



NS Basic/CE Handbuch

Im Februar 2007

© NS BASIC Corporation, 2007.
71 Hill Crescent
Toronto, Canada M1M 1J3
+1 (416) 264-5999

Dieses Handbuch, sowie die Software, die es beschreibt, ist urheberrechtlich geschützt und darf ohne ausdrückliche schriftliche Genehmigung der NS BASIC Corporation in keiner Weise, auch nicht auszugsweise, vervielfältigt, übersetzt oder in eine andere Darstellungsform gebracht werden.

Herausgeber, Übersetzer und Autoren dieser Publikation haben mit größter Sorgfalt die Texte, Abbildungen und Programme erarbeitet. Dennoch können Fehler nicht völlig ausgeschlossen werden. NS BASIC Corporation übernimmt daher weder eine Garantie noch eine juristische Verantwortung oder Haftung für Folgen, die auf fehlerhafte Angaben zurückgehen. Die in diesem Handbuch enthaltenen Angaben und Daten können ohne vorherige Ankündigung geändert werden.

Microsoft, MS, Windows, und Windows CE sind eingetragene Warenzeichen der Microsoft Corporation, registriert in den Vereinigten Staaten von Amerika und in anderen Ländern.

Diejenigen Bezeichnungen in dieser Publikation von Erzeugnissen und Verfahren, die zugleich Warenzeichen sind, wurden nicht besonders kenntlich gemacht. Solche Namen sind Warenzeichen der jeweiligen Warenzeicheninhaber. Aus dem Fehlen der Markierung R kann nicht geschlossen werden, dass diese Bezeichnungen freie Warenzeichen sind. Das Erwähnen von Produkten und Warenzeichen dient lediglich der Information und stellt in keiner Weise eine Empfehlung dar.

Die NS BASIC Corporation übernimmt keine Verantwortung im Hinblick auf die Performance und die Nutzung von NS BASIC, sowie den Einsatz von Produkten, die in diesem Handbuch erwähnt werden.

Canadian Cataloguing In Publication Data

Darden, Marcus M., 1970-

NS BASIC/CE Handbook

Includes index.

ISBN 0-9695844-4-X

BASIC (Computer program Language).

2. Windows (Computer file) - Programming.

I. Henne, George W.P. 1954- . II. Title

QA76.73.B3D37 1998 005.4'3 C98-932304-8

Deutsche Übersetzung von Ralf Puppe

LIZENZVEREINBARUNG

DIESE ÜBERSETZUNG DIENT LEDIGLICH DEM BESSEREN VERSTÄNDNIS. RECHTSGÜLTIG IST DIE, EBENFALLS IN DIESEM HANDBUCH ABGEDRUCKTE, ENGLISCHE ORIGINALFASSUNG.

BITTE LESEN SIE DIESE LIZENZVEREINBARUNG SORGFÄLTIG DURCH, BEVOR SIE DIE SOFTWARE BENUTZEN. DURCH DIE NUTZUNG DER SOFTWARE ERKLÄREN SIE SICH MIT DEM INHALT DIESER VEREINBARUNG IN ALLEN PUNKTEN EINVERSTANDEN. SOLLTEN SIE DER VEREINBARUNG NICHT ZUSTIMMEN, SENDEN SIE DAS PRODUKT BITTE UNVERZÜGLICH AN IHREN LIEFERANTEN ZURÜCK. SIE ERHALTEN DANN DEN KAUFPREIS ERSTATTET.

1. Lizenz: Die Anwendung, die Demoversion, die Systemkomponenten und andere Software, die dieser Lizenz beigelegt sind, sei es auf Diskette, im Read-Only-Memory, oder auf sonstigen Medien (im weiteren die „Software“ genannt), die zugehörige Dokumentation und die Zeichensätze (im weiteren „Fonts“ genannt) werden durch die NS BASIC Corporation (im weiteren mit „NSBC“ bezeichnet) lizenziert. Dem Lizenznehmer gehört das Medium, auf dem die Software und Fonts gespeichert sind. Die Rechte an der Software, der zugehörigen Dokumentation und den Fonts verbleiben bei der NSBC, ihren Lizenzgebern und Lieferanten, geschützt durch Urheberrechtsgesetze der Vereinigten Staaten von Amerika, durch Gesetze anderer Nation und durch internationale Verträge. Diese Lizenz erlaubt es Ihnen, die Software und Fonts auf einem einzelnen Windows CE - Produkt zu betreiben (damit soll im Rahmen dieser Lizenzvereinbarung ein Produkt gemeint sein, das ein Microsoft CE Logo trägt) und eine Kopie der Software und Fonts in maschinenlesbarer Form zu Sicherungszwecken zu erstellen. Diese Kopie muss alle NSBC Urheberrechtshinweise und Eigentumsvermerke der Originalmedien zu Software und Fonts enthalten. Sie können Ihre Lizenzrechte an der Software und den Fonts, den Sicherungskopien der Software und der Fonts, der zugehörigen Dokumentation und diesem Lizenzvertrag auf Dritte übertragen, vorausgesetzt diese dritte Partei liest diese Lizenzvereinbarung und erklärt sich mit dem Inhalt in allen Punkten einverstanden.

2. Einschränkungen: Die Software enthält urheberrechtlich geschütztes Material, Geschäftsgeheimnisse und andere Teile unter Eigentumsvorbehalt, zu deren Schutz es Ihnen untersagt ist, sie zu dekompile, rekonstruieren, disassemblieren oder sie auf irgendeine andere Weise in eine Form zu bringen, die geeignet ist, interne Strukturen aufzuzeigen. Sie dürfen sie nicht modifizieren, vernetzen, vermieten, verleasen, belasten, vertreiben oder Derivate zu erzeugen, die auf dieser Software basieren, weder im ganzen, noch in Teilen. Sie dürfen die Software nicht auf elektronischem Weg von einem Gerät zu einem anderen übertragen, auch nicht im Netzwerk.

3. Ende des Lizenzvertrags. Diese Lizenzvereinbarung besteht bis sie beendet wird. Sie können sie jederzeit beenden, indem Sie die Software, die zugehörige Dokumentation und die Fonts löschen. Diese Vereinbarung erlischt sofort, ohne eine Nachricht von NSBC, wenn Sie sich mit einer der Klauseln dieser Lizenzvereinbarung nicht einverstanden erklären. Sie müssen in diesem Fall unverzüglich die Software, die Dokumentation und die Fonts löschen.

4. Exportbestimmungen. Sie stimmen zu, dass weder die Software, noch irgendwelche technischen Daten, die Sie von NSBC erhalten haben, noch das Produkt selbst, in Länder außerhalb der Vereinigten Staaten von Amerika exportiert werden, sofern dies nicht durch Rechte und Verordnungen der Vereinigten Staaten von Amerika erlaubt wird. Sofern Sie die Software rechtmäßig außerhalb der Vereinigten Staaten von Amerika erworben haben, stimmen Sie zu, dass Sie weder die Software, noch irgendwelche technischen Daten, die Sie von NSBC erhalten haben, noch das Produkt selbst, in Länder außerhalb der Vereinigten Staaten von Amerika reexportieren werden, sofern dies nicht durch Rechte und Verordnungen der Vereinigten Staaten von Amerika und dem Land, in dem Sie Rechte an der Software erworben haben, erlaubt wird.

5. Regierungsanwender. Wenn Sie die Software im Auftrag einer Abteilung oder einer anderen Stelle der Regierung der Vereinigten Staaten von Amerika erwerben, gelten die nachfolgenden Bestimmungen. Die Regierung stimmt zu: (i) wenn die Software und die Fonts an das Verteidigungsministerium (DoD) geliefert werden, werden die Software und die Fonts als "Commercial Computer Software" klassifiziert und die Regierung erwirbt nur „eingeschränkte Rechte“, wie sie in Klausel 252.227-

7013(c)(1) der DFARS definiert sind, an der Software, der Dokumentation und den Fonts, und (ii), wenn die Software und Fonts an eine andere Abteilung oder Stelle der Vereinigten Staaten von Amerika als das Verteidigungsministerium geliefert werden, ergeben sich die Rechte der Regierung aus Klausel 52.227-19(c)(2) der FAR, oder im Falle der NASA, aus Klausel 18-52.227-86(d) der NASA Erweiterungen zur FAR.

6. NS BASIC ersetzt defekte Disketten oder Handbücher kostenlos innerhalb von 90 Tagen ab Kaufdatum. NS BASIC sichert zu, dass die Programme im Allgemeinen in Übereinstimmung mit der Dokumentation betrieben werden können. NS BASIC sichert nicht zu, dass die Programme und die Handbücher frei von Fehlern und Irrtümern sind.

7. Gewährleistungsausschluss. Sie erkennen ausdrücklich an, dass die Nutzung der Software und der Fonts auf Ihr eigenes Risiko geschieht. Die Software, die zugehörige Dokumentation und die Fonts werden geliefert, wie gesehen („as is“), ohne jegliche Garantie von NSBC oder Lizenznehmern von NSBC (für die Bestimmungen 7 und 8, sollen unter dem Begriff NSBC, sowohl NSBC, als auch Lizenznehmer von NSBC verstanden werden).
AUSDRÜCKLICH AUSGESCHLOSSEN SIND ALLE GEWÄHRLEISTUNGEN, SOWOHL EXPLIZID ERWÄHNTE, ALS AUCH KONKLUDENTE, EINSCHLIEßLICH; ABER NICHT BESCHRÄNKT AUF, MÄNGELGEWÄHRLEISTUNGEN: NSBC GARANTIERT NICHT, DASS DIE FUNKTIONEN IN DIESER SOFTWARE IHRE ANFORDERUNGEN ERFÜLLT, ODER DASS DIE SOFTWARE UNUNTERBROCHEN UND FEHLERFREI BETRIEBEN WERDEN KANN; ODER DASS DEFEKTE SOFTWARE ODER FONTS KORRIGIERT WERDEN. DESWEITEREN MACHT NSBC KEINE ÄUßERUNGEN IM BEZUG AUF DIE VERWENDBARKEIT DER MIT DER SOFTWARE UND DEN FONTS ERSTELLTEN PROGRAMME, HINSICHTLICH IHRER GENAUIGKEIT, KORREKTHEIT, ZUVERLÄSSIGKEIT, ODER ANDEREN EIGENSCHAFTEN. WEDER MÜNDLICHE, NOCH SCHRIFTLICHE ANWEISUNGEN, DIE SIE DURCH NSBC ODER EINEM AUTHORIZIERTEN VERTRETER VON NSBC ERHALTEN, STELLEN IN IRGEND EINER WEISE EINE ERWEITERUNG DIESER GEWÄHRLEISTUNGSVEREINBARUNG DAR. SOLLTE SICH DIE SOFTWARE ALS DEFEKT HERAUSSTELLEN, SO TRAGEN SIE (NICHT NSBC ODER EIN REPRÄSENTANT VON NSBC) DIE KOSTEN FÜR EVENTUELLE NACHBESSERUNGEN, REPARATUREN

ODER KORREKTUREN. MANCHE GERICHTSBARKEITEN
ERLAUBEN NICHT DEN AUSSCHLUß VON IMPLIZITEN
GARANTIEEN; IN DIESEM FALL TREFFEN DIE VORHER
GETROFFENEN AUSSAGEN FÜR SIE NICHT ZU.

8. Haftungsbeschränkung. Da Software an sich komplex und
niemals frei von Fehlern ist, raten wir Ihnen, die
Anwendungen, die mit unserer Software erstellt wurden,
genau zu prüfen. UNTER KEINEN UMSTÄNDEN, AUCH
NICHT BEI FAHRLÄSSIGKEIT, KANN NSBC FÜR
JEDLICHE ZUFÄLLIGEN, BESONDEREN ODER
FOLGERICHTIGEN SCHÄDEN VERANTWORTLICH
GEMACHT WERDEN; DIE SICH AUS GEBRAUCH ODER
LEISTUNGSSCHWÄCHEN DER SOFTWARE UND DER
ZUGEHÖRIGEN DOKUMENTATION ERGEBEN, AUCH
DANN NICHT; WENN NSBC ODER EIN
AUTHORISIERTER VERTRIEBSPARTNER ZUVOR AUF
DIE MÖGLICHKEIT EINES SOLCHEN SCHADENS
HINGEWIESEN WURDEN. MANCHE
GERICHTSBARKEITEN ERLAUBEN NICHT DEN
AUSSCHLUß ODER DIE LIMITIERUNG VON ZUFÄLLIGEN
ODER FOLGESCHÄDEN; IN DIESEM FALL TREFFEN DIE
VORHER GETROFFENEN AUSSAGEN FÜR SIE NICHT
ZU. In keinem Fall (weder aus Vertragsgründen, noch aus
unerlaubten Handlungen (einschließlich Fahrlässigkeit) noch
aus sonstigen Gründen) kann die Gewährleistung für
Schäden jeglicher Art, die Kaufsumme für die Software und
die Fonts überschreiten.

9. Risikoverteilung. Sie nehmen zur Kenntnis und stimmen
zu, dass diese Vereinbarung eine Risikoaufteilung zwischen
Ihnen und NSBC darstellt, so wie es durch den „Uniform
Commercial Code“ und andere anwendbare Gesetze
zugelassen ist und dass die Preisbildung der NSBC
Produkte diese Risikoverteilung und die
Haftungsbeschränkungen in dieser Vereinbarung
widerspiegeln. Sollte eine dieser Bestimmungen ihren
wesentlichen Zweck nicht erfüllen, so bleiben die anderen
Haftungsbeschränkungen trotzdem bestehen.

10. Problembeseitigung. NSBC kann nach eigenem
Ermessen Unterstützung bei der Behebung von Problemen
zu festgelegten Gebührensätzen anbieten. Die
Unterstützung unterliegt den Haftungsbeschränkungen
dieser Lizenz.

11. Zusätzliche Einschränkungen. Jede nachträgliche Verbesserung oder Erweiterung des Programms, die durch NSBC geliefert wird, darf nur genutzt werden, wenn die vorherige Version gelöscht wird. Sie unterliegt ebenfalls den Klauseln dieser Vereinbarung.

12. Rechtswahl und Salvatoresche Klausel. Diese Lizenzvereinbarung soll in Übereinstimmung mit den Gesetzen der Vereinigten Staaten von Amerika und des Staates Delaware ausgelegt und geregelt werden, so wie es die Gesetze von Delaware für Verträge zwischen Bürgern des Staates Delaware vorsehen. Sollte eine der Bestimmungen dieser Lizenz, oder Teile davon, durch ein zuständiges Gericht für unwirksam erklärt werden, so soll diese Klausel so ausgelegt werden, wie es die Vertragsparteien vorgesehen hatten. Die Wirksamkeit der übrigen Bestimmungen dieser Lizenz bleibt dadurch unberührt.

13. Schlussbestimmung. Diese Lizenz stellt die Gesamtvereinbarung zwischen den Parteien im Bezug auf Nutzung der Software, zugehöriger Dokumentation und Fonts dar. Durch sie verlieren alle früher oder gleichzeitig bestehenden Vereinbarungen oder Übereinkünfte, unabhängig davon, ob sie schriftlich oder verbal getroffen wurden, ihre Wirkung. Änderungen oder Ergänzungen dieser Lizenzvereinbarung bedürfen der Schriftform und sind nur wirksam, wenn sie durch einen Bevollmächtigten der NS BASIC Corporation gegengezeichnet wurden.

LICENSE AGREEMENT

PLEASE READ THIS LICENSE CAREFULLY BEFORE USING THE SOFTWARE. BY USING THE SOFTWARE, YOU ARE AGREEING TO BE BOUND BY THE TERMS OF THIS LICENSE. IF YOU DO NOT AGREE TO THE TERMS OF THIS LICENSE, PROMPTLY RETURN THE PRODUCT TO THE PLACE WHERE YOU OBTAINED IT AND YOUR MONEY WILL BE REFUNDED.

1. License. The application, demonstration, system, and other software accompanying this License, whether on disk, in read only memory, or on any other media (the "Software"), the related documentation and fonts are licensed to you by NS BASIC Corporation ("NSBC"). You own the media on which the Software and fonts are recorded but NSBC and or NSBC's Licensor(s) retain title to the Software, related documentation and fonts. This License allows you to use the Software and fonts on a single Windows CE Product (which, for purposes of this License, shall mean a product bearing Microsoft's Windows CE logo), and make one copy of the Software and fonts in machine-readable form for backup purposes only. You must reproduce on such copy the NSBC copyright notice and any other proprietary legends that were on the original copy of the Software and fonts. You may also transfer all your license rights in the Software and fonts, the backup copy of the Software and fonts, the related documentation and a copy of this License to another party, provided the other party reads and agrees to accept the terms and conditions of this License.

2. Restrictions. The Software contains copyrighted material, trade secrets and other proprietary material and in order to protect them you may not decompile, reverse engineer, disassemble or otherwise reduce the Software to a human-perceivable form. You may not modify, network, rent, lease, load, distribute or create derivative works based upon the Software in whole or in part. You may not electronically transmit the Software from one device to another or over a network.

3. Termination. This License is effective until terminated. You may terminate this License at any time by destroying the Software and related documentation and fonts. This License will terminate immediately without notice from NSBC if you fail to comply with any provision of this License. Upon termination you must destroy the Software, related documentation and fonts.

4. Export Law Assurances. You agree and certify that neither the Software nor any other technical data received from NSBC, nor the direct product thereof, will be exported outside the United States except as authorized and as permitted by the laws and regulations of the United States. If

the Software has been rightfully obtained by you outside of the United States, you agree that you will not reexport the Software nor any other technical data received from NSBC, nor the direct product thereof, except as permitted by the laws and regulations of the United States and the laws and regulations of the jurisdiction in which you obtained the Software.

5. Government End Users. If you are acquiring the Software and fonts on behalf of any unit or agency of the United States Government, the following provisions apply. The Government agrees: (i) if the Software and fonts are supplied to the Department of Defense (DoD), the Software and fonts are classified as "Commercial Computer Software" and the Government is acquiring only "restricted rights" in the Software, its documentation and fonts as that term is defined in Clause 252.227-7013(c)(1) of the DFARS; and (ii) if the Software and fonts are supplied to any unit or agency of the United States Government other than DoD, the Governments' rights in the Software, its documentation and fonts will be as defined in Clause 52.227-19(c)(2) of the FAR or, in the case of NASA, in Clause 18-52.227-86(d) of the NASA supplement to the FAR.

6. NS BASIC will replace at no charge defective disks or manuals within 90 days of the date of purchase. NS BASIC warrants that the programs will perform generally in compliance with the included documentation. NS BASIC does not warrant that the programs and manuals are free from all bugs, errors or omissions.

7. Disclaimer of Warranty on Software. You expressly acknowledge and agree that use of the Software and fonts is at your sole risk. The Software, related documentation and fonts are provided "AS IS" and without warranty of any kind and NSBC and NSBC's Licensor(s) (for the purposes of provisions 7 and 8, NSBC and NSBC's Licensor(s) shall be collectively referred to as "NSBC") EXPRESSLY DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. NSBC DOES NOT WARRANT THAT THE FUNCTIONS CONTAINED IN THE SOFTWARE WILL MEET YOUR REQUIREMENTS, OR THAT THE OPERATION OF THE SOFTWARE WILL BE UNINTERRUPTED OR ERROR-FREE, OR THAT DEFECTS IN THE SOFTWARE AND THE FONTS WILL BE CORRECTED. FURTHERMORE, NSBC DOES NOT WARRANT OR MAKE ANY REPRESENTATIONS REGARDING THE USE OR THE RESULTS OF THE USE OF THE SOFTWARE AND FONTS OR RELATED DOCUMENTATION IN TERMS OF THEIR CORRECTNESS, ACCURACY, RELIABILITY, OR OTHERWISE. NO ORAL OR WRITTEN INFORMATION OR ADVICE GIVEN BY NSBC OR A NSBC AUTHORIZED REPRESENTATIVE SHALL CREATE A WARRANTY OR IN ANY WAY INCREASE THE SCOPE OF THIS WARRANTY. SHOULD THE SOFTWARE PROVE DEFECTIVE, YOU (AND NOT NSBC OR AN NSBC AUTHORIZED REPRESENTATIVE) ASSUME THE ENTIRE COST

OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION. SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OF IMPLIED WARRANTIES, SO THE ABOVE EXCLUSION MAY NOT APPLY TO YOU.

8. Limitation of Liability. Because software is inherently complex and may not be free from errors, you are advised to verify the work produced by the Program. UNDER NO CIRCUMSTANCES INCLUDING NEGLIGENCE, SHALL NSBC BE LIABLE FOR ANY INCIDENTAL, SPECIAL OR CONSEQUENTIAL DAMAGES THAT RESULT FROM THE USE OR INABILITY TO USE THE SOFTWARE OR RELATED DOCUMENTATION, EVEN IF NSBC OR A NSBC AUTHORIZED REPRESENTATIVE HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. SOME JURISDICTIONS DO NOT ALLOW THE LIMITATION OR EXCLUSION OF LIABILITY FOR INCIDENTAL OR CONSEQUENTIAL DAMAGES SO THE ABOVE LIMITATION OR EXCLUSION MAY NOT APPLY TO YOU. In no event shall NSBC's total liability to you for all damages, losses, and causes of action (whether in contract, tort (including negligence) or otherwise) exceed the amount paid by you for the Software and fonts.

9. Allocation of Risk: You acknowledge and agree that this Agreement allocates risk between you and NSBC as authorized by the Uniform Commercial Code and other applicable law and that the pricing of NSBC's products reflects this allocation of risk and the limitations of liability contained in this Agreement. If any remedy hereunder is determined to have failed of its essential purpose, all limitations of liability and exclusions of damages as set forth in this Agreement will remain in effect.

10. Support. NSBC may, at its option, provide support services at its standard fees for such services. Such support services will be governed by the limitations of liability under this Agreement.

11. Additional Restrictions: Any upgrade or enhancement of the program subsequently supplied by NSBC may only be used upon the destruction of the prior version, and shall be governed by the terms of this Agreement.

12. Controlling Law and Severability. This License shall be governed by and construed in accordance with the laws of the United States and the State of Delaware, as applied to agreements entered into and to be performed entirely within Delaware between Delaware residents. If for any reason a court of competent jurisdiction finds any provision of this License, or portion thereof, to be unenforceable, that provision of the License shall be enforced to the maximum extent permissible so as to effect the intent of the parties, and the remainder of this License shall continue in full force and effect.

13. Complete Agreement. This License constitutes the entire agreement between the parties with respect to the use of the Software, related documentation and fonts, and supersedes all prior or contemporaneous understandings or agreements,

written or oral, regarding such subject matter. No amendment to or modification of this License will be binding unless in writing and signed by a duly authorized representative of NSBC.

INHALT

.....

1. Einleitung	17
1.1 Was ist BASIC?	18
1.1.1 NS Basic/CE	18
1.1.2 NS Basic/CE und Windows CE	18
1.2 System Anforderungen	19
1.3 Installation	19
1.3.1 Eingabe Ihrer Seriennummer	19
2. NS Basic/CE Konzepte	20
2.1 Typografische Konventionen	20
2.2 Die Elemente eines NS Basic/CE Programms	21
2.2.1 Mehrzeilige Anweisungen	21
2.2.2 Mehrere Anweisungen pro Zeile	21
2.2.3 Literale, Datentypen und Variablen	22
2.2.4 Numerische Datentypen	22
2.2.5 Datentyp Boolean	22
2.2.6 Datentyp Farbe (Color)	23
2.2.7 Datentyp String	23
2.2.8 Datentyp Array	23
2.2.9 Variablennamen	23
2.3 Ausdrücke und Operatoren	24
2.3.1 Arithmetische Operatoren	24
2.3.2 Vergleichs-Operatoren	25
2.3.3 Boolesche Operatoren	25
2.4 FUNKTIONEN und SUB Prozeduren	26
2.5 Projekte, Module, Forms (Formulare), Objekte und Steuerelemente	27
Steuerelemente und Objekte	27
Objekt Ereignisse	27
Forms (Formulare)	28
Form Methoden	28
Form Ereignisse	29
Programm Eigenschaften und Ereignisse	30
Programm Ereignisse	30
Module	30
3. Programmieren auf dem Windows/CE Gerät	31
3.1 Übersicht	31
3.2 Programm-Editor	31
3.3 Ausgabe-Fenster	33
3.4 Visual Designer	33
3.4.1 Eigenschaften-Editor	36
3.4.2 Menü-Editor	36
3.4.3 Anmerkung	36
3.5 Die NS Basic/CE Programmier Umgebung	37
3.5.1 Ein Programm erstellen	37
3.5.2 Editieren eines Programms	38
3.5.3 Formattieren eines Programms	38

3.5.4 Finden und Ersetzen.....	39
3.5.5 Overview (Übersicht).....	39
3.5.6 Ausführen eines Programms.....	40
3.5.7 Debuggen eines Programms	40
3.5.8 BREAK, Execute Funktion, Trace und Step....	43
3.5.9 Speichern und Laden eines Programms.....	44
3.5.10 Speichern und Laden von Programmen als Textdateien	45
4. Programmieren auf dem Desktop PC.....	47
4.1 Menü Optionen.....	47
4.2 Programm-Code Fenster	51
4.3 CE Bildschirm.....	52
4.4 Projekt Explorer.....	54
4.5 Eigenschaften-Fenster	55
4.6 Toolbox (Werkzeugkasten)	56
4.7 Toolbar (Werkzeugleiste)	58
4.8 Menu Editor (Menü-Editor)	59
4.9.1 Optionen – General.....	61
4.9.2 Optionen – Editor	62
4.9.3 Optionen – CE Screen	63
4.9.4 Optionen – Start.....	64
5. NS Basic/CE Referenz	65
6. Weiterführende Themen	245
6.1 Programmierrichtlinien	245
6.1.1 Richtlinien zur Namensgebung	245
6.1.2 Richtlinien zur Textformatierung	246
6.1.3 Kommentar-Richtlinien.....	246
6.2 Fehlerbehandlung	246
6.2.1 Defensives Programmieren.....	247
6.2.2 Abfangen von Fehlern.....	247
6.2.3 Abfangen von Fehlern bei Datei-Operationen	248
A. Error Codes	251
B. Konstanten	253

1. Einleitung

Willkommen bei NS Basic/CE für Windows CE. NS Basic/CE wurde entworfen um den Anforderungen von Windows CE Anwendern zu entsprechen. Es ist eine einfache, aber mächtige Programmiersprache, die zum Schreiben nahezu aller Arten von Anwendungen geeignet ist.

Auf der mitgelieferten CD gibt es eine Datei mit Namen Readme.txt. Sie enthält die aktuellsten Informationen zu NS Basic/CE, inklusive Änderungen zum Handbuch. Bitte beachten Sie diese Datei, bevor Sie NS Basic/CE installieren.

Wenn Sie sofort mit NS Basic/CE beginnen möchten, lesen Sie sich zunächst den Abschnitt Installation durch und wenden Sie sich danach dem Kapitel Mit NS Basic/CE starten zu.

Mit NS Basic/CE werden Beispielprogramme zum Lernen und Anwenden ausgeliefert. Sie können diese Beispielprogramme an Ihre eigenen Anforderungen anpassen. Das Installationsprogramm legt diese Beispiele in einem Ordner namens „Basic Samples“ im Zielverzeichnis der Installation ab.

Sie sollten mit den Grundlagen der Bedienung von Windows CE Geräten vertraut sein, ehe Sie dieses Handbuch benutzen. Es sollte Ihnen bekannt sein, wie man Anwendungen startet, den Stylus verwendet und andere Windows/CE Merkmale nutzt. Wenn Sie sich nicht mit diesen Dingen auskennen, ziehen Sie bitte Ihr Windows CE Handbuch zu Rate.

Zum Installieren der NS Basic/CE Software ist des weiteren grundsätzliches Verständnis der Bedienung eines Desktop-Computers (Windows 98/ME/NT/2K/XP) erforderlich.

Bei der Übersetzung dieses Handbuchs wurde weitestgehend auf die „Ein-Deutschung“ gängiger englischer Begriffe verzichtet. An manchen Stellen werden einzelne Worte zusammen mit der jeweiligen Übersetzung (in Klammern) dargestellt. Die Verwendung der englischen Begriffe erscheint dem Übersetzer jedoch als eindeutiger.

1.1 Was ist BASIC?

BASIC existiert schon seit nahezu 40 Jahren. In dieser Zeit wurden schon Hunderte Interpreter und Compiler für BASIC entwickelt und Berge von Programmen erstellt. Viele Bücher über BASIC wurden, und werden wohl noch weiterhin, geschrieben. BASIC Special Interest Groups existieren in den verschiedensten Formen.

BASIC tut der Seele irgendwie gut. Während Wellen von anderen Programmiersprachen kommen und gehen, läuft BASIC immer noch fast auf jedem System: ohne Standards passt es sich neuen Umgebungen leicht an und hält Schritt mit den schicken neuen Sprachen. Mit allen, die kommen und gehen.

Jeder, auch Bill Gates, hat mit BASIC angefangen. Irgendwie kommen wir alle immer wieder darauf zurück. Es ist immer noch die beste Sprache für schnell erstellte Programme und einfache Anwendungen. BASIC Interpreter, insbesondere die einfachen, können sehr viel Charme haben.

Die Hardware, auf der BASIC programmiert wird, hat sich seit dessen Entwicklung einmal um sich selbst gedreht. Die mächtige Programmiersprache, zu der nur Computerspezialisten und Programmierer von Großrechenanlagen Zugang hatten, läuft heutzutage auf einem Handheld Computer.

1.1.1 NS Basic/CE

NS Basic/CE für Windows CE ist eine echte Programmiersprache. Sie enthält alle üblichen BASIC Anweisungen in einer unkomplizierten Art und bietet darüber hinaus eine Vielzahl mächtiger Erweiterungen.

NS Basic Corporation betreibt eine Webseite unter <http://www.nsbasic.com>. Verwenden Sie ihren Webbrowser um diese Seite nach wichtigen Bekanntmachungen, technischen Informationen und NS Basic/CE Beispielprogrammen zu durchsuchen.

1.1.2 NS Basic/CE und Windows CE

Als Sie ihr Windows CE Gerät gekauft haben, dachten Sie sicherlich, dass Sie nun viele Papier-Bleistift-Aufgaben damit ersetzen können würden. Sie hofften wahrscheinlich auch, dass es als kleiner programmierbarer Computer fungieren würde. Genau für diese Zwecke wurde NS Basic/CE entworfen. Durch seinen Einsatz sind Sie in der Lage die Anwendungen, die Sie benötigen, in einer einfach zu nutzenden Programmiersprache, direkt auf Ihrem Handheld PC zu erstellen.

NS Basic/CE kann auch zur allgemeinen Programmierung verwendet werden. Jedes Programm, das in BASIC erstellt werden kann, kann ebenso in NS Basic/CE erstellt werden. Sie können individuell angepasste Datenbanken erstellen, komplexe Berechnungen durchführen oder sogar Spiele programmieren. Was NS Basic/CE hervorstechen lässt, ist seine Verfügbarkeit. Sie müssen keine neue, komplexe Sprache erlernen, nur um die mächtigen Fähigkeiten Ihres H/PC ausnutzen zu können.

1.2 System Anforderungen

Um NS Basic/CE installieren zu können benötigen Sie ein Windows CE Gerät, einen Desktop Computer, auf dem Microsoft ActiveSync installiert ist, und ein Kabel, mit dem sich das Windows CE Gerät mit dem Desktop Computer verbinden lässt. Für spezifische Informationen zu unterstützten Geräten und erforderlicher Software, lesen Sie bitte die Datei Readme.htm, die in Ihrem Produkt enthalten ist.

1.3 Installation

NS Basic/CE wird auf einer CD ausgeliefert. Sie müssen es mittels Microsoft ActiveSync auf Ihrem Windows CE Gerät installieren. Für spezifische Informationen zur Installation, lesen Sie bitte die Datei Readme.htm, die in Ihrem Produkt enthalten ist.

Sie sollten auch einen Besuch auf unserer Website machen, auf der wir die aktuellsten Informationen zu NS Basic/CE veröffentlichen. Suchen Sie unter Release Notes nach Updates, unter Tech Notes nach den letzten technischen Neuerungen und in der Downloads Sektion nach zusätzlichem Programmcode, Update-Paketen und den Runtime Modulen.

Unsere Website ist <http://www.nsbasic.com>.

1.3.1 Eingabe Ihrer Seriennummer

Ehe Sie NS Basic/CE verwenden können, müssen Sie Ihre Seriennummer eingeben. Auf Ihrem Windows/CE Gerät starten Sie bitte das BASIC Installer Programm. Auf Ihrem Desktop PC, wählen Sie Register im Hilfe-Menü. Ihre Seriennummer befindet sich auf der Rückseite Ihres Handbuches. Für weitere Informationen zur Installation, lesen Sie bitte die Datei Readme.htm.

Wenn Sie eine Registrierungskarte erhalten haben, füllen Sie sie bitte jetzt aus und senden sie an die Adresse, die auf der Karte abgedruckt ist. Sie können sich natürlich auch auf unserer Website Online registrieren.

2. NS Basic/CE Konzepte

2.1 Typografische Konventionen

In diesem Handbuch werden die folgenden typografischen Konventionen verwendet:

KEYWORDS

Großbuchstaben bezeichnen NS Basic/CE Schlüsselwörter (Keywords) und anderen Text, der genau so wie dargestellt, eingegeben werden muss. Im Zusammenhang mit diesem Handbuch, bezeichnet Text in Großbuchstaben den unbedingt erforderlichen Teil des Syntax einer Anweisung. NS Basic/CE unterscheidet nicht zwischen Groß- und Kleinschreibung: Schlüsselwörter werden als Großbuchstaben, als Kleinbuchstaben oder gemischt akzeptiert. Ein Schlüsselwort, wie beispielsweise PRINT, kann als print, Print, oder PRINT in Ihren Programtext eingegeben werden.

placeholders

Kursiver Text bezeichnet einen Platzhalter (placeholder) für Informationen, die Sie zur Verfügung stellen müssen. Im folgenden Befehl wird *expression* kursiv dargestellt, um zu zeigen, dass die EXECUTE Anweisung einen Ausdruck (*expression*) erfordert:

EXECUTE *expression*

Example

Die Schriftart Monaco (nichtproportionale Schriftart) bezeichnet Beispiel-Programmcode und Informationen, die auf Ihrem NS Basic/CE Bildschirm angezeigt werden. Das nachfolgende Beispiel zeigt eine Zeile aus einem NS Basic/CE Programm:

```
PRINT "Hello World!"
```

[Optional]

Eckige Klammern deuten an, dass die darin enthaltenen Elemente optional sind. Im folgenden Beispiel werden die eckigen Klammern verwendet um anzuzeigen, dass die Angabe eines zweiten Elementes zur Anzeige auf dem Display mittels des PRINT Befehles optional ist:

```
PRINT expression1 [ ,expression2 ]
```

Beide dieser PRINT Anweisungen sind zulässig, da der PRINT bis zu 20 Ausdrücke akzeptiert:

```
PRINT "Hello"
PRINT "Hello", "World"
```

|

Der vertikale Balken deutet an, dass die Elemente sich gegenseitig ausschliessen. Der Balken im folgenden Beispiel zeigt, dass die LEN Funktion entweder mit einer Zeichenkette (String) oder einer Variablen verwendet werden kann:

LEN(*string* | *variable*)

2.2 Die Elemente eines NS Basic/CE Programms

Ein NS Basic/CE Programm besteht aus einer Gruppe von Anweisungen. Jede NS Basic/CE Programmzeile kann aus den folgenden Elementen bestehen:

KEYWORD Argumente 'Kommentar

Ein KEYWORD oder Schlüsselwort ist ein Wort der Sprache, die NS Basic/CE interpretieren kann. Beispiele sind PRINT, INPUTBOX und IF. Die Anweisung und ihre Argumente bestimmen, welche Aktionen (falls es solche gibt) von NS Basic/CE beim Erreichen der Programmzeile durchgeführt werden.

Jeder Text, der einem ' in einer Zeile folgt, ist ein Kommentar und wird von NS Basic/CE ignoriert.

2.2.1 Mehrzeilige Anweisungen

Im Normalfall endet jede NS Basic/CE Anweisung am Ende der Befehlszeile. Wenn Sie sehr komplexe Anweisungen haben, kann diese Zeile sehr lang werden. Dadurch kann Ihr Programm schwer lesbar werden. Sie können lange Anweisungen aufteilen, wenn Sie am Ende einer Befehlszeile die sogenannte Zeilenfortführungs-Sequenz (_), ein Leerzeichen gefolgt von einem Unterstrich, verwenden. NS Basic/CE fügt die nächstfolgende Zeile an die Zeile an, die mit (_) endet. Hier ein Beispiel für die Zeilenfortführungs-Sequenz:

```
PRINT "Lange Befehle sind kein Problem in" _
      & " NS Basic/CE"
```

2.2.2 Mehrere Anweisungen pro Zeile

Durch Verwendung eines ":" Zeichens als Trennungssymbol, können Sie mehr als eine Anweisung in eine Zeile schreiben:

a=1: b=2: c=3

2.2.3 Literale, Datentypen und Variablen

Literale sind Zeichenfolgen, die zur Darstellung der Werte von Basistypen in Ihrem Programm verwendet werden. Sie benutzen sie ständig: zum Initialisieren von Variablen, zum Setzen der Start- und Stop-Werte einer FOR...NEXT Schleife, und so weiter. Wenn Sie mit der CONST Anweisung eine Konstante definieren, kann der Name der Konstanten wie ein Literal verwendet werden. Der Wert von Literalen kann nicht verändert werden.

Variablen sind Namen für Bereiche, in denen Ihre Daten abgelegt werden. Der Inhalt einer Variablen kann bei Bedarf geändert werden. Alle Variablen sind vom Datentyp Variant, mit verschiedenen Subtypen.

2.2.4 Numerische Datentypen

Es gibt verschiedene Subtypen für numerische Daten, die sich jedoch alle gleich verhalten. Sie können im Normalfall ohne Problem alle Subtypen beliebig mischen und umwandeln.

Subtyp	Größe	Bereich	Literal
Byte	8 Bit	0 to 255	2
Int	16 Bit	-32,768 to 32,767	100
Long	32 Bit	-2,147,483,648 to 2,147,483,647	500000
Single	32 Bit	$\pm 1.5 \times 10^{-45}$ to $\pm 3.4 \times 10^{38}$	1.0e100
Double	64 Bit	$\pm 5.0 \times 10^{-324}$ to $\pm 1.7 \times 10^{308}$	1.0e1000
Currency	32 Bit	-9.2×10^{14} to 9.2×10^{14}	1000000.00
Hex	32 Bit		&h000000F

2.2.5 Datentyp Boolean

Boolesche Werte können zwei Werte annehmen: TRUE (WAHR) und FALSE (FALSCH). Dieser Datentyp wird im Zusammenhang mit der IF Anweisung verwendet. Sie testet das boolesche Ergebnis eines Ausdrucks und wählt die THEN Anweisungen, wenn der Ausdruck TRUE ist, oder die ELSE Anweisungen, falls der Ausdruck FALSE ist. Die numerische Entsprechung für TRUE ist -1, die für FALSE ist 0.

2.2.6 Datentyp Farbe (Color)

Farben werden durch vorzeichenlose Long-Integer Werte im Bereich von 0 to 16,777,215 vertreten. Ein Farbwert wird durch Mischung der Beträge für Rot, Grün und Blau, die zwischen 0 für dunkel und 255 für hell schwanken können, erzeugt. Die Mischungsformel lautet:

$$\text{Farbe} = \text{Rot} + (\text{Grün} * 256) + (\text{Blau} * 65536)$$

Schwarz-Weiß-Schattierungen (siehe Tabelle) werden erzeugt, indem gleich große Beträge für Rot, Grün und Blau verwendet werden. In Anhang B finden Sie eine Liste der gängigsten Farbkonstanten.

Farbname	Rot, Grün, Blau	Farbwert
Black	0, 0, 0	0, vbBLACK
Dark gray	128, 128, 128	8,421,504
Light gray	192, 192, 192	12,632,256
White	255, 255, 255	16,777,215, vbWHITE

2.2.7 Datentyp String

Strings bestehen aus einer Folge von Zeichen. Ein String kann annähernd 2 Milliarden Zeichen lang sein. Es gibt einige Funktionen, die Strings verändern können. Der Verkettungs-Operator (&) fügt die String Darstellungen zweier Variablen zusammen. Ein literaler String wird in Anführungszeichen gesetzt:

```
"Das ist ein literaler String"
```

2.2.8 Datentyp Array

Arrays sind Container. Es sind Listen mit Werten, die unter einem einzelnen Namen abgespeichert werden. Auf einen bestimmten Wert bezieht man sich, durch eine Nummer, die in Klammern hinter dem Variablennamen steht (Index). Das erste Element eines Arrays hat immer den Index Null (0) und jedes Array kann viele Elemente haben. ARR(2) verweist auf das dritte Element im Array ARR. Jedes Element eines Arrays kann einen beliebigen Subtyp haben.

2.2.9 Variablennamen

Eine Variable ist ein Name, dem ein Wert zugeordnet werden kann. Der Name besteht aus einer Folge von alphabetischen und numerischen Zeichen, sowie dem (_) Unterstrich (underscore). In NS Basic/CE ist die Länge eines Variablennamens ungegrenzt und jedes Zeichen des Namens ist signifikant. Wie erwähnen das, weil in manchen älteren Basic-Dialekten nur kurze Namen benutzt werden können. Bei Variablennamen werden Groß- und Kleinschreibung nicht unterschieden und Leerzeichen oder andere Sonderzeichen dürfen nicht verwendet werden. Variablennamen müssen mit einem Buchstaben beginnen.

NS Basic/CE Konstanten dürfen nicht als Variablennamen verwendet werden. Eine komplette Liste der Konstanten finden Sie in Anhang B.

In der folgenden Liste finden Sie einige Variablennamen, die in NS Basic/CE erlaubt sind:

```
text
LLAMAS           // entspricht llamas
oder Llamas
Jupiter
WlSpec
SouthPark
```

Und manche sind nicht erlaubt:

```
1table           // beginnt mit einer
Nummer
X&Ycords         // benutzt
Sonderzeichen &
first counter    // enthält ein
Leerzeichen
%correct         // beginnt nicht mit
einem Buchstaben
print            // NS Basic/CE Keyword
```

2.3 Ausdrücke und Operatoren

Ein Ausdruck ist ein Literal, eine Variable, eine Formel oder ein Funktionsaufruf, die ein Ergebnis liefert. Hier einige Beispiele für Ausdrücke:

```
6/3              // Ergebnis ist 2
5+6/3            // Ergebnis ist 7
"This " & "that" // Ergebnis ist "This
that"
```

Ein String-Ausdruck kann ein String-Literal, eine String-Variable, oder das Ergebnis einer Kombination aus String-Literalen, String-Variablen und Substrings sein. Ganz ähnlich kann ein numerischer Ausdruck eine numerische Konstante, eine numerische Variable oder das Ergebnis einer Funktion/Variablen, die einen einzelnen Wert erzeugt, sein.

2.3.1 Arithmetische Operatoren

NS Basic/CE erlaubt die folgenden arithmetischen Operatoren in absteigender Priorität:

()		Klammer
^		Exponent
*	/ \	Multiplikation und Division
+	-	Addition und Subtraktion

Klammern können zum ändern der Auswertungsreihenfolge genutzt werden.

```
PRINT 2 + 3 * 4  
PRINT (2 + 3) * 4
```

Ausgabe

```
14  
20
```

NS Basic/CE unterstützt Fließkomma-Arithmetik. Die MOD Funktion kann verwendet werden, um den Rest einer Division zu erhalten. Der Backslash-Operator (\) wird für die Integer-Division (ganzzahlige Division) verwendet.

Arithmetische Operatoren können nur in numerischen Ausdrücken verwendet werden. Sie dürfen nicht auf Strings angewendet werden.

2.3.2 Vergleichs-Operatoren

Vergleichs-Operatoren vergleichen zwei Werte miteinander und geben einen booleschen Wert, FALSE oder TRUE, zurück. Dieses Ergebnis kann zum Ändern des Programmablaufes genutzt werden. Vergleichs-Operatoren haben eine geringe Priorität als arithmetische Operatoren. Die Vergleichs-Operatoren sind:

=	gleich
<>	Nicht gleich
<	Kleiner als
>	Größer als
<=	Kleiner oder gleich
>=	Größer oder gleich

In der SET Anweisung, wird das Gleichheitszeichen zum Zuweisen eines Wertes an eine Variable verwendet, nicht als Vergleichs-Operator.

2.3.3 Boolesche Operatoren

Boolesche Operatoren verbinden Ausdrücke miteinander und geben dabei ein TRUE oder FALSE als Antwort zurück. Arithmetische und Vergleichs-Operatoren werden vor Booleschen Operatoren ausgewertet. Der NOT Operator bezieht sich auf einen Ausdruck, alle anderen Operatoren erfordern zwei Ausdrücke.

Die Booleschen Operatoren sind:

AND	Gibt TRUE zurück, wenn beide Ausdrücke TRUE sind.
EQV	Gibt TRUE zurück, wenn beide Ausdrücke TRUE oder beide Ausdrücke FALSE sind.
IMP	Gibt TRUE zurück, wenn der erste Ausdruck den zweiten Ausdruck impliziert.
OR	Gibt TRUE zurück, wenn einer der beiden Ausdrücke oder beide TRUE ist.
NOT	Gibt TRUE zurück, wenn der Ausdruck FALSE ist und gibt FALSE zurück, wenn der Ausdruck
XOR	Gibt TRUE zurück, wenn beide Ausdrücke nicht den gleichen booleschen Wert haben.

Boolesche Operatoren können mit jedem Ausdruck verwendet werden, der einen booleschen Wert zurückgibt.

2.4 FUNKTIONEN und SUB Prozeduren

Prozeduren sind Blöcke von Programmanweisungen, die von anderen Anweisungen, optional mit Übergabe von Werten (Argumenten), unter bestimmten Bedingungen und wiederholt aufgerufen („called“) werden können. Wenn eine Prozedur aufgerufen wird, verhält sie sich genau wie eine Funktion oder eine Anweisung, die zusammen die Sprache NS Basic/CE bilden. FUNKTIONEN geben ein Ergebnis zurück, das in einer Variablen gespeichert, oder in anderen Anweisungen weiter verwendet werden kann. SUB Prozeduren laufen ab, ohne einen Wert zurück zu geben. Wenn eine FUNKTION aufgerufen wird und der Rückgabewert nicht verwendet wird, behandelt NS Basic/CE sie wie eine SUB Prozedur.

Zur Übergabe von mehreren Argumenten an eine FUNKTION, verwenden Sie eine durch Kommata getrennte, in Klammern eingeschlossene Liste. Um mehrere Argumente an eine SUB Prozedur zu übergeben, nutzen Sie eine durch Kommata getrennte Liste, ohne Klammern. Wenn ein einzelnes Argument an eine FUNKTION oder SUB Prozedur übergeben wird, können ebenfalls Klammern genutzt werden; Klammern, die ein einzelnes Argument einschließen, werden von NS Basic/CE zum Auswerten eines Ausdrucks, nicht zum Kennzeichnen einer Liste von Argumenten genutzt.

2.5 Projekte, Module, Forms (Formulare), Objekte und Steuerelemente

Steuerelemente und Objekte

Ein Steuerelement ist ein ausführbares Modul, auf das von anderen Programmen aus zugegriffen werden kann. Jede konkrete Ausprägung (Instanz) eines Steuerelementes in Ihrem Programm wird als Objekt bezeichnet. Um Ihrem Programm ein Objekt hinzu zu fügen, verwenden Sie die ADDOBJECT Anweisung. Objekte haben Eigenschaften, die abgefragt und gesetzt werden können, Methoden, die als FUNKTIONEN oder SUB Prozeduren aufgerufen werden können und Ereignisse, die durch Benutzeraktionen, das Betriebssystem oder andere Programme, ausgelöst werden können.

Zusammen mit NS Basic/CE werden zahlreiche Steuerelemente installiert. Dabei sind eine Checkbox (Ankreuzfeld), eine Combobox (Kombinationsfeld), eine Textbox (Textfeld) und weitere gebräuchliche Objekte. Im Zusammenhang mit diesen Objekten spricht man auch von immanenten Objekten, da sie nicht von der Existenz einer anderen Datei abhängig sind, sondern direkt mit der NS Basic Runtime Datei zur Verfügung gestellt werden.

Eine zweite Gruppe von Steuerelementen, die Standard-Steuerelemente genannt wird, ist in jeder NS Basic Installation vorhanden. Es handelt sich dabei um ActiveX Steuerelemente, die von Microsoft ausgeliefert werden, so z.b. die PictureBox (Bildfeld), das Grid (Gitter) und das FileSystem (Dateisystem). Jedes davon ist in einer separaten Datei vom Typ .dll abgelegt.

ActiveX Steuerelemente von Drittanbietern können ebenfalls in NS Basic/CE genutzt werden. Es gibt einige davon in NS Basic's Big Red Toolbox. Um mehr darüber zu erfahren, lesen Sie die Tech Note 1 oder suchen Sie im Internet danach.

Alle Installationen von NS Basic/CE beinhalten das PictureBox Steuerelement. Der Hintergrund des Ausgabefensters, das als Ausgabe-Objekt bezeichnet wird, ist eine PictureBox. Zusätzlich zum Ausgabe-Objekt wird automatisch in jedem laufenden NS Basic/CE Programm ein Err-Objekt (Fehler-Objekt) erzeugt, das optional zur Fehlerbehandlung verwendet werden kann.

Objekt Ereignisse

Ein Ereignis ist ein Programmaufruf, den ein Objekt an Ihr Programm richtet, als Reaktion auf eine Aktion, die an diesem Objekt ausgeführt wurde. Wenn Sie beispielsweise in Ihrem Programm eine Kommandotaste haben, wird ein Ereignis ausgelöst, wenn der Anwender diese Taste anklickt.

Wenn Ihre Taste den Namen "MyButton" hat, wird NS Basic dann die Funktion MyButton_Click() in Ihrem Programm aufrufen. Wenn Ihr Programm eine solche Funktion nicht enthält, wird das Ereignis ignoriert.

Ein anderes Beispiel wäre, wenn Sie die serielle Kommunikation verwenden würden. Eingehende Daten würden das OnComm Ereignis auslösen. Sobald Sie auf Ihr Kommunikations-Objekt "Comm" zugreifen würden, würde es versuchen, die Funktion Comm_OnComm() aufzurufen.

Forms (Formulare)

NS Basic/CE's Implementierung von Formularen ist einfacher als die auf dem Desktop. Obwohl das zugrunde liegende System das Konzept der Formulare nicht kennt, existiert eine Technik, die diese Funktionalität zur Verfügung stellt.

Ein Objekt, das einmal erstellt wurde, kann nicht wieder gelöscht werden. In Windows CE ist das ohnehin bedeutungslos: die Verwaltungsinformationen eines einzelnen Objektes sind so gering, dass es kein Problem darstellt, alle Objekte bis zum Ende Ihres Programm im Speicher zu halten.

Ein Form ist deshalb nur eine Gruppe von Objekten, die zu einem bestimmten Zeitpunkt angezeigt oder verborgen werden. Sie können auf jedes Objekt in einem Formular über ein Array zugreifen. Ein Form kann ausgeblendet werden, indem man dieses Array durchläuft und jedes Objekt, auf das es verwiesen wird, einzeln ausblendet.

Sie können das selbst durchführen, aber wenn Sie die Desktop IDE oder den Visual Designer verwenden, kümmert sich NS Basic/CE darum, das der erforderliche Code zum erstellen, ausblenden und anzeigen des Formulars erzeugt wird.

Forms basieren auf dem PictureBox Objekt und haben auch die gleichen Eigenschaften.

Form Methoden

Jedes Formular hat eine Ausblenden- und eine Anzeigen-Methode. Wenn Ihr Form den Namen "Form1" hat, wird es mit dem folgenden Code in Ihrem Programm ausgeblendet:

```
Form1_Hide
```

Zum Anzeigen des Forms verwenden Sie Form1_Show. Wenn das Formular zum ersten Mal angezeigt wird, erstellt NS Basic/CE die Objekten für dieses Form. Während das Formular erzeugt wird, ist die Variable Form1(0) auf falsch gesetzt. (sofern Ihr Form Form1 heißt). Sobald alle Objekte erzeugt wurden, wird Form1(0) auf wahr gesetzt. Sie können durch Abfragen dieser Variable feststellen, ob ein Formular

vollständig erzeugt wurde, bevor Sie auf Ereignisse reagieren, die von einem der Objekte auf dem Form erzeugt worden sein könnten.

Form Ereignisse

Wenn Ihr Programm mit der Desktop DIE erstellt worden ist, wird ein form load event an Ihr Programm geschickt. Nehmen wir an, Ihr Formular wird Form1 genannt, dann wird die Subroutine Form1_Load() aufgerufen, sobald alle Objekte auf dem Form durch Form1_Show erzeugt wurden. Ganz ähnlich wird ein Form1_Unload Ereignis gesendet, wenn das Formular geschlossen wird.

Programm Eigenschaften und Ereignisse

NS Basic Programme laufen innerhalb eines PictureBox-Objektes, das Output-Objekt genannt wird ab. Die unterstützten Eigenschaften und Ereignisse finden Sie im Kapitel über das PictureBox Steuerelement.

```
REM Applikation Hintergrund auf Blau
umstellen
Output.backcolor=vbBlue
```

Programm Ereignisse

Wenn ein Programm beendet wird (oder auf einem Pocket PC minimiert wird), wird ein Output_Close Ereignis an Ihr Programm gesendet. Dieses Ereignis wird an alle Programme gesendet, unabhängig, ob sie mit der Desktop IDE erstellt wurden oder nicht. Diese Eigenschaft ist besonders für Pocket PC Geräte sehr hilfreich. Microsoft's Pocket PC Richtlinien zur Anwenderschnittstelle legen fest, dass die oberste rechte Taste einer Anwendung, die eine Applikation in anderen Versionen von Windows schließt, die Anwendung nur minimalisieren soll, d.h. sie soll weiterhin laufen. Um dieses Verhalten in Ihrer Anwendung zu überschreiben, fügen Sie Ihrem Programm-Code folgende Zeilen zu:

```
REM SchlieÙe Applikation beim minimalisieren
Sub Output_Close()
    Bye
End Sub
```

Sie können auch die ShowOKButton Anweisung verwenden, um dieses Verhalten zu ändern.

Wenn die Größe des Ausgabefensters verändert wird, z.b. wenn das Fenster gedreht wird, wird ein Output_Size Ereignis an Ihr Programm geschickt.

Module

Ein Modul ist eine .txt, .cod oder .bas Datei, die in Ihrem Projekt enthalten ist. Man kann sie verwenden, um den Programm-Code in kleinere Teile auszuteilen, die leichter verwaltet werden können. Module können auch in mehr als einem Projekt verwendet werden, so lässt sich Programmcode während der Entwicklung gemeinsam nutzen. Zur Laufzeit werden alle Projekt-Module zu einem einzelnen laufenden Programm zusammengefügt. Aller Code aus Modulen wird an das Ende des Hauptprojektes gestellt.

3. Programmieren auf dem Windows/CE Gerät

3.1 Übersicht

NS Basic/CE 's Entwicklungsumgebung zum Programmieren auf dem Windows/CE gerät bietet zwei verschiedene Ansichten, den Programm-Editor und das Ausgabe-Fenster. Sie besitzt außerdem ein Hilfsmittel mit Namen Visual Designer, eine Möglichkeit Programmcode per Grafik zu erzeugen.

3.2 Programm-Editor

Der Programm-Editor ist das Werkzeug, mit dem Sie Ihre Programme schreiben. Er hat einen weißen Hintergrund mit einer vertikalen Bildlaufleiste und eine Werkzeugleiste, mit der sich NS Basic/CE bedienen lässt. Der Menübereich der Werkzeugleiste ist nachfolgend in Tabelle 1 skizziert.

Tabelle 1: NS Basic/CE Menüs






Menü	Befehl	Beschreibung
File	New	Neues Programm
	Open	Programm laden
	Save	Programm speichern
	Save As...	Programm unter neuem Namen speichern
	Encryption ON/OFF	Setzt den Verschlüsselungsstatus des gespeicherten Programms. Dadurch wird die Möglichkeit geschaffen, dass der Programmcode nur mit der NS Basic/CE Kopie desjenigen sichtbar ist, der ihn verschlüsselt hat.
	1...4	Letzte vier Programme
	Exit	NS Basic/CE beenden



Tabelle 1: NS Basic/CE Menüs

Menü	Befehl	Beschreibung
Edit	Undo	Letzte Aktion
	Cut	Selektierten Text ausschneiden
	Copy	Selektierten Text kopieren
	Paste	Inhalt der Zwischenablage einfügen
	Select All...	Gesamten Programmtext selektieren
	Find...	Suche durchführen
	Find Again	Letzte Suche wiederholen
	Find All...	Alle Übereinstimmungen finden
	Goto Line...	Gehe zu Zeile... im Editor, wenn
	Overview	Alle Prozeduren auflisten
Tools	Format	Programm neu formatieren
	Run	Ein Programm ausführen
	...Run	Das Programm... ausführen
	..Execute	Neu starten bei Funktion...
	..Trace	Fehlerprotokollierung starten
	...Step	Befehle Schritt für Schritt abarbeiten
	Execute	Temporäres Programm eingeben
	Show	Wert der Variablen... anzeigen
	Stats...	Informationen über Dateigröße
	Visual	Visual Forms Editor
Help	Help Topics...	Hilfe-Browser
	About NS	Titel, Version, Kontakt

Die Werkzeug-Symbole befinden sich rechts neben den Menü-Einträgen.

Tabelle 2: Werkzeug Symbole

Symbol	Menü-Element	Tastaturkürzel
	"File->New"	Ctrl+N
	"File->Save"	Ctrl+S
	"File->Open"	Ctrl+O
	"Edit->Cut"	Ctrl+X
	"Edit->Copy"	Ctrl+C

Symbol	Menü-Element	Tastaturkürzel
	"Edit->Paste"	Ctrl+V
	"Tools->Run"	Ctrl+R

Zwei weitere Tasten befinden sich in der rechten oberen Ecke des Programm-Editor Fensters. Die Hilfe-Taste öffnet den Hilfe-Browser mit der Online-Hilfe von NS Basic/CE; sie ist ein Kürzel für "Help->Help Topics..."



Die Close-Taste beendet NS Basic/CE und ist somit ein Kürzel für "File->Exit".



3.3 Ausgabe-Fenster

Das Ausgabe-Fenster wird angezeigt während Ihr Programm ausgeführt wird. In ihm werden alle Ein- und Ausgaben durchgeführt. Es hat einen hellgrauen Hintergrund und in der rechten oberen Ecke befindet sich eine Close-Taste (abhängig von der Windows CE Version). Eine schmale Linie in der oberen linken Ecke deutet an, wo ein Programm mittels des SETMENU Befehls ein eignes Menü hinzufügen kann. Die Close-Taste schließt das Ausgabe-Fenster; wurde das Programm aus dem Programm-Editor heraus gestartet, geht die Kontrolle an den Editor zurück, ansonsten wird NS Basic/CE beendet.

3.4 Visual Designer

Der Visual Designer ist ein Teil von NB Basic/CE, einer voll ausgestatteten BASIC Entwicklungsumgebung. Er ermöglicht es Ihnen, Formulare visuell auf Ihrem Windows CE Gerät zu erstellen. Er wird aus der NS Basic/CE Umgebung heraus gestartet.

Um einem Formular ein Objekt hinzuzufügen, selektieren Sie das gewünschte Objekt im Menü Objects. Es wird dann in der oberen linken Ecke Ihrer Form kreiert. Sie können es verschieben und in der Größe verändern um seine Position

festzulegen. Klicken Sie doppelt darauf und der Eigenschaften-Editor für dieses Objekt wird angezeigt. Sie können jede Eigenschaft dieses Objektes an Ihre Bedürfnisse anpassen.

Wenn Sie den Visual Designer schließen, werden Ihre Angaben in NS Basic/CE Programm-Code umgewandelt und Sie gelangen wieder zum normalen NS Basic/CE Editor-Fenster. Ihr Code kann dann sofort ausgeführt werden.

Sie können den Visual Designer jederzeit wieder aufrufen und weitere Änderungen an Ihrem Formular durchführen, selbst dann, wenn inzwischen andere Programmteile geändert wurden.

WARNUNG! Wenn Sie Code ändern, der vom Visual Designer generiert wurde, kann es sein, dass Sie ihn später nicht mehr mit dem Visual Designer bearbeiten können. Es kann auch geschehen, dass Ihre Änderungen verloren gehen, wenn Sie den Code neu generieren lassen.

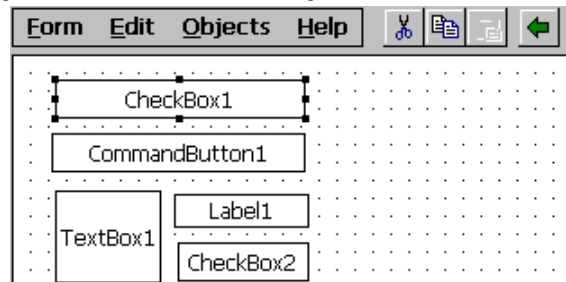


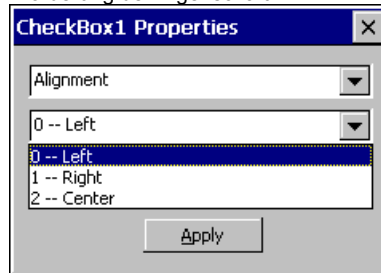
Tabelle 3: Visual Designer Menüs

Form	Add Form	Dem Programm ein neues Form hinzufügen.
	Delete	Ein Formular löschen.
	Set Default	Vorgabe auf anderes Formular einstellen.
	Exit	Zurück zum NS Basic Editor.
Edit	Undo	Letzte Aktion rückgängig machen
	Cut	Objekt ausschneiden
	Copy	Objekt kopieren
	Paste	Zwischenablage einfügen

	Clear	
	Select Object	Liste von selektierbaren Objekten anzeigen.
	Properties	Eigenschaften des aktuellen Objekts bearbeiten. Wurde kein Objekt selektiert, werden die Formular-Eigenschaften verwendet. Siehe Eigenschaften-Editor.
Object	Add Menu	Dem aktuellen Form ein Menü hinzufügen. Siehe Menü-Editor.
	Edit Menu	Menü auf dem aktuellen Form bearbeiten. Siehe Menü-Editor.
	Show Program	Programm-Code anzeigen
	Snap to Grid	Bewegte Objekte sollen am Gitter einrasten
	Grid Size...	Gitterabstand auf 4-20 Pixel setzen
	CheckBox	Checkbox-Objekt hinzufügen.
	ComboBox	ComboBox-Objekt hinzufügen.
	CommandButton	CommandButton-Objekt hinzufügen.
	Date	Date-Picker-Objekt hinzufügen.
	Label	Label-Objekt hinzufügen.
	ListBox	ListBox-Objekt hinzufügen.
	OptionButton	OptionButton-Objekt hinzufügen.
	TextBox	TextBox-Objekt hinzufügen.
	Time	Time-Picker-Objekt hinzufügen.
Help	Help Topics	Hilfethemen
	About VNSB	Über Visual Designer

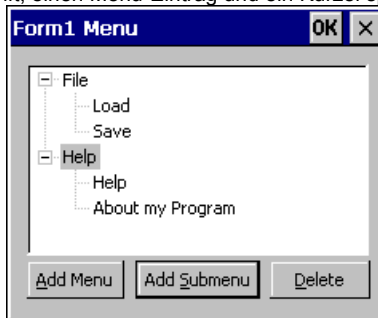
3.4.1 Eigenschaften-Editor

Jeder Objekttyp hat seine eigene Liste von Eigenschaften. Die obere ComboBox wird zum auswählen der Eigenschaft, die bearbeitet werden soll, benutzt. Die untere ComboBox zeigt eine Liste der erlaubten Werte oder ein Eingabefeld an, je nach Anforderung der Eigenschaft.



3.4.2 Menü-Editor

Menü können mit mehreren Stufen von Untermenüs aufgesetzt werden. Für jedes Menü können sie eine Überschrift, einen Menü-Eintrag und ein Kürzel spezifizieren.



3.4.3 Anmerkung

1. Die ComboBox und die ListBox haben eine spezielle Eigenschaft mit dem Namen List. Jede Zeile, die in die angezeigte EingabeBox eingegeben wird, wird zu einer separaten Auswahl des entsprechenden Objekts. Wenn Sie die Zeile mit einem einzelnen Single-Quote (') Zeichen beenden, wird sie als auszuwertender Ausdruck angesehen.

3.5 Die NS Basic/CE Programmier Umgebung

NS Basic/CE stellt eine vollständige BASIC Programmier Umgebung für Windows CE dar. Um Ihnen die Möglichkeiten dieser Umgebung aufzuzeigen, werden wir zusammen ein Beispielprogramm in NS Basic/CE erstellen. Jeder Schritt in diesem Prozess wird Sie mit einer neuen Eigenschaft dieser Umgebung vertraut machen. Starten Sie nun NS Basic/CE und folgen Sie uns!

3.5.1 Ein Programm erstellen

Wenn Sie ein neues Programm erstellen wollen, muss das Ausgabe-Fenster geschlossen und der Programm-Editor leer sein. Falls das Ausgabe-Fenster offen ist, klicken Sie die Close-Taste in der rechten oberen Ecke an und wählen Sie dann "New" im "File" Menü an.

Das Beispielprogramm wird einfach sein. Es wird den Gesamtbetrag eines Einkaufs durch Addition der Einzelposten und anfallende Steuern berechnen. Wir werden ein paar Fehler einbauen, um das Debuggen zu üben.

Geben Sie folgendes Programm ein:

```
REM Einkaufsrechner
OPTION EXPLICIT
'Konstanten deklarieren
APPTITLE = "Einkaufsrechner"
'Variable deklarieren
DIM SubTotal, TaxRate, ItemCost, Item

'Variablen initialisieren
SubTotal = 0
Item = 1

'Kosten der Einzelposten abfragen
DO
    ItemCost = INPUTBOX("Posten " & Item & _
        " Kosten", APPTITLE, 0)
    IF LEN(ItemCost) AND NOT ItemCost = 0 THEN
        PRINT FORMATCURRENCY(ItemCost)
        SubTotal = SubTotal + ItemCost
        Item = Item + 1
    END IF
LOOP WHILE LEN(ItemCost) AND NOT ItemCost = 0
IF SubTotal = 0 THEN BYE
```

```
'Ausgabe
PRINT FORMATCURRENCY(SubTotal), ,
"Zwischensumme"
TaxRate = INPUTBOX("Steuersatz (19% = 19)",
— APPTITLE, 0)
PRINT FORMATCURRENCY(SubTotal*TaxRate), , _
"Steuern"
SubTotal = SubTotal * (1 + TaxRate)
PRINT FORMATCURRENCY(SubTotal), ,
"Gesamtsumme"
```

ANMERKUNG: Der InputBox-Befehl im obigen Beispiel wird auf einem Pocket PC Gerät nicht ohne Extra-Code funktionieren. Bitte schauen Sie sich in diesem Fall das Dokument [Readme.htm](#) an.

3.5.2 Editieren eines Programms

Der Programm-Editor arbeitet wie ein normaler Text-Editor. Ein blinkender Cursor markiert die Stelle im Text, an der neuer Text hinzugefügt wird. Der Cursor kann mit den Pfeiltasten, den Tasten Page-Up bzw. Page-Down, der Home-Taste, der End-Taste oder durch Anklicken eines berührungsempfindlichen Feldes mit dem Stylus bewegt werden.

Die Funktionen Cut, Copy, und Paste ermöglichen es Ihnen, markierten Text zu verschieben oder zu löschen. Zum Markieren tippen Sie mit dem Stylus auf den Bildschirm und ziehen ihn in dieser Position in die gewünschte Richtung oder Sie positionieren den Cursor an einem Ende des zu markierenden Bereiches und navigieren mit festgehaltener Shift-Taste per Keyboard zum anderen Ende. "Cut", "Copy", und "Paste" finden Sie im "Edit"-Menü.

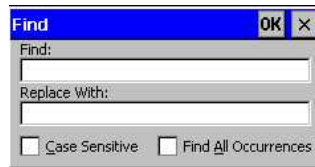
Um die letzte Änderung an Ihrem Programm rückgängig zu machen, wählen Sie bitte "Undo" aus dem "Edit"-Menü. Die maximale Größe eines Programmtextes in diesem Editor beträgt 1,048,575 Bytes: größere Programme können aber mit dem Desktop-Editor erstellt und bearbeitet werden.

3.5.3 Formattieren eines Programms

Um Ihr Programm leserlicher zu machen, benutzen Sie am besten Leerzeichen oder Tabulatoren zum Einrücken zusammen gehörender Befehle. Man macht das für gewöhnlich mit Befehlen innerhalb einer Prozedur oder Schleife. Die Standard-Tabulatorweite und -Einrückung in NS Basic/CE beträgt zwei Leerezeichen. Um Ihr Programm automatisch neu formatieren zu lassen, wählen Sie "Format" im "Tools" Menü an.

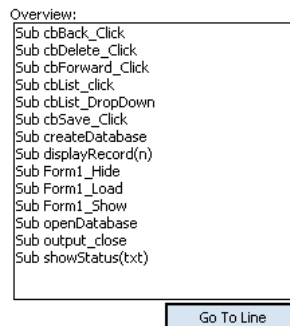
3.5.4 Finden und Ersetzen

Den Rumpf Ihres Programms können Sie auf zweierlei Arten durchsuchen. Mit Find finden Sie die Übereinstimmung, die der aktuellen Cursorposition am nächsten liegt; Find Next wiederholt die letzte Suche und kann zum sukzessiven Auffinden aller Übereinstimmungen mit dem Suchbegriff verwendet werden. Wenn auf diese Weise eine Übereinstimmung gefunden wurde, wird sie farbig hinterlegt und der Programm-Editor blättert (falls nötig) bis zur Fundstelle um sie anzuzeigen. Der Suchbegriff kann sämtliche Zeichen enthalten und optional Groß- bzw. Kleinschreibung unterscheiden. Find All Occurrences ermöglicht es Ihnen einen Suchbegriff einzugeben und liefert als Ergebnis eine Liste aller Zeilennummern des aktuellen Programms, die diesen Begriff enthalten. "Find" und "Find Next" befinden sich im "Edit" Menü.



3.5.5 Overview (Übersicht)

Der Overview Befehl zeigt eine Liste aller FUNCTION und SUB Prozeduren an. Die Prozeduren werden zunächst nach Typ (FUNCTION oder SUB), dann alphabetisch sortiert. Die Overview Anzeige kann auch verwendet werden, um zu einer selektierten Prozedur zu navigieren. Selektieren Sie eine Prozedur und klicken Sie die "Goto Line" Taste an oder klicken Sie doppelt auf eine Prozedur um die Overview Anzeige zu schließen und die Prozedur im Programm-Editor anzuzeigen.



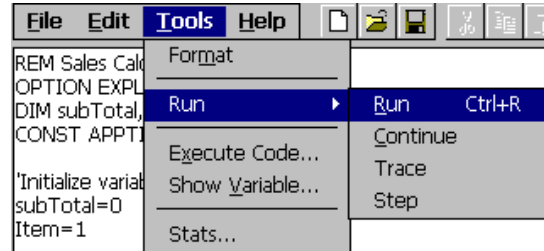
3.5.6 Ausführen eines Programms

Um Ihr Programm auszuführen, wählen Sie "Run" aus dem "Tools" Menü oder klicken das Run Symbol an.

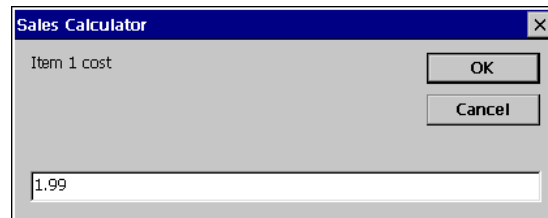


Lassen Sie uns nun unser Beispielprogramm mit drei Posten (\$1.99, \$2.99, \$3.99) und einem Steuersatz von 6% ausführen.

Programm ausführen...



Eingabe des ersten Postens...



Programmausgabe...

Program	
\$1.99	
\$2.99	
\$3.99	
\$8.97	Sub Total
\$53.82	Tax
\$62.79	Total

3.5.7 Debuggen eines Programms

Der Saldo, den wir errechnet haben, ist weit größer als die Summe der einzelnen Posten und des korrekten Steuerbetrags. Es sieht so aus, als hätten wir einen Fehler in unserem Programm. In NS Basic/CE gibt es drei

verschiedene mögliche Fehlerarten: Fehler während der Kompilierung, Fehler zur Laufzeit und logische Fehler.

Ein Kompilierungsfehler tritt auf, wenn die Syntax Ihres Programms fehlerhaft ist. Beispiele die zu Kompilierungsfehlern führen sind: fehlerhaft geschriebene Schlüsselwörter, der Aufruf einer SUB Prozedur mit Klammern und fehlende, erforderliche Argumente.

Wenn NS Basic/CE während der Ausführung eines Programms einen Befehl entdeckt, den nicht ausführen kann oder den es nicht versteht, ist ein Laufzeitfehler die Folge. Beispiele für Befehle, die nicht ausgeführt werden können: Teilen durch Null, Multiplizieren von Zeichenketten und Zugriff auf Objekte, die nicht erzeugt wurden.

Unverständliche Befehle sind beispielsweise: Befehle mit falsch geschriebenen Schlüsselwörtern, Befehle über mehrere Zeilen, die nicht die Verbindungssequenz Leerzeichen-Unterstrich (_) verwenden und Kommentare, die nicht mit einem REM oder einem Apostroph (') beginnen.

Logische Fehler treten auf, wenn Ihr Programm-Algorithmus fehlerhaft ist und das Programm ohne Fehlermeldung abläuft. Der Fehler (oder die Fehler) kann nur anhand falscher oder unerwarteter Ergebnisse erkannt werden. Wenn Sie demnächst vertraut mit der Programmierung in NS Basic/CE sind, dürften dies die einzige Art von Fehlern sein, die Ihr Programm generiert. Im Gegensatz zu Laufzeitfehlern, kann NS Basic/CE logische Fehler nicht erkennen.

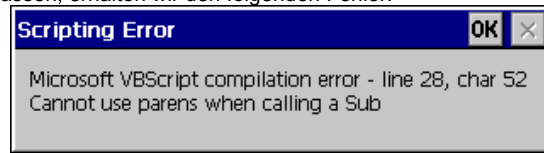
Logische Fehler sind verbreitet und unser Fehler ist auch ein solcher. Wir erhielten keine Fehlermeldung während das Programm ablief, aber die angezeigte Summe war zu hoch. Wir müssen den Programmfehler bereinigen (Debuggen)!

Der einfachste Weg um einen logischen Fehler zu beseitigen, ist es, während der Laufzeit mehr Informationen aus dem Programm heraus zu bekommen. Die beiden einfachsten Methoden dafür sind der PRINT Befehl und die MSGBOX Funktion.

Wenn wir die Ausgabe des Programms anschauen, erkennen wir, dass die Zwischensumme richtig errechnet wurde. Wir wissen also, das unser Programm den Steuerbetrag falsch errechnet. Lassen Sie uns die MSGBOX Funktion mit dem Wert des Steuersatzes und dem Faktor, der auf die Zwischensumme angewendet wird, aufrufen. Fügen Sie am Ende des Programms folgende Zeile zu:

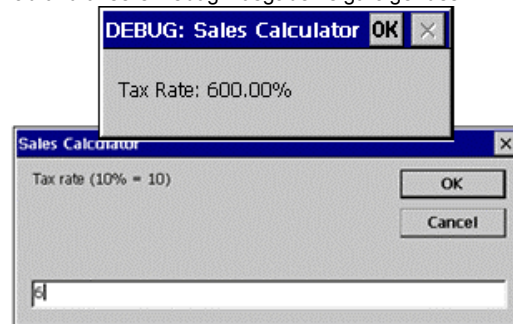
```
MSGBOX("Steuersatz: " _  
      & FORMATPERCENT(TaxRate), 0, "DEBUG: " _  
      & APPTITLE)
```

Wenn wir versuchen, das Programm wieder ablaufen zu lassen, erhalten wir den folgenden Fehler:



Dieser Kompilierungsfehler erfolgt aufgrund fehlerhafter Syntax. Wird das Ergebnis einer FUNCTION Prozedur, in unserem Fall der MSGBOX, keiner Variablen zugewiesen, so behandelt NS Basic/CE den Aufruf als SUB Prozedur. Bei SUB Prozeduren dürfen die Argumente nicht in Klammern übergeben werden, wir müssen sie also aus der gerade eingegebenen Zeile entfernen.

Sobald wir das getan haben, läuft das Programm ohne Fehler ab und unsere Debug-Ausgabe zeigt folgendes:

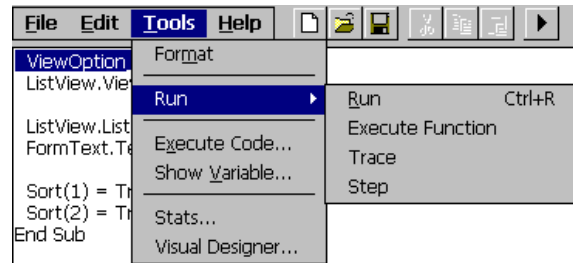


Wir erkennen daraus, dass der Steuersatz nicht richtig berechnet wird, er muss durch 100 dividiert werden. Sobald wir die Ausgabe Sektion des Programms auf die nachfolgend dargestellte Sektion angepasst haben, läuft das Programm fehlerfrei und errechnet das richtige Ergebnis. Zusätzlich haben wir der Ausgabe ein wenig zusätzliche Information durch das Anzeigen des Steuersatzes hinzugefügt.

```
'Ausgabe
PRINT FORMATCURRENCY(SubTotal), , "Sub
Total"
TaxRate = INPUTBOX("Steuersatz (19% = 19)",
- APPTITLE, 0) / 100
PRINT FORMATCURRENCY(SubTotal * TaxRate), ,
"Steuern (" & FORMATPERCENT(TaxRate) & ") "
SubTotal = SubTotal * (1 + TaxRate)
```

```
PRINT FORMATCURRENCY(SubTotal), ,
"Gesamtsumme"
```

3.5.8 BREAK, Execute Funktion, Trace und Step

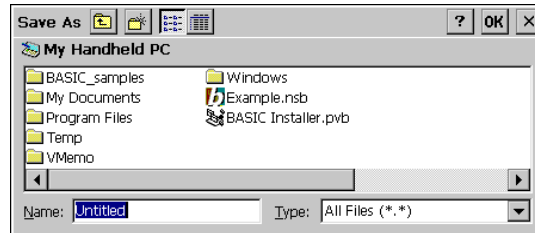


Diese Werkzeuge zusammen ergeben eine mächtige Möglichkeit zum Debuggen. Verwenden Sie den Break Befehl, um einen Haltepunkt in Ihrem Programm zu setzen. Wenn Sie während des Ablaufs des Programms jede einzelne Programmzeile angezeigt haben möchten, starten Sie das Programm per Step anstelle von Run. Dadurch wird automatisch nach jedem Befehl ein Break ausgeführt, so dass Sie die Möglichkeit bekommen, Schritt für Schritt, einen Befehl nach dem anderem, durch Ihr Programm zu gehen. Trace und Step funktionieren nicht richtig in Verbindung mit der OPTION EXPLICIT Anweisung, falls diese nicht die erste Anweisung des Programms ist.

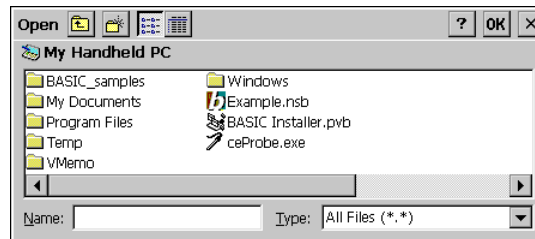
Execute Function zeigt eine Liste von Funktionen und Subroutinen an. Klicken Sie eine davon an, um sie direkt auszuführen. Das ist nützlich zum direkten Debuggen einer Funktion.

3.5.9 Speichern und Laden eines Programms

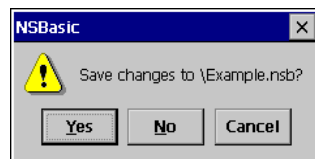
Wenn Sie ein neues Programm erstellen (über "File->New"), existiert es nur im Programm-Editor. Wenn Sie NS Basic/CE beenden oder ein anderes Programm laden wollen, möchten Sie Ihr aktuelles Programm wahrscheinlich abspeichern. Dazu müssen Sie "Save" aus dem "File" Menü anwählen. Es wird eine Dialogbox angezeigt in der Sie aufgefordert werden, einen Namen für die Datei, in der Ihr Programm gesichert werden soll, einzugeben. Lassen Sie uns unser Programm unter dem Namen SalesCalculator.nsb sichern.



Wenn Sie ein bereits gespeichertes Programm ändern oder ausführen wollen, müssen Sie es mit "Open" aus dem "File" Menü laden. Lassen Sie uns das gerade gespeicherte Beispielprogramm wieder laden.



Immer wenn NS Basic/CE dabei ist, ein Programm zu schließen, das modifiziert worden ist, werden Sie aufgefordert, anzugeben, ob Sie die Änderungen speichern möchten.



3.5.10 Speichern und Laden von Programmen als Textdateien

Wenn Sie Ihren Programmcode in anderen Anwendungen oder einem Desktop PC verwenden möchten, können Sie ihn als Textdatei abspeichern. Im Save As Dialog, fügen Sie dem Namen ein .txt an. Sie können auch Textdateien laden.

4. Programmieren auf dem Desktop PC

NS Basic/CE ermöglicht es Ihnen Programme auf dem Desktop PC zu erstellen, die auf Ihrem Windows CE Gerät ablaufen können. Mittels Virtual CE können Sie auf Ihrem Desktop ansehen, wie das Programm auf Ihrem Windows/CE Gerät ablaufen würde.

4.1 Menü Optionen

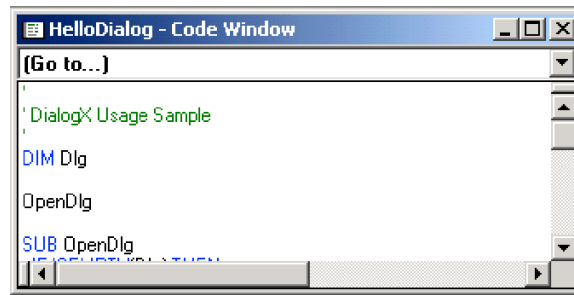
Datei	
Neues Projekt	Erstellen eines neuen Projekts, das aktuelle wird vorher geschlossen.
Projekt öffnen	Anzeigen des Öffnen Dialogs für bestehende und zuletzt bearbeitete Projekte. Siehe Öffnen Dialog für weitere Informationen.
Projekt speichern	Projekt sichern. Wenn das Projekt neu ist, wird ein Name angefragt.
Projekt speichern unter	Projekt unter neuem Namen sichern. Dateien können im .nsb oder .txt Format gespeichert werden.
Druckeinrichtung...	Der Standard Windows Drucker Setup.
Drucken...	Projekt drucken.
Letzte Dateien	Eine Liste zum Öffnen der zuletzt bearbeiteten Projekte.
Ende	Aktuelles Projekt schließen und IDE verlassen.
Bearbeiten	
Rückgängig	Macht die letzte Aktion im Code-Fenster rückgängig. Mehrfaches Rückgängigmachen ist möglich.
Wiederholen	Wiederholt die letzte Aktion im Code-Fenster.
Ausschneiden	Schneidet den ausgewählten Code oder das Objekt aus. Objekte werden

	dabei als Texte ausgeschnitten (und kopiert) und können somit in anderen Anwendungen angesehen und verändert werden.
Kopieren	Kopiere den ausgewählten Code oder das Objekt.
Einfügen	Fügt den ausgewählten Code oder das Objekt ein.
Löschen	Löscht den ausgewählten Code oder das Objekt.
Suchen...	Suchen sucht im Code nach einem String. Es kann im gesamten Projekt oder im aktuellen Code gesucht werden. Optionen existieren für Ganzwort-Suche und Abhängigkeit von Groß/Kleinschreibung.
Ersetzen...	Ersetzen ähnelt Suchen, kann aber verwendet werden, um einzelne oder alle Übereinstimmungen zu ersetzen.
Gehe zu Zeile...	Positioniert den Cursor an der spezifizierten Zeile im obersten Code-Fenster.
Ansicht	
Projekt Explorer	Projekt Explorer Fenster anzeigen oder verbergen.
Eigenschaften Fenster	Eigenschaften Fenster anzeigen oder verbergen.
Werkzeug	Werkzeugkasten anzeigen oder verbergen. Der Werkzeugkasten zeigt eine Liste der verfügbaren Objekte.
Werkzeugleiste	Werkzeugliste anzeigen oder verbergen (nur unter den Menüs)
Statuszeile	Statuszeile anzeigen oder verbergen (am unteren Ende der IDE).
Aktualisieren	Alle Fenster neu zeichnen.
Projekt	
Formular einfügen	Dem Projekt ein neues Formular hinzufügen. Ein Formular ist eine Kollektion von Objekten, die vom Programm zur gleichen Zeit angezeigt wird. Ein Programm kann verschiedene Formulare haben.
Neues Modul	Dem Projekt ein neues Code Modul

hinzufügen	hinzufügen. Es wird ein neues Code-Fenster eingeblendet.
Vorhandenes Modul hinzufügen...	Dem Projekt ein vorhandenes Modul hinzufügen. Der Öffnen Dialog wird angezeigt. Mögliche Dateitypen sind .bas oder .cod.
Format	
Am Gitter ausrichten	Verändert die Größe eines Objektes um es an das Gitter anzupassen. Die Gittergröße kann unter Tools...Option...CE Screen eingestellt werden.
Mitte des Formulars	Positioniert das ausgewählte Objekt in der Mitte des Formulars (horizontal oder vertikal).
Ausführen	
Start	Startet das Programm. Das Programm wird auf das Windows CE Gerät kopiert. Abhängig von den Einstellungen in Tools...Options...Start, wird es automatisch gestartet.
Active Sync Programm zum Gerät	Aktuelles Programm mittels ActiveSync auf das Windows CE Gerät kopieren.
Installationsprogramme	Zeigt eine Liste aller Installer im Installer-Verzeichnis für dieses Windows CE Gerät an.
Start Virtual CE	Startet Virtual CE. Ihr Windows CE Gerät muss dazu per ActiveSync verbunden sein.
Extras	
Menüeditor	Startet den Menü-Editor.
ActiveX Control Manager	
EVB2NSB	
Optionen...	Startet das Optionen-Fenster.
Fenster	
Kaskadieren	Ordnet die offenen CE Bildschirme und Code-Fenster ordentlich kaskadiert auf dem Screen an.
Horizontal teilen	Ordnet die offenen CE Bildschirme

	und Code-Fenster ordentlich untereinander auf dem Screen an.
Vertikal teilen	Ordnet die offenen CE Bildschirme und Code-Fenster ordentlich nebeneinander auf dem Screen an.
Hilfe	
Lese Mich!	
Sprachdefinition	
Register...	Geben Sie Ihre Seriennummer zum Aktivieren Ihrer Kopie ein. Sie finden sie auf der Rückseite Ihres Handbuchs.
NS Basic Website	Gehe zur NS Basic Website.
Tech Notes	Zeige Tech Notes mit zusätzlichen Informationen, die nicht im Handbuch stehen, an.
Big Red Toolbox	Zeige Informationen über zusätzliche ActiveX Steuerelemente, die für NS Basic verfügbar sind, an.
Über NS Basic/CE	Der Standard NS Basic About Bildschirm. Hier können Sie Ihre aktuelle Version von NS Basic anzeigen lassen.

4.2 Programm-Code Fenster



Ihr Programmcode wird im Code-Fenster eingegeben. Der gesamte Code befindet sich in einem einzigen Code-Modul, aber sobald Sie ein Objekt im CE Screen anklicken, wird der Cursor automatisch an die entsprechende Position des Programm-Codes gebracht.

Der Text wird je nach Typ farbig dargestellt:

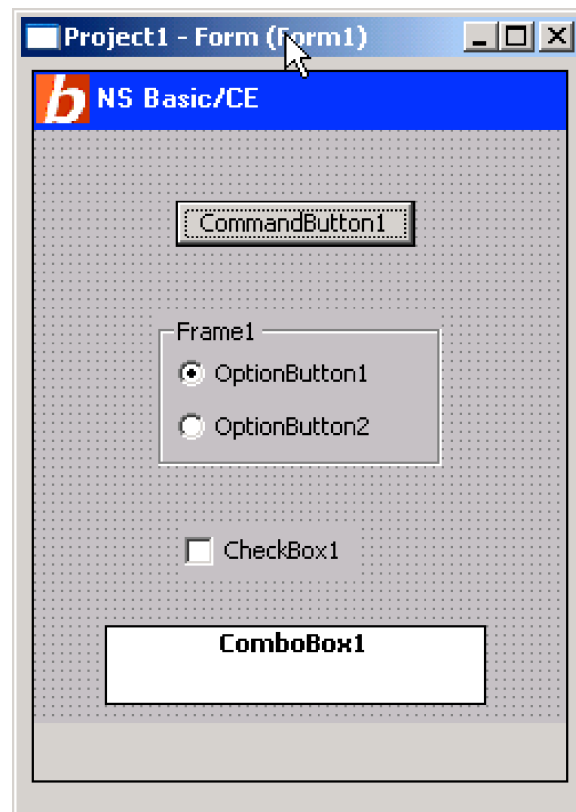
Black (Schwarz)	Programmtext
Blue (Blau)	NS Basic/CE Schlüsselwörter
Green (Grün)	Kommentare
Orange	Operatoren
Purple (Violett)	Strings
Yellow (Gelb)	Markierte Zeile

Über Extras -> Optionen -> Editoreinstellungen gelangen Sie zu ein paar hilfreichen Optionen. Die Auswahl an Eigenschaften erlaubt es Ihnen, zahlreiche Merkmale anzupassen, so die Textfarben, Tabulatoren, Zeichensätze, Tastaturkürzel und das Erscheinungsbild des Fensters.

Anweisungen können mittels "_" Zeichen am Ende einer Zeile auf mehrere Zeilen aufgeteilt werden.

Cut, Paste, Delete und andere ähnliche Funktionen können per Menü, per Werkzeugzeile oder durch Rechtsklick aufgerufen werden.

4.3 CE Bildschirm



Der CE Screen ist eine Nachahmung dessen, was Ihr Programm auf Ihrem Windows CE Gerät darstellen würde. Das Gitter im Hintergrund wird auf dem echten Gerät standardmäßig nicht angezeigt, es erleichtert Ihnen aber die Positionierungen auf Ihrem Formular. Sie können mit den Optionen die Gitterweite ändern oder es gänzlich ausschalten.

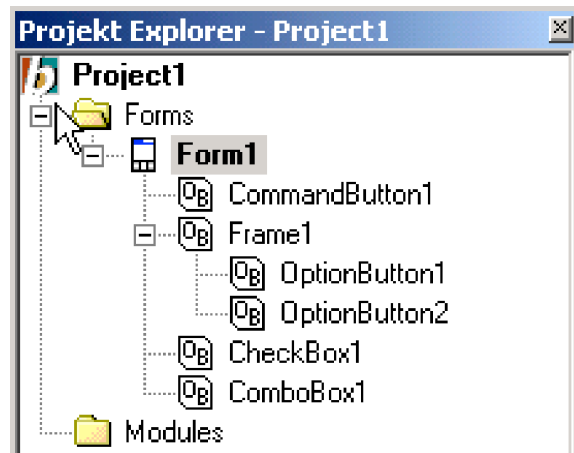
Um dem CE Screen ein neues Objekt hinzu zu fügen, wählen Sie es in der Werkzeugkiste an und klicken dann im CE Screen auf die Stelle, wo die linke obere Ecke erscheinen soll. Das neue Objekt wird dann angezeigt.

Ein Objekt wird per Klick ausgewählt. Mit den Funktionen des Format-Menüs können Objekte relative auf dem Bildschirm positioniert werden. Wenn ein Objekt selektiert ist, werden seine Eigenschaften im Eigenschaften-Fenster angezeigt. Wenn Sie dort Eigenschaften ändern, werden sie sofort auf dem Bildschirm wirksam. Um den Code eines Objektes zu ändern, klicken Sie es doppelt an und das entsprechende Code-Fenster wird sich öffnen.

Klicks außerhalb aller Objekte bringt das Eigenschaften-Fenster für das Formular auf den Bildschirm. Zum Umbenennen oder löschen eines Objektes klicken Sie es mit der rechten Maustaste im CE Screen an.

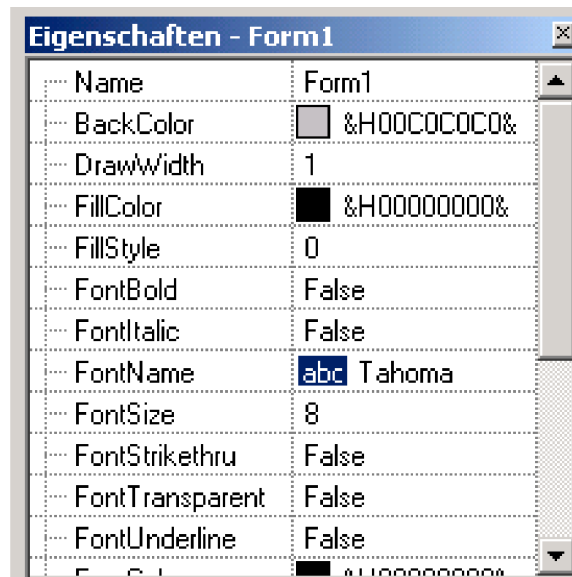
Doppeltes Klicken im Formular bringt die Form_Load Subroutine zum editieren auf den Screen. Die Form_Unload Subroutine müssen Sie selbst erstellen: Sie kann nicht automatisch angezeigt werden.

4.4 Projekt Explorer



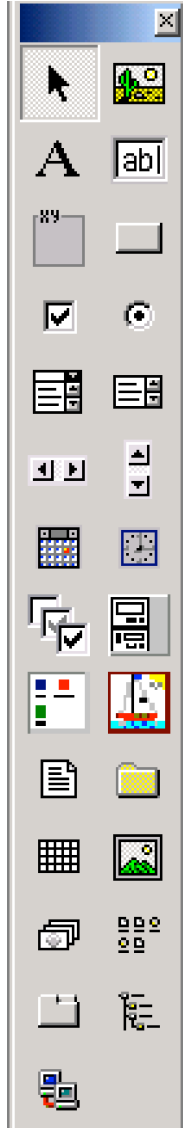
Das Project Explorer Fenster wird verwendet, um sich durch Ihr Projekt zu bewegen. Sie können es nutzen um festzustellen, welche Formulare Bestandteile Ihres Projektes sind. Wenn Sie eines auswählen, wird es am CE Screen angezeigt und im Eigenschaften-Fenster werden die Formular-Eigenschaften aufgelistet.

4.5 Eigenschaften-Fenster



Im Eigenschaften-Fenster werden die Werte für die Eigenschaften des Projektes, der Formulare und der Objekte auf den Formularen dargestellt. Die Werte können editiert werden und am CE Screen werden die Änderungen, sofern möglich, sofort angezeigt.

Zur Erklärung der verschiedenen Eigenschaften, lesen Sie bitte den Abschnitt "Eigenschaften" im Kapitel Referenz dieses Handbuchs.



4.6 Toolbox (Werkzeugkasten)

Die Toolbox beinhaltet Symbole für die Standard-Bedienelemente, die auf einem Formular (Form) platziert werden können. Wenn Sie den Cursor für einige Sekunden über dem Symbol verweilen lassen, wird eine Beschreibung des Bedienelements angezeigt.

Das ActiveX Symbol ist etwas besonderes. Es erlaubt es Ihnen, Steuerelemente zu benutzen, die nicht im Werkzeugkasten vorhanden sind. Klicken Sie das Symbol einfach an und legen Sie fest, welches Steuerelement Sie verwenden möchten. Es wird dann normalerweise dem Projekt hinzugefügt.

Mehr Informationen zu den Steuerelementen finden Sie in diesem Handbuch unter den Ausführungen zum AddObject Befehl im Kapitel Referenz. Immanente Objekte werden in der Sektion Referenz beschrieben. Die erweiterten Objekte sind in den Tech Notes dokumentiert.

Nicht alle ActiveX Steuerelemente können dem Werkzeugkasten hinzugefügt werden. Es muss eine Desktop-Version dazu existieren in der alle Eigenschaften durch den Autor korrekt definiert wurden. Sie können jedoch die Werkzeugkiste umgehen und Ihrem Projekt ActiveX Elemente über die AddObject Anweisung hinzufügen. In diesem Fall werden die Eigenschaften nicht im Eigenschaften-Fenster geändert, sondern in Ihrem Programmcode.

Um den Namen eines Objektes zu ändern nachdem es erzeugt wurde,

klicken Sie es mit der rechten Maustaste an und wählen aus dem aufgehenden Menü Rename aus.

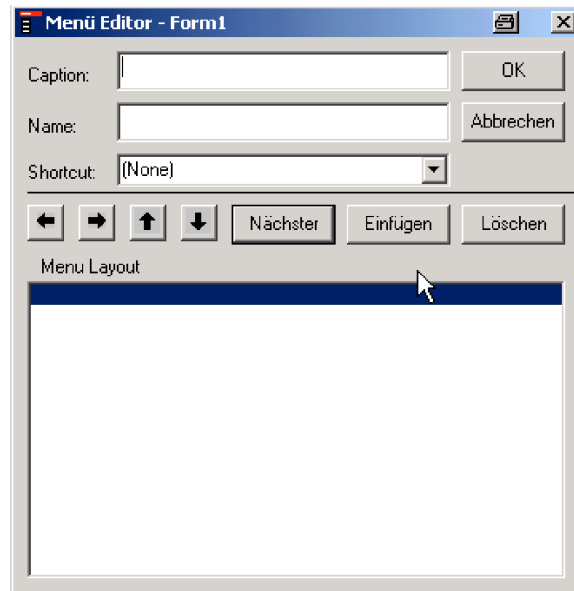
4.7 Toolbar (Werkzeugleiste)



Die Werkzeugleiste enthält Symbole für oft verwendete Operationen. Sie finden sie alle auch im Menü wieder.

Neu
Öffnen
Speichern
Drucken
Rückgängig
Wiederholen
Ausschneiden
Kopieren
Einfügen
Löschen
Suchen
Form einfügen
Start
Eigenschaften
Menüeditor

4.8 Menu Editor (Menü-Editor)



Mit dem Menü-Editor können Sie Ihre Menüs ändern. Jedes Formular kann ein Menü haben. Menüs können sich über mehrere Stufen erstrecken.

Caption	Der Titel des Menüs
Name	Der Name der Funktion, die bei Auswahl des Menüs aufgerufen wird. Leerzeichen und Sonderzeichen sind nicht erlaubt.
Shortcut	Kürzel zum Auswählen von Elementen.
OK	Änderungen sichern und Menü schließen.
Cancel	Änderungen verwerfen und Menü schließen.
Links	Verschiebe markiertes Element eine Stufe nach oben.
Rechts	Verschiebe markiertes Element eine Stufe nach unten.
Auf	Über vorherigem Element positionieren.
Ab	Unter nächstem Element positionieren.

Nächster	Zum nächsten Element verschieben. Falls bereits am Ende, neues Element einfügen.
Einfügen	Neues Element vor Markierung einfügen.
Löschen	Markierung löschen.

Um den Programmcode, der ausgeführt werden soll, wenn das Menü aufgerufen wird, einzugeben, schließen Sie zunächst den Menü-Editor und selektieren Sie das gewünschte Menü im CE Screen. Es wird sich ein Code-Fenster mit dem Beginn der Subroutine, die bei Auswahl dieses Menüs ausgeführt wird, öffnen.

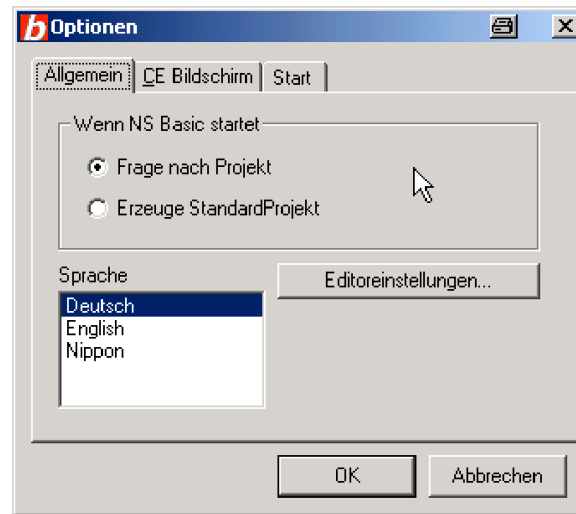
Die Unteroutine wird `menultem_click` benannt sein, wobei `menultem` der Name aus dem Menü-Editor ist. Beachten Sie, dass dieser Name sich nicht mit anderen Namen von Tasten, anderen Menüs oder sonstigen Objekten, auch nicht in anderen Formularen, überschneiden darf.

Zum Erzeugen von Untermenüs benutzen Sie den Pfeil nach rechts, um das Element einzurücken. Mit dem Pfeil nach links heben Sie diese Einrückung wieder auf.

Mit den Auf- bzw. Ab- Pfeilen können Sie die Reihenfolge der Menü-Elemente ändern. Markieren Sie das gewünschte Element und verschieben Sie es mit der entsprechenden Taste.

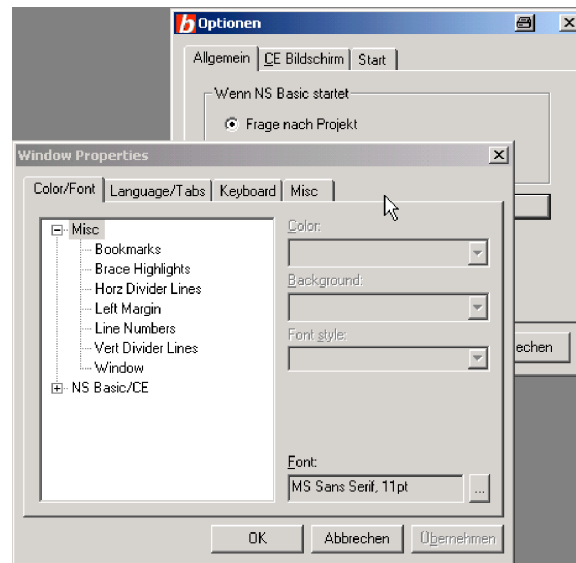
4.9 Optionen

4.9.1 Optionen – General



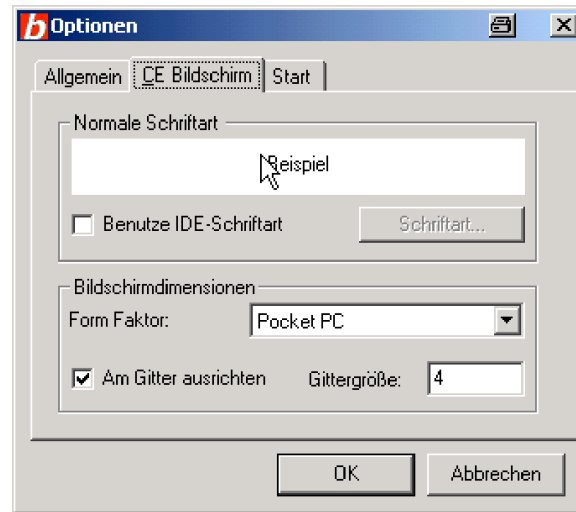
Frage nach Projekt	Falls ausgewählt, wird NS Basic/CE beim Start nachfragen, ob ein existierendes oder ein neues Projekt geöffnet werden soll.
Erzeuge StandardProject	NS Basic/CE wird beim Starten ein neues Projekt öffnen, falls dieser Punkt selektiert wurde.
Sprache	Legt fest, in welcher Sprache Meldungen angezeigt werden sollen. Angezeigt wird eine Liste der möglichen Sprachen aus dem Lang (Language) Verzeichnis von NS Basic/CE.
Editoreinstellungen	Eigenschaften des Code-Fensters festlegen, siehe 4.9.2.

4.9.2 Optionen – Editor



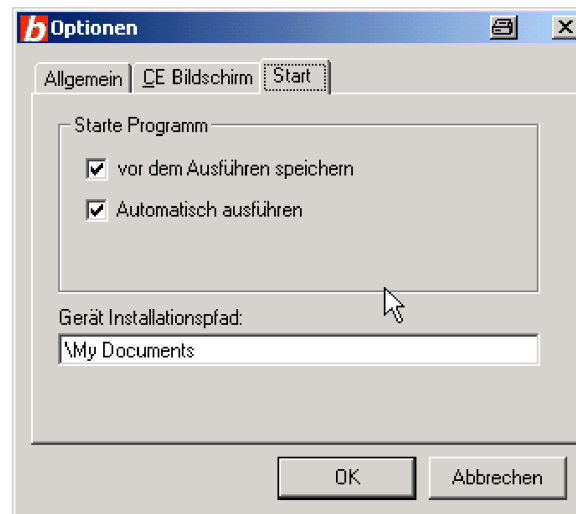
In den Editor-Optionen lassen sich die Eigenschaften des Code-Fensters in weiten Bereichen ändern, dazu gehören die Textfarben, Tabulatoren, Schriftarten, Tastaturkürzel und Erscheinungsbild des Fensters.

4.9.3 Optionen – CE Screen



Am Gitter ausrichten	Sollen Objekte im Formular automatisch am Raster ausgerichtet werden? Dadurch lässt sich einfacher ein schönes Formular-Layout erreichen.
Gittergröße	Wenn Snap to Grid gewählt ist, wird hier der Gitterabstand festgelegt.

4.9.4 Optionen – Start



In diesem Fenster werden Optionen festgelegt, die ausgeführt werden, wenn Start aus dem Run Menü aufgerufen wird.

Vor dem Ausführen speichern	Datei vor jedem Run...Start sichern?
Automatisch ausführen	Nachdem ActiveSync das Programm kopiert hat, wird es automatisch gestartet.
Gerät Installationspfad	Der Ort auf dem Windows CE Gerät, an dem die Kopie des Programms gespeichert werden soll. Dieser Ordner muss existieren. Der korrekte Pfad unterscheidet sich von Gerät zu Gerät. Für Pocket PC Geräte sollte er "\\My Documents" sein. Für andere Geräte einfach "\\".

5. NS Basic/CE Referenz

Das Kapitel Referenz enthält in alphabetischer Anordnung je einen Eintrag für jedes Kommando, jede Anweisung und jede Funktion, die in NS Basic/CE verwendet werden. Die Einträge werden im Index unter den Gruppen Kommandos, Anweisungen oder Funktionen aufgelistet.

Jeder Eintrag dieser Referenz besteht aus folgenden Informationen:

Name	Kategorie
------	-----------

BEGRIFF Parameter

Beschreibung

Hier werden BEGRIFF und die erforderlichen Parameter beschrieben. Es werden Details zur Verwendung von BEGRIFF gegeben, sowie etwaige Nutzungseinschränkungen.

Beispiel

Ein kleines Programm, das BEGRIFF verwendet wird hier aufgelistet.

Ausgabe

In diesem Bereich wird das Ergebnis des Beispielprogramms angezeigt.

Verwandte Begriffe

Hier finden Sie, falls es sie gibt, NS Basic/CE Kommandos, Anweisungen, Funktionen oder Objekte, die mit BEGRIFF verwandt sind. Oftmals bekommt man ein besseres Verständnis von BEGRIFF, wenn man sich die verwandten Begriffe ansieht.

ABS(number)

Beschreibung

ABS gibt den Absolutwert oder vorzeichenlosen Betrag eines numerischen Ausdrucks wieder. Der erforderliche Parameter, *number*, kann jeder gültige numerische Ausdruck sein. Ist *number* NULL, gibt ABS NULL zurück.

Beispiel

```
REM ABS Beispiel
'ABS gibt den Absolutwert der Zahl zurueck.
PRINT "ABS (-2) = " & ABS (-2)
PRINT "ABS (2) = " & ABS (2)
```

Ausgabe

```
ABS (-2) = 2
ABS (2) = 2
```

Verwandte Begriffe

SGN

ADDOBJECT *objectId:license[, objectname[, xpos, ypos, width, height, parent]]*

Beschreibung

ADDOBJECT fügt dem Programm ein Objekt hinzu. Sobald das geschehen ist, können die Objekteigenschaften abgefragt und die Objektmethoden innerhalb des Programms ausgeführt werden. Der erforderliche Parameter, *objectId*, ist eine Zeichenfolge, die den vollständigen Referenznamen des Objektes aus der Registrierung oder ein Kürzel für das Objekt enthält. In Tabelle 4 weiter unten werden gängige Objekte aufgelistet. Wenn das Steuerelement eine Lizenznummer erfordert (*license*), geben Sie diese getrennt durch einen Doppelpunkt, nach der *objectId* ein. Der optionale Parameter, *objectname*, ist eine Zeichenfolge, die den Standard-Namenskonventionen für Variablen entsprechen muss. Über die Variable *objectname* kann innerhalb des Programms auf das Objekt zugegriffen werden. Wenn *objectname* nicht angegeben wurde, wird standardmäßig *objectId* genutzt. Die optionalen Parameter *xpos*, *ypos*, *width*, und *height*, sind numerische Ausdrücke, die bei sichtbaren Objekten die Position und Größe in Pixeln angeben. Der *parent* Parameter macht das Objekt zum Kind des angegebenen Objekts: das neue Steuerelement wird also auf *parent* positioniert.

Die Beschreibung eines Objekts und seine Werte sind in seinen Eigenschaften definiert. (Siehe "Eigenschaften")

Objekte können Funktionen enthalten, die Sie in Ihrem Programm aufrufen können. (Siehe "Methoden")

Objekte die Bestandteil von NS Basic/CE selbst sind, werden in diesem Handbuch beschrieben. NS Basic/CE unterstützt auch externe Objekte, so genannte ActiveX Controls (Steuerelemente). Einige davon werden mit NS Basic/CE ausgeliefert; andere werden von Drittfirmen angeboten. Sie werden in den Tech Notes aus dem Hilfe-Menü beschrieben.

Objekte können über SUB Prozeduren Ereignismeldungen an Ihr Programm schicken. Der Name dieser Prozedur ist dann eine Kombination aus dem Objektnamen und dem Ereignis. Bei manchen Ereignissen kann optional eine durch Kommata getrennte Liste von Argumenten übergeben werden. Wichtig: Wenn ein Objekt eine Ereignismeldung sendet, müssen Sie immer alle 6 Argumente bereitstellen. (Siehe "Ereignisse in diesem Kapitel")

```
SUB objectname_event(arglist)]
```

```

'Ausfuehren, wenn objectname eine
'Ereignismeldung erhaelt
END SUB

```

Tabelle 4: Gängige Objekte

Kürzel	Ist dokumentiert in:
ADO	Data Base - Tech Note 10
CheckBox	In diesem Dokument
ComboBox	In diesem Dokument
Comm	Serial Comms - Tech Note 04
CommandButton	In diesem Dokument
Date	In diesem Dokument
Dialog	Dialog boxes - Tech Note 08
File	Simple File I/O - Tech Note 03
FileSystem	Files and Dirs - Tech Note 02
Frame	In diesem Dokument
Finance	Money calculations - Tech Note 11
Grid	Data Grid - Tech Note 06
HScrollbar	In diesem Dokument
Image	Image holder - Tech Note 20
ImageList	List of Images - Tech Note 21
Label	In diesem Dokument
ListBox	In diesem Dokument
ListView	Display Items - Tech Note 22
OptionButton	In diesem Dokument
Output	In diesem Dokument
PictureBox	Manipulate Pictures - Tech Note 07
TabStrip	Tab object - Tech Note 09
TextBox	In diesem Dokument
Time	In diesem Dokument
TreeView	Tech Note 23
TriButton	In diesem Dokument
VScrollBar	In diesem Dokument
WinSock	Internet stuff - Tech Note 05

Beispiel

```
REM ADDOBJECT fuegt dem Programm ein Objekt
REM zu
ADDOBJECT "ThirdParty.Control:123-ghs","TPC"
ADDOBJECT "Finance", "Finance"
ADDOBJECT "PictureBox", "PBox", 65, 10, 55,
20
PBox.BorderStyle = 1
PBox.BackColor = vbWhite
PBox.DrawText " Button"
SUB PBox_Click()
    PRINT FORMATCURRENCY(Finance.Pmt(.0058, _
        180, 130000))
END SUB
```

Ausgabe

Verwandte Begriffe

Events, Methods, Properties, SET

AND

Operator

result = x AND y

Beschreibung

AND gibt die logische Konjunktion zwei Ausdrücke. *result* ist dann und nur dann TRUE, wenn beide Ausdrücke TRUE ergeben. Ansonsten ist *result* FALSE.

Mit AND kann auch ein bit-weiser Vergleich zweier numerischer Ausdrücke durchgeführt werden. Jedes Bit in *result* wird auf 1 gesetzt, wenn beide korrespondierenden Bits in x und y auf 1 stehen, ansonsten auf 0.

Beispiel

```
REM AND Beispiel
'AND fuehrt logische und bit-weise
'Konjunktion durch
DIM Test1, Test2, x, y
x = 2
y = 7
Test1 = x > 0 AND y < 10
Test2 = x > 0 AND y > 10
PRINT "logisch:"
PRINT "  x > 0 AND y < 10 = " & Test1
PRINT "  x > 0 AND y > 10 = " & Test2
PRINT "Bitweise:"
PRINT "  x AND y = " & x AND y
```

Ausgabe

```
Logisch:
  x > 0 AND y < 10 = True
  x > 0 AND y > 10 = False
Bitweise:
  x AND y = 2
```

Verwandte Begriffe

EQV, IMP, NOT, OR, XOR

ARRAY

Funktion

ARRAY(*expressionlist*)

Beschreibung

ARRAY erzeugt ein dynamisches Array. Der erforderliche Parameter, *expressionlist*, ist eine Liste von Ausdrücken, die durch Kommata getrennt werden. Die Länge des entstehenden Array's entspricht der Anzahl der Elemente in *expressionlist*. Array's die mit ARRAY erzeugt werden haben eine Untergrenze von 0.

Beispiel

```
REM ARRAY Beispiel
'ARRAY erzeugt ein dynamisches Array
DIM MyArray, Message
'Ein Array mit 2 Strings erzeugen
MyArray = Array("Hallo", "Welt!")
'Auf das Array zugreifen und die Elemente
'verbinden
PRINT MyArray(0) & " " & MyArray(1)
```

Ausgabe

Hallo Welt!

Verwandte Begriffe

DIM

ASC

Funktion

ASC(*string*)

ASCB(*string*)

ASCW(*string*)

Beschreibung

ASC gibt den ANSI-Charakter Code eines Zeichens zurück. Der erforderliche Parameter, *string*, kann jeder gültige String sein. Wenn *string* länger als ein Zeichen ist, wird nur das erste Zeichen verwendet.

Die ASCB Funktion wird für Byte-Daten eines Strings verwendet. Anstelle eines Charakter-Codes gibt sie das erste Byte zurück. ASCW wird für 32-Bit Plattformen, die Unicode-Charakter einsetzen, verwendet. Sie gibt den Unicode-Charakter-Code (wide) zurück und man vermeidet dadurch die Umwandlung von Unicode zu ANSI.

Beispiel

```
REM ASC Beispiel
'ASC gibt einen ANSI-Charakter-Code zurueck
DIM CapitalA, LowerB
CapitalA = ASC("A steht fuer Apple")
PRINT "Charakter-Code fuer A = " & CapitalA
LowerB = ASC("b")
PRINT "Charakter-Code fuer b = " & LowerB
```

Ausgabe

```
Charakter-Code fuer A = 65
Charakter-Code fuer b = 98
```

Verwandte Begriffe

CHR

ATN(*number*)

Beschreibung

ATN gibt den Arkustangens einer Zahl im Bogenmass zurück. Der erforderliche Parameter, *number*, kann jeder numerische Ausdruck sein.

Um Grad in Radiant umzuwandeln, nehmen Sie die Gradzahl mit $\pi/180$ mal. Um Radiant in Grad zu konvertieren, multiplizieren Sie den Radiantwert mit $180/\pi$.

Beispiel

```
REM ATN Beispiel
'ATN kalkuliert den Arkustangens einer
'Zahl
DIM Pi
Pi = ATN(1) * 4
PRINT "Der Wert von Pi ist " & Pi
```

Ausgabe

Der Wert von Pi ist 3.141593

Verwandte Begriffe

COS, SIN, TAN

BREAK**Befehl**

BREAK [*prompt*[, *statements*]]

Beschreibung

BREAK unterbricht temporär die Programmausführung und zeigt eine Dialogbox an, die es dem Programmierer ermöglicht, Befehle auszuführen oder die Verarbeitung fortzusetzen. Der optionale Parameter, *prompt*, ist eine Zeichenfolge, die im Körper der Dialogbox angezeigt wird. Der ebenfalls optionale Parameter, *statements*, ist ein String der im Ausführungsbereich der Dialogbox dargestellt wird.

Beispiel

```
REM BREAK Beispiel
'BREAK unterbricht die Ausfuehrung temporaer
DIM i
FOR i = 0 TO 4
    PRINT (i + 1) * 10
    IF i = 2 THEN
        BREAK "Durchlauf = " & i, "PRINT i"
    END IF
NEXT
```

Ausgabe

Verwandte Begriffe

MSGBOX

BYE

Beschreibung

BYE beendet das laufende Programm und schließt das Ausgabefenster. BYE kann innerhalb von FUNCTION Prozeduren oder SUB Prozeduren aufgerufen werden, aber das hat keinen Effekt, falls die Prozedur nicht innerhalb einer Kette von Prozeduren aufgerufen wurde, die durch eine Ereignismeldung des Betriebssystems ausgelöst wurde.

Beispiel

```
REM BYE Befehl
'BYE schliesst das Ausgabefenster und
'beendet das Programm
DIM When, In10
In10 = MSGBOX("In 10 Sek. schliessen?",
vbYESNO)
IF In10 = vbYES THEN
    PRINT "Ende in 10 Sekunden..."
    When = DATEADD("s", 10, NOW)
    WHILE NOW < When
        '10 Sekunden nichts tun
    WEND
    BYE
END IF
SUB Output_Click()
    Quit
END SUB
SUB Quit
    BYE
END SUB
```

Ausgabe

Ende in 10 Sekunden...

(Ausgabefenster wird geschlossen)

Verwandte Begriffe

CALL *procedurename*[(*argList*)]

Beschreibung

CALL ist eine eindeutige Methode zum Aufruf einer FUNCTION oder SUB Prozedur. Der erforderliche Parameter, *procedurename*, kann jeder Prozedurname sein. Die optionale Komponente, *argList*, ist eine durch Kommata getrennte Liste von Variablen, die der aufzurufenden Prozedur übergeben wird. Das Schlüsselwort CALL ist optional, Prozeduren können ohne dieses Keyword aufgerufen werden, mit

name[(*argList*)]

Beispiel

```
REM CALL Beispiel
'CALL ruft explizit eine Prozedur auf
CALL Welcome
CALL Message("NS Basic/CE ist exzellent.")
Wave
FUNCTION Welcome
    PRINT "Hallo Welt!"
END FUNCTION
FUNCTION Message(Text)
    PRINT "Meldung: " & Text
END FUNCTION
SUB Wave
    PRINT "Tschüss!"
END SUB
```

Ausgabe

```
Hallo Welt!
Meldung: NS Basic/CE ist exzellent.
Tschüss!
```

Verwandte Begriffe

SUB, FUNCTION

CHAIN *pathname*, *reset*

Beschreibung

CHAIN lädt ein Programm in die Betriebssystem-Umgebung und startet es. Die erforderliche Komponente, *pathname*, ist eine Zeichenfolge, die den absoluten Pfadnamen des zu ladenden und auszuführenden Programms spezifiziert. Die ebenfalls erforderliche Komponente, *reset*, ist ein boolescher Ausdruck, der angibt, ob die Umgebung vor dem Laden und Ausführen des neuen Programms komplett zurückgesetzt werden soll (alle Konstanten, Variablen und Prozeduren aus dem Speicher löschen). Verwenden Sie den BYE Befehl nach CHAIN, um das aufrufende Programm zu beenden.

Wenn *reset* auf TRUE steht, verbleibt die Ausführung beim Original-Programm bis es beendet wird. Es werden alle Variablen und Subroutinen zurückgesetzt und erst dann wird das neue Programm geladen und ausgeführt.

Wenn *reset* auf FALSE steht, wird das neue Programm sofort gestartet, alle Variablen und Subroutinen bleiben bestehen. Sobald das neue Programm beendet wird, fährt das Original-Programm mit dem NEXT Befehl fort. Variablen und Subroutinen, die durch das per CHAIN aufgerufene Programm verändert oder hinzugefügt wurden, bleiben bestehen.

Tipp: Verwenden Sie CHAIN mit *reset* auf FALSE um gebräuchliche Programmbefehle in mehrere Programme zu "integrieren".

Beispiel

```
'Programm A
PRINT "Das ist Programm A"
A=100
CHAIN "B.nsb",TRUE 'Umgebung loeschen
PRINT "Das ist Programm A nach CHAIN"
PRINT A
'Programm B
PRINT "Das ist Programm B"
PRINT A
A=200
```

Ausgabe

```
Das ist Programm A
Das ist Programm A nach CHAIN
100
```

<Bildschirm wird gelöscht>

```
Das ist Programm B
```

<es wird kein Wert für A angezeigt, da die Variable in Programm B nicht definiert ist>

Beispiel 2

```
'Programm A
PRINT "Das ist Programm A"
A=100
CHAIN "B.nsb",FALSE 'Umgebung nicht loeschen
PRINT "Das ist Programm A nach CHAIN"
PRINT A
'Programm B
PRINT "Das ist Programm B"
PRINT A
A=200
```

Ausgabe

```
Das ist Programm A
Das ist Programm B
100
Das ist Programm A nach CHAIN
200
```

Verwandte Begriffe

EXECUTE

CheckBox

Objekt

ADDOBJECT "CheckBox", *name*, *xpos*, *ypos*, *width*, *height*

ADDOBJECT " TriButton ", *name*, *xpos*, *ypos*, *width*, *height*

Beschreibung

CheckBox und TriButton werden verwendet, um ein frei positionierbares Kontrollkästchen auf dem Bildschirm auszugeben. TriButton ist fast identisch mit CheckBox, abgesehen von einem dritten, ausgegrauten Modus. Die erforderliche Komponente, *name*, ist der Name der Variablen, die dem Programm als Referenz auf das Objekt dient. Sie muss den Standard-Namenskonventionen entsprechen. Die ebenfalls erforderlichen Komponenten, *xpos*, *ypos*, *width*, und *height* sind numerische Ausdrücke, die zusammen die Größe und Position (in Pixeln) des Objekts bestimmen, gemessen von der linken oberen Ecke. Die *Value* Eigenschaft ist 0, wenn das Häkchen nicht gesetzt ist; sie ist 1, bei gesetztem Häkchen.

Unterstützte Eigenschaften (siehe "Eigenschaften")

Alignment, BackColor, Caption, Enabled, FontBold, FontItalic, FontName, FontSize, FontStrikethru, FontUnderline, ForeColor, Height, Hwnd, Left, Name, ParentHwnd, TabStop, Tag, Text, Timer, Top, Value*, Visible, Width, WindowLong

*Die Value Eigenschaft ist ein Integerwert (0 oder 1), der auch durch TRUE oder FALSE gesetzt werden kann. Der TriButton hat einen zusätzlichen Wert, 2, für unbestimmten Modus.

Unterstützte Methoden (siehe "Methoden")

Hide, Move, SetFocus, Show

Unterstützte Ereignisse (Siehe "Ereignisse")

Change, Click, GotFocus, KeyDown, KeyPress, KeyUp, LostFocus, Timer

Beispiel

```
REM CheckBox Beispiel
'CheckBox ist ein Kontrollkaestchen
ADDOBJECT "CheckBox", "Check",10,10,200,20
Check.BackColor = Ausgabe.BackColor
Check.Text = "CheckBox Objekt"
Check.Value = TRUE
SUB Check_Click
    PRINT "CheckBox Value: " & Check.Value
END SUB
```

Ausgabe

☒ CheckBox Object

Beispiel

```
REM TriButton Beispiel
ADDOBJECT "TriButton", "Button", 120, 120,
100, 20
Button.Value = 2 'auf unbestimmt gesetzt
```

Ausgabe

☒ Tri Button

Verwandte Begriffe

ADDOBJECT, Events, Methods, Properties

CHR**Funktion**

CHR(*number*)

CHRB(*number*)

CHRW(*number*)

Beschreibung

CHR gibt das zugehörige ANSI-Zeichen eines Zeichencode-Wertes zurück. Der erforderliche Ausdruck, *number*, kann jeder numerische Ausdruck sein.

Die CHRB Funktion wird mit Byte-Daten in einem String verwendet. Anstelle eines Charakter-Codes, der ein oder zwei Byte haben könnte, gibt sie ein einzelnes Byte zurück. CHRW wird für 32-Bit Plattformen, die Unicode-Charakter einsetzen, verwendet. Ihr Argument ist ein Unicode-Charakter-Code (wide) und man vermeidet dadurch die Umwandlung von ANSI zu Unicode.

Beispiel

```
REM CHR Beispiel
'CHR gibt ein Zeichen zurueck
DIM Lowercase, Uppercase
Lowercase = CHR(97)
Uppercase = CHR(97 - 32)
PRINT "Kleinbuchstabe = " & Lowercase
PRINT "Grossbuchstabe = " & Uppercase
```

Ausgabe

```
Kleinbuchstabe = a
Grossbuchstabe = A
```

Verwandte Begriffe

ASC

CLASS**Befehl**

CLASS *name*

[*statements*]

END CLASS

Beschreibung

Deklariert eine Klasse, sowie deren Variablen, Eigenschaften und Methoden. Variablen werden als PUBLIC oder PRIVATE definiert. Eigenschaften werden mittels PROPERTY SET, PROPERTY LET und PROPERTY GET Anweisungen gesetzt. Methoden werden als SUB oder FUNKTION Blöcke definiert. Jede Klasse hat optional eine Class_Initialize und eine Class_Terminate Subroutine, die beim Erstellen oder Entfernen einer Klassen-Instanz aufgerufen wird. Gilt nur für Windows CE 4.0 oder später.

Beispiel

```
Class cGreeter
    Public Sub SayHello(who)
        Print "Hallo, " & who & ". Willkommen _
            bei " & Store & "."
    End Sub
    Public Store
    Public Property Get StoreNumber
        Select Case Store
            Case "Hauptsitz": StoreNumber = 1
            Case "Filiale:": StoreNumber = 2
        End Select
    End Property
    Sub Class_Initialize
        Store="Hauptsitz"
    End Sub
    Sub Class_Terminate
        Print "Begrueßung beendet"
    End Sub
End Class

Dim p
Set p = new cGreeter
p.sayHello("Meister")
Print p.StoreNumber
Set p=Nothing
```

Ausgabe

```
Hallo Meister. Willkommen bei Hauptsitz.
1
Begrueßung beendet
```

Verwandte Begriffe

DIM, FUNCTION, PROPERTY, SET, SUB, WITH

ADDOBJECT "ComboBox", *name*, *xpos*, *ypos*, *width*, *height*

Beschreibung

Die ComboBox wird verwendet, um frei positionierbare DropDownListen mit Eingabezeile im Ausgabefenster darzustellen. Der erforderliche Parameter, *name*, ist der Name der Variablen, die dem Programm als Referenz auf dieses Objekt hinzugefügt wurde. Sie muss den Standard-Nameskonventionen für Variablen entsprechen. Die erforderlichen Komponenten, *xpos*, *ypos*, *width*, und *height* sind numerische Ausdrücke, die zusammen die Größe und Position (in Pixeln) des Objekts bestimmen, gemessen von der linken oberen Ecke. Die Eigenschaften Text und Caption beinhalten die Werte für die Eingabezeile des Objekts. Die ListIndex Eigenschaft entspricht dem Index der Eingabe bezogen auf die Liste, angefangen mit der 0 für das erste Element. Stimmt die Eingabe mit keinem Listenelement überein, wird der ListIndex auf -1 gesetzt. Stellen Sie sicher, dass die Höhe des geöffneten Objekts angegeben ist: zur Laufzeit wird nur eine einzelne Zeile angezeigt, solange die ComboBox nicht aufgeklappt ist.

Unterstützte Eigenschaften (siehe "Eigenschaften")

BackColor, Caption, Enabled, ExpandedHeight, FontBold, FontItalic, FontName, FontSize, FontStrikethru, FontUnderline, Forecolor, Height, Hwnd, IntegralHeight, Left, ListCount (read-only), ListIndex, LowercaseOnly, Name, NewIndex, ParentHwnd, Redraw, Sorted, Style, TabStop, Tag, Text, Timer, Top, TopIndex, UppercaseOnly, Visible, Width, WindowLong

Unterstützte Methoden (siehe "Methoden")

AddItem, Clear, Hide, Move, RemoveItem, SetFocus, Show

Unterstützte Ereignisse (siehe "Ereignisse")

Change, Click, DropDown, GotFocus, KeyDown, KeyPress, KeyUp, LostFocus, Timer

Beispiel

```
REM ComboBox Beispiel
'ComboBox ist eine DropDownListe
DIM i
ADDOBJECT "ComboBox", "Combo", 5, 5, 100, 200
Combo.Style = 2
FOR i = vbSUNDAY TO vbSATURDAY
    Combo.AddItem WEEKDAYNAME(i)
NEXT
Combo.ListIndex = WEEKDAY(NOW-1)
```

Ausgabe

Verwandte Begriffe

ADDOBJECT, Events, ListBox, Methods, Properties

CommandButton

Objekt

ADDOBJECT "CommandButton", *name*, *xpos*, *ypos*,
width, *height*

Beschreibung

CommandButton wird verwendet um eine Standard-Taste im Ausgabe-Fenster anzuzeigen. Die erforderliche Komponente, *name*, ist der Name der Variablen, die dem Programm als Referenz auf dieses Objekt hinzugefügt wurde. Sie muss den Standard-Nameskonventionen für Variablen entsprechen. Die erforderlichen Komponenten, *xpos*, *ypos*, *width*, und *height* sind numerische Ausdrücke, die zusammen die Größe und Position (in Pixeln) des Objekts bestimmen, gemessen von der linken oberen Ecke.

Durch Setzen der Eigenschaft Value auf 1 wird die Taste betätigt, ein Click-Ereignis ausgelöst und Value wieder zurück auf 0 gesetzt.

Unterstützte Eigenschaften (siehe "Eigenschaften")

BackColor, Caption, Default, Enabled, FontBold, FontItalic, FontName, FontSize, FontStrikethru, FontUnderline, ForeColor, Height, Hwnd, Left, Name, ParentHwnd, TabStop, Tag, Text, Timer, Top, Visible, Width, WindowLong, Value

Unterstützte Methoden (siehe "Methoden")

Hide, Move, SetFocus, Show

Unterstützte Ereignisse (siehe "Ereignisse")

Click, GotFocus, KeyDown, KeyPress, KeyUp, LostFocus, Timer

Beispiel

```
REM CommandButton Beispiel
'CommandButton ist eine Standard-Taste
ADDOBJECT "CommandButton", "Taste", _
    100, 10, 80, 20
Taste.BackColor = Ausgabe.BackColor
Taste.Text = "Druecke Mich!"
```

Ausgabe.BackColor = vbWHITE

```
SUB Button_Click
    PRINT "Taste betaetigt"
    KillFocus
```

END SUB

Ausgabe

Verwandte Begriffe

ADDOBJECT, Events, Methods, Properties

CONST

Befehl

[PUBLIC | PRIVATE] CONST *name* = *expression*

Beschreibung

CONST deklariert Konstanten, die in Ausdrücken anstelle von literalen Werten verwendet werden können. Die erforderliche Komponente, *name*, muss den Standard-Namenskonventionen entsprechen. Die ebenfalls erforderliche Komponente, *expression*, kann jedes Literal, jede Konstante, oder die Kombination beider, einschließlich arithmetischer und logischer Operationen (Ausnahme: IS) sein.

Die optionalen Schlüsselwörter PUBLIC und PRIVATE werden auf Skript-Ebene eingesetzt, um Konstanten für FUNCTION und SUB Prozeduren als sichtbar oder unsichtbar zu kennzeichnen. Als Vorgabe sind alle Konstanten PUBLIC und über das Skript in allen Prozeduren verfügbar. Alle Konstanten, die innerhalb einer Prozedur deklariert werden, sind nur innerhalb dieser Prozedur verfügbar.

Mehrere Konstanten können innerhalb einer Zeile deklariert werden, wenn sie durch Kommata getrennt werden. Wird in einer solchen Zeile das Schlüsselwort PUBLIC oder PRIVATE verwendet, gilt es für alle Konstanten, die in dieser Zeile deklariert werden.

Beispiel

```
REM CONST Beispiel
'CONST definiert Konstanten
CONST SHAPE = "Rechteck"
PRIVATE CONST AREA = 51
PUBLIC CONST LENGTH = 7, WIDTH = 11
PrintArea LENGTH, WIDTH
SUB PrintArea(l, w)
    DIM Area
    Area = l * w
    PRINT SHAPE & " Flaeche: " & l & " * " & w
    & _
    " = " & Area
END SUB
```

Ausgabe

Rechteck Flaeche: 7 * 11 = 77

Verwandte Begriffe

FUNCTION, PUBLIC, PRIVATE, SUB

Conversions (Konvertierungen) Funktion

CBOOL(*expression*)

CBYTE(*expression*)

CCUR(*expression*)

CDATE(*expression*)

CDBL(*expression*)

CINT(*expression*)

CLNG(*expression*)

CSNG(*expression*)

CSTR(*expression*)

Beschreibung

Die Konvertierungsfunktionen geben *expression* als passenden Datentyp zurück. Der erforderliche Parameter, *expression*, kann jeder gültige Ausdruck sein.

Wenn der Rückgabewert außerhalb des Wertebereichs des Rückgabe-Datentyps liegt, erfolgt eine Fehlermeldung.

Tabelle 5: Konvertierungsfunktionen

Funktion	Rückgabetyt	Kommentar
CBOOL	Boolean	False, wenn der Ausdruck Null ergibt, sonst True
CBYTE	Byte	Eine ganze Zahl zwischen 0 und 255
CCUR	Currency	Ein Währungswert
CDATE	Date	Datum zwischen 1. Januar 100 und 31. Dezember 9999, gültige Ausdrücke sind Datumsangaben oder Datum/Zeit- Literale
CDBL	Double	Zahl zwischen -1.79769313486232E308 und -4.94065645841247E-324 für negative Werte und zwischen 4.94065645841247E-324 und 1.79769313486232E308 für positive Werte
CINT	Integer	Ganze Zahl zwischen -32,768 und 32,767, Werte mit Bruchangaben (fractional parts = <i>fp</i>) werden gerundet

		$fp < 0.5$ abgerundet $fp > 0.5$ aufgerundet $fp = 0.5$ gerundet auf nächstliegende gerade Zahl
CLNG	Long Integer	Ganze Zahl zwischen -2,147,483,648 und 2,147,483,647, Werte mit Bruchangaben (fractional parts = fp) werden gerundet $fp < 0.5$ abgerundet $fp > 0.5$ aufgerundet $fp = 0.5$ gerundet auf nächstliegende gerade Zahl
CSNG	Single	Zahl zwischen -3.403823E38 und -1.401298E-45 für negative Zahlen und zwischen 1.401298E-45 und 3.403823E38 für positive Zahlen
CSTR	String	<ul style="list-style-type: none"> •Boolesche Werte ergeben "True" oder "False" •Datumsangaben ergeben kurzes System-Datumsformat •Fehler werden als "Error <number>" angezeigt •Zahlen ergeben einen String mit der Zahl als Zeichen

Beispiel

```
REM Konvertierungsfunktionen Beispiel
PRINT "CBYTE(99.44) = " & CBYTE(99.44)
PRINT "CCUR(9283.066) = " & CCUR(9283.066)
PRINT "CDATE(8/18/98) = " & CDATE("8/18/98")
PRINT "CDBL(3.141593) = " & CDBL("3.141593")
PRINT "CINT(3.141593) = " & CINT("3.141593")
PRINT "CSNG(10) = " & CSNG(10)
PRINT "CSTR(TRUE) = " & CSTR(TRUE)
```

Ausgabe

```
CBYTE(99.44) = 99
CCUR(9283.066) = 9283.066
CDATE(8/18/98) = 8/18/1998
CDBL(3.141593) = 3.141593
CINT(3.141593) = 3
```



```
CSNG(10) = 10  
CSTR(TRUE) = True
```

Verwandte Begriffe

IS Funktion

`COS(number)`

Beschreibung

COS errechnet den Kosinus eines Winkelausdrucks im Bogenmaß. Der erforderliche Parameter, *number*, kann jeder numerische Ausdruck sein. Der Rückgabewert ist eine Fließkommazahl mit doppelter Genauigkeit, im Bereich von -1 bis 1.

Um Grad in Bogenmaß zu wandeln, multiplizieren Sie Grad mit $\pi/180$. Um Bogenmaß in Grad zu wandeln, multiplizieren Sie rad mit $180/\pi$.

Beispiel

```
REM COS Beispiel
'COS errechnet den Kosinus einer Zahl
PRINT "Der Kosinus von 0 ist " & COS(0)
```

Ausgabe

```
Der Kosinus von 0 ist 1
```

Verwandte Begriffe

SIN, TAN

CurrentPath

Beschreibung

CurrentPath gibt einen String mit der Pfadangabe zum aktuellen Programm zurück. Wenn das Programm noch nicht gespeichert wurde, ist er leer. Die globale Eigenschaft kann nur gelesen, nicht gesetzt werden.

Beispiel

```
REM CurrentPath Beispiel
PRINT "Der Pfad zum aktuellen Programm ist "
& CurrentPath
```

Ausgabe

```
Der Pfad zum aktuellen Programm ist
\test.nsb
```

Verwandte Begriffe

DATE

Funktion

DATE

Beschreibung

DATE gibt das aktuelle System-Datum zurück.

Beispiel

```
REM DATE Beispiel
'DATE gibt Systemdatum zurueck
DIM Today
Today = DATE
PRINT "Heute ist " & Today
```

Ausgabe

Heute ist 8/18/1998

(Beispiel- Ausgabe ist systemabhängig)

Verwandte Begriffe

NOW, TIME

Date**Objekt**

ADDOBJECT "Date", *name*, *xpos*, *ypos*, *width*, *height*

Beschreibung

Date wird verwendet, um ein Standard-Datumswahlfeld im Ausgabe-Fenster anzuzeigen. Die erforderliche Komponente, *name*, ist der Name der Variablen, die dem Programm als Referenz auf dieses Objekt hinzugefügt wurde. Sie muss den Standard-Nameskonventionen für Variablen entsprechen. Die erforderlichen Komponenten, *xpos*, *ypos*, *width*, und *height* sind numerische Ausdrücke, die zusammen die Größe und Position (in Pixeln) des Objekts bestimmen, gemessen von der linken oberen Ecke. Verwenden Sie keine MSGBOX innerhalb Ihres Date_Change Ereignisses: sie erzeugt einen Fehler. Dieses Objekt ist nicht auf Windows CE 2.0 Geräten verfügbar.

Unterstützte Eigenschaften (siehe "Eigenschaften")

BorderStyle, Date, Enabled, FontBold, FontItalic, FontName, FontSize, FontStrikethru, FontUnderline, LongFormat, Height, Hwnd, Left, Name, ParentHwnd, TabStop, Tag, Text, Timer, Top, Visible, Width, WindowLong

Unterstützte Methoden (siehe "Methoden")

Hide, Move, SetFocus, Show

Unterstützte Ereignisse (siehe "Ereignisse")

Change, DropDown

Beispiel

```
REM Date Beispiel
ADDOBJECT "Date", "Datum", 0,0,90,20
Datum.fontBold=true
SUB Datum_Dropdown
    PRINT "Datum Dropdown"
END SUB
SUB Datum_Change
    PRINT "Datum geaendert auf " & Datum.Date
END SUB
```

Ausgabe

Verwandte Begriffe

ADDOBJECT, Events, Methods, Properties

DATEADD

Funktion

DATEADD(*interval*, *number*, *date*)

Beschreibung

DATEADD gibt das Datum zurück, das man erhält, wenn eine bestimmte Anzahl von Intervallen einem vorgegebenen Datum hinzugerechnet werden. Der erforderliche Parameter, *interval*, ist eine Zeichenkette, die den Typ des zu addierenden Intervalls bestimmt, siehe Tabelle 6 weiter unten. Der erforderliche Parameter, *number*, kann jeder numerische Ausdruck sein, der die Anzahl der zu addierenden Intervalle darstellt. Der ebenfalls erforderliche Parameter, *date*, kann jeder Ausdruck sein, der ein Datum repräsentiert.

Tabelle 6: Intervallwerte

Wert	Beschreibung
yyyy	Jahr
q	Quartal
m	Monat
y	Tag im Jahr
d	Tag
w	Wochentag
ww	Woche im Jahr
h	Stunde
n	Minute
s	Sekunde

Beispiel

```
REM DATEADD Beispiel
'DATEADD addiert Intervalle zum Datum
PRINT "+10 Sekunden:", DATEADD("s", 10, NOW)
PRINT "-1 Jahr:", DATEADD("yyyy", -1, NOW)
```

Ausgabe

```
+10 Sekunden:  8/18/98 10:52:54 PM
-1 Jahr:       8/18/97 10:52:44 PM
```

Verwandte Begriffe

DATEDIFF, DATEPART

DATEDIFF**Funktion**

DATEDIFF(*interval*, *date1*, *date2*[, *firstdayofweek*[, *firstweekofyear*]])

Beschreibung

DATEDIFF gibt die Anzahl von Intervallen zwischen zwei Datumsangaben zurück. Der erforderliche Parameter, *interval*, ist eine Zeichenkette, siehe Tabelle 6. Die ebenfalls erforderlichen Parameter, *date1* und *date2*, können beliebige Datumsangaben sein. Der optionale Parameter *firstdayofweek* ist Sonntag, falls nicht anders vorgegeben. Der optionale Parameter *firstweekofyear* entspricht der Woche, die den 1. Januar enthält, falls nicht anders vorgegeben.

Tabelle 7: firstdayofweek Konstanten

Konstante	Wert	Beschreibung
vbUseSystem	0	NLS API Einstellung
vbSunday	1	Sonntag (Vorgabe)
vbMonday	2	Montag
vbTuesday	3	Dienstag
vbWednesday	4	Mittwoch
vbThursday	5	Donnerstag
vbFriday	6	Freitag
vbSaturday	7	Samstag

Tabelle 8: firstweekofyear Konstanten

Konstante	Wert	Beschreibung
vbUseSystem	0	NLS API Einstellung
vbFirstJan1	1	Woche mit 1. Januar
vbFirstFourDays	2	Erste Woche im Jahr mit mindestens 4 Tagen
vbFirstFullWeek	3	Erste volle Woche im Jahr

Beispiel

```
REM DATEDIFF Beispiel
'DATEDIFF errechnet Differenz zweier Termine
DIM Born
Born = INPUTBOX("Ihr Geburtsdatum: ")
Born = CDATE(Born)
PRINT "Seit " & Born & " sind"
```

```
PRINT DATEDIFF("d", Born, NOW) & " Tage  
vergangen"  
PRINT "oder"  
PRINT DATEDIFF("n", Born, NOW) & " Minuten"
```

Ausgabe

Seit 12/27/1970 sind
10096 Tage vergangen
oder
14539612 Minuten

(Beispiel- Ausgabe ist systemabhängig)

Verwandte Begriffe

DATEADD, DATEPART

DATEPART

Funktion

DATEPART(*interval*, *date*[, *firstdayofweek*[,
firstweekofyear]])

Beschreibung

DATEPART gibt eine Zahl zurück, die einem Ausschnitt eines vorgegebenen Datums entspricht. Der erforderliche Parameter, *interval*, bestimmt, welcher Ausschnitt des Datums berechnet und zurück gegeben wird. Der optionale Parameter *firstweekofyear* entspricht der Woche, die den 1. Januar enthält, falls nicht anders vorgegeben.

Beispiel

```
REM DATEPART Beispiel
'DATEPART ergibt Datums-Ausschnitt als Zahl
DIM QuarterPart, MonthPart, DayPart
QuarterPart = DATEPART("q", NOW)
MonthPart = DATEPART("m", NOW)
DayPart = DATEPART("d", NOW)
PRINT "Heute ist Tag " & DayPart
PRINT "des Monats " & MonthPart
PRINT "im Quartal " & QuarterPart
```

Ausgabe

```
Heute ist Tag 18
des Monats 8
im Quartal 3
(Beispiel-Ausgabe vom 18. August 1998)
```

Verwandte Begriffe

DATEADD, DATEDIFF

DATESERIAL

Funktion

DATESERIAL(*year*, *month*, *day*)

Beschreibung

DATESERIAL gibt ein Datum als Konstrukt der vorgegebenen Werte für *year*, *month*, und *day* aus. Der erforderliche Parameter, *year*, kann jeder numerische Ausdruck im Bereich von 100 bis 9999 sein. Die erforderlichen Parameter, *minute* und *second*, können beliebige numerische Ausdrücke sein.

Beispiel

```
REM DATESERIAL Beispiel
'DATESERIAL setzt ein Datum zusammen
DIM IndepDay, Birthday
IndepDay = DATESERIAL(1776, 7, 4)
Birthday = DATESERIAL(1957, 08, 24)
PRINT "Unabhaengigkeitstag:", IndepDay
PRINT "Mein Geburtstag:", Birthday
```

Ausgabe

```
Unabhaengigkeitstag:  7/4/1776
Mein Geburtstag:      08/24/1957
```

(Anzeige des Beispiels ist systemabhängig)

Verwandte Begriffe

TIMESERIAL

DATEVALUE(*date*)**Beschreibung**

DATEVALUE gibt ein Datum zurück. Der erforderliche Parameter, *date*, ist üblicherweise ein String, aber es kann jeder Ausdruck verwendet werden, der ein Datum im Bereich von 1. Januar 100 bis 31. Dezember 9999 repräsentiert.

Wenn die Zeichenkette *date* aus Zahlen besteht, die durch das Separatorzeichen für Datum getrennt sind, werden Monate, Tage und Jahre in der systemeigenen Kurzdarstellung des Datumsformats erkannt. Eine fehlende Jahresangabe wird durch das aktuelle Jahr ersetzt.

Eindeutige Monatsnamen werden sowohl abgekürzt, als auch in Langform erkannt.

Separatorzeichen für Datum sind Zeichen, die Tag, Monat und Jahr voneinander abgrenzen, wenn das Datum als formatierte Zeichenkette vorliegt. Sie sind systemabhängig.

Beispiel

```
REM DATEVALUE Beispiel
'DATEVALUE gibt ein Datum zurueck
DIM IndepDay, Birthday
IndepDay = DATEVALUE("July 4, 1776")
Birthday = DATEVALUE("Aug 24 1957")
PRINT "Unabhaengigkeitstag:", IndepDay
PRINT "Mein Geburtstag:", Birthday
```

Ausgabe

```
Unabhaengigkeitstag: 7/4/1776
Mein Geburtstag:      08/24/1957
```

(Anzeige des Beispiels ist systemabhängig)

Verwandte Begriffe

FORMAT, TIMEVALUE

DAY(*date*)

Beschreibung

DAY gibt eine ganze Zahl im Bereich von 1 bis 31 zurück, die den Tag im Monat für ein vorgegebenes Datum repräsentiert. Der erforderliche Parameter, *date*, kann jeder Ausdruck sein, der ein Datum darstellt.

Beispiel

```
REM DAY Beispiel
'DAY gibt eine Nummer fuer den Tag im Monat
'zurueck
DIM Today, NextQuarter
Today = DATE
NextQuarter = Today + 90
PRINT "Der Tag heute ist " & DAY(Today)
PRINT "90 Tage spaeter ist " _
      & DAY(NextQuarter)
```

Ausgabe

```
Der Tag heute ist 18
90 Tage spaeter ist 16
```

Verwandte Begriffe

HOUR, MINUTE, MONTH, NOW, SECOND, WEEKDAY, YEAR

DECLARE

Befehl

```
DECLARE "Sub name Lib "" libname"" [Alias  
""aliasname"" ]([arglist])"
```

```
Declare "Function name Lib "" libname"" [Alias  
""aliasname"" ]([arglist]) [As type]"
```

Beschreibung

Deklariert eine API Funktion zur Verwendung. Der *name* ist der Name der Funktion, den Sie innerhalb des NS Basic Programms für diese Funktion benutzen wollen. Wenn dieser Name sich vom Funktionsnamen innerhalb der DLL unterscheidet, verwenden Sie die Alias-Klausel, um den DLL-internen Namen anzugeben. Das ist dann hilfreich, wenn es zu Namenskonflikten mit anderen internen Namen kommt. Der *libname* ist der Name der DLL-Bibliothek. Dabei kann es sich um den vollständigen Pfadnamen, einen teilweisen Pfadnamen oder einen Dateinamen handeln. Letzteres veranlasst das System dazu, nach der DLL zu suchen. Die meisten gängigen DLL's werden im \Windows Verzeichnis abgelegt, wodurch auf die Angabe des Pfadnamens verzichtet werden kann und die Suche sehr schnell erfolgt. In der Tech Note on Declare finden Sie weitere Dokumentation zu diesem Thema.

Beispiel

```
REM Hole Geraetespeicher-Infos  
DIM lpAvailable, lpTotalBytes, lpTotalFree  
DECLARE "Funktion GetDiskFreeSpaceExW Lib  
""Coredll"" (_  
    ByVal lpDirectoryName As String, _  
    ByRef lpFreeBytesAvailable As Long, _  
    ByRef lpTotalNumberOfBytes As Long, _  
    ByRef lpTotalNumberOfFreeBytes As Long) _  
    As Boolean"  
res = GetDiskFreeSpaceExW("e:", _  
    lpAvailable, lpTotalBytes, lpTotalFree)  
MsgBox "Verfuegbar: " & lpAvailable
```

Ausgabe

Verfuegbar: 123,000,004

Verwandte Begriffe

DIM**Befehl**

```
DIM nameA[[subscriptA [,subscriptB [,subscriptC...]]]]  
[, nameB...[, nameC..., [...]]]
```

Beschreibung

DIM wird verwendet, um Variablen zu deklarieren und Speicherplatz zu reservieren. Die erforderliche Komponente, *nameA*, ist der Name der Variablen. Er muss den Standard-Namens-Konventionen entsprechen. Die optionale Liste von Indizes, *subscripts*, stellt die Obergrenzen der Dimensionen eines Array's dar. Es können bis zu 60, durch Kommata getrennte Dimensionen deklariert werden. Variablen auf Skriptebene stehen allen Prozeduren des Skripts zur Verfügung; Variablen, die in Prozeduren deklariert werden, sind nur in der deklarierenden Prozedur verfügbar. Die Untergrenze aller Array-Indizes ist immer 0.

Beispiel

```
REM DIM Beispiel  
'DIM Variablen deklarieren und Speicher  
'reservieren  
'Leere Variable namens "Foo"  
DIM Foo  
'Ein-dimensionales Array mit 10 Elementen  
DIM OneD(9)  
'Zwei-dimensional Array mit 600 Elementen  
'20 x 30  
DIM TwoD(19, 29)
```

Verwandte Begriffe

ARRAY, REDIM, SET

DOEVENTS

Beschreibung

DOEVENTS wird verwendet, um während langer, intensiver Schleifen die Verarbeitung von Ereignissen zu ermöglichen. Wenn DOEVENTS aufgerufen wird, werden alle Bildschirm- oder Timer-Ereignisse verarbeitet und danach wird die Schleife fortgesetzt.

Beispiel

```
REM 'DOEVENTS Beispiel
'DOEVENTS erlaubt die Abarbeitung von
'Screen-Events waehrend langer Schleifen
FOR i = 0 TO 100000
    j = j + 1
    DOEVENTS
NEXT
```

Ausgabe

(nichts in unserem Fall – aber andere Ereignisse könnten abgearbeitet werden)

Verwandte Begriffe

DO...LOOP

Befehl

```
DO [WHILE | UNTIL condition]  
  [statements]  
[EXIT DO]  
  [statements]  
LOOP  
oder  
DO  
  [statements]  
[EXIT DO]  
  [statements]  
LOOP [WHILE | UNTIL condition]
```

Beschreibung

DO...LOOP wiederholt einen Befehlsblock (*statements*) während (WHILE) eine vorgegebene Bedingung TRUE ist oder bis (UNTIL) eine vorgegebene Bedingung TRUE wird. Die erforderliche Komponente, *condition*, kann jeder Ausdruck sein, der TRUE oder FALSE ergibt. Die optionale Komponente, *statements*, sind Befehle, die während des Schleifendurchlaufs ausgeführt werden. Es kann eine beliebige Anzahl von optionalen EXIT DO Befehlen zum vorzeitigen Beenden der Schleife verwendet werden. DO...LOOP Befehle können verschachtelt werden und jeder EXIT DO Befehl in einer solchen Verschachtelung übergibt die Programmausführung an die übergeordnete Ebene der Schleife.

Wird die WHILE/UNTIL Klausel direkt hinter DO gesetzt, so wird *condition* ausgewertet, bevor der erste Schleifendurchlauf erfolgt, d.h. wenn die *condition* FALSE ist, werden die *statements* niemals ausgeführt. Befindet sich die WHILE/UNTIL Klausel hinter der LOOP Klausel, wird der Schleifenkörper ausgeführt, bevor die Bedingung ausgewertet wird, d.h. die *statements* werden immer mindestens einmal ausgeführt.

Beispiel

```
REM DO...LOOP Beispiel  
'DO...LOOP wiederholt Befehlsblock  
DIM Counter  
Counter = 1  
'Schleife gibt 1 bis 5 aus  
DO WHILE Counter < 6  
  PRINT "DO WHILE...LOOP:", Counter  
  Counter = Counter + 1
```



```

LOOP
PRINT
'Endlosschleife mit EXIT DO zum Abbrechen
Counter = 1000
DO
    PRINT "Endlosschleife:", Counter
    IF Counter >= 1000000 THEN
        EXIT DO
    Counter = Counter * 10
LOOP

```

Ausgabe

```

DO WHILE...LOOP:      1
DO WHILE...LOOP:      2
DO WHILE...LOOP:      3
DO WHILE...LOOP:      4
DO WHILE...LOOP:      5
Infinite Loop: 1000
Infinite Loop: 10000
Infinite Loop: 100000
Infinite Loop: 1000000

```

Verwandte Begriffe

EXIT, FOR EACH...NEXT, FOR...NEXT, WHILE...WEND

result = x EQV y

Beschreibung

EQV gibt das Ergebnis der logischen Äquivalenzfunktion zweier Ausdrücke zurück. *result* ist TRUE, wenn beide Ausdrücke x und y TRUE oder beide Ausdrücke x und y FALSE ergeben, ansonsten ist *result* FALSE.

EQV führt auch eine bitweise Verknüpfung zweier numerischer Werte durch. Jedes Bit in *result* wird auf 1 gesetzt, wenn beide korrespondierenden Bits in x und y 1 oder beide korrespondierenden Bits in x und y 0 sind, ansonsten wird es auf 0 gesetzt.

Beispiel

```
REM EQV Beispiel
'EQV fuehrt logische und bitweise
'Aequivalenzpruefung durch
DIM Test1, Test2, Test3, x, y
x = 4
y = 9
Test1 = x < 0 EQV y < 10
Test2 = x > 0 EQV y > 10
Test3 = x < 0 EQV y > 10
PRINT "Logisch:"
PRINT "  x < 0 EQV y < 10 = " & Test1
PRINT "  x > 0 EQV y > 10 = " & Test2
PRINT "  x < 0 EQV y > 10 = " & Test3
PRINT "Bitweise:"
PRINT "  x EQV y = " & (x EQV y)
```

Ausgabe

```
Logisch:
  x < 0 EQV y < 10 = False
  x > 0 EQV y > 10 = False
  x < 0 EQV y > 10 = True
Bitweise:
  x EQV y = -14
```

Verwandte Begriffe

AND, IMP, NOT, OR, XOR

ERASE *arrays*

Beschreibung

ERASE reinitialisiert Arrays mit fester Größe und gibt den Speicherplatz dynamischer Arrays frei. Die erforderliche Komponente, *arrays*, ist eine durch Kommata getrennte Liste von Arrayvariablen.

Beispiel

```
REM ERASE Beispiel
'ERASE reinitialisiert Arrays
DIM Children(3)
Children(0) = "Eric"
Children(1) = "Kenny"
Children(2) = "Kyle"
Children(3) = "Stan"
PrintArray Children, 4
ERASE Children
PrintArray Children, 4
FUNCTION PrintArray(arr, elements)
    DIM i
    FOR i = 1 to elements
        PRINT "#" & i & ":", "(" & arr(i - 1) & ")"
    NEXT
    PRINT
END FUNCTION
```

Ausgabe

```
#1:      (Eric)
#2:      (Kenny)
#3:      (Kyle)
#4:      (Stan)
#1:      ()
#2:      ()
#3:      ()
#4:      ()
```

Verwandte Begriffe

DIM, ARRAY

ERR

Objekt

Err

Err.Number

Err.Description

Err.Source

Err.Clear

Err.Raise

Beschreibung

Das Err-Objekt wird benutzt, um Laufzeitfehler zu bearbeiten. Das Err-Objekt ist immer verfügbar während das Programm ausgeführt wird. Es muss nicht mittels ADDOBJECT eingebunden werden. Wenn ein Fehler auftritt, werden die Number, Description und Source Eigenschaft des Err-Objekts gesetzt. Sobald ein Fehler abgearbeitet wurde, wird das Err-Objekt per Clear Methode zurückgesetzt, d.h. alle Eigenschaften werden gelöscht. Mit der Raise Methode kann ein Error-Ereignis ausgelöst werden.

Beispiel

```
REM Err Objekt Beispiel
'Err Objekt bearbeitet Laufzeitfehler
DO_DIV0(3)
SUB DO_DIV0(Num)
    ON ERROR RESUME NEXT
    PRINT Num / 0
    IF Err THEN
        PRINT Err.Number, Err.Description
        Err.Clear
    END IF
END SUB
```

Ausgabe

```
11      Division by zero
```

Verwandte Begriffe

ON ERROR

ESCAPE	Funktion
---------------	-----------------

ESCAPE(*string*)

UNESCAPE(*string*)

Beschreibung

ESCAPE gibt einen String zurück, bei dem alle Sonderzeichen durch ein %-Zeichen gefolgt von ihrem Hex-Code ersetzt wurden.

UNESCAPE kehrt die durch ESCAPE erzeugte Umwandlung um.

Nur bei Windows CE 4.0 und aktueller.

Beispiel

```
PRINT ESCAPE ("ABC!@#%$")
PRINT UNESCAPE ("ABC%0d%0aDEF")
```

Ausgabe

ABC%21@%23%24%25

ABC

DEF

Verwandte Begriffe

ESCAPE, UNESCAPE

EVAL(*string*)

Beschreibung

EVAL gibt einen Wert zurück, der dadurch erzeugt wird, dass der Ausdruck *string* wie eine Funktion behandelt wird. Der erforderliche Parameter, *string*, ist der String-Ausdruck der ausgeführt wird. Wenn mehrere Befehle ausgeführt werden sollen, trennen Sie diese mit einem Carriage Return (vbCRLF). Der temporären, virtuellen Prozedur, die erzeugt wird, werden die Werte ByVal übergeben, so dass die ursprünglichen Variablen durch die EVAL Funktion nicht verändert werden.

Beispiel

```
REM EVAL Beispiel
'EVAL fuehrt String als FUNKTION aus
DIM x
x = 5
PRINT EVAL("x")
EVAL("x = x * 10")
PRINT x
```

Ausgabe

```
5
5
```

Verwandte Begriffe

EXECUTE

Events

Objekt

```
[PUBLIC] SUB ObjectName_Event[(arglist)]  
    [statements]  
END SUB
```

Beschreibung

Objekt Events (Ereignisse) werden entweder durch ein Programm oder durch eine Benutzereingabe ausgelöst. Wenn ein Ereignis ausgelöst wurde, ruft ein Objekt eine PUBLIC SUB Prozedur innerhalb des Programms auf. Der Name dieser Prozedur, *ObjectName_Event*, ist eine Kombination aus dem Objektnamen und dem Ereignisnamen. Die optionale Komponente, *arglist*, ist eine durch Kommata getrennte Liste von Argumenten, die der Prozedur beim Aufruf übergeben werden könnte. Weiter unten sind einige wichtige Objekt Ereignisse aufgelistet.

Anmerkung: Wenn ein Programm keine Prozedur zum Abarbeiten eines Ereignisses hat, wird keine Fehlermeldung ausgegeben.

Tabelle 9: Objekt Ereignisse

Ereignis	Kommentar
Change	ComboBox (Element ausgewählt oder Texteingabe), ListBox (Element ausgewählt), TextBox (Text geändert)
Click	
DblClick	ListBox, TextBox
DropDown	ComboBox, Date
GotFocus	Objekt aktiviert für Tastatureingaben
KeyDown	Keycode, shift als Argumente Shift=1, 2-CTRL, 4=Alt
KeyPress	Zeichen als Argument
KeyUp	Keycode, shift als Argumente Shift=1, 2-CTRL, 4=Alt
LostFocus	Objekt deaktiviert
Timer	CheckBox, ComboBox, CommandButton, Frame, HScrollBar, Label, ListBox, OptionButton, TextBox, VScrollBar

Beispiel

```
REM Objekt Ereignisse Beispiel
```

```

'Objekt Ereignisprozedur im Programm
DIM When
ADDOBJECT "ComboBox", "Combo", 5, 30, 150, 80
Combo.Style = 2
SUB Combo_DropDown
    Combo.Clear
    Combo.AddItem DATE
    Combo.AddItem DATEADD("d", 1, DATE)
    Combo.AddItem DATEADD("ww", 1, DATE)
END SUB
SUB Combo_Click
    When = Combo.Text
    PRINT "Gewaehltes Element: " & When
END SUB

```

Ausgabe

Verwandte Begriffe

Methods, Properties

EXECUTE(*string*)

EXECUTE("file:|ascii:|unicode:" & *string*)

Beschreibung

EXECUTE führt einen Ausdruck oder eine Datei aus, als ob der Code direkt ins Programm integriert wäre. Der erforderliche Parameter, *string*, kann ein String-Ausdruck oder der Name einer Datei sein, die den auszuführenden Code enthält. Wenn mehrere Befehle ausgeführt werden sollen, trennen Sie diese mit einem Carriage Return (vbCRLF). Der Code kann alle Variablen des laufenden Programms lesen und ändern.

Wenn *string* mit "file:" oder "ascii:" beginnt, wird der Rest als Pfadname zu einer ASCII-Datei interpretiert. Startet *string* mit "unicode:" wird eine Datei im Unicode-Format erwartet. Die Verwendung eines Dateinamen ist nur während der Entwicklung mit NS Basic/CE möglich – nicht in der Laufzeit-Umgebung.

Beispiel

```
REM EXECUTE Beispiel
'EXECUTE String als SUB ausfuehren
DIM x
x = 5
EXECUTE("PRINT x * 10")
EXECUTE("x = x * 10")
' text.txt ist eine Datei mit Inhalt
' "PRINT 50"
EXECUTE("ascii:\my documents-est.txt")
PRINT x
```

Ausgabe

```
50
50
50
```

Verwandte Begriffe

EVAL, EXECUTEGLOBAL

EXECUTEGLOBAL(*string*)

Beschreibung

EXECUTEGLOBAL führt einen oder mehrere Befehle im globalen Namensraum eines Skriptes aus. Der erforderliche Parameter, *string*, kann entweder ein String-Ausdruck oder der Name einer Datei sein, die auszuführende Befehle enthält. Wenn mehrere Befehle ausgeführt werden sollen, trennen Sie diese mit einem Carriage Return (vbCRLF) oder Doppelpunkt. Alle Befehle, die mit EXECUTEGLOBAL ausgeführt werden, verwenden den globalen Namensraum des Programms. Damit erreicht man, dass dem Programm Codezeilen hinzugefügt werden können, die allen Prozeduren zugänglich sind. So kann zum Beispiel ein CLASS Befehl zur Laufzeit ausgeführt werden und Funktionen können danach neue Instanzen dieser Klasse erstellen.

Prozeduren und Klassen zur Laufzeit hinzuzufügen kann hilfreich sein, wirkt aber auch die Gefahr, dass existierende globale Funktionen und Variablen überschrieben werden. Wenn Sie auf eine Variable oder Funktion nicht außerhalb einer Prozedur zugreifen müssen, verwenden Sie den EXECUTE Befehl, der nur die aufrufende Funktion beeinflusst.

Beispiel

```
REM EXECUTEGLOBAL Beispiel
'EXECUTEGLOBAL fuehrt String als SUB aus
DIM x
x = ("DIM y" & vbcrLf & "y = 4 * 10" & _
    vbcrLf & "Print y")
EXECUTEGLOBAL x
```

Ausgabe

40

Verwandte Begriffe

EVAL, EXECUTE

EXIT**Befehl**

EXIT DO
EXIT FOR
EXIT FUNCTION
EXIT SUB

Beschreibung

EXIT beendet die Ausführung eines Code-Blocks in einer DO...LOOP, FOR...NEXT, FOR EACH...NEXT, FUNCTION, oder SUB Funktion. Wenn er zum Ausstieg aus einer DO...LOOP, FOR...NEXT, oder FOR EACH...NEXT Schleife verwendet wird, fährt das Programm mit dem ersten Befehl nach der Schleife fort. Handelt es sich um eine verschachtelte Schleife, wird die Kontrolle an die nächst höhere Schleife über der, in der EXIT erkannt wurde, übergeben. Wird er zum Beenden einer FUNCTION oder SUB Prozedur verwendet, fährt das Programm mit dem ersten Befehl nach dem Prozeduraufruf fort.

Beispiel

```
REM EXIT Beispiel
'EXIT beendet Loops und Prozeduren
DIM i
FOR i = 1 to 10
  IF i > 1 THEN
    EXIT FOR
  END IF
  PRINT "Versuche nichts zu tun"
  DoNothing
NEXT
PRINT "Geschafft"
SUB DoNothing
  EXIT SUB
  PRINT "Dieser Befehl wird nie ausgefuehrt"
END SUB
```

Ausgabe

Versuche nichts zu tun
Geschafft

Verwandte Begriffe

FOR...NEXT, DO...LOOP, FUNKTION, PROPERTY, SUB

EXP (*number*)

Beschreibung

EXP gibt eine Fließkommazahl doppelter Genauigkeit zurück, die e^{number} entspricht. Der erforderliche Parameter, *number*, kann jeder numerische Ausdruck sein. e ist die Basis des natürlichen Logarithmus. Ihr Wert beträgt angenähert 2,718282.

Beispiel

```
REM EXP Beispiel
'EXP potenziert eine Zahl
PRINT "EXP(0) = " & EXP(0)
PRINT "e = " & EXP(1)
```

Ausgabe

```
EXP(0) = 1
e = 2.718282
```

Verwandte Begriffe

LOG

FILTER

Funktion

`FILTER (stringarray, value[, include[, compare]])`

Beschreibung

FILTER gibt ein String-Array zurück, das die Untermenge der Eingangsdaten enthält, die den Filterkriterien entsprechen. Der erforderliche Parameter, *stringarray*, ist das zu filternde eindimensionale String-Array. Der ebenfalls erforderliche Parameter, *value*, ist die Zeichenfolge, nach der gesucht werden soll. Der optionale Parameter, *include*, ist ein boolescher Wert. Ist er TRUE, werden die Werte zurückgegeben, die *value* enthalten. Ist er FALSE, enthalten die zurückgegebenen Werte *value* nicht. Der Vorgabewert für *include* ist TRUE. Der optionale Parameter, *compare*, ist ein numerischer Ausdruck oder eine Konstante, die den durchzuführenden Vergleichstyp bestimmt, siehe unten. Der Vorgabewert für *compare* ist vbBinaryCompare.

Table 10: Vergleichskonstanten constants

Konstante	Wert	Beschreibung
vbBinaryCompare	0	Binärer Vergleich, Groß-Kleinschreibung wird unterschieden (Vorgabe)
vbTextCompare	1	Zeichenvergleich, Groß-Kleinschreibung wird nicht unterschieden

Beispiel

```
REM FILTER Beispiel
'FILTER findet Uebereinstimmungen in einem
'String-Array
DIM Who, TheKs, NotEric
Who = ARRAY("Eric", "Kenny", "Kyle", "Stan")
TheKs = FILTER(Who, "k", TRUE, vbTextCompare)
NotEric = FILTER(Who, "Eric", FALSE, _
    vbBinaryCompare)
PrintArray "Wer", Who
PrintArray "Die K's", TheKs
PrintArray "Jeder ausser Eric", NotEric
SUB PrintArray(ArrName, Arr)
    DIM i
    PRINT ArrName
    FOR i = 0 TO UBOUND(Arr)
        PRINT " " & Arr(i)
    NEXT
END SUB
```

Ausgabe

Wer
Eric
Kenny
Kyle
Stan
Die K's
Kenny
Kyle
Jeder ausser Eric
Kenny
Kyle
Stan

Verwandte Begriffe

REPLACE

FIX(*number*)

Beschreibung

FIX entfernt den gebrochenen Anteil einer Zahl und gibt die der 0 am nächsten liegende ganze Zahl zurück. Der erforderliche Parameter, *number*, kann jeder numerische Ausdruck sein. Wenn *number* positiv ist, wird die nächst kleinere ganze Zahl zurückgegeben; wenn *number* negativ ist, wird die nächstgrößere ganze Zahl zurückgegeben.

Beispiel

```
REM FIX Beispiel
'FIX Fliesskommazahl in naechste passende
'ganze Zahl in Richtung 0
DIM Pos, Neg
Pos = EXP(1)
Neg = -EXP(1)
PRINT "FIX(e) = " & FIX(Pos)
PRINT "FIX(-e) = " & FIX(Neg)
```

Ausgabe

```
FIX(e) = 2
FIX(-e) = -2
```

Verwandte Begriffe

INT

FOR...NEXT	Befehl
-------------------	---------------

FOR *counter* = *start* TO *end* [STEP *step*]

 [*statements*]

 [EXIT FOR]

 [*statements*]

NEXT

Beschreibung

FOR...NEXT wiederholt eine Gruppe von Befehlen. Die erforderliche Komponente, *counter*, ist eine Zahl mittels der auf den Schleifenzähler zugegriffen werden kann. Die erforderliche Komponenten, *start* und *end*, sind der Start- bzw. Endwert von *counter*. Beide können durch numerische Ausdrücke gebildet werden. Der optionale Parameter, *step*, kann verwendet werden, um eine Inkrementierungs-Schrittweite anzugeben, die Vorgabe ist 1. Die optionale Komponente, *statements*, wird bei jedem Schleifendurchgang ausgeführt. Es kann eine beliebige Anzahl von optionalen EXIT FOR Befehlen zum vorzeitigen Beenden der Schleife verwendet werden. FOR...NEXT Befehle können verschachtelt werden und jeder EXIT FOR Befehl in einer solchen Verschachtelung übergibt die Programmausführung an die übergeordnete Ebene der Schleife.

Beispiel

```
REM FOR...NEXT Beispiel
'FOR...NEXT wiederholt Befehlsgruppe
DIM Puppets
Puppets = ARRAY("Hat", "Twig")
FOR i = 0 to 1
    PRINT "Puppe: Hr. " & Puppets(i)
NEXT
FOR i = 0 to 10 STEP 5
    PRINT i
NEXT
```

Ausgabe

```
Puppe: Hr. Hat
Puppe: Hr. Twig
0
5
10
```

Verwandte Begriffe

DO...LOOP, EXIT, FOR EACH...NEXT, WHILE...WEND

FOR EACH...NEXT

Befehl

FOR EACH *element* IN *group*

[*statements*]

[EXIT FOR]

[*statements*]

NEXT [*element*]

Beschreibung

FOR EACH...NEXT wiederholt für jedes Element eines Arrays oder einer Kollektion eine Gruppe von Befehlen. Der erforderliche Parameter, *element*, ist ein Variablenname mit dem auf das aktuelle Element zugegriffen werden kann. Der erforderliche Parameter, *group*, ist der name eines Arrays oder einer Kollektion von Objekten. Die optionale Komponente, *statements*, wird bei jedem Durchlaufen der Schleife ausgeführt. Es kann eine beliebige Anzahl von optionalen EXIT FOR Befehlen zum vorzeitigen Beenden der Schleife verwendet werden. FOR EACH...NEXT Befehle können verschachtelt werden und jeder EXIT FOR Befehl in einer solchen Verschachtelung übergibt die Programmausführung an die übergeordnete Ebene der Schleife.

Anmerkung: FOR EACH...NEXT funktioniert nicht mit Arrays, die benutzerdefinierte Datentypen enthalten.

Beispiel

```
REM FOR EACH...NEXT Beispiel
DIM School
School = ARRAY("Rektor", "Mr. Garrison", _
    "Vorgesetzter")
FOR EACH Employee IN School
    PRINT "Angestellter der Schule:", Employee
NEXT
```

Ausgabe

```
Angestellter der Schule:      Rektor
Angestellter der Schule:      Mr. Garrison
Angestellter der Schule:      Chef
```

Verwandte Begriffe

DO...LOOP, EXIT, FOR...NEXT, WHILE...WEND

Format	Funktion
--------	----------

FORMATCURRENCY(*expression*[, *fractionaldigits*[,
leadingdigit[, *parensfornegative*[, *groupdigits*]]]])

FORMATDATETIME(*date*[, *formatname*])

FORMATNUMBER(*expression*[, *fractionaldigits*[,
leadingdigit[, *parensfornegative*[, *groupdigits*]]]])

FORMATPERCENT(*expression*[, *fractionaldigits*[,
leadingdigit[, *parensfornegative*[, *groupdigits*]]]])

Beschreibung

FORMATCURRENCY, FORMATNUMBER, und FORMATPERCENT geben einen String zurück, der die als Währung, Datum, Nummer oder Prozentzahl formatierte Darstellung des übergebenen Ausdrucks repräsentiert. Der erforderliche Parameter, *expression*, kann jeder gültige Ausdruck des passenden Typs sein. Der optionale Parameter, *fractionaldigits*, ist ein numerischer Ausdruck, der die Nachkommastellen angibt. Die Vorgabe ist -1, wodurch die Systemeinstellungen verwendet werden. Die optionalen Parameter, *leadingdigit*, *parensfornegative* und *groupdigits*, sind numerische Konstanten, die nachfolgend beschrieben sind. *leadingdigit* gibt an, ob eine führende Null bei Zahlen angezeigt werden soll, die nur Nachkommastellen enthalten. *parensfornegative* legt fest, ob negative Werte in Klammern angezeigt werden. *groupdigits* gibt an, ob Zahlengruppen mit einem Gruppen-Separator angezeigt werden sollen.

Tabelle 11: Tristate Werte

Name	Wert	Beschreibung
True	-1	True
False	0	False
UseDefault	-2	Systemvorgabe nutzen

FORMATDATETIME gibt einen String zurück, der durch Formatierung eines Datums gewonnen wird. Der erforderliche Parameter, *date*, kann jeder Ausdruck sein, der ein gültiges Datum darstellt. Der optionale Parameter, *formatname*, ist ein numerischer Ausdruck oder eine Konstante die angeben, wie das Datum zu formatieren ist.

Tabelle 12: Konstanten für Formatnamen

Konstante	Wer	Beschreibung
vbGeneralDate	0	Datum kurz, Zeit lang

vbLongDate	1	Datum lang
vbShortDate	2	Datum kurz
vbLongTime	3	Zeit lang
vbShortTime	4	Zeit kurz

Beispiel

```
REM Format Funktionen
DIM UseDefault
UseDefault=-2
'Waehrung
PRINT FORMATCURRENCY(-3.5, -1, _
    TristateUseDefault, True)
PRINT FORMATCURRENCY(123456, 0, True, False,
    True)
'Date/Time
PRINT FORMATDATETIME(NOW)
PRINT FORMATDATETIME(Birthdate, vbLongDate)
'General Numbers
PRINT FORMATNUMBER(-0.1429, 6, False)
PRINT FORMATNUMBER(987654.321, 3, True,
    False, True)
'Percentages
PRINT FORMATPERCENT(0.007, 2, False)
PRINT FORMATPERCENT(1234.56, 0, True, False,
    False)
```

Ausgabe

```
($3.50)
$123,456
8/18/1998 10:44 PM
August 18, 1998
-.142900
987,654.321
.70%
123456%
```

(Anzeige des Beispiels ist systemabhängig)

Verwandte Begriffe

ADDOBJECT "Frame", *name*, *xpos*, *ypos*, *width*, *height*

Beschreibung

Ein Rahmen (Frame) ist ein Container, der andere Objekte aufnehmen kann. Um ein Objekt als Kind eines Frame zu definieren, setzen Sie die ParentHwnd Eigenschaft des Kindes auf die Hwnd Eigenschaft des Rahmens. Methoden, die dem Frame zugeordnet wurden, gelten automatisch auch für die Abkömmlinge. Die Grenzen des Kindobjekts beziehen sich auf den Frame in dem es enthalten ist.

Unterstützte Eigenschaften (siehe "Eigenschaften")

BackColor, Caption, Enabled, Font, FontBold, FontItalic, FontStrikeThru, FontUnderline, FontName, FontSize, ForeColor, Height, Hwnd, Left, Name, ParentHwnd, Tag, Top, Timer, Visible, Width, WindowLong

Unterstützte Methoden (siehe "Methoden")

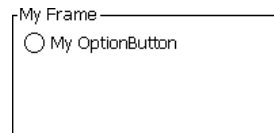
Hide, Move, Show

Unterstützte Ereignisse (siehe "Ereignisse")

Timer

Beispiel

```
REM Frame Beispiel
ADDOBJECT "Frame", "Frame", 10, 10, 100, 100
Frame.Caption = "My Frame"
ADDOBJECT "OptionButton", "Opt", 10, 20, 80, 20
Opt.Caption = "My OptionButton"
Opt.parentHwnd = Frame.Hwnd
Frame.move 100,100 'move frame and contents
```

Ausgabe**Verwandte Begriffe**

ADDOBJECT, Events, Methods, Properties

FUNCTION

Befehl

```
FUNCTION procedurename[(arglist)]  
    [statements]  
    [procedurename = expression]  
[EXIT FUNCTION]  
    [statements]  
    [procedurename = expression]  
END FUNCTION
```

Beschreibung

FUNCTION deklariert eine Prozedur, *procedurename*, die *statements* ausführt, mit *arglist* als Parameter, optional mit einem Rückgabewert. Der erforderliche Parameter, *procedurename*, wird verwendet um die Prozedur aufzurufen und muss den Konventionen für Standard-Variablenamen entsprechen. Die optionale Komponentenliste, *arglist*, ist eine durch Kommata getrennte Liste von Variablen, die die Variablen repräsentiert, die der Prozedur beim Aufruf übergeben werden. Die optionale Komponente, *statements*, wird als Rumpf der Prozedur ausgeführt. Es können beliebig viele optionale EXIT FUNCTION Befehle eingesetzt werden, um die Funktion zu beenden. Der Rückgabewert ist per Vorgabe EMPTY. Um einen anderen, als den Standardwert zurückzugeben, weisen Sie dem Funktionsnamen einen Wert zu. Der aktuelle Wert *procedurename* wird zurückgegeben, wenn die Prozedur normal beendet wird oder einen EXIT FUNCTION Befehl erkennt.

Jeder Eintrag in *arglist* ist ein Argument, das der Prozedur wie im folgenden beschrieben, übergeben wird:

[BYVAL | BYREF] *varname*[()]

Das optionale Schlüsselwort, BYVAL, gibt an, dass der Prozedur eine Kopie der Variablen übergeben wird, wodurch der Inhalt der Ursprungsvariablen außerhalb der Prozedur unverändert bleibt. Das optionale Schlüsselwort, BYREF, spezifiziert, dass der Prozedur die Speicheradresse der Variablen übergeben wird, wodurch der Inhalt der Variablen nach außen hin sichtbar durch die Prozedur verändert werden kann.

varname kann innerhalb der Prozedur als Referenz auf einen übergebenen Wert verwendet werden. Er muss den Standard-Variablenamen entsprechen.

Anmerkung: Wenn der Rückgabewert nicht in einer Variablen gespeichert wird oder Teil eines Ausdrucks ist,

wird sie als SUB Prozedur aufgerufen und mehrere Argumente können nicht in Klammern übergeben werden.

Beispiel

```
REM FUNCTION Beispiel
'FUNCTION: Prozedur die einen Wert uebergibt
DIM Selection, SalePrice
Selection = Menu("Mittwoch")
PRINT " Mittwoch's Menue Tipp:", Selection
SalePrice = Min(31, 29)
PRINT "Preis:", SalePrice
FUNCTION Menu(day)
    IF day = " Mittwoch" THEN
        Menu = "Salisbury Steak"
    END IF
END FUNCTION
FUNCTION Min(x, y)
    IF x > y THEN
        MIN = y
    EXIT FUNCTION
    ELSE
        MIN = x
    END IF
END FUNCTION
```

Ausgabe

```
Mittwoch's Menue Tipp: Salisbury Steak
Preis: 29
```

Verwandte Begriffe

CALL, SUB

GETCOMMANDLINE

Beschreibung

GETCOMMANDLINE wird verwendet um den Kommandozeilentext, der zum Starten des Programms verwendet wurde, zu übergeben. Wurde das Programm für sich selbst aufgerufen, wird der komplette Programmpfad übergeben. Wurde das Programm durch ein Dokument gestartet, das mit dem Programm verknüpft ist, wird der komplette Programmpfad, gefolgt vom kompletten Dokumentenpfad, übergeben.

Beispiel

```
REM GETCOMMANDLINE Beispiel
'GETCOMMANDLINE uebergibt Starttext
cl = GETCOMMANDLINE
prog = LEFT(cl, INSTR(cl, ".nsb") + 4)
doc = MID(cl, LEN(prog) + 1)
```

GETLOCALE Funktion

GETLOCALE

SETLOCALE *localeID*

Beschreibung

GETLOCALE gibt die aktuelle Locale-ID zurück. Eine Locale-ID bestimmt solche Dinge, wie Tastaturlayout, alphabetische Sortierreihenfolge, Datum, Zeitdarstellung und Währungsformat.

SETLOCALE setzt die Lokaleinstellung auf *localeID*. *LocaleID* kann entweder eine Zahl oder ein kurzer String, der die geografische Lokation eindeutig bestimmt, sein. Die Funktion gibt die alte Einstellung der Locale-ID zurück. Mit Setzen der *localeID* auf 0 wird die Locale-ID auf die aktuelle Systemeinstellung eingestellt. Eine komplette Liste der Locale-IDs gibt es hier:

<http://msdn.microsoft.com/library/default.asp?url=/library/en-us/script56/html/882ca1eb-81b6-4a73-839d-154c6440bf70.asp>

Einige gängige Einstellungen sind hier:

1031	de-de	Deutsch – Deutschland
1033	en-us	Englisch – United States
1041	ja	Japanisch
2057	en-uk	Englisch - UK

Nur in Windows CE 4.0 und später.

Beispiel

```
print getLocal
original=SETLOCALE("en-gb") 'Set for England
Print original, getlocale
```

Ausgabe

```
1033
1033 2057
```

Verwandte Begriffe

SETLOCALE, GETLOCALE

GETREF

Funktion

GETREF

Beschreibung

Gibt einen Referenzzeiger auf eine Funktion oder Subroutine zurück. Nur in Windows CE 4.0 und später.

Beispiel

```
Funktion refTest
    MsgBox "RefTest"
End Funktion

Dim x
Set x=getRef("refTest")
Print TypeName(x)
x
```

Ausgabe

```
Object
RefTest
```

VERWANDTE BEGRIFFE

GETSERIALNUMBER

Funktion

GETSERIALNUMBER

Beschreibung

GETSERIALNUMBER gibt die Seriennummer eines Gerätes zurück, sofern sie existiert. Mit Beginn der Pocket PC 2002 Generation gibt es optional Seriennummern und sie sollten in allen späteren Geräten vorhanden sein.

Beispiel

```
REM GETSERIALNUMBER Beispiel  
MsgBox GetSerialNumber()
```

Ausgabe

(hängt vom Gerät ab)

Verwandte Begriffe

HEX

Funktion

HEX(*number*)

Beschreibung

HEX gibt eine Zeichenfolge zurück, die dem Hexadezimalwert (Basis 16) von *number* entspricht. Der erforderliche Parameter, *number*, kann jeder numerische Ausdruck sein. Wenn *number* keine ganze Zahl ist, wird auf die benachbarteste ganze Zahl gerundet, ehe die Umwandlung erfolgt.

Beispiel

```
REM HEX Beispiel
'HEX gibt den Wert einer Zahl als
'Hexadezimal-String zurueck
PRINT "68 in hex:", HEX(68)
PRINT "1 in hex:", HEX(1)
PRINT "2605.45 in hex:", HEX(2605.45)
```

Ausgabe

```
68 in hex:      44
1 in hex:       1
2605.45 in hex:      A2D
```

Verwandte Begriffe

OCT

HOUR

Funktion

HOUR(*time*)

Beschreibung

HOUR gibt eine ganze Zahl im Bereich von 0 bis 23 zurück, die der Stundenangabe einer vorgegebenen Uhrzeit entspricht. Der erforderliche Parameter, *time*, kann jeder numerische Ausdruck oder String, oder jeder andere Ausdruck sein, sofern er eine Zeit darstellt.

Beispiel

```
REM HOUR Beispiel
' HOUR gibt Stundenangabe der Zeit zurueck
PRINT "Die Stunde von " &
FormatDateTime(Now,4) & _"ist " & HOUR(Now)
```

Ausgabe

Die Stunde von 8/18/1998 10:52:44 PM ist 22

(Beispiel-Ausgabe ist systemabhängig)

Verwandte Begriffe

DAY, MINUTE, MONTH, NOW, SECOND, TIME, YEAR

IF...THEN...ELSE **Befehl**

IF *condition* THEN *statements* [ELSE *elstatements*]

IF *condition* THEN

statements

[ELSEIF *condition-n* THEN

elseifstatements]]...

[ELSE

elstatements]]

END IF

Beschreibung

IF...THEN...ELSE wird verwendet um Gruppen von Befehlen in Abhängigkeit bestimmter Konditionen auszuführen. Die erforderliche Komponente, *condition*, kann jeder Ausdruck sein, der TRUE oder FALSE als Ergebnis liefert. Wird der Befehl innerhalb einer Zeile ohne Else-Klausel benutzt, ist die Komponente *statements* erforderlich, ansonsten ist *statements* optional. Wenn *condition* TRUE oder nicht Null ergibt, werden alle existierenden *statements* ausgeführt. Ergibt *condition* FALSE oder Null, verzweigt die Ausführung zur nächsten existierenden ELSEIF Klausel um *condition-n* auszuwerten oder zur ELSE Klausel, falls vorhanden.

Um mehrere Befehle innerhalb einer Zeile auszuführen, müssen diese durch Doppelpunkt (:) getrennt werden. Wenn ein einzeliger Befehl nur aus einem Prozeduraufruf ohne Argumente besteht, muss die Prozedur mit leeren Klammern aufgerufen werden.

Beispiel

```
REM IF...THEN...ELSE Beispiel
'IF...THEN...ELSE bedingungsabhaengige
'Ausfuehrung
DIM Who
IF TRUE THEN PRINT "TRUE" ELSE PRINT "FALSE"
IF Who = "Al" THEN
    PRINT "Big Al"
ELSEIF Who = "Alien" THEN
    PRINT "Alien Probe"
END IF
```

Ausgabe

TRUE

Verwandte Begriffe

SELECT CASE

IMP**Funktion**

result = x IMP y

Beschreibung

IMP gibt die logische Implikation zweier Ausdrücke zurück. Die logische Implikation ergibt TRUE wenn x eine hinreichende Bedingung für y ist. IMP führt auch einen bitweisen Vergleich zweier numerischer Ausdrücke durch.

Tabelle 13: Logische Implikation

x	y	Implikation
TRUE/1	TRUE/1	TRUE/1
TRUE/1	FALSE/0	FALSE/0
FALSE/0	TRUE/1	TRUE/1
FALSE/0	FALSE/0	TRUE/1

Beispiel

```
REM IMP Beispiel
'IMP fuer logische und bitweise Implikation
DIM Test1, Test2, Test3, x, y
x = 3
y = 5
Test1 = x < 0 IMP y < 10
Test2 = x > 0 IMP y > 10
Test3 = x < 0 IMP y > 10
PRINT "Logisch:"
PRINT "  x < 0 IMP y < 10 = " & Test1
PRINT "  x > 0 IMP y > 10 = " & Test2
PRINT "  x < 0 IMP y > 10 = " & Test3
PRINT "Bitweise:"
PRINT "  x IMP y = " & (x IMP y)
```

Ausgabe

```
Logisch:
  x < 0 IMP y < 10 = True
  x > 0 IMP y > 10 = False
  x < 0 IMP y > 10 = True
Bitweise:
  x IMP y = -3
```

Verwandte Begriffe

AND, EQV, NOT, OR, XOR

INPUTBOX

Funktion

INPUTBOX(*prompt*[, *title*[, *default*[, *xpos*, *ypos*]]])

Beschreibung

INPUTBOX öffnet eine Dialogbox in der der Anwender aufgefordert wird, Text einzugeben oder eine Taste zu drücken. Der Rückgabewert ist ein String mit dem Inhalt des Dialogbox-Textfeldes. Der erforderliche Parameter, *prompt*, ist ein String-Ausdruck, der im Rumpf der Dialogbox angezeigt wird. Der optionale Parameter, *title*, ist ein String-Ausdruck, der in der Titelzeile angezeigt wird. Der optionale Parameter, *default*, ist ein String-Ausdruck, der im Textfeld der Dialogbox angezeigt wird. Die optionalen Parameter, *xpos* und *ypos*, sind numerische Ausdrücke, die den horizontalen und vertikalen Abstand zwischen linker oberer Ecke des Bildschirms und linker oberer Ecke der Dialogbox angeben.

Wenn der Anwender auf OK tippt oder die Enter Taste drückt, gibt INPUTBOX den Inhalt des Textfeldes oder, bei fehlender Eingabe, einen leeren String ("") zurück. Betätigt der Anwender CANCEL oder drückt die Escape Taste (Esc), gibt INPUTBOX EMPTY zurück.

INPUTBOX funktioniert nicht mit Pocket PC Geräten. Schauen Sie in der ReadMe-Datei nach, wie Sie dieses Problem umgehen können.

Beispiel

```
DIM Return
Return = INPUTBOX("Nachrichtenbereich", _
    "INPUTBOX Beispiel", "Vorgabetext")
PRINT "Sie gaben ein:", Return
```

Ausgabe

Sie gaben ein: Vorgabetext

Verwandte Begriffe

INSTR([*start*,]*string1*, *string2*[, *compare*])

INSTRREV(*string1*, *string2*[, *start*[, *compare*]])

Beschreibung

INSTR gibt eine lange ganze Zahl zurück, die angibt, an welcher Position sich ein String innerhalb eines anderen befindet, gemessen vom Beginn des zu durchsuchenden Strings.

INSTRREV gibt eine lange ganze Zahl zurück, die angibt, an welcher Position sich ein String innerhalb eines anderen befindet, gemessen vom Ende des zu durchsuchenden Strings.

Der optionale Parameter, *start*, ist ein numerischer Ausdruck, der die Startposition der Suche innerhalb des Ziel-Strings angibt. Wenn *start* nicht angegeben wird, beginnt die Suche für INSTR bei 0, der Position des ersten Zeichens, oder für INSTRREV bei -1, der Position des letzten Zeichens. Der erforderliche Parameter, *string1*, ist die zu durchsuchende Zeichenfolge. Der erforderliche Parameter, *string2*, ist der String, nach dem gesucht wird. Der optionale Parameter, *compare*, wird verwendet um die Art der Suche zu spezifizieren. Wenn *string2* in *string1* gefunden wurde, wird eine positive ganze Zahl zurückgegeben, die der Position des ersten Zeichens entspricht, andernfalls wird 0 zurückgegeben.

Beispiel

```
REM INSTR/INSTRREV Beispiel
'INSTR/INSTRREV findet einen String in
'einem anderen
DIM Pos1, Pos2
Pos1 = INSTR("Cartman", "man")
Pos2 = INSTRREV("Big Al's Big Boat Ride", _
    "big", 4, vbTextCompare)
PRINT "Gefunden \"man\" vom Start:", Pos1
PRINT "Gefunden \"big\" vom Ende:", Pos2
```

Ausgabe

```
Gefunden "man" vom Start:      5
Gefunden "big" vom Ende:      1
```

Verwandte Begriffe

INT(*number*)

Beschreibung

INT entfernt die Nachkommastellen einer Zahl und gibt die nächst kleinere ganze Zahl zurück. Der erforderliche Parameter, *number*, kann jeder numerische Ausdruck sein.

Beispiel

```
REM INT Beispiel
'INT entfernt Nachkommastellen
DIM Pos, Neg
Pos = ATN(1) * 4
Neg = -Pos
PRINT "INT(pi) = " & INT(Pos)
PRINT "INT(-pi) = " & INT(Neg)
```

Ausgabe

```
INT(pi) = 3
INT(-pi) = -4
```

Verwandte Begriffe

FIX

result = *Objekt1* IS *Objekt2*

Beschreibung

IS gibt einen booleschen Wert zurück, der angibt, ob eine Objekt-Referenz identisch mit einer anderen ist. Die erforderlichen Komponenten, *Objekt1* und *Objekt2* sind zwei Objekt-Referenzen.

Beispiel

```
REM IS Beispiel
DIM Obj2
ADDOBJECT "Finance", "Obj1"
SET Obj2 = Obj1
CompareObjekts Obj1, Obj2
SET Obj2 = NOTHING
CompareObjekts Obj1, Obj2
SUB CompareObjekts(Obj1, Obj2)
  IF Obj1 IS Obj2 THEN
    PRINT "Gleich"
  ELSE
    PRINT "Unterschiedlich"
  END IF
END SUB
```

Ausgabe

Gleich
Unterschiedlich

Verwandte Begriffe

ADDOBJECT, SET

ISARRAY(*expression*)

ISDATE(*expression*)

ISEMPTY(*expression*)

ISNULL(*expression*)

ISNUMERIC(*expression*)

ISOBJEKT(*expression*)

Beschreibung

Die IS Funktionen geben TRUE zurück, wenn der Variablentyp zum Funktionsaufruf passt, ansonsten wird FALSE zurückgegeben. Der erforderliche Parameter, *expression*, ist die Variable deren Typ festgestellt werden soll.

Beispiel

```
REM IS Funktionsbeispiel
DIM Children(3), Chef, When
TestVariable Children
Chef = 1
TestVariable Chef
When = NOW
TestVariable When
SUB TestVariable(var)
  IF ISARRAY(var) THEN
    PRINT "Die Variable ist ein Array."
  ELSEIF ISDATE(var) THEN
    PRINT " Die Variable ist ein Datum."
  ELSEIF ISNUMERIC(var) THEN
    PRINT " Die Variable ist eine Nummer."
  END IF
END SUB
```

Ausgabe

```
Die Variable ist ein Array.
Die Variable ist eine Nummer.
Die Variable ist ein Datum.
```

Verwandte Begriffe

TYPENAME, VARTYPE

JOIN(*stringarray*[, *delimiter*])

Beschreibung

JOIN gibt eine Zeichenkette zurück, die durch Zusammenfügen einer Liste von Strings, optional mit einem Abgrenzungszeichen, erzeugt wurde. Das erforderliche Argument, *stringarray*, ist ein ein-dimensionales String-Array. Das optionale Argument, *delimiter*, ist ein String-Ausdruck dessen erstes Zeichen als Abgrenzungszeichen zwischen den Strings des *stringarray* verwendet wird; wenn kein String angegeben ist, wird ein Leerzeichen (" ") als Standardwert verwendet. Um ohne Abgrenzungszeichen zusammenzufügen, verwenden Sie einen leeren String ("") als *delimiter*.

Beispiel

```
REM JOIN Beispiel
'JOIN fuegt Strings zusammen
DIM Words, Letters
Words = ARRAY("Hallo", "Welt")
Letters = ARRAY("D", "a", "l", "l", "a", "s")
PRINT JOIN(Words) & "!"
PRINT JOIN(Letters, "")
```

Ausgabe

```
Hallo Welt!
Dallas
```

Verwandte Begriffe

SPLIT

KeyPreview = TRUE | FALSE

Beschreibung

KeyPreview ist eine globale Eigenschaft die es dem Ausgabe-Objekt ermöglicht, alle Tastatur-Ereignisse zu empfangen. KeyPreview ist standardmäßig auf FALSE gesetzt, was das Ausgabe-Objekt davor schützt, Tastatur-Ereignisse zu empfangen. Wird KeyPreview auf TRUE gesetzt, kann das Ausgabe-Objekt alle Tastatur-Ereignisse empfangen, unabhängig davon, welches Objekt gerade für Tastatur-Ereignisse aktiviert ist.

Beispiel

```
REM KeyPreview Beispiel
'KeyPreview ermoeeglicht Tastatur-Ereignisse
'in der Ausgabe
KeyPreview = TRUE
ADDOBJECT "TextBox", "Text", 50, 50, 90, 90
SUB Ausgabe_KeyPress(key)
    PRINT "Taste gedrueckt: ", key
END SUB
```

Ausgabe

Verwandte Begriffe

KeyDown, KeyPress, KeyUp, PictureBox, SetFocus, Ausgabe

KeyboardStatus

Globale Variable

KeyboardStatus=0|-1

KeyboardStatusChanged

Beschreibung

KeyboardStatus ist eine globale Variable, die festlegt und anzeigt, ob die Bildschirmtastatur eines Pocket PC Gerätes angezeigt wird oder nicht. Wenn sie 0 ist, wird die Tastatur nicht angezeigt. Der Wert -1 bedeutet, dass die Tastatur eingeblendet ist.

Wenn der Status der Bildschirmtastatur geändert wird, wird das KeyboardStatusChanged Ereignis an des Programm geschickt.

Beispiel

```
REM KeyboardStatus Beispiel
PRINT KeyboardStatus
IF KeyBoardStatus=0 THEN KeyBoardStatus=-1_
ELSE KeyBoardStatus=-1
SUB keyboardStatusChanged
    MSGBOX "Keyboard Status wurde geaendert"
END SUB
```

Ausgabe

(Bildschirmtastatur erscheint oder verschwindet am Bildschirm)

Verwandte Begriffe

KILLFOCUS**Befehl**

KillFocus

Beschreibung

KillFocus nimmt den Fokus vom aktuellen Objekt.

Beispiel

```
REM KillFocus Beispiel
ADDOBJECT "TextBox","TB",10,10,20,20
' Fokus auf TextBox setzen
tb.SetFocus
' Fokus von Textbox nehmen
KillFocus
```

Ausgabe

(Der Cursor wird in der TextBox platziert und verschwindet wieder)

Verwandte Begriffe

Label**Objekt**

ADDOBJECT "Label", *name*, *xpos*, *ypos*, *width*, *height*

Beschreibung

Label wird verwendet, um unveränderliche Texte als Objekt beliebig auf dem Bildschirm positionieren zu können. Die erforderliche Komponente, *name*, ist der Name der Variablen, die dem Programm als Referenz auf dieses Objekt hinzugefügt wird. Die optionale Komponenten, *xpos*, *ypos*, *width*, und *height* sind numerische Ausdrücke, die Position und Größe des Objekts in Pixel angeben, gemessen von der oberen linken Ecke.

Anmerkung: Als Vorgabe ist die TabStop-Eigenschaft für dieses Objekt FALSE, um Anwenderaktionen auf Label zu unterbinden.

Unterstützte Eigenschaften (siehe "Eigenschaften")

Alignment, BackColor, BorderStyle, Caption, Enabled, FontBold, FontItalic, FontName, FontSize, FontStrikethru, FontUnderline, ForeColor, Height, Hwnd, Left, Name, ParentHwnd, TabStop, Tag, Text, Timer, Top, UseMnemonic, Visible, Width, WindowLong

Unterstützte Methoden (siehe "Methoden")

Hide, Move, SetFocus, Show

Unterstützte Ereignisse (siehe "Ereignisse")

Click, Timer

Beispiel

```
REM Label Beispiel
ADDOBJECT "Label", "Label", 10, 10, 40, 20
Label.BackColor = Ausgabe.BackColor
ADDOBJECT "TextBox", "Input", 55, 10, 160, 20
Input.text = ""
Input.SetFocus
```

Ausgabe

Verwandte Begriffe

ADDOBJECT, Events, Methods, Properties, TextBox

LBOUND

Funktion

LBOUND(*array* [, *dimension*])

Beschreibung

LBOUND gibt eine Longintegerzahl zurück, die den kleinsten möglichen Index einer vorgegebenen Dimension eines Arrays darstellt. Der erforderliche Parameter, *array*, kann jede Array-Variable sein. Der optionale Parameter, *dimension*, gibt an, aus welcher Dimension die Obergrenze ermittelt werden soll, beginnend mit 1 als Vorgabe. Die untere Grenze jeder Dimension eines Arrays ist immer 0.

Beispiel

```
REM LBOUND Beispiel
'LBOUND gibt Untergrenze der Dimension eines
'Arrays zurueck
DIM Andere, Kinder(3), Eltern(3, 1)
Andere = ARRAY("Damien", "Pip", "Wendy")
PRINT "'Andere' Untergrenze:", _
LBOUND(Andere)
PRINT "'Kinder' Untergrenze:", _
LBOUND(Kinder, 1)
PRINT "'Eltern' Untergrenzen:", _
LBOUND(Eltern), LBOUND(Eltern, 2)
```

Ausgabe

```
'Andere' Untergrenze: 0
'Kinder' Untergrenze: 0
'Eltern' Untergrenzen: 0 0
```

Verwandte Begriffe

ARRAY, DIM, REDIM, UBOUND

LCASE

Funktion

LCASE(*string*)

Beschreibung

LCASE gibt *string* mit allen Zeichen in Kleinschreibung zurück. Der erforderliche Parameter, *string*, kann jeder gültige String-Ausdruck sein.

Beispiel

```
REM LCASE Beispiel
'LCASE gibt String in Kleinbuchstaben aus
DIM Uncle
Uncle = "JIM"
PRINT Uncle & " in Kleinbuchstaben ist " &
LCASE(Uncle)
```

Ausgabe

JIM in Kleinbuchstaben ist jim

Verwandte Begriffe

UCASE

LEFT

Funktion

LEFT(*string*, *length*)

LEFTB(*string*, *length*)

Beschreibung

LEFT gibt einen String zurück, der die angegebene Anzahl von Zeichen vom linken Ende einer Zeichenkette darstellt. Der erforderliche Parameter, *string*, kann jeder gültige String-Ausdruck sein. Der erforderliche Parameter, *length*, kann jeder gültige numerische Ausdruck sein. Wenn *length* 0 ist, wird ein leerer String ("") zurück gegeben. Ist *length* größer als die Länge des Strings, wird der gesamte String zurück gegeben.

LEFTB ist eine strikt byte-genaue Version von LEFT. Wenn sie mit Unicode Strings auf Windows CE Geräte eingesetzt wird, gibt sie jeden einzelnen Charakter als normalen Charakter zurück.

Beispiel

```
REM LEFT Beispiel
'LEFT gibt Teilstrings von links zurueck
DIM Wendy, Eric
Wendy = "Testaburger"
Eric = "Cartman"
PRINT "Das LEFT 4 von " & Wendy & ": " _
      & LEFT(Wendy, 4)
PRINT "Das LEFT 4 von " & Eric & ": " _
      & LEFT(Eric, 4)
```

Ausgabe

```
Das LEFT 4 von Testaburger: Test
Das LEFT 4 von Cartman: Cart
```

Verwandte Begriffe

LEN, MID, RIGHT

LEN(*string* | *variable*)

LENB(*string* | *variable*)

Beschreibung

LEN gibt eine lange ganze Zahl zurück, die der Anzahl der Zeichen in einem String oder der Anzahl der erforderlichen Bytes zur Anzeige einer Variablen entspricht. Der optionale Parameter, *string*, kann jeder gültige String-Ausdruck sein. Der optionale Parameter, *variable*, kann jede Variable sein. Wenn *variable* einen String enthält, wird dessen Länge zurück gegeben. Einer (und nur einer) der optionalen Parameter, *string* und *variable*, muss angegeben sein.

LENB ist die strikt byte-genaue Version von LEN. LENB gibt immer die Anzahl der Bytes zurück. Wird sie mit Unicode Strings auf einem Windows CE Gerät verwendet, entspricht die Anzahl der doppelten Anzahl die von LEN zurück gegeben wird.

Beispiel

```
REM LEN Beispiel
'LEN gibt Stringlaenge oder Variablengroesse
'zurueck
DIM Frog, Survived
Frog = "Staring"
Survived = 2
PRINT "LEN von Frog:", LEN(Frog)
PRINT "LENB von Frog:", LENB(Frog)
PRINT "LEN von Survived:", LEN(Survived)
```

Ausgabe

```
LEN of Frog:      7
LENB of Frog:    14
LEN of Survived:      1
```

Verwandte Begriffe

ADDOBJECT "ListBox", *name*, *xpos*, *ypos*, *width*, *height*

Beschreibung

ListBox wird verwendet, um eine Textliste anzuzeigen, die frei auf dem Bildschirm positioniert werden kann. Die erforderliche Komponente, *name*, ist der Name der Variablen, unter der dieses Objekt vom Programm angesprochen werden kann. Sie muss den Konventionen für Standard-Variablennamen genügen. Die erforderlichen Komponenten, *xpos*, *ypos*, *width*, und *height* sind numerische Ausdrücke, die Größe und Position des Objektes, gemessen von der linken oberen Ecke aus, angeben. Die ListIndex Eigenschaft ist der Index des ausgewählten Listenelements, beginnend mit der 0 für das erste Element der Liste; wurde kein Element ausgewählt, steht die ListIndex Eigenschaft auf -1. MultiSelect muss auf TRUE gesetzt sein, bevor der ListBox Elemente hinzugefügt werden können.

Unterstützte Eigenschaften (siehe "Eigenschaften")

BackColor, Enabled, FontBold, FontItalic, FontName, FontSize, FontStrikethru, FontUnderline, ForeColor, Height, Hwnd, IntegralHeight, Left, List (read only), ListCount (read-only), ListIndex, MultiSelect, Name, NewIndex, ParentHwnd, Redraw, ScrollBars, SelCount, Selected, Sorted, Tag, TabStop, Text, Timer, Top, TopIndex, Visible, Width, WindowLong

Unterstützte Methoden (siehe "Methoden")

AddItem, Clear, Hide, Move, RemoveItem, SetFocus, Show

Unterstützte Ereignisse (siehe "Ereignisse")

Change, Click, DblClick, GotFocus, KeyDown, KeyPress, KeyUp, LostFocus, Timer

Beispiel

```
REM ListBox Beispiel
'ListBox ist ein Text-Listen Objekt
DIM i
ADDOBJECT "ListBox", "List", 10, 10, 100, 100
List.ScrollBars = 2
FOR i = vbSUNDAY TO vbSATURDAY
    List.AddItem WEEKDAYNAME(i)
NEXT
List.ListIndex = WEEKDAY(NOW-1)
List.SetFocus
```

Ausgabe

Verwandte Begriffe

ADDOBJECT, ComboBox, Events, Methods, Properties

LOG(*number*)

Beschreibung

LOG gibt eine Zahl doppelter Genauigkeit zurück, die dem natürlichen Logarithmus von *number* entspricht. Der erforderliche Parameter, *number*, kann jeder numerische Ausdruck sein. Der natürliche Logarithmus hat zur Basis die Zahl e; e ist annähernd 2.718282.

Den Logarithmus zur Basis n einer Zahl x erhält man, wenn man den natürlichen Logarithmus der Zahl x durch den natürlichen Logarithmus der Zahl n teilt.

Beispiel

```
REM LOG Beispiel
'LOG kalkuliert natuerlichen Logarithmus
DIM e
e = 2.718282
PRINT "LOG(1) = " & LOG(1)
PRINT "LOG(e) = " & LOG(e)
PRINT "LOG10(2) = " & LogN(10, 2)
FUNKTION LogN(Base, Number)
    LogN = LOG(Number) / LOG(Base)
END FUNKTION
```

Ausgabe

```
LOG(1) = 0
LOG(e) = 1
LOG10(2) = 0.30103
```

Verwandte Begriffe

EXP

LTRIM

Funktion

LTRIM(*string*)

Beschreibung

LTRIM gibt eine Zeichenkette zurück, bei der alle Leerzeichen am Anfang entfernt wurden. Der erforderliche Parameter, *string*, kann jeder gültige Stringausdruck sein.

Beispiel

```
REM LTRIM Beispiel
'LTRIM trennt Leerzeichen am Anfang ab
DIM Spacey
Spacey = " K"
PRINT "(" & Spacey & ")"
PRINT "(" & LTRIM(Spacey) & ")"
```

Ausgabe

```
( K)
(K)
```

Verwandte Begriffe

RTRIM, TRIM

ObjectName.Method [arglist]

Beschreibung

Objektmethoden sind Prozeduren in Objekten. Die erforderliche Komponente, *ObjectName*, ist ein gültiger Objekt-Ausdruck, der auf ein Objekt verweist, das dem Programm per ADDOBJECT Befehl hinzugefügt wurde. Die erforderliche Komponente, *Method*, ist der Name einer FUNCTION oder SUB Prozedur eines Objekts (siehe die Liste der üblichen Objekt-Methoden weiter unten). Die optionale Komponente, *arglist*, ist eine durch Kommata getrennte Liste von Werten, die der Prozedur als Argumente übergeben werden. Der Aufruf einer Methode eines Objektes kann das Aussehen, das Verhalten und die Eigenschaften eines Objektes verändern.

Tabelle 14: Objektmethoden

Methode	Argumente	Kommentare
AddItem	text[, index]	Fügt <i>text</i> in die Liste der Elemente ein, optional mit einem <i>index</i> . Wird kein <i>index</i> mitgeliefert, wird <i>text</i> an das Ende einer unsortierten Liste gesetzt, oder <i>text</i> wird an die richtige Stelle einer sortierten Liste gesetzt. ComboBox, ListBox
Clear		Löscht alle Elemente aus der Liste. ComboBox, ListBox
Hide		Objekt verbergen und setzt die Eigenschaft <i>Visible</i> auf FALSE.
Move	<i>x, y, w, h</i>	Bewegt das Objekt zu neuen <i>x, y</i> Positionen, optional mit Größenänderung, wenn <i>w</i> und <i>h</i> angegeben sind.
RemoveItem	index	Entferne das durch <i>index</i> angegebene Element aus der Liste (0 = erstes Element). ComboBox, ListBox
SetFocus		Objekt für Tastatureingaben aktivieren.

Show		Objekt anzeigen und Eigenschaft Visible auf TRUE setzen.
------	--	--

Beispiel

```
REM Objekt Methoden Beispiel
'Methoden sind Prozeduren in Objekten
ADDOBJECT "ComboBox", "Combo", 5, 5, 150, 110
Combo.AddItem("Eric")
Combo.AddItem("Kenny")
Combo.AddItem("Kyle")
Combo.AddItem("Stan")
Combo.ListIndex = 0
Combo.SetFocus
```

Ausgabe

Verwandte Begriffe

Events, Properties

MID(*string*, *start*[, *length*])

MIDB(*string*, *start*[, *length*])

Beschreibung

MID gibt einen String aus der Mitte eines vorgegebenen Strings zurück. Der erforderliche Parameter, *string*, kann jeder gültige String-Ausdruck sein. Der erforderliche Parameter, *start*, kann jeder numerische Ausdruck sein. Ist er größer als die Länge der Zeichenkette, wird ein leerer String ("") zurück gegeben. Der optionale Parameter, *length*, kann jeder gültige numerische Ausdruck sein. Er gibt der Anzahl der Zeichen an, die zurück gegeben werden sollen. Wenn *length* nicht verwendet wird oder wenn er die Anzahl der übrig bleibenden Zeichen überschreitet, werden alle Zeichen von *start* bis Ende zurück gegeben.

MIDB ist eine strikt byte-genaue Version von MID. Wenn sie mit Unicode Strings auf Windows CE Geräte eingesetzt wird, gibt sie jeden einzelnen Charakter als normalen Charakter zurück.

Beispiel

```
REM MID Beispiel
'MID gibt Substring aus der Stringmitte aus
DIM Eric, Mister
Eric = "Cartman"
PRINT "Von Cartman:", MID(Eric, 2, 3)
Mister = "Hankey"
PRINT "Von Hankey:", MID(Mister, 4)
```

Ausgabe

```
Von Cartman: art
Von Hankey:   key
```

Verwandte Begriffe

LEFT, RIGHT

MINUTE

Funktion

MINUTE(*time*)

Beschreibung

MINUTE gibt eine ganze Zahl im Bereich von 0 bis 59 zurück, der die Minute in der Stunde für einen vorgegebenen Zeitpunkt repräsentiert. Der erforderliche Parameter, *time*, kann jeder Ausdruck sein, der einen Zeitpunkt darstellt.

Beispiel

```
REM MINUTE Beispiel
'MINUTE gibt die Minute in der Zeit an
DIM When
When = NOW
PRINT "Die MINUTE von " & When & " ist " _
      & MINUTE(When)
```

Ausgabe

Die MINUTE von 8/18/1998 10:52:44 PM ist 52

(Beispiel-Ausgabe ist systemabhängig)

Verwandte Begriffe

DAY, HOUR, MONTH, NOW, SECOND, TIME, YEAR

result = x MOD y

Beschreibung

MOD teilt x durch y und gibt den ganzzahligen Rest zurück. Die erforderlichen Parameter, x und y, sind numerische Ausdrücke. Der Wert von *result* ist eine ganze Zahl die kleiner als y ist.

Beispiel

```
REM MOD Beispiel
'MOD gibt Divisionsrest zurück
DIM Answer
Answer = 15 MOD 2
PRINT "15 MOD 2 = " & Answer
Answer = 21 MOD 3.7
PRINT "21 MOD 3.7 = " & Answer
```

Ausgabe

```
15 MOD 2 = 1
21 MOD 3.7 = 1
```

Verwandte Begriffe

MONTH**Funktion**

MONTH(*date*)

Beschreibung

MONTH gibt eine ganze Zahl im Bereich von 1 bis 12 zurück, die den Monat im Jahr für ein vorgegebenes Datum darstellt. Der erforderliche Parameter, *date*, kann jeder numerische oder String-Ausdruck oder jeder andere Ausdruck sein, der ein Datum repräsentiert.

Beispiel

```
REM MONTH Beispiel
'MONTH gibt Monat im Jahr zurueck
DIM When
When = NOW
PRINT "Der MONTH von " & When & " ist " _
      & MONTH(When)
```

Ausgabe

Der MONTH von 8/18/1998 10:52:44 PM ist 8

(Beispiel-Ausgabe ist systemabhängig)

Verwandte Begriffe

DAY, HOUR, MINUTE, NOW, SECOND, TIME, YEAR

MONTHNAME

Funktion

MONTHNAME(*month*[, *abbreviate*])

Beschreibung

MONTHNAME gibt den Namen des vorgegebenen Monats als String zurück. Der erforderliche Parameter, *month*, kann jeder numerische Ausdruck im Bereich von 1 bis 12 sein. Der optionale Parameter, *abbreviate*, ist ein boolescher Wert, der bestimmt, ob MONTHNAME den kompletten Namen oder eine Abkürzung mit 3 Zeichen zurück gibt.

Beispiel

```
REM MONTHNAME Beispiel
PRINT MONTHNAME (MONTH (NOW) )
PRINT MONTHNAME (12, TRUE)
```

Ausgabe

```
August
Dec
```

Verwandte Begriffe

WEEKDAYNAME

MSGBOX(prompt[, buttons[, title]])

Beschreibung

MSGBOX öffnet eine Dialogbox und wartet, dass der Anwender eine Taste betätigt. Der Rückgabewert ist eine ganze Zahl, die anzeigt, welche Taste gedrückt wurde. Der erforderliche Parameter, *prompt*, ist ein String-Ausdruck, der im Rumpf der Dialogbox angezeigt wird. Der optionale Parameter, *buttons*, ist ein numerischer Ausdruck oder eine Konstante, der festlegt, welche Taste angezeigt wird, welches Icon verwendet wird, welche Taste voreingestellt wird und in welchem Modus die Dialog betrieben wird. Die Voreinstellung für *buttons* ist 0. Andere Werte werden durch Addieren der Konstanten aus der unten dargestellten Tabelle gebildet. Der optionale Parameter, *title*, ist die Zeichenkette, die in der Titelseite der Dialogbox erscheint.

Tabelle 15: Button Konstanten

Konstante	Wert	Beschreibung
vbOKOnly	0	Nur OK Button
vbOKCancel	1	OK und Cancel Buttons
vbAbortRetryIgnore	2	Abort, Retry und Ignore Buttons
vbYesNoCancel	3	Yes, No und Cancel Buttons
vbYesNo	4	Yes und No Buttons
vbRetrvCancel	5	Retrv und Cancel Buttons
vbCritical	16	Critical Message Icon
vbQuestion	32	Warning Query Icon
vbExclamation	48	Warning Message Icon
vbInformation	64	Information Message Icon
vbDefaultButton1	0	Erster Button ist Default
vbDefaultButton2	256	Zweiter Button ist Default
vbDefaultButton3	512	Dritter Button ist Default
vbDefaultButton4	768	Vierter Button ist Default
vbApplicationModal	0	Applikations-Schnittstelle ist gesperrt, bis der User die Eingabe macht
vbSystemModal	4096	Applikations- und System-Schnittstelle sind gesperrt, bis der User die Eingabe macht

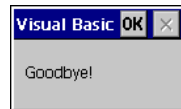
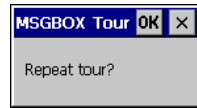
Tabelle 16: MSGBOX Rückgabewerte

Konstante	Wert	Beschreibung
vbOK	1	OK
vbCancel	2	Cancel
vbAbort	3	Abort
vbRetrv	4	Retrv
vbIgnore	5	Ignore
vbYes	6	Yes
vbNo	7	No

Beispiel

```
REM MSGBOX Beispiel
'MSGBOX gibt modale DialogBox aus
CONST TITLE = "MSGBOX Tour"
DIM Continue
MSGBOX("Hello World!")
MSGBOX "Brief tour of MSGBOX", 0, TITLE
Continue = MSGBOX("Continue tour?", _
    vbYesNo + vbQuestion, TITLE)
IF Continue = vbYes THEN
    Continue=MSGBOX("Short tour, huh?", _
        vbInformation + vbYesNo, TITLE)
END IF
Continue = MSGBOX("Repeat tour?", 257, TITLE)
IF Continue = vbCancel THEN
    MSGBOX("Goodbye!")
END IF
```

Ausgabe



Verwandte Begriffe
INPUTBOX

result = NOT *expression*

Beschreibung

NOT gibt die logische Negation eines Ausdrucks zurück. Das erforderliche Argument, *expression*, kann jeder gültige Ausdruck sein. *result* ist TRUE, wenn *expression* FALSE ist und FALSE, wenn *expression* TRUE ist, *result* ist ansonsten NULL.

NOT führt auch eine bitweise Inversion eines Ausdrucks durch. Jedes Bit in *result* wird auf 1 gesetzt, wenn das korrespondierende Bit in *expression* 0 ist, ansonsten wird es auf 0 gesetzt.

Beispiel

```
REM NOT Beispiel
'NOT fuehrt logische Negation & bitweise
'Inversion durch
DIM Test1, Test2, Test3, x, y
x = 3
y = 8
Test1 = NOT(x > 0)
Test2 = NOT(y > 10)
Test3 = NOT x
PRINT "Logisch:"
PRINT "  NOT(x > 0) = " & Test1
PRINT "  NOT(y > 10) = " & Test2
PRINT "Bitweise:"
PRINT "  NOT x = " & Test3
```

Ausgabe

```
Logisch:
  NOT(x > 0) = False
  NOT(y > 10) = True
Bitweise:
  NOT x = -4
```

Verwandte Begriffe

AND, EQV, IMP, OR, XOR

NOW**Funktion**

NOW

Beschreibung

NOW gibt das aktuelle Datum und die aktuelle Zeit entsprechend den Vorgaben Ihres Systems zurück.

Beispiel

```
REM NOW Beispiel  
PRINT FormatDateTime (Now)
```

Ausgabe

11/11/2004 9:52:27 AM

Verwandte Begriffe

Date, Day, Hour, Minute, Month, Second, Time, Weekday, Year

NSBVERSION**Globale Eigenschaft**

NSBVERSION

Beschreibung

NSBVersion gibt einen String mit der aktuell ausgeführten Version von NS Basic/CE aus. Diese globale Eigenschaft kann nur gelesen, nicht gesetzt werden.

Beispiel

```
REM NSBVersion Beispiel
PRINT "Die aktuelle Version von NS Basic ist
" & NSBVersion
```

Ausgabe

Die aktuelle Version von NS Basic ist v.
4.0.1

Verwandte Begriffe

OCT

Funktion

OCT(*number*)

Beschreibung

OCT gibt eine Zeichenfolge zurück, die dem Oktalwert (Basis 8) von *number* entspricht. Der erforderliche Parameter, *number*, kann jeder numerische Ausdruck sein. Wenn *number* keine ganze Zahl ist, wird auf die benachbarteste ganze Zahl gerundet, ehe die Umwandlung erfolgt.

Beispiel

```
REM OCT Beispiel
'OCT gibt eine Nummer als Oktal-String aus
PRINT "68 in oktal:", OCT(68)
PRINT "1 in oktal:", OCT(1)
PRINT "2605.45 in oktal:", OCT(2605.45)
```

Ausgabe

```
68 in oktal:   104
1 in oktal:    1
2605.45 in oktal: 5055
```

Verwandte Begriffe

HEX

ON ERROR**Befehl**

ON ERROR RESUME NEXT

ON ERROR GOTO 0

Beschreibung

ON ERROR wird eingesetzt, um Fehlerbehandlungen in Prozeduren durch Abfangen und Bearbeiten von Laufzeitfehlern zu erreichen, ohne das Programm dabei abubrechen. In der ersten Form, RESUME NEXT, wird die Programmausführung mit dem Befehl fortgesetzt, der dem Befehl folgt, der den Fehler ausgelöst hat. Die zweite Form, GOTO 0, wird verwendet um Fehlerbehandlungen in der aktuellen Prozedur zu unterbinden. Wenn keine ON ERROR Befehle eingesetzt werden, führen alle Laufzeitfehler zu fatalen Fehlern, die angezeigt werden und zum Programmabbruch führen.

Beispiel

```
REM ON ERROR Beispiel
'ON ERROR Fehlerbehandlung in Prozeduren
ADDOBJECT "File"
GetToDoList
SUB GetToDoList
    ON ERROR RESUME NEXT
    File.OPEN "ToDo", 1
    IF ERR.NUMBER THEN
        PRINT "Verzoegerte ERROR Behandlung"
        PRINT "ERROR Quelle: " & ERR.Source
    END IF
    ON ERROR GOTO 0
END SUB
```

Ausgabe

Verzoegerte ERROR Behandlung
ERROR Quelle: File

Verwandte Begriffe

Err Objekt

OPTION EXPLICIT

Beschreibung

OPTION EXPLICIT wird auf Skriptebene verwendet, um eine ausdrückliche Deklaration aller Variablen zu erzwingen. Wenn OPTION EXPLICIT eingesetzt wird, muss er vor jeder anderen FUNKTION oder SUB Prozedur eingegeben werden und jede nicht deklarierte Variable erzeugt dann einen Fehler. Verwenden Sie DIM oder REDIM zum Deklarieren der Variablen.

OPTION EXPLICIT kann die Programm Performance beachtlich verbessern und unterstützt einen guten Programmierstil. Dieser Befehl sollte möglichst immer verwendet werden.

Wenn Sie STEP oder TRACE verwenden wollen, muss OPTION EXPLICIT die erste Zeile in Ihrem Programm sein.

Beispiel

```
OPTION EXPLICIT
DIM Teacher
Teacher = "Mr. Garrison"
```

Verwandte Begriffe

DIM, REDIM

ADDOBJECT "OptionButton", *name*, *xpos*, *ypos*,
width, *height*

Beschreibung

OptionButton wird verwendet, um ein frei positionierbares rundes Umschalt-Objekt (toggle) auf dem Bildschirm auszugeben. Die erforderliche Komponente, *name*, ist der Name der Variablen, die dem Programm als Referenz auf das Objekt dient. Sie muss den Standard-Namenskonventionen entsprechen. Die ebenfalls erforderlichen Komponenten, *xpos*, *ypos*, *width*, und *height* sind numerische Ausdrücke, die zusammen die Größe und Position (in Pixeln) des Objekts bestimmen, gemessen von der linken oberen Ecke. Die *Value* Eigenschaft ist FALSE, wenn der Kreis nicht ausgefüllt ist; sie ist TRUE, bei ausgefülltem Kreis.

Um mehrere OptionButtons in einer exklusiven Gruppe, oft als RadioButtons bezeichnet, zusammen zu fassen, verwenden Sie die Group Eigenschaft. Das Setzen der Group Eigenschaft auf TRUE, erzeugt eine neue Gruppe. Nachfolgend erzeugte OptionButtons bei denen die Eigenschaft Group auf FALSE gesetzt wird, werden in der Reihenfolge ihrer Erzeugung dieser Gruppe zugeordnet.

Unterstützte Eigenschaften (siehe "Eigenschaften")

Alignment, BackColor, Caption, Enabled, FontBold, FontItalic, FontName, FontSize, FontStrikethru, FontUnderline, ForeColor, Group, Height, Hwnd, Left, Name, ParentHwnd, TabStop, Tag, Text, Timer, Top, Value, Visible, Width, WindowLong

Unterstützte Methoden (siehe "Methoden")

Hide, Move, SetFocus, Show

Unterstützte Ereignisse (siehe "Ereignisse")

Click, GotFocus, KeyDown, KeyPress, KeyUp, LostFocus, Timerf

Beispiel

```
REM OptionButton Beispiel
'OptionButton ist ein Umschalter mit Kreis
'oft als RadioButton bezeichnet
ADDOBJECT "OPTIONBUTTON", "O1", 10, 10, 100, 20
O1.CAPTION="FEMALE"
O1.VALUE=TRUE
O1.GROUP=FALSE
ADDOBJECT "OPTIONBUTTON", "O2", 10, 30, 100, 20
O2.CAPTION="MALE"
O2.GROUP=FALSE
```

```
ADDOBJECT "OPTIONBUTTON", "O3", 10, 50, 100, 20  
O3.CAPTION="UNDISCLOSED"  
O3.GROUP=FALSE
```

AUSGABE

- ☒ Female
- ☐ Male
- ☐ Undisclosed

Verwandte Begriffe

ADDOBJECT, CheckBox, Events, Methods, Properties

result = x OR y

Beschreibung

OR gibt das Ergebnis der logischen Disjunktion zweier Ausdrücke zurück. *result* ist TRUE, wenn einer oder beide Ausdrücke TRUE ergeben, ansonsten ist *result* FALSE.

OR führt auch einen bit-weisen Vergleich zweier numerischer Ausdrücke durch. Jedes Bit in *result* wird auf 1 gesetzt, wenn eines der korrespondierenden Bits in x oder y 1 ist, andernfalls auf 0.

Beispiel

```
REM OR Beispiel
'OR fuer logische oder bit-weise Disjunktion
DIM Test1, Test2, Test3, x, y
x = 1
y = 5
Test1 = x > 0 OR y < 10
Test2 = x > 0 OR y > 10
Test3 = x OR y
PRINT "Logisch:"
PRINT "  x > 0 OR y < 10 = " & Test1
PRINT "  x > 0 OR y > 10 = " & Test2
PRINT "Bitweise:"
PRINT "  x OR y = " & Test3
```

Ausgabe

```
Logisch:
  x > 0 OR y < 10 = True
  x > 0 OR y > 10 = True
Bitweise:
  x OR y = 5
```

Verwandte Begriffe

AND, EQV, IMP, NOT, XOR

`Output.property=value`

Beschreibung

Das Output-Objekt wird automatisch erzeugt, wenn Sie ein NS Basic/CE Programm ablaufen lassen und ist das Objekt, das am weitesten unten angezeigt wird. Es gehört zu den PictureBox-Objekten, so dass Sie die gleichen Eigenschaften, Methoden und Ereignisse nutzen können. Schauen Sie in Tech Note 7 nach, um die komplette Liste zu sehen. Zusätzlich werden auch die Ausgaben eines PRINT Befehls in diesem Objekt angezeigt.

Wenn das Output-Objekt per CloseBox in der rechten oberen Ecke geschlossen wird, erhält ihr Programm eine Form_close Ereignismeldung. Sie können das nutzen, um ein paar Aufräumarbeiten durchzuführen.

Unterstützte Eigenschaften (siehe "Eigenschaften")

Die gleichen, wie bei der PictureBox.

Unterstützte Methoden (siehe "Methoden")

Die gleichen, wie bei der PictureBox.

Unterstützte Ereignisse (siehe "Ereignisse")

Die gleichen, wie bei der PictureBox, plus Ausgabe_Close, Ausgabe_Size

Beispiel

```
REM CloseBox Aktion darstellen
SUB form_close
    MSGBOX "Die AusgabeBox wird geschossen"
END SUB
```

Ausgabe

Verwandte Begriffe

ADDOBJECT, Events, Methods, Properties, PictureBox, KeyPreview

```
ADDOBJECT "PictureBox"[, name[, xpos[, ypos  
[, width[, height]]]]]
```

Beschreibung

PictureBox wird verwendet, um ein frei positionierbares rundes Text- und Grafik-Objekt auf dem Bildschirm auszugeben. Die optionale Komponente, *name*, ist der Name der Variablen, die dem Programm als Referenz auf das Objekt dient. Sie muss den Standard-Namenskonventionen entsprechen. Die ebenfalls erforderlichen Komponenten, *xpos*, *ypos*, *width*, und *height* sind numerische Ausdrücke, die zusammen die Größe und Position (in Pixeln) des Objekts bestimmen, gemessen von der linken oberen Ecke. Das Ausgabefenster selbst, ist eine PictureBox. Die Programmvariable Output kann mit PictureBox- Eigenschaften, -Methoden und –Ereignissen verwendet werden. Im folgenden sehen Sie eine kurze Liste der Eigenschaften, Ereignisse und Methoden der PictureBox. Die gesamte Liste finden sie in der Tech Note 7.

Unterstützte Eigenschaften (siehe "Eigenschaften")

BackColor, FontBold, FontItalic, FontName, FontSize, FontStrikethru, FontUnderline, ForeColor, Height, Left, Name, Tag, Text, Top, Width

Tabelle 17: PictureBox Eigenschaften

Name	Beschreibung
BorderStyle	0 = kein, 1 = Rand Strichstärke 1
DrawWidth	Strichstärke für Zeichenobjekte
FillColor	Farbe, mit der Figuren, Kreise oder Boxen gefüllt werden
FillStyle	0 = ausgefüllt, 1 = transparent
FontTransparent	True/False
Picture	String Image-Dateinamen
ScaleHeight	Höheneinheiten bei Anwenderskalierung
ScaleLeft	Linke Ecke des Objekts, bei Skalierung

ScaleMode	0 durch Anwender definiert 1 Twips (1440 dpi) 2 Points (72 dpi) 3 Pixel 4 Charakter (horizontal = 120 twips, vertikal = 240 twips) 5 Inches
ScaleTop	Obere Ecke des aktuellen Objekt
ScaleWidth	Breiteneinheiten bei Anwenderskalierung
Tag	Zur frühen Verfügung

Unterstützte Methoden (siehe "Methoden")

Move

Tabelle 18: PictureBox Methoden

Name	Argumente	Beschreibung
Cls		Text und Grafik löschen
DrawCircle	x, y, radius[, color [, aspectRatio]]	Kreis oder Ellipse zeichnen
DrawLine	x1, y1, x2, y2[, color [, box [, fill]]]	box ist TRUE/FALSE
DrawPicture	filename, x1, y1[, width, height, x2, y2, width2, height2], rasterop]	Bild oder Teil davon auf dem Screen ausgeben (kann auch skaliert werden).
DrawPoint	x, y[, color]	Einzelnen Punkt zeichnen
DrawText	Text[,x,y]	Schreibt in die nächste Textzeile des Objekts oder ab x,y wenn angegeben
Refresh		Objekt erneut zeichnen
ScaleX	width, from, to	Breite ändern
ScaleY	height, from, to	Höhe ändern
SetScale	x1, y1, x2, y2	Skalieren auf ...
TextHeight	text	Gibt Texthöhe zurück, eingebettete CR's OK.

TextWidth	text	Gibt Länge der längsten Zeile zurück
-----------	------	--------------------------------------

Unterstützte Ereignisse (siehe "Ereignisse")

Click

Tabelle 19: PictureBox Ereignisse

Name	Argumente	Beschreibung
KeyDown†	keyCode+, shift*	Taste gedrückt
KeyPress†	keyCode+	Kombination von KeyDown und KeyUp.
KeyUp†	keyCode+, shift*	Taste losgelassen
MouseDown#	button, shift*, x, y	Objekt wurde mit Stylus oder Eingabegerät berührt
MouseMove#	button, shift*, x, y	Eingabegerät oder Stylus wurde verschoben während das Objekt berührt wurde
MouseUp#	button, shift*, x, y	Stylus wurde abgehoben oder Eingabegerät hat die Verbindung zum Objekt unterbrochen

- + Tastaturcode der Taste, die das Ereignis auslöste
- * 1 = Shift Taste, 2 = CTRL Taste, 4 = ALT Taste
- † Nur im Output PictureBox Objekt. Globale Eigenschaft KeyPreview muss TRUE sein, um dieses Ereignis zu empfangen.
- # Taste des Eingabegerätes mit dem das Objekt berührt wurde (0 = Stylus). x, y, geben die Lokation der Berührung relativ zum Objekt an

Beispiel

```
REM PictureBox Beispiel
ADDOBJECT "PictureBox", "PBox", 65, 0, 55, 20
PBox.BorderStyle = 1
PBox.BackColor = vbWHITE
PBox.DrawText " Button"
SUB PBox_Click
    PRINT Ausgabe.Width & " X " &
    Ausgabe.Height
```

END SUB

Ausgabe

(Bildschirmgröße ist maschinenabhängig)

Verwandte Begriffe

ADDOBJECT

PLAYSOUND(*file*, *hmod*, *flags*)**Beschreibung**

Die PlaySound Funktion spielt eine Melodie, die durch eine vorgegebene Datei, eine Ressource oder ein System-Ereignis bestimmt wird. *File* ist Der Name der zu spielenden Datei: eine 0 für dieses Argument stoppt die aktuelle Melodie. *Hmod* ist gewöhnlich 0 – wenn Sie eine Ressource-Datei verwenden, steht hier die Handle-Nummer der Ressource. *Flags* wird durch Addieren der nachfolgenden Flags gebildet:

Tabelle 20

Flag	Wert	Beschreibung
APP	?	Nutze spezifizierte Anwendung zum Spielen
ALIAS	&h00010000	<i>File</i> ist der Alias eines System-Ereignisses aus der Registry-Datei oder der WIN.INI Datei. Nicht mit FILENAME oder RESOURCE verwenden.
ASYNCH	&h00000001	Rücksprung nach Start der Melodie.
FILENAME	&h00020000	<i>File</i> ist ein Dateiname.
LOOP	&h00000008	Spielen bis PlaySound mit <i>file</i> auf 0 gesetzt aufgerufen wird.
MEMORY	&h00000004	<i>File</i> zeigt auf ein Melodie-Image im Speicher.
NODEFAULT	&h00000002	Leise abbrechen, wenn die Melodie nicht gefunden wird.
NOSTOP	&h00000010	Am Ende an andere Melodie übergeben.
NOWAIT	&h00002000	Rücksprung wenn Treiber nicht frei ist.
RESOURCE	&h00040004	<i>File</i> ist eine Ressource-ID; <i>Hmod</i> identifiziert die Instanz.
SYNC	&h00000000	PlaySound springt nach Beenden der Melodie zurück.

Beispiel

```
PlaySound "SystemStart",&h10000,0  
Sleep 500  
PlaySound 0,0,0 'stoppen nach .5 Sek.
```

Ausgabe

(Spielt Melodie für eine halbe Sekunde)

Verwandte Begriffe

WAVEVOLUME

PRINT**Befehl**

PRINT [*expressionA*[, *expressionB*[, *expressionC*[, ...]]]]

Beschreibung

PRINT schreibt Text ins Ausgabefenster. PRINT gibt bis zu 20, durch Kommata getrennte, Ausdrücke aus. Diese werden durch Tabulatoren voneinander getrennt und jeder Zeile wird ein CR angehängt. Wird PRINT ohne Ausdrücke verwendet, gelangt eine leere Zeile zur Ausgabe.

Sie sehen die Ausgabe dieses Befehls nicht, wenn Sie Forms verwenden, da diese das Ausgabefenster komplett verdecken.

Beispiel

```
REM PRINT Beispiel
'PRINT schreibt Text ins Ausgabefenster
PRINT "Hallo Welt!"
PRINT
PRINT "Jetzt ist es", NOW
```

Ausgabe

```
Hallo Welt!
Jetzt ist es      8/18/1998 10:52:44 PM
```

Verwandte Begriffe

ObjektName.Property[= *value*]

Beschreibung

Objekt Eigenschaften sind Variablen in Objekten. Die erforderliche Komponente, *ObjektName*, muss ein gültiger Objekt Ausdruck sein, der auf ein Objekt verweist, das dem Programm mittels ADDOBJECT Befehl hinzugefügt wurde. Die erforderliche Komponente, *Property*, ist der Name einer Objektvariablen. Weiter unten finden Sie eine Liste der gängigen Objekt Eigenschaften. Die optionale Komponente, *value*, wird der Objektvariablen zugewiesen. Änderungen an den Objekt-Eigenschaften können das Aussehen und das Verhalten eines Objekts beeinflussen.

Einige Eigenschaften haben Einfluss auf fundamentale Funktionen des Objekts und sollten daher unmittelbar nach Erstellen des Objekts gesetzt werden.

Manche Eigenschaften sind schreibgeschützt und können nur während der Programmierphase eingestellt werden. Nicht alle Eigenschaften können zur Laufzeit ausgelesen werden.

Tabelle 21: Eigenschaften

Eigenschaftename	Erlaubte Werte	Kommentare
Alignment+	Integer 0 = Left Justify 1 = Right Justify 2 = Center Justify	Text/Titel Text-Ausrichtung. Label, OptionButton
BackColor	Color (Integer)	Hintergrundfarbe des Objekts. Farben können RGB Werte oder der Index aus einer Farbpalette sein.
Bottom	Integer	überholt – Height verwenden
BorderStyle	0 = none 1 = line	Date, Label, TextBox, Time
Caption	String	Im Objekt angezeigter Text. CheckBox, ComboBox, CommandButton, Label, TextBox

Eigenschaftename	Erlaubte Werte	Kommentare
Date	Date type	Datum setzen, d.h. Date.date=cdate("8/5/02")
Default	true/FALSE	Taste als Vorgabe setzen, für den Fall, dass Enter gedrückt wird.
Encrypted	TRUE/FALSE	Projekt Eigenschaft. Gespeicherte Datei wird bei gesetzter Eigenschaft verschlüsselt. Nur im Desktop IDE verfügbar.
Enabled	True	Reagiert das Objekt auf Anwender Eingaben?
ExpandedHeight	Integer	Größe der aufgeklappten ComboBox.
FontBold	TRUE/FALSE	
FontItalic	TRUE/FALSE	
FontName	String	
FontSize	Integer	
FontStrikethru	true/FALSE	
FontUnderline	true/FALSE	
FontWeight	integer, 400 = normal 700 = bold	Werte größer 550 werden zu 700 konvertiert, alle anderen zu 400
ForeColor	Color (Integer)	Vordergrundfarbe des Objekts für Text und Zeichnungen. Farben können RGB Werte oder der Index aus einer Farbpalette sein.
LongFormat	true/FALSE	Format für Datumsanzeige. Date

Eigenschaftename	Erlaubte Werte	Kommentare
Group	TRUE/false	Startet eine neue OptionButton Gruppe. OptionButton.
Height	Integer	Anzahl der Pixel von oberer Ecke zu unterer Ecke
HideSelection	TRUE/false	Zeigt ausgewählten Text an oder verbirgt ihn, wenn das Steuerelement den Fokus verloren hat. TextBox
Hwnd†	Long	Interne Referenz auf das Objekt
IntegralHeight+	TRUE/FALSE	Beschränkt das Objekt auf eine Höhe, die das ganzzahlige Vielfache der individuellen Zeilenhöhe ist. ComboBox, ListBox
Left	Integer	Anzahl der Pixel von der linken Ecke des Ausgabefensters zur linken Ecke des Objekts
List(index†)	string	Array mit Werten. Index Argument ist erforderlich†
ListCount†	Integer	Anzahl der Elemente der Liste. ComboBox, ListBox
ListIndex	Integer	Derzeitig ausgewähltes Element. ComboBox, ListBox
Locked	true/FALSE	Erlaubt oder sperrt Änderungen am Text, beeinflusst nicht die Möglichkeit, den Text auszuwählen. TextBox

Eigenschaftename	Erlaubte Werte	Kommentare
LowercaseOnly	true/FALSE	Beschränkt die Texteingabe auf Kleinschreibung. ComboBox, TextBox
MaxLength	Integer (0-30,000)	Anzahl der erlaubten Zeichen. TextBox (mehrzeilige)
MultiLine+	TRUE/FALSE	Erlaubt eingebettete Carriage Returns und Textumbruch. TextBox
MultiSelect	true/FALSE	Erlaubt es, mehrere Elemente zu wählen. ListBox
Name†		Der Name des Objekts
NewIndex	Integer	Einfüge-Index für AddItem in unsortierten Listen-Objekten. ComboBox, ListBox
NumbersOnly	true/FALSE	Beschränkt Texteingaben auf numerische Zeichen. TextBox
ParentHWnd	Window Handle	Der Vorgänger eines Objekts
Password	true/FALSE	Versteckt den Text, indem alle Zeichen durch ein Asterix (*) ersetzt werden. TextBox
Redraw	TRUE/false	Screen-Ausgabe erlauben oder sperren, während mehrere Elemente einer Liste hinzugefügt werden. ComboBox, ListBox
Right	Integer	Überholt – Width verwenden

Eigenschaftename	Erlaubte Werte	Kommentare
Scrollbars+	Integer 0 = No scroll bar 1 = Horizontal 2 = Vertical 3 = Both	ListBox (nur vertikal), TextBox (mehrzeilige)
SelCount	Integer	Schreibgeschützt. Gibt Anzahl der angewählten Elemente aus. ListBox
Selected(index)	TRUE/FALSE	Nur zur Laufzeit. Auswahl-Status des Elements <i>index</i> einer Liste abfragen oder setzen. ListBox
SelLength	Integer	Anzahl der ausgewählten Zeichen. Textbox
SelStart	Integer	Erstes selektiertes Zeichen. Textbox
SelText	String	Der ausgewählte String. TextBox
Sorted+	TRUE/FALSE	Elemente, die mit AddItem hinzugefügt werden, werden alphabetisch eingeordnet. ListBox, ComboBox
Style+	Integer 0 = editable list 2 = read-only list	Eingabezeile ist editable/read-only. ComboBox
Tabstop	TRUE/FALSE*	Schaltet Objekt frei zum Aktivieren der Tastatureingabe mittels Tabulator.
Tag	String	Zur eigenen Verwendung
Text	String	Text der im Objekt angezeigt wird.
Timer	Long	Das Ereignis <i>Objekt_timer</i> wird nach dieser Anzahl von Millisekunden

Eigenschaftename	Erlaubte Werte	Kommentare
		ausgelöst. 0 zum abschalten.
Top	Integer	Anzahl der Pixel von der oberen Ecke des Ausgabefensters zur oberen Ecke des Objekts
TopIndex	Integer	Abfragen oder festlegen, welches Element als erstes angezeigt wird. ComboBox, ListBox
UppercaseOnly	true/FALSE	Beschränkt die Texteingabe auf Großschreibung. ComboBox, TextBox
UseMnemonic	TRUE/false	Das Unterstrich-Zeichen kann dem Fenstertitel per Ampersand (&) hinzugefügt werden. Label
Value	TRUE/FALSE oder 0/1	Objekt ist angewählt oder selektiert. CheckBox, OptionButton
Visible†	TRUE/FALSE	Setzen mit Hide- oder Show-Methoden.
Width	Integer	Anzahl der Pixel von der linken zur rechten Ecke des Objekts
WindowLong(index)	Index: 0=ExWindowStyle 1=WindowStyle	Siehe Microsoft Windows CE Dokumentation zu den Werten.

† schreibgeschützte Eigenschaft. Nur während der Programmierung festlegbar.

+ beeinflusst fundamental die Objekt Anzeige

* Vorgabe ist FALSE für das Label Objekt

Beispiel

```
REM Objekt Property Beispiel
'Eine Eigenschaft ist eine Variable in einem
'Objekt
ADDOBJECT "TextBox", "Input" 50, 50, 100, 100
ADDOBJECT "CommandButton", "Button", 150,
170, 20, 80
PRINT "Text: " & Button.Text
Button.Text = "Druecke mich!"
PRINT "Sichtbar: " & CBOOL(Input.Visible)
Input.Hide
PRINT " Sichtbar: " & CBOOL(Input.Visible)
```

Ausgabe

```
Text:
Sichtbar: True
Sichtbar: False
```

Verwandte Begriffe

Events, Methods

RANDOMIZE [*number*]

Beschreibung

RANDOMIZE initialisiert den Zufallsgenerator. Der optionale Parameter, *number*, kann jeder gültige numerische Ausdruck sein. Er wird als Grundlage der RND Funktion benutzt. Wird *number* weggelassen, so wird der aktuelle Wert der Systemuhr als Basiswert eingesetzt. Um einen Zufallsfolge zu wiederholen, rufen Sie RND mit einem negativen Argument auf, danach RANDOMIZE mit einem numerischen Argument; der alleinige Aufruf von RANDOMIZE mit demselben Wert für *number* führt nicht zur selben Zufallsfolge.

Beispiel

```
REM RANDOMIZE Beispiel
'RANDOMIZE initialisiert Zufallsgenerator
RANDOMIZE
Random
RANDOMIZE 44
Random
RND -1
RANDOMIZE 169
Random

SUB Random
  DIM Ret
  RET = ""
  FOR i = 1 to 4
    RET = RET & INT((RND * 1000) + 1) & " "
  NEXT
  PRINT "Vier Zufallszahlen:", RET
END SUB
```

Ausgabe

```
Vier Zufallszahlen: 8 912 43 537
Vier Zufallszahlen: 33 6 430 51
Vier Zufallszahlen: 36 6 192 54
```

Verwandte Begriffe

RND

REDIM**Befehl**

```
REDIM [PRESERVE] nameA[(subscriptA[, subscriptB[,  
    subscriptC...]])][, nameB...[, nameC...[...]]]
```

Beschreibung

REDIM wird verwendet, um Speicherplatz für feste Arrays neu zuzuteilen. Die erforderliche Komponente, *nameA*, ist der Name der Variablen. Er muss den Standard-Namens-Konventionen entsprechen. Die optionale Liste von Indizes, *subscripts*, stellt die Obergrenzen der Dimensionen eines Array's dar. Es können bis zu 60, durch Kommata getrennte numerische Ausdrücke verwendet werden. Das optionale Schlüsselwort, PRESERVE, bewahrt den Inhalt eines existierenden Arrays, wenn nur die letzte Dimension geändert wird. Die Untergrenze aller Array-Indizes ist immer 0.

Beispiel

```
REM REDIM Beispiel  
'REDIM weist Speicherplatz neu zu  
'Leere Variable namens "Foo"  
DIM Foo  
'Ein-dimensionales Array mit 10 Elementen  
REDIM Foo(9)  
'Zwei-dimensionales Array mit 600 Elementen  
'20 x 30  
REDIM Foo(19, 29)
```

Verwandte Begriffe

ARRAY, DIM

REM**Befehl**

REM *remarks*

'*remarks*

Beschreibung

REM wird eingesetzt, um Bemerkungen und Kommentare als nicht ausgeführten Text in ein Programm zu bringen. Der optionale Parameter, *remarks*, kann jeder Text sein. Wird REM als letzter Befehl zusammen mit anderen Befehlen in einer Zeile benutzt, muss er durch einen Doppelpunkt (:) von den anderen Befehlen getrennt werden. Das Apostroph (') Synonym steht stellvertretend für REM, jedoch ist zwischen Kommentar und Apostroph kein Leerzeichen erforderlich und kein Doppelpunkt bei mehreren Befehlen in einer Reihe.

Beispiel

```
REM REM Beispiel
REM Das Beispiel tut absolut nichts
'Es hat noch nicht einmal eine einzige
'lauffähigen Code
```

Ausgabe

Verwandte Begriffe

REPLACE

Funktion

REPLACE(*target*, *find*, *source*[, *start*[, *count*[, *compare*]]])

Beschreibung

REPLACE gibt einen String zurück, bei dem ein Teilstring durch einen anderen *count* – Mal ersetzt wurde. Der erforderliche Parameter, *target*, ist der String-Ausdruck in dem der Teilstring ersetzt werden soll. Der erforderliche Parameter, *find*, ist der String-Ausdruck der ersetzt werden soll. Der erforderliche Parameter, *source*, ist der String-Ausdruck der als Ersatz dienen soll. Der optionale Parameter, *start*, ist ein numerischer Ausdruck, der die Startposition der Suche innerhalb *target* spezifiziert. Wird *start* nicht angegeben, beginnt die Suche bei 1, also beim ersten Zeichen. Der optionale Parameter, *count*, ist ein numerischer Ausdruck, der die Anzahl der Ersetzungen angibt. Wird *count* nicht angegeben, wird -1 verwendet und alle passenden Ersetzungen werden durchgeführt. Der optionale Parameter, *compare*, wird verwendet, um den Typ der Suche zu spezifizieren.

Beispiel

```
REM REPLACE Beispiel
'REPLACE ersetzt Teilstrings
DIM Message, Ride
Message = "Guten Morgen, Kinder."
PRINT Message
Message = REPLACE(Message, "Morgen", _
    "Abend")
PRINT Message
Ride = "kleine Kinder - kleine Sorgen"
Ride = REPLACE(Ride, "kleine", "Kleine", _
    1, 1)
PRINT Ride
```

Ausgabe

```
Guten Morgen, Kinder.
Guten Abend, Kinder.
Kleine Kinder - kleine Sorgen
```

Verwandte Begriffe

RGB

Funktion

RGB(*red, green, blue*)

Beschreibung

RGB gibt eine ganze Zahl aus, die den Wert einer RGB Farbe darstellt. *Red*, *Green* und *Blue* sind Zahlen im Bereich von 0-255, die den entsprechenden Komponenten der Farbe zugeordnet werden.

Beispiel

```
REM RGB Beispiel
'RGB gibt RGB Farbe zurueck
ADDOBJECT "PictureBox","PB",50,50,20,20
REM Turn color to red
PB.backcolor=rgb(255,0,0)
REM Turn color to grey
PB.backcolor=RGB(127,127,127)
REM Turn color to white
PB.backcolor=RGB(255,255,255)
```

Ausgabe

(Erste ein rotes, dann ein graues und dann ein weißes Rechteck)

Verwandte Begriffe

RIGHT

Funktion

RIGHT(*string*, *length*)

RIGHTB(*string*, *length*)

Beschreibung

RIGHT gibt einen String zurück, der die angegebene Anzahl von Zeichen vom rechten Ende einer Zeichenkette darstellt. Der erforderliche Parameter, *string*, kann jeder gültige String-Ausdruck sein. Der erforderliche Parameter, *length*, kann jeder gültige numerische Ausdruck sein. Wenn *length* 0 ist, wird ein leerer String ("") zurück gegeben. Ist *length* größer als die Länge des Strings, wird der gesamte String zurück gegeben.

RIGHTB ist eine strikt byte-genaue Version von RIGHT. Wenn sie mit Unicode Strings auf Windows CE Geräte eingesetzt wird, gibt sie jeden einzelnen Charakter als normalen Charakter zurück.

Beispiel

```
REM RIGHT Beispiel
'RIGHT gibt Teilstring von rechts zurueck
DIM Wendy, Eric
Wendy = "Testaburger"
Eric = "Cartman"
PRINT "Das RIGHT 6 von " & Wendy & ":", _
    RIGHT(Wendy, 6)
PRINT "Das RIGHT 4 von " & Eric & ":", _
    RIGHT(Eric, 4)
```

Ausgabe

```
Das RIGHT 6 von Testaburger:  burger
Das RIGHT 4 von Cartman:      tman
```

Verwandte Begriffe

LEFT, MID

RND**Funktion**

RND[(*number*)]

Beschreibung

RND gibt eine Zahl einfacher Genauigkeit aus einer quasi-zufälligen Sequenz zwischen 0 und 1 aus. Der optionale Parameter, *number*, kann jeder gültige numerische Ausdruck sein. Er wird als Startpunkt für den Zufallsgenerator genutzt.

Tabelle 22: RND Startwerte

Startwert	RND erzeugt
< 0	Jedes Mal die gleiche Nummer
> 0 oder None	Nächste Zufallszahl der Sequenz
0	Zuletzt erzeugte Zahl

Beispiel

```
REM RND Beispiel
'RND erzeugt Zufallszahlen
Random
RND -1
Random
RND -1
Random

SUB Random
  DIM Ret, i
  RET = ""
  FOR i = 1 TO 4
    RET = RET & (INT(100 * RND) + 1) & " "
  NEXT
  PRINT "Vier Zufallszahlen:", RET
END SUB
```

Ausgabe

```
Vier Zufallszahlen:   6 71 1 19
Vier Zufallszahlen:  33 28 4 51
Vier Zufallszahlen:  33 28 4 51
```

Verwandte Begriffe

RANDOMIZE

ROUND

Funktion

ROUND(*number*[, *fractionaldigits*])

Beschreibung

ROUND gibt eine Zahl zurück, die auf die angegebene Anzahl von Nachkommastellen gerundet wurde. Das erforderliche Argument, *number*, kann jeder gültige numerische Ausdruck sein. Das optionale Argument, *fractionaldigits*, ist die Anzahl der Dezimalstellen, auf die gerundet werden soll. Wird *fractionaldigits* nicht angegeben, ist der Vorgabewert 0 und ROUND gibt eine ganze Zahl zurück.

Beispiel

```
REM ROUND Beispiel
'ROUND rundet auf angegebene Anzahl von
'Dezimalstellen
DIM Pi, Pure, Ate
Pi = ROUND(3.14159265, 4)
PRINT Pi
Pure = ROUND(99.4444, 2)
PRINT Pure
Ate = ROUND(SQR(69))
PRINT Ate
```

Ausgabe

```
3.1416
99.44
8
```

Verwandte Begriffe

INT, FIX

RTRIM

Funktion

RTRIM(*string*)

Beschreibung

RTRIM gibt eine Zeichenkette zurück, bei der alle Leerzeichen am Ende entfernt wurden. Der erforderliche Parameter, *string*, kann jeder gültige Stringausdruck sein.

Beispiel

```
REM RTRIM Beispiel
'RTRIM trennt Leerzeichen am Ende ab
DIM Spacey
Spacey = "K "
PRINT "(" & Spacey & ")"
PRINT "(" & RTRIM(Spacey) & ")"
```

Ausgabe

```
(K )
(K)
```

Verwandte Begriffe

LTRIM, TRIM

RUNAPPATEVENT

Befehl

RUNAPPATEVENT *app, event*

Beschreibung

RUNAPPATEVENT wird verwendet, um ein Programm in Abhängigkeit eines Betriebssystem-Ereignisses zu starten. Die erforderliche Komponente, *app*, ist der Kommandozeilentext, der ausgeführt werden soll. Die erforderliche Komponente, *event*, ist das Betriebssystem-Ereignis, das zum Auslösen des Programmstarts verwendet werden soll.

Ereignis	Wert
NONE	0
TIME_CHANGE	1
SYNC_END	2
ON_AC_POWER	3
OFF_AC_POWER	4
NET_CONNECT	5
NET_DISCONNECT	6
DEVICE_CHANGE	7
IR_DISCOVERED	8
RS232_DETECTED	9
RESTORE_END	10
WAKEUP	11
TZ_CHANGE	12

Beispiel

```
REM RUNAPPATEVENT Beispiel
'RUNAPPATEVENT startet ein Programm
'durch ein Betriebssystem-Ereignis
RUNAPPATEVENT "\\Windows\\player.exe", 3
```

Verwandte Begriffe

RUNAPPATIME

RUNAPPATTIME**Befehl**

RUNAPPATTIME *app, yy, mo, dd, hh, mm, ss*

Beschreibung

RUNAPPATTIME wird verwendet, um ein Programm zu einem bestimmten Zeitpunkt in der Zukunft zu starten. Die erforderliche Komponente, *app*, ist der Kommandozeilentext, der ausgeführt werden soll. Die erforderlichen Komponenten, *yy, mo, dd, hh, mm*, und *ss*, sind numerische Werte, die genutzt werden, um den Zeitpunkt des Programmstarts festzulegen.

Beispiel

```
REM RUNAPPATTIME Beispiel
'RUNAPPATTIME start ein Programm zu einem
'bestimmten Zeitpunkt
RUNAPPATTIME "\Windows\player.exe", 2004, _
    5, 25, 14, 30
```

Verwandte Begriffe

RUNAPPATEVENT

ScrollBar

Objekt

ADDOBJECT "HScrollbar", *name*, *xpos*, *ypos*, *width*, *height*

ADDOBJECT "VScrollbar", *name*, *xpos*, *ypos*, *width*, *height*

Beschreibung

Zeigt eine Bildlaufleiste (scrollbar) an. Es wird ein Change-Ereignis ausgelöst, wenn die Right, Left, Up oder Down Pfeile angetippt werden oder wenn der Schieberegler (slider) bewegt wird. SmallChange muss mindestens 1 sein.

Unterstützte Eigenschaften (siehe "Eigenschaften")

Enabled, Height, HWnd, LargeChange, Left, Max, Min, Name, ParentHWnd, SmallChange, TabStop, Tag, Top, Timer, Value, Visible, Width, WindowLong

Unterstützte Methoden (siehe "Methoden")

Hide, Move, SetFocus, Show

Unterstützte Ereignisse (siehe "Ereignisse")

Click, GotFocus, KeyDown, KeyPress, KeyUp, LostFocus, Timer

Beispiel

```
REM VScrollbar Beispiel
ADDOBJECT "VScrollbar", "Button", _
    120, 10, 20, 100
```

Ausgabe

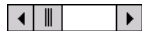


Beispiel

```
REM HScrollbar Beispiel
ADDOBJECT " HScrollbar ", "Button", _
    10, 120, 100, 20
```

```
SUB Button_Change
    PRINT "Taste gedrueckt"
    KillFocus
END SUB
```

Ausgabe



Verwandte Begriffe

ADDOBJECT, Events, Methods, Properties

SECOND

Funktion

SECOND(*time*)

Beschreibung

SECOND gibt eine ganze Zahl im Bereich von 0 bis 59 zurück, der die Sekunde in der Minute für einen vorgegebenen Zeitpunkt repräsentiert. Der erforderliche Parameter, *time*, kann jeder Ausdruck sein, der einen Zeitpunkt darstellt.

Beispiel

```
REM SECOND Beispiel
'SECOND gibt die Sekunden der Minute aus
PRINT "Die SECOND von " & NOW & " ist " _
      & SECOND(NOW)
```

Ausgabe

Die SECOND von 8/18/1998 10:52:44 PM ist 44

(Beispiel-Ausgabe ist systemabhängig)

Verwandte Begriffe

DATE, DAY, HOUR, MINUTE, MONTH, NOW, TIME, YEAR

SELECT CASE**Befehl**

```
SELECT CASE testexpression
  [CASE expressionlistA
    [statementsA]]
  [CASE expressionlistB
    [statementsB]]
  [CASE expressionlistC
    [statementsC]]...
  [CASE ELSE
    [elstatements]]
END SELECT
```

Beschreibung

SELECT CASE wertet einen Test-Ausdruck aus, um in Abhängigkeit des Ergebnisses, bestimmte Gruppen von Befehlen auszuführen. Eine *expressionlistN* Komponente ist für jede optionale CASE Klausel innerhalb des SELECT CASE Befehls erforderlich. Die optionale Komponente, *statementsN* ist die Gruppe der Befehle, die ausgeführt wird, wenn *testexpression* mit einem der Ausdrücke in der *expressionlistN* übereinstimmt. Innerhalb einer CASE Klausel werden die Befehle bis zur nächsten CASE Klausel oder bis zum END SELECT ausgeführt. Wenn alle Befehle einer CASE Klausel ausgeführt wurden, wird die Abarbeitung mit dem ersten Befehl nach dem END SELECT fortgeführt. Wenn *testexpression* mit keinem der Ausdrücke in der *expressionlists* übereinstimmt und CASE ELSE verwendet wird, werden die *elstatements* ausgeführt. Ist CASE ELSE nicht vorgesehen und *testexpression* findet keine Übereinstimmungen in einer der *expressionlists*, wird mit dem nächsten Befehl nach dem END SELECT weiter gemacht.

Beispiel

```
REM SELECT CASE Beispiel
'SELECT CASE fuer bedingte Ausfuehrung
CheckHat("Blue")
CheckHat("Orange")
SUB CheckHat(Hat)
  SELECT CASE Hat
  CASE "Blue"
    PRINT "Kyle's Hut"
  CASE "Green"
    PRINT "Stan's Hut"
  CASE "Cyan"
    PRINT "Eric's Hut"
```



```
CASE "Orange", "Hood"
    PRINT "Kenny's Hut"
CASE "White"
    PRINT "Chef's Hut"
CASE "Striped"
    PRINT "Mr. Hut"
CASE "Christmas", "Santa"
    PRINT "Hr. Hankey's Hut"
CASE ELSE
    PRINT "Unbekannter Hut"
END SELECT
END SUB
```

Ausgabe

Kyle's Hut
Kenny's Hut

Verwandte Begriffe

IF...THEN...ELSE

SENDKEY**Befehl**

SENDKEY *keyFlags*, *keyChar*

Beschreibung

SENDKEY wird verwendet, um per Programm Tastatureingaben durchzuführen. Die erforderliche Komponente, *keyFlags*, ist eine Kombination von Modifizierungsschlüsseln, die eine Tastatureingabe begleiten können (shift, ctrl oder alt). Die erforderliche Komponente, *keyChar*, ist der numerische Wert der Taste, die betätigt wird.

Beispiel

```
REM SENDKEY Beispiel
'SENDKEY taetigt Tastatureingabe
TextBox1.SetFocus
SENDKEY 0, 97
SENDKEY 0, 98
SENDKEY 0, 99
'Gibt abc in der TextBox ein
```

SET**Befehl**

SET *Objectvariable* = {*Objectexpression* | NOTHING}

Beschreibung

SET wird verwendet, um Objekte, Variablen zuzuweisen. Die erforderliche Komponente, *Objectvariable*, ist ein Variable, die den Namenskonventionen für Variablen entspricht. Die erforderliche Komponente, *Objectexpression*, ist der Name eines Objekts, einer Variable, die eine Referenz auf ein Objekt enthält oder FUNKTION Aufruf, der ein Objekt zurück gibt. Das optionale Schlüsselwort, NOTHING, entfernt die Verknüpfung zwischen *Objectvariable* und jedem verknüpften Objekt. Wenn ein Objekt keine Variablen mehr hat, die es referenzieren, werden die System- und Speicherressourcen, die ihm zugeteilt wurden, wieder frei gegeben.

Beispiel

```
REM SET Beispiel
'SET verknuepft Objekte mit Variablen
DIM ButtonRef
ADDOBJECT "Picturebox", "Button", 0, 0, 10, 10
SET ButtonRef = Button
```

Ausgabe

Verwandte Begriffe

IS

```
SETMENU "menustring[menukey]", menulist
```

Beschreibung

SETMENU fügt dem Ausgabefenster benutzerdefinierte Menüs zu, während das Programm ausgeführt wird. Die erforderliche Komponente, *menustring*, ist der Text, der mit dem Menüpunkt angezeigt wird. Die optionale Komponente, *menukey*, ist der Variablenname, der dem Menüpunkt zugeordnet wird. Ein *menukey* wird vom *menustring* durch zwei vertikale Balken (|) getrennt und sollte verwendet werden um: einen langen *menustring* abzukürzen, wenn der *menustring* Leerezeichen oder Sonderzeichen enthält, oder um mehrere Menüpunkte einem *menustring* zuzuordnen. Wird kein *menukey* angegeben, wird der Inhalt des *menustring* verwendet. Die erforderliche Komponente, *menulist*, ist ein Array von String-Ausdrücken, das die *menukeys* eines Untermenüs eines Menüs repräsentiert. Der *menukey* des Hauptmenüs der Anzeige ist "Titlebar."

Wenn ein Menüpunkt angewählt wird, erhält das Programm eine Ereignismeldung. Um auf dieses Ereignis zu reagieren, bauen Sie eine PUBLIC SUB Prozedur mit folgendem Syntax ein:

```
SUB menukey_click()  
    'Etwas als Antwort tun  
END SUB
```

Jeder *menustring* der mit einem Strich (hyphen) (-) beginnt, wird dem Menü als nicht anwählbarer Trennstrich hinzugefügt. Separatoren unterstützen keine Untermenüs.

SETMENU kann verwendet werden, um Menüs dynamisch zu ändern. Jedes Mal, wenn SETMENU aufgerufen wird, werden die betroffenen Menüs im Ausgabefenster aktualisiert. Wenn ein Menüpunkt verändert wird, werden alle seine Untermenüs gelöscht.

Um die Menü-Navigation per Tastatur zu unterstützen, kann 1 Zeichen eines *menustring* unterstrichen werden. Fügen Sie dazu vor dem entsprechenden Zeichen im *menustring* ein Ampersand Zeichen (&) ein.

Um dem Menü eine zweite Spalte hinzuzufügen (um z.B. Tastaturkürzel anzuzeigen), hängen Sie an den *menustring* ein Tabulatorzeichen (vbTAB) und den gewünschten zweiten String.

Beispiel

```
REM SETMENU Beispiel  
'SETMENU füegt der Ausgabe eigene Menues zu
```

```

DIM Menu
Menu = ARRAY("Datei", "Bearbeiten", "Hilfe")
SETMENU "Titel", Menu
Menu = ARRAY("&Neu" & vbTAB & "Ctrl+n||Neu", _
"&Oeffnen" & vbTAB & "Ctrl+o||Oeffnen", _
"&Sichern" & vbTAB & "Ctrl+s||Sichern", _
"_" , _
"&Letzte Dateien||Letzte", _
"_" , _
"&Ende" & vbTAB & "Alt+F4||Ende")
SETMENU "Datei", Menu
Menu = ARRAY("&1 Datei 1||RF1", _
"&2 Datei 2||RF2", "&3 Datei 3||RF3", _
"&4 Datei 4||RF4")
SETMENU "Letzte", Menu
SETMENU "Bearbeiten", _
    ARRAY("Ausschneiden", _
        "Kopieren", _
        "Einfuegen")
Menu = ARRAY("Ueber SETMENU
Beispiel...||BSP")
SETMENU "Hilfe", Menu
SUB New_Click()
    'Neue Datei erstellen
END SUB
SUB Open_Click()
    'Bestehende Datei oeffnen
END SUB

```

Ausgabe

SETPARENT

Befehl

SETPARENT *child, parent*

Beschreibung

SETPARENT wird verwendet, um ein Steuerelement in einen Container zu verschieben. Die erforderliche Komponente, *child*, ist das Steuerelement, das in den Container verschoben werden soll. Die erforderliche Komponente, *parent*, ist das Steuerelement, das später das *child* aufnehmen soll. Dabei muss es sich um ein Steuerelement handeln, bei dem die Container-Eigenschaft gesetzt ist (z.B. Frame). Beide Steuerelement müssen erzeugt worden sein, bevor SETPARENT aufgerufen wird. Das funktioniert sehr gut mit Steuerelementen, die Bestandteil von NS Basic sind. Steuerelemente von Drittanbietern stellen nicht immer die erforderlichen Schnittstellen für funktionierende Einbindung zur Verfügung.

Beispiel

```
REM SETPARENT Beispiel
'SETPARENT schiebt ein Steuerelement in ein
'anderes
ADDOBJECT "TextBox", "input", 5, 5, 140, 18
ADDOBJECT "Frame", "box", 0, 0, 200, 40
SETPARENT input, box
```

SGN(*number*)

Beschreibung

SGN gibt eine ganze Zahl zurück, die das Vorzeichen einer Zahl repräsentiert. Der erforderliche Parameter, *number*, kann jeder gültige numerische Ausdruck sein. Wenn *number* kleiner Null ist, wird -1 zurück gegeben. Ist *number* größer Null, so wird 1 zurück gegeben. Und ist *number* gleich Null, ist der Rückgabewert 0.

Beispiel

```
REM SGN Beispiel
'SGN gibt das Vorzeichen einer Zahl aus
DIM Pos, Neg, Zero
Pos = SGN(44)
Neg = SGN(-17)
Zero = SGN(100 - 100)
PRINT "Positiv:", Pos
PRINT "Negativ:", Neg
PRINT "Null:", Zero
```

Ausgabe

```
Positiv:      1
Negativ:     -1
Null:        0
```

Verwandte Begriffe

ABS

SHELLEXECUTE *verb*, *file* [,*parms*]

Beschreibung

SHELLEXECUTE führt ein externes Programm als separaten Prozess aus. Der erste Parameter, *verb*, gibt an, was mit dem zweiten Parameter *file* getan werden soll. Die dritte Komponente, *parms*, enthält die erforderlichen Parameter.

Table 23:

Verb	macht folgendes:
Open	Startet ein Programm
Print	Sendet ein Dokument an den Drucker (nicht in allen Windows CE Geräten verfügbar)
Explore	Öffnet den Pocket Internet Explorer

Beispiel

```
REM SHELLEXECUTE Beispiel
'Register ein Steuerelement eines
'Drittanbieters
shellExecute "open", "regsvrce.exe", _
"\windows\s309picture.dll"
'Druckt ein Dokument (systemabhaengig)
ShellExecute "print", "myReport.doc"
'oeffnet ein Bild im Webbrowser
shellExecute "explore", "myPict.jpg"
```

Ausgabe

(systemabhängig)

Verwandte Begriffe

SHOWFULLSCREEN

Befehl

SHOWFULLSCREEN *flags*

Beschreibung

SHOWFULLSCREEN wird verwendet, um verschiedene Sektionen des Bildschirms, die für das Betriebssystem reserviert sind, ein- oder auszublenden. Die erforderliche Komponente, *flags*, kann eine Kombination der Werte aus den nachfolgenden Tabelle sein.

Dieser Befehl funktioniert nur auf Pocket PC Geräten. Nicht alle Kombinationen funktionieren auf allen Geräten – Sie müssen sie daher selbst testen.

Flag	Wert
ShowTaskBar	1
HideTaskBar	2
ShowSIPButton	4
HideSIPButton	8
ShowStartIcon	16
HideStartIcon	32
ShowMenuBar	4096
HideMenuBar	8192

Beispiel

```
REM SHOWFULLSCREEN Beispiel
'SHOWFULLSCREEN zeigt/versteckt Bildschirm-
'Bereiche
SHOWFULLSCREEN 2 + 8 + 8192    'Zeigt den
'ganzen Bildschirm an
```

SHOWOKBUTTON**Befehl**

SHOWOKBUTTON *true|false*

Beschreibung

SHOWOKBUTTON wird verwendet, um die OK-Taste im Menübalken des Ausgabefensters auf einem Pocket PC Gerät ein- oder auszublenden. Per Vorgabe ist die OK-Taste versteckt und die Close Taste (X) wird angezeigt. Das Antippen der Close-Taste minimiert das Ausgabefenster, setzt aber die Programmausführung fort. Wenn die OK-Taste sichtbar ist und betätigt wird, wird das Ausgabefenster geschlossen und das Programm beendet.

Beispiel

```
REM SHOWOKBUTTON Beispiel
'SHOWOKBUTTON zeigt/versteckt OK im Menue
SHOWOKBUTTON true
```

SIN(*number*)

Beschreibung

SIN errechnet den Sinus eines Winkelausdrucks im Bogenmaß. Der erforderliche Parameter, *number*, kann jeder numerische Ausdruck sein. Der Rückgabewert ist eine Fließkommazahl mit doppelter Genauigkeit, im Bereich von -1 bis 1.

Um Grad in Bogenmaß zu wandeln, multiplizieren Sie Grad mit $\pi/180$. Um Bogenmaß in Grad zu wandeln, multiplizieren Sie rad mit $180/\pi$.

Beispiel

```
REM SIN Beispiel
'SIN errechnet den Sinus einer Zahl
PRINT "Der Sinus von 0 ist " & SIN(0)
```

Ausgabe

Der Sinus von 0 ist 0

Verwandte Begriffe

COS, TAN

SLEEP

Befehl

SLEEP *number*

Beschreibung

SLEEP überlässt die CPU anderen Prozessen. Der erforderliche Parameter, *number*, kann jeder numerische Ausdruck sein. Er gibt die Dauer der Unterbrechung in Millisekunden an..

Beispiel

```
REM SLEEP Beispiel
'SLEEP gibt die CPU frei
PRINT NOW
SLEEP(5000)
PRINT NOW
```

Ausgabe

```
8/18/1998 10:52:44 PM
8/18/1998 10:52:49 PM
```

Verwandte Begriffe

SPACE

Funktion

SPACE(*number*)

Beschreibung

SPACE gibt einen String zurück, der aus *number* Leerzeichen besteht. Das erforderliche Argument, *number*, kann jeder gültige numerische Ausdruck sein.

Beispiel

```
REM SPACE Beispiel
'SPACE erzeugt String aus Leerzeichen
DIM Spaces, Letter
Spaces = SPACE(4)
FOR i = 0 to 4
    Letter = LEFT(Spaces, i) & CHR(65 +
i)
    PRINT Letter
NEXT
```

Ausgabe

```
A
 B
  C
   D
    E
```

Verwandte Begriffe

STRING

SPECIALFOLDER(*ID*[, *True*])**Beschreibung**

Liest spezielle Ordernamen aus. Wenn True hinzugefügt wird, wird der Ordner erzeugt, falls er nicht existiert. Diese Funktion erlaubt es Ihnen, auf Geräten mit einer anderen Landessprache, als der Ihren, während der Laufzeit die richtigen Ordernamen zu verwenden und Ihr Programm auf zukünftige Geräte vorzubereiten. Die Funktion wird von Windows CE 3.0 und aktuelleren Geräten unterstützt. Wir haben auch festgestellt, dass die Ausgabe dieser Funktion systemabhängig ist.

2	\\Windows\\Start Menu\\Programs
3	Das Dateisystem-Verzeichnis, das die Benutzer-Programmgruppen, die auch Dateisystem-Verzeichnisse sind, enthält.
5	Das Dateisystem-Verzeichnis, das als gemeinsame Ablage für Dokumente dient.
6	Das Dateisystem-Verzeichnis, das als gemeinsame Ablage für die Favoriten.
7	Das Dateisystem-Verzeichnis, das der Startup Programmgruppe des Benutzers entspricht. Das System führt diese Programm nach dem Einschalten aus.
8	Das Dateisystem-Verzeichnis, das die zuletzt verwendeten Dokumente des Benutzers enthält.
13	Der Ordner, der Musikdateien enthält.
15	Der Ordner, der Videodateien enthält.
16	Das Dateisystem-Verzeichnis, in dem die Objekte des Desktops physikalisch gespeichert werden (nicht mit dem eigentlichen Desktop-Ordner verwechseln).
20	Der virtuelle Ordner, der Schriftarten enthält.
26	Das Dateisystem-Verzeichnis, das als gemeinsame Ablage für anwendungsspezifische Daten dient.
36	Der Windows Ordner.
38	Der Ordner, mit den Programmdateien.
39	Der Ordner, der Bilddateien enthält.

40	Der Ordner, der das Benutzerprofil enthält.
----	---

Beispiel

```
MsgBox SpecialFolder(6)
```

Ausgabe

(zeigt auf englischem Pocket PC "Windows\Favorites" an)

SPLIT

Funktion

SPLIT(*string*[, *delimiter*[, *count*[, *compare*]]])

Beschreibung

SPLIT gibt ein eindimensionales String-Array bestimmter Länge zurück, das durch Unterteilen eines einzelnen Strings an festgelegten Trennungszeichen erzeugt wird. Der erforderliche Parameter, *string*, kann jeder gültige String-Ausdruck sein. Wenn der String ein leerer String (""), gibt SPLIT ein leeres Array zurück. Das optionale Argument, *delimiter*, ist ein String-Ausdruck, dessen erstes Zeichen zum Separieren der Teilstrings verwendet wird. Wenn *delimiter* ein leerer String ist (""), wird ein Array mit einem Element zurück gegeben, das den kompletten String enthält. Der optionale Parameter, *count*, ist ein numerischer Ausdruck, der die Anzahl der Strings festlegt, die zurück gegeben werden sollen. Wenn *count* nicht angegeben ist, wird als Vorgabe -1 verwendet, wodurch alle Teilstrings zurück gegeben werden. Der optionale Parameter, *compare*, wird verwendet, um die Art der durchzuführenden Suche festzulegen.

Beispiel

```
REM SPLIT Beispiel
'SPLIT teilt einen String in Teilstrings
DIM List, Who, All, TopTwo
List = "Eric,Kenny,Kyle,Stan"
Who = SPLIT(List, ",")
PRINT Who(0), Who(1), Who(2), Who(3)
All = "First Second Third Fourth Fifth"
TopTwo = SPLIT(All, " ")
PRINT TopTwo(0), TopTwo(1)
```

Ausgabe

```
Eric           Kenny   Kyle           Stan
First  Second
```

Verwandte Begriffe

JOIN

SQR**Funktion**

SQR(*number*)

Beschreibung

SQR gibt die Quadratwurzel von *number* als Zahl doppelter Genauigkeit zurück. Der erforderliche Parameter, *number*, kann jeder gültige numerische Ausdruck sein, der Null oder eine größere Zahl ergibt.

Beispiel

```
REM SQR Beispiel
'SQR Quadratwurzel einer Zahl
PRINT "Die Wurzel von 69 ist " & SQR(69)
```

Ausgabe

```
Die Wurzel von 69 ist 8.30662386291807
```

Verwandte Begriffe

STRCOMP**Funktion**

STRCOMP(*string1*, *string2*[, *compare*])

Beschreibung

STRCOMP vergleicht zwei Strings und gibt als Ergebnis eine ganze Zahl zurück, die ausdrückt, in welchem Verhältnis die beiden Strings zueinander stehen. Die erforderlichen Parameter, *string1* und *string2*, müssen zwei gültige String-Ausdrücke sein. Der optionale Parameter, *compare*, gibt an, auf welche Art die Strings miteinander verglichen werden. STRCOMP gibt -1 zurück, wenn *string1* kleiner als *string2* ist, 0 wenn *string1* und *string2* gleich sind oder 1, wenn *string1* größer als *string2* ist.

Beispiel

```
REM STRCOMP Beispiel
'STRCOMP vergleicht zwei Strings
Sort "Kenny", "Kyle", vbBinaryCompare
Sort "Eric", "eric", vbTextCompare
Sort "Wendy", "Stan", vbBinaryCompare
SUB Sort(string1, string2, compare)
    DIM Order
    Order = STRCOMP(string1, string2, compare)
    IF Order < 0 THEN
        PRINT string1 & " kommt vor " & string2
    ELSEIF Order > 0 THEN
        PRINT string2 & " kommt vor " & string1
    ELSE
        PRINT string1 & " und " & string2 _
            & " sind gleich"
    END IF
END SUB
```

Ausgabe

```
Kenny kommt vor Kyle
Eric und eric sind gleich
Stan kommt vor Wendy
```

Verwandte Begriffe

STRING

Funktion

STRING(*number*, *character*)

Beschreibung

STRING gibt einen String zurück, der aus der vorgegebenen Anzahl eines bestimmten Zeichens erstellt wird. Das erforderliche Argument, *number*, kann jeder gültige numerische Ausdruck sein. Das ebenfalls erforderliche Argument, *character*, ist ein Zeichen oder ein String-Ausdruck, dessen erster Charakter im zurück gegebenen String wiederholt wird.

Beispiel

```
REM STRING Beispiel
'String zeugt einen String gleicher Zeichen
DIM Message
Message = STRING(10, "Hello World!")
PRINT Message
Message = STRING(10, CHR(ASC("i")))
PRINT Message
```

Ausgabe

```
HHHHHHHHHH
iiiiiiiiiii
```

Verwandte Begriffe

SPACE

STRREVERSE(*string*)

Beschreibung

STRREVERSE gibt den String zurück, der entsteht, wenn die Zeichen der Eingabe in umgekehrter Reihenfolge angeordnet werden. Das erforderliche Argument, *string*, kann jeder gültige String-Ausdruck sein.

Beispiel

```
REM STRREVERSE Beispiel
'STRREVERSE dreht die Zeichen eines Strings
DIM Show
Show = "Terrence and Phillip"
PRINT "Vorwaerts: " & Show
PRINT "Rueckwaerts: " & STRREVERSE(Show)
```

Ausgabe

```
Vorwaerts: Terrence and Phillip
Rueckwaerts: pillihP dna echerreT
```

Verwandte Begriffe

SUB**Befehl**

```
SUB procedurename[(arglist)]
```

```
    [statements]
```

```
    [EXIT SUB]
```

```
    [statements]
```

```
END SUB
```

Beschreibung

SUB deklariert eine Prozedur, *procedurename*, ohne Rückgabewert, die, unter Verwendung von *arglist* als Parametern, die Befehle *statements* ausführt. Der erforderliche Parameter, *procedurename*, wird verwendet, um die Prozedur aufzurufen und er muss den Standards für Variablennamen entsprechen. Die optionale Parameterliste, *arglist*, ist eine durch Kommata getrennte Liste von Variablen, die der Prozedur beim Aufruf übergeben wird. Die optionale Komponente, *statements*, wird als Rumpf der Prozedur ausgeführt. Es können optional beliebig viele EXIT SUB Befehle zum Verlassen der Prozedur verwendet werden.

Jeder Eintrag in *arglist* ist ein Argument, das der Prozedur wie im folgenden beschrieben, übergeben wird:

```
[BYVAL | BYREF] varname[(i)]
```

Das optionale Schlüsselwort, BYVAL, gibt an, dass der Prozedur eine Kopie der Variablen übergeben wird, wodurch der Inhalt der Ursprungsvariablen außerhalb der Prozedur unverändert bleibt. Das optionale Schlüsselwort, BYREF, spezifiziert, dass der Prozedur die Speicheradresse der Variablen übergeben wird, wodurch der Inhalt der Variablen nach außen hin sichtbar durch die Prozedur verändert werden kann. Standardmäßig werden die Parameter BYREF übergeben.

Eine FUNCTION Prozedur wird als eine SUB Prozedur aufgerufen, wenn der Rückgabewert im Aufruf keiner Variablen zugewiesen wird oder kein Teil eines Ausdrucks ist.

Eine SUB Prozedur, die ohne, oder nur mit einem Argument aufgerufen wird, kann mit leeren Klammern oder mit dem einzelnen Argument in Klammern aufgerufen werden. NS Basic/CE behandelt einen einzelnen Ausdruck in Klammern als einzelnen Ausdruck, nicht als Argumentenliste mit nur einem Element.

Beispiel

```
REM SUB Beispiel  
'SUB deklariert eine Prozedur
```

```

DIM PriceA, PriceB
PrintMenu "Mittwoch"
PriceA = 53
PriceB = 44
Sort PriceA, PriceB
PRINT "Tiefstpreis:", PriceA
SUB PrintMenu(day)
  IF day = "Mittwoch" THEN
    PRINT "Mittwoch ist Salisbury Steak Tag"
  END IF
END SUB
SUB Sort(BYREF x, BYREF y)
  DIM Temp
  IF x > y THEN
    Temp = y
    y = x
    x = Temp
  EXIT SUB
  ELSE
    Temp = x
    x = y
    y = Temp
  END IF
END SUB

```

Ausgabe

```

Mittwoch ist Salisbury Steak Tag
Tiefstpreis:  44

```

Verwandte Begriffe

CALL, FUNCTION

TAN

Funktion

TAN(*number*)

Beschreibung

TAN errechnet den Tangens eines Winkelausdrucks im Bogenmaß. Der erforderliche Parameter, *number*, kann jeder numerische Ausdruck sein. Der Rückgabewert ist eine Fließkommazahl mit doppelter Genauigkeit

Um Grad in Bogenmaß zu wandeln, multiplizieren Sie Grad mit $\pi/180$. Um Bogenmaß in Grad zu wandeln, multiplizieren Sie rad mit $180/\pi$.

Beispiel

```
REM TAN Beispiel
'TAN errechnet den Tangens einer Zahl
PRINT "Der Tangens von 0 ist " & TAN(0)
```

Ausgabe

Der Tangens von 0 ist 0

Verwandte Begriffe

COS, SIN

ADDOBJECT "TextBox", *name*, *xpos*, *ypos*, *width*, *height*

Beschreibung

TextBox wird verwendet, um veränderbaren Text in einem frei positionierbaren Objekt auf dem Bildschirm darzustellen. Eine TextBox ermöglicht Texteingaben über die Tastatur. Die erforderliche Komponente, *name*, ist der Name der Variablen, die dem Programm als Referenz auf das Objekt dient. Die erforderlichen Komponenten, *xpos*, *ypos*, *width*, und *height* sind numerische Ausdrücke, die Position und Größe des Objekts in Pixel angeben, gemessen von der oberen linken Ecke. Anmerkung: Setzen Sie die Multiline Eigenschaft auf TRUE, um eingebaute Zeilenrückläufe und Textumbrüche zu ermöglichen.

Unterstützte Eigenschaften (siehe "Eigenschaften")

BackColor, BorderStyle, Caption, Enabled, FontBold, FontItalic, FontName, FontSize, FontStrikethru, FontUnderline, ForeColor, Height, HideSelection, Hwnd, Left, Locked, LowercaseOnly, MaxLength, MultiLine, Name, NumbersOnly, ParentHwnd, Password, Scrollbars, selLength, selStart, selText, TabStop, Tag, Text, Timer, Top, UppercaseOnly, Visible, Width, WindowLong

Unterstützte Methoden (siehe "Methoden")

Hide, Move, SetFocus, Show

Unterstützte Ereignisse (siehe "Ereignisse")

Change, Click, GotFocus, KeyDown, KeyPress, KeyUp, LostFocus, Timer

Beispiel

```
REM TextBox Beispiel
'TextBox ist eine Texteingabebox
ADDOBJECT "TextBox", "Text", 10, 25, 90, 90
Text.Text = "Hallo Welt!"
SUB Text_Click
    PRINT "TextBox Text: " & Text.Text
END SUB
```

Ausgabe**Verwandte Begriffe**

ADDOBJECT

TIME

Funktion

TIME

Beschreibung

TIME gibt die aktuelle Systemzeit zurück.

Beispiel

```
REM TIME Beispiel
'TIME gibt die aktuelle Systemzeit aus
DIM RightNow
RightNow = TIME
PRINT "Es ist " & RightNow
```

Ausgabe

Es ist 10:52:44 PM

(Ausgabe ist systemabhängig)

Verwandte Begriffe

DATE, NOW

Time

Objekt

ADDOBJECT "Time", *name*, *xpos*, *ypos*, *width*, *height*

Beschreibung

Time wird benutzt, um ein Standard-Zeitauswahl-Objekt im Ausgabefenster anzuzeigen. Die erforderliche Komponente, *name*, ist der Name der Variablen, die dem Programm als Referenz auf das Objekt dient. Die erforderlichen Komponenten, *xpos*, *ypos*, *width*, und *height* sind numerische Ausdrücke, die Position und Größe des Objekts in Pixel angeben, gemessen von der oberen linken Ecke. Verwenden Sie keine MSGBOX innerhalb Ihres Time_Change Ereignisses: sie erzeugt einen Fehler. Dieses Objekt ist nicht auf Windows CE 2.0 Geräten verfügbar.

Unterstützte Eigenschaften (siehe "Eigenschaften")

BorderStyle, Date, Enabled, FontBold, FontItalic, FontName, FontSize, FontStrikethru, FontUnderline, Height, Hwnd, Left, Name, ParentHwnd, TabStop, Tag, Text, Timer, Top, Visible, Width, WindowLong

Unterstützte Methoden (siehe "Methoden")

Hide, Move, SetFocus, Show

Unterstützte Ereignisse (siehe "Ereignisse")

Change, DropDown

Beispiel

```
REM Time Beispiel
ADDOBJECT "Time", "Time", 20,50,120,20
Time.Date = "09/02/04 03:57:01 PM"
SUB Time_Change
    PRINT "Zeit geaendert auf " & time.text
END SUB
```

Ausgabe

Verwandte Begriffe

ADDOBJECT, Events, Methods, Properties

TIMESERIAL

Funktion

TIMESERIAL(*hour, minute, second*)

Beschreibung

TIMESERIAL gibt die Uhrzeit als Konstrukt der vorgegebenen Werte für *hour*, *minute*, und *second* aus. Der erforderliche Parameter, *hour*, kann jeder numerische Ausdruck im Bereich von 0 bis 23 sein. Die erforderlichen Parameter, *minute* und *second*, können beliebige numerische Ausdrücke sein.

Beispiel

```
REM TIMESERIAL Beispiel
'TIMESERIAL setzt eine Uhrzeit zusammen
DIM FiveThirty, Noon
FiveThirty = TIMESERIAL(11 - 6, 30, 0)
Noon = TIMESERIAL(12, 0, 0)
PRINT "Halb sechs:", FiveThirty
PRINT "Mittag:", Noon
```

Ausgabe

```
Halb sechs:      05:30:00 AM
Mittag: 12:00:00 PM
```

(Anzeige des Beispiels ist systemabhängig)

Verwandte Begriffe

DATESERIAL

TIMEVALUE(*time*)

Beschreibung

TIMEVALUE gibt eine Uhrzeit zurück die *time* entspricht. Üblicherweise ist *time* ein String, aber es kann jeder Ausdruck verwendet werden, der eine Uhrzeit im Bereich von 0:00:00 (12:00:00 AM) bis 23:59:59 (11:59:59 PM) repräsentiert.

Wenn die Zeichenkette *time* aus Zahlen besteht, die durch das Separatorzeichen für Zeit getrennt sind, werden Stunden, Minuten und Sekunden in der systemeigenen Kurzdarstellung des Zeitformats erkannt, wobei nicht spezifizierte Komponenten durch Null ersetzt werden. TIMEVALUE erkennt Zeit sowohl im 12-Stunden-, als auch im 24-Stundenformat.

Separatorzeichen für Zeit sind Zeichen, die Stunden, Minuten und Sekunden voneinander abgrenzen, wenn die Uhrzeit als formatierte Zeichenkette vorliegt. Sie sind systemabhängig.

Beispiel

```
REM TIMEVALUE Beispiel
'TIMEVALUE gibt eine Uhrzeit zurueck
DIM FiveThirty, Noon
FiveThirty = TIMEVALUE("5:30 PM")
Noon = TIMEVALUE("12:00")
PRINT "Halb sechs:", FiveThirty
PRINT "Mittag:", Noon
```

Ausgabe

```
Halb sechs:      05:30:00 PM
Mittag: 12:00:00 PM
```

(Anzeige des Beispiels ist systemabhängig)

Verwandte Begriffe

DATEVALUE

TRIM

Funktion

TRIM(*string*)

Beschreibung

TRIM gibt eine Zeichenkette zurück, bei der alle Leerzeichen am Anfang und am Ende entfernt wurden. Der erforderliche Parameter, *string*, kann jeder gültige Stringausdruck sein.

Beispiel

```
REM 'TRIM Beispiel
'TRIM löscht Leerzeichen vor und hinter dem
'String
DIM Spacey
Spacey = " K "
PRINT "(" & Spacey & ")"
PRINT "(" & TRIM(Spacey) & ")"
```

Ausgabe

```
( K )
(K)
```

Verwandte Begriffe

LTRIM, RTRIM

TYPENAME	Funktion
----------	----------

TYPENAME(*variable*)

Beschreibung

TYPENAME gibt einen String mit dem Typ einer Variablen zurück. Der erforderliche Parameter, *variable*, kann jede Variable sein.

Tabelle 24: TYPENAME Rückgabewerte

Rückgabewert	Beschreibung
Boolean	Boolescher Wert
Byte	Byte-Wert
Currency	Währungswert
Date	Datumswert
Decimal	Dezimalwert
Double	Fließkommawert, doppelte Genauigkeit
Empty	Nicht initialisiert
Error	Ein Fehlerwert
Integer	Ganzzahl
Long	Lange Ganzzahl
Nothing	Objektvariable die keine Objektreferenz enthält
Null	Wert ungültig
Object	Generelles Objekt
OObjecttype	Ein Objekt vom Typ Objecttype
Single	Fließkommawert, einfache Genauigkeit
String	Zeichenkette
Unknown	Unbekannt
Variant()	Array

Beispiel

```
REM TYPENAME Beispiel
'TYPENAME gibt Variablentyp als String aus
DIM nInteger, nSingle
nInteger = CINT(44)
PRINT 44 & " ist ein/e " &
TYPENAME(nInteger)
nSingle = CSNG(99.44)
PRINT 99.44 & " ist ein/e " &
TYPENAME(nSingle)
```

Ausgabe

44 ist ein/e Integer

99.44 ist ein/e Single

Verwandte Begriffe

ISARRAY, ISDATE, ISEMPY, ISNULL, ISNUMERIC,
ISOBJEKT, VARTYPE

UBOUND

Funktion

UBOUND(*array* [, *dimension*])

Beschreibung

UBOUND gibt eine Longintegerzahl zurück, die den größtmöglichen Index einer vorgegebenen Dimension eines Arrays darstellt. Der erforderliche Parameter, *array*, kann jede Array-Variable sein. Der optionale Parameter, *dimension*, gibt an, aus welcher Dimension die Obergrenze ermittelt werden soll, beginnend mit 1 als Vorgabe.

Beispiel

```
REM UBOUND Beispiel
'UBOUND gibt Array-Dimensions-Obergrenze an
DIM Other, Children(3), Parents(3, 1)
Other = ARRAY("Damien", "Pip", "Wendy")
PRINT "'Andere' Obergrenze:", UBOUND(Other)
PRINT "'Kinder' Obergrenze:", _
    UBOUND(CHILDREN, 1)
PRINT "'Eltern' Obergrenzen:", _
    UBOUND(Parents), UBOUND(Parents, 2)
```

Ausgabe

```
'Andere' Obergrenze:  2
'Kinder' Obergrenze:  3
'Eltern' Obergrenzen: 3      1
```

Verwandte Begriffe

ARRAY, DIM, LBOUND, REDIM

UCASE

Funktion

UCASE(*string*)

Beschreibung

UCASE gibt *string* mit allen Zeichen in Großschreibung zurück. Der erforderliche Parameter, *string*, kann jeder gültige String-Ausdruck sein.

Beispiel

```
REM UCASE Beispiel
'UCASE gibt String in Großbuchstaben aus
DIM Vet
Vet = "ned"
PRINT Vet & " in Grossbuchstaben ist " &
UCASE(Vet)
```

Ausgabe

```
ned in Grossbuchstaben ist NED
```

Verwandte Begriffe

LCASE

UPDATESCREEN

Befehl

UPDATESCREEN

Beschreibung

UPDATESCREEN erzwingt eine Neuzeichnung des Ausgabe-Fensters.

Beispiel

```
REM UPDATESCREEN Beispiel
'UPDATESCREEN zeichnet das Ausgabe-Fenster
'neu
UPDATESCREEN
```

Ausgabe

Verwandte Begriffe

VARTYPE(*variable*)

Beschreibung

VARTYPE gibt eine Ganzzahl zurück, die den Typ der Variablen widerspiegelt. Der erforderliche Parameter, *variable*, kann jede Variable sein, die keinen benutzerdefinierten Typ darstellt.

Wenn *variable* ein Array ist, besteht der Rückgabewert aus dem Wert für den Typ Array plus dem Wert, der den Element-Typ spezifiziert.

Tabelle 25: VARTYPE Rückgabewerte

Konstante	Wert	Beschreibung
vbEmpty	0	Nicht initialisiert (Vorgabe)
vbNull	1	Kein gültiger Wert
vbInteger	2	Ganzzahl
vbLong	3	Lange Ganzzahl
vbSingle	4	Fliesskommazahl (einfach)
vbDouble	5	Fliesskommazahl (doppelt)
vbCurrency	6	Währung
vbDate	7	Datum
vbString	8	String
vbObjekt	9	Objekt
vbError	10	Fehler
vbBoolean	11	Boolescher Wert
vbVariant	12	Variant (nur mit Variant-Arrays)
vbDataObjekt	13	Daten-Zugriffs-Objekt
vbByte	17	Byte
vbArray	8192	Array

Beispiel

```
REM VARTYPE Beispiel
'VARTYPE gibt Variablentyp als Zahl zurueck
DIM nInteger, nSingle
nInteger = CINT(44)
PRINT 44 & " ist VARTYPE " &
VARTYPE(nInteger)
nSingle = CSNG(99.44)
```

```
PRINT 99.44 & " ist VARTYPE " &  
VARTYPE(nSingle)
```

Ausgabe

```
44 ist VARTYPE 2  
99.44 ist VARTYPE 4
```

Verwandte Begriffe

ISARRAY, ISDATE, ISEMPY, ISNULL, ISNUMERIC,
ISOBJEKT, TYPENAME

WAITCURSOR**Befehl**

WAITCURSOR true|false

Beschreibung

Zum Anzeigen des WAITCURSOR wird dieser Wert auf TRUE gesetzt. Das Aussehen des Cursor hängt vom jeweiligen Gerät ab. Mit dem Setzen auf FALSE, schalten Sie den WAITCURSOR wieder ab.

Beispiel

```
REM WAITCURSOR Beispiel
WAITCURSOR TRUE
SLEEP 5000
WAITCURSOR FALSE
```

Ausgabe

Verwandte Begriffe

WAVEVOLUME

Funktion

WAVEVOLUME volume

Beschreibung

WAVEVOLUME wird verwendet, um den Lautstärkepegel des Gerätes einzustellen. Die erforderliche Komponente, *volume*, ist ein Wert von 0 bis 65535.

Beispiel

```
REM WAVEVOLUME Beispiel
'WAVEVOLUME setzt den Lautstaerkepegel des
'Geraetes
WAVEVOLUME=0           'Stummschaltung
WAVEVOLUME=32768       'Volumen 50%
WAVEVOLUME=65535       'Volles Volumen
MSGBOX WAVEVOLUME
```

Ausgabe

(MsgBox mit 65535)

Verwandte Begriffe

PLAYSOUND

WEEKDAY

Funktion

WEEKDAY(*date*[, *firstdayofweek*])

Beschreibung

WEEKDAY gibt eine ganze Zahl zurück, die den Tag in der Woche für ein vorgegebenes Datum repräsentiert. Der erforderliche Parameter, *date*, ist ein numerischer Ausdruck, eine Zeichenfolge oder jeder Ausdruck, der ein gültiges Datum darstellt. Der optionale Parameter, *firstdayofweek*, ist Sonntag, falls er nicht anders angegeben wird. Der Rückgabewert ist eine ganze Zahl aus der unten angegebenen Tabelle.

Für weitere Informationen zur *firstdayofweek* Konstante, sehen Sie unter "DateDiff" nach.

Tabelle 26: WEEKDAY Rückgabewerte

Konstante	Wert	Beschreibung
vbSunday	1	Sonntag
vbMonday	2	Montag
vbTuesday	3	Dienstag
vbWednesday	4	Mittwoch
vbThursday	5	Donnerstag
vbFriday	6	Freitag
vbSaturday	7	Samstag

Beispiel

```
REM WEEKDAY Beispiel
'WEEKDAY gibt Tag in der Woche als Zahl aus
DIM IndepDay, Birthday
IndepDay = WEEKDAY("July 4, 1776")
PRINT "Tag in der Woche fuer July 4, 1776:",
IndepDay
Birthday = WEEKDAY("12/27/70")
PRINT "Tag in der Woche fuer 12/27/70:",
Birthday
```

Ausgabe

```
Tag in der Woche fuer July 4, 1776:  5
Tag in der Woche fuer 12/27/70:      1
```

Verwandte Begriffe

WEEKDAYNAME

Funktion

WEEKDAYNAME(*number*[, *abbreviate*[, *firstdayofweek*]])

Beschreibung

WEEKDAYNAME gibt einen String zurück, der den Tag in der Woche darstellt. Das erforderliche Argument, *number*, ist eine ganze Zahl von 1 bis 7, die den Tag in der Woche repräsentiert. Das optionale Argument, *abbreviate*, ist ein boolescher Ausdruck. Ist er TRUE, wird eine Abkürzung des Wochentages mit drei Buchstaben zurückgegeben, ansonsten standardmäßig der volle Wochentagsname. Der optionale Parameter *firstdayofweek* ist Sonntag, falls er nicht anders angegeben wird.

Beispiel

```
REM WEEKDAYNAME Beispiel
'WEEKDAYNAME gibt Wochentagsname zurueck
DIM Day1, Day2
Day1 = 1
PRINT WEEKDAYNAME (Day1)
Day2 = 2
PRINT WEEKDAYNAME (Day2, TRUE, vbMonday)
```

Ausgabe

```
Sunday
Tue
```

Verwandte Begriffe

WITH *Object*
 [*statements*]

END WITH

Beschreibung

WITH ermöglicht es Ihnen, eine Reihe von Operationen an einem Objekt durchzuführen, ohne es jedes Mal explizit benennen zu müssen. Nur unter Windows CE 4.0 und folgenden.

Beispiel

```
ADDOBJECT "CommandButton", "CB", 124, 60, 108, 21
With CB
    .FontBold = True
    .Caption = "Druck Mich!"
    .FontItalic = True.
End With
```

Ausgabe

(CommandButton zeigt "Drueck Mich!" in fetter Schrägschrift an)

Verwandte Begriffe

CLASS

WHILE...WEND**Befehl**

```
WHILE condition  
    [statements]
```

```
WEND
```

Beschreibung

WHILE...WEND wiederholt eine Gruppe von Befehlen während eine angegebene Bedingung TRUE ist. Die erforderliche Komponente, *condition*, kann jeder Ausdruck, der TRUE oder FALSE als Ergebnis liefern kann, sein. Die optionale Komponente, *statements*, wird bei jedem Schleifendurchlauf ausgeführt. WHILE...WEND Befehle können verschachtelt werden und jeder WEND Befehl in einer verschachtelten Schleife gibt die Ausführung an die übergeordnete Schleife zurück.

Beispiel

```
REM WHILE...WEND Beispiel  
'WHILE...WEND wiederholt eine Gruppe von  
'Befehlen  
DIM Counter  
Counter = 1  
WHILE Counter < 5  
    PRINT "Counter = " & Counter  
    Counter = Counter + 1  
WEND
```

Ausgabe

```
Counter = 1  
Counter = 2  
Counter = 3  
Counter = 4
```

Verwandte Begriffe

DO...LOOP, FOR...NEXT, FOR EACH...NEXT

result = x XOR y

Beschreibung

XOR gibt das Ergebnis der logischen Exklusiv-oder Verknüpfung zweier Ausdrücke zurück. *result* ist TRUE, wenn einer (und auch nur dann) der beiden Ausdrücke x und y als TRUE ausgewertet wird, ansonsten ist *result* FALSE.

XOR kann auch für bitweisen Vergleich zweier numerischer Ausdrücke verwendet werden. Jedes Bit in *result* wird auf 1 gesetzt, wenn eines der beiden (aber nicht beide) korrespondierenden Bits in x oder y 1 ist, sonst wird es auf 0 gesetzt.

Beispiel

```
REM XOR Beispiel
'XOR fuehrt EXOR-Verknuepfung durch
DIM Test1, Test2, x, y
x = 2
y = 9
Test1 = x > 0 XOR y < 10
Test2 = x > 0 XOR y > 10
PRINT "Logisch:"
PRINT "  x > 0 XOR y < 10 = " & CSTR(Test1)
PRINT "  x > 0 XOR y > 10 = " & CSTR(Test2)
PRINT "Bitweise:"
PRINT "  x XOR y = " & (x XOR y)
```

Ausgabe

```
Logisch:
  x > 0 XOR y < 10 = False
  x > 0 XOR y > 10 = True
Bitweise:
  x XOR y = 11
```

Verwandte Begriffe

AND, EQV, IMP, NOT, OR

YEAR

Funktion

YEAR(*date*)

Beschreibung

YEAR gibt eine ganze Zahl zurück, die der Jahreszahl des eingegebenen Datums entspricht. Der erforderliche Parameter *date* kann jeder Ausdruck sein, der ein Datum darstellt.

Beispiel

```
REM YEAR Beispiel
'YEAR gibt das Jahr eines Datums als Integer
'zurueck
DIM IndepYear, Birthyear
IndepYear = YEAR("July 4, 1776")
PRINT "YEAR aus July 4, 1776:", IndepYear
Birthyear = YEAR("12/27/70")
PRINT "YEAR aus 12/27/70:", Birthyear
```

Ausgabe

```
YEAR aus July 4, 1776: 1776
YEAR aus 12/27/70:      1970
```

Verwandte Begriffe

DAY, HOUR, MINUTE, MONTH, NOW, SECOND, TIME

6. Weiterführende Themen

6.1 Programmierrichtlinien

Programmierrichtlinien sind eine Sammlung von Empfehlungen, die Ihnen beim Erstellen, Lesen, Debuggen und Warten von Programmen helfen sollen. Sie enthalten (sind aber nicht beschränkt auf):

- Richtlinien zur Namensgebung
- Richtlinien zur Textformatierung
- Kommentar-Richtlinien

Sinnvolle Programmierrichtlinien sollen Ihnen helfen, kontext-sensitive Informationen und verbesserte Lesbarkeit zu erreichen.

6.1.1 Richtlinien zur Namensgebung

Der einfachste Weg, in einem Programm kontext-sensitive Informationen bereit zu stellen, ist die Verwendung von präzisen, informativen Namen für Konstanten, Variablen und Prozeduren.

Nicht triviale Namen sollten den Sinn oder die Verwendung so genau wie möglich beschreiben. Wenn zusammengesetzte Wörter verwendet werden, wechseln Sie Groß/Kleinschreibung ab und/oder nutzen Sie das Unterstrich-Zeichen, um Worte abzugrenzen.

Konstanten sollten groß geschrieben werden, mit Unterstrichen zwischen den Worten.

Bei Variablennamen sollten alle Worte mit Großbuchstaben beginnen; sehen Sie von der Verwendung von Unterstrichen in Variablennamen ab. Wenn Sie weitere Informationen zur Verfügung stellen wollen, beginnen Sie jeden Variablennamen mit drei Buchstaben, die den Typ des in der Variablen gespeicherten Wertes charakterisieren (bln für BOOLEAN, int für INTEGER, str für STRING, etc.).

Prozedurnamen sollten klein geschrieben werden, wobei das erste Zeichen jedes Wortes jeweils groß sein sollte. Die Verwendung von Unterstrichen und Groß/Kleinschreibung sorgt dafür, dass Prozeduren leicht von Konstanten und Variablen unterschieden werden können..

6.1.2 Richtlinien zur Textformatierung

Textformatierung ist eine gute Möglichkeit die Lesbarkeit eines Programms zu verbessern. Verwenden Sie vertikal Leerzeichen um Anweisungen zu gruppieren oder zu separieren und horizontal, um Anweisungen nach ihrer logischen Struktur zu positionieren oder zu verschachteln.

Fügen Sie nach jeder Konstantendefinition, jeder Variablendefinition und jeder Prozedurdeklaration eine Leerzeile ein, um die Sektionen gegeneinander abzugrenzen. Eine Leerzeile kann auch auf Skriptebene hinter zusammengehörenden Anweisungen oder innerhalb von Prozeduren verwendet werden; dadurch wird das Bilden von Gruppen erleichtert.

Die Standard-Tabulatorweite beträgt zwei Leerzeichen. Verwenden Sie mindestens ein Tab um verschachtelte Gruppen von Anweisungen innerhalb von Prozeduren, Schleifen und Kontrollstrukturen darzustellen.

6.1.3 Kommentar-Richtlinien

Kommentare sollen zeigen, was der Programmierer erreichen wollte und wie er es erreichen wollte. Ebenso soll klar werden, was getan werden kann oder getan werden muss. Verwenden Sie Kommentare:

- Zu Beginn jedes Programms um Informationen zum Autor, zu den Aufgaben des Programms, zum Erstellungsdatum, zu den Änderungen inklusive den Anlässen dazu und den Rechten am Programm anzuzeigen.
- Um jeder Prozedur eine Einführung mit Erklärung der Funktionsweise, erwarteten Eingaben und erwarteten Ausgaben voranzustellen. Eine weitere Zeile für jede übergebene Variable, falls deren Zweck nicht eindeutig ist oder sie Bereichseinschränkungen unterliegt.
- Um eine Kurzbeschreibung der Verwendung und des Geltungsbereiches hinter jeder Konstanten- und Variablen-Deklaration anzufügen.
- Über einer Gruppe von zusammengehörenden Anweisungen um zu erklären, was sie tun.
- In Kontrollstrukturen um die einzelnen Zweige und Bedingungen genauer zu erklären.

6.2 Fehlerbehandlung

In einer perfekten Welt gibt es keine Fehler. Unsere Programme stellen selten perfekte Welten dar! Wir können unsere Anwender (oft auch uns selbst) mit zwei Techniken vor vielen Fehlern schützen: defensives Programmieren und Abfangen von Fehlern.

Die Grundidee der Fehlerbehandlung ist das Identifizieren von Programmteilen, die möglicherweise Laufzeitfehler erzeugen können und die Verwendung von speziellem Programmcode, um diese Fehler abzuarbeiten.

6.2.1 Defensives Programmieren

Defensives Programmieren hilft die Laufzeitfehler zu reduzieren, indem die Anzahl der Vermutungen während der Ausführung des Programms verringert wird. Dazu sollten Sie im folgendes tun:

- Initialisieren Sie Variablen direkt nachdem sie deklariert wurden und stellen Sie sicher, dass sie den richtigen Datentyp verwenden.
- Verwenden Sie die OPTION EXPLICIT Anweisung um sicher zu stellen, dass alle Variablen deklariert wurden, bevor sie verwendet werden. Dadurch werden Fehler durch falsch geschriebene Variablennamen vermieden.
- Das Verwenden der VARTYPE Funktion stellt sicher, dass selbstdefinierte Datentypen, deren Inhalte durch den Anwender eingegeben werden, die erwarteten Datentypen aufweisen.
- Stellen Sie sicher, dass numerische Werte nicht die Grenzen überschreiten, ehe Sie sie in numerischen Anweisungen verwenden.
- Informieren Sie sich über die Anzahl der Elemente in einem Array, ehe Sie versuchen auf eines zuzugreifen.

Je defensiver Sie programmieren, umso geringer wird die Gefahr, dass Laufzeitfehler auftreten. Gutes defensives Programmieren kann logische Fehler verhindern und Ihr Programm gegen Fehleingaben schützen.

6.2.2 Abfangen von Fehlern

Defensives Programmieren kann und sollte zum Verhindern möglichst vieler Laufzeitfehler verwendet werden. Für die Fälle, in denen das nicht reicht, kann die ON ERROR Anweisung genutzt werden, um auftretende Fehler abzufangen. Diese Methode wird als Abfangen von Fehlern (Error trapping) bezeichnet und sie ermöglicht es Ihnen, Fehler zu behandeln, ehe NS Basic/CE davon etwas bemerkt.

Laufzeitfehler, die grundsätzlich nicht durch defensives Programmieren verhindert werden können, treten auf, wenn das Programm mit dem Betriebssystem interagiert. Datei-Operationen verursachen am häufigsten Fehler, da sie System-Ressourcen anfragen, die zum aktuellen Zeitpunkt verfügbar sind oder aber auch nicht.

NS Basic/CE erlaubt das Abfangen von Fehlern nur innerhalb von Prozeduren und diese Möglichkeit kann sehr

einfach im Kern der Prozedur ein- oder ausgeschaltet werden. Wenn ein Programm einen Fehler erkennt, behandelt es diesen über eine Fehlerbehandlungsmethode (Error handler). Ein Error handler ist einfach ein Anweisungsblock, der ausgeführt wird, um einem Programm die Behandlung eines Fehlers zu ermöglichen. Lassen Sie uns kurz einen Blick auf eine einfache Fehlerbehandlung im Zusammenhang mit Datei-Operationen werfen:

6.2.3 Abfangen von Fehlern bei Datei-Operationen

```
ADDOBJECT "File", "MyFile"
DIM FileOpen

FileOpen = Open("Ein nicht existierender
Dateiname")
IF FileOpen THEN
    PRINT "Datei erfolgreich geoeffnet."
ELSE
    PRINT "Datei konnte nicht geoeffnet
werden."
END IF

FUNCTION Open(filename)
    ON ERROR RESUME NEXT
    MyFile.Open filename, 1
    IF ERR <> 0 THEN
        'Das ist der error handler der bebutzt
wird
        'wenn eine Datei nicht zum Lesen
geoeffnet werden kann.
        'Dieses Beispiel gibt einfach FALSE
zurueck.
        Open = FALSE
        ERR.Clear
        EXIT FUNCTION
    END IF
    MyFile.Close
    Open = TRUE
END FUNCTION
```

Ausgabe

Datei konnte nicht geoeffnet werden.

Einige Aktionen, die Ihr Error handler durchführen könnte:

- Eine hilfreiche Meldung ausgeben und die Operation wiederholen.
- Den Fehler durch setzen einer oder mehrerer Variablen auf einen Standardwert (z.b. könnten Sie eine Eingabe auf einen Maximalwert begrenzen).

- Eine eigene Fehlermeldung anzeigen, ein paar Aufräumarbeiten durchführen (vielleicht einen Datei-Update durchführen) und dann das Programm beenden.

A N H A N G

A

A. Error Codes

5	Prozeduraufruf oder Argument ungültig
6	Overflow (Überlauf)
7	Zu wenig Speicher
9	Index außerhalb des zulässigen Bereichs
10	Array unveränderlich oder temporär blockiert
11	Division durch Null
13	Datentyp passt nicht
14	Stringgröße überschritten
17	Anforderte Operation kann nicht ausgeführt werden
28	Stackbereich überschritten
35	Sub oder Funktion nicht definiert
48	Fehler beim Laden einer DLL
51	Interner Fehler
52	Falscher Dateiname oder falsche Dateinummer
53	Datei nicht gefunden
54	Falscher Dateimodus
55	Datei bereits geöffnet
57	Geräte I/O-Fehler
58	Datei existiert bereits
61	Disk voll
62	Eingabe hinter dem Ende der Datei
67	Zu viele Dateien
68	Gerät nicht verfügbar
70	Zugriff verweigert
71	Disk nicht bereit
74	Umbenennen in anderes Laufwerk nicht möglich
75	Pfad/Datei Zugriffsfehler
76	Pfad nicht gefunden
91	Objektvariable nicht gesetzt
92	For Schleife nicht initialisiert
94	Ungültige Verwendung von Null
322	Erforderliche temporäre Datei kann nicht erzeugt werden
424	Objekt erforderlich
429	ActiveX Komponente kann Objekt nicht erzeugen
430	Klasse unterstützt Automation nicht
432	Datei- oder Klassenname während Automation Operation nicht gefunden
438	Objekt unterstützt diese Eigenschaft oder Methode nicht

440	Automation Fehler
445	Objekt unterstützt diese Aktion nicht
446	Objekt unterstützt benannte Argumente nicht
447	Objekt unterstützt aktuelle Umgebungseinstellungen nicht
448	Benanntes Argument nicht gefunden
449	Argument nicht optional
450	Falsche Anzahl von Argumenten oder ungültige Eigenschaftszuweisung
451	Objekt ist keine Collection
453	Spezifizierte DLL Funktion nicht gefunden
455	Fehler beim Sperren der Code Ressource
457	Dieser Schlüssel ist bereits mit einem Element dieser Collection verbunden
458	Variable nutzt einen Automation-Typ der von VBScript nicht unterstützt wird
500	Variable ist nicht definiert
501	Illegale Anweisung
502	Objekt nicht sicher für Scripting
503	Objekt nicht sicher für Initialisierung
504	Objekt nicht sicher für Creating
505	Ungültige oder ungeeignete Referenz
506	Klasse nicht definiert
507	Eine Exception ist aufgetreten
32811	Element nicht gefunden
32812	Das spezifizierte Datum ist im aktuellen lokalen Kalender nicht verfügbar

A N H A N G

B

.....

B. Konstanten

Farbe	vbBLACK, vbRED, vbGREEN, vbBLUE, vbYELLOW, vbBLUE, vbMAGENTA, vbCYAN, vbWHITE
Vergleich	vbBINARYCOMPARE, vbTEXTCOMPARE
Datum/Zeit	vbSUNDAY, vbMONDAY, vbTUESDAY, vbWEDNESDAY, vbTHURSDAY, vbFRIDAY, vbSATURDAY, vbFIRSTJAN1, vbFIRSTFOURDAYS, vbFIRSTFULLWEEK, vbUSESYSTEM, vbUSESYSTEMDAYOFWEEK
Datumsformat	vbGENERALDATE, vbLONGDATE, vbSHORTDATE, vbLONGTIME, vbSHORTTIME
MSGBOX	<p>Display options (add to combine): vbOKONLY, vbOKCANCEL, vbABORTRETRYIGNORE, vbYESNOCANCEL, vbYESNO, vbRETRYCANCEL, vbCRITICAL, vbQUESTION, vbEXCLAMATION, vbINFORMATION, vbDEFAULTBUTTON1, vbDEFAULTBUTTON2, vbDEFAULTBUTTON3, vbDEFAULTBUTTON4, vbAPPLICATIONMODAL, vbSYSTEMMODAL</p> <p>Return values: vbOK, vbCANCEL, vbABORT, vbRETRY, vbIGNORE, vbYES, vbNO</p>
String	vbCR, vbCRLF, vbFORMFEED, vbLF, vbNEWLINE, vbNULLCHAR, vbTAB, vbNULLSTRING, vbVERTICALTAB
VARTYPE	vbEMPTY, vbNULL, vbINTEGER, vbLONG, vbSINGLE, vbDOUBLE, vbCURRENCY, vbDATE, vbSTRING, vbOBJECT, vbERROR, vbBOOLEAN, vbVARIANT, vbDATAOBJECT, vbDECIMAL, vbBYTE, vbARRAY
Global	CurrentPath, NSBVersion

'	189	EXIT	115
		FOR EACH...NEXT	121
		FOR...NEXT	120
&	23	FUNCTION	125
		IF...THEN...ELSE	133
" "		KILLFOCUS	143
—	51	ON ERROR	167
		OPTION EXPLICIT	168
ABS	66	PRINT	179
AddItem	153	RANDOMIZE	187
ADDOBJECT27, 67, 144, 169, 173		REDIM	188
Alignment	180	REM	189
AND	26, 70	RUNAPPATEVENT	196
Arithmetische Operatoren .24		RUNAPPATTIME	197
ARRAY	71	SELECT CASE	200
ASC	72	SENDKEY	202
ASCB	72	SET	203
ASCW	72	SETMENU	204
ATN	73	SETPARENT	206
Ausführen eines Programms	40	SHELLEXECUTE	208
Ausgabe-Fenster	33	SHOWFULLSCREEN	209
Ausgabe-Objekt	141	SHOWOKBUTTON	210
		SLEEP	212
BackColor	180	SUB	221
Befehle		UPDATESCREEN	234
ADDOBJECT	67	WAVEVOLUME	238
BREAK	74	WHILE...WEND	242
BYE	75	WITH	241
CALL	76	Beispielprogramme	17
CHAIN	77	Boolesche Operatoren	25
CLASS	82	BorderStyle	173, 180
CONST	86	Bottom	180
DECLARE	101	BREAK	43, 74
DIM	102	BYE	75
DO...LOOP	104		
DOEVENTS	103	CALL	76
ERASE	107	Caption	180
EXECUTE	113	CBOOL	87
EXECUTEGLOBAL	114	CBYTE	87

CCUR	87		
CDATE	87	Editieren eines Programms	
CDBL	87		38
CHAIN	77	Eigenschaften	
Change	111	Alignment	180
CheckBox	79	BackColor	180
CHR	81	BorderStyle	180
CHRB	81	Bottom	180
CHRW	81	Caption	180
CINT	87	Date	181
CLASS	82	Enabled	181
Clear	153	Encrypted	181
Click	111	ExpandedHeight ..	181
CLNG	87	FontBold	181
Cls	174	FontItalic	181
ComboBox	36, 83	FontName	181
CommandButton	85	FontSize	181
CONST	86	FontStrikethru	181
Copy	38	FontUnderline	181
COS	90	FontWeight	181
CSNG	87	ForeColor	181
CSTR	87	Group	182
CurrentPath	91	Height	182
Cut	38	Hwnd	182
		IntegralHeight	182
DATE	92, 93, 181	Left	182
DATEADD	94	List	182
DATEDIFF	95	ListCount	182
Datentyp Array	23	ListIndex	182
Datentyp Boolean	22	LongFormat	181
Datentyp Farbe	23	MaxLength	183
Datentyp String	23	MultiLine	183
DATEPART	97	NewIndex	183
DATESERIAL	98	ParentHWnd	183
DATEVALUE	99	Right	183
DAY	100	Scrollbars	184
DblClick	111	SelLength	184
Debuggen eines Programms		SelStart	184
	40	SelText	184
DECLARE	101	Sorted	184
DIM	102	Style	184
DO...LOOP	104	Tabstop	184
DOEVENTS	103	Tag	184
DrawCircle	174	Text	184
DrawLine	174	Timer	184
DrawPicture	174	Top	185
DrawPoint	174	Value	185
DrawText	174	Visible	185
DrawWidth	173	Width	185
DropDown	111	WindowLong	185
		Eigenschaften-Editor ..	36

Ein Programm erstellen	37	Formulare	27
Elemente	21	Frame	124
Enabled	181	FUNCTION	125
Encrypted	181	Funktionen	
EQV	26, 106	ABS	66
ERASE	107	ARRAY	71
Ereignisse		ASC	72
Change	111	ASCB	72
Click	111	ASCW	72
DbtClick	111	ATN	73
DropDown	111	CBOOL	87
GotFocus	111	CBYTE	87
Err	108	CCUR	87
error handler	248	CDATE	87
ESCAPE	109	CDBL	87
EVAL	110	CHR	81
EXECUTE	20, 43, 113	CHRB	81
EXECUTEGLOBAL	114	CHRW	81
EXIT	115	CINT	87
EXP	116	CLNG	87
ExpandedHeight	181	COS	90
		CSNG	87
FALSE	22	CSTR	87
FillColor	173	DATE	92
FillStyle	173	DATEADD	94
FILTER	117	DATEDIFF	95
Find	39	DATEPART	97
Find All	39	DATESERIAL	98
Find Next	39	DATEVALUE	99
firstdayofweek	95	DAY	100
firstweekofyear	95, 97	ESCAPE	109
FIX	119	EVAL	110
FontBold	181	EXP	116
FontItalic	181	FILTER	117
FontName	181	FIX	119
FontSize	181	FORMATCURRENCY	122
FontStrikethru	181	FORMATDATETIME	122
FontTransparent	173	FORMATNUMBER	122
FontUnderline	181	FORMATPERCENT	122
FontWeight	181	GETCOMMANDLINE	127
FOR EACH...NEXT	121	GETLOCALE	128
FOR...NEXT	120	GETREF	129
ForeColor	181	GETSERIALNUMBER	130
Form_close	172	HEX	131
Form_Load	29, 53	HOUR	132
Form_Unload	29, 53		
FORMATCURRENCY	122		
FORMATDATETIME	122		
FORMATNUMBER	122		
FORMATPERCENT	122		

IMP	134	TRIM	229
INPUTBOX	135	TYPENAME	230
INSTR	136	UBOUND	232
INSTRREV	136	UCASE	233
INT	137	UNESCAPE	109
ISARRAY	139	VARTYPE	235
ISDATE	139	WAITCURSOR	237
ISEMPTY	139	WEEKDAY	239
ISNULL	139	WEEKDAYNAME	240
ISNUMERIC	139	YEAR	244
ISOBJECT	139		
JOIN	140	GETCOMMANDLINE	127
LBOUND	145	GETLOCALE	128
LCASE	146	GETREF	129
LEFT	147	GETSERIALNUMBER	130
LEFTB	147	Globale Eigenschaften	
LEN	148	CurrentPath	91
LENB	148	KeyPreview	141
LOG	151	NSBVersion	165
LTRIM	152	Globale Variablen	
MID	155	KeyboardStatus ...	142
MIDB	155	KeyboardStatusChang	
MINUTE	156	ed	142
MONTH	158	GotFocus	111
MONTHNAME	159	Group	169, 182
MSGBOX	160		
NOW	164	Height	182
OCT	166	HEX	131
PLAYSOUND	177	Hide	153
REPLACE	190	HOUR	132
RGB	191	HScrollbar	198
RIGHT	192	Hwnd	182
RIGHTB	192		
RND	193	IF...THEN...ELSE	133
ROUND	194	IMP	26, 134
RTRIM	195	INPUTBOX	135
SECOND	199	Installation	19
SETLOCALE	128	INSTR	136
SGN	207	INSTRREV	136
SIN	211	INT	137
SPACE	213	IntegralHeight	182
SPECIALFOLDER	214	IS	138
SPLIT	216	ISARRAY	139
SQR	217	ISDATE	139
STRCOMP	218	ISEMPTY	139
STRING	219	ISNULL	139
STRREVERSE	220	ISNUMERIC	139
TAN	223	ISOBJECT	139
TIME	225		
TIMESERIAL	227		
TIMEVALUE	228		

JOIN.....	140	MouseDown	175
KeyboardStatus	142	MouseMove.....	175
KeyboardStatusChanged	142	MouseUp.....	175
KeyDown	111, 175	MSGBOX	41, 160
KeyPress	111, 175	MultiLine.....	183
KeyPreview	141	Name.....	183
KeyUp	111, 175	NewIndex	183
KEYWORD	21	NOT.....	26, 163
KILLFOCUS.....	143	NOW	164
Klammern	26	NSBVersion.....	165
		Numerische Datentypen	22
LBOUND.....	145	Objekte	
LCASE.....	146	ADDOBJECT	144,
LEFT	147	169, 173	
LEFTB.....	147	CheckBox.....	79
LEN	148	ComboBox	83
LENB	148	CommandButton....	85
license.....	67	Date	93
List	182	Err	108
ListBox	36, 149	Frame.....	124
ListCount.....	182	HScrollbar	198
ListIndex	182	ListBox	149
Literale.....	22	OUTPUT	172
LOG	151	TextBox.....	224
LongFormat	181	Time	226
LostFocus	111	TriButton	79
LTRIM	152	VScrollBar	198
		OCT.....	166
MaxLength	183	ON ERROR.....	167, 247
Mehrzeilige Anweisungen	21	Operatoren	24
Menü-Editor	36	AND	70
Menüs	31	EQV	106
Methoden.....	27	IS 138	
AddItem	153	MOD.....	157
Clear	153	NOT	163
Hide	153	OR.....	171
Move.....	153	XOR	243
RemoveItem	153	OPTION EXPLICIT	43, 168,
SetFocus	153	247	
Show.....	154	OR.....	26, 171
MID	155	OUTPUT	172
MIDB.....	155	Output_Close	172
MINUTE	156	Output_Close Ereignis	30
MOD	25, 157	Output_Size	30, 172
Module.....	27, 30	Output-Objekt.....	30
MONTH.....	158	Overview (Übersicht)..	39
MONTHNAME	159		

<i>parent</i>	67	SETPARENT.....	206
ParentHWND	183	SetScale.....	174
Paste.....	38	SGN	207
Picture.....	173	SHELLEXECUTE.....	208
PictureBox	27, 30	Show	154
PLAYSOUND.....	177	SHOWFULLSCREEN.....	209
PRINT	20, 41, 179	ShowOKButton... 30, 210	
PRIVATE	86	SIN	211
Programm-Editor	31	SLEEP.....	212
Programmierrichtlinien.....	245	Sorted.....	184
Projekte.....	27	SPACE	213
PUBLIC.....	86	SPECIALFOLDER ...	214
		Speichern und Laden eines Programms	44
RANDOMIZE	187	SPLIT	216
REDIM	188	SQR	217
Refresh	174	Step	43
REM.....	189	STRCOMP	218
RemoveItem	153	STRING.....	219
REPLACE	190	STRREVERSE.....	220
RGB	191	Style	184
Richtlinien zur Namensgebung ...	245	SUB	221
Right	183, 192	System Anforderungen.....	19
RIGHTB	192		
RND	193	Tabstop	184
ROUND.....	194	Tag	174, 184
RTRIM	195	TAN	223
RUNAPPATEVENT ..	196	Text	184
RUNAPPATTIME.....	197	TextBox	224
		TextHeight.....	174
ScaleHeight	173	TextWidth	175
ScaleLeft.....	173	TIME.....	225, 226
ScaleMode.....	174	Timer.....	111, 184
ScaleTop.....	174	TIMESERIAL.....	227
ScaleWidth.....	174	TIMEVALUE.....	228
ScaleX	174	Top	185
ScaleY	174	Trace	43
scrollbar	198	TriButton.....	79
Scrollbars	184	TRIM	229
SECOND	199	TRUE	22
SELECT CASE	200	TYPENAME	230
SelLength.....	184	typografische Konventionen	20
SelStart	184		
SelText.....	184		
SENDKEY	202	UBOUND.....	232
Seriennummer	19	UCASE.....	233
SET	25, 203	UNESCAPE	109
SetFocus.....	153	UPDATESCREEN.....	234
SETLOCALE.....	128		
SETMENU	204		

Value.....	185	WEEKDAYNAME.....	240
Variablen.....	22	WHILE...WEND.....	242
Variablenamen.....	23	Width.....	185
VARTYPE	235, 247	WindowLong	185
Vergleichs-Operatoren	25	WITH.....	241
Visible	185		
VScrollBar	198	XOR	26, 243
WAITCURSOR	237	YEAR	244
WAVEVOLUME	238		
WEEKDAY	239		

FORMULAR FÜR ANWENDERKOMMENTARE

Bitte benutzen Sie dieses Formular nur, um uns Fehler oder Änderungswünsche zum Handbuch mitzuteilen. Geben Sie bitte an, ob Sie eine Rückmeldung wünschen. Bitte senden an:

NS BASIC Corporation
71 Hill Crescent
Toronto, Canada M1M 1J3
fax (416) 264-5888

Eine alternative Möglichkeit, NS Basic bezüglich Publikationsfehlern oder Änderungswünschen zu kontaktieren ist uns ein Email zu senden an:

support@nsbasic.com

Im Betreff sollte der Titel der Publikation gefolgt vom Druckdatum (in der Fußzeile der ersten Seite) stehen.

Seite	Kommentar