten
mm	Minute mit vorangestellter Null bei einstelligen Minuten
s	Sekunde ohne vorangestellte Null bei einstelligen Sekunden
ss	Sekunde mit vorangestellter Null bei einstelligen Sekunden
t	ein Buchstabe als zusätzliche Zeitinformation, z. B. A oder P
tt	mehrere Buchstaben als zusätzliche Zeitinformation, z. B. AM oder PM

Millisekunden werden nicht ausgeben, es sei denn die Zeichenkette enthält ein **.sss**.

Das folgende Beispiel weist der Variablen **str** die Uhrzeit "12:17:42.720" zu:

```
Dim str as String
str = GetTimeFormat(0.5123,"hh:mm:ss.sss")
```

Eine Auflistung aller verfügbaren Standard Funktionen und Prozeduren befindet sich im Kapitel "Standardfunktionen und Befehle: Überblick", S. 150.

4.1.48 DateSerial

```
Function DateSerial (y as Integer, mm as Integer,
  dd as Integer) as Double
```

Konvertiert das übergebene Jahr **y**, den Monat **mm** und den Tag **dd** in eine Windows® Datumsangabe.

Eine Auflistung aller verfügbaren Standard Funktionen und Prozeduren befindet sich im Kapitel "Standardfunktionen und Befehle: Überblick", S. 150.

4.1.49 TimeSerial

```
Function TimeSerial (h as Integer, m as Integer,
  s as Integer, Optional ms as Integer = 0) as Double
```

Konvertiert die übergebene Stunde **h**, die Minute **m** und die Sekunde **s** in einen eine Windows® Uhrzeitangabe. Optional kann die Anzahl an Millisekunden mitübergeben werden.

Eine Auflistung aller verfügbaren Standard Funktionen und Prozeduren befindet sich im Kapitel "Standardfunktionen und Befehle: Überblick", S. 150.

4.1.50 ExtractDate

```
Sub ExtractDate (vDate as Double, Byref y as Integer,
  Byref m as Integer, Byref d as Integer)
```

Extrahiert das Jahr **y**, den Monat **mm** und den Tag **dd** aus der übergebenen Windows® Datumsangabe **vDate**.

Eine Auflistung aller verfügbaren Standard Funktionen und Prozeduren befindet sich im Kapitel "Standardfunktionen und Befehle: Überblick", S. 150.

4.1.51 ExtractTime

```
Sub ExtractTime (vDate as Double, Byref h as Integer, Byref
m as Integer, Byref s as Integer, Byref ms as Integer)
```

Extrahiert die Stunde **h**, die Minute **m**, die Sekunde **s** und die Millisekunde **ms** aus der übergebenen Windows® Uhrzeitangabe **vDate**.

Eine Auflistung aller verfügbaren Standard Funktionen und Prozeduren befindet sich im Kapitel "Standardfunktionen und Befehle: Überblick", S. 150.

4.1.52 GetYear

```
Function GetYear (vDate as Double) as Integer
```

Liefert das Jahr aus der übergebenen Datum-Uhrzeit-Angabe **vDate**.

Eine Auflistung aller verfügbaren Standard Funktionen und Prozeduren befindet sich im Kapitel "Standardfunktionen und Befehle: Überblick", S. 150.

4.1.53 GetMonth

```
Function GetMonth (vDate as Double) as Integer
```

Liefert den Monat aus der übergebenen Datum-Uhrzeit-Angabe **vDate**.

Eine Auflistung aller verfügbaren Standard Funktionen und Prozeduren befindet sich im Kapitel "Standardfunktionen und Befehle: Überblick", S. 150.

4.1.54 GetDay

```
Function GetDay (vDate as Double) as Integer
```

Liefert den Tag aus der übergebenen Datum-Uhrzeit-Angabe **vDate**.

Eine Auflistung aller verfügbaren Standard Funktionen und Prozeduren befindet sich im Kapitel "Standardfunktionen und Befehle: Überblick", S. 150.

4.1.55 GetHour

```
Function GetHour (vDate as Double) as Integer
```

Liefert die Stunde aus der übergebenen Datum-Uhrzeit-Angabe **vDate**.

Eine Auflistung aller verfügbaren Standard Funktionen und Prozeduren befindet sich im Kapitel "Standardfunktionen und Befehle: Überblick", S. 150.

4.1.56 GetMinute

```
Function GetMinute (vDate as Double) as Integer
```

Liefert die Minute aus der übergebenen Datum-Uhrzeit-Angabe **vDate**.

Eine Auflistung aller verfügbaren Standard Funktionen und Prozeduren befindet sich im Kapitel "Standardfunktionen und Befehle: Überblick", S. 150.

4.1.57 GetSecond

```
Function GetSecond (vDate as Double) as Integer
```

Liefert die Sekunde aus der übergebenen Datum-Uhrzeit-Angabe **vDate**.

Eine Auflistung aller verfügbaren Standard Funktionen und Prozeduren befindet sich im Kapitel "Standardfunktionen und Befehle: Überblick", S. 150.

4.1.58 GetMillisecond

```
Function GetMillisecond (vDate as Double) as Integer
```

Liefert die Millisekunde aus der übergebenen Datum-Uhrzeit-Angabe **vDate**.

Eine Auflistung aller verfügbaren Standard Funktionen und Prozeduren befindet sich im Kapitel "Standardfunktionen und Befehle: Überblick", S. 150.

4.1.59 GDN

```
Function GDN (d as Integer, m as Integer, y as Integer)
  as Integer
```

Konvertiert den Tag **d**, den Monat **m** und das Jahr **y** in die entsprechende Anzahl an Tagen im Gregorianischen Kalender.

Eine Auflistung aller verfügbaren Standard Funktionen und Prozeduren befindet sich im Kapitel "Standardfunktionen und Befehle: Überblick", S. 150.

4.1.60 IsNaN

```
Function IsNaN (double val) as Boolean
```

Überprüft, ob ein Messwert **val** einen gültigen Wert enthält.

Eine Auflistung aller verfügbaren Standard Funktionen und Prozeduren befindet sich im Kapitel "Standardfunktionen und Befehle: Überblick", S. 150.

4.2 Mathematische Funktionen

4.2.1 Mathematische Funktionen: Überblick

Funktion	Beschreibung
`Sin`	Sinus Funktion
`Cos`	Kosinus Funktion
`Tan`	Tangens Funktion
`Sinh`	Sinus Hyperbolicus Funktion
`Cosh`	Kosinus Hyperbolicus Funktion
`Tanh`	Tangens Hyperbolicus Funktion
`Asin`	Arcussinus Funktion
`Acos`	Arcuskosinus Funktion
`Atan`	Arcustangens Funktion
`Sqrt`	Wurzelfunktion
`Sqr`	Quadratfunktion
`Log`	Logarithmusfunktion (natürlicher Logarithmus)
`Exp`	Exponentialfunktion
`Round`	Rundungsfunktion
`Floor`	Abrundungsfunktion
`Ceil`	Aufrundungsfunktion
`Abs`	Absolutfunktion
`Pow`	Potenzfunktion
`Fmod`	Fließkomma Modulofunktion

4.2.2 Sin

```
Function Sin (angle As Double) As Double
```

Gibt den Sinus von **angle** zurück. Dabei muss **angle** im Bogenmaß angegeben werden.

Eine Auflistung aller zur Verfügung stehenden mathematischen Funktionen befindet sich im Kapitel "Mathematische Funktionen: Überblick", S. 175.

4.2.3 Cos

```
Function Cos (angle As Double) As Double
```

Gibt den Kosinus von **angle** zurück. Dabei muss **angle** im Bogenmaß angegeben werden.

Eine Auflistung aller zur Verfügung stehenden mathematischen Funktionen befindet sich im Kapitel "Mathematische Funktionen: Überblick", S. 175.

4.2.4 Tan

```
Function Tan (angle As Double) As Double
```

Gibt den Tangens von **angle** zurück. Dabei muss **angle** im Bogenmaß angegeben werden.

Eine Auflistung aller zur Verfügung stehenden mathematischen Funktionen befindet sich im Kapitel "Mathematische Funktionen: Überblick", S. 175.

4.2.5 Sinh

```
Function Sinh (angle As Double) As Double
```

Gibt den hyperbolischen Sinus von **angle** zurück. Dabei muss **angle** im Bogenmaß angegeben werden.

Eine Auflistung aller zur Verfügung stehenden mathematischen Funktionen befindet sich im Kapitel "Mathematische Funktionen: Überblick", S. 175.

4.2.6 Cosh

```
Function Cosh (angle As Double) As Double
```

Gibt den hyperbolischen Kosinus von **angle** zurück. Dabei muss **angle** im Bogenmaß angegeben werden.

Eine Auflistung aller zur Verfügung stehenden mathematischen Funktionen befindet sich im Kapitel "Mathematische Funktionen: Überblick", S. 175.

4.2.7 Tanh

```
Function Tanh (angle As Double) As Double
```

Gibt den hyperbolischen Tangens von **angle** zurück. Dabei muss **angle** im Bogenmaß angegeben werden.

Eine Auflistung aller zur Verfügung stehenden mathematischen Funktionen befindet sich im Kapitel "Mathematische Funktionen: Überblick", S. 175.

4.2.8 Asin

```
Function Asin (x As Double) As Double
```

Gibt den Arcussinus von **x** im Bogenmaß zwischen –π/2 und π/2 zurück. Dabei muss **x** zwischen –1 und 1 liegen, sonst liefert **Asin** "NaN" (keine Zahl) zurück.

Eine Auflistung aller zur Verfügung stehenden mathematischen Funktionen befindet sich im Kapitel "Mathematische Funktionen: Überblick", S. 175.

4.2.9 Acos

```
Function Acos (x As Double) As Double
```

Gibt den Arcuskosinus von **x** im Bogenmaß zwischen –π/2 und π/2 zurück. Dabei muss **x** zwischen –1 und 1 liegen, sonst liefert **Acos** "NaN" (keine Zahl) zurück.

Eine Auflistung aller zur Verfügung stehenden mathematischen Funktionen befindet sich im Kapitel "Mathematische Funktionen: Überblick", S. 175.

4.2.10 Atan

```
Function Atan (x As Double) As Double
```

Gibt den Arcustangens von **x** im Bogenmaß zwischen –π/2 und π/2 zurück. Falls **x** gleich Null ist liefert **Atan** "NaN" (keine Zahl) zurück.

Eine Auflistung aller zur Verfügung stehenden mathematischen Funktionen befindet sich im Kapitel "Mathematische Funktionen: Überblick", S. 175.

4.2.11 Sqrt

```
Function Sqrt (x As Double) As Double
```

Gibt die Quadratwurzel (*square root*) von **x** zurück. Falls **x** kleiner als Null ist liefert **Sqrt** "NaN" (keine Zahl) zurück.

Eine Auflistung aller zur Verfügung stehenden mathematischen Funktionen befindet sich im Kapitel "Mathematische Funktionen: Überblick", S. 175.

4.2.12 Sqr

```
Function Sqr (x As Double) As Double
```

Gibt das Quadrat (*square*) von **x** zurück.

Eine Auflistung aller zur Verfügung stehenden mathematischen Funktionen befindet sich im Kapitel "Mathematische Funktionen: Überblick", S. 175.

4.2.13 Log

```
Function Log (x As Double) As Double
```

Gibt den natürlichen Logarithmus von **x** zurück. Falls **x** kleiner Null ist liefert **Log** "NaN" (keine Zahl) zurück, bei **x** gleich Null "INF" (unendlich).

Eine Auflistung aller zur Verfügung stehenden mathematischen Funktionen befindet sich im Kapitel "Mathematische Funktionen: Überblick", S. 175.

4.2.14 Exp

```
Function Exp (x As Double) As Double
```

Gibt den Wert der Exponentialfunktion von **x** zurück.

Eine Auflistung aller zur Verfügung stehenden mathematischen Funktionen befindet sich im Kapitel "Mathematische Funktionen: Überblick", S. 175.

4.2.15 Round

```
Function Round (x As Double) As Double
```

Gibt den gerundeten, ganzzahligen Wert von **x** zurück.

Eine Auflistung aller zur Verfügung stehenden mathematischen Funktionen befindet sich im Kapitel "Mathematische Funktionen: Überblick", S. 175.

4.2.16 Floor

```
Function Floor (x As Double) As Double
```

Gibt den abgerundeten, ganzzahligen Wert von **x** zurück.

Eine Auflistung aller zur Verfügung stehenden mathematischen Funktionen befindet sich im Kapitel "Mathematische Funktionen: Überblick", S. 175.

4.2.17 Ceil

```
Function Ceil (x As Double) As Double
```

Gibt den aufgerundeten, ganzzahligen Wert von **x** zurück.

Eine Auflistung aller zur Verfügung stehenden mathematischen Funktionen befindet sich im Kapitel "Mathematische Funktionen: Überblick", S. 175.

4.2.18 Abs

```
Function Abs (angle As Double) As Double
```

Gibt den Absolutbetrag von **x** zurück.

Eine Auflistung aller zur Verfügung stehenden mathematischen Funktionen befindet sich im Kapitel "Mathematische Funktionen: Überblick", S. 175.

4.2.19 Pow

```
Function Pow (x As Double, y As Double) As Double
```

Gibt Wert von **x** hoch **y** zurück.

Eine Auflistung aller zur Verfügung stehenden mathematischen Funktionen befindet sich im Kapitel "Mathematische Funktionen: Überblick", S. 175.

4.2.20 Fmod

```
Function Fmod (x As Double, y As Double) As Double
```

Gibt den Fließkommarest von **x / y** zurück, so dass **x = i * y + fmod(x,y)** und **i** die größte ganze Zahl ist, mit **x** größer als **i * y**. Beispielsweise ist **fmod(9.8,5.5) = 4.3**.

Eine Auflistung aller zur Verfügung stehenden mathematischen Funktionen befindet sich im Kapitel "Mathematische Funktionen: Überblick", S. 175.

4.3 Standardklasse "Database"

Die Standardklasse "Database" erlaubt den Zugriff auf vorhandene Datenbanken des Systems. Ein Objekt der Klasse "Database**"** kann über den **`New`** Befehl angelegt werden.

Um die Standardklasse "Database" in NextView®4 Script zur Verfügung zu stellen, muss man am Anfang des Scripts das Modul "Database" (db) importieren (s. "Import", S. 163).

```
' Database Support am Anfang des NextView Scripts importieren
Import "db"
...

Sub Test ()
Dim db As Database

  Set db = New Database

  if db.Open ("DSN=myDSN") <> 0 then
    ...
  End If
End Sub
```

4.3.1 Database: Überblick

Elementfunktion	Beschreibung
`Database.Open`	öffnet eine Datenbank
`Database.Close`	schließt eine geöffnete Datenbank
`Database.Execute`	führt einen SQL Befehl aus
`Database.OpenRecordset`	führt einen SQL Befehl aus und liefert das dazugehörige Recordset zurück

4.3.2 Database.Open

```
Function Open (DSN As String) As Integer
```

Öffnet die Datenbank **DSN** (*Data Source Name*). Kann die Datenbank nicht geöffnet werden, liefert **Open()** den Wert 0 zurück.

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Standardklasse "Database" befindet sich im Kapitel "Database: Überblick", S. 182.

4.3.3 Database.Close

```
Sub Close ()
```

Schließt eine geöffnete Datenbank.

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Standardklasse "Database" befindet sich im Kapitel "Database: Überblick", S. 182.

4.3.4 Database.Execute

```
Function Execute (sqlQuery As String) As RecordSet
```

Führt die SQL Anfrage **sqlQuery** aus. Bei einem Fehler löst **Execute()** einen Laufzeitfehler aus.

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Standardklasse "Database" befindet sich im Kapitel "Database: Überblick", S. 182.

4.3.5 Database.OpenRecordset

```
Function OpenRecordset (sqlQuery As String, Optional _
                        dbType As Integer) As Recordset
```

Führt die SQL Anfrage **sqlQuery** aus und gibt das Ergebnis als **Recordset** zurück (s. "Standardklasse "Recordset", S. 185). Bei einem Fehler löst **OpenRecordset()** einen Laufzeitfehler aus.

Wird **dbType** nicht gesetzt, versucht **OpenRecordset()** automatisch einen Cursortyp einzustellen. Beachten Sie bitte, dass die erlaubten Werte bei der Wahl eines Cursors von Ihren Datenbankeinstellungen und der Datenbank, die Sie benutzen, abhängen.

dbType	Bedeutung
dbNone	Setzt keinen Cursortyp, der Standardcursor der Datenbank wird ausgewählt.
dbDynaset	Setzt den Cursortyp auf "dynamisch", d. h. man kann sich im Recordset beliebig bewegen. (Diese Option muss in der Datenbank möglich sein, sonst schlägt **OpenRecordset** fehl!)
dbForwardOnly	Setzt den Cursortyp auf "forward-only". Das bedeutet, man kann den Cursor nur mit **Recordset.MoveNext** nach vorne setzen.

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Standardklasse "Database" befindet sich im Kapitel "Database: Überblick", S. 182.

4.4 Standardklasse "Recordset"

Die Standardklasse "Recordset" erlaubt den Zugriff auf Daten in einer Datenbank. Ein Objekt der Klasse "Recordset" kann nicht über den **`New`** Befehl angelegt werden. Es wird stattdessen von **`Database.OpenRecordset`** der Standardklasse "Database" zurückgegeben.

- **Um die Standardklasse "Database" in NextView®4 Script zur Verfügung zu stellen, muss man am Anfang des Scripts das Modul "Database" (db) importieren (s. "Import", S. 163).**
- **Alle geöffneten Recordsets müssen vor dem Schließen oder erneutem Öffnen einer Datenbank unbedingt geschlossen werden.**

```
' Database Support am Anfang des NextView Scripts importieren.
Import "db"
...

Sub Test ()
  Dim i As Integer
  Dim db As Database

  Set db = New Database

  if db.Open ("DSN=myDSN") <> 0 then

    Dim rs As Recordset
    Set rs = db.OpenRecordset ("SELECT * FROM Serial_Table")

    For i = 1 To rs.Columns
      Print rs.ColumnName (i)
    Next
' alle Recordset vor erneutem Öffnen der Datenbank wieder freigeben
    set rs = Nothing

  Else
    Print "not opened"
  End if
End Sub
```

4.4.1 Recordset: Überblick

Elementfunktion	Beschreibung
`Recordset.Columns`	liefert die Anzahl der Spalten
`Recordset.ColumnName`	liefert den Namen einer Spalte
`Recordset.IsBof`	gibt an, ob der Cursor in der ersten Zeile ist
`Recordset.IsEof`	gibt an, ob der Cursor in der letzten Zeile ist
`Recordset.AddNew`	legt eine neue Zeile an
`Recordset.Update`	schreibt Änderungen an den Datenbankserver
`Recordset.Edit`	macht eine Zeile editierbar
`Recordset.MoveFirst`	bewegt den Cursor in die erste Zeile
`Recordset.MoveNext`	bewegt den Cursor in die letzte Zeile
`Recordset.Value`	setzt bzw. liefert den Spaltenwert der aktuellen Zeile

4.4.2 Recordset.Columns

```
Function Columns () As Integer
```

Die Funktion **`Columns()`** liefert die Anzahl der Spalten im Recordset.

Eine Auflistung aller verfügbaren Elementfunktionen der Standardklasse "Recordset" befindet sich im Kapitel "Recordset: Überblick", S. 186.

4.4.3 Recordset.ColumnName

```
Sub ColumnName (index As Integer)
```

Die Funktion **`Columnname()`** liefert den Namen der Spalte Nummer **`index`** im Recordset zurück.

```
...
Set rs = db.OpenRecordset (...)

For i = 1 To rs.Columns
  Print rs.ColumnName (i)
Next
...
' alle Recordset vor erneutem Öffnen der Datenbank wieder freigeben
    set rs = Nothing
...
```

Eine Auflistung aller verfügbaren Elementfunktionen der Standardklasse "Recordset" befindet sich im Kapitel "Recordset: Überblick", S. 186.

4.4.4 Recordset.IsBof

```
Function IsBof () As Boolean
```

Die Funktion **IsBof()** gibt an, ob der Cursor im Recordset in der ersten Zeile (*Beginning of File*) steht.

Eine Auflistung aller verfügbaren Elementfunktionen der Standardklasse "Recordset" befindet sich im Kapitel "Recordset: Überblick", S. 186.

4.4.5 Recordset.IsEof

```
Function IsEof () As Boolean
```

Die Funktion **IsEof()** gibt an, ob der Cursor im Recordset in der letzen Zeile (*End of File*) steht.

Eine Auflistung aller verfügbaren Elementfunktionen der Standardklasse "Recordset" befindet sich im Kapitel "Recordset: Überblick", S. 186.

4.4.6 Recordset.AddNew

```
Sub AddNew ()
```

Die Funktion **AddNew()** fügt dem Recordset eine neue Zeile am Ende hinzu. Die Änderung in der Datenbank wird erst nach dem Aufruf von **Update** (s. "**Recordset.Update**", S. 188) übernommen.

```
...
Dim rs As Recordset
Set rs = db.OpenRecordset ("SELECT FROM TestTable", dbDynaset)
...
rs.AddNew
rs.Value(rs.ColumnName (1)) = "xxx"
rs.Update
...
' alle Recordset vor erneutem Öffnen der Datenbank wieder freigeben
    set rs = Nothing
...
```

Eine Auflistung aller verfügbaren Elementfunktionen der Standardklasse "Recordset" befindet sich im Kapitel "Recordset: Überblick", S. 186.

4.4.7 Recordset.Update

```
Sub Update ()
```

Die Funktion **Update()** schreibt alle am Recordset vorgenommenen Änderungen an den Datenbankserver zurück.

Eine Auflistung aller verfügbaren Elementfunktionen der Standardklasse "Recordset" befindet sich im Kapitel "Recordset: Überblick", S. 186.

4.4.8 Recordset.Edit

```
Sub Edit ()
```

Die Funktion **Edit()** macht die Zeile im Recordset, in der der Cursor steht editierbar.

Eine Auflistung aller verfügbaren Elementfunktionen der Standardklasse "Recordset" befindet sich im Kapitel "Recordset: Überblick", S. 186.

4.4.9 Recordset.MoveFirst

```
Sub MoveFirst ()
```

Die Funktion **MoveFirst()** setzt den Cursor auf die erste Zeile im Recordset. Befinden sich im Recordset keine Daten, liefert danach **Recordset.IsEof()** "True" zurück.

Eine Auflistung aller verfügbaren Elementfunktionen der Standardklasse "Recordset" befindet sich im Kapitel "Recordset: Überblick", S. 186.

4.4.10 Recordset.MoveNext

```
Sub MoveNext ()
```

Die Funktion **MoveNext()** setzt den Cursor auf die nächste Zeile im Recordset. Liefert danach **Recordset.IsEof()** vor dem Aufruf von **MoveNext()** "True" zurück, so erzeugt **MoveNext()** einen Laufzeitfehler.

Eine Auflistung aller verfügbaren Elementfunktionen der Standardklasse "Recordset" befindet sich im Kapitel "Recordset: Überblick", S. 186.

4.4.11 Recordset.Value

```
'Wert lesen:
  Function Value (columnName As String) As String

'neuen Wert setzen:
  Value (columnName) = newValue  ' newValue als String
```

`Value()` liefert bzw. setzt in der aktuellen Zeile den Wert in der Spalte mit dem Namen **`columnName`**.

```
...
if db.Open ("DSN = DBNVTest") > 0 then

  Dim rs As Recordset
  Set rs = db.OpenRecordset ("SELECT FROM TestTable")
...
  rs.AddNew
  rs.Value(rs.ColumnName (1)) = "new product"
  rs.Update
print rs.Value(rs.ColumnName(1))
...
```

Eine Auflistung aller verfügbaren Elementfunktionen der Standardklasse "Recordset" befindet sich im Kapitel "Recordset: Überblick", S. 186.

4.5 Standardklasse "Directory"

Für den Zugriff auf Verzeichnisse und Dateien des Betriebsystems (Operating System) gibt es in **NextView®4 Script** die Standardklasse "Directory". Ein Objekt der Klasse "Directory" wird über den **New** Befehl angelegt.

- **Um die Standardklasse "Directory" in NextView®4 Script zur Verfügung zu stellen, muss man am Anfang des NextView®4 Scripts das Modul Operating System (OS) importieren (s. "Import", S. 163).**
- **Zur Ausführung der Routinen der Klasse "Directory" muss der Benutzer Schreib- und/oder Leserechte in den Verzeichnissen besitzen und die zu bearbeitenden Dateien dürfen nicht von einem Programm geöffnet sein.**

Das folgende Beispiel holt die Dateinamen der vorhandenen Dateien in dem Verzeichnis **c:\Programme\Nextview 4.4**:

```
' Operating System Support importieren.
Import "os"
...

Sub Test ()
  Dim dir as Directory
  Dim str, names(40) as String
  Dim i as Integer

  set dir = new Directory
  i = 0
  names(i) = dir.FindFirst("c:\Programme\NextView 4.4\*.*")
  do
    str = dir.FindNext
    if str <> "" then
     i= i+1
     names(i) = str
    End If
  Loop until (str = "" Or i >= ubound(names))
  dir.FindClose
  ...
End Sub
```

4.5.1 Directory: Überblick

Elementfunktion	Beschreibung
`Directory.FindFirst`	liefert die erste Datei im angegebenen Verzeichnispfad
`Directory.FindNext`	liefert die nächste Datei
`Directory.FindClose`	beendet die Dateisuche
`Directory.MakeDir`	legt ein Verzeichnis an
`Directory.RemoveDir`	löscht ein leeres Verzeichnis
`Directory.CopyFile`	kopiert eine vorhandene Datei
`Directory.MoveFile`	verschiebt eine vorhandene Datei oder Verzeichnis
`Directory.DeleteFile`	löscht eine vorhandene Datei

4.5.2 Directory.FindFirst

```
Function FindFirst (path as String) As String
```

Die Funktion **`FindFirst`** liefert den ersten Dateinamen im angegeben Verzeichnispfad **`path`**. Der Pfad **`path`** kann Platzhalter, wie **`*`** oder **`?`** enthalten.

```
Function FileExists (name as String) as Boolean
  Dim str as String
  Dim dir as Directory
  set dir = new Directory
  str = dir.FindFirst(name)
  if str <> "" then
    FileExists = True
  else
    FileExists = False
  end if
  dir.FindClose
end Function
```

Eine Auflistung aller verfügbaren Elementfunktionen der Standardklasse "Directory" befindet sich im Kapitel "Directory: Überblick", S. 192.

4.5.3 Directory.FindNext

```
Function FindNext () As String
```

Die Funktion **FindNext** liefert den nächsten Dateinamen im Verzeichnispfad, der im Befehl "**Directory.FindFirst**" angeben wurde.

Eine Auflistung aller verfügbaren Elementfunktionen der Standardklasse "Directory" befindet sich im Kapitel "Directory: Überblick", S. 192.

4.5.4 Directory.FindClose

```
Sub FindClose ()
```

Die Prozedur **FindClose** schliesst die Dateisuche von **Directory.FindFirst**.

Eine Auflistung aller verfügbaren Elementfunktionen der Standardklasse "Directory" befindet sich im Kapitel "Directory: Überblick", S. 192.

4.5.5 Directory.MakeDir

```
Function MakeDir (path as String) As Boolean
```

Die Funktion **MakeDir** legt ein Verzeichnis mit Namen **path** an und liefert **True** zurück, wenn das Erzeugen gelungen ist.

Eine Auflistung aller verfügbaren Elementfunktionen der Standardklasse "Directory" befindet sich im Kapitel "Directory: Überblick", S. 192.

4.5.6 Directory.RemoveDir

```
Function RemoveDir (path as String) As Boolean
```

Die Funktion **RemoveDir** löscht ein leeres Verzeichnis mit Namen **path** und liefert **True** zurück, wenn das Löschen erfolgreich war. Befindet sich noch eine Datei im Verzeichnis oder trat ein anderer Fehler auf, wird das Verzeichnis nicht gelöscht und **False** zurückgegeben.

Eine Auflistung aller verfügbaren Elementfunktionen der Standardklasse "Directory" befindet sich im Kapitel "Directory: Überblick", S. 192.

4.5.7 Directory.CopyFile

```
Function CopyFile (name as String, newname as String,
overwrite as Boolean) As Boolean
```

Die Funktion **CopyFile** kopiert eine vorhandene Datei **name** in eine neue Datei mit Namen **newname**. Ist **overwrite** gleich **True**, wird eine vorhandene Datei überschrieben.

Eine Auflistung aller verfügbaren Elementfunktionen der Standardklasse "Directory" befindet sich im Kapitel "Directory: Überblick", S. 192.

4.5.8 Directory.MoveFile

```
Function MoveFile (name as String, newname as String) As
Boolean
```

Die Funktion **MoveFile** verschiebt eine vorhandene Datei oder Verzeichnis **name** nach **newname**. Bei Verzeichnissen muss **newname** sich auf dem gleichen Laufwerk befinden. Weder die Datei oder das Verzeichnis darf existieren.

Eine Auflistung aller verfügbaren Elementfunktionen der Standardklasse "Directory" befindet sich im Kapitel "Directory: Überblick", S. 192.

4.5.9 Directory.DeleteFile

```
Function DeleteFile (name as String) As Boolean
```

Die Funktion **`DeleteFile`** löscht eine vorhandene Datei **`name`**.

Eine Auflistung aller verfügbaren Elementfunktionen der Standardklasse "Directory" befindet sich im Kapitel "Directory: Überblick", S. 192.

4.6 Standardklasse "Stream"

Für den Zugriff auf Textdateien und serielle Schnittstellen gibt es in **NextView®4 Script** die Standardklasse "Stream". Ein Objekt der Klasse "Stream" kann nicht über den **`New`** Befehl angelegt werden. Es wird stattdessen von der Funktion **`Stream.FileStream`** zurückgegeben.

Um die Standardklasse "Stream" in NextView®4 Script zur Verfügung zu stellen, muss man am Anfang des NextView®4 Scripts das Modul Operating System (OS) importieren (s. "Import", S. 163).

Das folgende Beispiel erzeugt eine Textdatei **`datafile.txt`**, hängt Daten an der Datei an, liest die Datei anschließend wieder ein und gibt deren Inhalt aus:

```
' Stream Support am Anfang des NextView Scripts importieren.
Import "os"
...

Sub Test ()
  Dim fio as Stream
  Dim i as Integer

  ' Textdatei datafile.txt erzeugen mit schreibendem Zugriff
  set fio = FileStream("datafile.txt", "w")
  if fio is Nothing then Exit Sub

  fio.Print "Dies ist eine Beispieldatei." & chr(13) & chr(10)
  fio.Close

  ' Textdatei datafile.txt öffnen um etwas am Ende einzufügen
  set fio = FileStream("datafile.txt", "a")
  if fio is Nothing then Exit Sub

  i = fio.Seek (0, StreamEnd)
  fio.Print "1" & chr(13) & chr(10)
  fio.Print "2" & chr(13) & chr(10)
  fio.Print "3" & chr(13) & chr(10)
  fio.Close
```

```
  ' Textdatei datafile.txt öffnen mit lesendem Zugriff
  set fio = FileStream("datafile.txt", "r")
  if fio is Nothing then Exit Sub

  do while Not(fio.IsEof)
    print fio.gets
  loop
  fio.Close

End Sub
```

Das folgende Beispiel liest eine durch Zeilenumschaltung (Carriage Return) terminierte Zeile von einer seriellen Schnittstelle und gibt diese aus:

```
' Stream Support am Anfang des NextView Scripts importieren.
Import "os"
...

Sub Test ()
  Dim ch as String
  Dim str as String
  Dim timeout as Integer
  Dim fio as Stream

  set fio = FileStream("com1", "w+")
  if fio is Nothing then
    Print "Error: Open COM failed."
    Exit Sub
  End If

  If fio.CommSetup ("38400", CommParityNone) <> 0 Then Exit Sub

  ' String an serielles Gerät schicken
  fio.Print "Sequenz starten" & chr(13) & chr(10)

  ' den durch ein Carriage Return beeendeten String empfangen
  timeout = TickCount

  do while (TickCount < (timeout+5000)) AND ch <> chr(13)
    ch = fio.getc
    if (ch <> "") then
      str = str & ch
      timeout = TickCount
    end if
  loop

  if (TickCount >= (timeout+5000)) then print "Timeout"
  print str
  fio.Close
End Sub
```

4.6.1 Stream: Überblick

Elementfunktion	Beschreibung
`Stream.FileStream`	liefert ein Stream Objekt zurück
`Stream.Close`	schließt den Stream
`Stream.Seek`	verändert die Schreib-/Leseposition im Stream
`Stream.Rewind`	setzt die Schreib-/Leseposition an den Anfang des Streams
`Stream.Gets`	liefert die nächste Zeile des Streams zurück
`Stream.Getc`	liefert ein Zeichen des Streams zurück
`Stream.IsEof`	gibt an, ob das Ende des Streams erreicht wurde
`Stream.Print`	gibt eine Zeile auf dem Stream aus
`Stream.Size`	gibt die Größe des Streams zurück
`Stream.Modtime`	gibt Datum und Uhrzeit der letzten Streamänderung zurück
`Stream.CommSetup`	setzt die Baudrate des seriellen COM Streams
`Stream.CommControl`	verändert die seriellen COM Leitungen DTR und RTS
`Stream.CommState`	holt den Status der seriellen COM Leitungen CTS, DSR, RING, und RLSD

4.6.2 Stream.FileStream

```
Function FileStream (path as String, mode as String) As
Stream
```

Die Funktion **`FileStream`** liefert den Stream mit dem Pfad **`path`** zurück mit dem eingestellten Zugriffsmodus **`mode`**. Der angegebene Pfad zu dem zu öffnenden Stream kann relativ oder absolut sein. Es können Textdateien und auch serielle Schnittstellen (z. B. "COM1") als Stream geöffnet werden. Als Zugriffsmodus **`mode`** sind folgende Optionen möglich:

mode	Bedeutung
r	Öffnet einen Stream mit lesendem Zugriff. Wenn der Stream nicht existiert oder gefunden wird, schlägt das Öffnen fehl.
w	Erzeugt einen leeren Stream mit schreibendem Zugriff. Existiert der Stream bereits, geht der Inhalt verloren.
a	Öffnet einen Stream um etwas anzuhängen. Existiert die Datei nicht, wird sie erzeugt. Wenn die Datei bereits existiert, wird ihr Inhalt nicht gelöscht.
r+	Öffnet einen Stream zum Lesen und Schreiben. Wenn der Stream nicht existiert oder gefunden wird, schlägt das Öffnen fehl.
w+	Erzeugt einen leeren Stream zum Lesen und Schreiben. Existiert der Stream bereits, geht der Inhalt verloren.
a+	Öffnet einen Stream zum Lesen und Anhängen. Existiert die Datei nicht, wird sie erzeugt. Wenn die Datei bereits existiert, wird ihr Inhalt nicht gelöscht.

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Standardklasse "Stream" befindet sich im Kapitel "Stream: Überblick", S. 198.

4.6.3 Stream.Close

```
Sub Close ()
```

Die Prozedur **Close()** schließt einen offenen Stream.

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Standardklasse "Stream" befindet sich im Kapitel "Stream: Überblick", S. 198.

4.6.4 Stream.Seek

```
Function Seek (pos as Integer, start as Integer) as Integer
```

Die Funktion **Seek()** setzt den Schreib-/Lesezeiger im Stream auf die Position **pos** relativ zu **start**. Die Funktion liefert den Wert des Schreib-/Lesezeigers nach dem Aufruf im Stream zurück.

Für **start** sind folgende Werte möglich:

start	Bedeutung
StreamBegin	Position ist relativ zum Stream Anfang
StreamEnd	Position ist relativ zum Stream Ende
StreamCurrent	Position ist relativ zur momentanen Position im Stream

Beispielsweise setzt **Seek(0, StreamEnd)** den Schreib-/Lesezeiger ans Ende des Streams.

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Standardklasse "Stream" befindet sich im Kapitel "Stream: Überblick", S. 198.

4.6.5 Stream.Rewind

```
Sub Rewind ()
```

Die Prozedur **Rewind()** setzt den Schreib-/Lesezeiger an den Anfang des Streams.

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Standardklasse "Stream" befindet sich im Kapitel "Stream: Überblick", S. 198.

4.6.6 Stream.Gets

```
Function Gets () as String
```

Die Funktion **Gets()** liest eine Zeile aus dem Stream. Es werden soviel Zeichen aus dem Stream gelesen, bis ein Zeilenende (*Carriage Return*) oder das Ende des Streams erreicht wurde, oder maximal 1024 Zeichen auf einmal gelesen wurden.

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Standardklasse "Stream" befindet sich im Kapitel "Stream: Überblick", S. 198.

4.6.7 Stream.Getc

```
Function Getc () as String
```

Die Funktion **Getc()** liest ein Zeichen aus dem Stream.

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Standardklasse "Stream" befindet sich im Kapitel "Stream: Überblick", S. 198.

4.6.8 Stream.IsEof

```
Function IsEof () as Boolean
```

Die Funktion **IsEof()** liefert **True**, wenn das Ende des Streams erreicht wurde.

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Standardklasse "Stream" befindet sich im Kapitel "Stream: Überblick", S. 198.

4.6.9 Stream.Print

```
Sub Print (str as String)
```

Die Prozedur **Print(str)** schreibt den String **str** in den Stream an der momentanen Schreib-/Lesezeiger Position.

Um bei einem existierenden Stream mit der Funktion `Print` Zeichen anzuhängen, muss man den Stream mit Zugriffsmodus "a" oder "a+" geöffnet haben (s. "`Stream.FileStream`", S. 198) und der Schreib-/Lesezeiger muss vor dem `Print`-Aufruf ans Ende des Streams gesetzt (`Seek(0,StreamEnd)`, s. "`Stream.Seek`", S. 199) worden sein.

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Standardklasse "Stream" befindet sich im Kapitel "Stream: Überblick", S. 198.

4.6.10 Stream.Size

```
Function Size () as Integer
```

Die Funktion **`Size()`** liefert die Größe des Streams zurück.

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Standardklasse "Stream" befindet sich im Kapitel "Stream: Überblick", S. 198.

4.6.11 Stream.Modtime

```
Function Modtime () as String
```

Die Funktion **`Modtime()`** gibt Datum und Uhrzeit der letzten Streamänderung zurück.

Dabei wird folgendes Format verwendet:

`YYYY-MM-DD hh:mm:ss` mit: **YYYY: Jahr**
MM: Monat
DD: Tag
hh: Stunde
mm: Minute
ss: Sekunde

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Standardklasse "Stream" befindet sich im Kapitel "Stream: Überblick", S. 198.

4.6.12 Stream.CommSetup

```
Function CommSetup (baud as String, Optional parity as
Integer = CommParityNone, Optional stop as Integer =
CommStopBits10) as Integer
```

Die Funktion **CommSetup(baud)** setzt die Baudrate des seriellen COM Streams auf **baud**. Die Funktion liefert den Wert **0** zurück, wenn der Aufruf funktioniert hat.

Für die Parity Bits **parity** und die Anzahl der Stopbits **stop** sind folgende Werte möglich:

Stop Bits	Wert
CommStopBits10	1 stop bit
CommStopBits15	1,5 stop bits
CommStopBits20	2 stop bits

Parity Prüfbits	Wert
CommParityNone	No parity
CommParityOdd	Odd parity
CommParityEven	Even parity
CommParityMark	Mark parity
CommParitySpace	Space parity

```
Dim fio as Stream
set fio = FileStream("com1", "w+")
If fio.CommSetup ("38400", CommParityOdd, ComStopBits10) <> 0 Then

   ...

End If
```

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Standardklasse "Stream" befindet sich im Kapitel "Stream: Überblick", S. 198.

4.6.13 Stream.CommControl

```
Function CommControl (dtr as Integer, rts as Integer) as
Integer
```

Die Funktion **CommControl(dtr,rts)** verändert die seriellen COM Leitungen DTR (*data-terminal-ready*) und RTS (*request-to-send*). Die Funktion liefert den Wert **0** zurück, wenn der Aufruf funktioniert hat.

Die jeweilige Leitung kann folgendermaßen beeinflusst werden:

Zustand	Bedeutung
-1	keine Änderung an der Leitung
0	Leitung wird gelöscht
1	Leitung wird gesetzt

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Standardklasse "Stream" befindet sich im Kapitel "Stream: Überblick", S. 198.

4.6.14 Stream.CommState

```
Function CommState (Byref cts As Integer, Byref dsr As
Integer, Byref ring As Integer, Byref rlsd As Integer) As
Integer
```

Die Funktion **CommState(cts, dsr, ring, rlsd)** holt den Status der seriellen COM Leitungen CTS (*clear-to-send*), DSR (*data-set-ready*), RING (*ring indicator*), und RLSD (*receive-line-signal-detect*). Die Funktion liefert den Wert **0** zurück, wenn der Aufruf funktioniert hat.

Die jeweiligen Leitungen haben folgenden Zustand:

Zustand	Bedeutung
`0`	Leitung ist nicht gesetzt
`1`	Leitung ist gesetzt

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Standardklasse "Stream" befindet sich im Kapitel "Stream: Überblick", S. 198.

4.7 Standardklasse "Clipboard"

Für den Zugriff auf die Zwischenablage des Betriebsystems (Operating System) Windows® gibt es in **NextView®4 Script** die Standardklasse "Clipboard". Ein Objekt der Klasse "Clipboard" wird über den **New** Befehl angelegt.

Um die Standardklasse "Clipboard" in NextView®4 Script zur Verfügung zu stellen, muss man am Anfang des NextView®4 Scripts das Modul Operating System (OS) importieren (s. "Import", S. 163).

Das folgende Beispiel zeigt die Verwendung der Standardklasse "Clipboard":

```
' Operating System Support importieren.
Import "os"
...

Sub Test ()
  Dim clip as Clipboard
  Dim str as String
  Dim i as Integer

  set clip = new Clipboard
  if clip.Open then
    str = ""
    for i = 1 to 16
      str = str & " " & NvAnalogIn(i).value
    next
    clip.Text = str
    clip.Close ' für andere Zwischenablage zugänglich machen
  end if
End Sub
```

4.7.1 Clipboard: Überblick

Elementfunktion	Beschreibung
`Clipboard.Open`	öffnet die Zwischenablage
`Clipboard.Text`	schreibt oder liest aus der Zwischenablage
`Clipboard.Close`	schließt die Zwischenablage

4.7.2 Clipboard.Open

```
Function Open () As Boolean
```

Die Funktion **`Open`** öffnet die Zwischenablage des Betriebsystems für die Benutzung in **NextView®4 Script**. Andere Programme können auf die Zwischenablage erst wieder zugreifen, wenn sie wieder geschlossen wurde (s. "**`Clipboard.Close`**", S. 208).

Eine Auflistung aller verfügbaren Elementfunktionen der Standardklasse "Clipboard" befindet sich im Kapitel "Clipboard: Überblick", S. 207.

4.7.3 Clipboard.Text

```
' Clipboard Inhalt auslesen
  Function Text () As String
' Clipboard Inhalt schreiben
  text = newText
```

Mit der Funktion **`text`** wird der Inhalt der Zwischenablage ausgelesen oder ein neuer Inhalt kann in die Zwischenablage abgelegt werden.

Eine Auflistung aller verfügbaren Elementfunktionen der Standardklasse "Clipboard" befindet sich im Kapitel "Clipboard: Überblick", S. 207.

4.7.4 Clipboard.Close

```
Sub Close()
```

Die Prozedur **Close** schließt die Zwischenablage für **NextView®4 Script**. Erst dann kann man auf die Zwischenablage von einem anderen Programm aus zugreifen.

Eine Auflistung aller verfügbaren Elementfunktionen der Standardklasse "Clipboard" befindet sich im Kapitel "Clipboard: Überblick", S. 207.

4.8 Standardklasse "XNF"

Für den Zugriff auf Initialisierungs-Textdateien im Windows® INI-Format gibt es in **NextView®4 Script** die Standardklasse "XNF" (*extended INI File format*). Ein Objekt der Klasse "XNF" wird über den **New** Befehl angelegt. Die Klasse "XNF" stellt einige einfache Prozeduren für den Zugriff auf INI-Dateien zur Verfügung.

Um die Standardklasse "XNF" in NextView®4 Script zur Verfügung zu stellen, muss man am Anfang des NextView®4 Scripts das Modul XNF importieren (s. "Import", S. 163).

Das folgende Beispiel zeigt die Verwendung der Standardklasse "XNF" für eine Initialisierungsdatei **Init.xnf** im Verzeichnis **c:\data** mit folgendem Inhalt:

```
[General]

Date=2010-01-22
AnalogOutNumber=2
[AnalogOut1]
Filename=Points1.txt
[AnalogOut2]
Filename=Points2.txt

Import "XNF"
...
```

```
Dim IniFile as XNF
Dim i as Integer
Dim nAnOut as Integer
Dim AnOutFileName(20) as String

Set IniFile = new XNF

If IniFile.LoadParams("c:\data\init.xnf") Then
  Print "Init Datum: " & IniFile.GetSectionParam ("General", "Date")
  nAnOut = IniFile.GetSectionParam ("General", "AnalogOutNumber")
  for i = 1 to nAnOut
    AnOutFileName(i) = IniFile.GetSectionParam ("AnalogOut"&i,
                                                "Filename")
  Next

Else
  print "Open XNF file failed."

End If
```

4.8.1 XNF: Überblick

Elementfunktion	Beschreibung
XNF.LoadParams	öffnet eine XNF Textdatei
XNF.SaveParams	schreibt eine XNF Textdatei
XNF.SectionCount	liefert die Anzahl der XNF Bereiche
XNF.GetSectionName	liefert einen XNF Bereichsnamen
XNF.SectionLineCount	liefert die Anzahl der Zeilen in einem XNF Bereich
XNF.GetSectionLine	liefert eine Zeile aus einem XNF Bereich
XNF.GetSectionParam	liest einen Parameter aus einem XNF Bereich
XNF.SetSectionParam	setzt einen Parameter in einem XNF Bereich
XNF.DeleteSection	löscht den Inhalt eines XNF Bereichs

4.8.2 XNF.LoadParams

```
Function LoadParams (filename as String) As Boolean
```

Die Funktion **LoadParams** öffnet eine Initialisierungs-Textdatei, die im Windows® INI-Format geschrieben ist.

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Standardklasse "XNF" befindet sich im Kapitel "XNF: Überblick", S. 209.

4.8.3 XNF.SaveParams

```
Function SaveParams (filename as String, merge as Integer) as
Boolean
```

Speichert die XNF-Bereiche in eine INI-Datei mit dem Namen **filename**. Bei **merge** = 1 werden die INI-Bereiche an die existierende Bereiche in der INI-Datei angehängt, andernfalls wird eine neue Datei erzeugt.

```
Dim IniFile as XNF
Set IniFile = new XNF

if IniFile.SetSectionParam ("AnalogOut1","Filename","AnOut1.txt")
then
  print "Set Ok"
End If

if IniFile.SaveParams("c:\data\newinit.xnf",0) then
  print "Save Ok"

End If
```

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Standardklasse "XNF" befindet sich im Kapitel "XNF: Überblick", S. 209.

4.8.4 XNF.SectionCount

```
Function SectionCount () as Integer
```

Liefert die Anzahl der XNF-Bereiche.

```
print "Anzahl Bereiche: " & IniFile.SectionCount
```

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Standardklasse "XNF" befindet sich im Kapitel "XNF: Überblick", S. 209.

4.8.5 XNF.GetSectionName

```
Function GetSectionName (index as Integer) as String
```

Liefert den Namen des XNF-Bereichs **index**. Dieser beginnt bei 1 bis zur Anzahl an XNF-Bereichen (s. "**XNF.SectionCount**", S. 210).

```
print "1. Bereich Name: " & IniFile.GetSectionName(1)
```

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Standardklasse "XNF" befindet sich im Kapitel "XNF: Überblick", S. 209.

4.8.6 XNF.SectionLineCount

```
Function SectionLineCount (section as String) as Integer
```

Liefert die Anzahl der Zeilen eines XNF-Bereichs.

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Standardklasse "XNF" befindet sich im Kapitel "XNF: Überblick", S. 209.

4.8.7 XNF.GetSectionLine

```
Function GetSectionLine (section as String, index as Integer)
as String
```

Liefert die Zeile **index** aus dem XNF Bereich **section**. **Index** beginnt bei 1 bis zur Anzahl an Zeilen in einem XNF-Bereich (s. "**XNF.SectionLineCount**", S. 211).

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Standardklasse "XNF" befindet sich im Kapitel "XNF: Überblick", S. 209.

4.8.8 XNF.GetSectionParam

```
Function GetSectionParam (section as String, name as String)
as String
```

Liest den Parameter **name** aus dem Bereich **section**.

```
 [Data1]
NameX=Testwert

Dim str as String
str = IniFile.GetSectionParam("Data1","NameX")
Print str ' gibt Testwert aus
```

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Standardklasse "XNF" befindet sich im Kapitel "XNF: Überblick", S. 209.

4.8.9 XNF.SetSectionParam

```
Function SetSectionParam (section as String, name as String,
value as String) as Boolean
```

Schreibt den Wert **value** des Parameters **name** in dem XNF-Bereich **section**.

```
Dim str as String
If IniFile.SetSectionParam("Section2","NameY","Testwert Y") Then
End If
```

Erzeugt folgenden Bereich in einer XNF-Datei:

```
[Section2]
NameY=Testwert Y
```

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Standardklasse "XNF" befindet sich im Kapitel "XNF: Überblick", S. 209.

4.8.10 XNF.DeleteSection

```
Sub DeleteSection (section as String)
```

Löscht den Inhalt eines Bereichs in einem XNF-Objekt.

Eine Auflistung aller zur Verfügung stehenden Elementfunktionen der Standardklasse "XNF" befindet sich im Kapitel "XNF: Überblick", S. 209.

5 Index

?

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y

Z