



**NS Basic/CE ハンドブック**

**December 1, 2006**

© NS BASIC Corporation, 2006.  
71 Hill Crescent  
Toronto, Canada M1M 1J3  
+001 (416) 264-5999

本マニュアル及び記述されるソフトウェアは著作権を持つ資料であり、著作権法により守られています。著作権法の下で、本マニュアルはNS BASIC Corporationからの文書での同意なしで、全体または一部をコピーすることは出来ません。法の下では、コピーとは別の言語か形式に翻訳するのを含みます。

本マニュアルの情報が確実に正確になるよう、あらゆる努力がされています。NS BASIC Corporationは印刷または事務員の誤りによる責任を負いません。仕様は予告なしで変更をすることを必要とします。

Microsoft、MS、Windows、およびWindowsCEは、合衆国および他国で登録されるマイクロソフト社の登録商標です。

第三者の製品とそれらの商標の言及は情報の目的だけのためにあって、支持や推薦は伴いません。NS BASIC Corporationは、NS BASICがこれらの製品の性能または使用に関して、どんな責任も負いません。

#### **Canadian Cataloguing In Publication Data**

Darden, Marcus M., 1970-  
NS BASIC/CE Handbook

Includes index.  
ISBN 0-9695844-4-X

1. 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

## 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.

#### 使用許諾契約\*1

本ソフトウェアを使用する前に、本使用許諾を注意してお読み下さい。本ソフトウェアを使用することで、本使用許諾の条項に同意することになります。本使用許諾の条項に同意しない場合は、即座に製品を購入した所に返品して頂ければ、返金させていただきます。

1. 使用許諾：本使用許諾に伴うアプリケーション、デモンストレーション、システム、および他のソフトウェア（読み込み専用メモリ、またはディスクか、いかなる他の媒体（「ソフトウェア」）にかかわらず）に関連する文書と書体はNS BASIC Corporation（「NSBC」）によって認可されます。ユーザーは本ソフトウェアと書体が記録される媒体を所有しますが、NSBCと（や）NSBCの使用許諾者が文書や書体に関連するソフトウェアの権利を保有します。この使用許諾によって単一のWindowsCE Product（本使用許諾の目的である製品、ペーシングマイクrosoftのWindowsCEロゴを意味する）で本ソフトウェアと書体を使用することができます。また、バックアップ目的でのみ機械可読なフォームでソフトウェアと書体の1部のコピーを作ることが出来ます。それらのコピーには、本ソフトウェアと書体のオリジナルのコピーにあったNSBC版権情報といかなる他の所有される説明文も再生されなければなりません。また、本ソフトウェアと書体、それらのバックアップコピー、そして本使用許諾に関連する文書とコピーを提供して、本使用許諾に関する条件で他人が受け入れに同意する場合、あなたのすべての使用許諾権利を譲渡することができます。

2. 制限：本ソフトウェアは著作権資料、企業秘密および他の所有される資料を含み、それらの保護の為、ユーザーは本ソフトウェアを逆コンパイルしたり、リパースエンジニアしたり、分解したり、もしくは人間知覚できるフォームに変えたりすることはできません。全体か一部ソフトウェアに基づいて派生している作品を変更、ネットワークに接続、貸借、賃貸、ロード、分配、または作成することができません。ユーザーは電子的に、もしくはネットワークを通して本ソフトウェアを1台の装置から別の装置に送ることはできません。

3. 契約期間：本使用許諾契約は終了まで有効です。ユーザーは随時、本ソフトウェア、関連する文書および書体を破壊することによって、このソフトウェアを終えることができます。ユーザーが本使用許諾のどんな規定にも従わない時、本契約はNSBCからの予告なしで即刻終了します。終了時、ユーザーは本ソフトウェア、関連する文書および書体を破壊しなければなりません。

4. 法保証の輸出：ユーザーは、合衆国の法と規則によって同意、公認されたものを除き、本ソフトウェアと、NSBCから受け取ったいかなる技術的なデータ、および原物製品も、合衆国外へ輸出しないことに同意し、保証します。このソフトウェアを合衆国外で正しい方法により入手した場合、ユーザーは本ソフトウェア、NSBCから受け取ったいかなる他の技術的なデータ、原物製品を、合衆国

の法と規則、本ソフトウェアを入手した司法権の法および規則によって許可を得た場合を除いて、再輸出しないことに同意します。

5. 政府のエンドユーザー：ユーザーが合衆国政府のどんなユニットか代理店でもそれを代表して本ソフトウェアと書体を取得している場合、以下の条項が適用されます。政府は以下の事に同意しています：(i)本ソフトウェアと書体が国防総省(DoD)に供給される場合、本ソフトウェアと書体は「市販のコンピュータソフトウェア」に分類される。政府は本ソフトウェアの「制限された権利」だけを取得している。その条件での文書と書体はDFARSのClause 252.227-7013(c)(1)で定義される。(ii)本ソフトウェアと書体がDoD以外の合衆国政府のいかなるユニットか代理店に供給される場合でも、ソフトウェアの政府の権利を除いて、その文書と書体がFARのClause 52.227-19(c)(2)で定義される。NASAの場合、FARへのNASAの増補のClause 18-52.227-86(d)で定義される。
6. NS BASICは、購買日付の90日以内にディスクかマニュアルに欠陥が見つかった場合、無料でお取り替えます。NS BASICは、含まれている文書に従って一般にプログラムが実行する事を保証します。NS BASICは、プログラムとマニュアルがすべてのバグ、誤りまたは省略がないという保証はできません。
7. ソフトウェアの保証の放棄：ユーザーは自身の責任で本ソフトウェアと書体を使用することを承認して同意します。文書、書体、および本ソフトウェアは、「現状渡し」で供給され、どの種の保証も含まず、NSBC及びNSBCの使用許諾者（条項8及び条項9の為に、NSBC及びNSBCの使用許諾者はまとめて"NSBC"として扱う）は、制限されていない特殊な目的用に市場向けの保証を示唆する全ての保証、表現または含意を放棄します。NSBCは本ソフトウェアに含まれる機能がユーザーの要求を満たすことや、誤りや障害が起きないことや、ソフトウェアや書体の欠陥を修正する事を保証しません。ソフトウェアの正当性、精度、信頼性の観点から、本ソフトウェア、書体および関連文書の使用やその結果に関して、陳述することやまたはそれを保証することはしません。NSBCがNSBC承認の代表による口頭または文書での情報では、何も保証されないし、現保証を増やすこともありません。仮に本ソフトウェアが欠陥があると判明しても、ユーザー(どんなNSBC、NSBCの代表でもない)はすべての必要な整備点検、修理または修正の全体の費用を負います。いくつかの司法は黙示的な保証の除外を認めないので、上記の除外はユーザーに適用されない。
8. 責任の制限：ソフトウェアは本来複雑で誤りを含む可能性がある為、このプログラムによる仕事を確認するよう忠告します。怠慢な状態も含み、たとえNSBCやNSBCの認可された代表がそのような損害賠償の可能性を指摘されたとしても、NSBCには偶然、特別もしくは必然の本ソフトウェアや関連する文書の使用による損害に対して責任がありません。いくつかの管轄区域は、偶然や必然の損害に対する責任の制限や除外を許していないので、上の制限や除外はユーザーに適用されない。どんな事件も、全ての損害、損失、及び行動の原因（契約にあるうとなかろうと、民事犯（怠慢を含む）もしくはそれ以外）に対するNSBCの総責任は、あなたが支払った本ソフトウェアと書体の額を越えないものとする。
9. リスクの配分：ユーザーは、この契約がユニフォーム商法と他の適切な法によって認可されるユーザーとNSBCの間のリスクを割り当て、NSBCの製品の値付けがそのリスクの配分と本契約に含まれる責任の制限を反映する事を承認して同意します。何か以下の救済法がその本質的意図の失敗と判断された場合、本契約に詳しく説明される責任の全ての制限と損害賠償の除外は有効なままで残ります。
10. サポート：NSBCはオプションとして、標準料金でサポートサービスを提供する事ができます。そのようなサポートサービスは本契約の下で責任の制限によって管理されます。
11. 追加制限：NSBCによって供給されたプログラムのどんなアップグレードや増進も先のバージョンの破壊のときに使用される事ができ、本契約の条件において定められるものとします。
12. 法の制御及び断絶の可能性：この使用許諾はデラウェアに結ばれた協定と、デラウェアの居住者の間での実行が適用されるように、合衆国とデラウェア州の法に従って治められ、解釈されます。いかなる理由でも、十分な司法権の法廷がこの使用許諾のどんな規定やその部分を見つける場合、「非・実施・可能」になるように、使用許諾の規定はパーティーの意図に作用するように最大の範囲に許された状態で実施されるものとし、本使用許諾の残りは十分な効力と効果を持ち続けるものとします。
13. 協定の終了：本使用許諾は、本ソフトウェア、関連する文書や書体の使用、その内容に関するすべての先行した、もしくは同時性の、文章か口頭での疎通や協定、の観点に於いての関係者間の全契約により構成されます。この使用許諾のいかなる修正および変更は、NSBCの正しく認可された代表による文章とサインがない限り拘束されず。

\*1  
日本語版使用許諾契約は情報目的のみの為提供されており、法的目的の為に英語版使用許諾契約（License Agreement）が適用されます。

# C O N T E N T S

1. 製品ガイド .....	9
1.1 BASICとは? .....	9
1.1.1 NS Basic/CE .....	10
1.1.2 NS Basic/CE と Windows CE .....	10
1.2 動作環境 .....	11
1.3 インストール .....	11
1.3.1 シリアル番号の入力 .....	11
2. NS Basic/CEのコンセプト .....	13
2.1 このハンドブックで使用される表記 .....	13
2.2 NS Basic/CEプログラムのエレメント .....	14
2.2.1 複数行ステートメント .....	14
2.2.2 一行における多重ステートメント .....	15
2.2.3 リテラル、データタイプおよび変数 .....	15
2.2.4 数値データ型 .....	15
2.2.5 ブールデータ型(Boolean Data Type) .....	15
2.2.6 色データ型 (Color Data Type) .....	16
2.2.7 文字列データ型 (String Data Type) .....	16
2.2.8 配列データ型 (Array Data Type) .....	16
2.2.9 変数名(Variable Names) .....	16
2.3 式と演算子(Expressions and Operators) .....	17
2.3.1 算術演算子(Arithmetic Operators) .....	17
2.3.2 関係演算子(Relational Operators) .....	18
2.3.3 論理演算子(Boolean Operators) .....	18
2.4 FUNCTIONおよびSUBプロシージャ .....	19
2.5 プロジェクト、モジュール、フォーム、オブジェク ト、コントロール .....	20
コントロールとオブジェクト .....	20
オブジェクトイベント .....	20
フォーム .....	21
フォーム メソッド .....	21
フォーム イベント .....	21
プログラムのプロパティとイベント .....	22
プログラムイベント .....	22
モジュール .....	22
3. デバイス上でのプログラミング .....	23
3.1 概要 .....	23
3.2 プログラムエディタ .....	23
3.3 アウトプット ウィンドウ .....	25
3.4 ビジュアルデザイナー (Visual Designer) .....	26
3.4.1 プロパティエディタ (Property Editor) .....	29
3.4.2 メニューエディタ (Menu Editor) .....	29
3.4.3 注記 .....	29
3.5 NS Basic/CEプログラミング環境 .....	30
3.5.1 プログラムの作成 .....	30
3.5.2 プログラムの編集 .....	31
3.5.3 プログラムのフォーマット .....	31
3.5.4 検索及び置換え .....	31
3.5.5 概要 (Overview) .....	32
3.5.6 プログラムの実行 .....	33

3.5.7 プログラムのデバッグ .....	34
3.5.8 BREAK、Execute Function、Trace、Step .....	36
3.5.9 プログラムの保存と読み込み .....	37
3.5.10 テキストファイルとしてプログラムの保存と 読み込み .....	38
4. デスクトップ上でプログラミング .....	39
4.1 メニュー .....	39
4.2 コードウィンドウ (Code Window) .....	42
4.3 CE Screen .....	43
4.4 プロジェクトエクスプローラ (Project Explorer) .....	45
4.5 プロパティウィンドウ (Properties Window) .....	46
4.6 ツールボックス (Toolbox) .....	47
4.7 ツールバー (Toolbar) .....	48
4.8 メニューエディタ (Menu Editor) .....	49
4.9 オプション (Options) .....	51
4.9.1 オプション - 一般 (General) .....	51
4.9.2 オプション - エディタ (Editor) .....	52
4.9.3 オプション - CE Screen .....	53
4.9.4 オプション - Start .....	54
5. NS Basic/CE リファレンス .....	55
6. アドバンストピック (Advanced Topics) .....	237
6.1 プログラミング慣習 .....	237
6.1.1 命名のガイドライン .....	237
6.1.2 テキスト形式のガイドライン .....	238
6.1.3 コメントのガイドライン .....	238
6.2 エラー処理 .....	238
6.2.1 防衛的なプログラミング .....	239
6.2.2 エラートラッピング .....	239
6.2.3 エラートラッピング ファイルオペレーション .....	240
A. エラーコード .....	241
B. 定数 .....	243
索引 .....	245



## 1. 製品ガイド

NS Basic/CEをお買い上げ頂きありがとうございます。NS Basic/CEはWindows CEのお客様のニーズに応えるよう設計されました。このシンプルでありながらパワフルなプログラミング言語は、ほぼ全てのアプリケーション作成にご使用頂けます。

本製品に付属のCDに収められているReadme.txtに、ハンドブックの更新等、最新情報が含まれています。NS Basic/CEのインストール前にお読み下さい。

すぐに本製品をご使用されたい場合は、インストールセクションを読み、Chapter2「NS Basic/CEを始める」へ進んで下さい。

機能の使い方の参考の為にサンプルプログラムがNS Basic/CEとともに提供されています。必要に応じてこれらのサンプルプログラムは変更して結構です。インストールフォルダ内の「BASIC Samples」と名付けられたフォルダーにこれらのサンプルがインストールされます。

本ハンドブックを使用し始める前にWindows CEデバイスの基本操作に多少詳しくなければなりません。例えば、スタイルスと他のWindows/CEの機能を使用した、アプリケーションの起動を知らなければなりません。機能を確認したい場合は、Windows CEのマニュアル等を見直してください。

デスクトップコンピュータ(Windows 98/NT/2K/XP)を操作する基本的な理解が、NS Basic/CEソフトウェアをインストールするのに必要です。

### 1.1 BASICとは？

BASICは、ほぼ40年前に作られました。それ以来、BASICのための何百ものインタプリタとコンパイラが開発され、山のようなアプリケーションコードが書かれてきました。この言語に関しての多くの本が、発行され続けています。BASIC Special Interest Groupsは多くの形で存在します。

BASICはなぜか精神に良いのです。新たな言語の波の行き来の中、BASICはまだほとんどの場所で動いています：規格がなければ、容易に新しい環境に慣れて、意匠をこらした新しい言語と足並をそろえます。

皆、あのビル・ゲイツでさえ、BASICから始めました。我々は皆、何度もその原点に戻ります。BASICは迅速にプログラムを書くには最も良い言語です。また、滑らかな学習プロセスによって、初心者から上級プログラマーへと進むのが容易です。

BASICがプログラムされるコンピュータハードウェアはそれが開発された時以来、様々な形となって広く使用されてきました。コンピュータ科学者とメインフレームプログラマーだけがアクセスすることができた強力な言語を今、携帯用のデバイス上で動かすことができます。

#### **1.1.1 NS Basic/CE**

NS Basic/CEはWindows CE用のプログラミング言語です。率直に言うと、それはすべての一般的に使用されるBASICステートメントを実行し、また多くの強力な拡張子を持っています。

NS Basic Corporationは<http://www.nsbasic.com>にウェブサイトを持っています。重要な発表、技術的な情報およびNS Basic/CEサンプルプログラム等についてはウェブサイトを確認して下さい。

#### **1.1.2 NS Basic/CE と Windows CE**

お客様がWindows CEデバイスを買った時、きっとそれがペーパー作業の多くに取って代えることができると思ったことでしょう。また、それが小さなプログラム可能なコンピュータとして機能することができることを望んでいたかもしれません。NS Basic/CEはこれらの目的のために設計されました。この使い易い言語を利用して、Handheld PC上で必要なアプリケーションを作成することができます。

また、汎用のプログラミングにNS Basic/CEを使用することもできます。BASICで作成出来るどんなプログラムもNS Basic/CEで作成することができます。カスタム設計されたデータベースを作成したり、複雑な計算を実行したり、またはゲームを作ることもできます。NS Basic/CEが他と違うのは、その入り易さです。H/PCが持つ強力な機能を利用するためだけに、複雑な新しい言語を学ぶ必要はありません。

## 1.2 動作環境

NS Basic/CEをインストールするために、Windows CEデバイス、Microsoft ActiveSyncをインストールしてあるデスクトップコンピュータ、およびWindows CEデバイスをデスクトップコンピュータに接続するためのケーブルが必要になります。サポートされているデバイスと必要なソフトウェアに関する情報は、本製品に含まれるReadme.htmファイルで確認してください。

## 1.3 インストール

NS Basic/CEはCDに収められています。Microsoft ActiveSyncを使って、お使いのWindows CEデバイスにインストールしなくてはなりません。製品に含まれるReadme.htmファイルで、インストール手順を確認して下さい。

また、当社のNS Basic/CEに関する最新ニュースを掲示するwebサイトをぜひ訪問してください。Release Notesにてアップデート情報を、テクニカルノート（Tech Notes）にて最新情報を、またダウンロードセクションにて追加サンプルコード、アップデーター、Runtimeモジュールの情報を確認してください。

当社ウェブサイトのアドレスは<http://www.nsbasic.com>です。

### 1.3.1 シリアル番号の入力

NS Basic/CEを使用する前に、BASIC Installerプログラムを使用して自分のシリアル番号を入力しなければなりません。デバイス上での登録は、BASIC Installerプログラムを起動して下さい。次にデスクトップ上での登録は、"Help"メニューの下にあるRegisterから行って下さい。シリアル番号はハンドブックの裏面に記載されています。インストールに関する追加情報をReadme.htmファイルで確認して下さい。

ユーザ登録を行うには、登録カードを記入し、カードに印刷されているアドレスに送るか、もしくは当社のウェブサイトでオンライン登録してください。



## 2. NS Basic/CEのコンセプト

### 2.1 このハンドブックで使用される表記

このハンドブックでは以下の表記が使用されています:

#### KEYWORDS

アルファベットの大文字で示される語句は、NS Basic/CEのキーワードを意味します。本マニュアルでは、大文字はステートメントシンタックスの必要な部分であることを強調しています。NS Basic/CEは大文字／小文字の認識がないのでキーワードは大文字でも小文字でも、または混ぜても構いません。例えばキーワードのPRINTはprint、Print、PRINTのどれでも結果は一緒です。

#### *placeholders*

斜体の語句は、あるタイプのインフォメーションが置かれる位置を示し、ユーザーが提供しなければなりません。以下のEXECUTEステートメントはexpression (式) が必要であることを示しています:

EXECUTE *expression*

#### Example

このMonaco書体は、サンプルプログラムのコードと、スクリーンへの出力であることを示します。以下はNS Basic/CEプログラムの例です:

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

#### [Optional]

箱括弧はアイテムがオプションであることを意味します。以下の例は、PRINTステートメントを使ってスクリーンへプリントする時に、二つ目のアイテムを指定するかはオプションであることを意味します:

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

PRINTステートメントは20個までの式が同時に書けるので、以下のどちらのPRINTステートメントも有効です：

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

|

縦棒はアイテムが排他的な関係にあることを示します。以下の例はLENファンクションはstringかvariableのどちらか一つだけしか使えないことを示します：

LEN(string | variable)

## 2.2 NS Basic/CEプログラムのエレメント

NS Basic/CEのプログラムはステートメントのセットです。以下のエレメントはNS Basic/CEプログラムの各ラインを形成するのに使われます。

KEYWORD arguments 'comment

KEYWORDはNS Basic/CEが理解する言語からの単語です。例えば、PRINT、INPUTBOX、IF等です。ステートメントとそのarguments（引数、パラメータ、コンポーネント等の必要な値）は、ラインが実行される時、（もしあるならば）どんな行動がNS Basic/CEによって取られるのかを決定します。

ライン上で引用符 ' に続くテキストはコメント（注釈）ですので、NS Basic/CEによって無視されます。

### 2.2.1 複数行ステートメント

通常、それぞれのNS Basic/CEステートメントは行の最後で終わります。複雑なステートメントの場合、一行が非常に長くなることがあります。これはプログラムを読みにくくします。スペースに続くアンダースコア( \_ )を行の終わりにつけるライン継続シーケンスを使用することによって、長いステートメントを複数行に分けることが出来ます。行が( \_ )で終わる場合、NS Basic/CEはその行を次の行と結合します。以下はライン継続の例です。

```
PRINT "Long statements are no problem for" _  
      & " NS Basic/CE"
```

### 2.2.2 一行における多重ステートメント

コロン(:)を分離符として使用すると、複数のステートメントが一行に入られます。

```
a=1: b=2: c=3
```

### 2.2.3 リテラル、データタイプおよび変数

リテラル（即値データ）はプログラム内で使用する値です。変数の初期値を設定したり、FOR...NEXTループの始めと終わりの値を設定するなど、頻繁に使用します。CONSTステートメントを使用して定数を定義すると、この定数をリテラルとして扱うことができます。この場合、このリテラル値は変更できません。

変数はデータを保管するための名前の付いた箱です。変数の値は必要に応じて変えられます。全ての変数はデータタイプを持っています。

### 2.2.4 数値データ型

数値データにはいくつかのサブタイプがありますが、動作の面から見てすべて同じように扱われます。一般的に、数値型同士でデータを混ぜ合わせることは、難しくありません。

サブタイプ	サイズ	範囲	リテラル
Byte	8 bits	0 to 255	2
Int	16 bits	-32,768 to 32,767	100
Long	32 bits	-2,147,483,648 to 2,147,483,647	500000
Single	32 bits	$\pm 1.5 \times 10^{-45}$ to $\pm 3.4 \times 10^{38}$	1.0e100
Double	64 bits	$\pm 5.0 \times 10^{-324}$ to $\pm 1.7 \times 10^{308}$	1.0e1000
Currency	32 bits	$-9.2 \times 10^{14}$ to $9.2 \times 10^{14}$	1000000.00
Hex	32 bits		&h000000F

### 2.2.5 ブールデータ型(Boolean Data Type)

ブール型は2つの値、真(TRUE)および偽(FALSE)、から成ります。このデータ型はIFステートメントで使用されます。このステートメントは与えられるブール値によって、TRUEならばTHENステートメントを選択し、FALSEならばELSE

ステートメント選びます。TRUEに相当する数値は -1  
で、FALSEには数値 0 が相当します。

### 2.2.6 色データ型 (Color Data Type)

色は0から16,777,215までの無符号の長整数 (long) です。  
色の値は、0~255(0 : 暗い、255 : 明るい)の値を持つ赤、  
緑、青が以下の公式に入り計算されます。 混合公式は、

$$\text{color} = \text{red} + (\text{green} * 256) + (\text{blue} * 65536)$$

赤、緑、青の値が同じ場合、色はグレーになります。  
Appendix Bに主な色の定数をリストしてありますので参照  
して下さい。

色	Red, Green, Blue	色の値
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 文字列データ型 (String Data Type)

文字列は一連のキャラクター (文字) から成ります。文字列  
は最高でおよそ20億文字が入ります。文字列を操作する多  
くのファンクション (関数) が用意されています。連結オペ  
レータ(&)は、2つの文字列型持った変数を結合させるのに使  
用されます。文字列リテラルは引用符で囲まれます。

"This is a string literal"

### 2.2.8 配列データ型 (Array Data Type)

配列は容器であり、一つの名前にて格納される値のリストで  
す。配列の各要素は、変数名の後の括弧に数 (インデックス  
/添字) を含むことによって参照できます。配列の最初の要  
素は常にインデックスがゼロ(0)です。各配列は多くの要素  
を持つことができます。例えば、ARR(2)はARR配列におけ  
る3番目の要素を示します。配列の要素はどんなデータタイ  
プでも構いません。

### 2.2.9 変数名 (Variable Names)

変数は値を保持する名前です。この名前はアルファベット、  
数字、アンダースコア(\_)の並びから構成できますが、必ず  
アルファベット文字から始まります。NS Basic/CEの変数名  
の長さには制限はなく、名前の中の全ての文字は意味を持ち  
ます。特にこれを言及する理由は、古いBASIC言語では短  
い名前のみを使用するようになっていた為です。変数名は格  
(大文字/小文字) による区別は認識しません。またスベ  
ースや他の特殊文字および、NS Basic/CE定数を変数名とし  
て使用することは出来ません。定数のリストは、Appendix B  
を参照して下さい。



以下のリストはNS Basic/CEによって許容されるいくつかの変数名の例：

```
text
LLAMAS           // llamas や Llamas でも同じ
Jupiter
WISpec
SouthPark
```

許容されない例：

```
1table           // 数字で始まる
X&Ycords         // 特殊文字(&)の使用
first counter    // スペースの挿入
%correct         // アルファベット以外の文字で始まる
print            // NS Basic/CEキーワード
```

2.3 式と演算子(Expressions and Operators)

式は値を持つリテラル、変数、公式、またはFUNCTIONプロシージャコールです。ここに、式のいくつかの例を挙げます。

```
6/3              // result is 2
5+6/3            // result is 7
"This " & "that" // result is "This that"
```

文字列式は文字列リテラル、文字列変数、またはサブ文字列を含んだこれらの組み合わせです。同様に、数値式は数値定数、数値変数、または一つの数値を生み出すファンクション/変数などです。本ハンドブックでは、必要なパラメータ値のタイプを数値型などと記載しておりますが、これは数値式等の意味であり、そのデータタイプが必要であることを意味しています。

2.3.1 算術演算子(Arithmetic Operators)

NS Basic/CEでは以下の算術演算子が使えます。リストの下に向かって優先度が下がります。

演算子		オペレーション内容
( )		括弧
^		指数
*	/ \	乗法および除法
+	-	加法および減法

括弧は計算する順序を変えるために使用できます。

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

は以下の結果を表示する

```
14  
20
```

NS Basic/CEでは浮動小数点の演算を行えます。MOD関数は、割算の余りを得るのに使用されます。バックスラッシュ演算子 (\) は整数の割算に使用されます。

算術演算子は数値型と共にだけ使用することができます。文字列型と一緒に使用出来ません。

### 2.3.2 関係演算子(Relational Operators)

関係演算子は2つの値を比較して、TRUEかFALSEのブール値を返します。この結果を使用することにより、プログラムのフロー（流れ）を変えることが出来ます。関係演算子の優先度は、算術演算子より低いです。関係演算子は以下の通りです：

演算子	オペレーション内容
=	等しい（イコール）
<>	等しくない
<	小なり
>	大なり
<=	小なりまたはイコール
>=	大なりまたはイコール

SETステートメントでは、変数に値を代入するのにイコールサイン(=)が使用されますが、これは関係演算子ではありません。

### 2.3.3 論理演算子(Boolean Operators)

論理演算子は式を評価し、TRUE（真）かFALSE（偽）の結果を返します。算術演算子および関係演算子は論理演算子より先に評価／計算されます。NOT演算子は1つの式に適用しますが、他の演算子は2つの式を必要とします。

論理演算子の内容は

演算子	オペレーション内容
AND	二つの式がTRUEの時、TRUEを返す。
EQV	二つの式が共にTRUE、もしくは共にFALSEの時、TRUEを返す。
IMP	はじめの式が二つ目の式を意味する時、TRUEを返す。
OR	どちらか一つの式か、二つの式がTRUEの時、TRUEを返す。
NOT	式がFALSEの時TRUEを、TRUEの時FALSEを返す。
XOR	二つの式が反対の時のみ、TRUEを返す。

論理演算子は、ブール値を返す式と共に使用することが出来ます。

## 2.4 FUNCTIONおよびSUBプロシージャ

プロシージャはプログラムステートメントの固まりであり、プログラムにおける他のステートメントから任意の値(引数)と共に呼ばれる事で、条件付きで繰り返し実行することが出来ます。プロシージャが呼ばれるとき、それはNS Basic/CE言語を作るファンクションやステートメントのように動作します。FUNCTIONプロシージャは1つの値を返します。その値は変数に代入するか、別の式で使います。SUBプロシージャは値を戻しません。FUNCTIONプロシージャが呼ばれて、戻り値がない場合、NS Basic/CEはSUBプロシージャとしてそれを実行します。

FUNCTIONプロシージャで複数の引数を渡すには、括弧内にカンマで区切られたリストを使用します。SUBプロシージャで複数の引数を渡すには、カンマで区切られたリストを括弧なしで付加します。単一の引数がFUNCTIONかSUBプロシージャで渡される時、括弧を使うことができます。NS Basic/CEでは、式を評価する為に、括弧を使用し単一の引数を括りますが、引数リストを示す為には使いません。

## 2.5 プロジェクト、モジュール、フォーム、オブジェクト、コントロール

### コントロールとオブジェクト

コントロールは実行可能なモジュールであり、他のプログラムによってアクセスすることができます。プログラム内に置かれたコントロールの各インスタンス（実体）をオブジェクトと呼びます。プログラムにオブジェクト追加するには、ADDOBJECTステートメントを使用します。オブジェクトはプロパティ（参照や設定可能）、メソッド（FUNCTIONかSUBプロシージャとして呼ぶことができる）、及びイベント（ユーザからの動作、オペレーティングシステムまたは他のプログラムによって引き起こされる）を持ちます。

数多くのコントロールがNS Basic/CEと共にインストールされます。これらはチェックボックス（checkbox）、コンボボックス（combobox）、テキストボックス（textbox）、および他の一般オブジェクトを含みます。これらは他のファイルに頼ることがなく、しかしNS Basic Runtimeに直接含まれているため、時に固有のオブジェクトとして考えられます。

コントロールの2つ目のグループは、標準コントロールと呼ばれ、NS Basicのインストールに含まれています。これらはMicrosoft社によって提供されているActiveXコントロール（PictureBox、Grid、FileSystem、等）です。

サードパーティのActiveXコントロールもNS Basic/CEと共に使用できます。これらの多くはNS BasicのBig Red Toolboxに収められます。Tech Note 1をご覧ください。またウェブ上で検索されることもお勧めします。

NS Basic/CEのインストールはPictureBoxオブジェクトを含んでいます。プログラム画面のバックグラウンド（Outputオブジェクト）はPictureBoxです。Outputオブジェクトに加え、Errオブジェクトが全ての実行中プログラムに自動的に作成されます。これはエラー処理に使うことができます。

### オブジェクトイベント

イベントは、オブジェクト内で起こるアクションの結果として、あなたのプログラムへ対して行う呼び出しです。例えばプログラム中にcommandbuttonがあるとします。ユーザがボタンをタップした時、イベントが発生します。ボタンの名前が“MyButton”の場合、NS Basicはプログラム中にあるMyButton\_Click()というファンクションを呼びます。もしこの名前のファンクションが存在しない時、イベントは無視されます。

別の例としては、シリアル通信を行うことです。入ってくるデータはOnCommイベントを発生させます。通信オブジェ

クトを"Comm"とした場合、Comm\_OnComm()というファンクションが呼ばれます。

### フォーム

NS Basic/CEのフォームの実行は、デスクトップのものとは比べ単純です。下層エンジンは、フォームの概念を持っていませんので、機能を提供するために、あるテクニックを使用します。

一旦オブジェクトが作成されると、その逆を行うことはできません。Windows CEでは、これは問題になりません。オブジェクトによる負荷は少ないため、プログラムが終了するまで、保持していても問題ありません。

従って、フォームは同時に現れたり、隠れたりするオブジェクトのグループです。フォームの各オブジェクトへの参照は配列に入れておくこともできます。この配列の各要素を移動しながら、各オブジェクトを隠していくことによって、フォームを隠すことができます。

これはコードを書いて行うこともできますが、Desktop IDEがVisual Designerを使った場合は、NS Basic/CEがフォームの作成、表示／非表示に必要なコードを作成します。

フォームはPictureBoxオブジェクトが基礎になっていて、同じプロパティを持っています。

### フォーム メソッド

各フォームは"Hide"（隠す）と"Show"（見せる）メソッドを持っています。フォームの名前が"Form1"の場合、次のコードによってフォームが隠れます。

```
Form1.Hide
```

フォームを表示する場合は、Form1.Showを使います。始めてフォームが表示される時、NS Basicはそのフォームのオブジェクトを作成します。フォームが作成される時、変数Form1(0)はfalseになります（フォームの名前はForm1と仮定）。全てのオブジェクトが作成されると、Form1(0)はtrueになります。この変数を使うと、フォーム上のオブジェクトが発生させるイベントに応答する前に、フォームが完成しているかを確認できます。

### フォーム イベント

Desktop IDEを使ってプログラムを作成した場合、フォーム読み込みイベントがプログラムに送られます。フォーム名がForm1とすると、Form1\_Showによってオブジェクトが作成された後、Form1\_Load()サブルーチンが呼ばれます。同様に、フォームを閉じるとForm1\_Unloadイベントが送られます。

## プログラムのプロパティとイベント

NS BasicプログラムはOutputオブジェクトと呼ばれるPictureBoxオブジェクトの中で実行されます。PictureBoxコントロールを参照して、サポートされているプロパティとイベントを確認して下さい。

```
REM Turn application background to blue
Output.backcolor=vbBlue
```

## プログラムイベント

プログラムが閉じられる（またはPocket PCで最小化）時、Output\_Closeイベントがプログラムに送られます。このイベントは、Desktop IDEで作られたかどうかにかわらず、全てのプログラムに送られます。この機能は特にPocket PCデバイスには便利です。マイクロソフトのPocket PCユーザインターフェース ガイドラインは、アプリケーションの右上のボタン（他のバージョンのWindowsではアプリケーションを閉じる）は、アプリケーションを最小化するだけで実行は継続するように指定しています。あなたのアプリケーション内でこの動作をオーバーライドするには、以下のコードを含めて下さい。

```
REM Close app when minimized
Sub Output_Close()
    Bye
End Sub
```

また、ShowOKButtonステートメントを使うと同じことが行えます。

ウィンドウのサイズが変わる場合（例えば、スクリーンの向きが変わる時）、Output\_Sizeイベントがあなたのプログラムに送られます。

## モジュール

モジュールとはプロジェクトに含まれる .txt, .cod, .basファイルのいずれかです。これらによってプログラムコードを小さなグループに分ける事ができ、プロジェクトが管理し易くなります。また、モジュールは複数のプロジェクトで使うことができ、開発時にコードの貸し借りが可能になります。実行時、プロジェクトの全てのモジュールは結合し1つのプログラムとなります。モジュール内のコードはメインプロジェクトの最後に置かれます。

### 3. デバイス上でのプログラミング

#### 3.1 概要

NS Basic/CEのデバイス上での開発環境は、2つの異なるビュー（プログラムエディタとアウトプットウィンドウ）を持っています。さらに、プログラムエディタ用にグラフィックを使ってコードを作成する、Visual Designerと呼ばれるツールがあります。

#### 3.2 プログラムエディタ

プログラムエディタはプログラムのコードを組む場所です。白いバックグラウンドに縦スクロールバー、及びNS Basic/CEをコントロールするツールバーから構成されています。ツールバーのメニューは以下に要略されています。Table 1 参照。

Table 1: NS Basic/CE メニュー

メニュー	コマンド	説明
File (ファイル)	New (新規)	新規プログラム
	Open (開く)	プログラムを開く
	Save (保存)	プログラムの保存
	Save As...(…で保存)	新しい名前でプログラムの保存
	Encryption ON/OFF (暗号化 ON/OFF)	保存されるプログラムの暗号化状態を設定する。暗号化を行った人だけがプログラムのソースコードを開ける
	1...4	最後に使われた4つのプログラム
	Exit(終了)	NS Basic/CEを終了

Table 1: NS Basic/CE メニュー

メニュー	コマンド	説明
Edit (エディット)	Undo (取消し)	最後の変更の取消し/やり直し
	Cut (カット)	選択テキストのカット
	Copy (コピー)	選択テキストのコピー
	Paste (ペースト)	クリップボードの内容をペースト
	Select All... (全指定)	全てのプログラムテキストを指定
	Find... (検索)	語句の検索
	Find Again (再検索)	検索の繰返し
	Find All... (全検索)	全ての一致を検索
	Goto Line... (ラインへ飛ぶ)	指定ラインの表示
	Overview (概要)	プロシージャのリスト
	Format (フォーマット)	プログラムの再フォーマット
Tools (ツール)	Run (実行)	プログラムの実行
	...Run (実行)	指定プログラムの実行
	..Execute Function (ファンクション実行)	ファンクションから再実行
	..Trace (トレース)	プログラム処理をトレース
	...Step (ステップ)	1 ステートメント毎
	Execute Code... (コード実行)	一時的なプログラムの実行
	Show Variable... (変数表示)	変数の値を表示
	Stats... (統計)	ファイルサイズデータ
	Visual Designer (ビジュアルデザイナー)	フォームエディタ
	Help Topics... (ヘルプトピック)	ヘルプブラウザ
Help (ヘルプ)	About NS Basic/CE... (NS Basic/CEについて)	タイトル、バージョン、連絡先



ツールボタンはメニューアイテムの右側にあります。

Table 2: ツールボタン

ボタン	メニュー選択	ショートカット
	"File->New"	Ctrl+N
	"File->Save"	Ctrl+S
	"File->Open"	Ctrl+O
	"Edit->Cut"	Ctrl+X
	"Edit->Copy"	Ctrl+C
	"Edit->Paste"	Ctrl+V
	"Tools->Run"	Ctrl+R

プログラムエディタ画面の右上コーナーに2つボタンがあります。ヘルプボタンは、"Help->Help Topics..."のショートカットで、NS Basic/CEオンラインヘルプを開きます。



クローズボタンは、"File->Exit"のショートカットで、NS Basic/CEを終了します。



### 3.3 アウトプット ウィンドウ

アウトプットウィンドウはプログラムの実行中に開かれ、インプット、アウトプットが行われます。灰色のバックグラウンドに右上角にクローズボタンがあります（Windows CEのバージョンによって異なります）。画面の左上角の小さな線が、SETMENUステートメントを使ってカスタムメニューを追加する場所です。クローズボタンはアウトプットウィンドウを閉じます。プログラムがプログラムエディタの中から実

行された場合は、コントロールはエディタに戻ります。それ以外の時、NS Basic/CEは終了します。

### 3.4 ビジュアルデザイナー (Visual Designer)

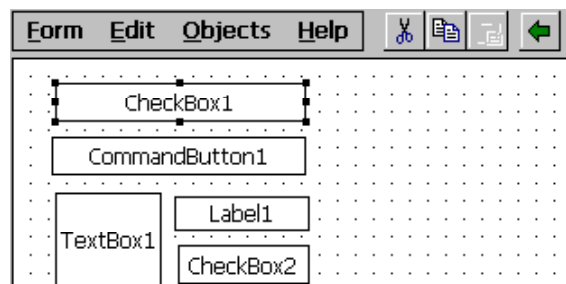
Visual DesignerはNB Basic/CEの一部で、完全なBASICの機能を備えた開発環境です。WindowsCE機上で、視覚的にフォームを作成することが出来ます。NS Basic/CE環境から起動します。

フォームにオブジェクトを加えるには、Objects (オブジェクト) メニューから欲しいオブジェクトを選択して下さい。オブジェクトがフォームの左上コーナーに作成されます。ドラッグすることによりサイズを変更し、オブジェクトの大きさを設定出来ます。オブジェクトの上をダブルタップをすると、オブジェクトのProperty Editorが現れます。オブジェクトの属性 (プロパティ) を変更/設定できます。

Visual Designerを閉じると、オブジェクトの仕様はNS Basic/CEコードに変えられ、通常のNS Basic/CE編集画面に戻ります。作られたコードはすぐに実行可能です。

フォーマットの変更をしたい場合、いつでもVisual Designerに戻ることが出来ます。

警告 : Visual Designerによって作成されたコードを修正すると、Visual Designerで再度編集することが出来なくなります。また、再度ジェネレートすると変更したものが失われるかもしれません。



**Table 3: ビジュアルデザイナー メニュー**

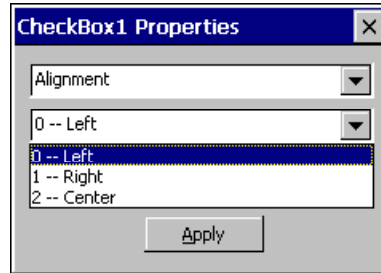
Form	<u>A</u> dd Form	新規フォームの追加
	<u>D</u> elete	フォームの消去
	S <u>e</u> t Default	現フォームをデフォルト設定
	E <u>x</u> it	終了して、NS Basic/CE編集にもどる
Edit	<u>U</u> ndo	取消し
	Cu <u>t</u>	カット
	<u>C</u> opy	コピー
	<u>P</u> aste	ペースト
	Clea <u>r</u>	消去
	S <u>e</u> lect Object	指定用にオブジェクトリストを表示
	<u>P</u> roperties	指定のオブジェクトの特性編集。 指定されていない場合は現フォームの編集。プロパティエディタ参照。

**Table 3: ビジュアルデザイナー メニュー**

Object	Add <u>M</u> enu	フォームにメニューの追加。メニューエディタ参照
	Edit Menu	メニューの編集。メニューエディタ参照
	Show Program	プログラムコードの表示
	<u>S</u> nap to Grid	オブジェクト移動時にグリッドへ揃える
	<u>G</u> rid Size...	グリッド値設定(4-8 ピクセル)
	Che <u>C</u> kBox	Checkboxオブジェクトの追加
	Co <u>M</u> boBox	ComboBoxオブジェクトの追加
	Co <u>M</u> mandButton	CommandButtonオブジェクトの追加
	D <u>a</u> te	Date pickerオブジェクトの追加
	L <u>a</u> bel	Labelオブジェクトの追加
	L <u>i</u> stBox	ListBoxオブジェクトの追加
	Op <u>t</u> ionButton	OptionButtonオブジェクトの追加
	T <u>e</u> xtBox	TextBoxオブジェクトの追加
	T <u>i</u> me	Time pickerオブジェクトの追加
Help	<u>H</u> elp Topics	
	<u>A</u> bout VNSB	

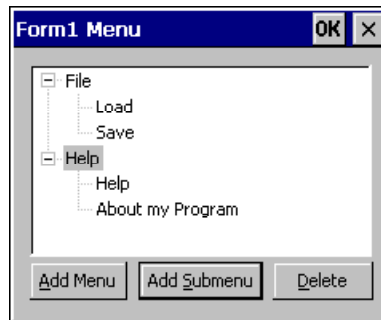
### 3.4.1 プロパティエディタ (Property Editor)

各オブジェクトのタイプには、それ自身のプロパティ (属性) のリストがあります。上部の窓にて、編集されるブ



ロパティを選択します。下部の窓は、プロパティの必要条件にあわせて、有効値のリストか入力フィールドになります。

### 3.4.2 メニューエディタ (Menu Editor)



メニューは多重レイヤのサブメニューを設定できます。各メニュー項目には、Caption、Menu KeyおよびAccelerator (ショートカット) を指定することができます。

### 3.4.3 注記

ComboBoxとListBoxには、Listという特別なプロパティがあります。編集窓の中に入力された各ラインは別々の選択肢になります。引用文字(')でラインが終わる場合、評価される式としてみなされます。

### 3.5 NS Basic/CEプログラミング環境

NS Basic/CEはWindows CE用にBASIC言語の機能を備えた環境を提供します。機能を紹介する目的でサンプルプログラムがNS Basic/CEに含まれています。プロセスの各ステップが機能を紹介しますので、NS Basic/CEをスタートして、以下を試してください。

#### 3.5.1 プログラムの作成

新規プログラム作成時、プログラムエディタは空に、アウトプットウィンドウは閉じられた状態にします。もしアウトプットウィンドウが開いている場合は、一旦それを閉じてから、"File"メニューの"New"を選択します。

サンプルプログラムはシンプルです。個々のアイテムのコストを足して、それに税金を加算し、コストの合計を計算します。このプログラムはエラーが伴っていますので、デバッグの練習にもなります。

以下のプログラムを入力して下さい。

```
REM Sales Calculator
OPTION EXPLICIT
'Declare Constants
APPTITLE = "Sales Calculator"
'Declare variables
DIM SubTotal, TaxRate, ItemCost, Item

'Initialize variables
SubTotal = 0
Item = 1

'Get item costs
DO
  ItemCost = INPUTBOX("Item " & Item & _
    " cost", 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

'Output
PRINT FORMATCURRENCY(SubTotal), , "Sub Total"
TaxRate = INPUTBOX("Tax Rate (10% = 10)", _
  APPTITLE, 0)
PRINT FORMATCURRENCY(SubTotal*TaxRate), , _
  "Tax"
```

```
SubTotal = SubTotal * (1 + TaxRate)
PRINT FORMATCURRENCY(SubTotal), , "Total"
```

**注記：**上記のInputBoxステートメントは、更に特別なコードなしではPocket PCデバイスでは動作しません。Readme.htmlに追加情報が記載されていますので、そちらを参考にして下さい。

### 3.5.2 プログラムの編集

プログラムエディタは標準のテキストエディタと同様の動きをします。カーソルの点滅がテキストの挿入位置を示します。矢印キー、page up及びpage down、home及びend、またはスタイラス（ペン）でタッチセンサー画面を押すことにより、カーソルは動かされます。

Cut、Copy、Pasteを使うことにより、テキストの選択されている部分を動かしたり取除いたり出来ます。テキストを選択するには、スタイラスでタップ及びドラッグを行うか、指定する場所の一方の端にカーソルを動かし、shiftキーを押しながらカーソルをもう一方の端へ操作して下さい。"Cut"、"Copy"、"Paste"は"Edit"メニューの中にあります。

一番最後に行った変更（操作）を取消す場合は、"Edit"メニューから"Undo"を指定して下さい。

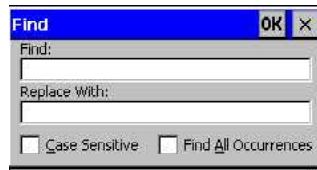
このエディタの最大サイズは1,048,575/バイトです。これより大きなプログラムの編集はDesktop IDEのエディタが使えます。

### 3.5.3 プログラムのフォーマット

プログラムを読み易くする為に、関連するステートメントをスペースやタブで段落を付けましょう。主にプロシージャやループの中のステートメントに使います。デフォルトのタブ及びインデントはスペース2つつ分です。プログラム全体を自動的にフォーマットするには、"Tools"メニューから"Format"を選んで下さい。

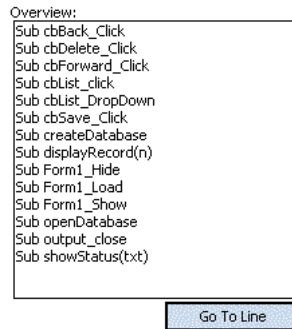
### 3.5.4 検索及び置換え

プログラムの検索は2通りあります。Findを使うとカーソルの位置から一番近く的一致を探せます。Find Nextは最後に行ったFindを繰返し、続けて検索する時に便利です。一致が見つかった場合はプログラムエディタはそれをハイライト（選択）し、必要に応じて画面をスクロールします。Case Sensitiveは格（大文字／小文字）による違いを認識させたい時に使います。Find All Occurrencesは一致が見つかった全てのラインを表にして戻します。"Find"、"Find Next"は"Edit"メニューの下にあります。



### 3.5.5 概要 (Overview)

Overviewコマンドは全てのFUNCTION及びSUBプロシージャを表示します。プロシージャはタイプ (FUNCTIONまたはSUB)で分けられた後にアルファベット順に並べられます。また、Overview表示から指定したプロシージャへ行くことができます。プロシージャを指定し"Goto Line"ボタンをクリックするか、プロシージャをダブルタップすることによりOverview表示を閉じプログラムエディタ内のプロシージャへジャンプします。





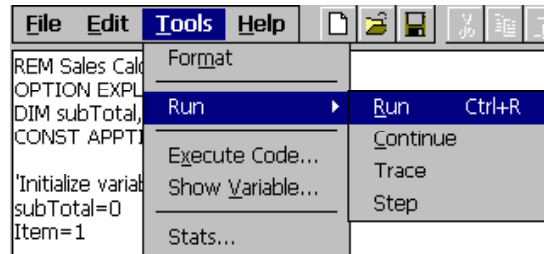
### 3.5.6 プログラムの実行

プログラムを実行するには、"Tools"メニューから  
"Run"を選ぶか、実行アイコンを選んで下さい。

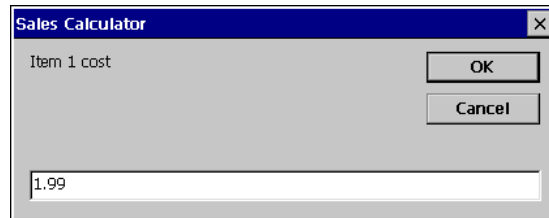


サンプルのプログラムを実行し、3つのコストを\$1.99、  
\$2.99、\$3.99と入力し、税率(Tax Rate)に6%を入れてみま  
しょう。

プログラムを実行...



最初のアイテムの入力...



プログラムの出力...

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

### 3.5.7 プログラムのデバッグ

計算した合計は、各アイテムに税を加算したものより遥かに多いです。プログラムにバグがあるようです。NS Basic/CEには3つのタイプのエラーがあります：コンパイル時エラー、実行時エラー、論理エラーです。

コンパイル時エラーは、プログラムのシンタックスが間違えている時に起こります。例えばキーワードのスペル間違い、括弧を付けてSUBプロシージャを呼び、必要な引数が抜けている場合などです。

実行時エラーは、プログラム実行時にNS Basic/CEが処理出来ないか、理解出来ないステートメントに直面した時に起こります。処理出来ないステートメントの例は、0による割算、文字列のかけ算、追加されていないオブジェクトの参照等です。理解出来ないステートメントの例は、タイプミスを含んだステートメント、スペース+アンダースコア(\_)を使用しないまま、複数のラインにまたがるステートメント、REMやアポストロフィで始まらないコメント等です。

論理エラーは、プログラムのアルゴリズムが正しくない場合で、プログラムはエラーなしで実行します。このエラーは間違っている結果や、予想しない結果になることによって検出できます。NS Basic/CEでのプログラムに慣れた後は、このタイプのエラーが主になります。実行時エラーと違い、論理エラーはNS Basic/CEでは検出できません。

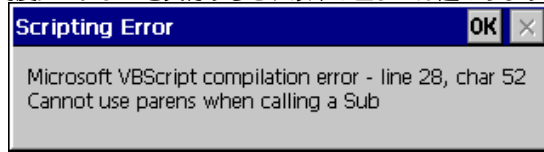
論理エラーは一般的で、我々のエラーも論理エラーです。プログラム実行時に何のエラーも起こりませんでした。結果のtotalが多すぎます。デバッグをしなくてはなりません。

論理エラーのデバッグを簡単にするには、実行中にもっと情報を得ることです。プログラムから出力を得るには2つの簡単な方法があります。PRINTステートメントとMSGBOXファンクションです。

アウトプットを見ると、Sub Totalは正しく計算されている事が確認できますので、税 (tax) の計算が間違えていることが分かります。税率の値および乗数と共にMSGBOXを呼ぶラインを挿入してみましょう。以下のラインをプログラムに追加します。

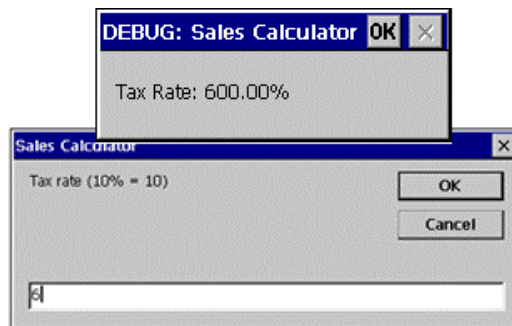
```
MSGBOX("Tax Rate: " _  
      & FORMATPERCENT(TaxRate), 0, "DEBUG: " _  
      & APPTITLE)
```

再度プログラムを実行すると、以下のエラーが起こります。



これはコンパイル時エラーで、シンタックスの間違いが原因です。FUNCTIONプロシージャ（この場合、MSGBOX）からの戻り値が変数に割り当てられない時、それはSUBプロシージャ扱いにされます。SUBプロシージャは括弧を使用した引数渡しが出来ないので、今追加したラインから括弧を取除きましょう。

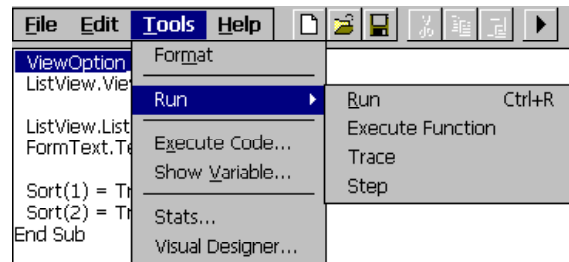
これを直した後、プログラムはエラー無しで実行され、次のデバッグアウトプットが表示されます。



これによりがTax Rateが正しく処理されていない事が分かり、100で割ることにより解決されるでしょう。プログラムのOutputセクションを以下と同様に変更した後、プログラムはエラー無しで実行され、結果も正しく計算されます。さらに、通常の出力に税率をレポートする情報も加えました。

```
'Output
PRINT FORMATCURRENCY(SubTotal), , "Sub Total"
TaxRate = INPUTBOX("Tax Rate (10% = 10)", _
    APPTITLE, 0) / 100
PRINT FORMATCURRENCY(SubTotal * TaxRate), , _
    "Tax (" & FORMATPERCENT(TaxRate) & ")"
SubTotal = SubTotal * (1 + TaxRate)
PRINT FORMATCURRENCY(SubTotal), , "Total"
```

### 3.5.8 BREAK、Execute Function、Trace、Step

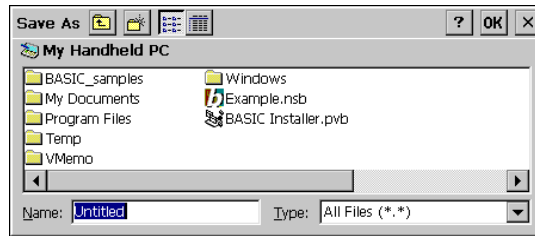


これらのツールを組合せて使うことにより、デバッグ能力を向上させることができます。Breakステートメントを使い、プログラムの中にブレークポイントを設定します。各ステートメントが実行される毎にそれらを確認したい場合は、Runの代わりにTraceを使います。Runの代わりにStepを選ぶと、各ステートメント毎にBreakが呼ばれ、プログラムの動きを確認しながら進むことが出来ます。OPTION EXPLICITがプログラムの一番最初のライン以外の場所にある場合、Trace及びStepの動きは不安定になりますので注意して下さい。

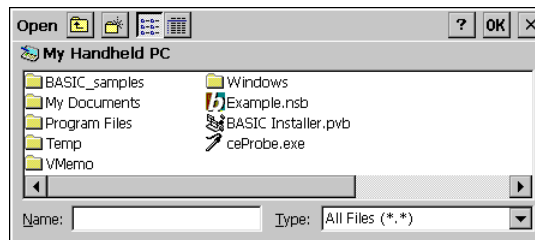
Execute Functionはファンクション及びサブルーチンのリストを表示します。一つを選択することにより、それを直接実行します。これはファンクションを直接デバッグするのに便利です。

### 3.5.9 プログラムの保存と読み込み

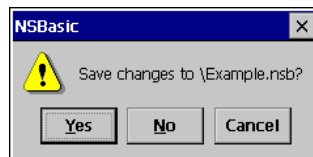
新しいプログラムを作成する時 ("File->New"を選ぶ)、それはプログラムエディタの中にだけしか存在しません。現在のプログラムを保存したい場合、"Save"を"File"メニューから選びます。これによりダイアログボックスが開き、どのような名前で保存するかを聞かれます。NS Basic/CEを終了したり、他のプログラムを開く場合も、同様にダイアログが開きます。では我々のプログラムをSalesCalculator.nsbという名前で保存してみましょう。



既に保存されているプログラムを編集または実行したい時は、"Open"を"File"メニューから選びます。いま保存したプログラムを開けてみましょう。



変更が加えられているプログラムを閉じようとする時はいつでも、ダイアログボックスが現れて、変更内容を保存したいか確認されます。



#### **3.5.10 テキストファイルとしてプログラムの保存と読み込み**

別のアプリケーションかデスクトップシステムで自分のコードを使用したい場合、テキストファイルとしてそれを保存できます。Save Asダイアログでファイル名の後に.txtを加えて下さい。また、テキストファイルをロード（読み込み）することが出来ます。

## 4. デスクトップ上でプログラミング

NS Basic/CEを使うと、Windows CEデバイスで動作するプログラムをデスクトップ上で開発できます。Virtual CEを使うと、デスクトップからデバイス上で動作するプログラムの操作が行えます。

### 4.1 メニュー

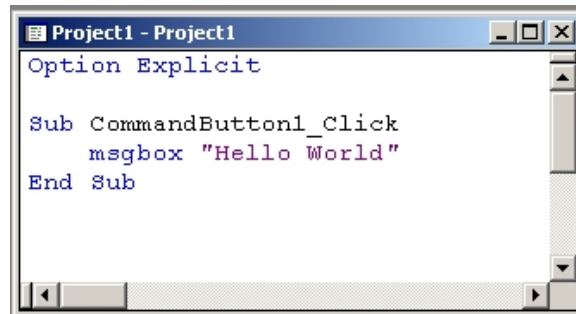
File	
New Project	新規プロジェクトを開く。開いているプロジェクトは先に閉じる
Open Project...	Openダイアログを開き、既存／最近のプロジェクトが選べる。詳細はOpenダイアログを参照
Save Project	プロジェクトを保存。プロジェクトが新しい場合、名前を要求される
Save Project As...	プロジェクトを別名で保存。ファイルは.nsbまたは.txtフォーマットで保存可能
Print Setup...	標準Windows印刷設定
Print...	プロジェクトを印刷
Recent Files	最近開いたプロジェクトのリスト
Exit	プロジェクトを閉じ、IDEを終了する
Edit	
Undo	コードウインドウ内での最後に行った編集を取り消す。複数回に渡る取り消しが可能
Redo	コードウインドウ内での取り消しをやり直す
Cut	選択したコードまたはオブジェクトを切り取る。オブジェクトは標準テキストフォーマットとして切り取られる（コピーも同様）ので他のアプリケーションで見たり編集したりできる
Copy	選択したコードまたはオブジェクトをコピーする
Paste	コードまたはオブジェクトを貼付ける
Delete	選択したコードまたはオブジェクトを削除する

Find...	キーワードでコードを検索。プロジェクト全体か、選んでいるコードのみかを検索可能。指定単語に完全に一致や大文字／小文字の区別をするオプションあり
Replace...	検索と似ているが、1つまたは全て的一致したものを指定単語と入れ替える
Go To Line	コードウインドウの指定行に移動
View	
Project Explorer	プロジェクトエクスプローラ(Project Explorer)ウインドウの表示／非表示
Properties Window	プロパティ(Properties)ウインドウの表示／非表示
Toolbox	ツールボックスの表示／非表示。ツールボックスは利用可能なオブジェクトを表示する
Toolbar	ツールバーの表示／非表示（メニューのすぐ下）
Status Bar	ステータスバーの表示／非表示（IDEの下部）
Refresh	全てのウインドウを再描画
Project	
Add Form	プロジェクトに新しいフォームを追加する。フォームはオブジェクトの集まりであり、それらはプログラムによって同時に表示される
Format	
Size to Grid	選択しているオブジェクトのサイズを変更しグリッドパターンに揃える。グリッドのサイズはTools...Option...CE Screenで変更可能
Center in Form	選択しているオブジェクトをフォームの中央に揃える（縦揃えまたは横揃え）
Run	
Start	プログラムを実行する。プログラムがデバイスへコピーされる。「Tools...Options...Start」の設定によって、プログラムは自動的に実行される。
Active Sync Program	ActiveSyncを使ってプログラムを実機にコピーする
Installers...	Installersフォルダーに入っている、デバイス用のインストーラのリストを表示。
Start Virtual CE	Virtual CEを起動する。ActiveSyncによりデバイスが接続されていることが必要。
Tools	
Menu Editor	メニューエディタを起動する
Options...	Optionsウインドウを開く



Window	
Cascade	画面上のCE Screenとコードウィンドウを階段状に並べる
Tile Horizontal	画面上のCE Screenとコードウィンドウのサイズを変更し、縦方向に並べる（各ウィンドウは横長になる）
Tile Vertical	画面上のCE Screenとコードウィンドウのサイズを変更し、横方向に並べる（各ウィンドウは縦長になる）
Help	
Register...	シリアル番号をここから入力し、NS Basic/CEを有効にする。シリアル番号はハンドブックの裏面に記載
Help	ハンドブックのリファレンスの部分が記載されたヘルプウィンドウを開く
NS Basic Website	NS BasicのWebサイトを開く
Tech Notes	ハンドブックに記載されていない追加情報を載せたTech Notesを開く
Big Red Toolbox	NS Basicユーザが利用可能な追加ActiveXコントロールの情報を表示する
About NS Basic/CE	標準NS Basic About画面。お使いのNS Basicのバージョンを確認できる

## 4.2 コードウィンドウ (Code Window)



あなたのプログラムのコードはコードウィンドウに入力します。全てのコードは1つのコードモジュールに入りますが、CE Screen中のオブジェクトをクリックすると、コード中の適切な場所へ自動的に飛びます。

テキストはタイプによって色付けされます。

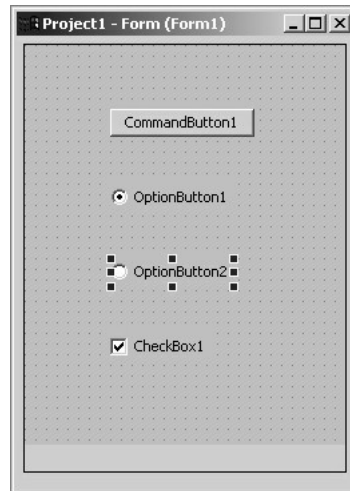
Black	プログラムテキスト
Blue	NS Basic/CEキーワード
Green	コメント
Orange	演算子
Purple	文字列
Yellow	ハイライトされているライン

Options...Editorへ行くといくつかの便利なオプションを開きます。プロパティの選択によって広範囲に渡る機能のカスタマイズが可能になります。それらはテキストの色付け、タブ、フォント、キーボードショートカット、ウィンドウの外観等です。

“ ”を行の最後に置くことによって、ステートメントを複数行にすることが可能です。

Cut、Paste、Delete、および他の同様な機能は、メニュー、ツールバー、または右クリックで行うことができます。

### 4.3 CE Screen



CE Screenはあなたのアプリケーションがデバイス上で動作する時、どのように見えるか映し出す疑似画面です。

背景のグリッド（デフォルト設定）はフォームのレイアウトを簡単にするためのもので、実機上では表示されません。Optionsで設定を変えることにより、グリッド幅を変えたり消したりすることが可能です。

CE Screenにオブジェクトを追加するには、ツールボックスからオブジェクトを選び、CE Screen上の配置したい場所をクリックして下さい。クリックした場所がオブジェクトの左上コーナーとなって表示されます。

オブジェクトを選択するには、それをクリックして下さい。Formatメニューの下にある機能を使うと、画面上でオブジェクトをきれいに配置できます。

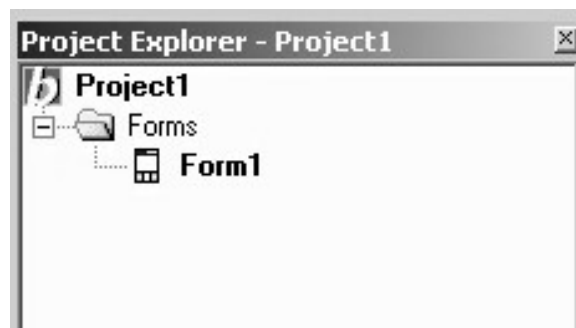
オブジェクトが選択されると、そのプロパティがPropertiesウィンドウに表示されます。プロパティを変更すると、直ちにCE Screen上に反映されます。オブジェクトのコードを編集するには、オブジェクトをダブルクリックしコードウィンドウを開いて下さい。

フォーム上でオブジェクト以外の部分をクリックすると、フォームのPropertiesウィンドウが開きます。オブジェクト

の名前を変えたり、削除する場合は、CE Screen上のオブジェクトを右クリックします。

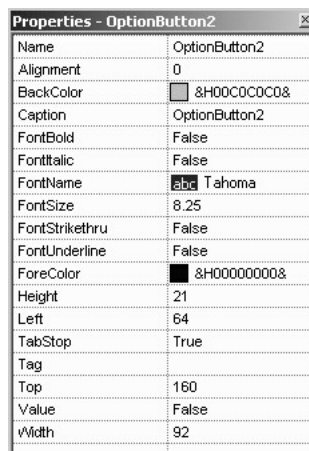
フォームの上をダブルクリックすると、Form\_Loadサブルーチンが開き、編集が行えます。Form\_Unloadサブルーチンは自動的に開きませんので、自分で追加する必要があります。




#### 4.4 プロジェクトエクスプローラ (Project Explorer)



プロジェクトエクスプローラはプロジェクトの構成を表示します。これを見るとフォームもプロジェクトの一部であることが分かります。一つのフォームを選択すると、CE Screenにそのレイアウトが、Propertiesウィンドウにプロパティの値が表示されます。

## 4.5 プロパティウィンドウ (Properties Window)



Name	OptionButton2
Alignment	0
BackColor	 &H00C0C0C0&
Caption	OptionButton2
FontBold	False
FontItalic	False
FontName	 Tahoma
FontSize	8.25
FontStrikethru	False
FontUnderline	False
ForeColor	 &H00000000&
Height	21
Left	64
TabStop	True
Tag	
Top	160
Value	False
Width	92

プロパティウィンドウは、プロジェクト、フォーム、オブジェクトのプロパティを表示します。値は変更可能です。反映されるものであれば、CE Screenが更新されます。

プロパティの説明は、ハンドブックのリファレンスチャプターにあるPropertiesをご覧ください。

## 4.6 ツールボックス (Toolbox)



ツールボックスは、アイコンの集まりで、それらはフォーム上に配置できる標準コントロールを示します。アイコンの上でカーソルを少しの間止めると、その説明が現れます。

ActiveXだけは特殊で、ツールボックスにない他のコントロールを使うことができます。ActiveXアイコンをクリックし、使いたいコントロールを選んで下さい。プロジェクトに追加されます。

コントロールに関する詳細は、ハンドブックのリファレンスチャプターにあるAddObjectステートメントをご覧ください。固有のオブジェクトはリファレンスセクションに、他のオブジェクトはTech Notesに記載されています。

ツールボックスには全てのActiveXコントロールは追加されません。これにはデスクトップバージョンが必要で、作者によって全てのプロパティを正しく定義してなければなりません。ツールボックスを使う代わりに、ActiveXコントロールはAddObjectステートメントを使ってプロジェクトに追加することができます。この場合、プロパティウィンドウを使う代わりに、コード内でプロパティを編集します。

オブジェクトが作成された後にその名前を変更するには、オブジェクトを右クリックし、“Rename”（名前変更）を選んで下さい。

## 4.7 ツールバー (Toolbar)

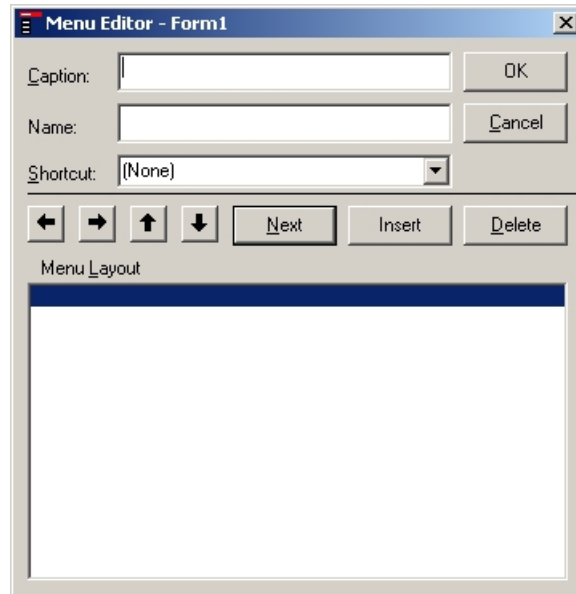


ツールバーは一般オペレーション用のアイコンを持っています。これらの機能は全てメニューにも含まれています。

New (新規)  
Open (開く)  
Save (保存)  
Print (印刷)  
Undo (取り消し)  
Redo (やり直し)  
Cut (切り取り)  
Copy (コピー)  
Paste (張り付け)  
Delete (削除)  
Search (検索)  
Add Form (フォームを追加)  
Start (実行)  
Properties (プロパティ)  
Menu Editor (メニューエディタ)



#### 4.8 メニューエディタ (Menu Editor)



メニューエディタを使うとメニューの編集が行えます。各フォームは1つのメニューを持つことができ、メニューは複数のレイヤーを持つことができます。

Caption	メニューに現れるタイトル
Name	メニューが選択された時に呼ばれるファンクションの名前。スペースおよび特殊文字を含むことはできません
Shortcut	アイテムを選択するためのショートカットキー
OK	変更をメニューに保存し、閉じる
Cancel	メニューへの変更は保存せずに、閉じる
Left	選択したアイテムのレベルを1つ上げる
Right	選択したアイテムのレベルを1つ下げる
Up	一つ前のアイテムに動く
Down	一つ後ろのアイテムに動く
Next	一つ後ろのアイテムに動く。最後の場合、新しいアイテムを最後に追加する

Insert	新しいアイテムを選択しているアイテムの前に追加する
Delete	選択しているアイテムを削除する

アイテムが選択された時に実行されるコードを追加するには、メニューエディタを閉じ、CE Screenからメニューアイテムを選んで下さい。メニューが選択された時に実行されるサブルーチンと共にコードウィンドウが現れます。

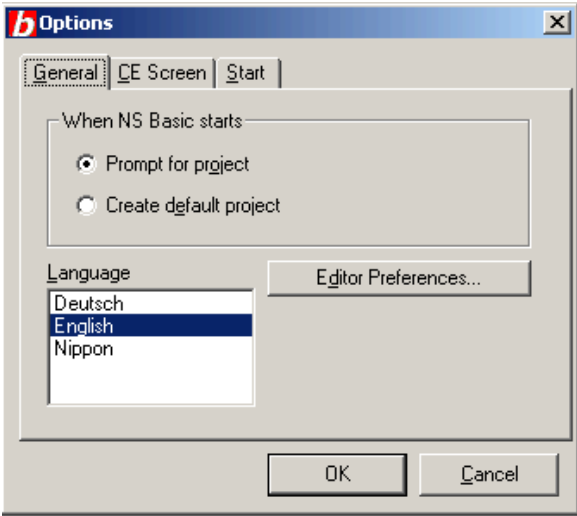
サブルーチンは `menuItem_click` として名前が付きます。MenuItemはメニューエディタ内のNameでしてした名前です。この名前が他のメニューやボタン、オブジェクト、またはフォームの名前と重ならないように注意下さい。

サブメニューを作成するには、右矢印キーを使ってアイテムをインデントさせて下さい。左矢印を使うと戻ります。

上下矢印はメニューアイテムの順番を変えるのに使います。アイテムを選択し、上または下矢印を使って下さい。

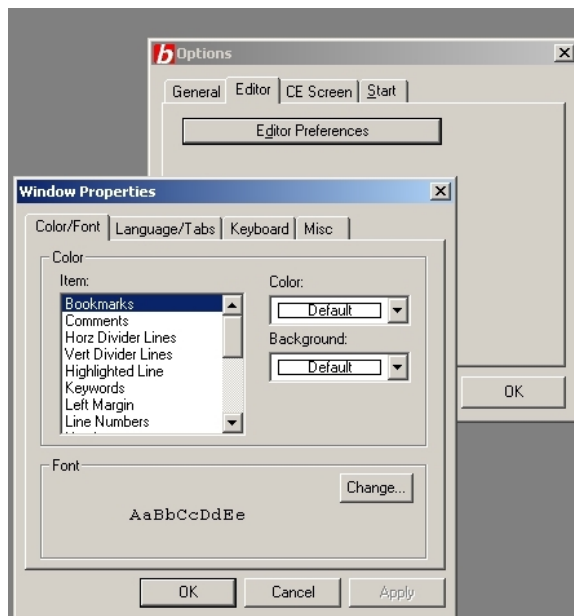
4.9 オプション (Options)

4.9.1 オプション – 一般 (General)



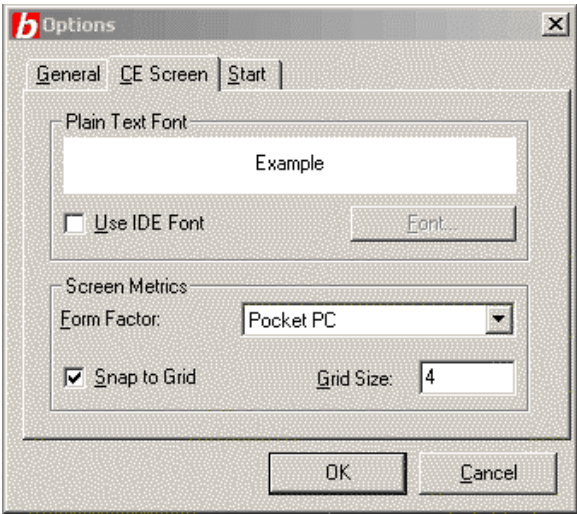
Prompt for project	選択された場合、NS Basic/CE起動時に、既存のプロジェクトを開くか、新規プロジェクトを開くかを聞く
Create default project	選択された場合、NS Basic/CE起動時に、新規プロジェクトを開く
Language	IDEの言語を設定する。NS Basic/CEのLangフォルダーにあるファイルがリストとなって表示される
Editor Preferences	Code Windowの設定。4.9.2参照

#### 4.9.2 オプション – エディタ (Editor)



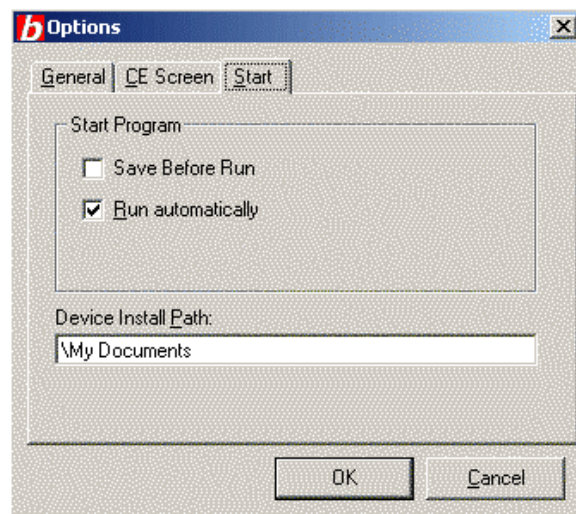
エディタオプションの画面は、コードウィンドウをカスタマイズするための幅広い機能を持っています。これらはテキストの色付け、タブ、フォント、キーボードショートカット、ウィンドウの外観等です。

4.9.3 オプション – CE Screen



Snap to Grid	チェックすると、オブジェクトは自動的にグリッドに揃って配置され、簡単に整ったレイアウトができます
Grid Size	“Snap to Grid”が選択されている場合、この設定によって、グリッドの間隔を設定します

#### 4.9.4 オプション – Start



この画面では、RunメニューのStartが選択された時に使われるオプションを設定できます。

Save Before Run	Run...Startで実行する前に毎回ファイルを保存する
Run automatically	ActiveSyncによってプログラムがコピーされた後、プログラムは直ちに実行される。
Device Install Path	プログラムがコピーされるデバイス上のロケーション。指定するフォルダーは存在しなくてはならない。正しいパス名はデバイスによって異なる。Pocket PCデバイスの場合、これは"My Documents"で、他のデバイスの場合は単に"\"。

## 5. NS Basic/CE リファレンス

このチャプターはNS Basic/CEのコマンド、ステートメント及びファンクションの各アイテムをアルファベット順に記載し説明します。各アイテムは索引のFunctions、Objects、Statementsから参照できます。

各アイテムは以下の内容を含んでいます。

名前	カテゴリー
----	-------

アイテム パラメータ

### 説明

このセクションではアイテム及びパラメータを説明します。アイテムの使用方法等の詳細及び制約等も含んでいます。

### 例

アイテムを使用している簡単なプログラムのセクションです。

### 結果

このセクションは上の例のプログラムの実行結果が表示されます。

### 関連アイテム

このアイテムに関連したNS Basic/CEコマンド、ステートメント、ファンクション及びオブジェクト名の一覧です。関連アイテムの参照は、ご覧になっているアイテムへの理解を助けます。

ABS(*number*)

**説明（絶対値）**

指定した数値の絶対値を戻す関数です。*number*は絶対値を求めたい数値式。*number*がNULLの場合、ABSはNULLを戻します。

**例**

```
REM ABS Example
'ABS returns the absolute value of a number.
PRINT "ABS(-2) = " & ABS(-2)
PRINT "ABS(2) = " & ABS(2)
```

**結果**

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

**関連アイテム**

SGN



```
ADDOBJECT objectID:license[, objectname[, xpos, ypos, width,  
Height, parent]]
```

#### 説明（オブジェクトの追加）

ADDOBJECTはオブジェクトをプログラムに加えます。一旦オブジェクトが加えられると、プログラム内でそのオブジェクトのプロパティにアクセスしたり、メソッドを実行できます。必要なパラメータ*objectID*は文字列型で、レジストリからのオブジェクトのフルネームまたはオブジェクトの短い名前です。*objectID*名はTable 4を参照して下さい。コントロールが*license*番号を必要とする場合、*objectID*の後にコロンを置き、その後に入力して下さい。オプションのパラメータ*objectname*は、プログラムから参照する為の名前です。*objectname*が指定されない場合、デフォルトとして*objectID*が使用されます。オプションのパラメータ*xpos*（X軸）、*ypos*（Y軸）、*width*（幅）、および*height*（高さ）は数値型で、視覚的オブジェクトの位置とサイズをピクセル値で設定します。*parent*（親）オブジェクトを指定することにより、作られるオブジェクトはその親の子となり、親オブジェクトの上に配置されます。

各オブジェクトの説明及び値はプロパティに含まれます。  
（Propertiesを参照）

オブジェクトはプログラムから呼ぶファンクションを含む場合があります。（Methodsを参照）

NS Basic/CEに含まれているオブジェクトは本ハンドブック内で説明しています。NS Basic/CEはActiveXとして知られる外部オブジェクトもサポートしています。これらの多くはNS Basic/CEに含まれていますが、それ以外はサードパーティから入手可能です。NS Basic/CEに含まれているものに関しては、テクニカルノート（Tech Notes）内で説明しています。Tech NotesはHelpメニューの下からアクセスできます。

オブジェクトはSUBプロシージャを呼ぶことによりプログラムにイベントを送ります。プロシージャ名はオブジェクト名とイベントの組み合わせです。いくつかのイベントはカンマで区切られたパラメータがオプションで付加されます。重要：オブジェクトがイベントを送る場合、全て（6つ）の引数を設定しなければなりません。（Eventsを参照）

```
SUB objectname event[(arglist)]  
'Execute this when objectname gets event  
END SUB
```

Table 4: 一般オブジェクト

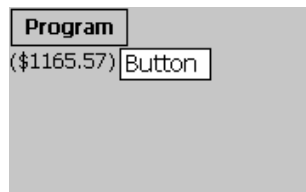
短縮名	記載されている場所
ADO	Data Base - Tech Note 10
CheckBox	本稿
ComboBox	本稿
Comm	Serial Comms - Tech Note 04
CommandButton	本稿
Date	本稿
Dialog	Dialog boxes - Tech Note 08
File	Simple File I/O - Tech Note 03
FileSystem	Files and Dirs - Tech Note 02
Finance	Money calculations - Tech Note 11
Frame	本稿
Grid	Data Grid - Tech Note 06
HScrollbar	本稿
Image	Image holder - Tech Note 20
ImageList	List of Images - Tech Note 21
Label	本稿
ListBox	本稿
ListView	Display Items - Tech Note 22
OptionButton	本稿
Output	本稿
PictureBox	Manipulate Pictures - Tech Note 07
TabStrip	Tab object - Tech Note 09
TextBox	本稿
Time	本稿
TreeView	Tech Note 23
TriButton	本稿
VScrollBar	本稿
WinSock	Internet stuff - Tech Note 05

**例**

```
REM ADDOBJECT adds an object to the program
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
```

**結果**



**関連アイテム**

Events, Methods, Properties, SET

*result* = *x* AND *y*

**説明**（論理積演算）

*x*と*y*の論理積を求めます。*x*、*y*がそれぞれ真(True)と評価されるときに限り、*result*は真(True)になります。どちらか一方の式が偽(False)と評価されるときは、*result*は偽(False)になります。

ANDは数値のビット単位の比較も行います。*x*および*y*の相対するビットが共に 1 の場合、*result*の*x*と*y*に相対するビットは 1 になり、それ以外は 0 をセットします。

**例**

```
REM AND Example
'AND performs logical and bitwise conjunction
DIM Test1, Test2, x, y
x = 2
y = 7
Test1 = x > 0 AND y < 10
Test2 = x > 0 AND y > 10
PRINT "Logical:"
PRINT "  x > 0 AND y < 10 = " & Test1
PRINT "  x > 0 AND y > 10 = " & Test2
PRINT "Bitwise:"
PRINT "  x AND y = " & x AND y
```

**結果**

```
Logical:
  x > 0 AND y < 10 = True
  x > 0 AND y > 10 = False
Bitwise:
  x AND y = 2
```

**関連アイテム**

EQV, IMP, NOT, OR, XOR

ARRAY(*expressionlist*)

説明 (ダイナミック配列作成)

ARRAYはプログラム実行中に配列を作成します。  
*expressionlist*には、格納される各要素に代入する式のリストを指定します。複数の式を指定するときは、カンマ(,)で区切ります。ARRAYで作成された配列のインデックスは0から始まります。

例

```
REM ARRAY Example
'ARRAY creates an array dynamically
DIM MyArray, Message
'Create an array with 2 string elements
MyArray = Array("こんにちは", "下江さん")
'Access the array and concatenate elements
PRINT MyArray(0) & " " & MyArray(1)
```

結果

こんにちは下江さん

関連アイテム

DIM

ASC(*string*)  
ASCB(*string*)  
ASCW(*string*)

**説明** (ANSI 文字コード)

指定した文字列の、最初の文字の文字コードを戻す変換関数です。*string*は文字列型です。もし1文字より長い文字列であれば、最初の文字だけが使われます。

ASCBは文字列に含まれるバイトデータと共に使われます。最初の文字の文字コードを返す代わりに、最初の1バイトを返します。ASCWはUnicodeを使う32ビットプラットフォーム用です。Unicodeで文字コードを返しますので、ANSIからUnicodeに変更する手間が省けます。

**例**

```
REM ASC Example
'ASC returns an ANSI character code
DIM CapitalA, LowerB
CapitalA = ASC("A is for Apple")
PRINT "Character code for A = " & CapitalA
LowerB = ASC("b")
PRINT "Character code for b = " & LowerB
```

**結果**

```
Character code for A = 65
Character code for b = 98
```

**関連アイテム**

CHR

ATN(*number*)

**説明** (アーктanジェント)

指定した $number$ のアークトanジェントをラジアンで戻す数値演算関数です。ATNは、直角三角形の2辺の比を引数 $number$ として受け取り、対応する角度を戻します。2辺とは、直角をはさむ2つの辺を指します。2辺の比は、求める角の反対側の辺の長さをもう一方の辺の長さで割った値です。

角度の単位を度からラジアンに変換するには、度に $\pi/180$ を掛けます。

ラジアンから度に変換するには、ラジアンに $180/\pi$ を掛けます。

**例**

```
REM ATN Example
'ATN calculates the arctangent of a number
DIM Pi
Pi = ATN(1) * 4
PRINT "The value of pi is " & Pi
```

**結果**

The value of pi is 3.141593

**関連アイテム**

COS, SIN, TAN

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

**説明**（実行の一時停止）

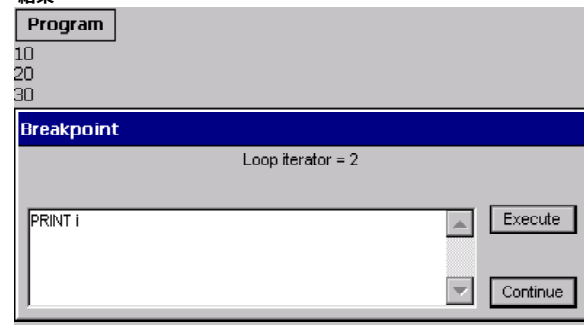
BREAKは実行を一時的に止めてダイアログボックスを開き、プログラマーにステートメントを実行するか、または実行を続けるかを促します。オプションのパラメータ*prompt*は文字列型で、ダイアログボックス上に表示されます。オプションのパラメータ*statements*は文字列型で、ダイアログボックスの実行領域に表示されます。

**例**

```
REM BREAK Example
'BREAK temporarily halts execution
DIM i

FOR i = 0 TO 4
  PRINT (i + 1) * 10
  IF i = 2 THEN
    BREAK "Loop iterator = " & i, "PRINT i"
  END IF
NEXT
```

**結果**



**関連アイテム**

MSGBOX



BYE

**説明** (プログラムの終了)

BYEは現在のプログラムを終わらせて、アウトプットウィンドウを閉じます。BYEはFUNCTIONまたはSUB内でオペレーティングシステムから送られたイベントによって呼ばれない限り効果がありません。

**例**

```
REM BYE Statement
'BYE closes output window and ends the program
DIM When, In10
In10 = MSGBOX("Close in 10 seconds?", vbYESNO)
IF In10 = vbYES THEN
    PRINT "Closing in 10 seconds..."
    When = DATEADD("s", 10, NOW)
    WHILE NOW < When
        'Doing nothing for 10 seconds
    WEND
    BYE
END IF

SUB Output_Click()
    Quit
END SUB

SUB Quit
    BYE
END SUB
```

**結果**

Open for 10 seconds...  
(output window closes)

**関連アイテム**

```
CALL procedurename[(argList)]
```

**説明** (プロシージャを呼ぶ)

CALLはSubプロシージャ、Functionプロシージャ、に制御を渡すフロー制御ステートメントです。オプションのコンポーネント *argList*は、呼ばれるプロシージャ *procedurename* に渡す変数でカンマ区切で記載します。キーワード "CALL" はオプションで、以下のようにキーワードなしでもプロシージャは実行できます。

```
name[(argList)]
```

**例**

```
REM CALL Example
'CALL explicitly executes a procedure
CALL Welcome
CALL Message("NS Basic/CE is excellent.")
Wave
```

```
FUNCTION Welcome
    PRINT "Hello World!"
END FUNCTION
```

```
FUNCTION Message(Text)
    PRINT "Message: " & Text
END FUNCTION
```

```
SUB Wave
    PRINT "Goodbye!"
END SUB
```

**結果**

```
Hello World!
Message: NS Basic/CE is excellent.
Goodbye!
```

**関連アイテム**

SUB, FUNCTION

CHAIN *pathname*, *reset*

**説明** (プログラムのロード及び実行)

CHAINは環境に1つのプログラムを読み込み、実行を始めます。必要なコンポーネント *pathname* は文字列型で、読み込んで実行するプログラムの絶対パス名です。必要なコンポーネント *reset* はブール型 (TRUE/FALSE) で、新しいプログラムを取り込んで実行する前に、環境が完全にリセットされるべきであるかどうか (全ての定数、変数、およびプロシージャがメモリから消去される) を指定します。

呼ぶ側のプログラムの実行を完全に終わらせるには、BYEステートメントをCHAINの後で使います。

*reset* がTRUEならば、止まるまで、オリジナルプログラムの実行が続きます。全ての変数とサブルーチンがリセットされた後、新しいプログラムは読み込まれ実行されます。

*reset* がFALSEならば、すべての変数とサブルーチンは影響を受けないまま、すぐに新しいプログラムが実行されます。新しいプログラムの終了時、オリジナルプログラムの次のステートメントから続きます。CHAINされるプログラムで加えられるか、または変更される変数とサブルーチンは残ります。

ヒント: *reset* をFALSEにしてCHAINを使用し、複数のプログラムに共通のプログラムステートメントを含むようにします。

**例**

```
'Program A
PRINT "This is program A"
A=100
CHAIN "B.nsb",TRUE 'reset the environment
PRINT "This is program A after CHAIN"
PRINT A

'Program B
PRINT "This is program B"
PRINT A
A=200
```

**結果**

```
This is program A
This is program A after CHAIN
100
```

```
<screen clears>
This is program B
<no value prints as A is undefined in program B>
```

## 例 2

```
'Program A
PRINT "This is program A"
A=100
CHAIN "B.nsb",FALSE 'don't reset the environment
PRINT "This is program A after CHAIN"
PRINT A

'Program B
PRINT "This is program B"
PRINT A
A=200
```

## 結果

```
This is program A
This is program B
100
This is program A after CHAIN
200
```

関連アイテム  
EXECUTE

```
ADDOBJECT "CheckBox", name, xpos, ypos, width, height  
ADDOBJECT "TriButton", name, xpos, ypos, width, height
```

**説明** (チェックボックス)

CheckBoxはアウトプットウィンドウ上に、チェックマークがあるボタンオブジェクト (2つの相対状態を表す) を表示します。TriButtonはCheckBoxと同じですが3つ目のグレースになった中立状態があります。nameはオブジェクトを参照する為にプログラムに追加される変数名です。xpos (X軸)、ypos (Y軸)、width (幅)、およびheight (高さ) は数値型で、オブジェクトの画面左上角からの位置とサイズをピクセル値で設定します。チェックマークが外れている時、Valueプロパティは0で、チェックマークが付いている時、Valueプロパティは1です。

**サポートされるプロパティ** (PROPERTIES 参照)

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

\*Valueプロパティは整数(0または1)で、TRUEかFALSEで設定もできます。TriButtonは中立状態の値"2"を持っています。

**サポートされるメソッド** (METHODS 参照)

Hide, Move, SetFocus, Show

**サポートされるイベント** (EVENTS 参照)

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

**例**

```
REM CheckBox Example
'CheckBox is a button with a check mark toggle
ADDOBJECT "CheckBox", "Check", 10, 10, 200, 20
Check.BackColor = Output.BackColor
Check.Text = "CheckBox Object"
Check.Value = TRUE
```

```
SUB Check_Click
    PRINT "CheckBox Value: " & Check.Value
END SUB
```

**結果**

☒ CheckBox Object

**例**

```
REM TriButton Example
ADDOBJECT "TriButton", "Button", 120, 120, 100, 20
Button.Value = 2 'set to disabled
```

**結果**

☒ Tri Button

**関連アイテム**

ADDOBJECT, Events, Methods, Properties

CHR(*number*)  
CHRB*number*)  
CHRW*number*)

**説明** (ANSI 文字)

指定したANSI文字コード*number*に対応する文字を返します。

CHRBは文字列に含まれるバイトデータと共に使われます。最初の文字の文字コード（1バイトまたは2バイト）を返す代わりに、必ず最初の1バイトを返します。CHRWはUnicodeを使う32ビットプラットフォーム用です。Unicodeで文字コードを返しますので、ANSIからUnicodeに変更する手間が省けます。

**例**

```
REM CHR Example
'CHR returns characters from numbers
DIM Lowercase, Uppercase
Lowercase = CHR(97)
Uppercase = CHR(97 - 32)
PRINT "Lowercase = " & Lowercase
PRINT "Uppercase = " & Uppercase
```

**結果**

```
Lowercase = a
Uppercase = A
```

**関連アイテム**

ASC

```
CLASS name  
[statements]  
END CLASS
```

**説明 (クラス)**

変数、プロパティ、メソッドと共にクラスを定義します。変数はPUBLICまたはPRIVATEとして定義します。プロパティはPROPERTY SET、PROPERTY LET、またはPROPERTY GETコードブロックとして定義します。メソッドはSUBまたはFUNCTIONブロックとして定義します。また、各クラスはインスタンスが作成された時に呼ばれるClass\_Initializeと削除された時に呼ばれるClass\_Terminateサブルーチンを持っています。Windows CE 4.0以降で使用可能。

**例**

```
Class cGreeter  
    Public Sub SayHello(who)  
        Print "Hello, " & who & ". Welcome to " & Store  
        & "."  
    End Sub  
    Public Store  
    Public Property Get StoreNumber  
        Select Case Store  
            Case "Main Branch": StoreNumber = 1  
            Case "DownTown:": StoreNumber = 2  
        End Select  
    End Property  
    Sub Class_initialize  
        Store="Main Branch"  
    End Sub  
    Sub Class_terminate  
        Print "Greeter terminated"  
    End Sub  
End Class  
  
Dim p  
Set p = new cGreeter  
p.sayHello("Cartman")  
Print p.StoreNumber  
Set p=Nothing
```

**結果**

```
Hello Cartman. Welcome to Main Branch.  
1  
Greeter terminated
```



関連アイテム  
DIM, FUNCTION, PROPERTY, SET, SUB, WITH

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

#### 説明 (コンボボックス)

ComboBoxは、ドロップダウンリストオブジェクトを表示するのに使用され、アウトプットウィンドウ上には、選択されている1ライン (エントリーライン) が現れます。*name* はオブジェクトを参照する為にプログラムに追加される変数名です。必要なコンポーネント *xpos* (X軸)、*ypos* (Y軸)、*width* (幅)、および *height* (高さ) は数値型で、オブジェクトの画面左上角からの位置とサイズをピクセル値で設定します。TextとCaptionプロパティはオブジェクトのエントリーラインの値を保持します。ListIndexプロパティは、リスト中のエントリーラインのインデックス番号です。最初の項目のインデックス番号は0です。エントリーラインのテキストがリスト中の項目のどれとも一致しない場合、ListIndexのプロパティは-1です。オブジェクトのheightはリストが広がった時のサイズを入力してください。

#### サポートされるプロパティ (PROPERTIES 参照)

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

#### サポートされるメソッド (METHODS 参照)

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

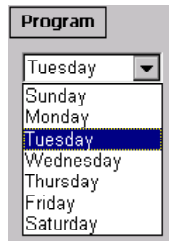
#### サポートされるイベント (EVENTS 参照)

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

**例**

```
REM ComboBox Example
'ComboBox is a drop-down list object
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)
```

**結果**



**関連アイテム**

ADDOBJECT, Events, ListBox, Methods, Properties

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

**説明** (ボタン作成)

CommandButtonは、画面上にボタンを作成します。  
*name*は、オブジェクトを参照する為にプログラムに付加されるオブジェクト名です。*xpos* (X軸)、*ypos* (Y軸)、*width* (幅)、および*height* (高さ) は数値型で、オブジェクトの画面左上コーナーからの位置とサイズをピクセル値で設定します。

Valueプロパティを 1 に設定すると、ボタンを反転させ、Clickイベントを引き起こし、再度Valueを 0 に戻します。

**サポートされるプロパティ** (PROPERTIES 参照)

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

**サポートされるメソッド** (METHODS 参照)

Hide, Move, SetFocus, Show

**サポートされるイベント** (EVENTS 参照)

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

#### 例

```
REM CommandButton Example
'CommandButton is a standard button object
ADDOBJECT "CommandButton", "Button", _
    100, 10, 80, 20
Button.BackColor = Output.BackColor
Button.Text = "Click Me!"
Output.BackColor = vbWHITE

SUB Button_Click
    PRINT "Button clicked"
    KillFocus
END SUB
```

#### 結果



#### 関連アイテム

ADDOBJECT, Events, Methods, Properties

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

**説明（定数の定義）**

CONSTはリテラルにとって代わる定数を定義します。それらは式の中で使うことができます。これは必要なコンポーネント *name* は定義する定数の名前を指定します。必要なコンポーネント *expression* は、文字、定数、算術または論理演算子（ISは除く）を含んだどのような結合でもかまいません。

オプションのPUBLIC及びPRIVATEはスクリプトレベルで使用され、参照可能な範囲を決めるものです。デフォルト定数はPUBLICであり、スクリプト内、さらに全てのプロシージャで参照可能です。プロシージャの中で定義されたすべての定数は、プロシージャ内だけで参照可能です。PRIVATEは、定義が行われるモジュール内のプロシージャからのみ参照できる定数を定義するときに指定します。

複数の定数は、カンマによって分離することによって単一のラインに定義できます。PUBLICまたはPRIVATEキーワードがこの方法で使われる場合、そのラインに定義されたすべての定数に適用されます。

**例**

```
REM CONST Example
'CONST defines constants
CONST SHAPE = "Rectangle"
PRIVATE CONST AREA = 51
PUBLIC CONST LENGTH = 7, WIDTH = 11
PrintArea LENGTH, WIDTH

SUB PrintArea(l, w)
  DIM Area
  Area = l * w
  PRINT SHAPE & " area: " & l & " * " & w & _
    " = " & Area
END SUB
```

**結果**

```
Rectangle area: 7 * 11 = 77
```

**関連アイテム**

FUNCTION, PUBLIC, PRIVATE, SUB

CBOOL(expression)  
CBYTE(expression)  
CCUR(expression)  
CDATE(expression)  
CDBL(expression)  
CINT(expression)  
CLNG(expression)  
CSNG(expression)  
CSTR(expression)

**説明（タイプ変換）**

各変換機能は、適切なタイプに変換された式を戻します。必要なパラメータ *expression* は有効であればどんな式でもかまいません。

もし戻り値が、その戻り値のタイプで容認できる範囲外であれば、エラーが起こります。

**Table 5: Conversion Functions**

Function	戻り値	説明
CBOOL	Boolean	式が 0 の時はFALSE、それ以外はTRUE
CBYTE	Byte	0から255の全ての整数
CCUR	Currency	通貨データ
CDATE	Date	100年1月1日から9999年12月31日までの日付。 有効な式は日付型、または日付／時間リテラル

Table 5: Conversion Functions

Function	戻り値	説明
CDBL	Double	負の数は、 -1.79769313486232E308 から -4.94065645841247E-324 まで  正の数は、 4.94065645841247E-324から 1.79769313486232E308まで
CINT	Integer	小数点以下 (fp) の-32,768から 32,767までの数値は、以下の様 に概算される  fp < 0.5 切り捨て  fp > 0.5 切り上げ  fp = 0.5 最も近い偶数に切り上 げ、または切り捨て
CLNG	Long Integer	小数点以下 (fp) の -2,147,483,648から 2,147,483,647までの数値は、 以下の様に概算される  fp < 0.5 切り捨て  fp > 0.5 切り上げ  fp = 0.5 最も近い偶数に切り上 げ、または切り捨て
CSNG	Single	-3.403823E38 から -1.401298E-45 までの負の数 と、1.401298E-45 から 3.403823E38 までの正の数
CSTR	String	•"TRUE"か"FALSE"に変換され るブール型  •Short-date型に変換される 日付  •"Error<数>"に変換されるエ ラー  •文字列として表記される数値



#### 例

REM Conversion Functions Example

```
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)
```

#### 結果

```
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
```

#### 関連アイテム

Is Functions

COS(*number*)

**説明** (コサイン/余弦)

*number*のコサインをラジアンで戻す数値演算関数です。  
COSは、引数として角度を受け取り、その角度を含む直角三角形の2辺の比を戻します。ここでいう2辺とは、引数 *number*に指定した角をはさむ2つの辺を指します。2辺の比は、短い方の辺(底辺)の長さを、もう一方の辺(斜辺)の長さで割った値です。戻り値は、-1 ~ 1の範囲の値になります。

角度の単位を度からラジアンに変換するには、度に  $\pi/180$  を掛けます。

ラジアンから度に変換するには、ラジアンに  $180/\pi$  を掛けます。

**例**

```
REM COS Example
'COS calculates the cosine of a number
PRINT "The cosine of 0 is " & COS(0)
```

**結果**

The cosine of 0 is 1

**関連アイテム**

SIN, TAN

CurrentPath

**説明** (プログラムへのパス)

CurrentPathはプログラムへのパスを文字列で戻します。プログラムがまだ保存されていない場合、戻り値は空です。このグローバルプロパティは、読み込み専用です。

**例**

```
REM CurrentPath Example
PRINT "The path to the current program is " &
CurrentPath
```

**結果**

The path to the current program is \test.nsb

**関連アイテム**

## DATE

**説明** （現在の日付）  
DATEはシステムの現在の日付を戻します。

**例**

```
REM DATE Example
'DATE returns current system date
DIM Today
Today = DATE
PRINT "Today is " & Today
```

**結果**

Today is 8/18/1998  
(結果はシステムの設定により異なります。)

**関連アイテム**

NOW, TIME

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

**説明** (日付オブジェクト)

Dateはアウトプットウィンドウに、標準日付ピッカーオブジェクトを表示するのに使用されます。*name*は、オブジェクトを参照する為にプログラムに追加される変数名です。*xpos* (X軸)、*ypos* (Y軸)、*width* (幅)、および*height* (高さ)は数値型で、オブジェクトの画面左上角からの位置とサイズをピクセル値で設定します。

Date\_Changeイベントの中でMSGBOXを使用しないで下さい: エラーを引き起こします。このオブジェクトはWindows CE 2.0機では使用できません。

**サポートされるプロパティ** (PROPERTIES 参照)

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

**サポートされるメソッド** (METHODS 参照)

Hide, Move, SetFocus, Show

**サポートされるイベント** (EVENTS 参照)

Change, DropDown

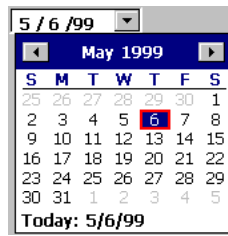
**例**

```
REM Date Example
ADDOBJECT "Date", "Date", 0,0,90,20
Date.fontBold=true

SUB Date_Dropdown
    PRINT "Date Dropdown"
END SUB

SUB Date_Change
    PRINT "Date Changed to " & Date.Date
END SUB
```

結果



関連アイテム

ADDOBJECT, Events, Methods, Properties

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

**説明**（日時の加減算）

*date*の日付より*interval*の間隔で*number*の数だけ進めた日付を戻します。パラメータ*interval*は日付のタイプを示します。Table 6を参照して下さい。*number*は数値型で、*date*は日付型です。

**Table 6: Interval values**

値	説明
yyyy	年 (Year)
q	1/4期 (Quarter)
m	月 (Month)
y	年を通した日 (Day of year)
d	日 (Day)
w	平日 (Weekday)
ww	週 (Week of year)
h	時 (Hour)
n	分 (Minute)
s	秒 (Second)

**例**

REM DATEADD Example

'DATEADD adds date intervals to a date

PRINT "+10 seconds:", DATEADD("s", 10, NOW)

PRINT "-1 year:", DATEADD("yyyy", -1, NOW)

**結果**

+10 seconds: 8/18/98 10:52:54 PM

-1 year: 8/18/97 10:52:44 PM

**関連アイテム**

DATEDIFF, DATEPART

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

**説明**（日時間隔）

DATEDIFFは2つの日付の間隔数を戻します。*interval*は文字列型です。Table 6を参照して下さい。*date1*および*date2*は日付型です。オプションのパラメータ*firstdayofweek*は、もし指定されないならば日曜日です。オプションの*firstweekofyear*は1月1日を含んでいる週です。

**Table 7: firstdayofweek 定数**

定数	値	説明
vbUseSystem	0	NLS API 設定
vbSunday	1	日曜日（デフォルト）
vbMonday	2	月曜日
vbTuesday	3	火曜日
vbWednesday	4	水曜日
vbThursday	5	木曜日
vbFriday	6	金曜日
vbSaturday	7	土曜日

**Table 8: firstweekofyear 定数**

定数	値	説明
vbUseSystem	0	NLS API 設定
vbFirstJan1	1	1 月 1 日の週
vbFirstFourDays	2	少なくとも 4 日以上ある 年始めの週
vbFirstFullWeek	3	年始めのまる一週間



#### 例

```
REM DATEDIFF Example
'DATEDIFF calculate difference between 2 dates
DIM Born
Born = INPUTBOX("Enter your birthdate")
Born = CDATE(Born)
PRINT "Since " & Born & " there have been"
PRINT DATEDIFF("d", Born, NOW) & " days"
PRINT "or"
PRINT DATEDIFF("n", Born, NOW) & " minutes"
```

#### 結果

```
Since 12/27/1970 there have been
10096 days
or
14539612 minutes
(結果はシステムの設定により異なります)
```

#### 関連アイテム

DATEADD, DATEPART

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

**説明** (部分的日付)

DATEPARTは*date*で指定する日付をタイプ毎に戻します。  
*interval*は結果のタイプ指定をします。オプションのパラメータ*firstdayofweek*は、指定されなければ日曜日です。オプションの*firstweekofyear*は指定されなければ1月1日の週です。

**例**

```
REM DATEPART Example
'DATEPART returns a number from part of a date
DIM QuarterPart, MonthPart, DayPart
QuarterPart = DATEPART("q", NOW)
MonthPart = DATEPART("m", NOW)
DayPart = DATEPART("d", NOW)
PRINT "Today is day " & DayPart
PRINT "of month " & MonthPart
PRINT "in quarter " & QuarterPart
```

**結果**

```
Today is day 18
of month 8
in quarter 3
(例題の結果は1998年8月18日のものです)
```

**関連アイテム**

DATEADD, DATEDIFF

DATESERIAL(*year, month, day*)

説明 (年、月、日による日付構成)

DATESERIALは、与えられた年、月、日から構成された日付を戻します。必要なパラメータ *year* は、100から9999までの範囲の数値型です。必要なパラメータ *month* および *day* は数値型です。

例

```
REM DATESERIAL Example
'DATESERIAL builds a date from its parts
DIM IndepDay, Birthday
IndepDay = DATESERIAL(1776, 7, 4)
Birthday = DATESERIAL(1970, 12, 27)
PRINT "Independence Day:", IndepDay
PRINT "My birthday:", Birthday
```

結果

```
Independence Day: 7/4/1776
My birthday: 12/27/1970
(結果はシステムの設定により異なります)
```

関連アイテム

TIMESERIAL

DATEVALUE(*date*)

**説明** (文字列による日付構成)

DATEVALUEは日付を戻します。必要なパラメータ*date*は通常文字列ですが、日付型、時間型、またはJanuary 1, 100からDecember 31, 9999までの日付、時間、または日付と時間を表わす式が使えます。

*date*が、日付セパレータにより分離された数から成っている文字列の場合、システムに指定された日付フォーマットに従って年、月、日、を戻します。*date*の中の年が省略された場合、現在の年が使われます。明確な月の名前はlong値または日付ショートフォーマットによって認識されます。

日付セパレータは、日付が文字列にフォーマットされる時、年、月、日を分離する文字で、お使いのシステム設定により決められます。

**例**

```
REM DATEVALUE Example
'DATEVALUE returns a date
DIM IndepDay, Birthday
IndepDay = DATEVALUE("July 4, 1776")
Birthday = DATEVALUE("Dec 27 1970")
PRINT "Independence Day:", IndepDay
PRINT "My birthday:", Birthday
```

**結果**

```
Independence Day: 7/4/1776
My birthday: 12/27/1970
(結果はシステムの設定により異なります)
```

**関連アイテム**

FORMAT, TIMEVALUE

DAY(*date*)

説明（日付：日）

DAYは、与えられた日付の日（整数1から31までの範囲）を返します。必要なパラメータ *date* は日付型です。

例

```
REM DAY Example
'DAY returns number representing day of month
DIM Today, NextQuarter
Today = DATE
NextQuarter = Today + 90
PRINT "Today's day is " & DAY(Today)
PRINT "90 days from now it will be " _
      & DAY(NextQuarter)
```

結果

```
Today's day is 18
90 days from now it will be 16
```

関連アイテム

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

```
DECLARE "Sub name Lib ""libname"" [Alias ""aliasname""]  
[[arglist]]"  
Declare "Function name Lib ""libname"" [Alias ""aliasname""]  
[[arglist]] [As type]"
```

#### 説明（APIファンクションの定義）

*name*はNS Basicプログラム内で使うファンクション名またはサブルーチン名。DLL内部の名前が異なる場合、*Alias*を使ってファンクションの内部名を与えます。これは内部名が重なってしまう時に便利です。*libname*はDLLライブラリ名。これはDLLへのフルパスまたは部分パスです。また、単にファイル名の場合はシステムがDLLを検索します。ほとんどのDLLはWindows ディレクトリに保存され、この場合、パスは省略することができ、システムが迅速に探し当ててくれます。詳細はDeclareのTech Noteを参照してください。

#### 例

```
Rem Get storage info on device  
Dim lpAvailable, lpTotalBytes, lpTotalFree  
Declare "Function 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 "Available: " & lpAvailable
```

#### 結果

Available: 123,000,004

#### 関連アイテム

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

**説明（変数定義及びメモリスペース確保）**

DIMは変数を定義し、保管スペースを割り当てるために使用されます。必要なコンポーネント *nameA* は変数名です。オプションのリストは配列変数の次元の上限です。カンマで分離された、最高で60までの次元が定義可能です。スクリプトレベルの変数は、スクリプト内の全てのプロシージャに有効であり、プロシージャレベルの変数は定義されたプロシージャ内のみで有効です。配列の下限は常に0です。

**例**

```
REM DIM Example  
'DIM declares variables and allocates storage  
'An empty variable named "Foo"  
DIM Foo  
'A one-dimensional array with 10 elements  
DIM OneD(9)  
'A two-dimensional array with 600 elements  
'20 x 30  
DIM TwoD(19, 29)
```

**関連アイテム**

ARRAY, REDIM, SET

## DOEVENTS

## 説明（割り込みイベント）

長いループの処理中に割り込んでイベントを処理します。  
DOEVENTSが呼ばれると、全てのスクリーンまたはTimerイベントは実行され、ループは引き続き処理されます。

## 例

```
REM DOEVENTS Example
'DOEVENTS allows screen events to occur
'during long loops
FOR i = 0 TO 100000
    j = j + 1
    DOEVENTS
NEXT
```

## 結果

(何もありませんが、イベントが起こります)

## 関連アイテム



```
DO [WHILE | UNTIL condition]  
    [statements]  
    [EXIT DO]  
    [statements]  
LOOP  
  
または  
  
DO  
    [statements]  
    [EXIT DO]  
    [statements]  
LOOP [WHILE | UNTIL condition]
```

**説明 (DO...LOOP ループ)**

DO...LOOPステートメントはWHILEで`condition`がTRUEの間、もしくはUNTILで`condition`がTRUEになるまでループを繰り返します。EXIT DOはDOとLOOPの間にはないといけません。必要なコンポーネント `condition`はTRUEまたはFALSEの結果を生む式です。`statements`はループ毎に実行されるステートメントの固まりです。DO...LOOPは別のループステートメント内に書くことが出来ます。その場合のEXIT DOは1つのレベルのみ抜け出します。

DOのすぐ後にWHILE/UNTILを置くと、最初のループが実行される前に`condition`が評価されます。`condition`がFALSEになる場合、`statements`部分は1度も実行されません。

LOOPの後にWHILE/UNTILを置く場合、`condition`が評価される前に`statements`部分を実行しますので、少なくとも最低1回は`statements`部分が実行されます。

#### 例

```
REM DO..LOOP Example
'DO..LOOP repeats a block of statements
DIM Counter
Counter = 1
'Loop that prints 1 to 5
DO WHILE Counter < 6
    PRINT "DO WHILE..LOOP:", Counter
    Counter = Counter + 1
LOOP
PRINT
'Infinite loop that uses EXIT DO to terminate
Counter = 1000
DO
    PRINT "Infinite Loop:", Counter
    IF Counter >= 1000000 THEN
        EXIT DO
    Counter = Counter * 10
LOOP
```

#### 結果

```
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
```

#### 関連アイテム

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

*result* = *x* EQV *y*

**説明**（論理等価演算）

EQVは2つの式の論理的な等価を戻します。式*x*と*y*の両方がTRUE、または両方がFALSEの場合、結果はTRUEです。それ以外はFALSEです。

EQVは数値のビット単位の比較も行います。*result*の各ビットには、相対する*x*と*y*のビットが共に1か共に0の場合、1がセットされ、それ以外は0がセットされます。

**例**

```
REM EQV Example
'EQV performs logical and bitwise equivalence
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 "Logical:"
PRINT "  x < 0 EQV y < 10 = " & Test1
PRINT "  x > 0 EQV y > 10 = " & Test2
PRINT "  x < 0 EQV y > 10 = " & Test3
PRINT "Bitwise:"
PRINT "  x EQV y = " & (x EQV y)
```

**結果**

```
Logical:
  x < 0 EQV y < 10 = False
  x > 0 EQV y > 10 = False
  x < 0 EQV y > 10 = True
Bitwise:
  x EQV y = -14
```

**関連アイテム**

AND, IMP, NOT, OR, XOR

ERASE *arrays*

**説明** (配列の解放)

固定配列が指定された場合、配列を再初期化し、ダイナミック配列が指定された場合、割り当てられていた、メモリーを解放します。必要なコンポーネント *arrays* は、カンマで区切られた1つ以上の配列変数のリストです。

**例**

```
REM ERASE Example
'ERASE reinitializes arrays
DIM Children(3)
Children(0) = "す"
Children(1) = "き"
Children(2) = "で"
Children(3) = "す"
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
```

**結果**

```
#1: (す)
#2: (き)
#3: (で)
#4: (す)
```

```
#1: ()
#2: ()
#3: ()
#4: ()
```

**関連アイテム**  
DIM, ARRAY

Err  
Err.Number  
Err.Description  
Err.Source  
Err.Clear  
Err.Raise

**説明（エラー処理）**

ERRオブジェクトは、プログラム実行時にエラーを処理するために使用します。ERRオブジェクトはプログラムが実行中は常に利用可能です。ADDOBJECTを使う必要はありません。エラーが起こる時には、NUMBER、DESCRIPTION、およびSOURCEプロパティがERRオブジェクトに設定されます。一旦エラーが処理されたら、Clearメソッドはプロパティ内のエラーデータを削除してErrオブジェクトをリセットします。Raiseメソッドを使うことによりエラーイベントを発生させることが出来ます。

**例**

```
REM Err Object Example
'Err object manages run-time errors
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
```

**結果**

```
11 Division by zero
```

**関連アイテム**

ON ERROR

ESCAPE(*string*)  
UNESCAPE(*string*)

**説明**（特殊文字のHEX値）

ESCAPEは全ての特殊文字を%文字に続くhex値に変換した文字列を返します。UNESCAPEはその逆です。Windows CE 4.0以降のみ利用可能。

**例**

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

**結果**

ABC%21@%23%24%25

ABC

DEF

**関連アイテム**

ESCAPE, UNESCAPE

EVAL(*string*)

説明（文字列の関数評価）

EVALは、*string*を関数のように実行することによって作成された値を戻します。必要なパラメータ*string*は実行される文字列型です。複数のステートメントを実行する場合は、キャリッジ・リターン（vbCRLF）によってそれらを分離します。この一時的なバーチャルプロシージャは、値として渡される全ての変数を保存しますので、EVALファンクションでは変数の値を変更することが出来ません。

例

```
REM EVAL Example
'EVAL execute a string as a FUNCTION
DIM x
x = 5
PRINT EVAL("x")
EVAL("x = x * 10")
PRINT x
```

結果

5  
5

関連アイテム

EXECUTE

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

#### 説明（イベントオブジェクト）

オブジェクトのイベントはプログラムで引き起こすか、ユーザの動作によって引き起こされます。イベントが引き起こされる時、オブジェクトはプログラムにあるPUBLIC SUBプロシージャを呼びます。ObjectName\_Eventというプロシージャの名前は、オブジェクト名とイベント名の組み合わせです。オプションのコンポーネントarglistは、カンマで区切られたパラメータリストで、イベントによってプロシージャに含むことができます。以下のTable 9は一般的なオブジェクトイベントのリストです。

#### 注意:

イベントに対応するプロシージャがプログラムに含まれていない場合、とくにエラーは発生しません。

Table 9: オブジェクトイベント

イベント	コメント
Change	ComboBox (アイテムの選択かテキスト入力), ListBox (アイテムの選択), TextBox (テキスト変更)
Click	
DbClick	ListBox, TextBox
DropDown	ComboBox, Date
GotFocus	オブジェクトがフォーカスを得た（キーボード入力を受ける状態）
KeyDown	Keycode, shift as args Shift=1, 2-CTRL, 4=Alt
KeyPress	Char as arg
KeyUp	Keycode, shift as args Shift=1, 2-CTRL, 4=Alt
LostFocus	オブジェクトがフォーカスを失った
Timer	CheckBox, ComboBox, CommandButton, Frame, HScrollBar, Label, ListBox, OptionButton, TextBox, VScrollBar



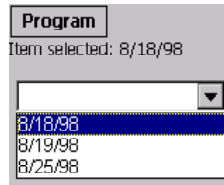
#### 例

```
REM Object Events Example
'Object Events call procedures in the program
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 "Item selected: " & When
END SUB
```

#### 結果



#### 関連アイテム

Methods, Properties

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

**説明**（式又はファイルの実行）  
EXECUTEは、式やファイルをそれがプログラムに付加されたコードかのように実行します。必要なパラメータ *string* は実行される文字列式またはステートメントを含むファイルの名前です。複数のステートメントを実行したい場合、キャリッジ・リターン（vbCRLF）によってそれらを分離します。EXECUTEは、実行中のプログラムの全ての変数にアクセスしたり、変更が可能です。

"file:"または"ascii:"に続く文字列はアスキーファイルへのパス名です。"unicode:"はユニコードフォーマットのファイルに使用されます。ファイル名機能はNS Basic/CE開発用で実行時用ではありません。

**例**

```
REM EXECUTE Example  
'EXECUTE execute a string as a SUB  
DIM x  
x = 5  
EXECUTE("PRINT x * 10")  
EXECUTE("x = x * 10")  
'text.txt is a file with "PRINT 50" in it  
EXECUTE("ascii:\my documents\test.txt")  
PRINT x
```

**結果**

```
50  
50  
50
```

**関連アイテム**

EVAL, EXECUTEGLOBAL

EXECUTEGLOBAL(*string*)

**説明** (グローバル領域で式又はファイルの実行)  
グローバル領域にある1つまたは複数のステートメントを実行します。*String*は文字列式または実行されるステートメントが含まれたファイル名。複数のステートメントを実行する場合、キャレットリターン(vbCRLF)またはコロンによって区切ります。EXECUTEGLOBALによって処理される全てのステートメントは、プログラムのグローバル領域で実行されます。従って、全てのプロシージャがアクセスできるコードをプログラムに追加できます。例えば、Classステートメントを実行時に処理し、その後、ファンクションがそのクラスのインスタンスを作成することなどが可能です。プロシージャやクラスを実行時に追加するのは便利ですが、グローバル変数やファンクションを実行時に上書きする可能性も生まれます。プロシージャの外にある変数やファンクションにアクセスする必要がない場合は、EXECUTEステートメントを使用する方が良いでしょう。

**例**

```
REM EXECUTEGLOBAL Example
'EXECUTEGLOBAL executes a string as a SUB
DIM x
x = ("DIM y" & vbcrLf & "y = 4 * 10" & _
    vbcrLf & "Print y")
EXECUTEGLOBAL x
```

**結果**

40

**関連アイテム**

EVAL, EXECUTE

```
EXIT DO
EXIT FOR
EXIT FUNCTION
EXIT SUB
```

**説明** (ループから抜出す)

EXITは、DO...LOOP、FOR...NEXT、FOR EACH...NEXT、FUNCTION、またはSUBの実行を終わらせます。ループのDO...LOOP、FOR...NEXT、またはFOR EACH...NEXTを出るために使用される時には、ループ後の最初のステートメントが継続されます。ループが別のループの中で実行しているなら、EXITによって実行中のループのすぐ外に抜出し、別のブールが継続されます。FUNCTIONまたはSUBを出るために使用される時には、それを呼んだステートメントの後から継続されます。

**例**

```
REM EXIT Example
'EXIT terminates loops and procedures
DIM i
FOR i = 1 to 10
  IF i > 1 THEN
    EXIT FOR
  END IF
  PRINT "Attempting to do nothing"
  DoNothing
NEXT
PRINT "Done"

SUB DoNothing
  EXIT SUB
  PRINT "This statement is never executed"
END SUB
```

**結果**

```
Attempting to do nothing
Done
```

**関連アイテム**

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

EXP (*number*)

**説明 (指数  $e^x$ )**

EXPは $e^{\text{number}}$ で求められる指数を倍精度値 (double) で戻します。必要なパラメータ $number$ は数値型です。 $e$ は自然対数のベースであり、おおよそ2.718282です。

**例**

```
REM EXP Example
'EXP raises a number to the eth power
PRINT "EXP(0) = " & EXP(0)
PRINT "e = " & EXP(1)
```

**結果**

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

**関連アイテム**

LOG

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

説明（配列から文字列の抽出）

必要なパラメータ *stringarray* は文字列の 1 次元の配列。必要なパラメータ *value* は検索項目を示す文字列型です。オプションのパラメータ *include* に TRUE を設定すると結果は *value* の値を含むものになり、FALSE を設定すると *value* の値を含まないものを結果として戻します。Include のデフォルト値は TRUE です。オプションのパラメータ *compare* は数値または下表に示される定数名を使用して下さい。*compare* のデフォルト値は `vbBinaryCompare` です。

Table 10: 比較定数

定数	値	説明
<code>vbBinaryCompare</code>	0	ビット単位で比較するので、格（大文字／小文字）による区別をします。(デフォルト)
<code>vbTextCompare</code>	1	文章比較なので格による区別はしません。

#### 例

```
REM FILTER Example
'FILTER finds matches in an array of strings
DIM Who, TheKs, NotEric
Who = ARRAY("Eric", "Kenny", "Kyle", "Stan")
TheKs = FILTER(Who, "k", TRUE, vbTextCompare)
NotEric = FILTER(Who, "Eric", FALSE, _
    vbBinaryCompare)
PrintArray "Who", Who
PrintArray "The K's", TheKs
PrintArray "Everyone but Eric", NotEric

SUB PrintArray(ArrName, Arr)
    DIM i
    PRINT ArrName
    FOR i = 0 TO UBOUND(Arr)
        PRINT "  " & Arr(i)
    NEXT
END SUB
```

#### 結果

```
Who
  Eric
  Kenny
  Kyle
  Stan
The K's
  Kenny
  Kyle
Everyone but Eric
  Kenny
  Kyle
  Stan
```

#### 関連アイテム

REPLACE

FIX(*number*)

**説明**（端数の省略）

FIXは、数字の端数部分を削除し、0方向へ最も近い整数を返します。必要なパラメータ*number*は数値型です。*number*が正数の場合、次に小さな整数を返し、負数の場合は、次に大きな整数を返します。

**例**

```
REM FIX Example
'FIX converts floats to int nearest 0
DIM Pos, Neg
Pos = EXP(1)
Neg = -EXP(1)
PRINT "FIX(e) = " & FIX(Pos)
PRINT "FIX(-e) = " & FIX(Neg)
```

**結果**

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

**関連アイテム**

INT



```
FOR counter = start TO end [STEP step]
    [statements]
[EXIT FOR]
    [statements]
NEXT
```

**説明** (FOR...NEXT ループ)

FOR...NEXTは指定した回数だけ *statements* の文列を繰り返します。必要なコンポーネント *counter* は、現在の繰り返し回数を参照できる数値です。必要なコンポーネント *start* と *end* は始まりと終わりの値を示す整数型です。オプションのパラメータ *step* (デフォルトは1) は、*counter* の加算値をセットするために使用できます。オプションのEXIT FORステートメントは途中でループを終了するため使用され、複数存在しても構いません。FOR...NEXTは別のループステートメント内に書くことが出来ます。その場合のEXIT FORは1つのレベルのみ抜け出します。

**例**

```
REM FOR..NEXT Example
'FOR..NEXT repeats a group of statements
DIM Puppets
Puppets = ARRAY("Hat", "Twig")
FOR i = 0 to 1
    PRINT "Puppet: Mr. " & Puppets(i)
NEXT
FOR i = 0 to 10 STEP 5
    PRINT i
NEXT
```

**結果**

```
Puppet: Mr. Hat
Puppet: Mr. Twig
0
5
10
```

**関連アイテム**

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

```
FOR EACH element IN group
    [statements]
    [EXIT FOR]
    [statements]
NEXT [element]
```

**説明（配列の巡回）**

FOR EACH...NEXTは、配列又はコレクションの各要素ごとに*statements*を実行します。必要なパラメータ*element*は、実行中それぞれの要素を参照する変数名です。必要なパラメータ*group*は、配列名またはオブジェクトのコレクション名です。オプションのEXIT FORステートメントは途中でループを終了するため使用され、複数存在しても構いません。FOR...NEXTは別のループステートメント内に書くことが出来ます。その場合のEXIT FORは1つのレベルのみ抜け出します。

注) FOR EACH...NEXTは、ユーザーによって定義されたタイプの配列では機能しません。

**例**

```
REM FOR EACH...NEXT Example
DIM School
School = ARRAY("Principal", "Mr. Garrison", _
    "Chef")
FOR EACH Employee IN School
    PRINT "School employee:", Employee
NEXT
```

**結果**

```
School employee: Principal
School employee: Mr. Garrison
School employee: Chef
```

**関連アイテム**

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

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

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

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

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

#### 説明（フォーマット）

FORMATCURRENCY、FORMATNUMBER、およびFORMATPERCENTは、与えられた値を通貨、数、またはパーセントにフォーマットしたものを文字列にして戻します。必要なパラメータ*expression*は、与えられたタイプの式です。オプションのパラメータ*fractionaldigits*は、小数点以下を何桁にするかを設定します。デフォルト値は-1で、これはシステム設定を使用することを意味します。オプションのパラメータ*leadingdigit*、*parensfornegative*及び*groupdigits*は数の変数であるが、下表に示される定数をお使い下さい。*leadingdigit*は小数点以下のみに数がある値に対し、頭に0（ゼロ）を付けるかどうかを指定します。*parensfornegative*は負数の値に包囲括弧を付加するかどうかを指定します。*groupdigits*はシステムが設定する数値の区分け方法を使用するかどうか指定します。

Table 11: Tristate値

定数	値	説明
True	-1	True
False	0	False
UseDefault	-2	システム設定使用

FORMATDATETIMEは、日付をフォーマットすることによって得られた文字列を戻します。必要なパラメータ*date*は日付型です。オプションのパラメータ*formatname*は、フォーマット形式を選ぶもので、下表で示されている数値または定数を指定して下さい。

Table 12: formatname 定数

定数	値	説明
vbGeneralDate	0	Short date, Long time
vbLongDate	1	Long date
vbShortDate	2	Short date
vbLongTime	3	Long time
vbShortTime	4	Short time

**例**

```
REM Format Functions
DIM UseDefault
UseDefault=-2
'Currency
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)
```

**結果**

```
($3.50)
$123,456
8/18/1998 10:44 PM
August 18, 1998
-.142900
987,654.321
.70%
123456%
(結果はシステムにより異なります)
```

**関連アイテム**

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

#### 説明 (フレーム)

フレームは他のオブジェクトを含むことができる枠です。オブジェクトをフレームの子供にする (内側に含める) には、そのオブジェクトのParentHwndプロパティに、フレームのHwndを設定します。フレームに適用されるメソッドは、自動的にその子供に適用されます。子供の境界は、それを含むフレームに直接関係します。

#### サポートされるプロパティ (PROPERTIES 参照)

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

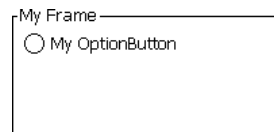
#### サポートされるメソッド (METHODS 参照)

Hide, Move, Show

#### 例

```
REM Frame Example
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
```

#### 結果



#### 関連アイテム

ADDOBJECT, Events, Methods, Properties

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

**説明 (FUNCTIONプロシージャ)**

FUNCTIONはプロシージャを作成しボディーステートメント、パラメータ及び戻り値を設定します。必要なパラメータ *procedurename* はプロシージャが呼ばれる時に使われ、標準可変命名慣習（半角英数文字）に従わなければなりません。オプションのパラメータリスト *arglist* は、プロシージャが呼ばれる時に渡される変数であり、カンマで区切られたリストです。オプションの *statements* はプロシージャのボディー部分として実行されます。EXIT FUNCTION ステートメントはプロシージャを抜出す為に使用され、複数存在できません。プロシージャから返還されるデフォルト値はEMPTYです。デフォルト以外の値を戻す場合は [*procedurename* = 値] を使い、全てのステートメントが実行された時か、処理がEXIT FUNCTIONに到達した時にその値が戻されます。

*arglist* の個々の引数は、下で定義されるようにプロシージャに手渡されます：

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

BYVAL: 引数 *varname* が値渡しで渡されることを示し、オリジナルの値の変更は不可能です。BYREF: 引数 *varname* が参照渡しで渡されることを示し、オリジナルの値はプロシージャ内から変更可能です。デフォルトはBYREFです。

注) 戻される値が変数に代入されない場合や、使用されない場合は、プロシージャはSUBプロシージャと呼ばれ複数の引数を括弧内に入れることは出来ません。

#### 例

```
REM FUNCTION Example
'FUNCTION: a procedure that returns a value
DIM Selection, SalePrice
Selection = Menu("Wednesday")
PRINT "Wednesday's menu feature:", Selection
SalePrice = Min(31, 29)
PRINT "Sale Price:", SalePrice
```

```
FUNCTION Menu(day)
  IF day = "Wednesday" 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
```

#### 結果

```
Wednesday's menu feature: Salisbury Steak
Sale Price: 29
```

#### 関連アイテム

CALL, SUB

## GETCOMMANDLINE

## 説明（コマンドライン）

GETCOMMANDLINEはプログラムを起動するために使われたコマンドラインテキストを戻します。プログラムがそれ自身で起動した場合、戻るテキストはプログラムのパスです。プログラムがそれに伴うドキュメントによって起動された場合、戻るテキストはプログラムのパスとドキュメントのパスです。

## 例

```
REM GETCOMMANDLINE Example
'GETCOMMANDLINE returns launch text
c1 = GETCOMMANDLINE
prog = LEFT(c1, INSTR(c1, ".nsb") + 4)
doc = MID(c1, LEN(prog) + 1)
```



GETLOCALE  
SETLOCALE *localeID*

**説明（地域ID）**

GETLOCALEはLocale IDを返します。Locale ID（地域ID）はキーボード・レイアウト、アルファベットの並べ替え順、日付／時刻／通貨フォーマット等を定めるものです。SETLOCALEはLocale IDをセットし、今まで設定されていたIDを返します。*localeID*は国や地域を示す番号または短い文字列です。*localeID*を0にするとLocale IDは現在のシステム設定にセットされます。Locale IDのリストはここをご覧ください：

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

いくつかの一般的なIDは：

1031	de-de	German – Germany
1033	en-us	English – United States
1041	ja	Japanese
2057	en-uk	English - UK

Windows CE 4.0 以降のみ利用可能。

**例**

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

**結果**

```
1033  
1033 2057
```

**関連アイテム**

SETLOCALE, GETLOCALE

## GETREF

**説明** (リファレンス・ポインター)  
ファンクションまたはサブルーチンへのリファレンス・ポインターを返す。  
Windows CE 4.0 以降のみ利用可能。

**例**

```
Function refTest
    MsgBox "RefTest"
End Function

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

**結果**

```
Object
RefTest
```

**関連アイテム**

## GETSERIALNUMBER

## 説明（シリアル番号）

GETSERIALNUMBERは（もしある場合）デバイスのシリアル番号を戻します。Pocket PC 2002から付くようになった番号は、それ以降に製造されている全てのデバイスに割り当てられていると考えられます。

## 例

```
REM GETSERIALNUMBER Example
MsgBox GetSerialNumber()
```

HEX(*number*)

**説明**（16進法表現）

HEXは、16進法の値を文字列にして戻します。必要なパラメータ *number* は数値型ですが、*number* が整数でない場合は最も近い整数に丸められます。

**例**

```
REM HEX Example
'HEX returns a number as a hexadecimal string
PRINT "68 in hex:", HEX(68)
PRINT "1 in hex:", HEX(1)
PRINT "2605.45 in hex:", HEX(2605.45)
```

**結果**

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

**関連アイテム**

OCT

HOUR(*time*)

**説明**（時刻：時）

HOURは、*time*によって与えられた時間が何時を表しているか、0から23までの範囲で整数を戻します。必要なパラメータ *time* は数値型または時間型です。

**例**

```
REM HOUR Example
'HOUR returns hour of day from a time
PRINT "The HOUR of " & FormatDateTime(Now,4) & _
"is " & HOUR(Now)
```

**結果**

The HOUR of 8/18/1998 10:52:44 PM is 22  
(結果はシステムの設定により異なります)

**関連アイテム**

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

```
IF condition THEN statements [ELSE elasticsearch]
```

```
IF condition THEN  
    [statements]  
[ELSEIF condition-n THEN  
    [elseifstatements]]...  
[ELSE  
    [elasticsearch]]  
END IF
```

#### 説明 (IF...THEN...ELSE条件ステートメント)

IF...THEN...ELSEは、ひと固まりのステートメントを条件付きで実行するために使用されます。必要なコンポーネント *condition*は、TRUEまたはFALSEに評価する式であらわします。ELSEがなくIFがインラインバージョン（一行）で 사용되는場合、*statements*が必要になります。それ以外は *statements*はオプションです。条件がTRUEまたは 0 以外に評価される場合、*statements*が実行されます。条件がFALSEまたは 0 に評価される場合、ELSEIFがある場合は次のELSEIFへ飛び *condition-n*を評価し、ELSEがある場合は *elasticsearch*を実行します。

インラインバージョンにおいて複数のステートメントを実行するには、ステートメントはコロン(:)によって分離されなければなりません。またインラインバージョンの中で、一つのプロシージャを引数なしで呼ぶ場合、そのプロシージャは空の括弧が伴わなければなりません。

#### 例

```
REM IF...THEN...ELSE Example  
'IF...THEN...ELSE performs conditional execution  
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
```

#### 結果

```
TRUE
```

#### 関連アイテム

```
SELECT CASE
```

`result = x IMP y`

#### 説明（論理包含演算）

IMPは2つ式（ $x$ と $y$ ）の論理包含演算を行います。戻り値は以下の表をご覧ください。IMPは2つの数のビット単位の比較も行います。

**Table 13: 論理包含(Logical Implication)**

x	y	結果
TRUE/1	TRUE/1	TRUE/1
TRUE/1	FALSE/0	FALSE/0
FALSE/0	TRUE/1	TRUE/1
FALSE/0	FALSE/0	TRUE/1

#### 例

```
REM IMP Example
'IMP preforms logical and bitwise implication
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 "Logical:"
PRINT "  x < 0 IMP y < 10 = " & Test1
PRINT "  x > 0 IMP y > 10 = " & Test2
PRINT "  x < 0 IMP y > 10 = " & Test3
PRINT "Bitwise:"
PRINT "  x IMP y = " & (x IMP y)
```

#### 結果

```
Logical:
  x < 0 IMP y < 10 = True
  x > 0 IMP y > 10 = False
  x < 0 IMP y > 10 = True
Bitwise:
  x IMP y = -3
```

#### 関連アイテム

AND, EQV, NOT, OR, XOR

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

**説明** (インプットボックス)

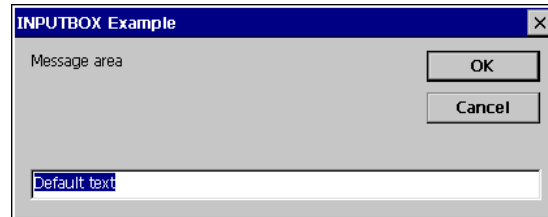
INPUTBOXは、ユーザにテキスト入力またはボタンのクリックを促す為に、ダイアログボックスを開きます。ダイアログのテキストフィールドからの内容は文字列として戻されます。必要なパラメータpromptは、ダイアログボックスのボディに表示される文字列型です。オプションのパラメータtitleは、ダイアログボックスのタイトル・バーに表示される文字列型です。オプションのパラメータdefaultは、ダイアログボックスのテキストフィールドに表示される文字列型です。オプションのパラメータxposおよびyposは、スクリーンの左上コーナーとダイアログボックスの左上コーナーの間距離を指定する数値型です。

ユーザがOKを指定するかEnterキーを押した場合、INPUTBOXはテキストフィールドの文章を戻すか、何も入力されていない場合は空の文字列("")を戻します。Cancelが指定されるかエスケープキー(ESC)が押された場合はINPUTBOXはEMPTYを戻します。

**例**

```
REM INPUTBOX Example
DIM Return
Return = INPUTBOX("Message area", _
    "INPUTBOX Example", "Default text")
PRINT "You entered:", Return
```

**結果**



You entered: Default Text

**関連アイテム**



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

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

**説明** (文字列の検索)

INSTRは、文字列string1をstartで示されたポジションから検索し、文字列string2が見つかるまでの文字数をlong値で返します。

INSTRREVはINSTRとは逆に文字列の終わりから検索します。

startが指定されない場合、INSTRのデフォルト値 0 により、最初の文字のポジションを示し、INSTRREVはデフォルト値 -1 により、最後の文字のポジションを示します。オプションのcompareは検索のタイプを指定します。

(Table 10を参照) string2がstring1内に見つかった場合は正数が返され、見つからなかった場合は 0 が返されます。

**例**

```
REM INSTR/INSTRREV Example
'INSTR/INSTRREV finds one string in another
DIM Pos1, Pos2
Pos1 = INSTR("Cartman", "man")
Pos2 = INSTRREV("Big Al's Big Boat Ride", _
    "big", 4, vbTextCompare)
PRINT "Finding \"man\" from start:", Pos1
PRINT "Finding \"big\" from end:", Pos2
```

**結果**

```
Finding "man":    5
Finding "big":    1
```

**関連アイテム**

INT(*number*)

**説明**（整数へ変換）

INTは与えられた数の端数部分を切り捨て、それより小さな整数へ丸め、その結果を戻します。必要なパラメータ *number* は数値型です。

**例**

```
REM INT Example
'INT converts floats to the next smallest int
DIM Pos, Neg
Pos = ATN(1) * 4
Neg = -Pos
PRINT "INT(pi) = " & INT(Pos)
PRINT "INT(-pi) = " & INT(Neg)
```

**結果**

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

**関連アイテム**

FIX

```
result = object1 IS object2
```

**説明**（オブジェクトの比較）

*object1*と*object2*が同じオブジェクトを参照している時Trueを返します。必要なコンポーネント*object1*および*object2*は、オブジェクトの参照です。

**例**

```
REM IS Example
DIM Obj2
ADDOBJECT "Finance", "Obj1"
SET Obj2 = Obj1
CompareObjects Obj1, Obj2
SET Obj2 = NOTHING
CompareObjects Obj1, Obj2

SUB CompareObjects(Obj1, Obj2)
  IF Obj1 IS Obj2 THEN
    PRINT "Same"
  ELSE
    PRINT "Different"
  END IF
END SUB
```

**結果**

```
Same
Different
```

**関連アイテム**

ADDOBJECT, SET

ISARRAY(*expression*)   ／配列？  
ISDATE(*expression*)   ／日付？  
ISEMPTY(*expression*)   ／空白？  
ISNULL(*expression*)   ／NULL？  
ISNUMERIC(*expression*)   ／数値？  
ISOBJECT(*expression*)   ／オブジェクト？

**説明** (変数タイプのチェック)

ISファンクションは*expression*で示されている変数のタイプを確認するもので、タイプが合えばTRUEを返し、合わない場合はFALSEを返します。

**例**

```
REM Is Functions Example
DIM Children(3), Chef, When
TestVariable Children
Chef = 1
TestVariable Chef
When = NOW
TestVariable When

SUB TestVariable(var)
  IF ISARRAY(var) THEN
    PRINT "The variable is an array."
  ELSEIF ISDATE(var) THEN
    PRINT "The variable is a date."
  ELSEIF ISNUMERIC(var) THEN
    PRINT "The variable is a number."
  END IF
END SUB
```

**結果**

```
The variable is an array.
The variable is a number.
The variable is a date.
```

**関連アイテム**

TYPENAME, VARTYPE

JOIN(stringarray[, delimiter])

**説明**（文字列の連結）

JOINは、文字列配列の一連の要素を`delimiter`で示される文字で連結して作成される文字列を戻します。`stringarray`は文字列の1次元配列です。オプションの`delimiter`は文字列型ですが最初の1文字だけが使われます。`delimiter`が指定されない場合、デフォルト値(" ")が使われます。連結される2つの文字列の間にスペースを入れたくない場合は`delimiter`に("")を指定して下さい。

**例**

```
REM JOIN Example
'JOIN concatenates strings
DIM Words, Letters
Words = ARRAY("Hello", "World")
Letters = ARRAY("D", "a", "l", "l", "a", "s")
PRINT JOIN(Words) & "!"
PRINT JOIN(Letters, "")
```

**結果**

```
Hello World!
Dallas
```

**関連アイテム**

SPLIT

KeyPreview = TRUE | **FALSE**

#### 説明（キー入力のフラグ）

KeyPreviewはグローバルプロパティで、Outputオブジェクトに全てのキーストロークイベントを受けるのを可能にさせます。KeyPreviewのデフォルトはFALSEであり、Outputオブジェクトがどんなキーストロークイベントも受けないようにします。KeyPreviewにTRUEを設定することで、Outputオブジェクトに全てのキーストロークイベントを受けさせる事を可能にします。これはキーボード入力を受けるために、どのオブジェクトが設定されたかは関係なく行われます。

#### 例

```
REM KeyPreview Example
'KeyPreview enables keystroke events in Output
KeyPreview = TRUE
AddObject "TextBox", "Text", 50, 50, 90, 90

SUB Output_KeyPress(key)
    PRINT "Key Pressed: ", key
END SUB
```

#### 結果



#### 関連アイテム

KeyDown, KeyPress, KeyUp, PictureBox, SetFocus, OUTPUT

KeyboardStatus=0|-1

KeyboardStatusChanged

**説明**（オンスクリーン・キーボードの状態）

KeyboardStatusはグローバル変数で、Pocket PCのオンスクリーン・キーボードの表示／非表示状態を設定／取得します。0の場合、キーボードは表示されません。-1の場合、スクリーン上にキーボードが表示されます。

スクリーン上のキーボードの有無が変更された場合、KeyboardStatusChangedイベントがプログラムに送られます。

**例**

```
REM KeyboardStatus Example
PRINT KeyboardStatus
IF KeyboardStatus=0 THEN KeyboardStatus=-1_
ELSE KeyboardStatus=-1

SUB keyboardStatusChanged
  MSGBOX "Keyboard Status Changed"
END SUB
```

**結果**

(キーボードが現れたり隠れたりする)

**関連アイテム**

KillFocus

**説明**（フォーカスを取り除く）

KillFocusはフォーカスされているオブジェクトからフォーカスを取り除きます。

**例**

```
REM KillFocus Example
AddObject "TextBox","TB",10,10,20,20
' set focus to TextBox
tb.SetFocus

' remove focus from Textbox
KillFocus
```

**結果**

（一旦カーソルがTextBoxに置かれ、その後消える）

**関連アイテム**



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

**説明**（ラベル）

Labelは出力ウィンドウ上のテキストを表示するオブジェクトです。Labelは読み専用で、テキストの内容を変えることはできません。*name*はオブジェクトの参照用にプログラムに追加される変数名です。*xpos*（X軸）、*ypos*（Y軸）、*width*（幅）及び*height*（高さ）は数値型で、画面左上コーナーからのオブジェクトの位置とサイズをピクセル値で設定します。

注意：TabStopプロパティのデフォルトはFALSEで、ユーザーインタフェースからのキーボード入力を可能にするのを防ぎます。

**サポートされるプロパティ**（PROPERTIES 参照）

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

**サポートされるメソッド**（METHODS 参照）

Hide, Move, SetFocus, Show

**サポートされるイベント**（EVENTS 参照）

Click, Timer

**例**

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

**結果****関連アイテム**

ADDOBJECT, Events, Methods, Properties, TextBox

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

**説明**（配列の下限）

LBOUNDは、配列*array*の*dimension*で示される次元の下限の添字（インデックス）をlong値で戻します。必要なパラメータ*array*はどのような配列変数でもかまいません。オプションのパラメータ*dimension*は、どの次元の下限が戻されるかを指定します。デフォルトは1です。配列の全ての次元の下限は常に0です。

**例**

```
REM LBOUND Example
' LBOUND returns lower bound of array dimension
DIM Other, Children(3), Parents(3, 1)
Other = ARRAY("Damien", "Pip", "Wendy")
PRINT "'Other' Lower Bound:", LBOUND(Other)
PRINT "'Children' Lower Bound:", _
    LBOUND(CHILDREN, 1)
PRINT "'Parents' Lower Bounds:", _
    LBOUND(Parents), LBOUND(Parents, 2)
```

**結果**

```
'Other' Lower Bound: 0
'Children' Lower Bound: 0
'Parents' Lower Bounds: 0 0
```

**関連アイテム**

ARRAY, DIM, REDIM, UBOUND

LCASE(*string*)

説明 (小文字に変換)

LCASEは文字列*string*の中の全ての大文字を小文字に変換します。必要なパラメータ*string*は文字列型です。

例

```
REM LCASE Example
'LCASE returns string with all lowercase chars
DIM Uncle
Uncle = "JIM"
PRINT Uncle & " lowercase is " & LCASE(Uncle)
```

結果

JIM lowercase is jim

関連アイテム

UCASE

LEFT(*string*, *length*)

LEFTB(*string*, *length*)

**説明**（文字列の左側抜き出し）

LEFTは、文字列の左から指定の数の文字を含んでいる文字列を戻します。必要なパラメータ*string*は有効な文字列型です。必要なパラメータ*length*は数値型で、0の場合は空の文字列("")が戻され、*string*のサイズより大きい場合は*string*の文字列全てが戻されます。

LEFTBはLEFTのバイト単位のバージョンです。  
Windows CE機のユニコード文字列において使われる時は、部分的な文字を通常の文字として戻します。

**例**

```
REM LEFT Example
'LEFT returns substring from string left end
DIM Wendy, Eric
Wendy = "Testaburger"
Eric = "Cartman"
PRINT "The LEFT 4 of " & Wendy & ": " _
    & LEFT(Wendy, 4)
PRINT "The LEFT 4 of " & Eric & ": " _
    & LEFT(Eric, 4)
```

**結果**

```
The LEFT 4 of Testaburger: Test
The LEFT 4 of Cartman: Cart
```

**関連アイテム**

LEN, MID, RIGHT

LEN(*string* | *variable*)

LENB(*string* | *variable*)

**説明**（文字列／変数の長さ）

LENはメモリ内の文字列*string*の文字数、または変数*variable*の値を出力する為に必要なバイト数を返します。オプションのパラメータ*string*は文字列型です。オプションのパラメータ*variable*はどのような変数でも構いません。LENには必ず1つのパラメータを入れて下さい。

LENBはLENのバイト単位のバージョンで常にバイト数を返します。Windows CE機器のユニコード文字列において使われる時には、LENBの戻り値は、LENにより戻される値の2倍になります。

**例**

```
REM LEN Example
'LEN returns string length or variable size
DIM Frog, Survived
Frog = "Staring"
Survived = 2
PRINT "LEN of Frog:", LEN(Frog)
PRINT "LENB of Frog:", LENB(Frog)
PRINT "LEN of Survived:", LEN(Survived)
```

**結果**

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

**関連アイテム**

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

#### 説明 (リストボックス)

ListBoxは、アウトプットウィンドウの上にテキストリストオブジェクトを表示するのに使用されます。必要なコンポーネント *name* はオブジェクトを参照する為にプログラムに追加される変数名です。必要なコンポーネント *xpos* (X軸)、*ypos* (Y軸)、*width* (幅)、および *height* (高さ) は数値型で、オブジェクトの画面左上コーナーからの位置とサイズをピクセル値で設定します。ListIndexプロパティはリスト中の選択されている項目のインデックス番号であり、最初の項目のインデックス番号は0です。何も項目が選択されていない場合、ListIndexプロパティは-1です。リストボックスにアイテムを入れる前にMultiSelectがTrueにセットされていなければなりません。

#### サポートされるプロパティ (PROPERTIES 参照)

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

#### サポートされるメソッド (METHODS 参照)

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

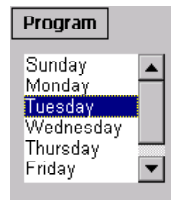
#### サポートされるイベント (EVENTS 参照)

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

**例**

```
REM ListBox Example
'ListBox is a text list object
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
```

**結果**



**関連アイテム**

ADDOBJECT, ComboBox, Events, Methods, Properties

LOG(*number*)

**説明**（自然対数）

LOGは*number*の自然対数を倍精度値(double)で戻します。  
必要なパラメータ*number*は数値型です。自然対数はベース  
e対数です。eはおおよそ2.718282です。

log<sub>x</sub>（ベースn）は、xの自然対数をnの自然対数で割ること  
により求められます。

**例**

```
REM LOG Example
'LOG calculates natural logarithms
DIM e
e = 2.718282
PRINT "LOG(1) = " & LOG(1)
PRINT "LOG(e) = " & LOG(e)
PRINT "LOG10(2) = " & LogN(10, 2)
```

```
FUNCTION LogN(Base, Number)
    LogN = LOG(Number) / LOG(Base)
END FUNCTION
```

**結果**

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

**関連アイテム**

EXP



LTRIM(*string*)

説明（文字列の左空白の削除）

LTRIMは、文字列*string*の最初の文字までの、全てのスペースを取り除いた文字列を戻します。必要なパラメータ*string*は文字列型です。

例

```
REM LTRIM Example
'LTRIM trims all leading spaces
DIM Spacey
Spacey = "-K"
PRINT "(" & Spacey & ")"
PRINT "(" & LTRIM(Spacey) & ")"
```

結果

(-K)

(K)

関連アイテム

RTRIM, TRIM

*ObjectName.Method [arglist]*

説明（オブジェクトのプロシージャ）

オブジェクトMethodsはオブジェクトのプロシージャです。必要なコンポーネント *ObjectName* は、ADDOBJECTステートメントでプログラムに追加されるオブジェクトを参照する式です。必要なコンポーネント *Method* は、オブジェクトのFUNCTIONまたはSUBプロシージャの名前であり、Table 14に主なオブジェクトメソッドをリストにしてあります。オプションのコンポーネント *arglist* は、カンマで区切られた値のリストで、引数としてプロシージャに渡されます。オブジェクトのメソッドを呼び出すことによって、オブジェクトがどう表示されるか、どう作用されるか、及びプロパティの値を変更することが出来ます。

Table 14: オブジェクトメソッド

メソッド	引数	コメント
AddItem	text[, index]	text（文章）を、任意のindex（インデックス）とアイテムのリストに加える。インデックスが供給されない場合、textは整列されていないリストの最後に入るか、整列されたリストの適切な場所に入る。ComboBox, ListBox
Clear		リストから全アイテムを消去する。ComboBox, ListBox
Hide		オブジェクトを隠し、VisibleプロパティをFALSEにセットする。
Move	x, y, w, h	オブジェクトを新しいx、yの場所に動かす。wとhが供給されている場合は、任意にサイズを改める。
RemoveItem	index	インデックス(0 = first item)によりアイテムをリストから削除する。ComboBox, ListBox

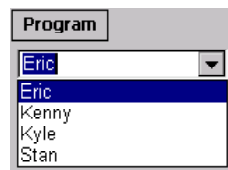
Table 14: オブジェクトメソッド

メソッド	引数	コメント
SetFocus		キーボード入力を受け取る為にオブジェクトにフォーカスを移す。
Show		オブジェクトを表示し、VisibleプロパティをTRUEにセットする。

#### 例

```
REM Object Methods Example
'Methods are procedures in objects
ADDOBJECT "ComboBox", "Combo", 5, 5, 150, 110
Combo.AddItem("Eric")
Combo.AddItem("Kenny")
Combo.AddItem("Kyle")
Combo.AddItem("Stan")
Combo.ListIndex = 0
Combo.SetFocus
```

#### 結果



#### 関連アイテム

Events, Properties

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

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

説明（文字列の中央抜き出し）

MIDは、文字列*string*の中から先頭文字ポジション*start*以降、文字数*length*の長さの文字列を戻します。*start*及び*length*は数値型です。*start*が*string*の文字数より大きい場合は("")が戻されます。*length*が指定されないか、または文字列の残りの文字数を越えている場合は、*start*から文字列の終わりへのすべての文字が戻されます。

MIDBはMIDのバイト単位のバージョンです。

Windows CE機器のユニコード文字列において使われる時には、それは正常な文字としてどのような部分的な文字でも戻します。

例

```
REM MID Example
'MID returns substring from string middle
DIM Eric, Mister
Eric = "Cartman"
PRINT "From Cartman:", MID(Eric, 2, 3)
Mister = "Hankey"
PRINT "From Hankey:", MID(Mister, 4)
```

結果

```
From Cartman: art
From Hankey: key
```

関連アイテム

LEFT, RIGHT

MINUTE(*time*)

**説明**（時刻：分）

MINUTEは、*time*によって与えられた時間が何分を表しているか、0から59までの範囲で整数を戻します。必要なパラメータ*time*は数値型、文字列型、または日付型です。

**例**

```
REM MINUTE Example
'MINUTE returns minute of hour from a time
DIM When
When = NOW
PRINT "The MINUTE of " & When & " is " _
      & MINUTE(When)
```

**結果**

The MINUTE of 8/18/1998 10:52:44 PM is 52  
(結果はシステムの設定により異なります)

**関連アイテム**

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

---

$result = x \text{ MOD } y$

**説明（割算の余り）**

MODは $x$ を $y$ で割り、余りを整数で戻します。必要なパラメータ $x$ および $y$ は数値型です。 $result$ の絶対値は常に $y$ の絶対値より小さくなります。

**例**

```
REM MOD Example
'MOD returns quotient remainder as integer
DIM Answer
Answer = 15 MOD 2
PRINT "15 MOD 2 = " & Answer
Answer = 21 MOD 3.7
PRINT "21 MOD 3.7 = " & Answer
```

**結果**

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

**関連アイテム**

MONTH(*date*)

**説明**（月の数値表現）

MONTHは与えられた日付 *date* が何月を表しているか、1から12までの範囲で整数を返します。必要なパラメータ *date* は数値型、文字列型、または日付型です。

**例**

REM MONTH Example

'MONTH returns month of year of a date

DIM When

When = NOW

PRINT "The MONTH of " & When & " is " \_  
      & MONTH(When)

**結果**

The MONTH of 8/18/1998 10:52:44 PM is 8

(結果はシステムの設定により異なります)

**関連アイテム**

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

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

**説明**（月の文字列表現）

MONTHNAMEは与えられた月の文字列名を戻します。必要なパラメータ*month*は1から12の範囲の数値型です。オプションのパラメータ*abbreviate*（短縮）はMONTHNAMEを3文字の短い表現にするかを指定するブール型です。

**例**

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

**結果**

```
August
Dec
```

**関連アイテム**

WEEKDAYNAME



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

#### 説明 (ダイアログボックス)

MSGBOXはダイアログボックスを開き、ユーザがボタンをタップするのを待ちます。戻り値は、どのボタンが指定されたかを示す整数です。必要なパラメータpromptはダイアログボックスのボディにおいて表示される文字列型です。オプションのパラメータbuttonsは数値型で、ボタン及びアイコンの選択、デフォルトボタン及びダイアログボックスの形態を指定します。デフォルト値は0で、他の値は下表から定数を加算することによって得られます。オプションのパラメータtitleは文字列型で、ダイアログボックスのタイトルバーに表示される文章です。

Table 15: ボタン定数

定数	値	説明
vbOKOnly	0	OKボタンのみ
vbOKCancel	1	OKと X ボタン
vbAbortRetryIgnore	2	中止、再試行、無視のボタン
vbYesNoCancel	3	はい、いいえ、キャンセルのボタン
vbYesNo	4	はい、いいえのボタン
vbRetryCancel	5	再試行、キャンセルのボタン
vbCritical	16	警告アイコン
vbQuestion	32	問い合わせアイコン
vbExclamation	48	注意アイコン
vbInformation	64	情報アイコン
vbDefaultButton1	0	第 1 ボタンがデフォルト
vbDefaultButton2	256	第 2 ボタンがデフォルト
vbDefaultButton3	512	第 3 ボタンがデフォルト
vbDefaultButton4	768	第 4 ボタンがデフォルト

Table 15: ボタン定数

定数	値	説明
vbApplicationModal	0	アプリケーションのインターフェイスは、ユーザーのダイアログボックスに対する返答がない限り動作しません。
vbSystemModal	4096	アプリケーションとシステムインターフェイスは、ユーザーのダイアログボックスに対する返答がない限り動作しません。

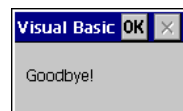
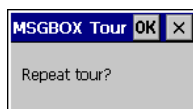
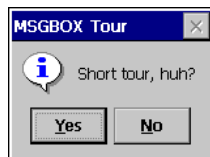
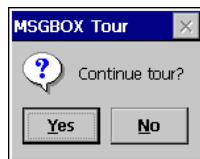
Table 16: MSGBOX 戻り値

定数	値	説明
vbOK	1	OK
vbCancel	2	キャンセル
vbAbort	3	中止
vbRetry	4	再試行
vbIgnore	5	無視
vbYes	6	Yes
vbNo	7	No

例

```
REM MSGBOX Example
'MSGBOX displays a modal dialog box
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
```

## 結果



## 関連アイテム INPUTBOX

*result* = NOT *expression*

**説明**（論理否定演算）

式*expression*の論理否定を評価します。*expression*は有効な式に限ります。*result*は*expression*がFALSEの時にTRUEになり、TRUEの時にFALSEになり、それ以外はNULLになります。

NOTはビット単位の比較も行います。*result*の各ビットは、*expression*の相対するビットが0の場合1になり、1の場合0になります。

**例**

```
REM NOT Example
'NOT does logical negation & bitwise inversion
DIM Test1, Test2, Test3, x, y
x = 3
y = 8
Test1 = NOT(x > 0)
Test2 = NOT(y > 10)
Test3 = NOT x
PRINT "Logical:"
PRINT "  NOT(x > 0) = " & Test1
PRINT "  NOT(y > 10) = " & Test2
PRINT "Bitwise:"
PRINT "  NOT x = " & Test3
```

**結果**

```
Logical:
  NOT(x > 0) = False
  NOT(y > 10) = True
Bitwise:
  NOT x = -4
```

**関連アイテム**

AND, EQV, IMP, OR, XOR

**NOW****説明**（現在の日時）

NOWはコンピュータのシステムクロックに従って現在の日付と時間を戻します。

**例**

```
REM NOW Example
'NOW returns system date and time
PRINT "NOW is " & NOW
```

**結果**

NOW is 8/18/1998 10:52:44 PM  
(結果はシステムの設定により異なります)

**関連アイテム**

DAY, DATE, MINUTE, MONTH, SECOND, TIME, YEAR

NSBVERSION

**説明** (NS BASIC/CEのバージョン)  
NSBVersionはNS Basic/CEのバージョンを文字列で返します。このグローバルプロパティは読み込み専用です。

**例**

```
REM NSBVersion Example
PRINT "The current version of NS Basic is " &
NSBVersion
```

**結果**

The current version of NS Basic is v. 4.0.1

**関連アイテム**

OCT(*number*)

説明 (8進法表現)

OCTは、数の8進法（ベース8）の値を文字列にして戻します。必要なパラメータ*number*は数値型です。*number*が整数でない場合は、変換される前に最も近い整数に丸められます。

例

REM OCT Example

'OCT returns a number as an octal string

PRINT "68 in octal:", OCT(68)

PRINT "1 in octal:", OCT(1)

PRINT "2605.45 in octal:", OCT(2605.45)

結果

68 in octal: 104

1 in octal: 1

2605.45 in octal: 5055

関連アイテム

HEX

ON ERROR RESUME NEXT

ON ERROR GOTO 0

**説明**（実行時エラーの処理）

ON ERRORはエラー処理をプロシージャ内で行うことにより、致命的なエラーを検出し、プログラムの継続を可能します。RESUME NEXTを使うと、エラーが起こったステートメントのすぐ後のステートメントから処理が継続されます。GOTO 0を使う場合、プロシージャ内でのエラー処理を無効にします。ON ERRORステートメントが使われない場合、全ての実行時エラーは致命的であり、エラーメッセージを表示して実行を終了させます。

**例**

```
REM ON ERROR Example
'ON ERROR does error-handling in procedures
ADDOBJECT "File"
GetToDoList

SUB GetToDoList
  ON ERROR RESUME NEXT
  File.OPEN "ToDo", 1
  IF ERR.NUMBER THEN
    PRINT "Delayed ERROR handling"
    PRINT "ERROR Source: " & ERR.Source
  END IF
  ON ERROR GOTO 0
END SUB
```

**結果**

```
Delayed ERROR Handling
ERROR Source: File
```

**関連アイテム**

Err Object



## OPTION EXPLICIT

## 説明（変数定義の強制）

OPTION EXPLICITは、すべての変数の定義を強制するために、スクリプトレベルにおいて使われます。OPTION EXPLICITは全てのFUNCTIONまたはSUBプロシージャよりも前に書かれなければなりません。これにより全ての無定義の変数はエラーになります。変数の定義にはDIMまたはREDIMを使用して下さい。

OPTION EXPLICITはプログラム性能を改善でき、よいプログラミング習慣を実施しますので、常に使用されることをお勧めします。

STEPまたはTRACEを使用時、OPTION EXPLICITはプログラムの最初のラインに書かれなければなりません。

## 例

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

## 関連アイテム

DIM, REDIM

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

**説明** (ラジオボタン)

OptionButtonは、アウトプットウィンドウ上に円形のボタンオブジェクト (ラジオボタン) を表示するのに使用されます。*name*はオブジェクトを参照する為にプログラムに追加される変数名です。*xpos* (X軸)、*ypos* (Y軸)、*width* (幅)、および*height* (高さ) は数値型で、オブジェクトの画面左上コーナーからの位置とサイズをピクセル値で設定します。ラジオボタンが空の時、ValueプロパティはFALSEです。ラジオボタンが指定されている時、ValueプロパティはTRUEです。

**サポートされるプロパティ** (PROPERTIES 参照)

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

**サポートされるメソッド** (METHODS 参照)

Hide, Move, SetFocus, Show

**サポートされるイベント** (EVENTS 参照)

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

#### 例

```
REM OptionButton Example
'OptionButton is a toggle button with a circle
'often referred to as a Radio Button
ADDOBJECT "OPTIONBUTTON", "O1", 10, 10, 100, 20
O1.CAPTION="FEMALE"
O1.VALUE=TRUE

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
```

#### 結果

☒ Female  
☐ Male  
☐ Undisclosed

#### 関連アイテム

ADDOBJECT, CheckBox, Events, Methods, Properties

*result* = *x* OR *y*

**説明**（論理和演算）

式*x*と*y*のどちらか一方、または両方がTRUEの時、*result*はTRUEになります。それ以外はFALSEです。

ORは2つ数値のビット単位の比較も行います。*x*と*y*の相対するビットのどちらか一方又は両方が1の場合、*result*の*x*と*y*に相対するビットは1になり、それ以外は0をセットします。

**例**

```
REM OR Example
'OR performs logical and bitwise disjunction
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 "Logical:"
PRINT "  x > 0 OR y < 10 = " & Test1
PRINT "  x > 0 OR y > 10 = " & Test2
PRINT "Bitwise:"
PRINT "  x OR y = " & Test3
```

**結果**

```
Logical:
  x > 0 OR y < 10 = True
  x > 0 OR y > 10 = True
Bitwise:
  x OR y = 5
```

**関連アイテム**

AND, EQV, IMP, NOT, XOR

OUTPUT.*property*=*value*

#### 説明 (アウトプットオブジェクト)

NS Basic/CEプログラムを実行すると、Outputオブジェクトが、一番下に位置するオブジェクトとして自動的に作成されます。これはPictureBoxオブジェクトですので、全て同じプロパティ、メソッド、およびイベントを使用することができます。(Tech Note 7を参照)さらに、PRINTステートメントからの出力はOutputオブジェクトに渡ります。

右上コーナーのクローズボックスを使用してOutputオブジェクトが閉じられるとき、Form\_closeイベントがプログラムに送られます。これを使用することによって、プログラムのクリーンアップ処理を行うことができます。

#### サポートされるプロパティ (PROPERTIES 参照)

PictureBoxと同じ

#### サポートされるメソッド (METHODS 参照)

PictureBoxと同じ

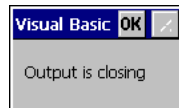
#### サポートされるイベント (EVENTS 参照)

PictureBoxと同じイベントと Output\_Close, Output\_Size

#### 例

```
REM Show close box action
SUB form_close
  MSGBOX "The Output box is closing"
END SUB
```

#### 結果



#### 関連アイテム

ADDOBJECT, Events, Methods, Properties, PictureBox, KeyPreview

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

#### 説明 (ピクチャーボックス)

PICTUREBOXはアウトプットウィンドウ上に、テキスト及びグラフィックを表示するために使用されます。*name*はプログラムに追加されるオブジェクトを参照する変数名です。*xpos* (X軸)、*ypos* (Y軸)、*width* (幅)、および*height* (高さ)は画面の左上コーナーからオブジェクトへの位置とサイズを、ピクセル値で設定します。アウトプットウィンドウそのものはPICTUREBOXであり、プログラム変数OutputはPICTUREBOXプロパティ、メソッド及びイベントと共に使用できます。PictureBoxのプロパティ、イベント、メソッドの簡略したリストは以下です。詳細なリストはTech Note 7を参照してください。

#### サポートされるプロパティ (PROPERTIES 参照)

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

Table 17: PictureBox プロパティ

名前	説明
BorderStyle	0: 無、1: 単線ボーダー
DrawWidth	ドローイングメソッド用の線幅
FillColor	形、円、ボックス用の色
FillStyle	0: 無地、1: 透明
FontTransparent	真 (True) / 偽 (False)
Picture	イメージファイル名の文字列
ScaleHeight	カスタムスケール使用の高さの単位
ScaleLeft	オブジェクトの左端 (拡大/縮小用)

Table 17: PictureBox プロパティ

名前	説明
ScaleMode	0 ユーザー指定 1 Twips (トウイプ)(1440 dpi) 2 Points (ポイント)(72 dpi) 3 Pixels(ピクセル) 4 Characters (文字)(水平 = 120 twips, 垂直 = 240 twips) 5 inches(インチ) 6 millimeters(ミリメートル) 7 centimeters(センチメートル)
ScaleTop	現オブジェクトのトップ
ScaleWidth	指定スケールを使った幅の単位
Tag	何にでも使用

サポートされるメソッド (METHODS 参照)  
Move

Table 18: PictureBox メソッド

名前	引数	説明
Cls		Textとグラフィックを 消去、プリントは含ま ない
DrawCircle	x, y, radius[, color [, aspectRatio]]	正円か、楕円を描く
DrawLine	x1, y1, x2, y2[, color [, box [, fill]]]	box は TRUE/FALSE
DrawPicture	filename, x1, y1[, width, height, x2, y2, width2, height2], rasterop]	オブジェクト上にイメ ージファイルの部分ま たは全体を描く(再スケ ールも可能)
DrawPoint	x, y[, color]	一つの点を描く
DrawText	Text[.x,y]	オブジェクト上の次の ラインか、x、yで指定 された位置 に描く。
Refresh		オブジェクトの再描画
ScaleX	width, from, to	幅のスケールを変換す る

Table 18: PictureBox メソッド

名前	引数	説明
ScaleY	height, from, to	高さのスケールを変換する
SetScale	x1, y1, x2, y2	スケールの変更に使用
TextHeight	text	Textの高さを戻す、埋め込みCR'sはOK
TextWidth	text	最も幅のあるラインの幅を戻す

サポートされるイベント (EVENTS 参照)  
Click

Table 19: PictureBox イベント

名前	引数	説明
KeyDown†	keyCode+, shift*	Keyが押される
KeyPress†	keyCode+	Key-DownとKey-Upのコンビネーション
KeyUp†	keyCode+, shift*	Keyが放される
MouseDown#	button, shift*, x, y	オブジェクトがスタイラスか入力装置で接触される
MouseMove#	button, shift*, x, y	スタイラスか入力装置がオブジェクトと接触の間に動かされる
MouseUp#	button, shift*, x, y	スタイラスがオブジェクトから離れるか、入力装置が対象との接触を中断する

+ イベントを発生させるキーのキーボードコード

\* 1 = Shiftキー, 2 = CTRLキー, 4 = ALTキー

† Output PictureBox オブジェクトでのみ。このイベントを受ける為にはグローバルプロパティのKeyPreviewをTRUEにします

# button はオブジェクトに接触するデバイスのボタンです(0 = スタイラス)。x, y, はオブジェクトに対しての接触の位置を示します。

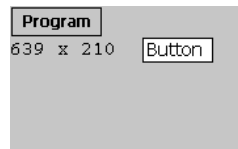


**例**

```
REM PictureBox Example
ADDOBJECT "PictureBox", "PBox", 65, 0, 55, 20
PBox.BorderStyle = 1
PBox.BackColor = vbWHITE
PBox.DrawText " Button"

SUB PBox_Click
    PRINT Output.Width & " X " & Output.Height
END SUB
```

**結果**



(スクリーンサイズは機種により異なります)

**関連アイテム**

ADDOBJECT

PLAYSOUND(*file, hmod, flags*)**説明（音の再生）**

PlaySoundファンクションは、指定されたファイル、リソース、またはシステムイベントによって指定されるサウンドを再生します。*file*は再生用のファイル名です：この引数に0を使用することにより、現在のサウンドを止めます。*hmod*は通常0です（リソースファイルを使用しているならば、リソースへのハンドルです）。*flags*は、以下のFlagsを足すことによって形成されます。

**Table 20:**

Flags	値	説明
APP	?	指定されたアプリケーションを使った再生
ALIAS	&h00010000	<i>file</i> は、レジストリがWIN.INIファイルの中のシステムイベントの通称である。FILENAMEやRESOURCEと一緒に使用しないこと。
ASYNCH	&h00000001	再生が始まった時点で、このファンクションは戻る。
FILENAME	&h00020000	<i>file</i> はファイル名
LOOP	&h00000008	<i>file</i> を0にセットしてPlaySoundが呼ばれるまで再生
MEMORY	&h00000004	<i>file</i> がメモリの中の音のイメージを指す
NODEFAULT	&h00000002	音が見つげられない場合、無音のまま戻る
NOSTOP	&h00000010	もし再生中の場合、他の音に譲る
NOWAIT	&h00002000	ドライバ使用中の場合は戻る
RESOURCE	&h00040004	<i>file</i> はリソース名; Hmodはインスタンスを示す
SYNC	&h00000000	再生が完全に終わった時、このファンクションは戻る。

**例**

```
PlaySound "SystemStart",&h10000,0  
Sleep 500  
PlaySound 0,0,0 'stop it after .5 secs
```

**結果**

(0.5 秒間音が再生されます)

**関連アイテム**

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

**説明** (プリント)

PRINTはテキストをアウトプットウィンドウに書きます。  
PRINTは、最高20個のカンマで区切られた式を書くことができ、出力時にそれぞれはタブで区切られ、最後にキャリッジ・リターンが付加されます。*expression*がない場合は、PRINTは単なる改行として使われます。

フォームを使っている場合、フォームがアウトプットウィンドウを隠してしまうため、このステートメントによる出力は見えません。

**例**

```
REM PRINT Example
'PRINT writes text to the output window
PRINT "Hello World!"
PRINT
PRINT "The time is", NOW
```

**結果**

Hello World!

The time is 8/18/1998 10:52:44 PM

**関連アイテム**

`ObjectName.Property[ = value]`

#### 説明（オブジェクトの属性）

オブジェクトのプロパティはオブジェクトが持つ変数です。必要なコンポーネント *ObjectName* はオブジェクト型で、`ADDOBJECT` ステートメントでプログラムに追加されるオブジェクトを参照します。必要なコンポーネント *Property* はオブジェクトの変数名で、下表に共通のオブジェクトプロパティをリストにしてあります。オプションのコンポーネント *value* は、オブジェクトの変数に割当てられます。オブジェクトのプロパティを変えることで、オブジェクトがどのように表示されて、それがどのように作用されるかを変更することができます。

いくつかのプロパティはオブジェクトの現れ方に影響するので、オブジェクトの作成直後に設定するようにして下さい。

いくつかのプロパティは書き込み禁止ですので、デザイン時のみセットできます。実行時はいくつかのプロパティは読むことができません。

Table 21: プロパティ

プロパティ名	有効値	コメント
Alignment+	整数型 0 = 左寄せ 1 = 右寄せ 2 = 中央寄せ	Text/Captionテキスト整列。Label, OptionButton
BackColor	Color (整数型)	オブジェクトの背景色。色はRGB値またはシステムパレットにある色。
Bottom	整数型	廃止。代わりにHeightを使用
BorderStyle	0 = none 1 = line	ボーダー（縁取り）Date, Label, TextBox, Time
Caption	文字列型	オブジェクトに表示されるテキスト。CheckBox, ComboBox, CommandButton, Label, TextBox
Date	日付型	日付を設定。例： Date.date=cdate("8/5/02")
Default	true/FALSE	enterキーが使われる時のボタンのデフォルト値。

Table 21: プロパティ

プロパティ名	有効値	コメント
Encrypted	TRUE/FALSE	プロジェクトプロパティのみ。セットされると、保存されるファイルは暗号化される。Desktop IDEのみ。
Enabled	True	オブジェクトはユーザーイベントに応答するの か？
ExpandedHeight	Integer	拡大した時の Comboboxのサイズ
FontBold	TRUE/FALSE	強調文字(ボールド)
FontItalic	TRUE/FALSE	斜体文字
FontName	文字列型	フォント名
FontSize	整数型	フォントサイズ
FontStrikethru	TRUE/FALSE	抹消ライン
FontUnderline	TRUE/FALSE	アンダーライン
FontWeight	整数型 400 = 標準 700 = ボールド	550以上の値は700に その他の値は400に 変換される
ForeColor	Color (整数型)	テキストと図形のオブ ジェクトの前景色 (p.16 参照)
LongFormat	true/FALSE	日付表示の形式 Date
Group	TRUE/false	新しいオプショングル ープを始める。 OptionButton
Height	整数型	オブジェクトの上端か ら下端までのピクセル
HideSelection	TRUE/false	コントロールがフォー カスを失った時に、選 択されているテキスト を隠す。TextBox
Hwnd†	Long	オブジェクトの内部参 照符
IntegralHeight+	TRUE/FALSE	個別のラインの積分倍 数である高さへのオブ ジェクトの強制 ComboBox, ListBox
Left	整数型	アウトプットウインド ウの左端からオブジェ クトの左端までのピク セル
List(index †)	文字列型	値の配列。インデック スの引数が必要 †

Table 21: プロパティ

プロパティ名	有効値	コメント
ListCount†	整数型	リストの中のアイテム数, ComboBox, ListBox
ListIndex	整数型	指定されているアイテム, ComboBox, ListBox
Locked	true/FALSE	テキストの変更を規制する。テキストの選択は引き続き可能。 TextBox
LowercaseOnly	true/FALSE	テキストの入力を小文字のみに制約する。 ComboBox, TextBox
MaxLength	整数型 (0-30,000)	許容された文字数, TextBox (multiline)
MultiLine+	TRUE/FALSE	埋め込みキャリッジリターンと、テキストラッピング（自動改行）を可能にする, TextBox
MultiSelect	true/FALSE	複数アイテムの選択を許容。ListBox
Name†		オブジェクト名
NewIndex	整数型	AddItemの為のソートされていないオブジェクトのリストへのインデックス挿入, ComboBox, ListBox
NumbersOnly	true/FALSE	数値文字のみの入力に制約。TextBox
ParentHWnd	ウィンドウハンドル	オブジェクトの親
Password	true/FALSE	入力内容を隠すために、入力文字をアスタリスク(*)と取り替える。 TextBox
Redraw	TRUE/false	リストに複数のアイテムを入れている間の画面の更新。 CombBox, ListBox
Right	整数型	廃止。代わりにWidthを使用

Table 21: プロパティ

プロパティ名	有効値	コメント
Scrollbars+	整数型 0 = 無し 1 = 水平 2 = 垂直 3 = 両方	スクロールバー ListBox (垂直のみ), TextBox (multiline)
SelCount†	整数型	選択されているアイテム数。ListBox
Selected(index)	TRUE/FALSE	実行時のみ。リスト中のindexで指定されるアイテムの選択状況。
SelLength	整数型	選択されている文字数。Textbox
SelStart	整数型	選択されている最初の文字。Textbox
SelText	文字列型	選択されている文字列。TextBox
Sorted+	TRUE/FALSE	AddItemを使って加えられたアイテムはアルファベット順に挿入される。ListBox, ComboBox
Style+	整数型 0 = 編集可能 2 = 書き込み禁止	入力ラインは編集可能／書き込み禁止。ComboBox
Tabstop	TRUE/FALSE*	オブジェクトをタブによるキーボード入力を受ける状態にする
Tag	文字列型	独自の目的に利用可能
Text	文字列型	テキストをオブジェクトに表示
Timer	Long	入力数値に100分の1秒を掛けた時間が経過後、object_timerイベントが起こる。キャンセルは0をセットする。
Right	整数型	廃止。代わりにWidthを使用



Table 21: プロパティ

プロパティ名	有効値	コメント
Scrollbars+	整数型 0 = 無し 1 = 水平 2 = 垂直 3 = 両方	スクロールバー ListBox (垂直のみ), TextBox (multiline)
SelCount†	整数型	選択されているアイテム数。ListBox
Selected(index)	TRUE/FALSE	実行時のみ。リスト中のindexで指定されるアイテムの選択状況。
Top	整数型	アウトプットウィンドウの上端からオブジェクトの上端までのピクセル
TopIndex	整数型	リストの一番上に見えるアイテム。 ComboBox, ListBox
UppercaseOnly	true/FALSE	テキストの入力を大文字のみに制約する。 ComboBox, TextBox
UseMnemonic	TRUE/false	これがTrueの時、'&'文字を使うことにより、次に来る文字にアンダースコアが引ける。Label
Value	TRUE/FALSE または 0/1	オブジェクトがチェックされたか、選択されたか、CheckBox, OptionButton
Visible†	TRUE/FALSE	Hide/Showメソッドでセット
Width	整数型	オブジェクトの左端から右端までのピクセル
WindowLong(index)	Index: 0=ExWindowStyle 1=WindowStyle	マイクロソフトのWindows CEのドキュメントを参照

- † 書き込み禁止プロパティ。デザイン時に設定  
+ 基本的なオブジェクトの表示に影響  
\* Labelオブジェクト用にデフォルトはFALSE

#### 例

```
REM Object Property Example
'A property is a variable in an object
ADDOBJECT "TextBox", "Input" 50, 50, 100, 100
ADDOBJECT "CommandButton", "Button", 150, 170, 20, 80

PRINT "Text: " & Button.Text
Button.Text = "Push Me!"
PRINT "Visible: " & CBOOL(Input.Visible)
Input.Hide
PRINT "Visible: " & CBOOL(Input.Visible)
```

#### 結果

```
Text:
Visible: True
Visible: False
```

#### 関連アイテム

Events, Methods

RANDOMIZE [*number*]

説明 (乱数ジェネレータの初期化)

RANDOMIZEは乱数ジェネレータを初期設定します。  
*number*には、RND関数の乱数ジェネレータに与える新しい種値 (seed) を指定します。*number*を省略した場合、システムタイマーから取得した値が種値として使われます。一連の乱数を繰り返すには、負数の引数と共にRNDを呼び、その後*number*に数値を入れてRANDOMIZEを呼びます。  
(同じ数値を引数に使ってRANDOMIZEを呼んでも同じ結果にはなりません。)

例

```
REM RANDOMIZE Example
'RANDOMIZE initialize random-number generator
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 "Four random numbers:", RET
END SUB
```

結果

```
Four random numbers: 8 912 43 537
Four random numbers: 33 6 430 51
Four random numbers: 36 6 192 54
```

関連アイテム

RND

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

**説明（配列の再配分）**

REDIMは、固定配列のためのメモリー領域を再配分するために使用します。必要なコンポーネント *nameA* は変数名です。オプションの引数 *subscript1* は、カンマで区切ることで、配列の大きさを最高60次元まで表せます。最後の次元のサイズだけが交換される時には、オプションのキーワード *PRESERVE* により既存の配列のデータを保存します。配列のインデックスは常に0から始まります。

**例**

```
REM REDIM Example  
'REDIM reallocates array storage  
'An empty variable named "Foo"  
DIM Foo  
'A one-dimensional array with 10 elements  
REDIM Foo(9)  
'A two-dimensional array with 600 elements  
'20 x 30  
REDIM Foo(19, 29)
```

**関連アイテム**

ARRAY, DIM

REM *remarks*

'remarks

#### 説明（コメント）

REMに続く文章は全てコメント扱いされ、実行されません。単一のラインの途中からコメントにする場合は、REMは前にコロン（:）を入れなければなりません。

アポストロフィ（'）は、REMの代わりに使用できますが、REMと違いアポストロフィと文章の間にスペースは必要ありません。また途中からコメントにする場合もコロンは必要ありません。

#### 例

```
REM REM Example
REM This example does absolutely nothing
'It doesn't even have a single line of
'executable code
```

#### 結果

#### 関連アイテム

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

**説明**（文字の置換え）

REPLACEは、文字列*target*の中の別の文字列*find*と与えられる文字列*source*とを交換し、その結果の文字列を返します。オプションの*start*及び*count*は数値型です。*start*は検索開始位置を指定し、指定されていない場合のデフォルト値は1が設定され、*target*の最初の文字から検索します。*count*はいくつ入れ替えを行うかを指定し、指定されない場合のデフォルト値は-1が設定され、全ての有効な入れ替えが行われます。*compare*は検索タイプを指定するもので、Table 10を参照して下さい。

**例**

```
REM REPLACE Example
'REPLACE performs string substitutions
DIM Message, Ride
Message = "Good morning, class."
PRINT Message
Message = REPLACE(Message, "morning", _
    "afternoon")
PRINT Message
Ride = "big Al's big boat ride"
Ride = REPLACE(Ride, "big", "Big", 1, 1)
PRINT Ride
```

**結果**

```
Good morning, class.
Good afternoon, class.
Big Al's big boat ride
```

**関連アイテム**

RGB(*red, green, blue*)

**説明** (RGBカラー)

RGBはRGBカラーを表す整数を戻します。*red* (赤)、*green* (緑)、*blue* (青) は0から255の値で、それぞれの色度合いを表します。

**例**

```
REM RGB Example
'RGB returns RGB Color
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)
```

**結果**

(赤い四角、グレーの四角、白い四角)

**関連アイテム**

RIGHT(*string*, *length*)

RIGHTB(*string*, *length*)

**説明** (文字列の右側抜出し)

RIGHTは、*string*の右から*length*で指定された文字数の文字列を戻します。必要なパラメータ*string*は文字列型です。パラメータ*length*は数値型で、0の場合は空の文字列("")が戻されます。

RIGHTBはRIGHTのバイト単位のバージョンです。Windows CEデバイスのユニコード文字列に使われると、正常な文字として部分的な文字を戻します。

**例**

```
REM RIGHT Example
'RIGHT returns substring from string right end
DIM Wendy, Eric
Wendy = "Testaburger"
Eric = "Cartman"
PRINT "The RIGHT 6 of " & Wendy & ":", _
    RIGHT(Wendy, 6)
PRINT "The RIGHT 4 of " & Eric & ":", _
    RIGHT(Eric, 4)
```

**結果**

```
The RIGHT 6 of Testaburger:   burger
The RIGHT 4 of Cartman:      tman
```

**関連アイテム**

LEFT, MID



RND[(*number*)]

説明 (乱数の作成関数)

RNDは0から1の間の乱数を単精度値 (single) で戻します。オプションのパラメータ*number*は数値型で、乱数作成の為に使われる種値 (seed) です。

Table 22: RND seed values

種値 (Seed)	RND による結果
< 0	毎回同じ数値
> 0 又は無指定	数列内の次の乱数
0	最後に作られた数

例

```
REM RND Example
'RND generates random numbers
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 "Four random numbers:", RET
END SUB
```

結果

```
Four random numbers: 6 71 1 19
Four random numbers: 33 28 4 51
Four random numbers: 33 28 4 51
```

関連アイテム

RANDOMIZE

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

**説明** （概数）

ROUNDは、*number*を*fractionaldigits*の数が定める小数点何桁に丸めます。ROUNDが返す値の最後の数字は必ず偶数になります。*number*は数値型です。*fractionaldigit*が指定されない場合はデフォルト値0が設定され、ROUNDは整数を戻します。

**例**

```
REM ROUND Example
'ROUND rounds numbers to a given decimal place
DIM Pi, Pure, Ate
Pi = ROUND(3.14159265, 4)
PRINT Pi
Pure = ROUND(99.4444, 2)
PRINT Pure
Ate = ROUND(SQR(69))
PRINT Ate
```

**結果**

```
3.1416
99.44
8
```

**関連アイテム**

INT, FIX

RTRIM(*string*)

**説明** (文字列の右空白の削除)

RTRIMは、文字列*string*の最後にあるスペースを削除した文字列を戻します。必要なパラメータ*string*は文字列型です。

**例**

```
REM RTRIM Example
'RTRIM trims all trailing spaces
DIM Spacey
Spacey = "K-"
PRINT "(" & Spacey & ")"
PRINT "(" & RTRIM(Spacey) & ")"
```

**結果**

(K-)

(K)

**関連アイテム**

LTRIM, TRIM

RUNAPPATEVENT *app*, *event*

**説明** (イベントによってアプリ起動)  
オペレーティングシステムの特定期間イベントを引き金に、プログラムを起動します。 *app*は実行されるコマンドラインです。 *event*はプログラムを起動させるシステムイベントを指定します。

Event	Value
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

**例**

```
REM RUNAPPATEVENT Example
'RUNAPPATEVENT launches a program triggered
'by a system event
RUNAPPATEVENT "\Windows\player.exe", 3
```

**関連アイテム**

Runappattime

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

**説明** （特定時刻にアプリ起動）  
未来の特定時刻にプログラムを起動します。*app*は実行されるコマンドラインです。*yy, mo, dd, hh, mm, ss*はプログラムを起動させる日時を指定します。

**例**

```
REM RUNAPPATTIME Example
'RUNAPPATTIME launches a program at a specific time
RUNAPPATTIME "\Windows\player.exe", 2004, _
    5, 25, 14, 30
```

**関連アイテム**

RunappatEVENTT

```
ADDOBJECT "HScrollbar", name, xpos, ypos, width, height  
ADDOBJECT "VScrollBar", name, xpos, ypos, width, height
```

**説明** (スクロールバー)

スクロールバーを表示します。右、左、上、下のどれかの矢印がタップされるか、スライドバーが動かされるとChangeイベントが起こります。SmallChangeは最低でも1をセットしなければなりません。

**サポートされるプロパティ** (PROPERTIES 参照)

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

**サポートされるメソッド** (METHODS 参照)

Hide, Move, SetFocus, Show

**サポートされるイベント** (EVENTS 参照)

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

**例**

```
REM VScrollBar Example  
ADDOBJECT "VScrollBar", "Button", 120, 10, 20, 100
```

**結果**

例

```
REM HScrollbar Example
ADDOBJECT "HScrollbar", "Button", 10, 120, 100, 20

SUB Button_Change
    PRINT "Button clicked"
    KillFocus
END SUB
```

結果



関連アイテム

ADDOBJECT, Events, Methods, Properties

SECOND(*time*)

**説明** (時刻: 秒)

SECONDは、*time*によって与えられた時間が何秒を表しているか、0から59までの範囲で整数を返します。必要なパラメータ*time*は数値型、文字列型、または時間型です。

**例**

REM SECOND Example

'SECOND returns second of minute of given time

PRINT "The SECOND of " & NOW & " is " \_  
      & SECOND(NOW)

**結果**

The SECOND of 8/18/1998 10:52:44 PM is 44  
(結果はシステムの設定により異なります)

**関連アイテム**

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



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

**説明** (SELECT CASE 条件選択)

SELECT CASEは、各グループのステートメントのうちの1つを条件付きで実行します。全てのCASE条項には *expressionlist*が必要になります。各*statements*は、ひと固まりのステートメントであり、その*expressionlist*が *testexpression*と一致した時に、次のCASE条項の手前又はEND SELECTの手前まで実行され、END SELECTの次のステートメントに続きます。*testexpression*がどの*expressionlist*とも一致しない場合、*elstatements*があればそれを実行し、なければ実行はEND SELECTの後に続きます。

#### 例

```
REM SELECT CASE Example
'SELECT CASE performs conditional execution
CheckHat("Blue")
CheckHat("Orange")
```

```
SUB CheckHat(Hat)
  SELECT CASE Hat
    CASE "Blue"
      PRINT "Kyle's hat"
    CASE "Green"
      PRINT "Stan's hat"
    CASE "Cyan"
      PRINT "Eric's hat"
    CASE "Orange", "Hood"
      PRINT "Kenny's hat"
    CASE "White"
      PRINT "Chef's hat"
    CASE "Striped"
      PRINT "Mr. Hat"
    CASE "Christmas", "Santa"
      PRINT "Mr. Hankey's hat"
    CASE ELSE
      PRINT "Unknown Hat"
  END SELECT
END SUB
```

#### 結果

```
Kyle's hat
Kenny's hat
```

#### 関連アイテム

```
IF...THEN...ELSE
```

SENDKEY *keyFlags*, *keyChar*

**説明** (キー入力)

プログラムによってキーボード入力を行います。*keyFlags* はキー入力に同時に使うモディファイア・キー (shift, ctrl, alt) のコンビネーションです。*keyChar*は入力するキーの数値です。

**例**

```
REM SENDKEY Example
'SENDKEY performs keyboard input
TextBox1.SetFocus
SENDKEY 0, 97
SENDKEY 0, 98
SENDKEY 0, 99
'テキストボックスにabcを入力
```

SET *objectvariable* = {*objectexpression* | NOTHING}

**説明** (変数の設定)

SETは、オブジェクト参照を変数に代入するために使用されます。必要なコンポーネント *objectvariable* は変数名です。必要なコンポーネント *objectexpression* は、オブジェクト参照を含んでいる変数またはオブジェクトを戻すファンクションコールです。オプションのキーワード NOTHING は *objectvariable* が指すオブジェクトへの参照を解除します。変数による参照がなくなった時、オブジェクトのシステム及びメモリリソースは解放されます。

**例**

```
REM SET Example
'SET assigns objects to variables
DIM ButtonRef
ADDOBJECT "Picturebox", "Button", 0, 0, 10, 10
SET ButtonRef = Button
```

**結果**

**関連アイテム**

IS

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

#### 説明（メニュー）

SETMENUは、プログラム実行中に、アウトプットウィンドウに独自のメニューを加えます。*menustring*は、メニュー項目に表示されるテキストです。オプションのコンポーネント *menukey*は、メニュー項目とのつながりを持つ変数名です。*menukey*は2つの縦棒文字(|)によって*menustring*と切り離されていて、*menustring*がスペースか特殊文字を含むとき、長い*menustring*を簡略化する為か、または同じ*menustring*で複数のメニュー項目を持つために使用されます。*menukey*の指定がない場合、*menustring*の値が使用されます。*menulist*は文字列型の配列で、メニュー項目の下にあるサブメニューの*menukeys*を表します。アウトプットウィンドウのルートメニューの*menukey*は "Titlebar"です。

メニュー項目が選択されると、イベントがプログラムに送られます。イベントに応じるには、以下の構文があるPUBLIC SUBプロシージャを加えます。

```
SUB menukey_click()  
    '応答する何かをする  
END SUB
```

ハイフン文字(-)と共に始まる*menustring*は、分離ライン（選択不可能）としてメニューに追加されます。分離ラインはサブメニューを持つことは出来ません。

SETMENUは、実行時にメニューを変える為に使用できます。SETMENUが呼ばれると、対応しているメニューはアウトプットウィンドウでアップデートされます。メニュー項目が変更されるならば、そのサブメニュー項目のすべてがクリアされます。

キーボードのメニュー操作の為に、*menustring*における文字はアンダーラインを引くことができます。*menustring*のアンダーラインを引きたい文字の前に、アンパーサンド文字(&)を挿入して下さい。

メニューに第2コラム（キーボードショートカット等用）を表示するには、タブ文字(vbTAB)と2番目のコラムへの文字列を*menustring*に追加して下さい。

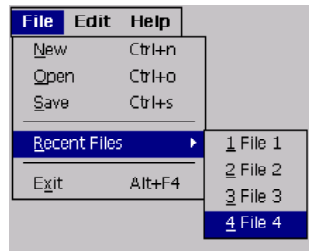
#### 例

```
REM SETMENU Example
'SETMENU adds custom menus to output window
DIM Menu
Menu = ARRAY("File", "Edit", "Help")
SETMENU "Titlebar", Menu
Menu = ARRAY("&New" & vbTAB & "Ctrl+n|New", _
"&Open" & vbTAB & "Ctrl+o|Open", _
"&Save" & vbTAB & "Ctrl+s|Save", _
"-", _
"&Recent Files|Recent", _
"-", _
"E&xit" & vbTAB & "Alt+F4|Exit")
SETMENU "File", Menu
Menu = ARRAY("&1 File 1|RF1", _
"&2 File 2|RF2", "&3 File 3|RF3", _
"&4 File 4|RF4")
SETMENU "Recent", Menu
SETMENU "Edit", ARRAY("Cut", "Copy", "Paste")
Menu = ARRAY("About SETMENU Example...|ASE")
SETMENU "Help", Menu

SUB New_Click()
'Open a new file
END SUB

SUB Open_Click()
'Open an existing file
END SUB
```

#### 結果



#### 関連アイテム

SETPARENT *child, parent*

**説明** (入れ子の設定)

入れ物コントロールに他のコントロールを入れます。*child* (子)は入れ物に入れるコントロールです。*parent*(親)は*child*を受けるコントロールで、入れ物の性質を持ったコントロールでなければなりません(例: Frame)。SETPARENTが呼ばれる前に、両方のコントロールは既に作成されていなければなりません。NS Basicに組み込まれた固有のコントロールは正しく動作しますが、サードパーティのコントロールは必要なインターフェースが備わっている保証はありません。

**例**

```
REM SETPARENT Example
'SETPARENT moves one control inside another
ADDOBJECT "TextBox", "input", 5, 5, 140, 18
ADDOBJECT "Frame", "box", 0, 0, 200, 40
SETPARENT input, box
```

SETWAVEVOLUME *volume*

**説明**（ボリュームの設定）

デバイスのメインボリュームの設定を行います。*volume*は0から65535までの値が有効です。

**例**

```
REM SETWAVEVOLUME Example
'SETWAVEVOLUME sets the volume level of the device
SETWAVEVOLUME 0      'Mute on
SETWAVEVOLUME 32768  'Volume 50%
SETWAVEVOLUME 65535  'Full volume
```



SGN(*number*)

**説明** (符合)

SGNは数値の符合を整数にて戻します。必要なパラメータ *number*は数値型です。負数の場合は-1、整数の場合は 1、0の場合は 0 が戻されます。

**例**

```
REM SGN Example
'SGN returns the sign of a number as -1 or 1
DIM Pos, Neg, Zero
Pos = SGN(44)
Neg = SGN(-17)
Zero = SGN(100 - 100)
PRINT "Positive:", Pos
PRINT "Negative:", Neg
PRINT "Zero:", Zero
```

**結果**

```
Positive:    1
Negative:   -1
Zero:        0
```

**関連アイテム**

ABS

SHELLEXECUTE verb, file [,parms]

**説明**（外部プログラムの実行）

SHELLEXECUTEは、別のプロセスとして外部のプログラムを実行します。最初のパラメータverbは、2番目のパラメータfileによって、どんな動作が取られるべきであるかを指示します。3番目のパラメータは、必要であるどんなパラメータでも含めます。

**Table 23:**

Verb	動作内容
Open	プログラムの実行
Print	プリンターに書類を送る（全てのWindowsCE機には対応してません）
Explore	Pocket Internet Explorerを開く

**例**

```
REM SHELLEXECUTE Example
'Register a third party control
shellExecute "open","regsvrce.exe",_
"\windows\s309picture.dll"
'print a document (not on all devices)
ShellExecute "print","myReport.doc"
'open a picture using web browser
shellExecute "explore", "myPict.jpg"
```

**結果**

(varies)

**関連アイテム**

SHOWFULLSCREEN *flags*

## 説明（フルスクリーン）

SHOWFULLSCREENはオペレーティングシステムによって保有されている、スクリーンの各部分を見せたり隠したりするのに使います。*flags*は以下のテーブルのフラグの値を組み合わせて使います。

このステートメントはPocket PCデバイスでのみ動作します。全ての組み合わせがどのデバイスでも安定して動くわけではありませんので、確認のテストは必須です。

フラグ	値
ShowTaskBar	1
HideTaskBar	2
ShowSIPButton	4
HideSIPButton	8
ShowStartIcon	16
HideStartIcon	32
ShowMenuBar	4096
HideMenuBar	8192

## 例

```
REM SHOWFULLSCREEN Example
```

```
'SHOWFULLSCREEN will show/hide areas of the screen
```

```
SHOWFULLSCREEN 2 + 8 + 8192    'Shows full screen
```

SHOWOKBUTTON *true|false*

**説明** (OKボタン)

SHOWOKBUTTONはPocket PCデバイスのメニューバーに「OK」ボタンを表示するのに使います。デフォルトでは「OK」ボタンは表示されず、クローズボタン「X」が表示されます。クローズボタンをタップするとウィンドウが最小化され、実行は継続されます。「OK」ボタンが表示中にタップされると、ウィンドウが閉じられプログラムが終了します。

**例**

```
REM SHOWOKBUTTON Example
'SHOWOKBUTTON shows or hides OK in menu bar
SHOWOKBUTTON true
```

SIN(*number*)

**説明** (サイン/正弦)

SINは、角度を表現している数の正弦をラジアンにて計算します。必要なパラメータ*number*は数値型です。戻り値は -1 から 1 までの範囲の倍精度浮動小数点数 (double) です。

角度の単位を度からラジアンに変換するには、度に  $\pi/180$  を掛けます。

ラジアンから度に変換するには、ラジアンに  $180/\pi$  を掛けます。

**例**

REM SIN Example

'SIN calculates the sine of a number

PRINT "The sine of 0 is " & SIN(0)

**結果**

The sine of 0 is 0

**関連アイテム**

COS, TAN

SLEEP *number*

**説明** (スリープ)

SLEEPは他のプロセスにCPUを譲ります。必要なパラメータ*number*は数値型で、スリープ間隔をミリ秒 (1/1000秒) 単位で指定します。

**例**

```
REM SLEEP Example
'SLEEP yields the CPU for other processes
PRINT NOW
SLEEP(5000)
PRINT NOW
```

**結果**

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

**関連アイテム**

SPACE(*number*)

**説明**（空白文字）

SPACEは、*number*で指定されている数のスペース（空白）から形成する文字列を戻します。必要な引数*number*は有効な数値型です。

**例**

```
REM SPACE Example
'SPACE creates a string of spaces
DIM Spaces, Letter
Spaces = SPACE(4)
FOR i = 0 to 4
    Letter = LEFT(Spaces, i) & CHR(65 + i)
    PRINT Letter
NEXT
```

**結果**

```
A
 B
  C
   D
    E
```

**関連アイテム**

STRING

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

**説明**（文字列の分離）

SPLITは、文字列を *delimiter* ごとに区切り、配列に格納します。必要な引数 *string* は文字列型です。空白（" "）の時、SPLITは空の配列を返します。オプションの引数 *delimiter* は文字列型で、その最初の文字が *string* の内容を区切るのに使われます。デリミタが空白の文字列（" "）の場合、文字列全体を含んでいる単要素の配列が返されます。オプションの引数 *count* は配列に入れる要素の数を指定します。*count* が指定されない場合はデフォルト値の"-1"が設定され、文字列の全てが配列に納められます。オプションの *compare* は検索タイプを指定するもので、Table 10を参照して下さい。

**例**

```
REM SPLIT Example
'SPLIT divides a string into substrings
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)
```

**結果**

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

**関連アイテム**

JOIN



SQR(*number*)

**説明**（平方根）

SQRは、*number*の平方根を倍精度値（double）で戻します。必要なパラメータ*number*は 0 以上の数値型です。

**例**

REM SQR Example

'SQR calculates square root of a number

PRINT "The square root of 69 is " & SQR(69)

**結果**

The square root of 69 is 8.30662386291807

**関連アイテム**

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

**説明**（文字列の比較）

STRCOMPは2つの文字列のアルファベット順の関係を比較し、結果を整数で戻します。必要なパラメータ *string1* および *string2* は文字列型です。オプションのパラメータ *compare* は比較するタイプを指定するために使われるもので、Table 10を参照して下さい。STRCOMPは *string1* が *string2* よりアルファベット順で早い場合は-1を、逆の場合は1を戻します。2つの文字列が同じ場合は0を戻します。

**例**

```
REM STRCOMP Example
'STRCOMP compares two 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 & " precedes " & string2
    ELSEIF Order > 0 THEN
        PRINT string2 & " precedes " & string1
    ELSE
        PRINT string1 & " and " & string2 _
            & " are equivalent"
    END IF
END SUB
```

**結果**

```
Kenny precedes Kyle
Eric and eric are equivalent
Stan precedes Wendy
```

**関連アイテム**

STRING(*number*, *character*)

説明（単一文字の文字列）

STRINGは、*character*で与えられた文字を*number*の数だけ並べた文字列を返します。必要な引数*number*は数値型です。必要な引数*character*は文字型または文字列型です。文字列型の場合は最初の文字が使われます。

例

```
REM STRING Example
'String creates a string repeating a character
DIM Message
Message = STRING(10, "Hello World!")
PRINT Message
Message = STRING(10, CHR(ASC("i")))
PRINT Message
```

結果

```
HHHHHHHHHH
iiiiiiiiii
```

関連アイテム

SPACE

STRREVERSE(*string*)

**説明**（逆さ文字）

STRREVERSEは文字列*string*の文字順を反対にしたものを返します。必要な引数*string*は文字列型です。

**例**

```
REM STRREVERSE Example
'STRREVERSE reverses characters in a string
DIM Show
Show = "Terrence and Phillip"
PRINT "Forwards: " & Show
PRINT "Reverse: " & STRREVERSE(Show)
```

**結果**

```
Forwards: Terrence and Phillip
Reverse: pillihP dna ecnerreT
```

**関連アイテム**

```
SUB procedurename[(arglist)]  
    [statements]  
    [EXIT SUB]  
    [statements]  
END SUB
```

**説明 (SUB プロシージャ)**

SUBは1つのプロシージャを作成します。プロシージャ名 *procedurename*はプロシージャが呼ばれる時に使われる名前です。パラメータリストである*arglist*はカンマで区切られた変数名が入り、プロシージャが呼ばれた時にデータが渡されます。*statements*はプロシージャのボディ部分であり、途中にいくつものEXIT SUB (SUBから抜出す)を入れることが出来ます。

*arglist*の個々の引数は、下で定義されるようにプロシージャに手渡されます：

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

BYVAL：引数*varname*が値渡しで渡されることを示し、オリジナルの値の変更は不可能です。

BYREF：引数*varname*が参照渡しで渡されることを示し、オリジナルの値はプロシージャ内から変更可能です。デフォルトはBYREFです。

戻される値が変数に保存されない場合や、使用されない場合は、FUNCTIONプロシージャはSUBプロシージャと呼ばれます。

SUBプロシージャは、一つの引数または引数がない時は括弧無しで呼ばれ、さらに一つの引数の時は、括弧に入れても結構です。NS Basic/CEは括弧内の式を一つの式と判断し、一つのエレメントの引数リストとは見なしません。

例

```
REM SUB Example
'SUB declares a procedure
DIM PriceA, PriceB
PrintMenu "Wednesday"
PriceA = 53
PriceB = 44
Sort PriceA, PriceB
PRINT "Lowest Price:", PriceA

SUB PrintMenu(day)
  IF day = "Wednesday" THEN
    PRINT "Wednesday is Salisbury Steak day"
  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
```

結果

```
Wednesday is Salisbury Steak day
Lowest Price:    44
```

関連アイテム

CALL, FUNCTION

TAN(*number*)

**説明** (タンジェント)

TANは、角度を表現している数のタンジェントをラジアンで計算します。必要なパラメータ*number*は数値型です。戻り値は倍精度浮動小数点数 (double) です。

角度の単位を度からラジアンに変換するには、度に  $\pi/180$  を掛けます。  
ラジアンから度に変換するには、ラジアンに  $180/\pi$  を掛けます。

**例**

```
REM TAN Example
'TAN calculates the tangent of a number
PRINT "The tangent of 0 is " & TAN(0)
```

**結果**

The tangent of 0 is 0

**関連アイテム**

COS, SIN

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

**説明** (テキストボックス)

TEXTBOXはアウトプットウィンドウ上にエディット可能なテキストボックスを表示します。TEXTBOXはキーボードからのテキスト入力を可能にします。*name*は、オブジェクト参照用にプログラムに追加される変数名です。*xpos* (X軸)、*ypos* (Y軸)、*width* (幅) *height* (高さ) は数値型で、テキストボックスの左上コーナーからの距離と大きさを、ピクセル値で指定します。

注) 埋め込みキャリッジリターンと、テキストラッピングを可能にする場合は、MultilineプロパティをTRUEに設定して下さい。

**サポートされるプロパティ** (PROPERTIES 参照)

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

**サポートされるメソッド** (METHODS 参照)

Hide, Move, SetFocus, Show

**サポートされるイベント** (EVENTS 参照)

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



例

```
REM TextBox Example
'TextBox is an editable text field
ADDOBJECT "TextBox", "Text", 10, 25, 90, 90
Text.Text = "Hello World!"

SUB Text_Click
    PRINT "TextBox Text: " & Text.Text
END SUB
```

結果



関連アイテム  
ADDOBJECT

## TIME

**説明** (システム時刻)  
TIMEは現在のシステム時間を戻します。

**例**

```
REM TIME Example
'TIME returns current system time
DIM RightNow
RightNow = TIME
PRINT "The time now is " & RightNow
```

**結果**

The time now is 10:52:44 PM  
(結果はシステムの設定により異なります)

**関連アイテム**

DATE, NOW

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

**説明** (タイムピッカー オブジェクト)

Timeはアウトプットウィンドウに、標準タイムピッカーオブジェクトを表示する為に使用されます。*name*はオブジェクトを参照する為にプログラムに追加される変数名です。*xpos* (X軸)、*ypos* (Y軸)、*width* (幅)、および*height* (高さ)は数値型で、オブジェクトの画面左上コーナーからの位置とサイズをピクセル値で設定します。Time\_Changeイベントの中で、MSGBOXは使用しないで下さい：エラーを引き起こします。このオブジェクトはWindows CE 2.0デバイスでは使用できません。

**サポートされるプロパティ** (PROPERTIES 参照)

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

**サポートされるメソッド** (METHODS 参照)

Hide, Move, SetFocus, Show

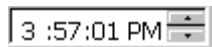
**サポートされるイベント** (EVENTS 参照)

Change, DropDown

**例**

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

**結果**



**関連アイテム**

ADDOBJECT, Events, Methods, Properties

TIMESERIAL(*hour, minute, second*)

**説明**（時刻の構成）

TIMESERIALは与えられた時間、分、および秒から構成される時間を戻します。 *hour*は0から23までの範囲の数値型です。 *minute*および*second*は数値型です。

**例**

```
REM TIMESERIAL Example
'TIMESERIAL builds a time from its parts
DIM FiveThirty, Noon
FiveThirty = TIMESERIAL(11 - 6, 30, 0)
Noon = TIMESERIAL(12, 0, 0)
PRINT "Half past five:", FiveThirty
PRINT "Noon:", Noon
```

**結果**

```
Half past five: 05:30:00 AM
Noon: 12:00:00 PM
(結果はシステムの設定により異なります)
```

**関連アイテム**

DATESERIAL

TIMEVALUE(*time*)

**説明**（時間の値）

TIMEVALUEは通常文字列である*time*の式からの時間を戻します。0時00分00秒から 23時59分59秒までの範囲です。

*time*がタイムセパレータによって区切られた文字列の場合、システムのショートタイムフォーマットによって指定される時-分-秒の順で認識します。（無指定のコンポーネントは0）TIMEVALUEは12時間及び24時間フォーマットを認識します。

タイムセパレータは時間が文字列にフォーマットされる時に時、分、秒を分離する文字で、お使いのシステム設定により決まります。

**例**

```
REM TIMEVALUE Example
'TIMEVALUE returns a time
DIM FiveThirty, Noon
FiveThirty = TIMEVALUE("5:30 PM")
Noon = TIMEVALUE("12:00")
PRINT "Half past five:", FiveThirty
PRINT "Noon:", Noon
```

**結果**

```
Half past five:  05:30:00 PM
Noon:           12:00:00 PM
(結果はシステムの設定により異なります)
```

**関連アイテム**

DATEVALUE

TRIM(*string*)

**説明**（文字列の左右空白の削除）

TRIMは、文字列*string*の最初と最後にある全てのスペースを取り除いた文字列を戻します。必要なパラメータ*string*は文字列型です。

**例**

```
REM TRIM Example
'TRIM trims leading and trailing spaces
DIM Spacey
Spacey = "-K-"
PRINT "(" & Spacey & ")"
PRINT "(" & TRIM(Spacey) & ")"
```

**結果**

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

**関連アイテム**

LTRIM, RTRIM

TYPENAME(variable)

説明 (変数のタイプ名)

TYPENAMEは、変数のタイプを文字列で戻します。必要なパラメータvariableはどのような変数でもかまいません。

Table 24: TYPENAME 戻り値

戻り値	内容
Boolean	ブール値
Byte	バイト値
Currency	通貨値
Date	日付値
Decimal	10進数
Double	倍精度浮動小数点数値
Empty	未初期値
Error	エラー値
Integer	整数値
Long	Long整数値
Nothing	オブジェクト参照を含まないオブジェクト変数
Null	有効な値は無し
Object	一般オブジェクト
Oobjecttype	objecttypeタイプのオブジェクト
Single	単精度浮動小数点数値 (Single)
String	文字列値
Unknown	不明
Variant()	配列

#### 例

```
REM TYPENAME Example
'TYPENAME returns variable type as a string
DIM nInteger, nSingle
nInteger = CINT(44)
PRINT 44 & " is a/an " & TYPENAME(nInteger)
nSingle = CSNG(99.44)
PRINT 99.44 & " is a/an " & TYPENAME(nSingle)
```

#### 結果

```
44 is a/an Integer
99.44 is a/an Single
```

#### 関連アイテム

ISARRAY, ISDATE, ISEMPY, ISNULL, ISNUMERIC, ISOBJECT,  
VARTYPE



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

**説明** (配列の上限)

UBOUNDは、与えられた配列の中の指定された次元における最後のインデックスをlong値で戻します。必要なパラメータ*array*はどのような配列変数でもかまいません。オプションのパラメータ*dimension*はどの次元かを指定し、デフォルトは1です。

**例**

```
REM UBOUND Example
'UBOUND returns upper bound of array dimension
DIM Other, Children(3), Parents(3, 1)
Other = ARRAY("Damien", "Pip", "Wendy")
PRINT "'Other' Upper Bound:", UBOUND(Other)
PRINT "'Children' Upper Bound:", _
    UBOUND(CHILDREN, 1)
PRINT "'Parents' Upper Bounds:", _
    UBOUND(Parents), UBOUND(Parents, 2)
```

**結果**

```
'Other' Upper Bound: 2
'Children' Upper Bound: 3
'Parents' Upper Bounds: 3 1
```

**関連アイテム**

ARRAY, DIM, LBOUND, REDIM

UCASE(*string*)

説明（大文字に変換）

UCASEは文字列*string*の中の全ての小文字を大文字に変換します。必要なパラメータ*string*は文字列型です。

例

```
REM UCASE Example
'UCASE returns string with all uppercase chars
DIM Vet
Vet = "ned"
PRINT Vet & " uppercase is " & UCASE(Vet)
```

結果

ned uppercase is NED

関連アイテム

LCASE

UPDATESCREEN

**説明**（画面の再描画）

UPDATESCREENはアウトプットウィンドウを再描画します。

**例**

```
REM UPDATESCREEN Example
'UPDATESCREEN redraws the output window
UPDATESCREEN
```

**結果**

**関連アイテム**

VARTYPE(*variable*)

説明（変数タイプ（数値））

VARTYPEは、変数のタイプを示す整数を戻します。必要なパラメータ*variable*はユーザ指定されたタイプ以外であればどのような変数でも構いません。

変数が配列の時、配列を示す定数に加え、要素のタイプを示す定数が戻されます。

Table 25: VARTYPE 戻り値

定数	値	説明
vbEmpty	0	未初期（デフォルト）
vbNull	1	有効な値は無し
vbInteger	2	整数
vbLong	3	Long整数
vbSingle	4	単精度浮動小数点数値（Single）
vbDouble	5	倍精度浮動小数点数値（Double）
vbCurrency	6	通貨
vbDate	7	日付
vbString	8	文字列
vbObject	9	オブジェクト
vbError	10	エラー
vbBoolean	11	ブール値
vbVariant	12	可変値（可変値の配列のみ）
vbDataObject	13	Data-accessオブジェクト
vbByte	17	バイト
vbArray	8192	配列

#### 例

```
REM VARTYPE Example
'VARTYPE returns variable type as an integer
DIM nInteger, Single
nInteger = CINT(44)
PRINT 44 & " is VARTYPE " & VARTYPE(nInteger)
nSingle = CSNG(99.44)
PRINT 99.44 & " is VARTYPE " & VARTYPE(nSingle)
```

#### 結果

```
44 is VARTYPE 2
99.44 is VARTYPE 4
```

#### 関連アイテム

ISARRAY, ISDATE, ISEMPY, ISNULL, ISNUMERIC, ISOBJECT,  
TYPENAME

WAITCURSOR true|false

**説明**（砂時計カーソル）

待ちカーソルを表示するためにはWAITCURSORにTRUEを設定してください。表示を消すためにはFALSE（デフォルト）を設定してください。カーソルのデザインはデバイスによって異なります。

**例**

```
REM WAITCURSOR Example
WAITCURSOR TRUE
SLEEP 5000
WAITCURSOR FALSE
```

**結果**



**関連アイテム**

WEEKDAY(*date* [, *firstdayofweek*])

#### 説明（曜日値）

WEEKDAYは、与えられた日付の曜日を示す整数を返します。必要なパラメータ*date*は、日付型、文字列型、または数値型です。オプションのパラメータ*firstdayofweek*は、指定されない時は日曜日がデフォルトです。戻り値は下表にある整数です。

*firstdayofweek*はDateDiffを参照して下さい。

Table 26: WEEKDAY 戻り値

定数	値	説明
vbSunday	1	日曜日
vbMonday	2	月曜日
vbTuesday	3	火曜日
vbWednesday	4	水曜日
vbThursday	5	木曜日
vbFriday	6	金曜日
vbSaturday	7	土曜日

#### 例

REM WEEKDAY Example

'WEEKDAY returns day of week as an integer

DIM IndepDay, Birthday

IndepDay = WEEKDAY("July 4, 1776")

PRINT "WEEKDAY of July 4, 1776:", IndepDay

Birthday = WEEKDAY("12/27/70")

PRINT "WEEKDAY of 12/27/70:", Birthday

#### 結果

WEEKDAY of July 4, 1776: 5

WEEKDAY of 12/27/70: 1

#### 関連アイテム

WEEKDAYNAME(*number*[, *abbreviate*[, *firstdayofweek*]])

**説明**（曜日名）

WEEKDAYNAMEは曜日の文字列を返します。必要な引数 *number*は曜日の番号を表している1から7までの範囲の整数です。オプションの*abbreviate*（短縮）をTRUEにする場合は3文字で示す曜日名が返されます。デフォルトはFALSEですので、完全な曜日名が返されます。オプションのパラメータ*firstdayofweek*が指定されない時は、日曜日がデフォルトです。*firstdayofweek*はDateDiffを参照して下さい。

**例**

```
REM WEEKDAYNAME Example
'WEEKDAYNAME returns string name of day
DIM Day1, Day2
Day1 = 1
PRINT WEEKDAYNAME(Day1)
Day2 = 2
PRINT WEEKDAYNAME(Day2, TRUE, vbMonday)
```

**結果**

```
Sunday
Tue
```

**関連アイテム**



```
WHILE condition  
    [statements]  
WEND
```

**説明 (WHILE...WENDループ)**

与えられた条件がTRUEである間、WHILE...WENDはひと固まりのステートメントを繰り返します。必要なコンポーネント *condition* は、TRUEまたはFALSEに評価する有効な式です。オプションのコンポーネント *statements* はループの個々の繰返しの間に実行されます。WHILE...WENDステートメントはネストすることが出来、ネストされたループはWENDステートメントによりループを終了し一つ外側のレベルに実行を移します。

**例**

```
REM WHILE..WEND Example  
'WHILE..WEND repeats a group of statements  
DIM Counter  
Counter = 1  
WHILE Counter < 5  
    PRINT "Counter = " & Counter  
    Counter = Counter + 1  
WEND
```

**結果**

```
Counter = 1  
Counter = 2  
Counter = 3  
Counter = 4
```

**関連アイテム**

DO...LOOP, FOR...NEXT, FOR EACH...NEXT

```
WITH object  
    [statements]  
END WITH
```

説明（オブジェクトのオペレーション）  
オブジェクト名を毎回使わずに、オブジェクトのオペレーションをまとめて行う。  
Windows CE 4.0以降のみ利用可能。

**例**

```
AddObject "CommandButton", "CB", 124, 60, 108, 21  
With CB  
    .FontBold = True  
    .Caption = "Tap Me!"  
    .FontItalic = True.  
End With
```

**結果**

(太字のイタリックで"Tap ME!"と書かれたCommandButton)

**関連アイテム**

CLASS

```
result = x XOR y
```

**説明**（排他的論理和演算）

XORは2つの式の排他論理を評価します。xかyのどちらか一方だけがTRUEの時、*result*がTRUEになり、それ以外はFALSEになります。

XORは2つ数値式のビット単位の比較も行います。xおよびyの相対するビットのどちらか一方だけが1の場合、*result*のxとyに相対するビットは1になり、それ以外は0をセットします。

**例**

```
REM XOR Example
'XOR performs exclusive disjunctions
DIM Test1, Test2, x, y
x = 2
y = 9
Test1 = x > 0 XOR y < 10
Test2 = x > 0 XOR y > 10
PRINT "Logical:"
PRINT "  x > 0 XOR y < 10 = " & CSTR(Test1)
PRINT "  x > 0 XOR y > 10 = " & CSTR(Test2)
PRINT "Bitwise:"
PRINT "  x XOR y = " & (x XOR y)
```

**結果**

```
Logical:
  x > 0 XOR y < 10 = False
  x > 0 XOR y > 10 = True
Bitwise:
  x XOR y = 11
```

**関連アイテム**

AND, EQV, IMP, NOT, OR

YEAR(*date*)

説明（日付：年）

YEARは与えられた日付の年を表している整数を戻します。  
必要なパラメータ*date*は日付型です。

例

```
REM YEAR Example
'YEAR returns year of a date as an integer
DIM IndepYear, Birthyear
IndepYear = YEAR("July 4, 1776")
PRINT "YEAR of July 4, 1776:", IndepYear
Birthyear = YEAR("12/27/70")
PRINT "YEAR of 12/27/70:", Birthyear
```

結果

```
YEAR of July 4, 1776: 1776
YEAR of 12/27/70: 1970
```

関連アイテム

DAY, HOUR, MINUTE, MONTH, NOW, SECOND, TIME

## 6. アドバンストピック (Advanced Topics)

### 6.1 プログラミング慣習

プログラミング慣習はプログラムの作成、読解、デバッグ、および維持を援助する為の提案です。とくに制限はありませんが、プログラミング慣習は以下のものを含みます：

- 命名のガイドライン
- テキスト形式のガイドライン
- コメントのガイドライン

プログラミング慣習の遂行は、文脈依存情報と、改善された視覚的読み易さを提供するのを助けるでしょう。

#### 6.1.1 命名のガイドライン

プログラムにおける文脈依存情報を最も提供しやすい方法は、簡潔で、意味を伝える名前を定数、変数、およびプロセスに与えることです。

ある程度重要な名前は、目的か用途を記述するために、出来る限り説明的にすると良いでしょう。複数の単語が使用されるとき、大文字／小文字や、アンダースコア文字(\_)を使用して、単語を区切ります。

定数名にはすべて大文字を使用し、単語間にはアンダースコア文字(\_)がおかれます。

変数名には小文字を使用し、各単語の最初の文字を大文字で書き、変数名の中のアンダースコアの使用を抑えます。追加情報…変数が格納するタイプを記述する3文字の接頭語から各変数名を始めます…(整数型 (INTEGER) 用にint、文字列型 (STRING) 用にstr、ブール型 (BOOLEAN) 用にbln, …)

プロセス名には小文字を使用し、各単語の最初の文字を大文字で書きます。アンダースコアの使用および大文字／小文字の使い分けは、定数及び変数との区別を容易にするでしょう。

### 6.1.2 テキスト形式のガイドライン

テキスト形式はプログラムの視覚的な読み易さを向上させる方法です。ステートメントをグループにしたり、分類したりするのに、空白ラインを入れ、ステートメントの論理的な構造やネスト（入れ子）の為に、タブ等を使用し各ステートメントの始まりの位置を変えます。

セクションを切り離しておくために、定数定義、変数定義、および各プロシージャの定義の後に空白行を挿入します。スクリプトレベルにおいては、関連するステートメントの後やプロシージャの中で、空白行を挿入することによってグループ形成を助けます。

標準のタブインデントはスペース2つです。プロシージャ、ループ、及び条件付き実行ステートメントがネストにされたグループであることを見せる為に、少なくともタブ1つは使用する。

### 6.1.3 コメントのガイドライン

プログラマーは何をどの様にしたいか、また、何が成し遂げられるかのコメント（注釈）を記載すべきです。以下の様にコメントを使用します。

- 誰がそれを書いて、どんなサービスを提供することができて、いつ書かれて、誰がプログラムへの権利を所有するのか、さらにアップデートの期日及びその目的（原因）、に関する情報を各プログラムの初めに含める。
- 期待される入力及び出力と共に、各プロシージャの目的を説明する前書きを入れる。それぞれの渡される変数の目的が明白でない時や、制限が伴う時は、ラインを追加して説明する。
- 定数および変数の定義の後に、範囲と用法に関する情報を加えて、簡潔な説明を追加する。
- 関連するステートメントからできるグループの上に、ステートメントがしていることについて説明する。
- 条件付きの実行ステートメントには、各条件とどう生じるのかをはっきり明示する。

## 6.2 エラー処理

完全な世界には、決してどんなエラーも存在しない。我々のプログラムはめったに完全な世界にならない！多くのエラーから、ユーザー（しばしばプログラマー自身）を保護する2つのテクニック、防衛的なプログラミングとエラートラッピングを使用します。

エラーを取り扱う基本的な考え方は、プログラムのどの部分が実行時エラーを引き起こすかを予測することと、それを扱う特別なプログラムコードをセットすることです。

### 6.2.1 防衛的なプログラミング

プログラムが実行される時の仮説を減らす方法で防衛的にプログラムを書くことは、実行時エラーを避けるのを助けます。さらに、常に以下のことをするべきです。

- 使用される前に適切なデータタイプを保持するのを確実にする為に、変数を定義した後に初期化する。
- 使用される前にすべての変数が定義されるのを確実にする為に、OPTION EXPLICITステートメントを使用する。これはスペルミスによるエラーを防ぐのを助けます。
- ユーザによって入力される変数が期待されたデータ型の値を含むのを確実にする為に、VARTYPEファンクションを使用する。
- 数値式で使われる前に、数値が有効な範囲の中にあることを確認する。
- 配列にアクセスする前に、要素の数を知っておく。

より防衛的にプログラムを作ることにより、実行時エラーの起こる機会は減ります。良い防衛的なプログラマーは、ユーザーからの作用によって引き起こされる論理エラーを除去して、プログラムを保護することができます。

### 6.2.2 エラートラッピング

多くの実行時エラーを防ぐ為に、防衛的なプログラミングを実行すべきです。防衛的なプログラミングではプログラムからエラーを取り除けない時、ON ERRORステートメントを使用することにより、発生するエラーを捕らえることができます。この方法がエラートラッピングであり、NS Basic/CEがエラーを通知される前に、エラー処理を行うことができます。

防衛的なプログラミングによって防ぐことができない実行時エラーは、一般的にプログラムがオペレーティングシステムと相互作用している時に発生します。利用可能でないかもしれないシステムリソースを要求するので、ファイルオペレーションは最も一般的なエラーを引き起こします。

NS Basic/CEはプロシージャの中でのみエラートラッピングを許容するので、プロシージャ単位でエラートラッピングを有効にしたり無効にしたりするのは容易です。プログラムがエラーを捕らえた時、エラーハンドラーを通してそれを処理します。エラーハンドラーは単に1ブロックのステートメントで、プログラムにエラーを処理させる為に実行されます。ファイルオペレーションに伴う簡単なエラートラッピングと取り扱いを見てみましょう。

### 6.2.3 エラートラッピング ファイルオペレーション

```
ADDOBJECT "File", "MyFile"
DIM FileOpen

FileOpen = Open("Some non-existent filename")
IF FileOpen THEN
    PRINT "File opened successfully."
ELSE
    PRINT "Unable to open file."
END IF

FUNCTION Open(filename)
    ON ERROR RESUME NEXT
    MyFile.Open filename, 1
    IF ERR <> 0 THEN
        'This is the error handler that is used
        'when a file cannot be opened for reading.
        'This simple example just returns FALSE.
        Open = FALSE
        ERR.Clear
        EXIT FUNCTION
    END IF
    MyFile.Close
    Open = TRUE
END FUNCTION
```

#### 結果

Unable to open file.

エラーハンドラーが行ういくつかの処理は

- 有用なメッセージを表示して、オペレーションを再試行する。
- 1つまたは複数の変数をデフォルト値に設定することによって、エラーを修正する。(入力を最大値に制限することができる)
- 独自のエラーメッセージを表示し、クリーンアップ（ファイルのアップデート等）を行ってから、プログラムを終了する。



# A P P E N D I X

## A

### A. エラーコード

5	不正なプロシージャコールまたは引数
6	オーバーフロー
7	メモリ不足
9	添字（インデックス）範囲外
10	配列の固定または一時的ロック
11	0による除法（零因子）
13	タイプミスマッチ
14	文字列のスペース不足
17	要求されたオペレーションが実行出来ない
28	スタックスペース不足
35	SubまたはFunctionの未定義
48	DLLロードのエラー
51	内部エラー
52	不適切なファイル名または番号
53	ファイルが見つからない
54	不適切なファイルモード
55	ファイルがすでに開いている
57	デバイス I/O エラー
58	ファイルがすでに存在している
61	ディスクが満タン
62	ファイルの終わりに到達
67	ファイルの持ちすぎ
68	デバイス利用不可能
70	アクセス許可が下りない
71	ディスクの準備ができていない
74	別のドライブで名前の変更ができない
75	経路（パス）/ファイルアクセスエラー
76	パスが見つからない
91	オブジェクト変数が設定されていない
92	Forループが初期化されていない
94	Nullの不適切な使用
322	必要な一時的ファイルが作成出来ない
424	オブジェクトが必要
429	ActiveXコンポーネントがオブジェクトを作成できない
430	クラスがAutomationをサポートしない
432	Automationオペレーション中にファイル名またはクラス名が見つからない
438	オブジェクトがプロパティまたはメソッドをサポートしていない
440	Automationエラー

445	オブジェクトがこの動作をサポートしない
446	オブジェクトが命名された引数をサポートしない
447	オブジェクトが現エリア設定をサポートしない
448	命名された引数が見つからない
449	引数はオプションではない
450	引数の数が違うか不適切なプロパティの割当て
451	オブジェクトが集合でない
453	指定DLLファンクションが見つからない
455	コードリソースロックエラー
457	このキーはすでにこの集合の要素に結びついている
458	VBScriptをサポートしていないAutomationタイプの変数が使用
500	変数が未定義
501	不法代入
502	書き込みにはオブジェクトが安定状態でない
503	初期化にはオブジェクトが安定状態でない
504	作成にはにオブジェクトが安定状態でない
505	無効な、または不適当なリファレンス
506	クラスは定義されていない
507	エクセプション発生
32811	要素が見つからない
32812	指定された日付は、現在設定されてるカレンダーでは使えない

# A P P E N D I X

## B

### B. 定数

Color	vbBLACK, vbRED, vbGREEN, vbBLUE, vbYELLOW, vbBLUE, vbMAGENTA, vbCYAN, vbWHITE
Comparison	vbBINARYCOMPARE, vbTEXTCOMPARE
Date/Time	vbSUNDAY, vbMONDAY, vbTUESDAY, vbWEDNESDAY, vbTHURSDAY, vbFRIDAY, vbSATURDAY, vbFIRSTJAN1, vbFIRSTFOURDAYS, vbFIRSTFULLWEEK, vbUSESYSYSTEM, vbUSESYSYSTEMDAYOFWEEK
Date Format	vbGENERALDATE, vbLONGDATE, vbSHORTDATE, vbLONGTIME, vbSHORTTIME
MSGBOX	Display options (add to combine):  vbOKONLY, vbOKCANCEL, vbABORTRETRYIGNORE, vbYESNOCANCEL, vbYESNO, vbRETRYCANCEL, vbCRITICAL, vbQUESTION, vbEXCLAMATION, vbINFORMATION, vbDEFAULTBUTTON1, vbDEFAULTBUTTON2, vbDEFAULTBUTTON3, vbDEFAULTBUTTON4, vbAPPLICATIONMODAL, vbSYSTEMMODAL
	Return values:  vbOK, vbCANCEL, vbABORT, vbRETRY, vbIGNORE, vbYES, vbNO
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

## 索引

- &, 197  
' , 29  
16進法表現, 124  
8進法表現, 159
- A -  
ABS, 56  
AddItem, 146  
ADDOBJECT, 57, 137, 162, 166  
Alignment, 173  
AND, 19, 60  
ANSI, 62, 71  
ANSI 文字, 71  
ANSI 文字コード, 62  
APIフアンクションの定義, 94  
ARRAY, 61  
ASC, 62  
ASCB, 62  
ASCW, 62  
ATN, 63
- B -  
BackColor, 173  
BorderStyle, 166, 173  
Bottom, 173  
BREAK, 36, 64  
BYE, 65
- C -  
CALL, 66  
Caption, 173  
CBOOL, 79  
CBYTE, 79  
CCUR, 79  
CDATE, 79  
CDBL, 79  
CE Screen, 43  
CHAIN, 67  
Change, 104  
CheckBox, 69  
CHR, 71  
CHRB, 71  
CHRW, 71  
CINT, 79  
CLASS, 72  
Class\_Initialize, 72  
Class\_Terminate, 72  
Clear, 146  
Click, 104  
CLNG, 79  
Cls, 167  
Coding Conventions, 237  
ComboBox, 29, 74  
CommandButton, 76  
CONST, 78  
COS, 82  
Creating a Program, 30  
CSNG, 79  
CSTR, 79  
CURRENTPATH, 83
- D -  
Data Types  
    Array, 16  
    Boolean, 15  
    Numeric, 15  
DATE, 84, 85, 173  
DATEADD, 87  
DATEDIFF, 88  
DATEPART, 90  
DATESERIAL, 91  
DATEVALUE, 92  
DAY, 93  
DbClick, 104  
Debugging a Program, 34  
DECLARE, 94  
DEVICE\_CHANGE, 188  
DIM, 95  
DLL, 94  
DOEVENTS, 96  
DrawCircle, 167  
DrawLine, 167  
DrawPicture, 167  
DrawPoint, 167  
DrawText, 167  
DrawWidth, 166  
DropDown, 104
- E -  
Editing a Program, 31

EmCE エミュレータ, 54  
 EQV, 19, 99  
 ERASE, 100  
 Err, 101  
 ESCAPE, 102  
 EVAL, 103  
 Events  
   Change, 104  
   Click, 104  
   DbtClick, 104  
   DropDown, 104  
   GotFocus, 104  
   KeyDown, 104  
   KeyPress, 104  
   KeyUp, 104  
   LostFocus, 104  
   Timer, 104  
 Execute, 36, 106  
 EXECUTEGLOBAL, 107  
 EXIT, 108  
 EXP, 109  
  
 - F -  
 FillColor, 166  
 FillStyle, 166  
 FILTER, 110  
 firstdayofweek, 88  
 firstweekofyear, 88  
 FIX, 112  
 FontStrikethru, 174  
 FontTransparent, 166  
 FontUnderline, 174  
 FontWeight, 174  
 ForeColor, 174  
 Form\_Load, 44  
 FORMATCURRENCY,  
   115  
 FORMATDATETIME,  
   115  
 FormatLong, 174  
 FORMATNUMBER, 115  
 FORMATPERCENT, 115  
 Frame, 117, 199  
 FUNCTION, 72, 118  
 Functions  
   ABS, 56  
   ARRAY, 61  
   ASC, 62  
   ATN, 63  
   CBOOL, 79  
   CBYTE, 79  
   CCUR, 79  
   CDATE, 79  
   CDBL, 79  
   CHR, 71  
   CINT, 79  
   CLNG, 79  
   COS, 82  
   CSNG, 79  
   CSTR, 79  
   DATE, 84  
   DATEADD, 87  
   DATEDIFF, 88  
   DATEPART, 90  
   DATESERIAL, 91  
   DATEVALUE, 92  
   DAY, 93  
   ESCAPE, 102  
   EVAL, 103  
   EXP, 109  
   FILTER, 110  
   FIX, 112  
   FORMATCURRENCY,  
     115  
   FORMATDATETIME,  
     115  
   FORMATNUMBER,  
     115  
   FORMATPERCENT,  
     115  
   GETCOMMANDLINE,  
     120  
   GETLOCALE, 121  
   GETREF, 122  
   GETSERIALNUMBE  
     R, 123  
   HEX, 124  
   HOUR, 125  
   IMP, 127  
   INPUTBOX, 128  
   INSTR, 129  
   INSTRREV, 129  
   INT, 130  
   JOIN, 133  
   LBOUND, 138  
   LCASE, 139  
   LEFT, 140  
   LEFTB, 140  
   LEN, 141  
   LENB, 141  
   LOG, 144  
   LTRIM, 145  
   MID, 148  
   MIDB, 148

MINUTE, 149	KeyPreview, 134
MONTH, 151	NSBVersion, 158
MONTHNAME, 152	GotFocus, 104
MSGBOX, 153	Group, 162
NOW, 157	
OCT, 159	- H -
PLAYSOUND, 170	Height, 174
REPLACE, 182	HEX, 124
RGB, 183	hex值, 102
RIGHT, 184	Hide, 146
RIGHTB, 184	HOUR, 125
RND, 185	HScrollbar, 190
ROUND, 186	Hwnd, 174
RTRIM, 187	
SECOND, 192	- I -
SETLOCALE, 121	IMP, 19, 127
SGN, 201	INPUTBOX, 128
SIN, 205	Installation, 11
SPACE, 207	INSTR, 129
SPLIT, 208	INSTRREV, 129
SQR, 209	INT, 130
STRCOMP, 210	IntegralHeight, 174
STRING, 211	IR_DISCOVERED, 188
STRREVERSE, 212	IS, 131
TAN, 215	
TIME, 218	- J -
TIMESERIAL, 220	JOIN, 133
TIMEVALUE, 221	
TRIM, 222	- K -
TYPENAME, 223	KeyboardStatus, 135
UBOUND, 225	KeyboardStatusChange
UCASE, 226	d, 135
UNESCAPE, 102	KeyDown, 104, 168
VARTYPE, 228	KeyPress, 104, 168
WEEKDAY, 231	KeyPreview, 134
WEEKDAYNAME,	KeyUp, 104, 168
232	KILLFOCUS, 136
YEAR, 236	
- G -	- L -
GETCOMMANDLINE,	LBOUND, 138
120	LCASE, 139
GETLOCALE, 121	LEFT, 140
GETREF, 122	LEFTB, 140
GETSERIALNUMBER,	LEN, 141
123	LENB, 141
Global	List, 29, 174
CURRENTPATH, 83	ListBox, 29, 142
Global Properties	ListCount, 175
KeyboardStatus, 135	ListIndex, 175
KeyboardStatusCha	Locale ID, 121
nged, 135	LOG, 144

LostFocus, 104  
 LTRIM, 145  
  
**- M -**  
 MaxLength, 175  
 Menus, 23  
 Methods  
     AddItem, 146  
     Clear, 146  
     Hide, 146  
     Move, 146  
     RemoveItem, 146  
     SetFocus, 147  
     Show, 147  
 MID, 148  
 MIDB, 148  
 MINUTE, 149  
 MOD, 150  
 MONTH, 151  
 MONTHNAME, 152  
 MouseDown, 168  
 MouseMove, 168  
 MouseUp, 168  
 MSGBOX, 153  
 MSGBOX 戻り値, 154  
 MultiLine, 175  
 Multi-Line Statements, 14  
  
**- N -**  
 Name, 175  
 Naming guidelines, 237  
 NET\_CONNECT, 188  
 NET\_DISCONNECT, 188  
 NewIndex, 175  
 NOT, 19, 156  
 NOW, 157  
 NS Basic/CEのバージョン, 158  
 NSBVersion, 158  
  
**- O -**  
 Objects, 20  
     ADDOBJECT, 137, 162, 166  
     CheckBox, 69  
     ComboBox, 74  
     CommandButton, 76  
     Date, 85  
     Err, 101  
     Frame, 117  
     HScrollbar, 190  
     ListBox, 142  
     OUTPUT, 165  
     TextBox, 216  
     Time, 219  
     VScrollbar, 190  
 OCT, 159  
 OFF\_AC\_POWER, 188  
 OKボタン, 204  
 ON ERROR, 160  
 ON\_AC\_POWER, 188  
 Operator  
     XOR, 235  
 Operators, 17  
     AND, 60  
     Arithmetic, 17  
     EQV, 99  
     IS, 131  
     MOD, 150  
     NOT, 156  
     OR, 164  
     Relational, 18  
 OPTION EXPLICIT, 161  
 Options, 51  
 OR, 19, 164  
 OUTPUT, 165  
 Output Window, 25  
 Overview, 32  
  
**- P -**  
 Picture, 166  
 PictureBox イベント, 168  
 PictureBox プロパティ, 166  
 PictureBox メソッド, 167  
 PLAYSOUND, 170  
 PRINT, 172  
 PRIVATE, 72, 78  
 Properties  
     Alignment, 173  
     BackColor, 173  
     BorderStyle, 173  
     Bottom, 173  
     Caption, 173  
     Date, 173  
     FontStrikethru, 174  
     FontUnderline, 174  
     FontWeight, 174  
     ForeColor, 174



FormatLong, 174  
 Height, 174  
 Hwnd, 174  
 IntegralHeight, 174  
 Left, 174  
 List, 174  
 ListCount, 175  
 ListIndex, 175  
 MaxLength, 175  
 MultiLine, 175  
 NewIndex, 175  
 Right, 175, 176  
 Scrollbars, 176, 177  
 SelLength, 176  
 SelStart, 176  
 SelText, 176  
 Sorted, 176  
 Style, 176  
 Tabstop, 176  
 Tag, 176  
 Text, 176  
 Top, 177  
 Value, 177  
 Visible, 177  
 Width, 177  
 WindowLong, 177  
 Property Editor, 29  
 PROPERTY GET, 72  
 PROPERTY LET, 72  
 PROPERTY SET, 72  
 PUBLIC, 72, 78

- R -  
 RANDOMIZE, 179  
 REDIM, 180  
 Refresh, 167  
 REM, 181  
 RemoveItem, 146  
 REPLACE, 182  
 RESTORE\_END, 188  
 RGB, 183  
 RGBカラー, 183  
 Right, 175, 176, 184  
 RIGHTB, 184  
 RND, 185  
 ROUND, 186  
 RS232\_DETECTED, 188  
 RTRIM, 187  
 RUNAPPATEVENT, 188  
 RUNAPPATTIME, 189  
 Running a Program, 33

- S -  
 Saving and Loading a  
   Program, 37  
 ScaleHeight, 166  
 ScaleLeft, 166  
 ScaleMode, 167  
 ScaleTop, 167  
 ScaleWidth, 167  
 ScaleX, 167  
 ScaleY, 168  
 Scrollbars, 176, 177  
 SECOND, 192  
 seed, 185  
 SELECT CASE, 193  
 SelLength, 176  
 SelStart, 176  
 SelText, 176  
 SENDKEY, 195  
 Serial Number, 11  
 SET, 196  
 SetFocus, 147  
 SETLOCALE, 121  
 SETMENU, 197  
 SETPARENT, 199  
 SetScale, 168  
 SETWAVEVOLUME, 200  
 SGN, 201  
 SHELLEXECUTE, 202  
 Show, 147  
 SHOWFULLSCREEN,  
   203  
 SIN, 205  
 SLEEP, 206  
 Sorted, 176  
 SPACE, 207  
 SPLIT, 208  
 SQR, 209  
 Statements  
   ADDOBJECT, 57  
   BREAK, 64  
   BYE. See  
   CALL, 66  
   CHAIN, 67  
   CLASS, 72  
   CONST, 78  
   DECLARE, 94  
   DIM, 95  
   DOEVENTS, 96  
   ERASE, 100  
   EXECUTE, 106  
   EXECUTEGLOBAL,  
     107

EXIT, 108  
 FUNCTION, 118  
 KILLFOCUS, 136  
 ON ERROR, 160  
 OPTION EXPLICIT, 161  
 PRINT, 172  
 RANDOMIZE, 179  
 REDIM, 180  
 REM, 181  
 RUNAPPATEVENT, 188  
 RUNAPPATIME, 189  
 SELECT CASE, 193  
 SENDKEY, 195  
 SET, 196  
 SETMENU, 197  
 SETPARENT, 199  
 SETWAVEVOLUME, 200  
 SHELLEXECUTE, 202  
 SHOWFULLSCREEN, 203  
 SLEEP, 206  
 SUB, 213  
 WAITCURSOR, 230  
 WITH, 234  
 Step, 36  
 STRCOMP, 210  
 STRING, 211  
 STRREVERSE, 212  
 Style, 176  
 SUB, 72, 213  
 SYNC\_END, 188  
 System Requirements, 11  
  
 - T -  
 Tabstop, 176  
 Tag, 167, 176  
 TAN, 215  
 Text, 176  
 TextBox, 216  
 TextHeight, 168  
 TextWidth, 168  
 TIME, 218, 219  
 TIME\_CHANGE, 188  
 Timer, 104  
 TIMESERIAL, 220  
 TIMEVALUE, 221  
  
 Top, 177  
 Trace, 36  
 TRIM, 222  
 TYPENAME, 223  
 TZ\_CHANGE, 188  
  
 - U -  
 UBOUND, 225  
 UCASE, 226  
 UNESCAPE, 102  
 Unicode, 62, 71  
  
 - V -  
 Value, 177  
 Variables, 15  
 VARTYPE, 228  
 Visible, 177  
 Visual Designer, 26  
 VScrollBar, 190  
  
 - W -  
 WAITCURSOR, 230  
 WAKEUP, 188  
 WEEKDAY, 231  
 WEEKDAYNAME, 232  
 Width, 177  
 WindowLong, 177  
 Windows CE 4.0, 102, 121, 122, 234  
 Windowsディレクトリ, 94  
 WITH, 234  
  
 - X -  
 XOR, 19, 235  
  
 - Y -  
 YEAR, 236  
  
 - あ -  
 アークタンジェント, 63  
 アウトプット  
   ウインドウ, 25  
 アウトプットオブジェクト, 165  
 アルファベット, 121  
 アンパーサンド, 197  
 イベント, 20, 96  
 イベントオブジェクト, 104

イベントによってアプリ  
起動, 188  
入れ子の設定, 199  
インストール, 11  
インプットボックス, 128  
エディタ, 23  
エラー, 238  
エラーコード, 241  
エラー処理, 101  
演算子, 17  
算術, 17  
論理, 18  
関係, 18  
大文字に変換, 226  
置換え, 31  
音の再生, 170  
オブジェクト, 20  
ブローシージャ, 146  
属性, 173  
比較, 131  
追加, 57  
オブジェクトのオペレー  
ション, 234  
オブション, 51  
オンスクリーン・キーボ  
ード, 135

- カ -  
概数, 186  
外部プログラムの実行,  
202  
概要, 32  
画面の再描画, 227  
キーボード ショートカ  
ット, 197  
キーボード・レイアウト  
, 121  
キーボードの状態, 135  
キーボード入力, 195  
キー入力, 195  
キー入力のフラグ, 134  
偶数, 186  
空白文字, 207  
クラス, 72, 107  
クローズボタン, 25  
グローバル変数, 134,  
135  
グローバル領域で式又は  
ファイルの実行, 107  
現在の日時, 157  
現在の日付, 84  
コサイン/余弦, 82

コマンドライン, 120  
コメント, 181  
小文字に変換, 139  
コントロール, 20  
コンパイル時エラー, 34  
コンボボックス, 74

- さ -  
サイン/正弦, 205  
逆さ文字, 212  
サブルーチン, 122  
時間の値, 221  
式, 17  
式又はファイルの実行,  
106  
時刻, 121  
時刻: 分, 149  
時刻: 時, 125  
時刻: 秒, 192  
時刻の構成, 220  
指数  $E^x$ , 109  
システム時刻, 218  
自然対数, 144  
実行時エラー, 34  
実行時エラーの処理, 160  
実行の一時停止, 64  
条件ステートメント, 126  
条件選択, 193  
シリアル番号, 11, 123  
スクロールバー, 190  
砂時計カーソル, 230  
スペース, 207  
スリープ, 206  
整数へ変換, 130  
絶対値, 56

- た -  
ダイアログボックス, 153  
ダイナミック配列作成,  
61  
タイプ変換, 79  
タイムピッカー, 219  
種値, 185  
単一文字の文字列, 211  
タンジェント, 215  
地域ID, 121  
チェックボックス, 69  
通貨, 121  
ツールバー, 48  
ツールボックス, 47  
月の数値表現, 151  
月の文字列表現, 152

定数, 243  
定数の定義, 78  
データタイプ, 15  
    ブール, 15  
    数値, 15  
    文字列, 16  
    色, 16  
    配列, 16  
テキストファイル, 38  
テキストボックス, 216  
デスクトップ, 39  
デバッグ, 34  
検索, 31  
動作環境, 11  
特殊文字のhex値, 102  
特定時刻にアプリ起動,  
    189  
  
- な -  
日時間隔, 88  
日時の加減算, 87  
年、月、日による日付構  
    成, 91  
  
- は -  
排他的論理和演算, 235  
バイト, 62, 71  
ハイフン, 197  
配列, 61  
    上限, 225  
    下限, 138  
    再配分, 180  
    巡回, 114  
    文字列の抜き出し, 110  
    解放, 100  
端数の省略, 112  
ピクチャーボックス, 166  
ビジュアルデザイナー,  
    26  
    メニュー, 27  
日付, 121  
日付: 年, 236  
日付: 日, 93  
日付オブジェクト, 85  
フォーマット, 121  
ファンクション, 122  
フォーカスを取り除く,  
    136  
フォーマット, 115  
フォーム, 21  
    イベント, 21  
    メソッド, 21  
  
複数行ステートメント,  
    14  
符合, 201  
部分的日付, 90  
プリント, 172  
フルスクリーン, 203  
フレーム, 117  
プログラミング慣習, 237  
プログラム  
    エディット, 31  
    オープン, 37  
    デバッグ, 34  
    フォーマット, 31  
    ロード及び実行, 67  
    作成, 30  
    保存, 37  
    実行, 33  
    終了, 65  
プログラムエディタ, 23  
プログラムへのパス, 83  
プロシージャ, 19, 118,  
    213  
プロシージャを呼ぶ, 66  
プロジェクトエクスポ  
    ーラ, 45  
プロパティ, 72, 173  
プロパティウィンドウ,  
    46  
プロパティエディタ, 29  
平方根, 209  
ヘルプボタン, 25  
変数, 15, 72  
変数タイプ, 228  
変数タイプのチェック,  
    132  
変数定義, 95  
変数定義の強制, 161  
変数のタイプ名, 223  
変数の設定, 196  
変数名, 16  
ボタン作成, 76  
ボタン定数, 153  
ボリュームの設定, 200  
  
- ま -  
命名のガイドライン, 237  
メソッド, 72, 146  
メニュー, 39, 197  
メニューエディタ, 29,  
    49  
メモリスペース確保, 95  
文字の置換え, 182

文字列  
  中央抜き出し, 148  
  分離, 208  
  右側抜き出し, 184  
  右空白の削除, 187  
  左側抜き出し, 140  
  左右空白の削除, 222  
  左空白の削除, 145  
  日付構成, 92  
  検索, 129  
  比較, 210  
  連結, 133  
  関数評価, 103  
文字列／変数の長さ, 141  
  
- や -  
曜日値, 231  
曜日名, 232  
  
- ら -  
ラジオボタン, 162  
  
ラベル, 137  
乱数ジェネレータの初期化, 179  
乱数の作成関数, 185  
リストボックス, 142  
リテラル, 15  
リファレンス, 55  
リファレンス・ポインタ, 122  
ループ, 96, 97, 113, 233  
ループから抜出す, 108  
論理エラー, 34  
論理積演算, 60  
論理等価演算, 99  
論理否定演算, 156  
論理包含演算, 127  
論理和演算, 164  
  
- わ -  
割り込みイベント, 96  
割算の余り, 150

### お客様用コメントフォーム

本ドキュメントの出版に関するエラーや変更要求の為に、この用紙をお使い下さい。送り先は：

NS BASIC Corporation  
71 Hill Crescent  
Toronto, Canada M1M 1J3  
fax (416) 264-5888

また同様の出版に関するお問い合わせは、emailでも受けております。

[publications@nsbasic.com](mailto:publications@nsbasic.com)

コメントの主題および、出版日（本ハンドブックの最初の頁に記載）を冒頭にお書き下さい。

頁	コメント