136399

ハンディターミナル

BT シリーズ

BT 開発・運用ツール (BT-H1A)

スクリプトリファレンス

Version 2.33

はじめに

本マニュアルは、BT-W100/W80/W70/1000/1500/600シリーズ<以降<BTシリーズ>>用の端末アプリケーションをカスタマイズする際に使用する、専用のスクリプトについて説明しています。

本マニュアルは、パソコンに関する基本的な知識とプログラミング経験のあるユーザを対象として書かれています。

■対応型式

BT-W300/200シリーズ . . .BT-W300/W300G/W300GM/W300GI/W350/W350G/W350GM/W350GI/W370/W370G/W200/W200G/W200GM/W200GI/W250/W250G/W250GM/W250GI
BT-W100シリーズ. BT-W100/W100G/W100GM/W100GW/W155/W155G/W155GM/W155GW
BT-W80シリーズ BT-W80/W80G/W80GM/W80GW/W85/W85G/W85GM/W85GW/W85T
BT-W70シリーズ BT-W70/W70G/W70GM/W70GW/W75/W75G/W75GM/W75GW
BT-1000シリーズ BT-1010/1010B/1010W/1010WB/1010WBG/1010WBGM/1010WBGW
BT-1500シリーズ BT-1550/1550WB/1550WBG/1550WBGM/1550WBGW
BT-600シリーズ BT-600/600B/600C

■記号の見方

この取扱説明書では、次のような記号を用いて重要な部分がひとめでわかるようにしています。必ずお読みください。

⚠ 危険	**ここに記載されている記載事項を遵守しない場合、結果的に死亡又は重傷を引き起こす危害が発生します。**
⚠ 警告	**ここに記載されている記載事項を遵守しない場合、結果的に死亡又は重傷を引き起こす危害が発生する可能性があります。**
⚠ 注意	**ここに記載されている記載事項を遵守しない場合、中程度の傷害又は軽傷を引き起こす危害が発生する可能性があります。**
注記	**ここに記載されている記載事項を遵守しない場合、商品自体の損害（自損）のみならず、他の財物に対する損傷を引き起こす可能性があります。**

▶ 重要　**かならずおこなう操作などについての注意を示しています。**

ポイント　**誤りやすい操作などについての注意を示しています。**

参考　本文の理解を深める事項や、知っておくと役に立つ情報を示しています。

関連する情報が記載されているページやマニュアルを示しています。

■ご注意

- 本書に記載されている以外の方法でアプリケーションを変更された場合の結果については保証できかねますのでご留意ください。
- 本書の一部または全部を無断で使用、複製することはできません。
- 本書に記載されている情報は、予告なく変更されることがあります。
- Microsoft、Windowsは、米国Microsoft　Corporationの米国およびその他の国における登録商標または商標です。
- 本マニュアルに記載されている会社名、および製品名は、それぞれ各社の登録商標、または商標です。

安全にご使用いただくために

⚠ 警 告	• 人体、および人体の一部を保護する目的でこの商品を使用してはいけません。 • この商品は、防爆エリアで使用することを想定していない商品ですので、防爆エリアでは決して使用しないでください。

⚠ 注 意	機器が万一故障した場合、各種の損害を防止するための十分な安全対策を施してください。

注 記	• 仕様に示された規格以外でのご使用、または改造された製品については、機能および性能の保証はできかねますのでご留意ください。 • 機器を組み合わせてご使用になる場合、使用条件、環境などにより、機能および性能を十分に発揮できない場合がありますので、よくご検討のうえ、ご使用ください。

マニュアルの読みかた

このマニュアルでは、次のような記号を使ってメニューやボタンなどを表しています。

記　号	説　明
【　】	画面上のメニュー名です。
［　］	ウィンドウ、ダイアログボックス、タブ、およびこれらの画面上の項目名です。
〔　〕	ダイアログボックスなどのボタン名です。
「　」	画面上のメッセージまたは本書内の参照先タイトル、および強調の意味を示しています。
『　』	参照先が関連マニュアルの場合のマニュアル名を示します。

マニュアルの構成

1章	スクリプトの概要	スクリプトの概要について説明します。
2章	スクリプトで使用する文字と数値	スクリプトで使用する文字列、数値、変数、配列変数などについて説明しています。
3章	スクリプトで使用する演算子	スクリプトで使用する演算子について説明しています。
4章	処理の制御	処理の分岐、繰り返しなどを説明しています。
5章	スクリプト構成	スクリプトの構成について説明します。
6章	コントロールの基礎知識	BTシリーズの動作を制御するコントロールの基礎知識について説明しています。
7章	コントロール一覧	BTシリーズの動作を制御するコントロールについて説明しています。
	付録	キャラクタコード表、エラー一覧、コントロールリファレンスなどを記載しています。

目　次

はじめに
マニュアルの読みかた …… 2
マニュアルの構成 …… 3
目　次 …… 4

1章　スクリプトの概要

1-1　概要 …… 1-2
　スクリプトとは …… 1-2
　スクリプトの特長 …… 1-2
1-2　スクリプトファイルの構成要素 …… 1-3

2章　スクリプトで使用する文字と数値

2-1　名前 …… 2-2
　名前の付け方 …… 2-2
2-2　変数 …… 2-3
　変数の定義 …… 2-3
　変数に代入できる文字と数値 …… 2-6
2-3　配列変数 …… 2-9
　配列変数の定義 …… 2-9
2-4　変数、配列変数のプロパティとメソッド …… 2-11
　変数のプロパティとメソッド …… 2-11
　配列変数のプロパティとメソッド …… 2-16
　変数のメソッド－戻り値が配列のメソッド …… 2-19

3章　スクリプトで使用する演算子

3-1　計算する …… 3-2
　算術演算子 …… 3-2
3-2　比較する …… 3-4
　数値比較演算子 …… 3-4
　文字列比較演算子 …… 3-5
　論理型比較演算子(is) …… 3-7
3-3　条件を組み合わせる …… 3-8
　否定演算子(not) …… 3-8
　論理和(Or)・論理積(And) …… 3-9
3-4　文字列を連結する …… 3-10
　文字列連結演算子(&) …… 3-10
3-5　演算子の優先順位 …… 3-11
　優先順位 …… 3-11

4章　処理の制御

4-1　処理を分岐する ····· 4-2
If、Then、ElseIf、Else、End If ····· 4-2
Select Case、Case、Case Else、End Select ····· 4-4
分岐条件の入れ子（ネスト） ····· 4-6
4-2　処理を繰り返す ····· 4-7
While、Wend、Wbreak、Wcontinue ····· 4-7
For、Next、Fbreak、Fcontinue ····· 4-9

5章　スクリプト構成

5-1　概要 ····· 5-2
スクリプトの構成 ····· 5-2
GUIのプログラミング ····· 5-4
5-2　メソッド ····· 5-5
メソッドの定義 ····· 5-5
メソッドの呼び出し ····· 5-7
mainメソッド ····· 5-9
メソッドから抜ける ····· 5-10
5-3　パッケージ ····· 5-11
パッケージの定義 ····· 5-11
パッケージへのアクセス ····· 5-12
With、EndWith（プロパティ、メソッドへのアクセス） ····· 5-13
5-4　コントロール ····· 5-14
コントロールへのアクセス ····· 5-14
タグ指定コントロール ····· 5-15
Widget操作コントロール ····· 5-17
イベント対応コントロール ····· 5-20
5-5　名前の参照、スコープ ····· 5-23
定義場所 ····· 5-23
スコープと参照 ····· 5-23
予約語の例外規則 ····· 5-25
5-6　別スクリプトファイルの読み込み ····· 5-26
Include（別スクリプトの読み込み） ····· 5-26
5-7　別プログラムの実行 ····· 5-27
Load（プログラムの実行、起動） ····· 5-27
5-8　実行時例外／エラー ····· 5-28
実行時例外 ····· 5-28
実行時エラー ····· 5-29

5-9 空白文字とコメント ……5-30
空白文字 …… 5-31
単行コメント …… 5-31
複数行コメント …… 5-31

6章 コントロールの基礎知識

6-1 画面表示(BT-W100/W80/W70 シリーズ) …… 6-2
表示のしくみ(GUIプログラム) …… 6-2
文字と図形の表示－タッチパネル表示－ …… 6-5
文字と図形の表示－エミュレータ表示－ …… 6-8
6-2 画面表示(BT-600/1000/1500 シリーズ) ……6-12
表示のしくみ(GUIプログラム) …… 6-12
表示のしくみ(CUIプログラム) …… 6-14
文字と図形の表示(GUIプログラム) …… 6-15
文字と図形の表示(CUIプログラム) …… 6-18
6-3 入力処理(BT-W100/W80/W70 シリーズ) ……6-22
タッチパネル表示での入力処理 …… 6-22
エミュレータ表示での入力処理 …… 6-24
入力確定と取り消し―タッチパネル表示、エミュレータ表示― …… 6-25
キー入力動作―タッチパネル表示、エミュレータ表示― …… 6-25
6-4 入力処理(BT-600/1000/1500 シリーズ) ……6-28
GUIプログラムでの入力処理 …… 6-28
CUIプログラムでの入力処理 …… 6-30
入力確定と取り消し …… 6-31
キー入力動作 …… 6-31
6-5 バーコードリーダ ……6-34
読み取り動作の設定 …… 6-34
バーコード合成読み取り機能(レーザータイプのみ) …… 6-38
6-6 デバイス ……6-39
使用目的 …… 6-39
動作設定 …… 6-39
6-7 マスタファイル ……6-40
マスタファイルの種類 …… 6-40
ファイル構成 …… 6-43
置き換えマスタ …… 6-44
指示マスタ …… 6-46
選択マスタ …… 6-47

6-8　サーバ PC との通信 …… 6-48
リクエストマネージャを使用したシステム …… 6-48
TCP/IPのソケット通信、またはFTPを使用したシステム …… 6-50
データ転送ソフトを使用したシステム …… 6-51
通信ライブラリを使用したシステム …… 6-52
Bluetooth搭載携帯電話を利用した遠隔地通信システム …… 6-54
アナログモデムを利用した遠隔地通信システム …… 6-55
6-9　その他の通信 …… 6-58
プリンタとの通信 …… 6-58
RS-232Cポートを使用した通信 …… 6-59
USBポートを使用した通信 …… 6-59
BTシリーズ間の通信 …… 6-60
6-10　省電力機能 …… 6-61
オートパワーオフ機能 …… 6-61
バックライトのオートオフと明るさ調整 …… 6-61

7章　コントロール一覧

7-1　別スクリプトの呼び出し …… 7-2
System …… 7-2
7-2　Widget 操作 …… 7-5
Widget共通 …… 7-5
Window …… 7-12
Canvas …… 7-16
GroupBox …… 7-20
TextField …… 7-23
TextBox …… 7-34
Label …… 7-41
ListBox …… 7-44
ComboBox …… 7-51
GridBox …… 7-57
CheckBox …… 7-66
RadioButton …… 7-69
Button …… 7-73
ListTable …… 7-78
7-3　イベントの取得 …… 7-81
Event …… 7-81
7-4　データの入力 …… 7-85
InputString …… 7-85
InputDecimal …… 7-95
InputDate …… 7-98
InputByList …… 7-100
7-5　液晶表示部(LCD)の設定 …… 7-103
Screen …… 7-103
LCD …… 7-113
Drawing …… 7-117
Dialog …… 7-128

7-6　読み取り動作の設定 ····· 7-133
JAN ····· 7-133
CODE39 ····· 7-135
EAN128 / CODE128 ····· 7-136
ITF ····· 7-137
NW7 ····· 7-138
CODE93 ····· 7-139
TOF / COOP ····· 7-140
RSS ····· 7-141
QR ····· 7-142
DataMatrix ····· 7-143
PDF417 ····· 7-144
Composite ····· 7-145
BCR ····· 7-146
OCR ····· 7-153
7-7　デバイスの動作設定 ····· 7-157
Led ····· 7-157
Vibration ····· 7-158
Buzzer ····· 7-159
7-8　ファイルの操作 ····· 7-160
FileStream ····· 7-160
File ····· 7-166
FileSystem ····· 7-172
SQLite ····· 7-183
SQLiteCommand ····· 7-184
7-9　マスタファイルとの照合 ····· 7-190
Master ····· 7-190
7-10　ログファイルの操作 ····· 7-203
LogRecord ····· 7-203
7-11　高速検索 ····· 7-211
Search ····· 7-211
7-12　サーバとのデータ送受信 ····· 7-214
RemoteAccess ····· 7-214
RemoteDB ····· 7-217
RemoteFile ····· 7-229
RemoteUD ····· 7-236

7-13 サーバ PC やプリンタとの通信 ····· 7-240
WLAN ····· 7-240
Network ····· 7-244
Serial ····· 7-249
Comm ····· 7-252
Socket ····· 7-260
FTP ····· 7-267
Bluetooth ····· 7-274
PPP ····· 7-282
Printer ····· 7-283
ZGN ····· 7-286
CommRF ····· 7-291
Comm1 ····· 7-298
Comm2 ····· 7-301
BiCom ····· 7-304
7-14 ユーティリティ ····· 7-308
Utility ····· 7-308
UserObj ····· 7-313
7-15 デバイス設定、メニュー表示、キー入力 ····· 7-315
Key ····· 7-315
Timer ····· 7-318
Handy ····· 7-320

付録

1 アスキーコード表 ····· 付 -2
2 予約語一覧 ····· 付 -3
3 変数の内部型 ····· 付 -4
内部型(数値型、文字列型、論理型) ····· 付-4
4 置き換えマスタ ····· 付 -6
5 コンパイルエラー一覧 ····· 付 -10
6 実行時例外一覧 ····· 付 -12
7 索　引 ····· 付 -14

MEMO

1

スクリプトの概要

アプリケーションビルダ クイックカスタマイズ機能で使用するスクリプトの概要について説明します。

1-1 概要 ……1-2
1-2 スクリプトファイルの構成要素 ……1-3

1-1 概要

スクリプトの概要について説明します。

スクリプトとは

本書で説明するスクリプトは、BTシリーズで動作するアプリケーションを作成するためにBTシリーズ専用に開発された言語です。
スクリプトを活用することによって、業務に最適なアプリケーションを作成できます。

スクリプトの特長

●簡単設計
難しいプログラミング言語の知識がなくても、スクリプトの記述ルールに従って手軽にプログラミングできます。

●変数の型宣言が不要
Basic言語のように、数値、文字列などの型に捕らわれず、柔軟なプログラミングができます。

●コマンドで簡単動作制御
BTシリーズのデータ入力、ファイル入出力など、動作制御を簡単におこなうためのコマンドを用意しています。

1-2 スクリプトファイルの構成要素

スクリプトファイルは、「mainメソッド」と「パッケージ」で構成されています。

●「mainメソッド」

1つのスクリプトファイルに、必ず1つ記述されています。
スクリプトプログラムはmainメソッドから始まります。
「mainメソッド」(5-9ページ)

●「パッケージ」

機能別に複数記述されています。
「5-3 パッケージ」(5-11ページ)

参考 「コメント」は、スクリプトファイルの任意の場所に記述できます。

●「ファイルのコメント」

1つのスクリプトファイルに、任意に記述されています。
スクリプトに関する情報の記述です。
「5-9 空白文字とコメント」(5-30ページ)

●「ファイルのインクルード」

1つのスクリプトファイルに、任意に記述されています。
スクリプトに関する情報の記述です。
「5-6 別スクリプトファイルの読み込み」(5-26ページ)

MEMO

2

スクリプトで使用する文字と数値

文字と数値は、変数、配列変数に格納して使用します。
ここでは、文字と数値について、またこれらを格納する変数と配列変数について説明します。

2-1　名前 ……2-2
2-2　変数 ……2-3
2-3　配列変数 ……2-9
2-4　変数、配列変数のプロパティとメソッド …… 2-11

2-1 名前

スクリプトプログラムで扱う変数、配列変数、パッケージ、スクリプトファイルなどには名前があります。これらには任意の名前を設定できます。
コントロールの名前、予約語は、システムで定義されているので、変更できません。
ここでは、名前を付けるときの規則について説明します。

参考
- コントロールについては、「5-4 コントロール」(5-14ページ)を参照してください。
- 予約語については、「付録2 予約語一覧」(付-3ページ)を参照してください。

名前の付け方

■スクリプトファイル名

スクリプトファイル名は、最大31文字まで記述できます(拡張子「scp」を除く)。
ファイル名には全角文字、半角英数字および "_"(アンダーバー)が使用できます。

■スクリプトプログラムの中で定義する名前

スクリプトプログラム中で定義する名前には、以下の種類があります。

- 変数名、配列変数名
- メソッド名
- メソッド内の変数名(メソッド引数、ローカル変数)
- コントロールのタグ名
- パッケージ名
- パッケージ内のメソッド名
- パッケージのメソッド内の変数名(メソッド引数、ローカル変数)

メソッドについては、「5-2 メソッド」(5-5ページ)を参照してください。
コントロールのタグ名については、「タグ指定コントロール」(5-15ページ)を参照してください。
パッケージについては、「5-3 パッケージ」(5-11ページ)を参照してください。

●コントロールのタグ名の定義

コントロールのタグ名は、全角文字または半角英文字で始め、最大255バイト(全角で127文字、半角で255文字)以内です。全角文字、半角英数字と"_"(アンダーバー)が使用できます。

●その他の名前の定義

名前は、全角文字または半角英文字で始め、最大126バイト(全角で63文字、半角で126文字)以内です。全角文字、半角英数字と”_”(アンダーバー)が使用できます。

ポイント

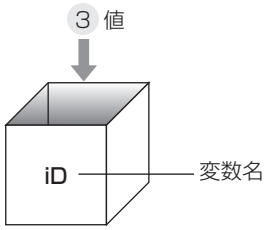
コントロールのタグ名、その他の名前に使用する半角英文字は、次の点に注意してください。

- **先頭文字に半角数字と半角カタカナは使用できません。**
- **大文字と小文字は区別されます。たとえば、"A"と"a"は別の文字として扱われます。**
- **予約語と同じ名前は、大文字・小文字の区別に関係なく使用できません。**
  たとえば、"BEGIN"、"begin"、"RETURN"、"return"などは使用できません。
  予約語については、「付録2 予約語一覧」(付-3ページ)を参照してください。

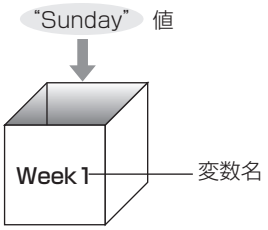
参考 名前の定義場所によっては、スコープ(有効範囲)が異なるため、スクリプトプログラム中に同名が定義できる場合もあります。スコープについては、「5-5 名前の参照、スコープ」(5-23ページ)を参照してください。

2-2 変数

スクリプトプログラムで使用する文字列、数値などは、変数という入れ物に入れて扱います。
変数へ格納された値を参照したり、入れ替えたりすることによって、スクリプトプログラムを簡潔に作成することができます。



変数idに数値3を格納します。



変数Week 1に文字列"Sunday"を格納します。

変数の定義

変数には、数値、文字列、演算結果などの値が代入できます。

▶ 重 要

1つのスクリプトファイルで記述できる変数は、最大1000個(配列変数の要素数は含まない)です。
配列変数については、「2-3 配列変数」(2-9ページ)を参照してください。

■変数名

変数名は、全角文字または半角英文字で始め、最大126バイト(全角で63文字、半角で126文字)以内です。全角文字、半角英数字と”_”(アンダーバー)が使用できます

ポイント

半角英文字は、次の点に注意してください。
- **先頭文字に半角数字と半角カタカナは使用できません。**
- **大文字と小文字は区別されます。たとえば、"A"と"a"は別の文字として扱われます。**
- **予約語と同じ名前は、大文字・小文字の区別に関係なく使用できません。**
  たとえば、"BEGIN"、"begin"、"RETURN"、"return"などは使用できません。
  予約語については、「付録2 予約語一覧」(付-3ページ)を参照してください。

■変数の宣言

変数はメソッド内のMethodとBeginの間に宣言します。

例

```
Method Sample()
      AAA     ……変数
Begin
End Method
```

メソッドについては、「5-2 メソッド」(5-5ページ)を参照してください。

ポイント

変数を宣言するときには、文字列型、数値型などの型の指定は必要ありません。
ただし、システム内部では変数は型を持っています。詳細については、「付録3　変数の内部型」(付-4ページ)を参照してください。

■変数の初期化

変数の宣言時に値を代入することを初期化といいます。

例　AAA＝123

変数を定数として扱うことができます。
変数の前にConstをつけて宣言し、宣言と同時に値を代入します。

Const 変数名＝値

例　Const BBB=0.05
Const CCC="Hello"

▶ 重 要

スクリプトプログラムの中で繰り返し同じ値を参照するときに定数を使用します。
定数は値を変更できませんが、それ以外の扱いは変数と同じです。

■変数の参照・代入

変数を参照するときは、必ず初期化が必要です。初期化しないで変数を参照すると、実行時例外になります。

変数へ数値、文字列、演算結果を代入するときは、その変数を数値として参照するか文字列として参照するかによって、数値または文字列へ変換されます。
変換できないときは、実行時例外となります。

▶ 重 要

次のような変換がおこなわれたときに、実行時例外となります。

- **–2147483647～2147483647の範囲外の文字列が数値変換されたとき。**
- **空白文字など、数値でない文字が含まれる文字列が数値変換されたとき。**

数値変換の例については、 「例1」(2-5ページ)を参照してください。

例1　変数Aに、次の表のようにいろいろな値を代入します。
変数Aを数値として参照するときと、文字列として参照するときの両方の結果を示します。

Aに値を代入	数値として Aを参照	文字列として Aを参照	説明
A=012	12	"12"	数値と文字列の両方で参照できます。
A="012"	12	"012"	数値と文字列の両方で参照できます。
A=0x200	512	"512"	16進数は10進数に変換されて、数値と文字列の両方で参照できます。
A="0x200"	512	"0x200"	16進数の文字列を数値として参照するときは、10進数に変換された値になります。
A=+10	10	"10"	+、−の符号付き数値は、符号のない値で参照されます。
A="+10"	10	"+10"	数値として参照するときは、+、−の符号のない値になります。
A="1.2345"	1.235	"1.2345"	小数点以下4桁を含む値の文字列を数値として参照するときは、小数点以下4桁目で四捨五入されます。
A=".5"	0.5	".5"	小数の文字列を数値として参照するときは、小数点の前に0が付加されます。
A=ABC	変数ABCと同じ値	変数ABCと同じ値	変数を代入します。 変数ABCが定義されていない、または初期化されていないと、実行時例外になります。
BB="0123" A=BB	123	"0123"	Aは変数BBと全く同じ状態を引き継ぎます。

以下のような例は、数値変換されるときに実行時例外となります。

Aに値を代入	数値として Aを参照	文字列として Aを参照	説明
A="ABC"	実行時例外	"ABC"	"ABC"は、数値変換できません。
A="2147483648"	実行時例外	"2147483648"	2147483647の値を超える文字列は、数値変換できません。
A="1.5E+3"	実行時例外	"1.5E+3"	指数表現の値の文字列は、数値変換できません。
A="400L"	実行時例外	"400L"	"L"のような数値以外の不正な文字が含まれるので、数値変換できません。
A="50%"	実行時例外	"50%"	"%"のような数値以外の不正な文字が含まれるので、数値変換できません。
A="(20)"	実行時例外	"(20)"	"()"のような数値以外の不正な文字が含まれるので、数値変換できません。
A="2003/9/10"	実行時例外	"2003/9/10"	日付は数値変換できません。
A="百二十三"	実行時例外	"百二十三"	漢数字は数値変換できません。
A="1,000,000"	実行時例外	"1,000,000"	値に桁区切りのカンマがあると、数値変換できません。

以下のような例は、コンパイルエラーとなります。

Aに値を代入	数値として Aを参照	文字列として Aを参照	説明
A=50%	コンパイルエラー	コンパイルエラー	%は剰余算の演算子です。 続く式によっては、コンパイルエラーとならないことがあります。
A=1,000,000	コンパイルエラー	コンパイルエラー	値に桁区切りのカンマは不要です。

変数に代入できる文字と数値

変数に代入できる値には、文字列、数値、論理値の3種類あります。

文字列 :「"Script"」、「"棚卸し"」のような半角、全角文字で記述します。
　　また、「¥x41¥x42¥x43」のように、バイナリデータで記述できます。
数値　:「1」、「2.0」のような整数、小数で記述します。
論理値 :「true」(真)、「false」(偽)、「nil」(不定値)があります。
　　(システム内部では、falseはnilと同じ不定値です)。

■文字列

""で囲みます。
""の中には、数字、記号、文字(半角、全角)などの文字列が記述できます。

例

"ABC"
"123&ABC"
"バーコード"

▶ 重 要

半角で最大8192文字(全角で4096文字)まで記述できます。

●バイナリ表記について

文字列は、次のようにアスキーコードを使用してバイナリデータで記述できます。
文字列の最初に"¥x"を記述します。

例

"¥x41¥x42¥x43". . . . . .文字列表記では"ABC"
"¥x31¥x32¥x33¥x26¥x41¥x42¥x43" . . . . . .文字列表記では"123&ABC"

●制御文字について

下記の制御文字を使用できます。
""の中に、制御文字を記述します。制御文字の長さは半角1文字となります。

制御文字	意味
¥"	ダブルクォーテーション
¥¥	円マーク
¥n	LF(0x0A) ラインフィード
¥r	CR(0x0D) キャリッジリターン
¥t	タブ
¥e	ESC(0x1B) エスケープ
¥x	以降のデータが、バイナリ表記

ポイント

上記以外の制御文字を記述すると、コンパイルエラーとなります。

■数値

●正負の整数(10進)

例　0
123456

符号桁として、「+」「−」が記述できます。

例　+512
−512

▶ 重 要　**使用できる範囲は、「−2147483647〜2147483647」になります。**

●正負の小数(10進)

例　15.5
−15.5

演算の結果は、小数第3位まで保証されます(小数点以下4桁目を四捨五入)。

▶ 重 要　**小数点以下を含む場合、使用できる範囲は、「-2147483.647〜2147483.647」になります。小数点以下の桁数に関係なく、使用できる範囲は変わりません。**

参 考　小数は、システム内部では倍精度浮動小数を使用しているため、誤差が生じる可能性があります。

●16進数

先頭に0xをつけた数値

例　0xa

参 考　0と正の整数を、16進で指定することができます。
負の整数、小数は指定できません。

▶ 重 要　**指数、8進数は、記述できません。**

■論理値

論理値とは、たとえば「コマンドが正常終了した」に対し、処理の結果が正しい(true)か、正しくない(false)かを判断させるときなどに使用します。

論理値には以下の3つがあります。

- true
- false
- nil

trueは真、falseは偽、nilは不定値(初期化されていない値)を意味します。
上記以外の記述はできません。
システム内部では、falseはnilと同じ不定値です。
nilは、プロパティに初期値を代入するときに使用できます。

例　Led:color=nil

プロパティについては、「5-4 コントロール」(5-14ページ)を参照してください。

論理値を変数に代入したときは、以下のようになります。

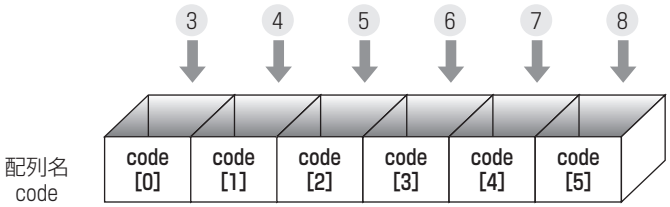
Aに値を代入	数値としてAを参照	文字列としてAを参照	説明
A=true	実行時例外	実行時例外	trueは、数値、文字列へ変換できません。
A=false	実行時例外	実行時例外	falseは、数値、文字列へ変換できません。
A=nil	実行時例外	実行時例外	nilは、数値、文字列へ変換できません。

2-3 配列変数

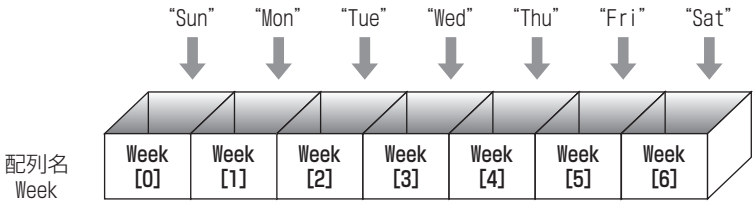
配列変数は、複数の値をまとめて格納できる入れ物です。変数と同じように、参照、代入ができます。
代入できる文字と数字については、「変数に代入できる文字と数値」(2-6ページ)を参照してください。

例

- 配列変数codeに3, 4, 5, 6, 7, 8を格納します。



- 配列変数Weekに"Sun","Mon","Tue","Wed","Thu","Fri","Sat"を格納します。



配列変数の定義

配列変数には、数値、文字列、演算結果などの値が代入できます。

▶ 重要

1つのスクリプトファイルに記述できる配列変数は、変数の数と合わせて、最大1000個です。

■変数名

変数名は、全角文字または半角英文字で始め、最大126バイト(全角で63文字、半角で126文字)以内です。全角文字、半角英数字と"_"(アンダーバー)が使用できます。

ポイント

- **変数名と同名は、定義できません。**
- **半角英文字は、次の点に注意してください。**
- **先頭文字に半角数字と半角カタカナは使用できません。**
- **大文字と小文字は区別されます。たとえば、"A"と"a"は別の文字として扱われます。**
- **予約語と同じ名前は、大文字・小文字の区別に関係なく使用できません。**
  たとえば、"BEGIN"、"begin"、"RETURN"、"return"などは使用できません。
  予約語については、「付録2 予約語一覧」(付-3ページ)を参照してください。

■配列変数の宣言

配列変数は、メソッド内のMethodとBeginの間に宣言します。
メソッドについては、「5-2 メソッド」(5-5ページ)を参照してください。

例
```
Method Sample()
        AAA[5]·····配列変数
Begin
End Method
```

ポイント

- **[ ](大括弧)の中は、要素数を数値で指定します。変数は使えません。**

  例 **aaa[5]**

- **配列変数を宣言するときには、文字列型、数値型などの型の指定は必要ありません。ただし、システム内部では変数は型を持っています。詳細については、「付録3 変数の内部型」(付-4ページ)を参照してください。**

■配列変数の初期化

配列変数は、{ }(中括弧)と値(数値、文字列、論理値)を用いて初期化できます。値を「,」(カンマ)で区切り、複数個連結します。

例
```
aaa[5]={1,2,3,4,5}
array[3]={1,"001",true}
```

- 初期化付きの宣言は、要素数を省略することができます。

  例
  ```
  aaa[]={1,2,3,"001","002","003"}
  配列要素数6が省略されています。
  ```

- 要素数より初期化要素が少ないときは、要素をnilで初期化します。

  例
  ```
  aaa[5]={1,2,3}
  aaa[3]とaaa[4]にnilが代入されます。
  ```

- 配列変数を定数として扱うことができます。
  変数の前にConstをつけて宣言し、宣言と同時に値を代入します。
  Const 変数名[ ]={値}

  例 Const BBB[2]={0.03,0.05}

■配列変数の参照・代入

配列変数の参照、代入については、変数の扱いと同様です。
「変数の参照・代入」(2-4ページ)

2-4 変数、配列変数のプロパティとメソッド

変数と配列変数には、プロパティとメソッドがあります。
プロパティは、数値、文字列などの変数に格納されているデータの型、データ長などの属性をあらわします。
また、メソッドを使用して、変数、配列変数に含まれる文字列の検索、削除などができます。

参考　配列変数の要素1つ1つは、変数のプロパティとメソッドが使用できます。

変数のプロパティとメソッド

■プロパティ

変数を参照、代入するには、「変数名.プロパティ名」と記述します。

名称	説明	範囲	代入
length	文字列の長さ※ 変数が数値、文字列のとき、文字列の長さ。	0以上の整数	×
isDigit	数値変換できるかの判定 数値に変換できるかどうかを判定します。	true：数値変換できる false：数値変換できない	×
isBoolean	論理型の判定 true/false/nilかどうかを判定します。	true：論理型 false：論理型でない	×
isString	文字列の判定 文字列型かどうかの判定をします。	true：文字型 false：文字型でない	×
isInt	数値が整数かの判定 整数かどうかを判定します。	true：整数 false：整数でない	×
toInt	数値の整数部参照 整数、小数の整数部を参照します。小数点以下は切り捨てられます。	整数値	×

※ 漢字やひらがななどの2バイト文字は2と数えます。

■メソッド

変数を参照、代入するには、「変数名.メソッド名(メソッド引数・・・)」と記述します。

名称	説明	引数:意味(範囲)	戻り値
Left	文字列の抽出(先頭から)	nCount:抽出文字列長 (1～変数のlength)※1	文字列(抽出文字列)※2
Right	文字列の抽出(最後から)	nCount:抽出文字列長 (1～変数のlength)※1	文字列(抽出文字列)※2
Mid	文字列の抽出 (指定位置から)	nFirst:抽出位置 (0～変数のlength-1)※3 nCount:抽出文字列長 (1～変数のlength)※1	文字列(抽出文字列)※2
Find	文字列の検索 (指定位置から最後へ検索)	strSub:検索文字列 nFirst:検索開始位置 (0～変数のlength-1)※3	0以上の整数値(検索文字列の先頭位置) nil(見つからないとき)
Rfind	文字列の検索 (指定位置から先頭へ検索)	strSub:検索文字列 nFirst:検索開始位置 (0～変数のlength-1)※3	0以上の整数値(検索文字列の最後位置) nil(見つからないとき)
Remove	指定された文字を削除	cSub:削除する文字列	文字列(cSubを除いた文字列)
Hex	文字列を16進数に変換	caps:true/false (16進変換文字列の英字を大文字変換する／小文字変換する)	16進文字列(変換文字列)
Repeat	文字列の繰り返し	count:繰り返し回数(1～)	文字列(繰り返し処理された文字列)

※1 漢字やひらがなどの2バイト文字は2と数えます。
※2 戻り値の最初や最後の文字が2バイト文字の途中のとき、不正な文字が返ることがあります。
※3 文字列の1文字目の位置は「0」、最後の位置は「length-1」を指定します。

●Hex

変数が16進文字列のときにも、英字の大文字、小文字変換ができます。
変換方法は、以下のようになります。

1 変数を数値として評価します。
数値変換できなかったときは、実行時例外となります。

2 数値変換した値が0と正の数のとき、それを16進数に変換した文字列を戻り値として返します。
数値変換した値が負の数や小数のときは、実行時例外となります。

●Repeat

以下のような使い方をしたときは、実行時例外となります。

- 繰り返し回数がマイナス値や小数など不正なとき
- 繰り返し処理後の文字列長が8192バイト(半角8192文字、全角4096文字)を超えるとき

●各プロパティの参照例

【条件】

var1="ABCDEFGHIJ"、var2="122345"、var3=12345、var4=true

【結果】

プロパティ	結果	プロパティ	結果	プロパティ	結果
var1.length	10	var2.isDigit	true	var3.isDigit	true
var1.isDigit	false	var2.isBoolean	false	var3.isBoolean	false
var1.isBoolean	false	var2.isString	true	var3.isString	false
var1.isString	true	var2.isInt	true	var3.isInt	true
var1.isInt	false	var3.length	5	var4.isBoolean	true

●各メソッド実行例

【条件】

var="ABCDEFCD12"

【結果】

メソッド	結果	解説
var.Left(5)	"ABCDE"	−
var.Left(15)	"ABCDEFCD12"	「抽出文字列長」が変数の「length」以上のときは、変数と同じ値が返ります。
var.Right(5)	"FCD12"	−
var.Right(15)	"ABCDEFCD12"	「抽出文字列長」が変数の「length」以上のときは、変数と同じ値が返ります。
var.Mid(2,5)	"CDEFC"	−
var.Mid(2,20)	"CDEFCD12"	「抽出開始位置」が変数の「length」-「抽出開始位置」以上のときは、「抽出開始位置〜文字列最後」間での値が返ります。
var.Mid(15,5)	""(文字列長0)	「抽出開始位置」が変数の「length」より長いときは、""(空文字列)が返ります。
var.Find("CD",0)	2	−
var.Find("CD",15)	nil	「検索開始位置」が変数の「length」以上のときは、nilが返ります。
var.Find("",0)	nil	「検索する文字列」が""(空文字列)のときは、nilが返ります。
var.Find("ABCDEFCD1234",0)	nil	−
var.Rfind("CD",9)	7	−
var.Rfind("CD",15)	7	「検索開始位置」が変数の「length」以上のときは、変数の「length」-1とみなします。
var.Rfind("CD",-1)	nil	「検索開始位置」が0より小さいときは、nilが返ります。
var.Rfind("",9)	nil	「検索する文字列」が""(空文字列)のときは、nilが返ります。
var.Rfind("ABCDEFCD123",9)	nil	−
var.Remove("CD")	"ABEF12"	−
var.Remove("XYZ")	"ABCDEFCD12"	「検索文字列」が見つからないときは、元の文字列が返ります。
var.Remove("")	"ABCDEFCD12"	−
var.Repeat(2)	"ABCDEFCD12ABCDEFCD12"	−

●メソッドの使用例

例1 「Find」メソッドの使用例
文字列strの先頭から"pattern"を検索し、見つかったら配列aの先頭から順に発見位置を格納します。発見できなくなったら、検索処理を終了します。

```
cnt=0
idx=0                          //Findメソッドの引数にする変数(検索開始位置)
ptn="pattern"                  //Findメソッドの引数にする変数(検索文字)
a[100]
Begin

While true
  idx=str.Find(ptn,idx)        //Findメソッド
  If idx is nil Then
    Wbreak
  Endif
  a[cnt]=idx
  cnt=cnt+1
  idx=idx+ptn.length
Wend
```

例2 「Rfind」メソッドの使用例
文字列strの最後から"pattern"を検索し、見つかったら配列bの先頭から順に発見位置を格納します。発見できなくなったら、検索処理を終了します。

```
cnt=0
idx=str.length-1               //Rfindメソッドの引数にする変数(検索開始位置)
ptn="pattern"                  //Rfindメソッドの引数にする変数(検索文字)
b[100]
Begin

While true
  idx=str.Rfind(ptn,idx)       //Rfindメソッド
  If idx is nil Then
    Wbreak
  End If
  idx=idx-ptn.length
  b[cnt]=idx+1
  cnt=cnt+1
Wend
```

例3 「Hex」メソッドの使用例

```
aaa="13"
bbb
Begin

bbb=aaa.Hex(true)                    //bbbの値は"0xD"
bbb=bbb.Hex(false)                   //bbbの値は"0xd"
```

例4 「Hex」メソッドで、数値変換した値が小数のため、実行時例外となる例

```
aaa="0.5"
bbb=aaa.Hex(false)                   //ここで実行時例外が発生します。
```

例5 「Repeat」メソッドの使用例

```
aaa="ABC"
bbb
Begin

bbb=aaa.Repeat(3)                    //bbbの値は"ABCABCABC"
```

配列変数のプロパティとメソッド

■プロパティ

配列変数を参照、代入するには、「配列変数名.プロパティ名」と記述します。

名称	説明	範囲	代入
size	配列の要素数	1～8192	×

例　ary[5]={1,2,3,4,5}のとき、ary.sizeは5となります。

■メソッド

配列変数を参照、代入するには、「配列変数名.メソッド名(メソッド引数･･･)」と記述します。

名称	説明	引数:意味(範囲)	戻り値
Merge	配列の各要素を文字列とみなし、セパレータで結合した結果を返します。	strSeparator:セパレータ文字列	変数(連結文字列)
Pack	配列の各要素を文字コードとみなし、文字列に合成します。	なし	変数(文字列)
PackBin	配列の各要素をバイナリデータとみなし、バイナリデータに合成します。	なし	変数(文字列)

●Merge

配列の要素に論理値の「false」、「nil」が含まれるときは、その要素を文字列長0の文字として連結します。

▶ 重 要

次のような文字が含まれているときは、実行時例外となります。

- **「true」が含まれている**
- **文字列変換できない値("でくくられていない)が含まれている**

また、連結した文字列が最大文字列長8192バイト(半角8192文字、または全角4096文字)を超えるときも、実行時例外となります。

●Pack

- 配列の最後の要素には終端記号として、数値0(0x00)を必ず記述します。
- 記述できる文字コードの範囲は、1～255の整数値です。

例　ary[　]={48,49,50,0}　　･････"0","1","2"の文字コード
ary[　]={0x30,0x31,0x32,0x00}　　･････"0","1","2"の文字コード

▶ 重 要

次のような記述をしたときには、実行時例外となります。

- **文字コードの範囲が1～255以外のとき**
- **終端記号がないとき**

●PackBin

- 記述できる文字コードの範囲は、"1"～"255"です。

例　ary[　]={"0x30","0x31","0x32"}　　・・・・・"0","1","2"の文字コード

▶ 重 要

次のような記述をしたときは、以下の動きになります。

- **文字コードの範囲が"1"～"255"以外のときは、先頭から範囲外の文字コードが現れるまで処理します。**
- **終端記号がないときは、自動的に終端記号を付加します。**

●メソッドの使用例

例1　「Merge」の例

セパレータとして"//"を指定し、次の2つの配列変数をMergeします。
配列変数ary1[5]="あ","い","う","え","お"のとき、変数str1に"あ//い//う//え//お"が格納されます。
配列変数ary2[5]={"あ","い","","え"}のとき、変数str2に"あ//い////え"が格納されます。

```
Method func1()
  ary1[5]={"あ","い","う","え","お"}                    //Mergeに使用する配列変数
  ary2[5]={"あ","い","","え"}                          //Mergeに使用する配列変数
  str1
  str2
  Begin
    str1=ary1.Merge("//")                              //ary1をMergeする
    str2=ary2.Merge("//")                              //ary2をMergeする
End Method
```

例2　「Pack」の例

配列変数ary[　]={0x30,0x31,0x32,0x41,0x42,0x43,0}は、Packすると変数strに"012ABC"が格納されます。

```
Method func2()
  ary[7]={0x30,0x31,0x32,0x41,0x42,0x43,0}      //Packに使用する配列変数
  str
  Begin
    str=ary.Pack                                 //aryをPackする
End Method
```

例3　「PackBin」の例
配列変数ary[ ]={"0x30","0x31","0x32","0x41","0x42","0x43"}は、PackBinすると変数binに"012ABC"が格納されます。

```
Method  func3()
  ary[6]={"0x30","0x31","0x32","0x41","0x42","0x43"}
  bin=""
  Begin
  bin=ary.PackBin                    //"012ABC"がバイナリデータとして設定されます。
End Method
```

変数のメソッド－戻り値が配列のメソッド

文字列を分割、分解する変数のメソッドです。
変数のメソッドの戻り値に配列変数を返します。

■メソッド

名称	説明	引数:意味(範囲)	戻り値
split	指定するセパレータで文字列を分割し、配列変数に格納します。	セパレータ	配列変数
Unpack	文字列を文字コードの数値として分解し、配列変数に格納します。	なし	配列変数
UnpackBin	文字列をバイナリコードの数値として分解し、配列変数に格納します。	なし	配列変数

●split

- 戻り値の配列の要素数は、文字列長以上を宣言します。少ないと実行時例外となります。
- 文字列にセパレータが連続するときは、空白文字列として分解されます。
  nilには変換されません。

●Unpack

- 文字列の終端記号として、数値0が必ず付加されます。
- 変換される文字コードの範囲は、1 ～ 255 の整数値です。それ以外の文字のときは、実行時例外となります。
- 戻り値の配列の要素数は、「文字列長＋1」以上を宣言します。少ないと実行時例外となります。

参考 Unpackは、式の途中の文字列を処理できますが、Packは、配列変数のみ処理します。

コンパイルエラーとなる例	正しい記述
–	配列変数=配列変数.Pack.Unpack
文字列=文字列.Unpack.Pack	配列変数=文字列.Unpack 文字列=配列変数.Pack

●UnpackBin

- 文字列の終端記号として、"¥x00"が必ず付加されます。
- 変換される文字コードの範囲は、"¥x00"～"¥xFF"です。それ以外のときは、実行時例外となります。
- 戻り値の配列の要素数は、「文字列長+1」以上を宣言します。少ないと実行時例外となります。

参考 UnpackBinは、式の途中の文字列を処理できますが、PackBinは、配列変数のみ処理します。

コンパイルエラーとなる例	正しい記述
–	配列変数=配列変数.PackBin.UnpackBin
文字列=文字列.UnpackBin.PackBin	配列変数=文字列.UnpackBin 文字列＝配列変数.PackBin

●メソッドの使用例

例1 「split」の例

"あ//い//う//え//お"をセパレータ"//"で分割し、配列変数aryに格納します。
結果は、以下のようにaryに格納されます。
ary[0]:"あ"
ary[1]:"い"
ary[2]:"う"
ary[3]:"え"
ary[4]:"お"
ary[5]～ary[9]は代入されないのでnilのままです。

```
Method  func1()
  ary[10]
  Begin
    ary="あ//い//う//え//お".split("//")
End Method
```

例2 「Unpack」の例

"ABC"を文字コードの数値として分割し、配列変数aryに格納します。
結果は、以下のようにaryに格納されます。
ary[0]:0x41　　·····"A"の文字コード
ary[1]:0x42　　·····"B"の文字コード
ary[2]:0x43　　·····"C"の文字コード
ary[3]:0
ary[4]～ary[9]は、nilのままです。

```
Method  func2()
  ary[10]
  Begin
    ary="ABC".Unpack
End Method
```

例3 「UnpackBin」の例

"ABC"を文字コードの数値として分割し、配列変数aryに格納します。
結果は、以下のようにaryに格納されます。
ary[0]:"65"　　·····"A"の文字コード
ary[1]:"66"　　·····"B"の文字コード
ary[2]:"67"　　·····"C"の文字コード
ary[3]:"0"
ary[4]～ary[9]は、nilのままです。

```
Method  func3()
  ary[10]
  Begin
    ary="ABC".UnpackBin                    // {"65","66","67","0"} が設定されます。
End Method
```

3

スクリプトで使用する演算子

演算子を使用して、数値や文字列の計算、比較などをおこないます。
ここでは、スクリプトで使用する演算子について説明します。

3-1　計算する ……3-2
3-2　比較する ……3-4
3-3　条件を組み合わせる ……3-8
3-4　文字列を連結する …… 3-10
3-5　演算子の優先順位 …… 3-11

3-1 計算する

式は、スクリプトプログラムを構成する最小単位です。たとえば「1 +1」、「ID>1」というように、変数、演算子などを使用して演算式、条件式を記述します。
ここでは、足し算、引き算、掛け算、割り算などの計算に使用する演算子について説明します。

算術演算子

算術演算する演算子です。算術演算式を記述するときに使用します。

■書式

演算子	意味
+	加算(足し算)
-	減算(引き算)
*	乗算(掛け算)
/	除算(割り算)
%またはMOD	剰余算(除算の余を求める)

- 各演算子の左辺と右辺には、数値、数値の変数を使用して演算式を記述します。
  配列変数は記述できません。

  例　A=B+1

  ▶ 重 要

  変数に文字列を代入しても算術がおこなわれます。ただし、変数が演算式で評価されるときに数値変換できない場合は、実行時例外となります。
  文字列の数値変換については、「付録3 変数の内部型」(付-4ページ)を参照してください。

- 除算の結果は、小数点を含む実数で得られます。
  小数点以下3位まで有効(小数点以下4桁目を四捨五入)です。

  参 考　整数の結果を得るには、変数の「toInt」プロパティを使って、数値の整数部だけを参照します。
  「2-4 変数、配列変数のプロパティとメソッド」(2-11ページ)

- 剰余算する変数が小数のときは、小数点以下は切り捨てられて演算がおこなわれます。

例 演算の例を示します。
変数には、下記の値を設定します。
a="5"、b="1.5"、c="9.5"

演算式	結果
a+b	6.5
a-b	3.5
a*b	7.5
a/b	3.333
b%c	1 (1.5%9.5→1%9) 1.5が1に、9.5が9に変換されたあとに、剰余算がおこなわれます
c%a	4 (9.5%5→9%5) 9.5が9に変換されたあとに、剰余算がおこなわれます

3-2 比較する

数値や論理値を比較するための演算子について説明します。

数値比較演算子

数値を比較する演算子です。数値比較の条件式を記述するときに使用します。

■書式

演算子	意味
>	左辺の値が右辺より大きい
>=	左辺の値が右辺より大きいか等しい
==	左辺と右辺の値が等しい
<=	左辺の値が右辺より小さいか等しい
<	左辺の値が右辺より小さい
<>	左辺と右辺の値が等しくない

- 各演算子の左辺と右辺には、数値、数値の変数、演算式を記述します。
  配列変数は、記述できません。

例 A > B*10

▶ 重 要

変数に文字列を代入しても比較できます。ただし、変数が条件式で評価されるときに数値変換できない場合は、実行時例外となります。
文字列の数値変換については、「付録3 変数の内部型」(付-4ページ)を参照してください。

例 a=10.5 b="5"のとき
a>b ・・・・・・・・・・・aが10.5、bが5のとき、aはbより大きい

文字列比較演算子

文字列を比較する演算子です。文字列比較の条件式を記述するときに使用します。

▶ 重 要

- **文字列の先頭から、1バイト単位で比較されます。**
- **日本語やワイルドカードを含む文字列の比較も可能です。**
- **文字列の大小は、文字コードの順序で比較されます。**

■書式

演算子	意味
gt	左辺の値が右辺より大きい
ge	左辺の値が右辺より大きいか等しい
eq	左辺と右辺の値が等しい
le	左辺の値が右辺より小さいか等しい
lt	左辺の値が右辺より小さい
ne	左辺と右辺の値が等しくない
weq	左辺と右辺の値が等しい(ワイルドカード使用)

- 演算子の左辺と右辺には、文字列、文字列の変数を記述します。
  配列変数を記述することはできません。

  例　A eq "0001"

- ワイルドカードは、以下のように記述できます。

  ！：　複数の文字列にマッチします。文字列の中に1つだけ記述できます。

  例　"!12"　‥‥‥12で終わるすべての文字列("12"も含む)

  ？：　1個の文字列にマッチします。文字列の中に複数記述できます。

  例　"?12"　‥‥‥12で終わる3文字の文字列

  [ ]：　範囲を指定します。文字列の中に1つだけ記述できます。

  例　[c-f]　‥‥‥c、d、e、fのどれか
  [0-9]　‥‥‥0、1、2、3、4、5、6、7、8、9のどれか

●文字列比較演算子の使用例

条件式	意味
"a" lt "b"	文字列aは文字列bより小さいか
"012" lt "3"	文字列012は文字列3より小さいか※1
"012" ne "12"	文字列012と文字列12は等しくないか
"012" weq "!"	文字列012は0個以上の文字列か
"012" weq "!12"	文字列012は12で終わる文字列か
"012" weq "01!2"	文字列012は01で始まり2で終わる文字列か
"012" weq "?12"	文字列012は12で終わる3文字の文字列か
"012" weq "[0-9]12"	文字列012は0～9で始まり、12で終わる3文字の文字列か
"012" weq "[0-9]!"	文字列012は0～9で始まる文字列か

※1　文字列比較は、数値の比較の結果と一致しないことがあります。上記の例は、文字列比較では012は3より小は正しいが、数値比較では012は3より小は正しくありません。

論理型比較演算子(is)

演算子の左辺と右辺を論理型で評価して、比較する演算子です。
論理値を比較する条件式が記述できます。

■書式

書式	演算式	意味
1	変数　is　true	変数を論理型で評価した値がtrueかどうか
	変数　is　false	変数を論理型で評価した値がfalseかどうか
	変数　is　nil	変数を論理型で評価した値がnilかどうか
2	変数　is　変数	左辺の変数と右辺の変数を論理型で評価した値が一致するかどうか
3	変数　is　式	左辺の変数と右辺の式の結果を論理型に評価した値が一致するかどうか

●比較結果

左辺の値と右辺の値を論理型で評価して比較し、両方の値が一致すればtrueを、不一致ならばfalseを返します。
論理型の詳細については、「付録3 変数の内部型」(付-4ページ)を参照してください。

- falseとnilは、システム内部では同じ値のため、比較結果は一致します。
- 数値、文字列の変数は、trueと評価されて、比較されます。
- 初期化していない変数は、nilと評価されて、比較されます。

●論理型比較演算子の使用例

- 書式1は、下記のようなときに使用できます。

  例　メソッド引数のチェック(引数として渡された変数が初期化されているかのチェック)
  A is nil

  メソッドについては、「5-2 メソッド」(5-5ページ)を参照してください。

- 書式2は、下記のように使用できます。

  例　A is B

- 書式3は、下記のようなときに使用できます。

  例　メソッドの戻り値をチェック
  A is TestMethod()

  メソッドについては、「5-2 メソッド」(5-5ページ)を参照してください。

3-3 条件を組み合わせる

演算結果や条件式を否定する演算子(not)と、条件式を組み合わせる演算子の論理和(Or)・論理積(And)について説明します。

否定演算子(not)

条件式の値を論理否定します。

■書式

not　条件式

演算子の右には、条件式や演算式を記述します。

例　not (value is nil) ・・・・・・・変数valueがnilでない
not (value eq "A")・・・・・・・変数valueが"A"でない

論理和(Or)・論理積(And)

2つの条件式の論理和、論理積を求めます。

■書式

- **論理和**
  条件式1　Or　条件式2

- **論理積**
  条件式1　And　条件式2

●評価結果

まず条件1と条件2をそれぞれ評価し、さらにそれらの論理和、論理積を求めて、次のような結果を返します。

演算子	条件式1の評価	条件式2の評価	結果
And	true	true	true
	true	false	false
	false	true	false
	false	false	false
Or	true	true	true
	true	false	true
	false	true	true
	false	false	false

例　(Code>=10) Or (Code<=0)·····Codeが10以上、または0以下
(Code>=1) And (Code<=9)·····Codeが1以上、かつ9以下

3-4 文字列を連結する

文字列の連結に使用する演算子について説明します。

文字列連結演算子(&)

文字列の変数を連結します。

▶ 重 要

連結したとき、最大文字数(8192バイト)を超えると、実行時例外になります。

■書式

&

文字列と文字列の間に&(アンパサンド)を記述します。

例

A="000"　　B="123"のとき

C=A & B·····Cは"000123"となります。

3-5 演算子の優先順位

ステートメントに複数の演算子を記述すると、優先順位の高い順に実行されます。
実行される優先順位を高くするには、括弧を使います。

優先順位

同一順位のものはステートメントの左から順に評価されます。
括弧を使用すると、ステートメントに記述されている評価の優先順位を高くします。

優先順位	演算子
1	:(コロン-プロパティやメソッド呼び出し)※1　[ ](配列要素へのアクセス)　括弧(x)
2	単項+　単項–※2
3	.(ピリオド–変数プロパティやメソッドの呼び出し)
4	＊　/　%　Mod
5	+　–※3
6	&
7	==　<>　>　>=　<　<=　eq　ne　gt　ge　lt　le　weq
8	not
9	is
10	And
11	Or

※1　メソッド呼び出しについては、「メソッドの呼び出し」(5-7ページ)を参照してください。
※2　「単項+」、「単項–」は、「+3」や「–5」の符号を指します。
※3　「+」、「–」は、加算、減算を指します。

参考　下記の演算子は、同一ステートメントに記述しないことをおすすめします。
記述する場合は、()で優先順位を明確にしてください。

- 数値演算と文字列演算
- 数値比較と文字列比較

MEMO

4

処理の制御

処理の流れの制御には、処理の分岐や処理の繰り返しがあります。
ここでは、処理の流れの制御について説明します。

4-1　処理を分岐する ……4-2
4-2　処理を繰り返す ……4-7

4-1 処理を分岐する

処理を分岐するIf文、Select文について説明します。
処理とは、1つまたは複数のステートメント(文)のことです。

▶ 重 要

If文、Select文、While文、For文を組み合わせて、入れ子にできるのは、最大20階層までになります。
入れ子については、「分岐条件の入れ子(ネスト)」(4-6ページ)を参照してください。

If、Then、ElseIf、Else、End If

条件を指定して、その評価結果によって処理を分岐します。

■書式

- **分岐が1つの場合**

```
If      条件1  Then
        処理1
End If
```

「条件1」を満たすとき「処理1」を実行します。満たさないときは何も処理をしません。

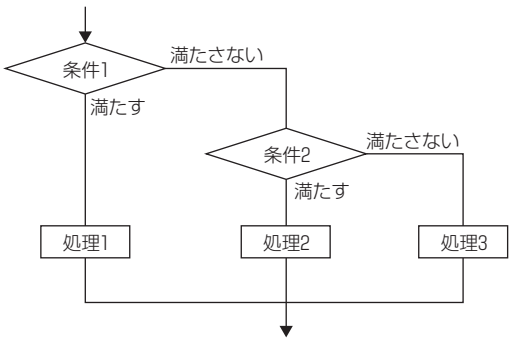
- **分岐が2つの場合(「Else」を使う)**

```
If      条件1  Then
        処理1
Else
        処理2
End If
```

「条件1」を満たすとき「処理1」を、満たさないときは「処理2」を実行します。

- **分岐が3つ以上の場合(「ElseIf」を使う)**

```
If      条件1   Then
        処理1
ElseIf  条件2   Then
        処理2
Else
        処理3
End If
```



参 考 「ElseIf 条件 Then 処理」は、複数記述することができます。

「条件1」を満たすとき「処理1」を、「条件1」は満たさないが「条件2」を満たすとき「処理2」を、「条件1」も「条件2」も満たさないときは「処理3」を実行します。

●条件式について

「If」、「ElseIf」の「条件」には、下記のような条件式を記述します。

数値比較の例

If (i < 10)　　·····iが10より小さいとき

「<」については、 「数値比較演算子」(3-4ページ)を参照してください。

文字列比較の例

If (Str eq "A")　·····Strが"A"のとき

「eq」については、「文字列比較演算子」(3-5ページ)を参照してください。

論理型比較の例

If (i is true)　·····iがtrueのとき

「is」については、「論理型比較演算子(is)」(3-7ページ)を参照してください。

参考

- 条件式の組み合わせ式も使用できます。
  「3-3 条件を組み合わせる」(3-8ページ)
- 処理には、複数のステートメント(文)が記述できます。

例1　typeが10未満のとき、termIDに1を代入します。

```
If type<10 Then
  termID=1
End If
```

例2　typeが10未満のとき、termIDに10を代入し、codeIDに1を代入します。
typeが10以上20未満のとき、termIDに20を代入し、codeIDに2を代入します。
typeが20以上のとき、termIDに30を代入し、codeIDに3を代入します。

```
If type<10 Then
  termID=10 codeID=1
ElseIf type<20 Then
  termID=20 codeID=2
Else
  termID=30 codeID=3
End If
```

Select Case、Case、Case Else、End Select

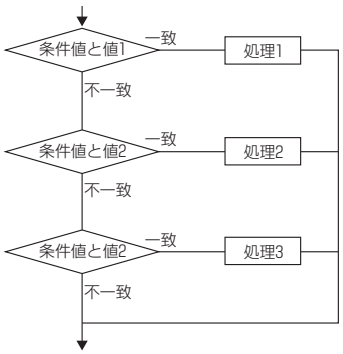
条件式(条件値)を定数(値)と比較して、処理を分岐します。

参 考　If文の分岐で「ElseIf Then」を複数記述するかわりに、Select文が使用できます。
3つ以上の処理を分岐するときに使用すると便利です。

■書式

- **条件に一致する処理だけおこなう**

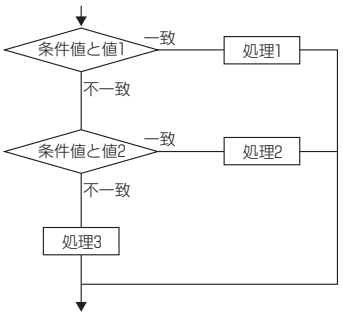
```
Select Case　条件値
    Case 値1
        処理1
    Case 値2
        処理2
    Case 値3
        処理3
End Select
```



「条件値」が「値１」に一致しているときに「処理１」を、「値2」に一致しているときに「処理2」を、「値3」に一致しているときに「処理3」を実行します。

- **条件に一致する処理と一致しない処理をおこなう(「Case Else」を使う)**

```
Select Case　条件値
    Case 値1
        処理1
    Case 値2
        処理2
    Case Else
        処理3
End Select
```



参 考　「Case 値 処理」は、複数記述することができます。
最大127まで記述できます。

「条件値」が「値１」に一致しているときに「処理１」を、「値2」に一致しているときに「処理2」を、「値１」にも「値2」にも一致しないときは「処理3」を実行します。

- 「Select Case」の「条件値」には、変数、またはメソッドの戻り値を1つ記述します。
  メソッドについては、「5-2 メソッド」(5-5ページ)を参照してください。

  例 Select Case termId……変数
  Select Case FileStream <"2:ファイル1.txt"> :Open() メソッドの戻り値
  「FileStream」については、「7-8 ファイルの操作」(7-160ページ)を参照してください。

- 「Case 値」の値には、下記のように数値または文字列を記述します。
  変数での記述はできません。

  例
  Case 1 …… 数値
  Case "001" …… 文字列
  Case true…… 論理値
  Case 2,"002" …… 数値、文字列を複数記述

- 処理には、複数のステートメント(文)が記述できます。

  例1 typeの値が1または"001"のとき、termIDに10を代入します。
  typeの値が2または"002"のとき、termIDに20を代入します。
  typeの値が3または"003"のとき、termIDに30を代入します。

```
Select Case type
  Case 1,"001"
    termID=10
  Case 2,"002"
    termID=20
  Case 3,"003"
    termID=30
End Select
```

  例2 typeの値が1または"001"のとき、termIDに10を代入します。
  typeの値が2または"002"のとき、termIDに20を代入します。
  typeの値が1でも"001"でも2でも"002"でもないとき、termIDに30を代入します。

```
Select Case type
  Case 1,"001"
    termID=10
  Case 2,"002"
    termID=20
  Case Else
    termID=30
End Select
```

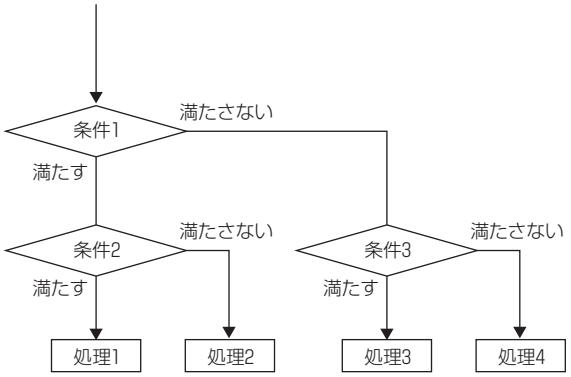
分岐条件の入れ子(ネスト)

条件分岐の中に条件分岐を入れることで、複雑な条件分岐を処理することができます。
繰り返し処理の中に条件分岐を入れることもできます。
繰り返し処理については、「4-2 処理を繰り返す」(4-7ページ)を参照してください。

▶ 重 要 **If文、Select文、While文、For文を組み合わせて、入れ子(ネスト)ができるのは、最大20階層までになります。**

■書式

```
If  条件1  Then
    if  条件2  Then
        処理1
    Else
        処理2
    End If
Else
    if  条件3  Then
        処理3
    Else
        処理4
    End If
End If
```



参 考 If文の入れ子に、Select文を記述することもできます。

例 modeがtrueのとき、keycodeにより処理を分岐し、true以外のときLEDを点灯します。

```
If  mode is true Then                                    //条件分岐のIf文
  Select Case keycode                                    //条件分岐の入れ子のSelect文
    Case "F1"
      Log:log_main() Return("loopout")
    Case "F2"
      Job1:job1_main() Return("loopout")
    Case "F3"
      Return("loopout")
  End Select
Else
  With Led
    :onTime = 1 :offTime = 0 :loopCount = 3 :color = "orange"
    :Exec()
  EndWith
End If
```

4-2 処理を繰り返す

While文やFor文を使用して、処理を繰り返します。

▶ 重 要

If文、Select文、While文、For文を組み合わせて、入れ子にできるのは最大20階層までになります。
入れ子については、「分岐条件の入れ子(ネスト)」(4-6ページ)を参照してください。

While、Wend、Wbreak、Wcontinue

継続条件を指定して、繰り返し処理を行います。
繰り返し処理を途中で抜けるには、「Wbreak」、または「Wcontinue」を使います。

書式

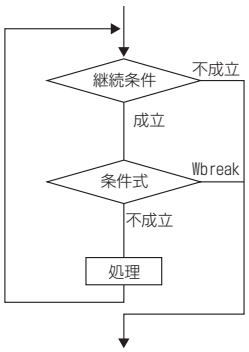
- **基本**

```
While　継続条件
　　処理
Wend
```

継続条件が満たされているあいだ、処理を繰り返します。継続条件が満たされなくなると、繰り返し処理は終了します。

- **「Wbreak」:処理から抜ける**

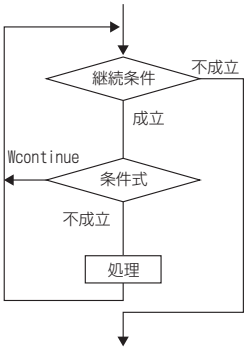
```
While　継続条件
　　条件式
　　Wbreak
　　処理
Wend
```



「Wbreak」は、繰り返し処理から抜けるときに使用します。

- **「Wcontinue」:処理の先頭に戻る**

```
While　継続条件
    条件式
    Wcontinue
    処理
Wend
```



「Wcontinue」は、繰り返し処理の途中で抜けて、処理の先頭に戻すときに使用します。

- 「While」の「継続条件」は、条件式を記述します。

数値比較の例

While (i < 10)　　·····iが10より小さい間

「<」については、📖「数値比較演算子」(3-4ページ)を参照してください。

文字列比較の例

While　(Str lt "Z")　·····Strが"Z"より小さい間

「lt」については、📖「文字列比較演算子」(3-5ページ)を参照してください。

論理型比較の例

While　(i is true)　·····iがtrue(真)の間

「is」については、📖「論理型比較演算子(is)」(3-7ページ)を参照してください。

「While true」と記述することもできます。この場合、終了しないで繰り返すという意味です。

参考　条件式の組み合わせ式も使用できます。
📖「3-3 条件を組み合わせる」(3-8ページ)

- 「Wbreak」や「Wcontinue」は、下記のように条件式の処理の中に記述します。

例

```
If 条件 Then
  Wbreak(またはWcontinue)
End If
```

idxを1ずつ加算し、配列変数keycodeに順次値を代入します。100になったら繰り返し処理から抜けます。

```
idx=0
While true
  If idx >=100 Then
    Wbreak
  End If
  keycode[idx]=idx*10
  idx = idx+1
Wend
```

For、Next、Fbreak、Fcontinue

繰り返し回数を数えるカウンタを使って、処理を繰り返します。
繰り返し処理を途中で抜けるには、「Fbreak」、または「Fcontinue」を使います。

■書式

- **基本**

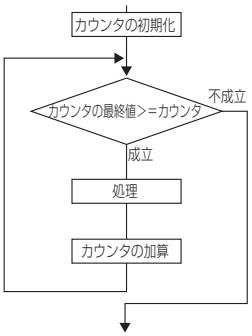
```
For　カウンタの初期化式　to　カウンタの最終値
　　処理
Next
```

  - カウンタ(変数)に初期値を設定し、カウンタが最終値を超えるまでの間、処理が繰り返されます。
  - カウンタは、処理が1回終えると1ずつ加算されます。

- **「カウンタの増分」を指定する**

```
For　カウンタの初期化式　to　カウンタの最終値　step　カウンタの増分処理
　　処理
Next
```

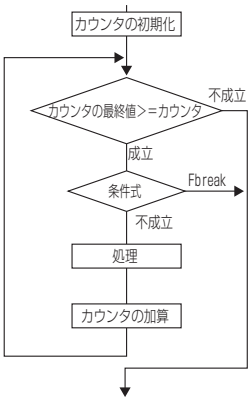
  - カウンタ(変数)に初期値を設定し、カウンタが最終値を超えるまでの間、処理が繰り返されます。
  - カウンタは、処理が1回終わると「stepカウンタの増分」で指定された正負の整数値だけ加算されます。
  - 右図は、カウンタの初期値＜カウンタの最終値の場合です。



- **「Fbreak」:処理から抜ける**

```
For　カウンタの初期化式to　カウンタの最終値
　　条件式
　　Fbreak
　　処理
Next
```

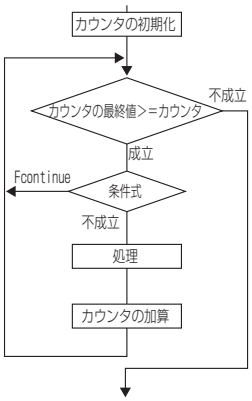
  - 「Fbreak」は、繰り返し処理を抜けるときに使用します。
  - 書式2のようにカウンタの増分を指定することもできます。
    右図は、カウンタの初期値＜カウンタの最終値の場合です。



- 「Fcontinue」:処理の先頭に戻る

  For　カウンタの初期化式　to　カウンタの最終値
  **　　条件式**
  **　　Fcontinue**
  **　　処理**
  Next

  - 「Fcontinue」は、繰り返し処理を抜けて、繰り返し処理の先頭に戻すときに使用します。
  - 書式 2 のようにカウンタの増分を指定することもできます。右図は、カウンタの初期値＜カウンタの最終値の場合です。



●注意事項

- 「カウンタの初期化式」は、繰り返しカウンタ(変数)に初期値を設定します。カウンタ(変数)は、メソッド内で定義した変数しか使用できません。

  例　i=0

- 「to カウンタの最終値」は、カウンタの最終値を記述します。この値を超えたら、処理の繰り返しから抜けます。

  例　to　100 ……iの最終値は100、101になったら繰り返し処理が終了

- 「step カウンタの増分」には、正負の整数値を記述します。
  カウンタの増分に、変数や式での記述はできません。

  例　step　2

- 「Fbreak」、「Fcontinue」は、下記のように条件式の処理の中に記述します。

  例
  ```
  If 条件 Then
    Fbreak(またはFcontinue)
  End If
  ```

- 「処理」には、複数のステートメントが記述できます。

  例1　昇順にary配列の要素を取り出し、1000倍してary2に代入します。
  ```
  For i=0 to ary.size-1
    ary2[i]=ary[i]*1000
  Next
  ```

  参考　「For i=10 to 0 step –1」と記述すると、iを10から1ずつ減らして0になるまで処理を繰り返します。

  例2　降順にary配列の要素を取り出し、1000倍してary2に代入します。

  ```
  For idx=ary.size-1 to 0 step -1
    ary2[idx]=ary[idx]*1000
  Next
  ```

5

スクリプト構成

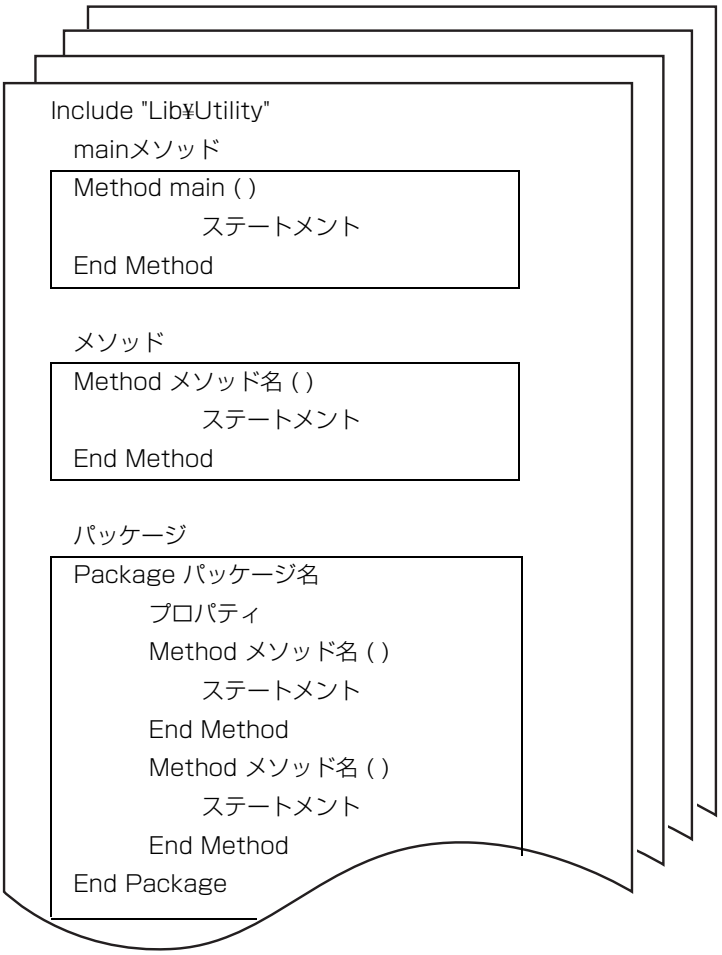
スクリプトプログラムを構成するmainメソッド、メソッド、パッケージ、コントロールについて説明します。

5-1　概要 ……5-2
5-2　メソッド ……5-5
5-3　パッケージ …… 5-11
5-4　コントロール …… 5-14
5-5　名前の参照、スコープ …… 5-23
5-6　別スクリプトファイルの読み込み …… 5-26
5-7　別プログラムの実行 …… 5-27
5-8　実行時例外／エラー …… 5-28
5-9　空白文字とコメント …… 5-30

5-1 概要

スクリプトの構成

スクリプトプログラムを記述する構成単位には、「main メソッド」、「メソッド」、「パッケージ」があります。これらを必要に応じて定義することによって、スクリプトプログラムを作成、修正することができます。
また、システムが提供する「コントロール」があります。「コントロール」を使用して、《BTシリーズ》の動作を簡単に制御できます。



●mainメソッド

「mainメソッド」は、1つのスクリプトファイルに1つだけ記述します。
スクリプトプログラムは、ここから実行されます。「mainメソッド」のステートメントから、他のメソッドや、パッケージのメソッド、コントロールのメソッドを呼び出します。

●メソッド

スクリプトプログラムの基本構成単位です。
一連のステートメントを記述し、1つのまとまった処理をおこないます。「mainメソッド」や「パッケージ」のメソッドから呼び出されることにより、処理が実行されます。

●パッケージ

パッケージ内に、必要に応じて複数の「メソッド」が定義できます。関連のある処理をグループにまとめることができます。
複数のメソッドが共通に参照できるプロパティ（変数）を、定義できます。

●コントロール

システムで定義されています。《BTシリーズ》のデータ入力、ファイル入出力などの動作制御ができます。
mainメソッド、メソッド、パッケージのメソッドから呼び出すことができます。

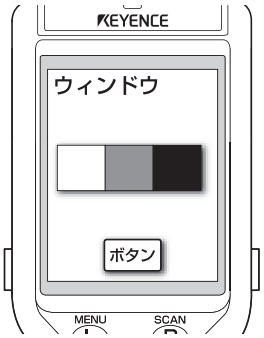
GUIのプログラミング

GUIのプログラミングは、各Widget表示とイベント駆動型プログラムという2つの構成要素があります。

●タッチパネルの表示

GUI の基本構成単位は、ウィンドウです。下図のように、ウィンドウにラベル、ボタンなどを配置して画面を作成します。

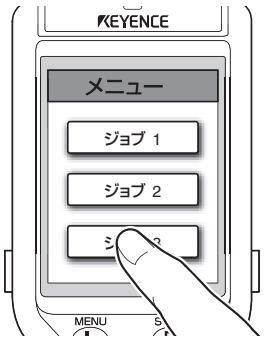
各Widgetの表示は、Widget操作コントロールでプログラミングします。



●イベント駆動型プログラム

画面上のボタンを押す、何か項目を選択する、文字を入力するなどの操作が実行されたときに、プログラムで発信される信号をイベントといいます。発信されたイベントにしたがって、ボタンを押したときの処理などをプログラミングすることを、イベント駆動型プログラムといいます。

イベントに対応しているWidget操作コントロールやEventコントロールなどでプログラミングします。



ボタンを押す

内容を表示する

5-2 メソッド

メソッドには、1 つのまとまった処理を記述します。メソッドはスクリプトファイルの任意の場所に定義ができ、任意の場所から呼び出しができます。何回も同じ処理をおこなうときに、メソッドを使用すると、簡潔なスクリプトプログラムが作成できます。

参 考　変数、配列変数にもメソッドがありますが、定義内容、呼び出し方が異なります。
「2-4 変数、配列変数のプロパティとメソッド」(2-11ページ)

メソッドの定義

メソッドはステートメントの集合です。1つのまとまった処理を記述するときに定義します。
- メソッドには引数があり、ステートメントの処理に必要な値を渡すことができます。
- メソッドには戻り値があり、処理結果の値を返すことができます。

■書式

```
Method メソッド名(メソッド引数・・)
    変数
    Begin
    ステートメント
End Method
```

- 「変数」と「ステートメント」は、複数記述できます。
- 「Begin」は、「変数」領域と「ステートメント」領域を区切る予約語です。
- メソッドの中で記述する変数を「ローカル変数」と呼びます。配列変数も記述できます。

- メソッドの中で、別のメソッド(メソッド、コントロールのメソッド、パッケージのメソッド)を呼び出すことができます。
「メソッドの呼び出し」(5-7ページ)

▶ 重 要　**メソッドから別メソッドの呼び出しができるのは、1 つのスクリプトファイルで最大20 階層までになります。ただし、20 階層以下でもデータ格納領域が不足して実行時例外となることがあります。**

- メソッドの中から、そのメソッド自身を呼び出すことができます(再帰呼び出し)。

参 考　再帰呼び出しをおこなうと、無限に自身のメソッドを呼び出し、データ格納領域が不足して実行時例外となることがあります。適切な終了条件を記述することが必要です。

■メソッド名

メソッド名は、全角文字または半角英文字で始め、最大126バイト(全角で63文字、半角で126文字)以内です。全角文字、半角英数字と”_”(アンダーバー)が使用できます。

ポイント
- **同一パッケージ内で同じメソッド名は定義できません。**
- **半角英文字は、次の点に注意してください。**
  先頭文字に半角数字と半角カタカナは使用できません。
  大文字と小文字は区別されます。たとえば、"A"と"a"は別の文字として扱われます。
  予約語と同じ名前は、大文字・小文字の区別に関係なく使用できません。
  たとえば、"BEGIN"、"begin"、"RETURN"、"return"などは使用できません。
  予約語については、「付録2 予約語一覧」(付-3ページ)を参照してください。

■メソッド引数

メソッド引数には、変数、配列変数が記述できます。

- Const付きの変数は記述できません。
- Const付きの配列変数は記述できます。

例

```
Method Sub(data1,data2)   ·····変数
Method Sub(data[ ])       ·····配列変数
Method Sub(Const  data[ ])·····Const付きの配列変数
```

▶ 重 要

1つのメソッドに、最大50個のメソッド引数が定義できます。

■戻り値

Return(引数)

- 引数には、変数を記述します。配列変数は使用できません。
- メソッドの中に、複数記述することができます。
- 戻り値が不要なときは、記述しません。

「Return」の詳細については、「メソッドから抜ける」(5-10ページ)を参照してください。

例 メソッド引数の文字列dataの文字列長がlenより短いとき、dataの後ろをスペースで詰めます。

```
Method fix_length(len,data)          //メソッドの宣言(引数はlenとdata)
  cnt                                //ローカル変数の宣言
  strdata
  Begin
      strdata=data
      If data.length < len Then
        For cnt=1 to len-data.length
          strdata = strdata & " "
        Next
      EndIf
  Return(strdata)                    //戻り値の設定
End Method                           //メソッドの締めくくり
```

メソッドの呼び出し

呼び出したメソッドの処理を実行します。

■書式

メソッド名(メソッド引数・・・)

例　fix_length(len,data)

■メソッド引数の受け渡し

メソッド引数を使用して、呼び出すメソッドに値を渡すことができます。
()の間に下記をカンマで区切って記述します。

- 数値、文字列、論理値
- 変数、配列変数
- 演算式
- プロパティ値(パッケージのプロパティ、コントロールのプロパティ)
- メソッド値(メソッド、パッケージのメソッド、コントロールのメソッド)

パッケージのプロパティとメソッドについては、「5-3 パッケージ」(5-11ページ)を参照してください。
コントロールのプロパティとメソッドについては、「5-4 コントロール」(5-14ページ)を参照してください。

例　x=10 y=20 ary[ ]={1,2,3}のとき

```
Sub(5,"Red",true)          ・・・・・数値、文字列、論理値
Sub(x,ary)                 ・・・・・変数、配列変数
Sub(x+y)                   ・・・・・演算式
Sub(InputString:data)      ・・・・・コントロールのプロパティ値
Sub(InputString:Exec())    ・・・・・コントロールのメソッド値
```

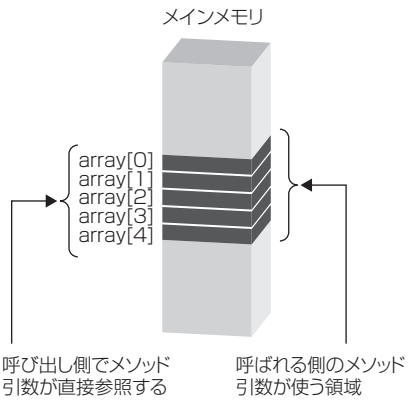
▶ 重 要　**引数に演算式やメソッドを複数記述するときは、右側から評価されるため、引数の順序に注意する必要があります。**

メソッド引数は、下表のように呼び出すときに使用できる変数の型と、呼ばれる側の変数の型と組み合わせがあります。これ以外の組み合わせは、コンパイルエラーとなります。

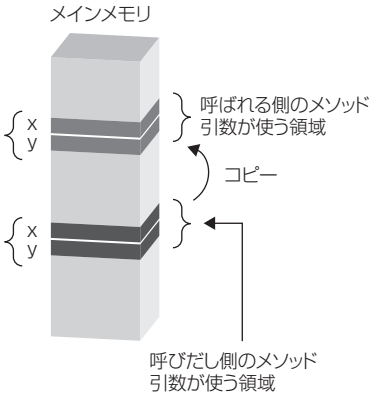
呼ばれる側のメソッド引数 (メソッド定義側)	呼び出すときに使用できるメソッド引数 (参照側)	メソッド実行後の 引数の値※1
変数	文字列、数値、論理値、 変数(Const付き変数を含む) 演算式、 プロパティ値、別メソッドの戻り値	変わらない
配列変数	配列変数	変わることがある
Const付き配列変数	配列変数(Const付き配列変数を含む)	変わらない

※1　メソッド引数は、定義したときに変数、配列変数により、渡し方が以下のように異なるため、メソッド実行後の引数の値が変わるものと、変わらないものがあります。

配列変数の引数:参照渡し



変数の引数:値渡し



- 配列変数の引数は、参照渡しです(左図)。
  引数の呼び出し側と、呼ばれる側が同じメモリ領域を使用しているため、値が変わることがあります。
- 配列変数以外の引数は、値渡しです(右図)。
  引数の呼び出し側と、呼ばれる側が異なるメモリ領域を使用しているため、値が変わることはありません。

例 fix_lengthメソッドを呼び出します。

```
Method main()
  fix_str=""
  str="A123"
  size=10
  Begin
      fix_str=fix_length(size,str)      //fix_lengthメソッドの呼び出し。
                                        //メソッドの戻り値がfix_strに代入されます。
End Method

//メソッドfix_lengthの定義
Method fix_length(len,data)             //メソッドの宣言(引数はlenとdata)
  cnt                                   //ローカル変数の宣言
  strdata
  Begin
      strdata=data
      If data.length < len Then
        For cnt=1 to len-data.length
          strdata = strdata & " "
        Next
      EndIf
  Return(strdata)                       //戻り値の設定
End Method                              //メソッドの終了
```

mainメソッド

mainメソッドは、スクリプトプログラムで最初に実行されます。

▶ 重 要

- **1つのスクリプトファイルに、1つのmainメソッドを必ず記述します。**
- **スクリプトファイルにmainメソッドがないと、コンパイルエラーとなります。**
- **複数のスクリプトファイルでプログラムが構成されているときには、辞書式でソートして先頭にくるスクリプトファイルの「main」メソッドが最初に実行されます。**

■書式

Method main()
　変数
　Begin
　ステートメント
End Method

メソッド引数はありません。

- 「Begin」は、「変数」領域と「ステートメント」領域を区切る予約語です。
- main メソッドから、コントロールやパッケージメソッドを呼び出すことによって、プログラムの実行を継続します。
- 「System：Load」メソッドを使用して、他のスクリプトファイルのmainメソッドが呼び出せます。
「5-7 別プログラムの実行」(5-27ページ)

例　mainメソッドからメソッドSubを呼び出します。

```
Method main()                    //mainメソッドの宣言
  Begin
    Sub(10,5,7)                  //メソッド呼び出し
  End Method

Method Sub(x, y, z)
  Begin

End Method
```

メソッドから抜ける

メソッドから抜けるときは、Return文を使用します。
Return文が実行されると、メソッドの処理を終了して、呼び出し元に戻ります。
また、呼び出し元にメソッドの戻り値を返すことができます。
メソッドについては、「メソッドの定義」(5-5ページ)を参照してください。

■書式

Return(引数)

■引数

引数には、数値、文字列、論理値、変数、メソッドの戻り値を記述します。

例

```
Return(1)                       ・・・・・数値
Return("OK")                    ・・・・・文字列
Return(true)                    ・・・・・論理値
Return(KeyCode)                 ・・・・・変数
Return(InputString:Exec())      ・・・・・メソッドの戻り値
Return(nil)                     ・・・・・戻り値なし
```

例

メソッド処理の中で、まず条件が一致していれば、そのままメソッドを抜けます。
条件が一致していなければ、次の処理を行い、メソッドを抜けます。

```
Method fix_length(len,data)          //メソッドの宣言(引数はlenとdata)
  cnt                                //ローカル変数の宣言
  strdata
  Begin
    strdata=data
    If data.length=len Then
      Return(strdata)                //メソッドの途中で処理を抜けます
    End If
    For cnt=1 to len-data.length
      strdata = strdata & " "
    Next
  Return(strdata)                    //メソッドの戻り値を設定します
End Method
```

5-3 パッケージ

パッケージを使用すると、関連のある複数のメソッドを1つのグループにまとめることができます。
パッケージは、スクリプトファイルの任意の場所に定義ができ、同一スクリプトファイルの任意の場所からアクセスできます。

▶ 重 要 **パッケージの定義は、1つのスクリプトファイルに最大100個までになります。**

パッケージの定義

パッケージは、プロパティ(変数)とメソッドの集まりです。
関連のあるメソッドを同一パッケージに記述し、共通のプロパティが参照できます。

■書式

```
Package　パッケージ名
    プロパティ
    Method メソッド名(メソッド引数・・)
        ローカル変数
        Begin
            ステートメント
    End Method
End Package
```

- プロパティとメソッド(Method～End Method)は、複数記述することができます。

▶ 重 要 **パッケージ内(Package～End Package)にパッケージを記述(入れ子構造)することはできません。**

- プロパティには、変数、配列変数が記述できます。
  「2-2 変数」(2-3ページ)
  「2-3 配列変数」(2-9ページ)

▶ 重 要 **1つのスクリプトファイルに、最大1000個までのプロパティ(配列要素数を含まない)を記述できます。**

- パッケージ内のメソッドの定義は、通常のメソッドと同一です。
  「メソッドの定義」(5-5ページ)

■名前

パッケージ名、プロパティ名、メソッド名は、全角文字または半角英文字で始め、最大126バイト(全角で63文字、半角で126文字)以内です。全角文字、半角英数字と”_”(アンダーバー)が使用できます。

ポイント

- **半角英文字は、次の点に注意してください。**
  - **先頭文字に半角数字と半角カタカナは使用できません。**
  - **大文字と小文字は区別されます。たとえば、"A"と"a"は別の文字として扱われます。**
  - **予約語と同じ名前は、大文字・小文字の区別に関係なく使用できません。**
    たとえば、"BEGIN"、"begin"、"RETURN"、"return"などは使用できません。
    予約語については、「付録2 予約語一覧」(付-3ページ)を参照してください。
- **他のパッケージ名、コントロール名と同じ名前は定義できません。**
- **同一パッケージ内で、プロパティ名、メソッド名、メソッド引数名、ローカル変数名は、同じ名前は定義できません。**

パッケージへのアクセス

■書式

- **同一パッケージ内**
  プロパティ名、メソッド名でアクセスします。

- **同一パッケージ以外**
  パッケージ名:プロパティ名
  パッケージ名:メソッド名(メソッド引数・・・)

例　mainメソッドからパッケージMenuのメソッドSubを呼び出します。

```
Method main()
    Begin
        Menu:Sub(100,100,10)//パッケージMenuのSubメソッドの呼び出し
End Method

Package Menu                    //パッケージMenuの定義
    Method Sub(x, y, z)         //メソッドSubの定義
    Begin
    End Method
End Package
```

With、EndWith(プロパティ、メソッドへのアクセス)

パッケージまたはコントロールのプロパティ、メソッドへアクセスするとき、パッケージ名やコントロール名を省略できます。
With、EndWithを使用することによって、シンプルで見やすい記述ができます。
コントロールの説明については、「5-4 コントロール」(5-14ページ)を参照してください。

■書式

- **パッケージへのアクセス**

```
With パッケージ名
    :プロパティ名
    :メソッド名
EndWith
```

- **コントロールへのアクセス**

```
With コントロール名
    :プロパティ名
    :メソッド名
EndWith
```

- プロパティとメソッドは複数記述できます。
- 「With EndWith」の間に、変数が記述できます。

▶ 重 要 **「With」と「EndWith」の間に、別のコントロールまたはパッケージの「With」、「EndWith」が記述できます。**

例

- **With　EndWithを使用してLedメソッドを呼び出します。**

```
With　Led
    :onTime=10　:offTime=5　:loopCount=5　:color="red" :Exec()
EndWith
```

- **With　EndWithを使用しないとき**

```
LED:onTime=10
LED:offTime=5
LED:loopCount=5
LED:color="red"
LED:Exec()
```

5-4 コントロール

システムが提供するコントロールを使用して、《BTシリーズ》の動作を制御することができます。
画面動作、入力動作、出力動作などの各種コントロールがあります。
「7章 コントロール一覧」(7-1ページ)

コントロールへのアクセス

■書式

各コントロールには、プロパティとメソッドがあります。下記の書式でアクセスします。

- **コントロール名:プロパティ名**
- **コントロール名:メソッド名(引数・・・)**
  - プロパティは、メソッド、パッケージのメソッドから参照できます。
  - メソッドを呼び出すことによって、《BTシリーズ》の動作を命令できます。

例
```
Buzzer:onTime=5
Buzzer:offTime=1
Buzzer:loopCount=2
Buzzer:Exec()
```

  - 「With」、「EndWith」が使用できます。

例
```
With Buzzer
  :onTime=5 :offTime=1 :loopCount=2  :Exec()
EndWith
```

タグ指定コントロール

コントロールに任意の名前(タグ)を付与して、コントロールの識別子として使用します。
タグ指定できるコントロールは、次のとおりです。

- Widget操作コントロール
  「Widget操作コントロール」(5-17ページ)を参照してください。
- FileStreamコントロール
- Timerコントロール
- Serialコントロール

■書式

タグ指定コントロールには、下記の書式でアクセスします。

- **コントロール名<タグ名>:プロパティ名**
- **コントロール名<タグ名>:メソッド名(引数・・・)**

■タグ名

- **Widget操作コントロール**
  タグ名は、全角文字または半角英文字で始め、最大255 バイト(全角で127 文字、半角で255文字)以内です。全角文字、半角英数字と"_(" アンダーバー)が使用できます。

注 記

- **半角英数字は次のことに注意してください。**
  - **先頭文字に半角数字と半角カタカナは使用できません。**
  - **大文字と小文字は区別されます。たとえば、"A"と"a"は別の文字として扱われます。**

予約語と同じ名前は、大文字・小文字の区別に関係なく使用できません。
たとえば、"BEGIN"、"begin"、"RETURN"、"return"などは使用できません。
予約語については、「付録2 予約語一覧」(付-3ページ)を参照してください。

- **FileStreamコントロール**
  タグ名には、ファイル名を次の形式で指定します。
  ドライブ番号:パス付きファイル名(最大255バイト以内)

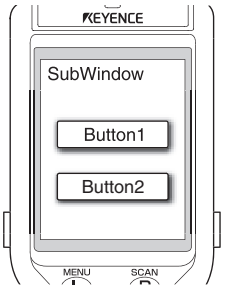
例 2:A¥A-1¥ccc.txt

- **Timerコントロール**
  タグ名には、タイマー番号を1～8で指定します。

- **Serialコントロール**
  タグ名には、通信で使用するデバイスによって、次のように指定します。

デバイス	タグ名
Bluetooth通信	"Bluetooth"
赤外線通信	"IrDA"
モデム通信	"Modem"
RS-232C通信	"232C" ※BT-1000/1500シリーズのみ
USB通信	"USB" ※BT-1000/1500シリーズのみ

例 Widget操作コントロールのタグ名




```
/* 「SubWindow」というタグ名のウィンドウを、
   ルートウィンドウ上に作成します。*/
Window<"SubWindow">:Create("ROOT_WINDOW")

/* 「Button1」というタグ名のボタンを、「SubWindow」と
   いうタグ名のウィンドウ上に作成します。*/
Button<"Button1">:Create("SubWindow")

/*「Button2」というタグ名のボタンを、
  「SubWindow」というタグ名のウィンドウ上に作成します。*/
Button<"Button2">:Create("SubWindow")
```

例 FileStreamコントロールのタグ名

```
/* ドライブ2にあるSampleディレクトリ上のFile1.txtファイルを、
   読み込みモードでオープンします。*/
FileStream<"2:Sample¥File1.txt">:Open("r")
```

例 Timerコントロールのタグ名

```
// タイマー番号1のタイマーに10秒を設定します。
Timer<1>:Set(1000)
```

例 Serialコントロールのタグ名

```
// Bluetoothの送受信に使用するバッファをクリアします。
Serial<"Bluetooth">:Clear(3)
```

Widget操作コントロール

タッチパネルを表示、操作する画面部品をWidgetと呼びます。
Widgetの動作について記述するためのコントロール群を、Widget操作コントロールといいます。

■Widget操作コントロールの種類

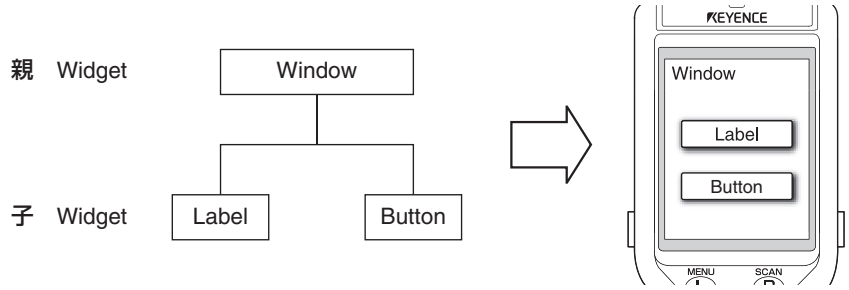
まず親Widgetを作成してから、親Widget上に子Widgetを作成(複数可)します。
上記で作成した子Widgetが親Widgetになって、そのWidget上に子Widgetを作成できます。
使用方法については、「タッチパネルの構成」(5-17ページ)を参照してください。

コントロール	画面部品の説明	親Widgetになるコントロール	子Widgetになるコントロール
Window	ウィンドウです。	Window/ Canvas/ GroupBox	すべてのWidget操作コントロール※
Canvas	ウィンドウのひとつです。直接描画が可能です。		
GroupBox	ウィンドウ上に枠を表示して、複数のWidgetを1つのグループとして扱います。		
TextField	1 行のテキスト表示、入力をします		–
TextBox	複数行のテキスト表示、入力をします。		
Label	編集できないテキストを表示します。		
ListBox	選択候補を一覧で表示･選択します。		
ComboBox	選択候補をドロップダウンリストで表示･選択します。		
CheckBox	選択したい項目にチェックを入れることで選択するボタン(複数選択可能)です。		
RadioButton	選択したい項目にチェックを入れることで、排他的に選択するボタン(複数選択不可)です。		
Button	タッチパネルの押下に反応して動作するボタンです。		

※ ルートウィンドウとエミュレートウィンドウは、子Widgetになりません。
ルートウィンドウとエミュレートウィンドウについては、「タッチパネルの構成」(5-17ページ)を参照してください。

■タッチパネルの構成

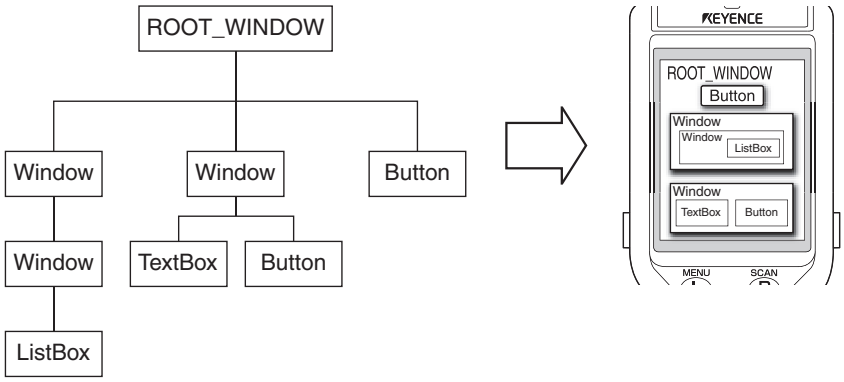
タッチパネルの基本構成単位は、Windowです。
Window(親Widget)に画面部品(子Widget)を配置して、タッチパネルを作成します。



●ルートウィンドウ

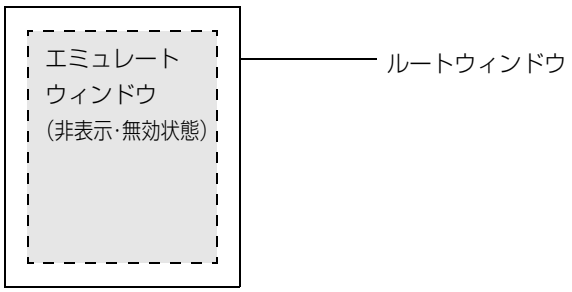
システムには、すべてのWidgetの親Widgetになる「ルートウィンドウ」があります。
ルートウィンドウに、ウィンドウを配置したり、ボタンを直接配置したりできます。
ルートウィンドウは、作成したり、削除したり、大きさを変えたりできません。

タッチパネル作成例



●エミュレートウィンドウ(BT-W100/W80/W70シリーズのみ)

BT-1000/1500/600シリーズの動作互換をさせるためのウィンドウです。
エミュレートウィンドウは、最初からルートウィンドウ上に非表示･無効状態で存在する特別なウィンドウです。Widgetの貼り付けも可能です。



■書式

▶ 重 要 **Widgetを使用するには、まずWidgetの作成(またはコピー)が必要です。**

- **Widget名<タグ名>:Create(親Widget名)**
- **Widget名<タグ名>:Copy(親Widget名, コピー元Widget名)**

Widgetの表示位置、大きさなどは、Widget操作コントロールのプロパティで指定します。

例 ウィンドウを作成します。

```
// 「サブウィンドウ1」をルートウィンドウ上に作成します。
Window<"サブウィンドウ1">:Create("ROOT_WINDOW")
```

例 ウィンドウをコピーして作成します。

```
/* 「サブウィンドウ1」をコピーして 「サブウィンドウ2」として、ルートウィンドウ上に
   作成します。*/
With Window<"サブウィンドウ2">
  :Copy("ROOT_WINDOW", "サブウィンドウ1")
EndWith
```

例 ボタンを作成します。

```
// 「ボタン1」を「サブウィンドウ1」上に作成します。
Button<"ボタン1">:Create("サブウィンドウ1")
```

例 ボタンをコピーして作成します。

```
// 「ボタン1」をコピーして 「ボタン2」として、「サブウィンドウ1」上に作成します。
With Button<"ボタン2">
  :Copy("サブウィンドウ1", "ボタン1")
EndWith
```

イベント対応コントロール

■イベント駆動型プログラム

タッチパネル上のボタンを押す、文字を入力するなどの操作が実行されたときに、プログラムで発信される信号をイベントといいます。イベントが発生したときに対応する処理を記述するプログラムが、イベント駆動型プログラムです。

下図に、タッチパネルでの動作を例にして、プログラムでのイベントの流れを示します。

(1) イベント発生時に実行する処理を設定

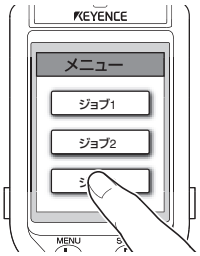
↓

(2) イベントを待つ

↓

(3) イベントの発生

【例】ボタン押下



↓

(4) (1)で設定した処理を実行

【例】ボタン押下時の表示

スクリプトプログラムには、イベント発生させるためのイベント登録、イベント取得、イベント処理を記述します。

●イベントの種類

イベントの種類を一覧で示します。

イベントの種類	イベント発生のタイミング
キーイベント	キーを押したとき
Widgetイベント	キーとタッチパネル上で操作したとき
コード読み取りイベント	コードを読み取ったとき、読み取りエラー/タイムアウトになったとき
無線デバイスイベント	無線接続/切断したとき、圏外になったとき DHCPの更新に失敗したとき
ハンディデバイスイベント	電源が入ったとき(レジューム有効時のみ) 置き台に設置･置き台から取り外したとき メインバッテリの残量が変化したとき
タイマーイベント	タイムアップしたとき

■イベントの登録と取得

●イベントの登録

イベントの登録は、イベント対応コントロールのプロパティでおこないます。

登録した種類のイベントだけ、イベント取得ができます。

イベントの種類 コントロール	イベント発生のタイミング	コントロール	プロパティ
キーイベント	キーを押したとき	Key	onPress
Widgetイベント	Widgetにフォーカスが入ったとき/ Widgetにフォーカスしなくなったとき/ タッチパネルからリリースしたとき	Window Canvas GroupBox TextField TextBox ListBox ComboBox CheckBox RadioButton Button	onFocusIn/ onFocusOut/ onTouchUp
	編集開始したとき/ 編集完了したとき/ 編集をキャンセルしたとき/ 編集状態が解除になったとき	TextField TextBox ListBox ComboBox	onEditStart/ onEditEnd/ onEditCancel/ onTouchOut
	コードを読み取ったとき/	TextField TextBox	onScanComplete
	選択項目を変更したとき	ListBox ComboBox	onSelectChange
	チェック項目を変更したとき	CheckBox RadioBottun	onCheckChange
	押下したとき	Button	onButton
コード読み取り イベント	コードを読み取ったとき／ 読み取りエラーが起きたとき／ タイムアウトになったとき	BCR	onScanComplete/ onScanError/ onScanTimeout
無線デバイス イベント	無線接続／ 切断したとき／ 圏外になったとき	Network/ WLAN/ Bluetooth	onConnect/ onRefused/ onOutrange
	DHCPでIPアドレスを更新したとき	Network	onDHCPIpChanged
ハンディ デバイスイベント	電源が入ったとき(レジューム有効時のみ)/ 置き台に設置･置き台から取り外したとき/ メインバッテリの残量が変化したとき	Handy	onResume/ onCradle/ onBatteryLevelChange
タイマーイベント	タイムアップしたとき	Timer	onTimer

●イベントの取得
登録したすべてのイベントは、「Event」コントロールで取得します。

●イベントの処理
イベントを取得したときに処理する「メソッド」を作成します。

■書式

重要

イベントの登録では、イベントが発生したときに実行したいメソッドを指定しておきます。イベント処理のメソッドには、かならず"sender"の引数を記述します。senderには発生したイベントに応じた値／文字列が渡されます。

- **イベント登録**
  コントロール名<タグ名>:イベントプロパティ名=イベント処理のメソッド名

- **イベント取得**
  Event:Wait()

- **イベント処理**
  メソッド名(sender)

例 「ボタン1」を押下時のイベントの登録と取得

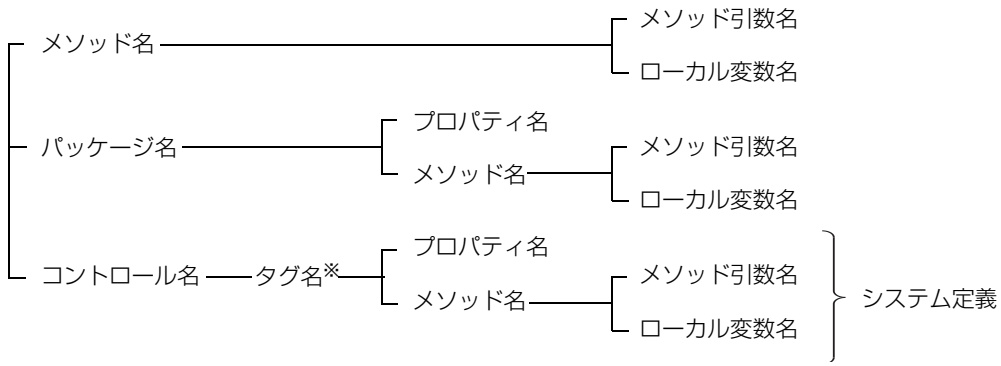
```
Method  main()
  Begin
    With Button<"ボタン1">
    :onButton=OnButton1
    //イベントを登録して、イベントが発生したときのイベント処理を指定

    EndWith

    Event:Wait()                    //イベント取得
EndMethod

//イベント処理のメソッド
Method OnButton1(sender)
  Begin
     Handy:ShowMessageBox(sender & "が押されました")
 EndMethod
```

5-5 名前の参照、スコープ

変数、メソッド、パッケージなど名前を持つものには、スコープ(有効範囲)、参照方法が決められています。

定義場所

メソッド名、パッケージ名は、スクリプトファイルの任意の場所に定義できます。
それ以外の名前は以下のような包含関係と階層があり、定義場所が決められています。



※　コントロールによっては、コントロール名のうしろにタグ名を任意に追加できます。

スコープと参照

名前のスコープ(有効範囲)と参照方法は、下表のようになります。

名前	スコープと参照方法
メソッド引数名 ローカル変数名	定義されたメソッド、パッケージのメソッド内でのみ、ローカル変数名、メソッド引数名はそのままで参照できます。※1 他のメソッドで定義されたメソッド引数、ローカル変数は参照できません。
パッケージのプロパティ名 パッケージのメソッド名	同一パッケージ内では、プロパティ名、メソッド名はそのままで参照できます。 同一パッケージ以外から参照するときは、下記のように記述します。 • パッケージ名:プロパティ名 • パッケージ名:メソッド名(メソッド引数・・・)
コントロールのプロパティ名 コントロールのメソッド名	任意の場所から参照できます。 • コントロール名:プロパティ名 • コントロール名:メソッド名(メソッド引数・・・) • コントロール名<タグ名>:プロパティ名 • コントロール名<タグ名>:メソッド名(メソッド引数・・・)
メソッド名	任意の場所から参照できます。 • メソッド名(メソッド引数)

※1　ローカル変数で参照できるのは、前に宣言したローカル変数のみです(下記例)。

例　local2を使用して、それより前に宣言したlocal1とargの値を参照します。

```
Method func(arg)
    local1=2
    local2=local1+arg        // 自己のローカル変数より前に宣言した
                             // local1の値を参照
    local3                   // ここで、local3=local4-2は、記述できません。
    local4=5                 // local4がlocal3より後ろに定義されている
                             // からです。
    Begin
End Method
```

■参照順位

- パッケージ内での、名前の参照順位は次のようになります。
  ただし、「パッケージ名:」を指定しない記述のとき
  1）自己パッケージのメソッドのメソッド引数、ローカル変数
  2）自己パッケージ内のプロパティ、メソッド
  3）同一パッケージに属さないメソッド
- メソッド内での名前の参照順位は、次のようになります。
  1）自己メソッド内のローカル変数名、メソッド引数名
  2）自己あるいは他のメソッド名

■名前の重複

スクリプトファイル中で名前の重複が可能なのは、以下のような場合です。

- **互いに独立関係にあるとき**
  たとえば、異なるメソッドAとBの中で、同じローカル変数名localNameは、宣言できます。

```
Method A(x1, y1, z1)
    localName
    Begin
End Method

Method B(x2, y2, z2)
    localName
    Begin
End Method
```

- **異なる階層の名前を参照するとき**
  たとえば、パッケージ名、メソッド名、メソッド引数名は、同じ名前が使えます。

  例1　name:name(name)
  nameをメソッド引数にして、nameパッケージのnameメソッドを参照します。

  例2　name(name)
  nameをメソッド引数にしてメソッドnameを呼び出します。

●重複禁止

名前の重複ができないのは、以下のような場合です。

- すべての「パッケージ名」「コントロール名」同士
  タグ指定コントロールでは、「コントロール名<タグ名>」同士
- 同一パッケージ内にあるすべての名前（プロパティ名、メソッド名、メソッド引数名、ローカル変数名同士）

予約語の例外規則

予約語の例外として、下記のルールが適応されます。

複数の予約語で1つの意味を表す語については、間の空白を省略できます。

End Method → EndMethod
End Package → EndPackage
Select Case → SelectCase
Case Else → CaseElse
End Select → EndSelect
End If → EndIf

ポイント

Else If は、1つの意味を表していません。
この空白を省略すると違う意味の予約語 ElseIf になります。

5-6 別スクリプトファイルの読み込み

別スクリプトファイルの読み込みをおこなう、Includeについて説明します。

Include(別スクリプトの読み込み)

スクリプトファイルの任意の位置に、別のスクリプトファイルを読み込みます。

参考　C言語の#includeと同じ機能です。

■書式

Include "ファイル名"

例　Include　"Lib¥mypackage"

- 任意の位置(ただし行頭)に記述できます。1行に1つのみです。
- ファイル名には、フォルダ情報を含めて記述できます。
- 指定できるファイルは、「.scp」ファイルのみです。
- 拡張子(.scp)は、記述しません。
- Includeで読み込んだスクリプトファイル内でIncludeを使用することもできます。

▶ 重要

- **最大ネスト数は50です。それを超えると、コンパイルエラーになります。**
- **Includeで読み込むスクリプトファイルには、mainメソッドは不要です。**

「mainメソッド」(5-9ページ)

5-7 別プログラムの実行

Load(プログラムの実行、起動)

Loadには、次の2つの機能があります。

- 別スクリプトプログラムの実行
  《BTシリーズ》の動作プログラムは、通常複数のスクリプトファイルで構成されます。
  Loadコントロールを使用して、別のスクリプトを実行します。

ポイント **別スクリプトファイルの任意のステートメントを呼び出すことはできません。**

- アプリケーションの起動
  スクリプトで作成した端末アプリケーション、またはC言語で作成したアプリケーションを起動します。

■書式

- 別スクリプトの実行
  System:Load(ファイル名, メソッド引数･･･)

  例 System:Load("JOB1","App",0,0,0)

  呼び出したスクリプトプログラムのmainメソッドが実行されます。

- スクリプトで作成した端末アプリケーションの起動
  System:LoadSB3(ファイル名)

  例 System:LoadSB3("K_Apl2.sb3")

  端末アプリケーション起動時に、《BTシリーズ》はリセットされません。
  呼び出し元の中間言語ファイルは、更新されません。

- C言語で作成した端末アプリケーションの起動(BT-1000/1500/600シリーズのみ)
  System:LoadAPP(ファイル名)

  例 System:LoadAPP("BT.APP")

  端末アプリケーション起動時に、《BTシリーズ》がリセットされます。
  端末アプリケーションをドライブ0にコピーしてから実行します。

▶ 重要 **LoadとLoadSB3とLoadAPPメソッドの違いを下表に示します。**

メソッド	実行時／引数	呼び出し元に戻る／戻れない	《BTシリーズ》の再起動
Load	渡せる	戻れる	しない
LoadSB3	渡せない	戻れる	しない
LoadAPP	渡せない	戻れない	する

参考 「System」コントロールのメソッドです。
「7-1 別スクリプトの呼び出し」(7-2ページ)

5-8 実行時例外／エラー

スクリプトプログラムの実行中に発生する、実行時例外と実行時エラーについて説明します。
システムが検出できる実行時例外と、システムが復旧できない実行時エラーがあります。

実行時例外

実行時例外は致命的エラーを指します。「Catch」を使って致命的エラーを検出し、システムを復旧することができます。

参 考　スタックオーバーフローは、実行時例外に含まれます。

■書式

Catch

メソッド、パッケージのメソッドの中に記述します。

▶ 重 要　**Catchは1メソッドにつき、1つしか定義できません。**

- メソッド内で実行時例外が発生したとき、メソッドを抜け出す前に後処理（Catch～End　Methodまでの処理）が実行されます。
- Catch が未定義のときに実行時例外が発生した場合は、そのメソッドを中断し、呼び出し元メソッドの例外キャッチへ戻ります。これによって、メソッドを多重に抜け出すことができます。
- 実行時例外のメソッド多重抜け出しによって、システムまで到達したときは、システムがメッセージボックスを表示します。
- 実行時例外からの復旧は、例外キャッチによって後処理され、そのメソッドを抜け出し、呼び出しメソッドに戻り、その「実行時例外」を発生したメソッド呼び出しの次のステートメントから続行されます。
- コントロールのメソッド呼び出しで、実行時例外が発生することがあります。
「7章 コントロール一覧」（7-1ページ）

例　p3が0のとき、0除算例外が発生しCatchステートメントの次のステートメントにジャンプします。例外が発生せずに、Catch ステートメントに到達したときは、Catch以降のステートメントは実行されません。

```
Method function(p1, p2, p3)
  L1
  L2
  Begin
    L1=p1/p3
    L2=p2/p3
    Return(L1+L2)
Catch                              //入力値が異常なとき、デフォルト値nilを返します。
  Return(nil)

End Method
```

実行時エラー

通常は発生しません。実行時例外の「Catch」で復旧できないエラーが発生したときに起こります。
実行時エラーが発生すると、スクリプトプログラムの先頭にジャンプします。

5-9 空白文字とコメント

空白文字とコメントについて説明します。
スクリプトファイルには、下図のように空白文字とコメントが使用できます。

```
/************************************************************
ファイル名　:App.scp
バージョン　:Ver X.X.X.X
作成日付　　:XX/XX/XXXX
動作説明　　:Keyence BT-Series アプリケーション
************************************************************/
Include "Lib\Trace"
Include "RES\App_Res"

Include "Lib\Common_Utility"
//-----------------------------------------------------
// メインメソッド
Method main()

Begin

	APP_RES:InitializeTraceLog()

	Screen:topPos=1
	Screen:coordinate = "graphic"
	APP_RES:make_file_list()

	APP_RES:set_application_action()

	While (1)
		APP:menu_main()
	Wend

// スクリプトカスタマイズをする場合は下記の2行をコメントアウトするとデバッグ効率が良くなります
Catch
	Dialog:Message(nil, "アプリケーションに問題が発生しました("&System:error&")", "ok", "", "", "middle" )
End Method


Package APP
	model="BT600"

	//-----------------------------------------------------
	// メインメソッド
	Method menu_main()
	Begin

		Screen:Clear()

		If  Handy:model.Left(6)  ne  "BT-600" Then
			Dialog:Message(nil,  "このアプリケーションは　BT-600 用です\nこの端末では実行できません","confirm", "", "", "middle")
			Handy:Reset()
		EndIf

		APP_RES:ShowMainScreen()

	End Method


EndPackage
```

複数行コメント

単行コメント

空白文字

空白文字

スクリプトプログラムの空白部分は、空白文字として解釈されます。
空白文字は、トークンの区切りとして認識されます。

参 考　トークンは、字句解析記号を指します。予約語、演算子、変数、数値、文字列などがあります。

■書式

␣ ……半角空白文字
　タブ、改行文字も同じ意味として解釈されます。

単行コメント

■書式

- **//　……半角スラッシュ2つ**
- **#　……半角シャープ**

例　AA=10　　//ここからコメントです。AA=10はコメントではありません。
#ここはコメントです。

- 記号を記述した場所から行末まで、コメントと解釈されます。

複数行コメント

■書式

```
/*  */
```

「/*」と「*/」で囲った部分がコメントとみなされます。

- 単行、または複数行にわたって、コメントとみなされます。

例　/*コメント単行*/

```
/*コメント1行目
コメント2行目
コメント3行目
*/
```

▶ 重 要　**複数行コメントの入れ子(ネスト)はできません。**

MEMO

6

コントロールの基礎知識

コントロールを使用すると、《BTシリーズ》の動作を制御できます。
ここでは、コントロールを使う上での基本的な考え方について説明します。

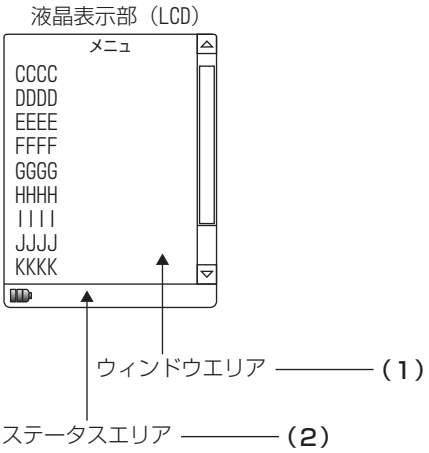
6-1　画面表示(BT-W100/W80/W70シリーズ) ····6-2
6-2　画面表示(BT-600/1000/1500シリーズ) ·· 6-12
6-3　入力処理(BT-W100/W80/W70シリーズ) ·· 6-22
6-4　入力処理(BT-600/1000/1500シリーズ) ·· 6-28
6-5　バーコードリーダ ····· 6-34
6-6　デバイス ····· 6-39
6-7　マスタファイル ····· 6-40
6-8　サーバPCとの通信 ····· 6-48
6-9　その他の通信 ····· 6-58
6-10　省電力機能 ····· 6-61

6-1 画面表示(BT-W100/W80/W70シリーズ)

液晶表示部(LCD)へ表示する方法について説明します。BT-W100/W80/W70シリーズは、GUIプログラムで開発します。

表示のしくみ（GUIプログラム）

液晶表示部(LCD)の表示領域には、下図のようにウィンドウエリアとステータスエリアがあります。

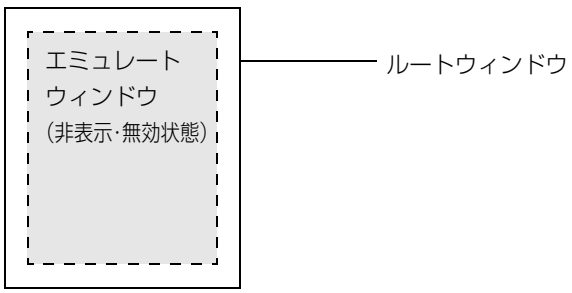


(1) ウィンドウエリア

ウィンドウエリアには、通常のタッチパネルでの表示と、BT-1000/1500/600シリーズを動作互換させるための表示(エミュレータ表示)ができます。

- 通常のウィンドウエリアへの表示は、最初から存在するルートウィンドウにWidgetコントロールを使用し、タッチパネル表示を実現しています。

- エミュレータ表示では、エミュレートウィンドウを表示･有効状態にして、Screen コントロールなどBT-1000/1500/600シリーズの互換コントロールを使用して表示画面を作成します。Widgetコントロールを使用することもできます。



詳細な説明は、「ウィンドウエリアの表示」(6-3ページ)を参照してください。

(2) ステータスエリア

バッテリ残量などを表示できます。表示／非表示、表示する位置は変更できます。
ステータスエリアの高さは、24ドットです。
詳細な説明は、「ステータスエリアの表示」(6-14ページ)を参照してください。

■ウィンドウエリアの表示

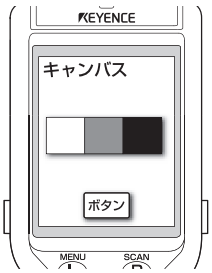
▶ 重 要

ウィンドウエリアの表示様式は、通常のタッチパネル表示と、BT-1000/1500/600シリーズを動作互換させるエミュレータ表示の2種類があります。
表示様式によって、表示に使用するコントロールが異なります。

●タッチパネル表示

タッチパネルの表示の基本は、ウィンドウ単位です。
ウィンドウにラベル、ボタンなどを配置して、ウィンドウ単位で表示します。

文字表示



図形表示

- **使用するウィンドウ**

  ルートウィンドウ: タッチパネル表示でかならず使用する基本のウィンドウで、システムに最初から存在します。
  ルートウィンドウに、ウィンドウを配置したり、直接ラベルやボタンを配置できます。

  ウィンドウ: ルートウィンドウ以外のウィンドウを使用するときに、新しく作成するウィンドウです。

  キャンバス: 主に図形描画するときに新しく作成するウィンドウです。
  使用するコントロールは、Widget 操作コントロールと Drawing コントロールです。

- **VRAM**

  VRAMは、Video Random Access Memoryの略で、画面表示のためのメモリです。
  サイズは、縦1280ドット、横240ドットです。

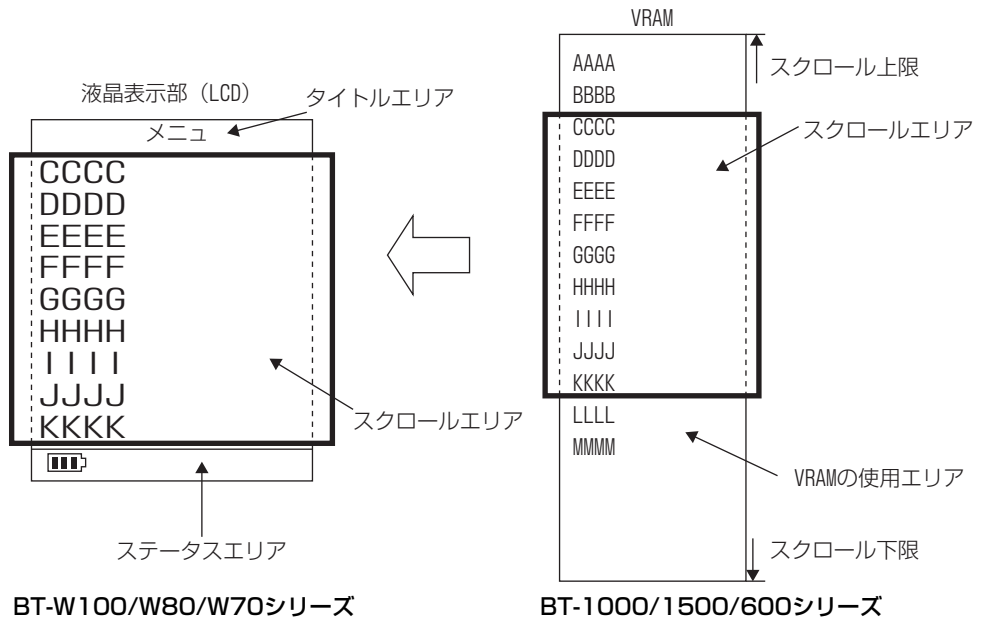
●エミュレータ表示

エミュレータ表示は、BT-1000/1500/600シリーズを動作互換させるための表示です。

- **エミュレートウィンドウ**

  エミュレートウィンドウは、最初から非表示・無効状態で存在するウィンドウです。
  エミュレータ表示するときは、Windowコントロールのプロパティ値で表示状態・有効状態にします。
  作成、削除はできません。
  タッチパネル表示で使用するときは、ウィンドウの1つとして使用できます。

  BT-1000/1500/600シリーズのスクリプトプログラムを取り込んで表示できます。液晶表示部(LCD)がBT-1000/1500/600シリーズより大きいので、下図のようにScreenコントロールで描画する文字などは拡大表示されます。
  画面が大きい比率分拡大表示します。



BT-W100/W80/W70シリーズ　　BT-1000/1500/600シリーズ

参考　画面スクロールするとき、タイトルエリア、スクロールエリアを使用します。
タイトルエリアは画面スクロールしない固定文字列を表示し、スクロールエリアには画面スクロールする文字列を表示します。
画面スクロールしないときは、エリアの区別はなく、LCDの1行目から文字列を表示できます。

- **使用するコントロール:Screen**

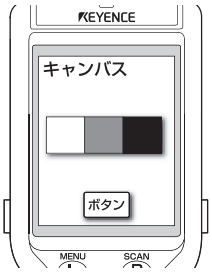
■ステータスエリアの表示

ステータスエリアに各種の情報を表示できます。
ステータスエリアの表示は、以下の設定から選択できます。

- 下表示
- 上表示
- 下表示(時計表示付き)
- 上表示(時計表示付き)
- 表示しない

文字と図形の表示ータッチパネル表示ー

文字表示



図形表示

■文字表示

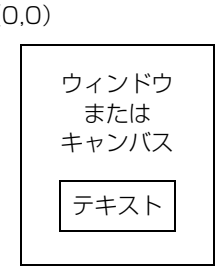
文字表示には、ウィンドウに配置する Widget ごとの文字表示と、指定キャンバス(図形描画ウィンドウ)に直接文字表示するものがあります。

- **使用するコントロール**
  Widgetごとの文字表示:各Widget操作コントロール
  キャンバスに直接表示: Canvas
  Drawing

注 記 **Canvasからはみ出る座標を指定してDrawingコントロールのメソッドを実行すると、戻り値にエラーが返ります。**

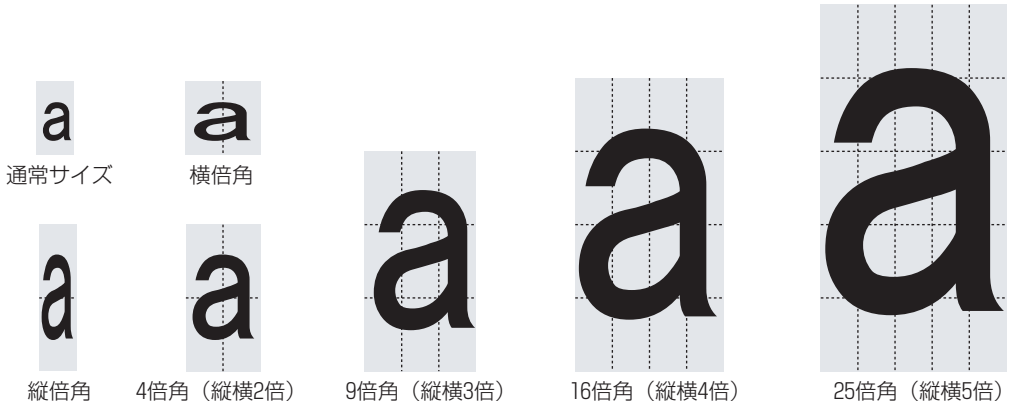
- **表示色**
  文字色と背景色は、RGBで指定できます。

- **表示位置**
  親Widgetに対する相対座標を指定します。
  親Widgetについては、「Widget操作コントロール」(5-17ページ)を参照してください。



最大X座標=ウィンドウ/キャンバスの幅-1
最大Y座標=ウィンドウ/キャンバスの高さ-1

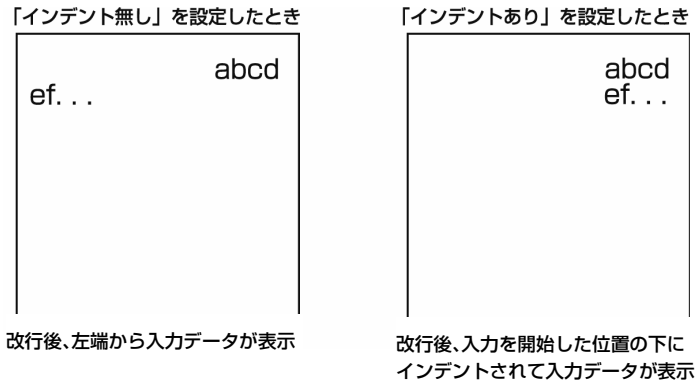
- **文字の大きさ(キャンバスに直接表示のみ)**
  Drawingコントロールを使用して指定するキャンバスにじかに表示する文字は、通常の大きさと、以下のような拡大表示が設定できます。
  横倍角、縦倍角、4倍角、9倍角、16倍角、25倍角



  さらに、太字での表示もできます。

- **文字の折り返し表示(キャンバスに直接表示のみ)**
  Drawingコントロールを使用して指定するキャンバスにじかに文字を複数行表示する場合、折り返しの設定ができます。
  インデントのあり / なしの2 種類の設定があります。どちらもキャンバス右端までデータが入力されると自動的に改行されますが、改行後のデータの表示方法が異なります。

  - **インデントなし**
    キャンバスの右端で改行後、左端からデータを表示します。
  - **インデントあり**
    キャンバスの右端で改行後、入力を開始した位置からデータを表示します。



■図形表示

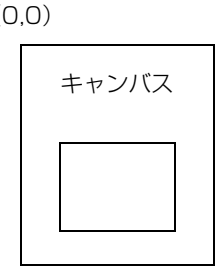
指定するキャンバス(図形描画ウィンドウ)に図形を直接描画して表示できます。

- **使用するコントロール:Canvas**
  Drawing

- **図形の種類**
  線、四角形、円弧を表示できます。

- **表示色**
  線色と図形の描画領域内の色は、RGBで指定できます。

- **表示位置**
  キャンバスに対する相対座標を指定します。



最大X座標=キャンバスの幅-1
最大Y座標=キャンバスの高さ-1

注 記

Canvasからはみ出る座標を指定してDrawingコントロールのメソッドを実行すると、戻り値にエラーが返ります。

文字と図形の表示－エミュレータ表示－

Screenコントロールを使用して、文字、線と四角形を表示できます。

■図形表示

図形を表示するには、始点と終点を座標で指定します。円弧を表示する場合には、中心・半径・開始と終了の角度を指定します。

- **使用するコントロール** :Screen
  :LCD

- **図形の種類**
  線、四角形、円弧を表示できます。

- **表示色**
  線色と図形の描画領域内の色は、RGBで指定できます。

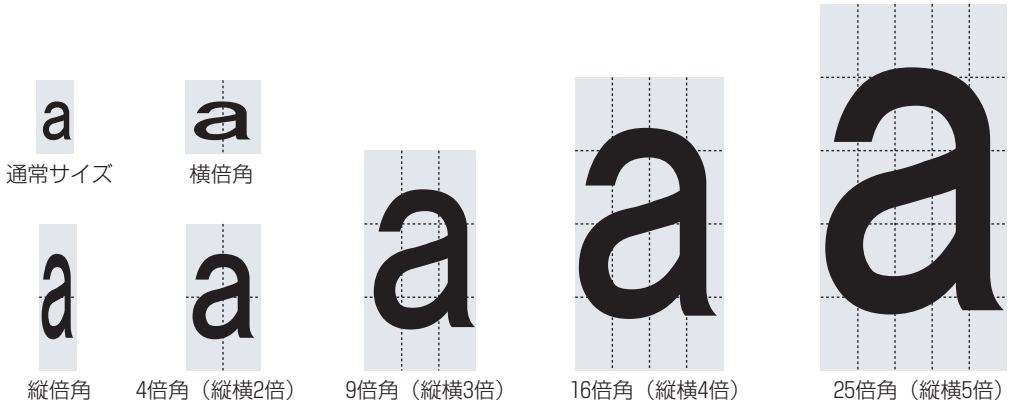
■文字表示

液晶表示部(LCD)に表示できる文字のフォントや大きさ、表示する位置の指定方法、文字間隔の調整方法について説明します。また、入力時の折り返し表示の説明をします。

●フォントと文字の大きさ

- **使用できるフォント**
  16ドットフォント(16×16)
  24ドットフォント(24×24)
  32ドットフォント(32×32)

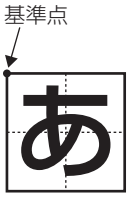
- **文字の大きさ**
  通常の大きさの文字表示と、以下のような拡大表示が設定できます。
  横倍角、縦倍角、4倍角、9倍角、16倍角、25倍角



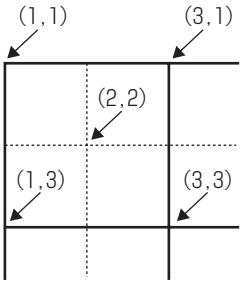
■座標

●テキスト座標

液晶表示部(LCD)に表示する文字の位置は座標で指定します。下図が指定の基準点となります。

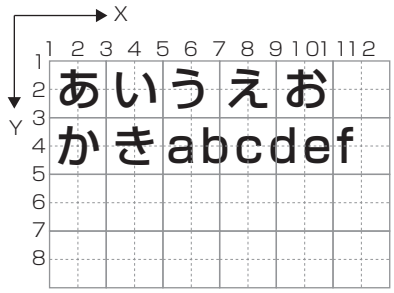


座標の原点は、下図のように(1、1)となります。



半角1文字は、X軸1座標、Y軸2座標で表示されます。
全角1文字は、X軸2座標、Y軸2座標で表示されます。

下記左図は、「あ」が(1,1)、「い」が(3,1)、「a」が(5,3)です。



12ドットフォント指定時: 縦横6ドットで1座標
16ドットフォント指定時: 縦横8ドットで1座標
24ドットフォント指定時: 縦横12ドットで1座標
32ドットフォント指定時: 縦横16ドットで1座標

参 考　文字と文字の間、行と行の間を調整する機能もあります。
「文字間ギャップ、行間ギャップ(fontGapx、fontGapyプロパティ)」(7-108ページ)

●グラフィック座標

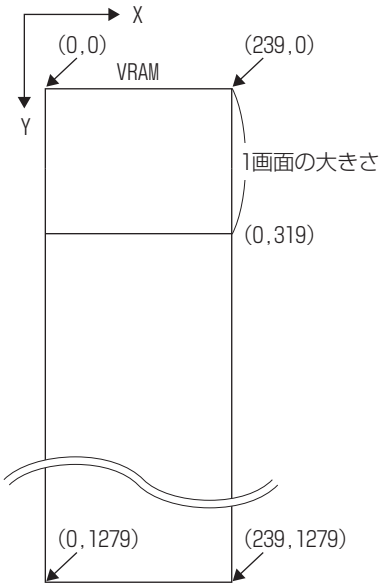
直線や四角形など、図形を描画するときに使います。

■BT-W100(QVGA)/W80/W70/1000/1500シリーズ

X座標:0～239(ドット)

Y座標:0～1279(ドット)

- 座標の原点は、画面左上座標が(0,0)です。
- X軸は右方向が正、Y軸は下方向が正です。



■BT-W100(VGA)シリーズ

X座標:0～479(ドット)

Y座標:0～2559(ドット)

- 座標の原点は、画面左上座標が(0,0)です。
- X軸は右方向が正、Y軸は下方向が正です。

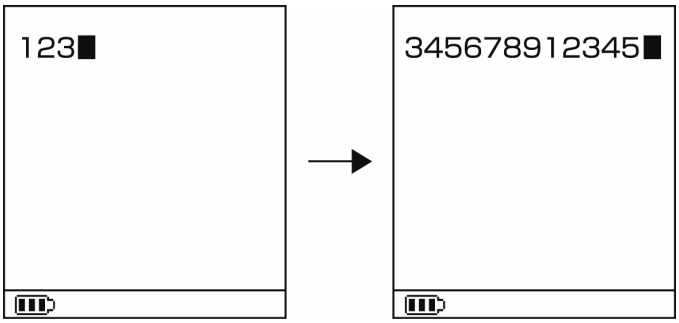
(その後、前後と同様に図を描く)

■入力データの折り返し表示

入力したデータを1行で表示できないとき、折り返して表示するように設定できます。
折り返しのあり、なしによって表示領域が異なります。

●折り返しなしのとき

入力データが表示領域よりも長くても改行されません。
下図は「12345678912345」と入力したときの例です。
入力データが1行で表示できなくなり、先頭データが左にスクロールされ表示されます。

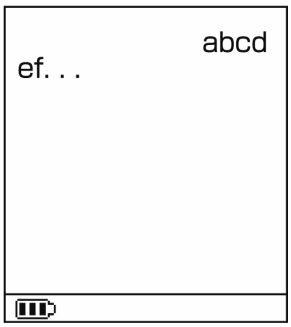


●折り返しありのとき

折り返しありには、インデントのあり / なしの2 種類の設定があります。どちらも液晶表示部(LCD)の右端までデータが入力されると自動的に改行されますが、改行後のデータの表示方法が異なります。

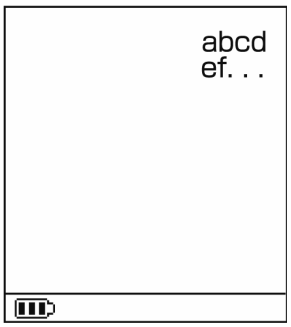
- **折り返しあり(インデントなし)**
  液晶表示部(LCD)の右端で改行後、左端からデータを表示します。
- **折り返しあり(インデントあり)**
  液晶表示部(LCD)の右端で改行後、入力を開始した位置からデータを表示します。

「インデント無し」を設定したとき



改行後、左端から入力データが表示

「インデントあり」を設定したとき



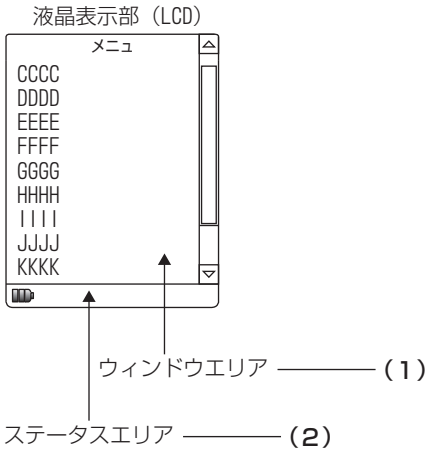
**改行後、入力を開始した位置の下に
インデントされて入力データが表示**

6-2 画面表示(BT-600/1000/1500シリーズ)

液晶表示部(LCD)へ表示する方法について説明します。BT-600/1000/1500シリーズは、GUIプログラムとCUIプログラムのどちらかで開発します。GUIとCUIを混在する開発はできません。

表示のしくみ(GUIプログラム)

液晶表示部(LCD)の表示領域には、下図のようにウィンドウエリアとステータスエリアがあります。



(1) ウィンドウエリア

- ウィンドウエリアへの表示は、最初から存在するルートウィンドウにWidgetコントロールを使用しています。

詳細な説明は、「ウィンドウエリアの表示」(6-3ページ)を参照してください。

(2) ステータスエリア

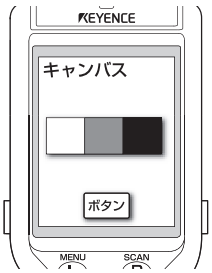
バッテリ残量などを表示できます。表示／非表示、表示する位置は変更できます。
ステータスエリアの高さは、24ドットです。
詳細な説明は、「ステータスエリアの表示」(6-14ページ)を参照してください。

■ウィンドウエリアの表示

●タッチパネル表示

タッチパネルの表示の基本は、ウィンドウ単位です。
ウィンドウにラベル、ボタンなどを配置して、ウィンドウ単位で表示します。

文字表示



図形表示

- **使用するウィンドウ**

  ルートウィンドウ: タッチパネル表示でかならず使用する基本のウィンドウで、システムに最初から存在します。
  ルートウィンドウに、ウィンドウを配置したり、直接ラベルやボタンを配置できます。

  ウィンドウ: ルートウィンドウ以外のウィンドウを使用するときに、新しく作成するウィンドウです。

  キャンバス: 主に図形描画するときに新しく作成するウィンドウです。
  図形表示はVRAMの情報を取得して、キャンバスに描画することで実現しています。
  使用するコントロールは、Widget操作コントロールとDrawingコントロールです。

- **VRAM**

  VRAMは、Video Random Access Memoryの略で、画面表示のためのメモリです。

■ステータスエリアの表示

ステータスエリアに各種の情報を表示できます。
ステータスエリアの表示は、以下の設定から選択できます。

- 下表示
- 上表示
- 下表示(時計表示付き)
- 上表示(時計表示付き)
- 表示しない

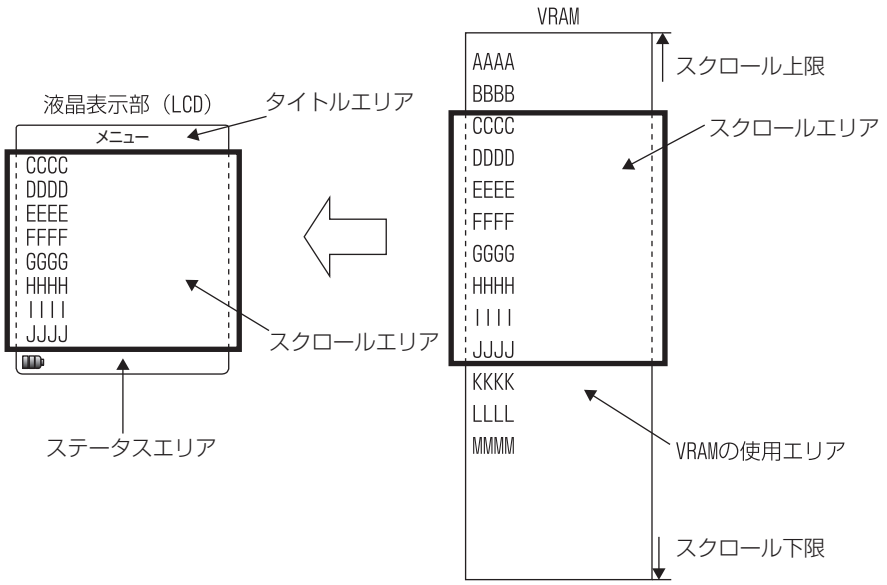
表示のしくみ(CUIプログラム)

液晶表示部(LCD)の表示は、下図のように VRAM のデータを描画することによって実現します。したがって、表示に使用するScreenコントロールのプロパティやメソッド引数の座標値は、VRAMを対象に設定します。

Screenコントロールのcoordinateプロパティにより、「テキスト座標」、「グラフィック座標」に切り替えが可能です。

「テキスト座標」(6-19ページ)
「グラフィック座標」(6-20ページ)

参 考　VRAMは、Video Random Access Memoryの略で、画面表示のためのメモリです。



次に、上図の各部の説明をします。

(1) VRAM

液晶表示部(LCD)に表示するすべてのデータを格納しています。

(2) 液晶表示部(LCD)

BTシリーズの画面です。

参 考
- 画面スクロールするとき、タイトルエリア、スクロールエリアを使用します。
  タイトルエリアは画面スクロールしない固定文字列を表示し、スクロールエリアには画面スクロールする文字列を表示します。
  画面スクロールしないときは、エリアの区別はなく、LCDの1行目から文字列を表示できます。

■ステータスエリアの表示

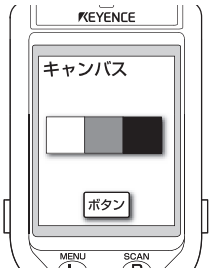
ステータスエリアに各種の情報を表示できます。

ステータスエリアの表示は、以下の設定から選択できます。

- 下表示
- 上表示
- 下表示(時計表示付き)
- 上表示(時計表示付き)
- 表示しない

文字と図形の表示(GUIプログラム)

文字表示



図形表示

■文字表示

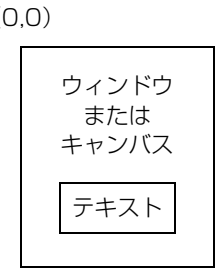
文字表示には、ウィンドウに配置する Widget ごとの文字表示と、指定キャンバス(図形描画ウィンドウ)に直接文字表示するものがあります。

- **使用するコントロール**
  Widgetごとの文字表示:各Widget操作コントロール
  キャンバスに直接表示: Canvas
  Drawing

注 記 **Canvasからはみ出る座標を指定してDrawingコントロールのメソッドを実行すると、戻り値にエラーが返ります。**

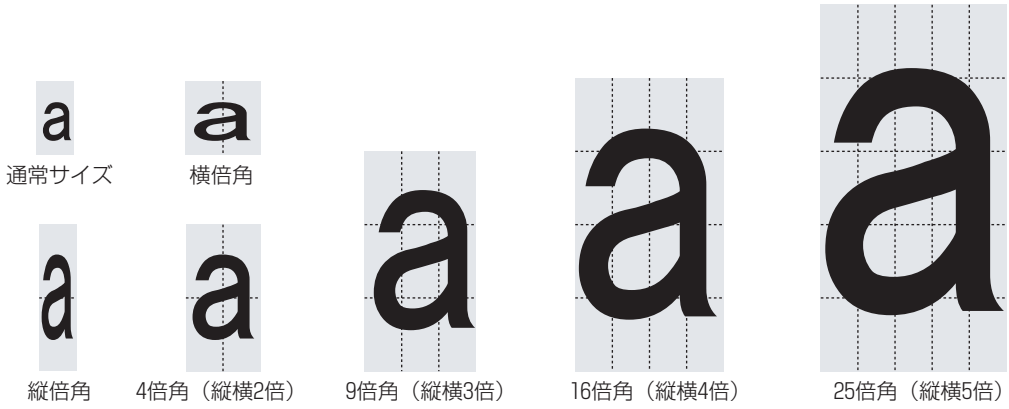
- **表示色**
  文字色と背景色は、RGBで指定できます。

- **表示位置**
  親Widgetに対する相対座標を指定します。
  親Widgetについては、「Widget操作コントロール」(5-17ページ)を参照してください。



最大X座標=ウィンドウ/キャンバスの幅-1
最大Y座標=ウィンドウ/キャンバスの高さ-1

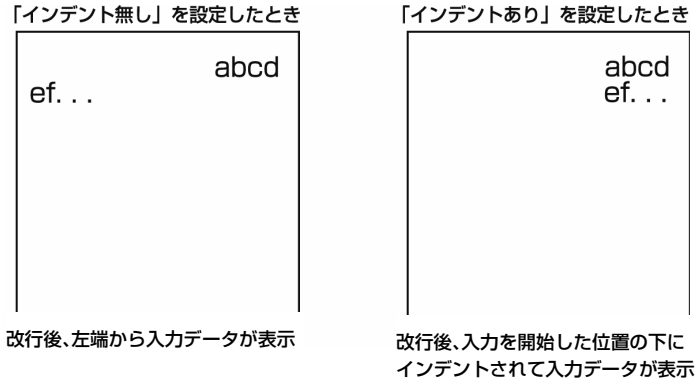
- **文字の大きさ(キャンバスに直接表示のみ)**
  Drawingコントロールを使用して指定するキャンバスにじかに表示する文字は、通常の大きさと、以下のような拡大表示が設定できます。
  横倍角、縦倍角、4倍角、9倍角、16倍角、25倍角



さらに、太字での表示もできます。

- **文字の折り返し表示(キャンバスに直接表示のみ)**
  Drawingコントロールを使用して指定するキャンバスにじかに文字を複数行表示する場合、折り返しの設定ができます。
  インデントのあり / なしの2 種類の設定があります。どちらもキャンバス右端までデータが入力されると自動的に改行されますが、改行後のデータの表示方法が異なります。

  - **インデントなし**
    キャンバスの右端で改行後、左端からデータを表示します。
  - **インデントあり**
    キャンバスの右端で改行後、入力を開始した位置からデータを表示します。



■図形表示

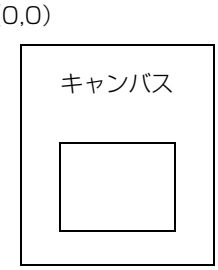
指定するキャンバス(図形描画ウィンドウ)に図形を直接描画して表示できます。

- **使用するコントロール:Canvas**
  Drawing

- **図形の種類**
  線、四角形、円弧を表示できます。

- **表示色**
  線色と図形の描画領域内の色は、RGBで指定できます。

- **表示位置**
  キャンバスに対する相対座標を指定します。



最大X座標＝キャンバスの幅-1
最大Y座標＝キャンバスの高さ-1

注 記

Canvasからはみ出る座標を指定してDrawingコントロールのメソッドを実行すると、戻り値にエラーが返ります。

文字と図形の表示(CUIプログラム)

Screenコントロールを使用して、文字、線と四角形を表示できます。

■図形表示

図形を表示するには、始点と終点を座標で指定します。円弧を表示する場合は、中心・半径・開始と終了の角度を指定します。

- **使用するコントロール:Screen**
- 図形の種類
  線、四角形、円弧を表示できます。
- **表示色**
  線色と図形の描画領域内の色は、RGBで指定できます。

■文字表示

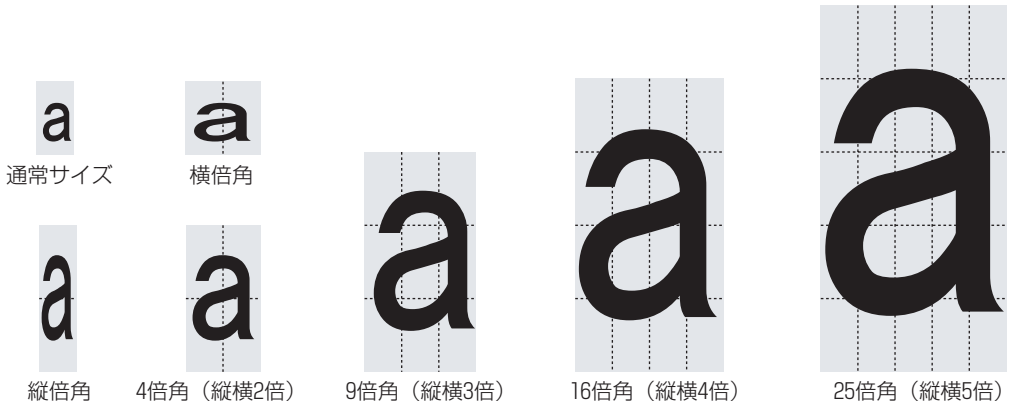
液晶表示部(LCD)に表示できる文字のフォントや大きさ、表示する位置の指定方法、文字間隔の調整方法について説明します。また、入力時の折り返し表示の説明をします。

●フォントと文字の大きさ

- **使用できるフォント**

●BT-1000/1500シリーズ	●BT-600シリーズ
16ドットフォント(16×16)	12ドットフォント(12×12)
24ドットフォント(24×24)	14ドットフォント(14×14)
32ドットフォント(32×32)	16ドットフォント(16×16)

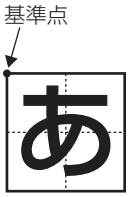
- **文字の大きさ**
  通常の大きさの文字表示と、以下のような拡大表示が設定できます。
  横倍角、縦倍角、4倍角、9倍角、16倍角、25倍角



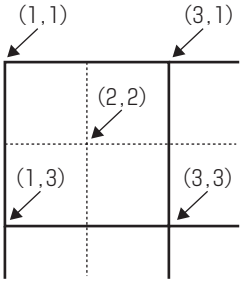
■座標

●テキスト座標

液晶表示部(LCD)に表示する文字の位置は座標で指定します。下図が指定の基準点となります。

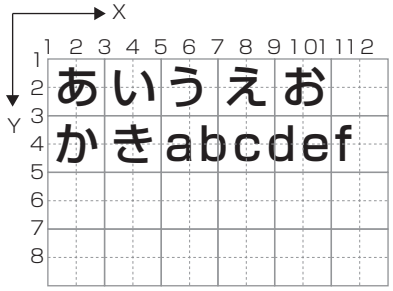


座標の原点は、下図のように(1、1)となります。



半角1文字は、X軸1座標、Y軸2座標で表示されます。
全角1文字は、X軸2座標、Y軸2座標で表示されます。

下記左図は、「あ」が(1,1)、「い」が(3,1)、「a」が(5,3)です。



12ドットフォント指定時:縦横6ドットで1座標
14ドットフォント指定時:縦横7ドットで1座標
16ドットフォント指定時:縦横8ドットで1座標
24ドットフォント指定時:縦横12ドットで1座標
32ドットフォント指定時:縦横16ドットで1座標

参 考　文字と文字の間、行と行の間を調整する機能もあります。
「文字間ギャップ、行間ギャップ(fontGapx、fontGapyプロパティ)」(7-108ページ)

●グラフィック座標

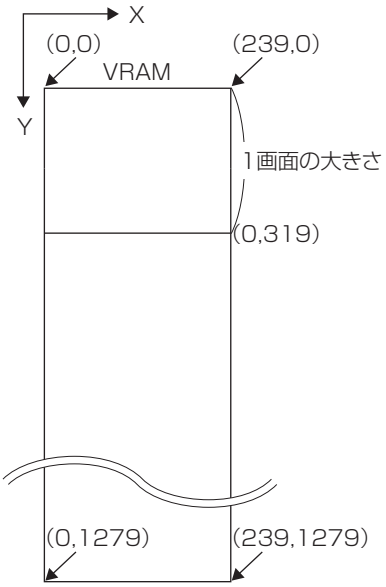
直線や四角形など、図形を描画するときに使います。

■BT-1000/1500シリーズ

X座標:0～239(ドット)

Y座標:0～1279(ドット)

- 座標の原点は、画面左上座標が(0,0)です。
- X軸は右方向が正、Y軸は下方向が正です。

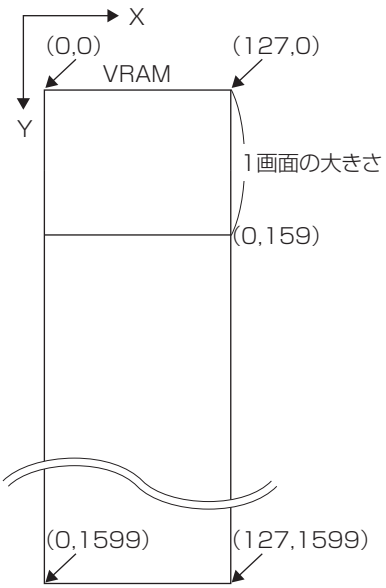


■BT-600シリーズ

X座標:0～127(ドット)

Y座標:0～159(ドット)

- 座標の原点は、画面左上座標が(0,0)です。
- X軸は右方向が正、Y軸は下方向が正です。

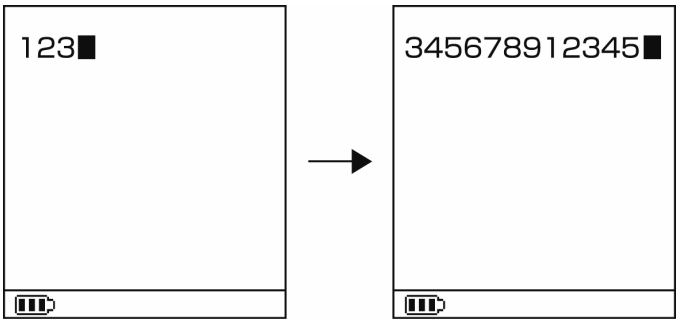


■入力データの折り返し表示

入力したデータを1行で表示できないとき、折り返して表示するように設定できます。
折り返しのあり、なしによって表示領域が異なります。

●折り返しなしのとき

入力データが表示領域よりも長くても改行されません。
下図は「12345678912345」と入力したときの例です。
入力データが1行で表示できなくなり、先頭データが左にスクロールされ表示されます。

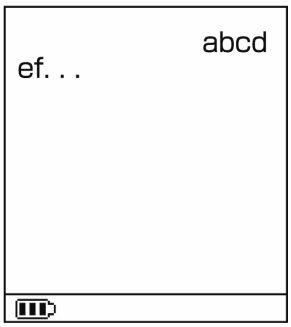


●折り返しありのとき

折り返しありには、インデントのあり/なしの2種類の設定があります。どちらも液晶表示部(LCD)の右端までデータが入力されると自動的に改行されますが、改行後のデータの表示方法が異なります。

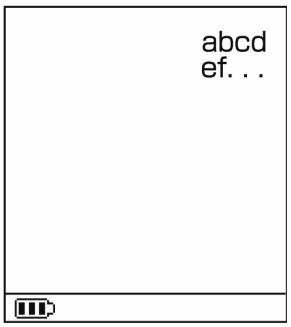
- **折り返しあり(インデントなし)**
  液晶表示部(LCD)の右端で改行後、左端からデータを表示します。
- **折り返しあり(インデントあり)**
  液晶表示部(LCD)の右端で改行後、入力を開始した位置からデータを表示します。

「インデント無し」を設定したとき



改行後、左端から入力データが表示

「インデントあり」を設定したとき



改行後、入力を開始した位置の下に
インデントされて入力データが表示

6-3 入力処理(BT-W100/W80/W70シリーズ)

バーコード入力、キー入力などの入力について説明します。
また、キー入力時の動作について説明します。

入力処理に使用するコントロールは、液晶表示部(LCD)のウィンドウエリアの表示様式によって異なります。

- タッチパネル表示の場合:Widget操作コントロールを使用します。
- エミュレータ表示の場合:エミュレータ表示専用のコントロールを使用します。

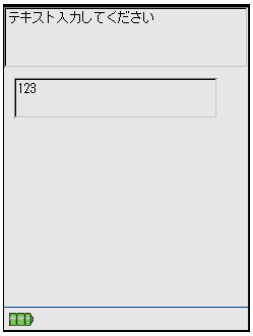
タッチパネル表示での入力処理

タッチパネル表示での入力処理は、Widget操作コントロールを使用します。

1 バーコード入力

トリガーボタンを押して、バーコードリーダから入力ができます。

- **使用するコントロール:**TextField
  TextBox



2 文字入力

文字の入力は、キー入力でおこないます。

- **使用するコントロール:**TextField
  TextBox

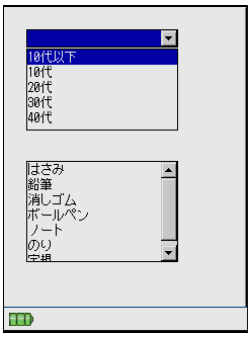
  inputOptionプロパティを使用すると、半角英数文字列または日本語(漢字変換可能)を、0～8192バイトの範囲で入力できます。

3 表示項目の選択入力

タッチパネル画面に表示されるリスト形式のデータから選択入力ができます。

- **使用するコントロール:**ComboBox
  ListBox
  CheckBox
  RadioButton

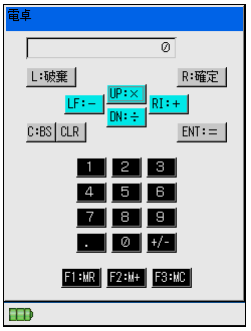
  ファイルから読み込んだデータを表示する場合は、あらかじめファイル読み込みをおこってから、上記コントロールを使用します。



4 電卓入力

電卓で計算した結果を、入力値として取りあつかうことができます。

- **使用するコントロール:**TextField
  TextBox



5 日付入力

日付(YYYY/MM/DD)の入力ができます。

- **使用するコントロール: Dialog**

参 考 上記コントロールを使用すると、入力処理プログラムを簡単に作成できます。
上記コントロールでは実現できない、より複雑な入力処理をおこなう場合には、以下のコントロールのプロパティ、メソッドを使用します。

- バーコード読み取り処理
  BCR:trigger
  BCR:data
  BCR:Input()
- キー入力処理
  Handy:KeyWait()
  Handy:KeyWaitEx()
  Handy:KeySense()

エミュレータ表示での入力処理

エミュレータ表示での入力処理は、エミュレータ表示専用のコントロールを使用します。

1 バーコード入力

バーコードリーダからの入力をおこないます。

- **使用するコントロール: InputString**

2 キー入力

キーからの入力をおこないます。文字列入力と数値入力の2種類あります。

＜文字列入力＞

- **使用するコントロール: InputString**
  shiftプロパティを使用すると、半角英数文字列を、0～8192バイト入力できます。また、useEditプロパティを使用すると、日本語(漢字変換可能)/全角、半角英数文字列を入力できます。

＜数値入力＞

- **使用するコントロール: InputDecimal**
  −2147483647～2147483647の範囲で入力できます。

3 表示項目の選択入力

あらかじめ選択マスタファイルを登録すると、キーからの入力以外に、液晶表示部(LCD)に表示されるデータを選択入力することができます。

- **使用するコントロール: InputByList**
  選択マスタファイルの作成方法については、「6-7 マスタファイル」(6-40ページ)を参照してください。

4 電卓入力

電卓で計算した結果を、入力値として取りあつかうことができます。

- **使用するコントロール: InputDecimal**

5 日付入力

日付(YYYY/MM/DD)を入力することができます。

- **使用するコントロール: InputDate**

参考 上記コントロールを使用すると、入力処理プログラムを簡単に作成できます。
上記コントロールでは実現できない、より複雑な入力処理をおこなう場合には、以下のコントロールのプロパティ、メソッドを使用します。

- バーコード読み取り処理
  BCR:trigger
  BCR:data
  BCR:Input()
- キー入力処理
  Handy:KeyWait()
  Handy:KeyWaitEx()
  Handy:KeySense()

入力確定と取り消し―タッチパネル表示、エミュレータ表示―

- バーコードリーダ以外の入力は、「ENT」キーを押すと、入力が確定し、入力データが得られます。
- 未入力状態で、「F1」キー～「F3」キー、「L」キー、「R」キーを押すと、入力がキャンセルされ、入力データは得られません。
- TextField/TextBoxコントロールのinputOptionプロパティ、またはInputStringコントロールのscanModeプロパティでバーコード読み取りを設定することによって、「トリガ」キーを押すと、バーコード読み取りができます。
  バーコード読み取り待ち状態のときは、入力取り消しはできません。
  バーコード読み取りについては、「6-5 バーコードリーダ」(6-34ページ)を参照してください。

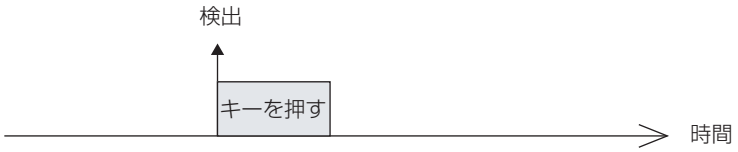
キー入力動作―タッチパネル表示、エミュレータ表示―

キーの検出、キーの割り付け、カーソル位置などのキー入力動作について説明します。

■キーの検出

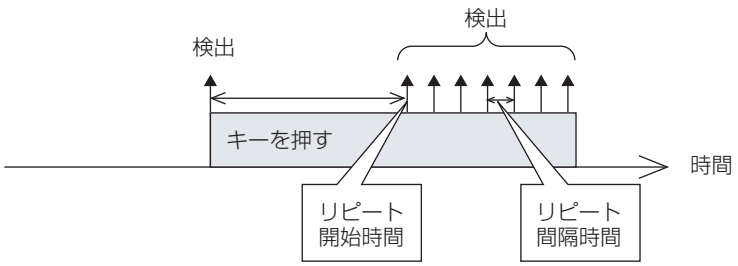
キーの検出は、キーの種類と押し方により、タイミングと回数が異なります。

- **長く押しても1度だけ検出されるキー**
  「TRG」、「ENT」、「L」、「R」、「F1」～「F3」



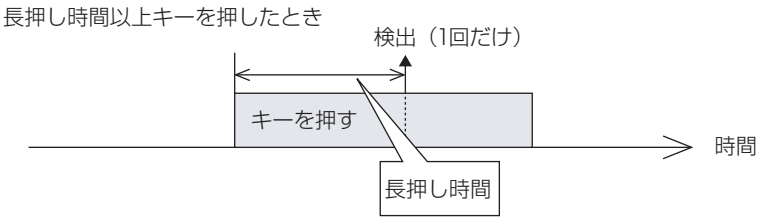
- **長く押し続けたときに繰り返し検出されるキー**
  「0」～「9」、「▲」、「▼」、「◀」、「▶」、「−」、「·」

  押したときにキー検出されます。
  キーを押し続けたとき、リピート開始時間(1s)後にキー検出されます。
  さらにキーを押し続けたとき、リピート間隔時間(100ms)ごとにキー検出されます。



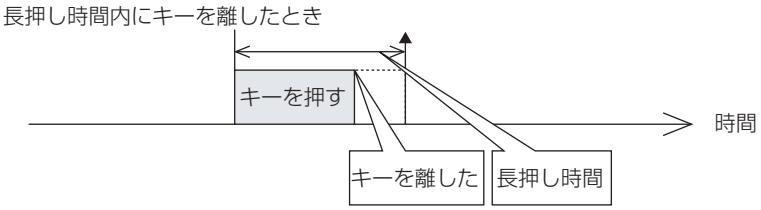
- **長く押したときのみ検出されるキー**
  「C」

  長押し時間(1ｓ)を経過したとき、1回だけキー検出されます。



- **長押し時間内に離すと検出されるキー**
  「C(BS)」

  長押し時間内に「C」キーを離すと、「BS」キーとして検出されます。



■キーの割り付け

「L」キーと「R」キーに、「トリガ」キーまたは「ENT」キーを割り付けることができます。

- **使用するコントロール:** Handy
  Key

■カーソル位置と入力動作

カーソル位置と入力動作について説明します。
カーソルと入力動作は、タッチパネル表示とエミュレータ表示で同じですが、カーソル表示が異なります。

- **カーソルの位置の移動について**
  カーソルキーを移動するには、十字キーを使用します。
  - 入力途中のカーソルの位置は、入力データの右にあります。

    タッチパネル表示の場合　　エミュレータ表示の場合

    １２３４　　１２３４

  - 「◀」キーを押すと、左に移動します。

    タッチパネル表示の場合　　エミュレータ表示の場合

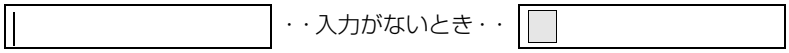
    １２３4　　１２３4

- **カーソルの移動範囲について**

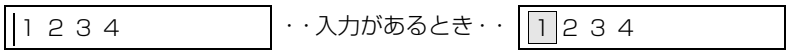
  カーソルの移動範囲は、入力データの先頭から末尾の1文字右までです。

  - カーソルの先頭位置

    タッチパネル表示の場合　　エミュレータ表示の場合

    タッチパネル表示の場合　　エミュレータ表示の場合

  - カーソルの末尾位置

    タッチパネル表示の場合　　エミュレータ表示の場合

- **カーソル位置とテンキー入力について**

  カーソルのある位置でテンキー入力をおこなうと、入力データが挿入されます。

  下記は、「3」にカーソルがあるときに、「4」を入力したときの例です。

  タッチパネル表示の場合　　エミュレータ表示の場合

- **カーソル位置と「C」キーの動作について**

  カーソルが入力データ上にあるときに「C」キーを押すと、カーソル上のデータが取り消されます。

  下記は、「3」にカーソルがあるときに、「C」キーを押したときの例です。

  タッチパネル表示の場合　　エミュレータ表示の場合

  - カーソルが入力データの右にあるときに「C」キーを押すと、BS(バックスペース)の動作をします。

    下記は、「4」の右にカーソルがあるときに、「C」キーを押したときの例です。

    タッチパネル表示の場合　　エミュレータ表示の場合

  - 「C」キーを長押しすると、クリア動作となり、入力データをすべて消去します。

    タッチパネル表示の場合　　エミュレータ表示の場合

6-4 入力処理(BT-600/1000/1500シリーズ)

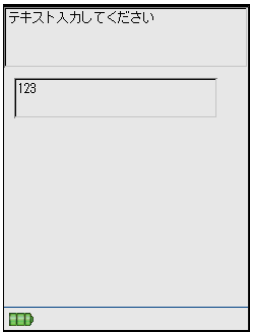
バーコード入力、キー入力などの入力について説明します。
また、キー入力時の動作について説明します。BT-600/1000/1500シリーズは、GUIプログラムとCUIプログラムのどちらかで開発します。GUIとCUIを混在する開発はできません。

GUIプログラムでの入力処理

Widget操作コントロールを使用します。

1 バーコード入力

トリガーボタンを押して、バーコードリーダから入力ができます。



- **使用するコントロール:**TextField
  TextBox

2 文字入力

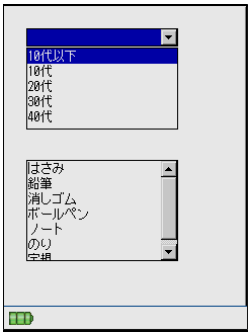
文字の入力は、キー入力でおこないます。

- **使用するコントロール:**TextField
  TextBox

  inputOptionプロパティを使用すると、半角英数文字列または日本語(漢字変換可能)を、0~8192バイトの範囲で入力できます。

3 表示項目の選択入力

画面に表示されるリスト形式のデータから選択入力ができます。

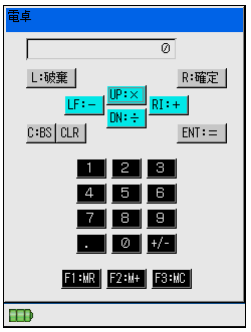


- **使用するコントロール:**ComboBox
  ListBox
  CheckBox
  RadioButton

  ファイルから読み込んだデータを表示する場合は、あらかじめファイル読み込みをおこってから、上記コントロールを使用します。

4 電卓入力

電卓で計算した結果を、入力値として取りあつかうことができます。



- **使用するコントロール:**TextField
  TextBox

5 日付入力

日付(YYYY/MM/DD)の入力ができます。

- **使用するコントロール: Dialog**

参 考 上記コントロールを使用すると、入力処理プログラムを簡単に作成できます。
上記コントロールでは実現できない、より複雑な入力処理をおこなう場合には、以下のコントロールのプロパティ、メソッドを使用します。

- バーコード読み取り処理
  BCR:trigger
  BCR:data
  BCR:Input()
- キー入力処理
  Handy:KeyWait()
  Handy:KeyWaitEx()
  Handy:KeySense()

CUIプログラムでの入力処理

1 バーコード

バーコードリーダからの入力処理をおこないます。

- **使用するコントロール: InputString**

2 キー入力

キーからの入力処理をおこないます。文字列入力と数値入力の2種類あります。

＜文字列入力＞

- **使用するコントロール: InputString**

  shiftプロパティを使用すると、半角英数文字列を、0～8192バイト入力できます。また、useEditプロパティを使用すると、日本語(漢字変換可能)/全角、半角英数文字列を入力できます。

＜数値入力＞

- **使用するコントロール: InputDecimal**

  −2147483647～2147483647の範囲で入力できます。

3 表示項目の選択入力

あらかじめ選択マスタファイルを登録すると、キーからの入力以外に、液晶表示部(LCD)に表示されるデータを選択入力することができます。

- **使用するコントロール: InputByList**

  選択マスタファイルの作成方法については、「6-7 マスタファイル」(6-40ページ)を参照してください。

4 電卓入力

電卓で計算した結果を、入力値として取りあつかうことができます。

- **使用するコントロール: InputDecimal**

5 日付入力

日付(YYYY/MM/DD)を入力することができます。

- **使用するコントロール: InputDate**

参考 上記コントロールを使用すると、入力処理プログラムを簡単に作成できます。
上記コントロールでは実現できない、より複雑な入力処理をおこなう場合には、以下のコントロールのプロパティ、メソッドを使用します。

- **バーコード読み取り処理**
  BCR:trigger
  BCR:data
  BCR:Input()
- **キー入力処理**
  Handy:KeyWait()
  Handy:KeyWaitEx()
  Handy:KeySense()

入力確定と取り消し

- バーコードリーダ以外の入力は、「ENT」キーを押すと、入力が確定し、入力データが得られます。
- 未入力状態で、「F1」キー～「F3」キー、「L」キー、「R」キーを押すと、入力がキャンセルされ、入力データは得られません。
- TextField/TextBox コントロールのinputOption プロパティ、またはInputStringコントロールのscanModeプロパティでバーコード読み取りを設定することによって、「トリガ」キーを押すと、バーコード読み取りができます。
  バーコード読み取り待ち状態のときは、入力取り消しはできません。
  バーコード読み取りについては、「6-5 バーコードリーダ」(6-34ページ)を参照してください。

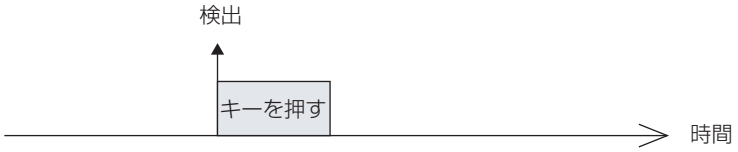
キー入力動作

キーの検出、キーの割り付け、カーソル位置などのキー入力動作について説明します。

■キーの検出

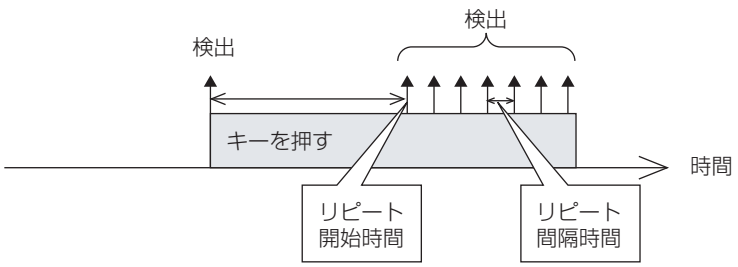
キーの検出は、キーの種類と押し方により、タイミングと回数が異なります。

- **長く押しても1度だけ検出されるキー**
  「TRG」、「ENT」、「L」、「R」、「F1」～「F3」



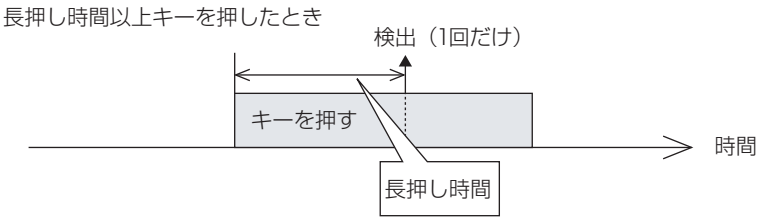
- **長く押し続けたときに繰り返し検出されるキー**
  「0」～「9」、「▲」、「▼」、「◀」、「▶」、「−」、「·」

  押したときにキー検出されます。
  キーを押し続けたとき、リピート開始時間(1s)後にキー検出されます。
  さらにキーを押し続けたとき、リピート間隔時間(100ms)ごとにキー検出されます。



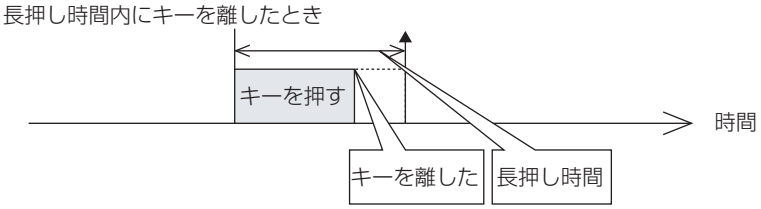
- **長く押したときのみ検出されるキー**
  「C」

  長押し時間(1s)を経過したとき、1回だけキー検出されます。



- **長押し時間内に離すと検出されるキー**
  「C(BS)」

  長押し時間内に「C」キーを離すと、「BS」キーとして検出されます。



■キーの割り付け

「L」キーと「R」キーに、「トリガ」キーまたは「ENT」キーを割り付けることができます。

- **使用するコントロール: Handy**

■カーソル位置と入力動作

カーソル位置と入力動作について説明します。

- **カーソルの位置の移動について**
  カーソルキーを移動するには、十字キーを使用します。
  - 入力途中のカーソルの位置は、入力データの右にあります。

    １２３４□
  - 「◀」キーを押すと、左に移動します。

    １２３[4]

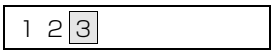
- **カーソルの移動範囲について**
  カーソルの移動範囲は、入力データの先頭から末尾の1文字右までです。
  - カーソルの先頭位置

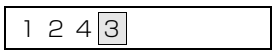
    □ ・・入力がないとき

    [1]２３４ ・・入力があるとき
  - カーソルの末尾位置

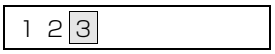
    １２３４□

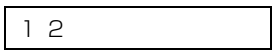
- **カーソル位置とテンキー入力について**
  カーソルのある位置でテンキー入力をおこなうと、入力データが挿入されます。
  下記は、「3」にカーソルがあるときに、「4」を入力したときの例です。



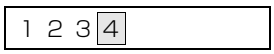


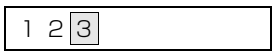
- **カーソル位置と「C」キーの動作について**
  カーソルが入力データ上にあるときに「C」キーを押すと、カーソル上のデータが取り消されます。
  下記は、「3」にカーソルがあるときに、「C」キーを押したときの例です。



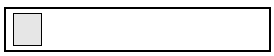


  - カーソルが入力データの右にあるときに「C」キーを押すと、BS(バックスペース)の動作をします。
    下記は、「4」の右にカーソルがあるときに、「C」キーを押したときの例です。





  - 「C」キーを長押しすると、クリア動作となり、入力データをすべて消去します。



6-5 バーコードリーダ

バーコードリーダ/2次元コードリーダ部の動作設定について説明します。

- **バーコード**
  JAN/EAN/UPC、CODE39、NW－7、CODE128、GS1-128(EAN128)、ITF、CODE93、インダストリアル2of5（TOF)、COOP、GS1-DataBar(RSS)
- **2次元コード**
  QRコード、マイクロQR、DataMatrix(ECC200)、PDF417、マイクロPDF、GSI DataBar（RSS)、合成シンボル（GS1(EAN/UCC) Composite)

読み取り動作の設定

トリガモード、読み取り回数、チェックディジットの有無など、バーコード/2次元コードリーダの動作設定ができます。

■読み取り動作の設定

コードを読み取るまでの時間、読み取り回数の設定ができます。

- **使用するコントロール: BCR**

  - **タイムアウト時間**
    設定したタイムアウト時間内に、コードの読み取りがなかった場合に、読み取りを終了できます。
    トリガモードが、オートオフ、離して読み、ダブルクリック読み、ソフトウェアトリガの場合に有効です。

  - **読み取り一致回数**
    バーコードをスキャン中、指定した回数だけ読み取れた場合に、読み取り成功と判定するように設定できます。読み取り一致回数を増やすことで、バーコード読み取りの信頼性を向上できます。

  - **二度読み防止時間**
    レーザ/光源用LED発光中に、同じコードを連続で2回以上読み取らないように、二度読み防止時間を設定します。
    トリガモードが連続読み、ソフトウェアトリガの場合に有効です。

  - **スキャンタイムアウト時間**
    レーザ / ターゲットポインタ点灯後、コードの読み取り開始するまでのタイムアウト時間を設定します。
    トリガモードが、離して読み、ダブルクリック読み、ソフトウェアトリガの場合に有効です。

  - **読み取りモード(カメラタイプのみ)**
    読み取りエリア内に複数個のバーコード / 2 次元コードがある場合に、どのように読み取るかを設定できます。

  - **表裏反転(カメラタイプのみ)**
    鏡像で印刷された2次元コードを読み取りできます。

- **白黒反転**
  白黒反転で印刷されたバーコード/2次元コードを読み取りできます。

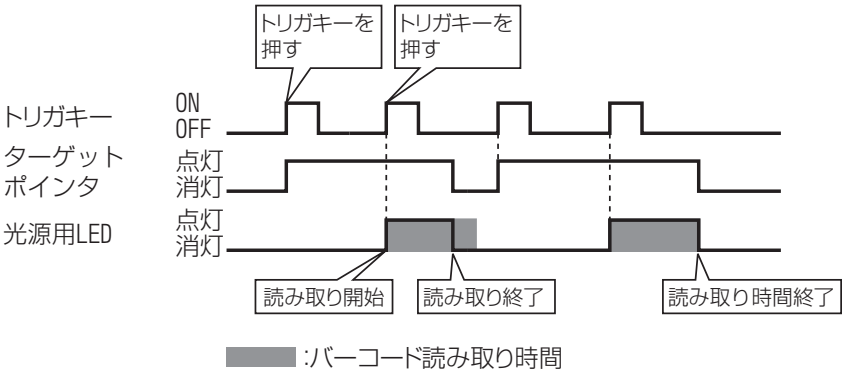
- **読み取り動作モード(カメラタイプのみ)**
  コード読み取り時にレスポンスを重視するか、読み取り精度を重視するかを設定できます。

■トリガモード

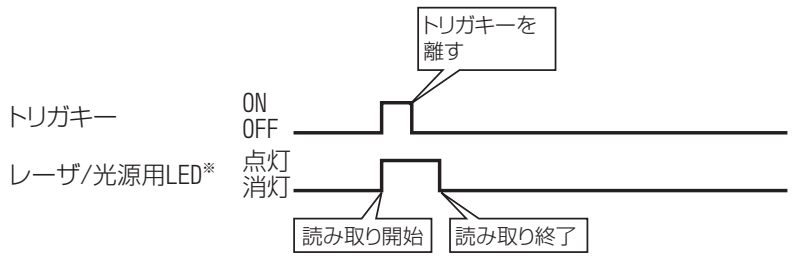
コードを読み取るときのレーザの点灯と消灯のタイミングを以下から選択できます。

- **使用するコントロール:TextField / TextBox / InputString**

- **ダブルクリック読み**
  トリガキーを押すと、ターゲットポインタが点灯します。この状態で再度トリガキーを押すと、光源用LEDが点灯して、読み取りを開始します。
  コードを読み取れたとき、またはコード読み取り時間内に読み取れなかったときには、ターゲットポインタおよび光源用LEDが消灯して、読み取りを終了します。
  また、読み取り開始後(光源用LED 点灯中)にトリガキーを押した場合にも、ターゲットポインタおよび光源用LEDが消灯して、読み取りを終了します。



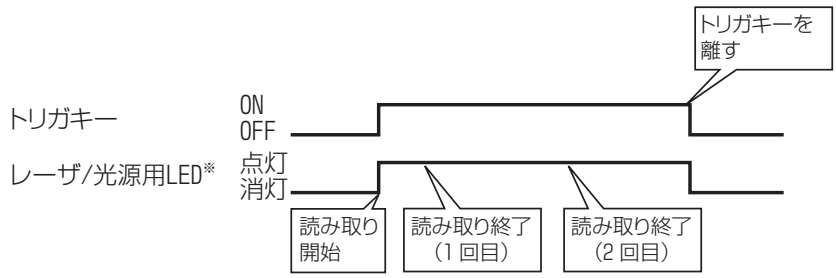
- **オートオフ**
  トリガキーを押すと、レーザ/光源用LED※が点灯して、読み取りを開始します。
  コードを読み取れたとき、またはコード読み取り時間内に読み取れなかったときには、レーザ/光源用LED※が消灯して、読み取りを終了します。
  また、読み取り開始後(レーザ点灯中)にトリガキーを押した場合にも、レーザ/光源用LED※が消灯して、読み取りを終了します。



※ ターゲットポインタも同時に点灯･消灯します。

- **連続読み**

  コードを連続読み取りするときに使用します。
  トリガキーを押している間、レーザ/光源用LED※が点灯して、読み取りを開始します。
  トリガキーを押したまま、コードを連続して読み取りできます。
  トリガキーを離すと、レーザ/光源用LED※が消灯して、読み取りを終了します。



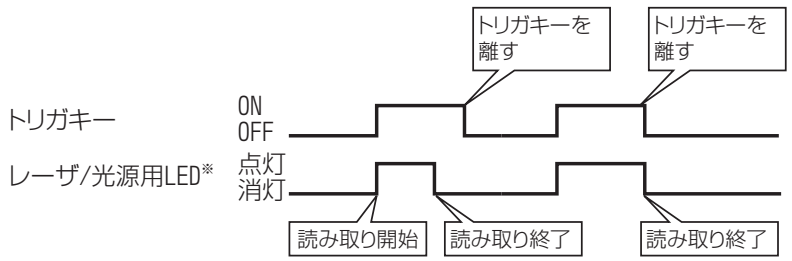
※ ターゲットポインタも同時に点灯･消灯します。

ポイント

コードを読み取る度にInputStringからは抜けてきますので、whileなどでループさせるようにプログラムを組むことで連続して読み取ることが可能になります。

- **モーメンタリ**

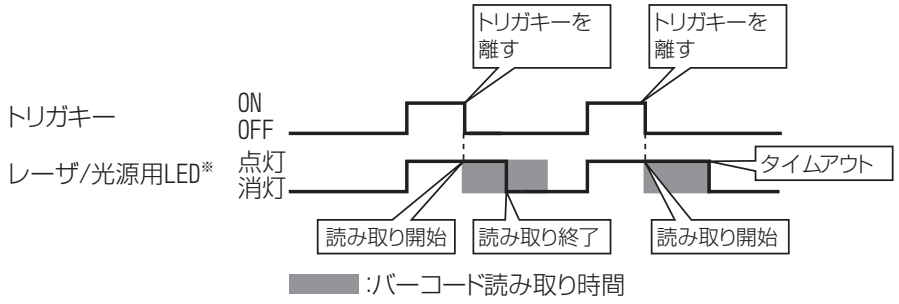
  トリガキーを押すと、レーザ/光源用LED※が点灯して読み取りを開始し、トリガキーを離すと終了します。
  コードを読み取れたときにもレーザ/光源用LED※が消灯して、読み取りを終了します。



※ ターゲットポインタも同時に点灯･消灯します。

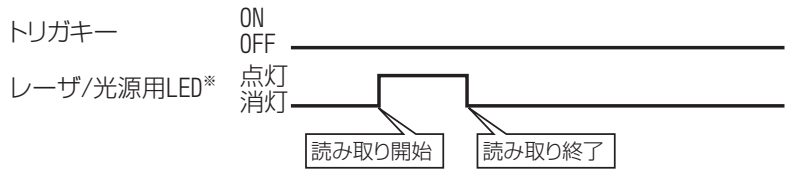
- **離して読み**

  トリガキーを押している間、レーザ / 光源用ＬＥＤ※ が点灯します。この状態では、コードの読み取りを開始しません。
  トリガキーを離すと、コードの読み取りを開始します。コードを読み取れたとき、またはコード読み取り時間内に読み取れなかったときには、レーザ/光源用LED※が消灯して、読み取りを終了します。



※ ターゲットポインタも同時に点灯･消灯します。

- **ソフトウェアトリガ**
  トリガキーを押さずに、メソッドを呼び出してレーザ/光源用LED※を点灯させ、読み取りを開始します。
  メソッドが実行されると、レーザ/光源用LED※が点灯して、コード読み取りを1回おこないます。
  コードを読み取れたとき、またはコード読み取り時間内に読み取れなかったときには、レーザ/光源用LED※が消灯して、読み取りを終了します。



※ ターゲットポインタも同時に点灯･消灯します。

■コード読み取り桁数、チェックディジット(CD)

コードに付加されるチェックディジット、読み取り桁数について説明します。

コード種別	付加されるCD	読み取り桁数
CODE39	モジュラス43	3～50
CODE128/ GS1-128(EAN128)	モジュラス103	1～50
JAN/EAN/UPC	モジュラス10/3ウェイト	8、または13
ITF	モジュラス10/3ウェイト	2～50
NW-7	モジュラス16、 モジュラス11、 モジュラス10/2ウェイト、 モジュラス10/3ウェイト、 7チェックDR、 加重モジュラス11、 ルーンズ 上記、7種類から選択できます。	3～50
CODE93	モジュラス47	1～50
インダストリアル2of5	—	1～50
COOP	—	1～50
QR code	—	1～7089
DataMatrix	—	1～3116
MaxiCode	—	1～138
PDF417	—	1～2710
GS1 DataBar(RSS)	—	1～74※
合成シンボル	—	1～74

※ GS1DataBar(RSS)については、それぞれのコードの桁数の制限に従います。

▶ 重 要

上記はソフトウェア上の制限であり、この範囲の桁数すべての読み取りを保証するものではありません。必ず使用するバーコード/2次元コードで読み取りテストを実施してください。

バーコード合成読み取り機能(レーザータイプのみ)

通常、レーザ式バーコードリーダの場合は、バーコードの端から端までレーザ光が横切っていないと読み取れません。「バーコード合成読み取り機能」は、下図のようにバーコードが斜めになっていても、全体をスキャンすることで、スキャンしたバーコードデータを合成して読み取ります。

- **使用するコントロール: BCR**

バーコード合成読み取り機能を使用する上で、以下のような制限があります。

- **対象となるコード種**
  JAN、CODE128(GS1-128(EAN-128))、CODE39、NW-7、ITF(チェックデジット有り)、CODE93の中から1種類のみを読み取り対象としてください。

- **読み取り桁数の制限**
  読み取り最大桁・最小桁の設定を同一桁にすること。JANの場合も同様に、13桁のみ、または8桁のみと、同一桁のみ読み取るような設定が必要です。

- **読み取る角度について**
  バーコードの長さの半分以上をレーザが横切るような角度にしてください。

ポイント

- **バーコード合成読み取り機能を使用する場合は、導入前に使用するバーコードを使って十分なテストをしてください。**
- **GS1 DataBar(RSS)は対象外です。**
- **CODE128、GS1-128は、どちらも読める設定にする必要があります。**

6-6 デバイス

LED、バイブレータ、ブザーなどのデバイスの制御について説明します。

使用目的

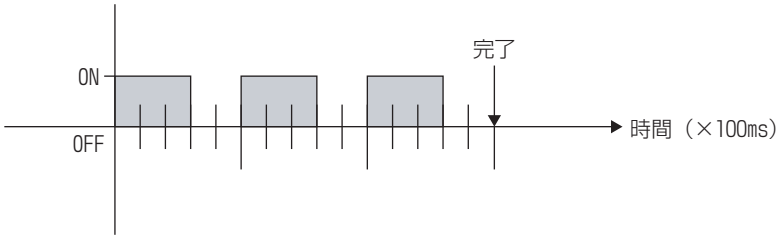
バーコード読み取り、またはキー入力されたデータが正しいかエラーがあるかの確認は、動作確認LED、ブザー音、バイブレータでおこなうことができます。

- **使用するコントロール：Led** …… 動作確認LEDの制御
  Buzzer …… ブザー音の制御
  Vibration …… バイブレータ機能の制御

動作設定

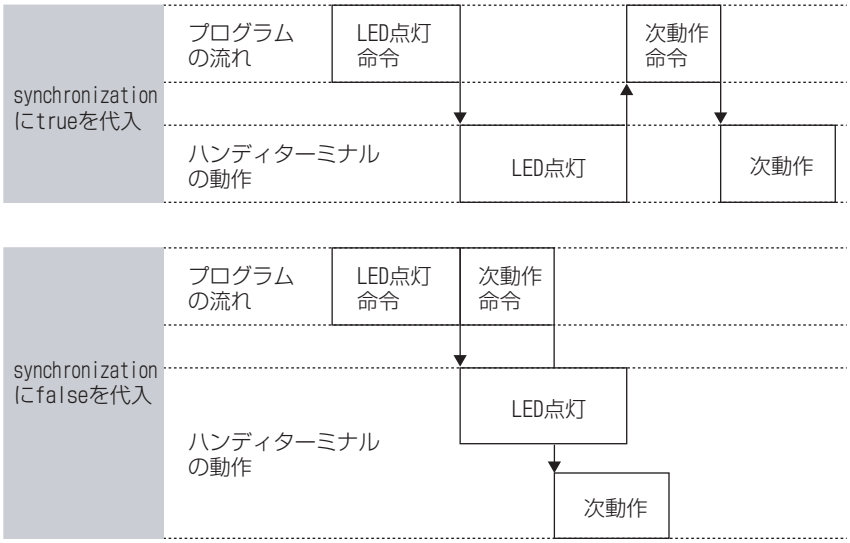
Led、Buzzer、Vibrationはいずれも、動作オン時間、動作オフ時間、動作オンと動作オフの繰り返し回数、同期／非同期などの詳細設定ができます。
以下に、Ledを使用した動作設定について説明します。

例1　点灯時間300ms、消灯時間200ms、繰り返し回数3回のとき、(300+200)×3=1.5sで処理が完了します。



例2　**同期／非同期**

下図のように、同期を設定するとLEDの実行動作が完了してから次動作に移りますが、非同期を設定すると完了を待たずに次動作に移ります。



同期はsynchronization＝true、　非同期はsynchronization＝falseで設定します。

6-7 マスタファイル

マスタファイルは、読み取ったバーコード、キー入力したデータを、照合、または他のデータに置き換えるときに使用します。

マスタファイルの種類

マスタファイルには、以下の3種類があります。

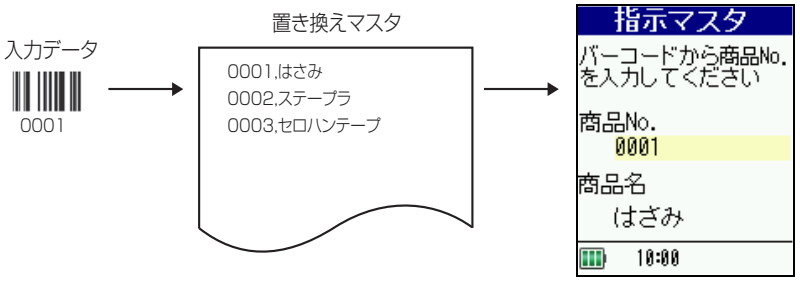
- **置き換えマスタ**
  使用するコントロール: Search

- **指示マスタ**
  使用するコントロール: Master

- **選択マスタ**
  使用するコントロール: ListBox/ComboBox/InputByList

■置き換えマスタ

読み取ったバーコード、またはキー入力したデータを、他の文字列に置き換えるときに使用します。
たとえば、下図では、入力は数値でおこない、画面には対応する商品名を表示しています。



■指示マスタ

「順序指定あり」と「順序指定なし」の2つの使用方法があります。

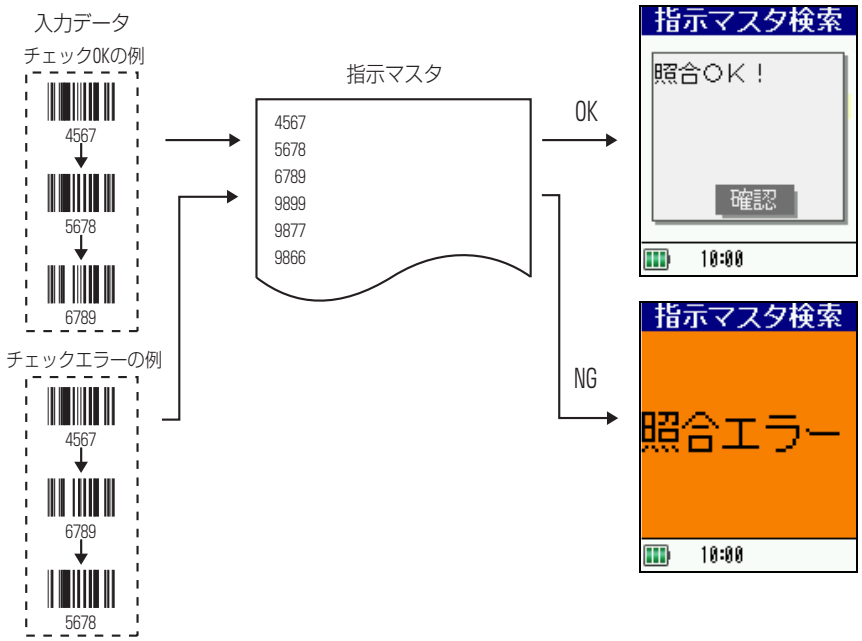
●順序指定あり

バーコードリーダとキーから入力したデータが、指示マスタに登録した順序どおりかを照合します。

例　ピッキング作業

1 **商品コードをピッキングする順番に指示マスタに登録しておきます。**

2 **指示マスタに登録した順番に商品Noを読み取り、ピッキング作業をおこないます。**



●順序指定なし

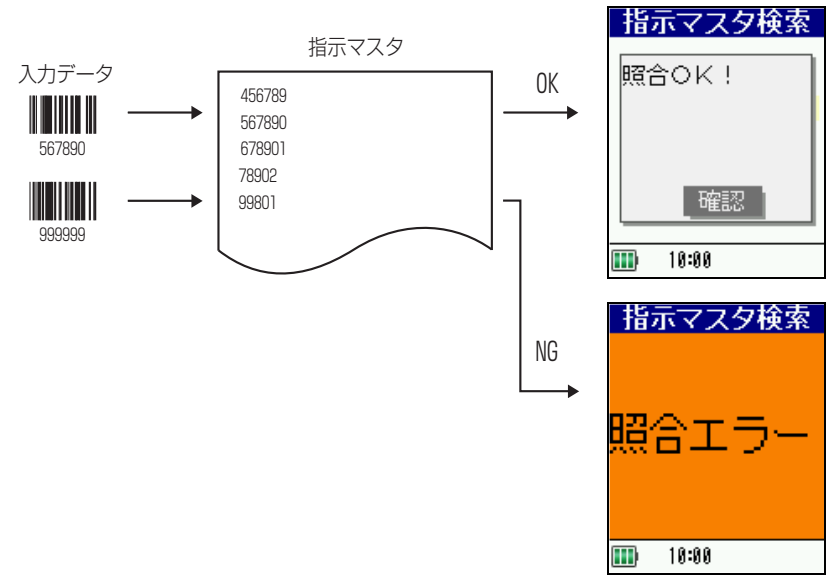
読み取ったバーコード、またはキー入力したデータが、指示マスタに存在するかを照合します。

例　検品作業

1 **検品する商品の商品Noを指示マスタに登録しておきます。**

2 **検品する商品の商品Noを読み取ります。**

3 **読み取られた商品Noが、指示マスタに含まれているか照合されます。**



■選択マスタ

マスタファイルに登録したデータを画面に表示し、それらを選択するときに使用します。

画面には、選択マスタのデータが表示されます。「▲」キーと「▼」キーでデータを選択します。

例　データ3つ目の「20代」を選択すると、入力データとして「003」が取得できます。

ファイル構成

マスタファイルを作成するための、ファイル名の付け方、ファイルの構成、レコードフォーマットなどの基本的な仕組みについて説明します。

■ファイル名の付け方

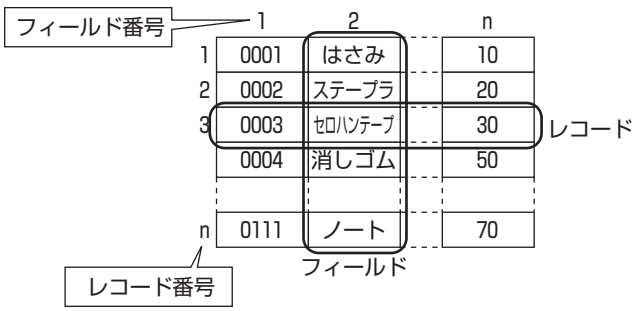
ファイル名は、拡張子を含め32文字以内の制限があります。
置き換えマスタのみ、拡張子は「.txt」「.csv」のいずれかを選択します。指示マスタ、選択マスタでは、拡張子に制限はありません。

- **指示マスタ、選択マスタ**
  ファイル名は、拡張子を含め32文字以内で設定してください。

- **置き換えマスタ**
  拡張子が「.txt」または「.csv」のいずれかを選択します。
  ファイル名は、拡張子を除いて28文字以内で設定してください。

■ファイルの構成

ファイルは下図のように、フィールド、レコードから構成されます。



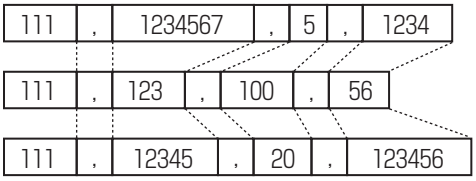
- フィールドは、「商品No」、「数量」などの項目に使用します。
  同一フィールドには、同一の型のデータが格納されます。

- レコードは、複数のフィールドから構成されます。
  可変長レコードと固定長レコードがあります。

■可変長形式と固定長形式(置き換えマスタ、指示マスタ)

置き換えマスタ、指示マスタのデータ形式には、可変長形式と固定長形式があります。

- **可変長形式**
  各フィールドのデータ長が不定で、各データはセパレータ(区切り文字)で区切られています。メモリを効率的に利用できます。



- **固定長形式**
  各フィールドのデータ長は固定で、セパレータは使用しません。セパレータを検索する必要がないので高速検索ができます。

111	1234567	005	001234
111	0000123	100	000056
111	0012345	020	123456

置き換えマスタ

置き換えマスタの作成方法、BTシリーズへの転送方法と、フォーマット(可変長形式／固定長形式)について説明します。

■作成方法と転送方法

ファイル作成は、「置き換えマスタ」と「インデックス情報定義ファイル」の2種類が必要です。

作成した置き換えマスタは、「データ転送ソフト」を使用して、BTシリーズへ転送できます。インデックス情報定義ファイルは、転送する必要はありません。

参考 インデックス情報ファイルのフォーマットについては、「付録4 置き換えマスタ」(付-6ページ)を参照してください。

- ハンディ上で置き換えマスタを作成することも可能です。「7 章 FileSystem コントロール」「付録4 置き換えマスタ」(付-6ページ)を参照してください。

■可変長形式のフォーマット

単行、または複数行のレコードから構成されます。各レコードのフォーマットを、下図に示します。

▶ 重 要

1データ最大255バイトです。

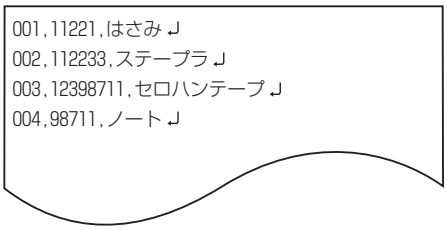
データ1	セパレータ	データ2		データn	改行コード

- **データ1～データn(nは10まで)**
  データは、1バイト以上の文字列(半角、全角)で記述します。

- **セパレータ**
  データとデータの区切り文字です。カンマ、タブ、またはスペースで指定します。

- **改行コード**
  改行コード(ASCIIで0x0D0A)を指定します。

以下に、可変長形式の置き換えマスタの例を示します。セパレータにカンマを使用しています。



▶ 重 要

データに全角文字が含まれる場合、0x01～0x3F(シフトJIS文字コード)のセパレータをご使用ください。シフトJIS の規格上, 全角文字とセパレータの文字コードが重複する場合があります。

■固定長形式のフォーマット

単行、または複数行のレコードから構成されます。
各レコードのフォーマットを、下図に示します。

▶ 重 要

1データ最大255バイトです。

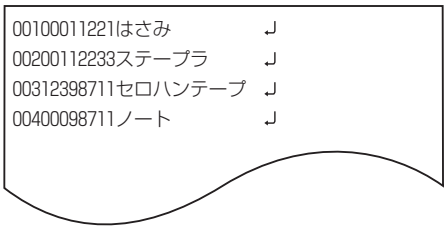
データ1	データ2		データn	改行コード

- **データ1～データn(nは10まで)**
  データは、固定長形式で1バイト以上の文字列(半角、全角)を記述します。

  固定長形式の記述方法は、スクリプトの記述ルールに従って自由に指定できます。
  たとえば、「アプリケーション簡単作成ツール」で作成したときは、次のようになります。
  記述例:"ABCDE"値をデータ桁数10桁に設定するとき
  "ABCDE " ・・・・・文字列の後ろにスペースを詰めます

- **改行コード**
  改行コード(ASCIIで0x0D0A)を指定します。
  以下に、固定長形式の置き換えマスタの例を示します。

  例　データ1の桁数3(数値)、データ2の桁数8(数値)、データ3の桁数16文字列)に設定しています。



■インデックス情報定義ファイル

インデックス情報定義ファイルは、置き変えマスタが高速に検索を可能にするための情報を記録したファイルです。

- ファイル名:　置き換えマスタと同名のファイル名+".idx"

インデックス情報定義ファイルについては、「付録4　置き換えマスタ」(付-6ページ)を参照してください。

指示マスタ

指示マスタの作成方法、BT シリーズへの転送方法と、フォーマット(可変長形式／固定長形式)について説明します。

■作成方法と転送方法

指示マスタは、テキストエディタなどで作成します。

作成した指示マスタは、「データ転送ソフト」を使ってBTシリーズへ転送できます。

■フォーマット

指示マスタのフォーマットは、置き換えマスタと同じです。

▶ 重 要　**指示マスタの1レコードは最大8192バイト、最大レコード件数はBT-1000/1500シリーズは32767件、BT-W100/W80/W70/600シリーズは160000件です。**

- 可変長レコードのフォーマットについては、「可変長形式のフォーマット」(6-45ページ)を参照してください。
- 固定長レコードのフォーマットについては、「固定長形式のフォーマット」(6-45ページ)を参照してください。

選択マスタ

選択マスタの作成方法、BTシリーズへの転送方法と、フォーマットについて説明します。

■作成方法と転送方法

選択マスタは、テキストエディタなどで作成します。
作成した選択マスタは、「データ転送ソフト」を使ってBTシリーズへ転送できます。

■フォーマット

選択マスタは、可変長形式のみです。

▶ 重 要 **選択マスタの1レコードは最大256バイトです。**

単行、または複数行のレコードから構成されます。
各レコードのフォーマットを、下記に示します。

選択項目	セパレータ	確定時の値	改行コード

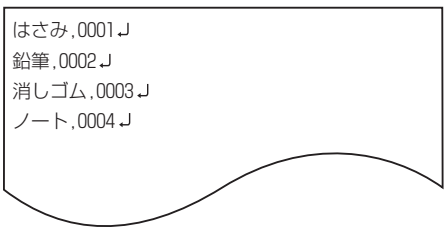
- **選択項目**
  選択入力するときに画面表示される内容です。文字列で記述します。

- **セパレータ**
  選択項目と確定時の値の区切り文字です。カンマを記述します。

- **確定時の値**
  画面で選択したとき取得できるデータ内容です。文字列で記述します。

- **改行コード**
  改行コード(ASCIIで0x0D0A)を指定します。

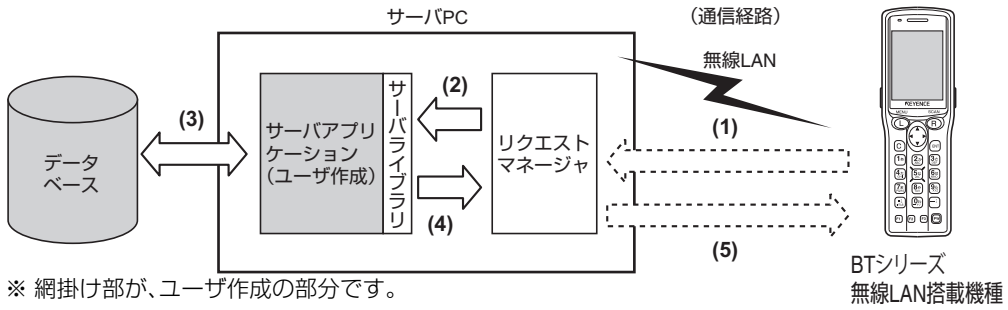
以下に、選択マスタの例を示します。
たとえば、1行目の「はさみ」を選択すると、入力データとしては「0001」が取得できます。



6-8 サーバPCとの通信

BTシリーズとサーバPC間を無線で通信するときに使用するシステムについて説明します。

リクエストマネージャを使用したシステム



※ 網掛け部が、ユーザ作成の部分です。

「リクエストマネージャ(ミドルウェア)」を使用して、BTシリーズ無線LAN搭載機種(以下BT)とサーバアプリケーションの間でデータ送受信、ファイル送受信をおこなうシステムです。

「リクエストマネージャ(ミドルウェア)」は、BT との無線通信の制御をおこないますので、ユーザが作成する「サーバアプリケーション」では、「リクエストマネージャ」からの要求に対して必要な処理(データベースの検索など)を実行して、実行結果を「リクエストマネージャ」に応答するだけでよいため、アプリケーション開発工数を大幅に削減することができます。

また、実際の運用に近いサンプルプログラムとして、「ベースアプリケーション」を用意しています。「ベースアプリケーション」のソースファイルも用意していますので、それをもとに必要な機能をカスタマイズして、「サーバアプリケーション」を開発してください。

ユーザが作成するサーバアプリケーションでは、以下のような処理を実行する必要があります。詳細については、『通信ライブラリリファレンス(リアルタイムデータ通信編)』を参照してください。

(1) BTから、「リクエストマネージャ」 に対して、データ (製品の在庫数など) を要求します。
(2) 「リクエストマネージャ」は、その「要求」が送信されたことを「サーバアプリケーション(ユーザ作成)」に通知します。
(3) 「サーバアプリケーション」は、その「要求」に対して、必要な処理(データベースの検索など)をおこないます。
(4) (3)で取得した情報を「リクエストマネージャ」に返します。
(5) 「リクエストマネージャ」 は、BTに対して、その情報を返します。

▶ 重 要

- **ベースアプリケーションは運用に近い形のサンプルプログラムです。システム運用の際には、必ずユーザで作成したサーバアプリケーションをご使用ください。**
- **1 サーバあたりの BT の通信可能台数は最大 100 台です。ただし、通信頻度、通信量、使用環境により変動します。**
- **リクエストマネージャを使った通信は、必ずBT側から開始されます。サーバPCから一方的にBTへデータを送信することはできません。**
- **リクエストマネージャを使用する場合、ネットワークドライブを利用した運用は動作サポート外となります。**

●必要なソフトウェア

リクエストマネージャを使用するシステムでは、以下のソフトウェアが必要です。

- **リクエストマネージャ**
  BTの端末アプリケーションとサーバアプリケーションの間で、データやファイルの送受信を仲介するソフトウェアです。
  サーバアプリケーションから呼び出して使用します。リクエストマネージャ単体では使用できません。
  リクエストマネージャの詳細については、『通信ライブラリリファレンス(リアルタイムデータ通信編)』を参照してください。

- **サーバアプリケーション**
  リクエストマネージャと通信をおこない、サーバ上のデータやファイルなどをBTと送受信します。
  サーバアプリケーションは、「サーバライブラリ」を使用して開発します。サーバアプリケーションの詳細については、『通信ライブラリリファレンス(リアルタイムデータ通信編)』を参照してください。

- **サーバライブラリ**
  リクエストマネージャとサーバアプリケーションのインターフェースとなるライブラリです。
  DLL、ActiveX、.NET用のライブラリが用意されています。

●使用するコントロール

リクエストマネージャとの通信に使用するコントロールには、以下のものがあります。

- RemoteAccess
  無線通信ポートのオープン/クローズ、端末アプリケーションの更新などを実行します。

- RemoteDB
  サーバ上のデータベースのデータ検索、削除、更新などを、リクエストマネージャへ要求します。

- RemoteFile
  サーバ上のファイルの読み込み、書き込みを、リクエストマネージャへ要求します。

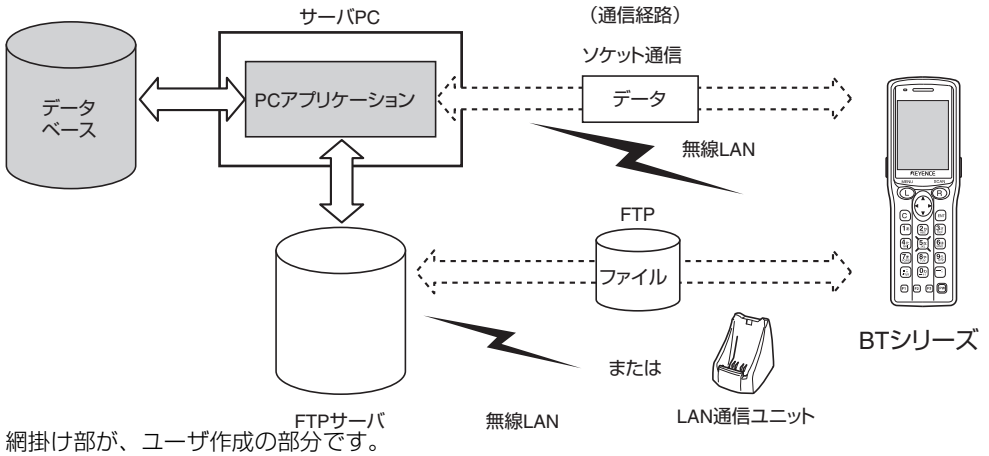
- RemoteUD
  あらかじめメッセージを書き込みしたファイルを、リクエストマネージャを経由して送受信します。

参考 BT-900用通信ライブラリで作成したPCアプリケーションと通信するメソッド、プロパティも用意しています。

- **使用するコントロール:CommRF**

CommRFコントロールを使用する場合、「RemoteDB」、「RemoteFile」、「RemoteUD」を使用して、リクエストマネージャと通信することはできません。

TCP/IPのソケット通信、またはFTPを使用したシステム



※ 網掛け部が、ユーザ作成の部分です。

- 「ソケット通信」を使用したデータ送受信、「FTP(File Transfer Protocol)」を使用したファイル送受信などTCP/IPの汎用プロトコルを使用したシステムです。
  汎用プロトコルであるため、OS環境に依存されないシステム構築が可能です。Linuxサーバなどでの運用が可能です。
- 通信経路は、無線LAN経由(BTシリーズ無線LAN搭載機種のみ)、およびLAN通信ユニット経由のいずれかを使用することができます。

●使用するコントロール

- Socket
  ソケット通信を使用して、データ送受信をおこないます。
- FTP
  FTPを使用して、ファイル送受信をおこないます。

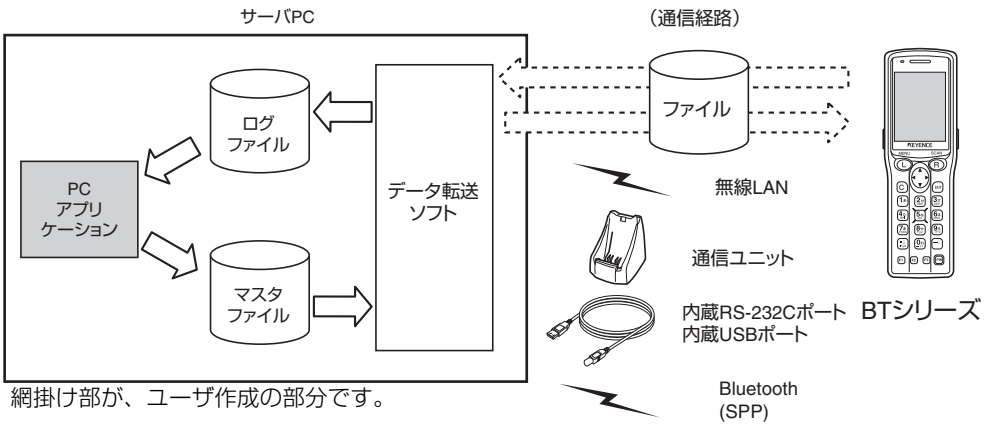
▶ 重 要

- **BT-1000/1500 シリーズでは全銀 TCP/IP 手順での通信も可能です。事前に当社営業担当にお問い合わせください。**
- **「ソケット通信」、「FTP」を使用した PC アプリケーションの開発環境については、それぞれの通信に対応した開発ツールを別途ご用意ください。**

参 考

- 「ソケット通信」、「FTP」で使用するポート番号は、端末アプリケーション /PC アプリケーションで自由に設定できます。
- データ送受信には「ソケット通信」、ファイル送受信には「通信ライブラリ」を使用して通信するシステムを構築することもできます。
  「通信ライブラリ」を使用した通信については、「通信ライブラリを使用したシステム」(6-52ページ)を参照してください。

データ転送ソフトを使用したシステム



※ 網掛け部が、ユーザ作成の部分です。

- 「データ転送ソフト」を使用して、ファイル送受信（キーエンス手順）をおこなうシステムです。
  「データ転送ソフト」の操作方法については、『アプリケーション開発マニュアル』を参照してください。
- 通信経路は、無線 LAN 経由（BT シリーズ無線 LAN 搭載機種のみ）、通信ユニット経由、内蔵 RS-232C/USBポート経由（BT-1000/1500シリーズのみ）、Bluetooth（SPP）（Bluetooth搭載機種のみ）経由のいずれかで通信可能です。

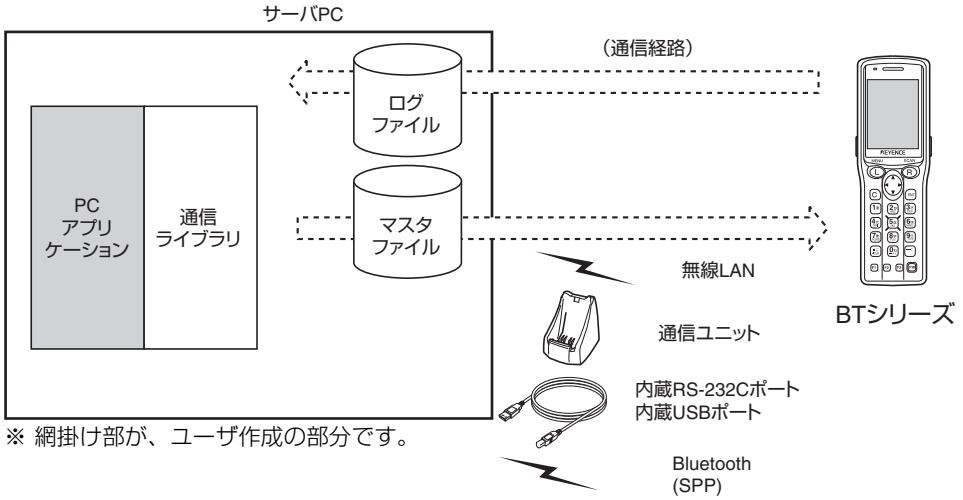
●使用するコントロール

- Comm
  無線、通信ユニット、内蔵RS-232C/USBポート経由でのファイル送受信をおこないます。

▶ 重 要

- **Bluetoothドングルを使用してPCに接続する場合、接続できるハンディは1台です。**
- **「データ転送ソフト」を使用する場合、BTシリーズからのアクションでファイル送受信をおこなうようなシステムをおすすめします。**
- **「データ転送ソフト」を使用する場合、ネットワークドライブを利用した運用は動作サポート外となります。**

通信ライブラリを使用したシステム



※ 網掛け部が、ユーザ作成の部分です。

- 「通信ライブラリ」を使用して開発した PC アプリケーションを使用し、ファイル送受信（キーエンス手順）をおこなうシステムです。「通信ライブラリ」を使った開発方法については、『通信ライブラリリファレンス』を参照してください。
- 通信経路は、無線LAN経由（BTシリーズ無線LAN搭載機種のみ）、通信ユニット経由、内蔵RS-232C/USBポート経由（BT-1000/1500シリーズのみ）、Bluetooth（SPP）経由（Bluetooth搭載機種のみ）のいずれかで通信可能です。

●使用するコントロール

- Comm

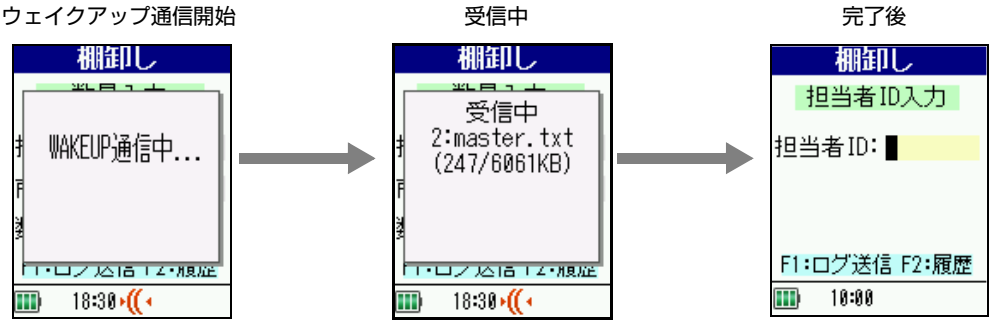
  無線、通信ユニット、内蔵RS-232C/USBポート経由でのファイル送受信をおこないます。

▶ 重 要

- **Bluetoothドングルを使用してPCに接続する場合、接続できるハンディは1台です。**
- **「通信ライブラリ」を使用して開発したPCアプリケーションで運用する場合、BTシリーズからのアクションでファイル送受信をおこなうようなシステムをおすすめします。**
- **「通信ライブラリ」を使用する場合、ネットワークドライブを利用した運用は動作サポート外となります。**

■ウェイクアップ機能(BT-600/1000/1500シリーズのみ)

ウェイクアップ機能とは、「データ転送ソフト」または「通信ライブラリ」を使用したPCアプリケーションから通信を開始すると、BTシリーズが強制的に通信待機状態になり、ファイル送受信をおこなう機能です。BTシリーズが電源OFFの状態でも、強制的に電源をONして通信を開始します。



●使用するコントロール

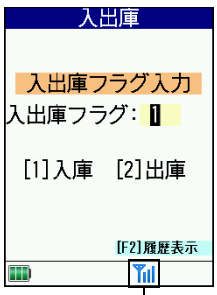
- Handy

▶ 重 要

- ウェイクアップ機能を実行してファイル送受信が終了すると、BTシリーズは再起動状態(アプリケーション先頭から開始する状態)になるため、それまでの操作が無効になります。運用で使用する際は十分にご注意ください。
- ウェイクアップ機能で送受信できるのは、ファイルのみです。

ポイント

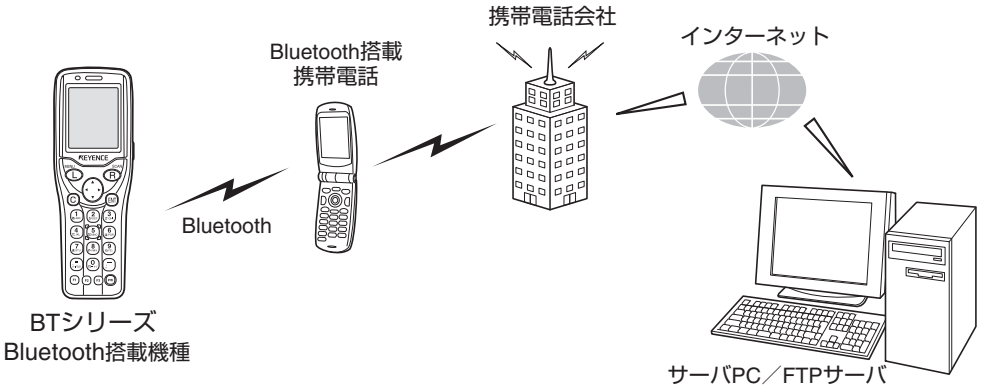
- BTシリーズ無線LAN搭載機種では、以下の状態のときに、無線を使用したウェイクアップ機能が有効になります。
  - 充電ユニット / 通信ユニットにセットした状態(充電 LED 点灯時)
  - 無線をオープンしている状態(ステータス表示にアンテナマークが表示されます)
- 充電ユニット / 通信ユニットにセットしていない状態でも、無線を使用したウェイクアップ機能を有効にするようにカスタマイズ可能です。この場合、BTシリーズを使用していないときにも、電池が消費されます。
  - 「リクエストマネージャ」「ソケット /FTP」を使ったシステム、内蔵RS-232C/USB、Bluetooth、モデム経由での通信では、ウェイクアップ機能は使用できません。



無線がオープン状態のときに表示されます。

Bluetooth搭載携帯電話を利用した遠隔地通信システム

BTシリーズからBluetooth搭載携帯電話を経由してインターネット接続し、遠隔地にあるサーバと通信するシステムです。
インターネット接続後は、FTPサーバ、またはサーバPC内の「データ転送ソフト」とファイル送受信が可能です。FTPS（FTP over SSL/TLS）にも対応しています。



- BTは、Bluetooth搭載携帯電話経由で、携帯電話会社（またはインターネットサービスプロバイダ）のインターネット接続サービス／モバイル通信サービスを使いインターネットに接続します。
- BTからインターネット経由でサーバと通信するためには、サーバにグローバルIP（固定IP）を割り当て、インターネット上に公開することが必要です。契約するインターネットサービスプロバイダーが提供する固定IPサービスをご利用ください。
- サーバPC側でFTPサーバ、または、当社通信ライブラリ、「データ転送ソフト」で受信したファイルを処理するPCアプリケーションの開発が必要です。
- Bluetooth搭載で、DUN-GW（Dial-up Networking Profile Gateway）対応の携帯電話をご使用ください。

●使用するコントロール

- Bluetooth
- PPP
- Comm
- FTP

アナログモデムを利用した遠隔地通信システム

《BT-1000/1500シリーズ》の内蔵RS-232Cポート、またはBT-UC10R/BT-UC6Rに接続したBT専用アナログモデムBT-MD1を使用し、遠隔地にあるサーバとファイル通信するシステムです。
通信方法として、以下の2つがあります。

■ アナログ公衆電話回線経由で通信先アナログモデムに接続する方法(シリアル接続)

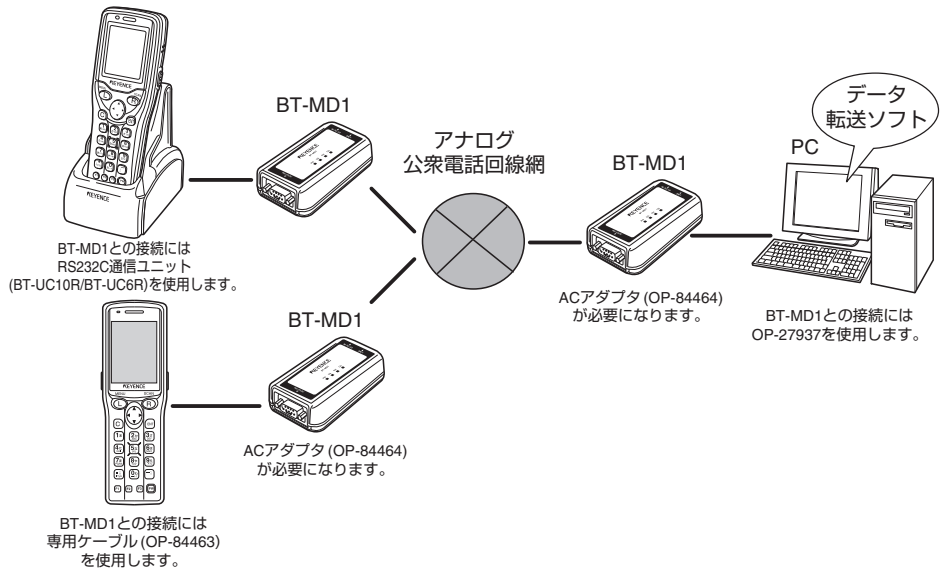
アナログ公衆電話回線経由で通信先のアナログモデムに接続し、接続先PC 内のデータ転送ソフトまたは通信ライブラリと通信します。
以下のような手順になります。

1 **《BT-1000/1500シリーズ》は、内蔵 RS-232C、または BT-UC10R に接続した BT 専用アナログモ デムBT-MD1に対して、《BT-600シリーズ》は、BT-UC6Rに接続したBT専用アナログモデムBT-MD1に対して、モデム用コマンド(ATコマンド)を送信し、通信先モデムの電話番号にダイヤルします。**
※《BT-1000/1500/600シリーズ》は、ATコマンドを意識しないプログラミングが可能です。

2 **通信先モデムが着信すると、その後、双方のモデムが接続を開始します。**

3 **接続が完了すると、《BT-1000/1500/600シリーズ》からキーエンス手順にてファイル送受信を実行します。**
※ PC 側は、データ転送ソフトまたは通信ライブラリを使って作成した PC アプリケーションが、ファイル待ち受け受信状態になっている必要があります。

4 **ファイル送受信が完了したら、通話を切断します。**



●使用するコントロール

- Network
- Comm

ポイント

- NTTまたは直収型電話サービス事業者が提供するアナログ公衆電話回線が必要です。
- 各機器の接続方法については、『BT 専用アナログモデム BT-MD1 ユーザーズマニュアル』をお読みください。
- PC側のアナログモデムは市販のものが使用できます。
- PC側でRASサーバを使用する場合、PC側モデムにBT-MD1は対応していません。

- BT-UC10R/BT-UC6RとBT-MD1をBT-UC10R/BT-UC6R付属のRS-232Cケーブルで接続した場合、BT-MD1の電源はBT-UC10R/BT-UC6Rから供給されるため、専用ACアダプタ(OP-84464)は不要です。
- 「データ転送ソフト」や「通信ライブラリ」を使用する場合、ネットワークドライブを利用した運用は動作サポート外となります。

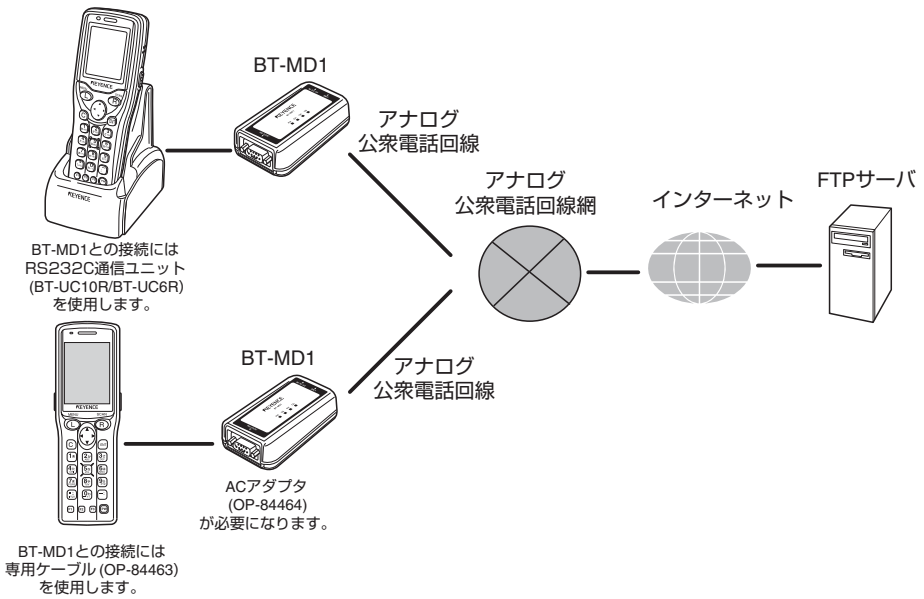
■インターネットに接続する方法(PPP接続)

アナログ公衆電話回線経由でインターネットにダイヤルアップ接続(インターネットサービスプロバイダにPPP接続)し、FTPサーバなどにファイル通信します。
以下のような手順になります。

1 《BT-1000/1500シリーズ》は、内蔵RS-232C、またはBT-UC10Rに接続したBT専用アナログモ デムBT-MD1経由で、《BT-600シリーズ》は、BT-UC6Rに接続したBT専用アナログモデムBT-MD1経由で、インターネットにダイヤルアップ接続(PPP)します。

2 インターネット接続後は、《BT-1000/1500/600シリーズ》からFTPなどでファイル送受信をおこないます。
《BT-1000/1500/600シリーズ》からインターネット経由でサーバと通信するためには、サーバにグローバルIP(固定IP)を割り当て、インターネット上に公開する必要があります。
契約するインターネットサービスプロバイダが提供する固定IPサービスをご利用ください。

3 ファイル送受信が完了したら、通話を切断します。



●使用するコントロール

- Network
- PPP
- Comm
- FTP

ポイント

- BT-MD1 は、NTT または直収型電話サービス事業者が提供するアナログ公衆電話回線に接続します。
- 各機器の接続方法については、『BT 専用アナログモデム BT-MD1 ユーザーズマニュアル』をお読みください。
- 《BT-1000/1500シリーズ》の場合、全銀TCP/IP手順での通信も可能です。事前に当社営業担当にお問い合わせください。
- データ転送ソフトまたは通信ライブラリを使って作成した PC アプリケーションでの運用も可能です。
- BT-UC10R/BT-UC6RとBT-MD1をBT-UC10R/BT-UC6R付属のRS-232Cケーブルで接続した場合、BT-MD1の電源はBT-UC10R/BT-UC6Rから供給されるため、専用ACアダプタ(OP-84464)は不要です。

6-9 その他の通信

プリンタとの通信、RS-232C/USBポートを使用した通信、BTシリーズどうしでの通信について説明します。

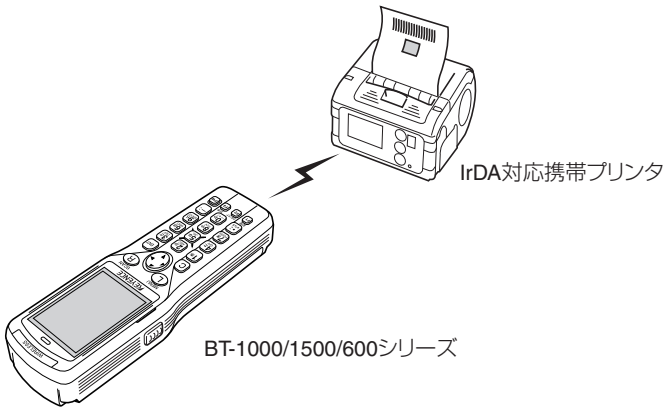
プリンタとの通信

無線、または赤外線通信（IrDA）ポート、RS-232Cポートから、プリンタへ印刷コマンドを送信できます。

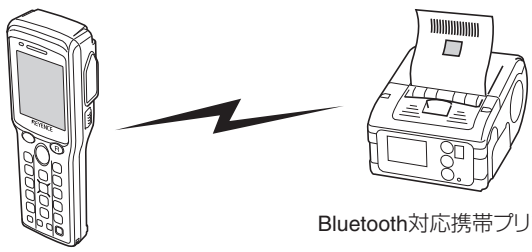
●使用するコントロール

- Printer

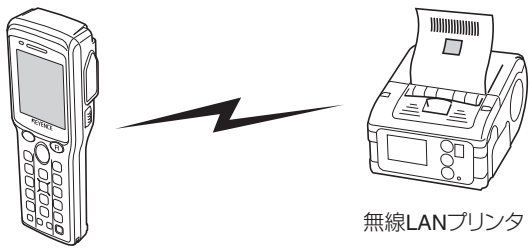
例1　赤外線通信（IrDA）ポートからIrDA対応携帯プリンタへコマンドを送信している例です。



例2　Bluetoothを使用してBluetooth対応携帯プリンタへコマンドを送信している例です。

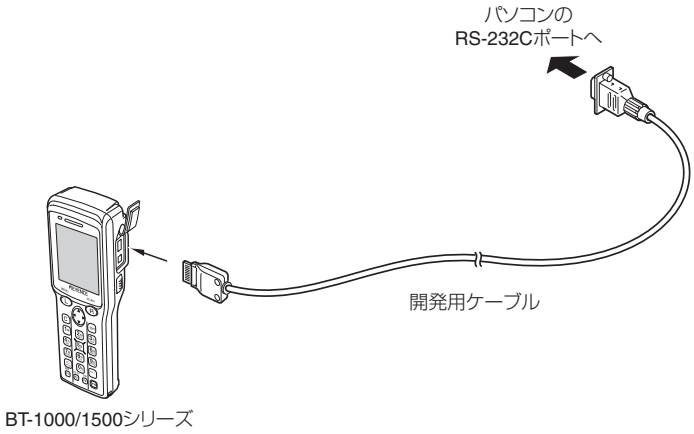


例3　無線LANを使用して無線LANプリンタへコマンドを送信している例です。



RS-232Cポートを使用した通信

RS-232Cポートを使ってパソコンとデータ通信ができます。
《BT-1000/1500シリーズ》のRS-232Cポートとパソコンの RS-232Cポートを、開発用ケーブルで接続します。

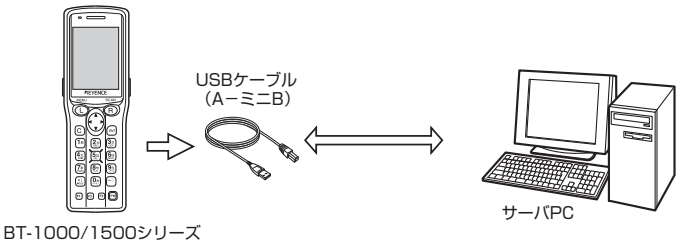


●使用するコントロール

- Serial
- Comm2

USBポートを使用した通信

USBポートを使ってパソコンとデータ通信ができます。
《BT-1000/1500シリーズ》のUSBポートとパソコンのUSBポートを、USBケーブルで接続します。



●使用するコントロール

- Serial

BTシリーズ間の通信

■赤外線通信(IrDA)ポートを使用した通信(BT-1000/1500/600シリーズ)

赤外線通信(ＩｒＤＡ)ポートを使って、BTシリーズどうし、または赤外線通信(IrDA)ポートを持つプリンタとデータ通信ができます。

下図のように、赤外線通信(IrDA)ポートを向かい合わせます。

BT-1000/1500/600シリーズ　BT-1000/1500/600シリーズ

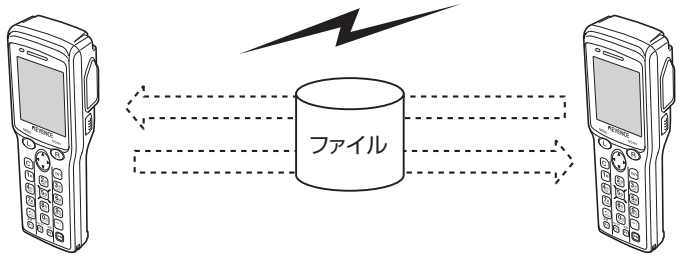
また、IrDAを持つ外部機器とのデータ通信もできます。

●使用するコントロール

- Serial
- Comm1

■無線を使用した通信(《BTシリーズ無線LAN搭載機種》)

無線を使用して、《BT-1000/1500 シリーズ無線LAN搭載機種》どうしでデータ･ファイル通信ができます。



BTシリーズ無線LAN搭載機種　BTシリーズ無線LAN搭載機種

●使用するコントロール

- Comm
- CommRF
- WLAN
- Network

6-10 省電力機能

BTシリーズの省電力機能について説明します。

オートパワーオフ機能

設定した時間内に入力がなかったときに、自動的にパワーオフにする機能です。

●使用するコントロール

- Handy

バックライトのオートオフと明るさ調整

液晶表示部(LCD)のバックライトは、オートオフ機能と明るさ調整機能があります。

- **オートオフ機能**
  設定した時間内に入力がなかったときに、自動的にバックライトを消灯します。
- **明るさ調整機能**
  バックライトの明るさを、4段階で設定できます。明度を下げると、バッテリーの節約になります。

●使用するコントロール

- LCD

MEMO

7

コントロール一覧

ハンディターミナルに動作を命令するコントロールについて説明します。

各コントロールのサンプルは、BTサポートページにアップされています。
BTサポートページは、BT-Navigatorランチャーの設定メニューにリンクされています。

7-1　別スクリプトの呼び出し ……7-2
7-2　Widget操作 ……7-5
7-3　イベントの取得 …… 7-81
7-4　データの入力 …… 7-85
7-5　液晶表示部(LCD)の設定 ……7-103
7-6　読み取り動作の設定 ……7-133
7-7　デバイスの動作設定 ……7-157
7-8　ファイルの操作 ……7-160
7-9　マスタファイルとの照合 ……7-190
7-10　ログファイルの操作 ……7-203
7-11　高速検索 ……7-211
7-12　サーバとのデータ送受信 ……7-214
7-13　サーバPCやプリンタとの通信 ……7-240
7-14　ユーティリティ ……7-308
7-15　デバイス設定、メニュー表示、キー入力 ……7-315

7-1 別スクリプトの呼び出し

実行中のスクリプトから別スクリプトを呼び出しできます。

System	別スクリプトを呼び出します。また、端末アプリケーションの切り替えをおこないます。

プロパティ

名称	説明	範囲	初期値	代入
arg1	別スクリプトから渡されたパラメータの第1引数	任意	—	不可
arg2	別スクリプトから渡されたパラメータの第2引数		—	不可
arg3	別スクリプトから渡されたパラメータの第3引数		—	不可
arg4	別スクリプトから渡されたパラメータの第4引数		—	不可
error	実行時例外コード	実行時例外が発生したときにエラー内容を示すエラーコードが格納されます。	0	不可
errorInf1	実行時例外コードの詳細情報 呼び出したスクリプト内でエラーが発生したときに、エラー情報が格納されます。	関数エラー番号	0	不可
errorInf2		エラー情報文字列	""	不可
errorInf3		未使用	0	不可
stack_res	スタック領域の残りサイズ※1	0～524288 (512KB)	端末アプリケーションに依存	不可
heap_res	ヒープ領域の残りサイズ※2	0～229376 (224KB)	端末アプリケーションに依存	不可

※1 端末アプリケーション起動時に確保される領域から、コード領域と静的変数領域を差し引いた残りのサイズです。
コード領域＋静的変数領域＋stack_res＝512KB

※2 文字列データ("" で囲まれたデータ)用にプログラムで確保される領域です。端末アプリケーション実行時に確保される領域とは別の領域で確保されます。

メソッド

名称	説明	引数	戻り値
Load	別スクリプトを呼び出します。	fileName:スクリプトファイル名※1 arg1:各スクリプトに渡す引数※2 arg2:各スクリプトに渡す引数※2 arg3:各スクリプトに渡す引数※2 arg4:各スクリプトに渡す引数※2	なし
LoadSB3	端末アプリケーション(.sb3)を起動します※5。	fileName:sb3ファイル名+拡張子(.sb3) (ドライブ番号:ファイル名)	なし
LoadAPP	端末アプリケーション(.APP)を起動します※3、※4。 BT-1000/1500/600シリーズのみ有効です。	fileName:APPファイル名+拡張子(.APP) (ドライブ番号:ファイル名)	なし

※1　拡張子(.scp)はつけないで指定します。

※2　arg1～4の引数の値が必要ないときはnilを指定します。

▶ 重 要

- **該当するファイルが存在しない場合は、コンパイルエラーとなります。**
- **スクリプトを呼び出した以降のステートメントは実行されません。**
  たとえば、次のようなスクリプトを記述した場合、「b=b+1」は実行されません。

```
a=a+1
System:Load("Sub01",0,1,2,3)
b=b+1
```

参 考

- Loadしたスクリプトの中で、さらにスクリプトをLoadできます。
- 引数は、ファイル間でのデータの受け渡しに使用できます。

※3　引数で指定したファイル起動時に、BTシリーズがリセットされます。

※4　起動時にドライブ0にC言語で作成した端末アプリケーション(.APP)を上書きコピーします。このため、実行中の端末アプリケーションは消去されます。必要に応じて別ドライブに退避してください。

※5　「LoadSB3」メソッドの実行が完了すると、それ以降のプログラムファイルのステートメントは実行されません。呼び出し後、すべてのコントロールのプロパティ値などが初期化されます。実行結果が失敗の場合は、メソッドから戻ってきて次のステートメントが実行されます。

使用例

例1　「Menu00.scp」ファイルから、「System2.scp」のスクリプトを呼び出します。

```
// Menu00.scpファイルの中身
Method main()
    Begin
        System:Load("System2",0,1,2,3)
End Method

// Sub01.scpファイルの中身
Method main()
    Begin
        Handy:ShowMessageBox(System:arg1,"ok") // arg1の値は0
End Method
```

例2　条件によって、「JOB1」～「JOB3」のスクリプトを呼び出しまたは端末アプリケーション(.sb3 または .APP)を起動します。

```
Method main()

btvMenupos
btvItem1 = "JOB1|JOB2|JOB3|SB3起動"
btvItem2 = "APP起動"
btvItem3 = nil
btvItem4 = nil

Begin
    While 1
        // メニューを表示
        btvMenupos = 1
        btvMenupos = Handy:ShowMenu("メニュー",btvItem1,
                                    btvItem2,btvItem3,
                                    btvItem4,btvMenupos)
        Select Case btvMenupos
        Case 1
            System:Load("JOB1","App",0,0,0)    // JOB1スクリプトの呼び出し
        Case 2
            System:Load("JOB2","App",0,0,0)    // JOB2スクリプトの呼び出し
        Case 3
            System:Load("JOB3","App",0,0,0)    // JOB3スクリプトの呼び出し
        Case 4
            System:LoadSB3("1:sample.sb3")   // sample.sb3を起動します
        Case 5
            System:LoadAPP("1:sample.APP")  // sample.APP を起動します
        End Select
    Wend
End Method
```

7-2 Widget操作

GUIプログラムの表示、入力処理は、Widget操作コントロールを使用します。
ここでは、Widget操作コントロールすべてに共通なプロパティ、イベントプロパティ、メソッドについて説明したあと、コントロールごとに説明をします。

Widget共通	Widget共通のプロパティ、イベントプロパティ、メソッドです。 Widget操作コントロールすべてが、タグ指定コントロールです。

注 記

Widgetによっては、使用できないプロパティ、イベントプロパティがあります。
「Widget共通プロパティとWidget操作コントロールの対応一覧」(7-7ページ)

Widget共通プロパティ

名称	説明	範囲	初期値	代入
parent	親Widgetのタグ名 親Widgetのタグ名が格納されます。 「親WidgetとtargetWidget(parentプロパティ、Create、Copyメソッド)」(7-9ページ)	Widgetのタグ名	nil	不可
frame	枠種別 Widgetを装飾する枠を設定します。 「枠種別(frameプロパティ)」(7-8ページ)	“none”:なし “thin”:線枠 “raised”:3D凸型 “recessed”:3D凹型	Widgetごとに違います。	可
foreColor	前景色 前景色をRGBで設定します。 「色の指定(foreColor、backColor、selTextColor、selBackColor、focusColorプロパティ)」(7-9ページ)	RGBの文字列 (“0～255 | 0～255 | 0～255”)※	Widgetごとに違います。	可
backColor	背景色 背景色をRGBで設定します。 「色の指定(foreColor、backColor、selTextColor、selBackColor、focusColorプロパティ)」(7-9ページ)	RGBの文字列 (“0～255 | 0～255 | 0～255”)※	Widgetごとに違います。	可
selTextColor	選択行の文字色 選択行の文字色をRGBで設定します。 「色の指定(foreColor、backColor、selTextColor、selBackColor、focusColorプロパティ)」(7-9ページ)	RGBの文字列 (“0～255 | 0～255 | 0～255”)※	“255 | 255 | 255”	可
selBackColor	選択行の背景色 選択行の背景色をRGBで設定します。 「色の指定(foreColor、backColor、selTextColor、selBackColor、focusColorプロパティ)」(7-9ページ)	RGBの文字列 (“0～255 | 0～255 | 0～255”)※	“0 | 0 | 184”	可
focusColor	フォーカス枠色 フォーカス枠色をRGBで指定します。 「色の指定(foreColor、backColor、selTextColor、selBackColor、focusColorプロパティ)」(7-9ページ)	RGBの文字列 (“0～255 | 0～255 | 0～255”)※	Widgetごとに違います。	可

名称	説明	範囲	初期値	代入
visible	表示／非表示 Widgetを表示するかしないかを設定します。 📖「Widgetの表示／非表示(visibleプロパティ)」(7-10ページ)	true:表示 false:表示しない	false	可
enable	有効／無効 Widgetを有効にするかしないかを設定します。	true:有効 false:無効	false	可
left	最左X座標 targetWidgetに表示するときの最左X座標を設定します。 📖「親WidgetとtargetWidget(parentプロパティ、Create、Copyメソッド)」(7-9ページ) 📖「座標の指定(left、topプロパティ)」(7-11ページ)	0～239	nil	可
top	最上Y座標 targetWidgetに表示するときの最上Y座標を設定します。 📖「親WidgetとtargetWidget(parentプロパティ、Create、Copyメソッド)」(7-9ページ) 📖「座標の指定(left、topプロパティ)」(7-11ページ)	0～1279	nil	可
width	幅 Widgetの幅を設定します。 📖「幅と高さ(width、heightプロパティ)」(7-11ページ)	1～240(ドット)	nil	可
height	高さ Widgetの高さを設定します。 📖「幅と高さ(width、heightプロパティ)」(7-11ページ)	1～1280(ドット)	nil	可
fontSize	フォントサイズ Widgetに表示するテキストのフォントサイズを設定します。	12:12ドットフォント 16:16ドットフォント 24:24ドットフォント	16	可
text	テキスト Widgetに表示するテキストを設定します。	文字列 (最大8192バイト)	“”	可
textAlign	テキストの表示位置 Widgetに表示するテキストの位置を設定します。	“left”:左詰め “center”:中央 “right”:右詰め	Widgetごとに違います。	可

※ "|"と数値の間にスペースは不要です。

Widget共通イベントプロパティ

名称	発生タイミング	sender※
onFocusIn	Widgetにフォーカスが入ったとき	Widgetのタグ名
onFocusOut	Widgetへのフォーカスが抜けたとき	Widgetのタグ名
onTouchUp	タッチパネルが押されている状態から解放されたとき BT-W100/W80/W70シリーズのみ有効です。	Widgetのタグ名

※ イベントの発生元情報です。この情報は、イベントが発生したとき、イベント処理メソッドの引数に自動的に渡されます。

「使用例」(7-84ページ)

■Widget共通プロパティとWidget操作コントロールの対応一覧

	Window	Canvas	Group Box	Text Field	Text Box	Label	ListBox	Combo Box	Check Box	Radio Button	Buttun
parent	○	○	○	○	○	○	○	○	○	○	○
frame	○	○	×	○	○	○	○	○	×	×	○
foreColor	×	×	○	○	○	○	○	○	○	○	○
backColor	○	○	×	○	○	○	○	○	×	×	○
selTextColor	×	×	×	○	○	×	○	○	×	×	×
selBackColor	×	×	×	○	○	×	○	○	×	×	×
focusColor	×	×	×	○	○	×	○	○	○	○	○
visible	○	○	○	○	○	○	○	○	○	○	○
enable	○	○	○	○	○	○	○	○	○	○	○
left	○	○	○	○	○	○	○	○	○	○	○
top	○	○	○	○	○	○	○	○	○	○	○
width	○	○	○	○	○	○	○	○	○	○	○
height	○	○	○	○	○	○	○	○	○	○	○
fontSize	×	×	○	○	○	○	○	○	○	○	○
text	×	×	○	○	○	○	○	○	○	○	○
textAlign	×	×	○	○	○	○	○	○	○	○	○
onFocusIn	○	○	○	○	○	×	○	○	○	○	○
onFocusOut	○	○	○	○	○	×	○	○	○	○	○
onTouchUp	○	○	○	○	○	×	○	○	○	○	○

Widget共通メソッド

名称	説明	引数	戻り値
Create	Widgetを作成します。	targetWidget:作成するWidgetの所属先になるWidgetのタグ名 📖「親WidgetとtargetWidget(parentプロパティ、Create、Copyメソッド)」(7-9ページ)	true:正常終了 false:エラー終了
Copy	Widgetをコピーします。	targetWidget:複製Widgetの所属先になるWidgetのタグ名 copyWidget:コピー元になるWidgetのタグ名 📖「親WidgetとtargetWidget(parentプロパティ、Create、Copyメソッド)」(7-9ページ)	true:正常終了 false:エラー終了
Delete	Widgetを削除します。	なし	true:正常終了 false:エラー終了
Update※	Widgetの表示を最新情報に更新します。	なし	true:正常終了 false:エラー終了
SetFocus	Widgetにフォーカスを当てます。	なし	true:正常終了 false:エラー終了
GetFocus	現在フォーカスが当たっているWidgetのタグ名を取得します。	なし	Widgetのタグ名

※ 描画を変更したときは、必ずUpdateメソッドを実行してください。

■枠種別(frameプロパティ)

Widgetを装飾する枠を次の4種類の中から選択できます。


:"none"(なし)


:"thin"(線枠)


:"raised"(3D凸型)


:"recessed"(3D凹型)

■色の指定(foreColor、backColor、selTextColor、selBackColor、focusColor プロパティ)

色の指定は、赤(R)、緑(G)、青(B)をそれぞれ0～255の範囲で、次のようにおこないます。

●フォーマット

"Rの値|Gの値|Bの値"

【例】赤色の例:"255|0|0"
緑色の例:"0|255|0"
青色の例:"0|0|255"

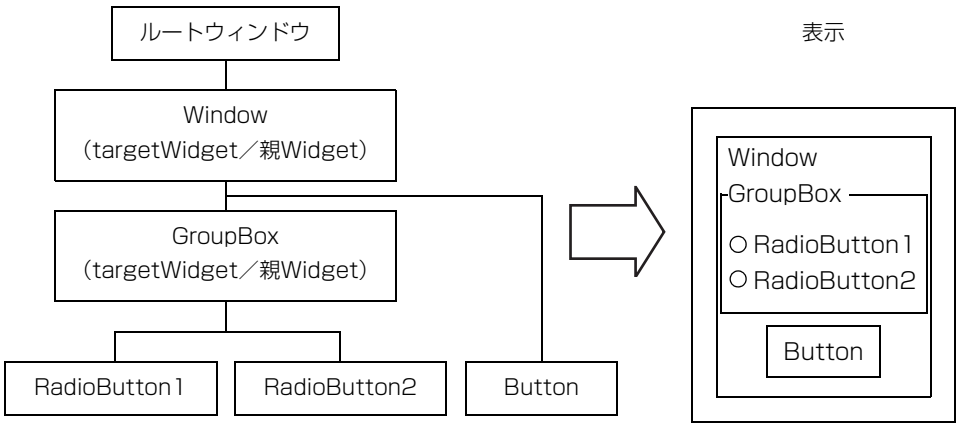
"|"と数値の間にスペースは不要です。

■親WidgetとtargetWidget(parentプロパティ、Create、Copyメソッド)

- targetWidget:　Widget作成時、Widgetを配置する親Widgetを指定します。
- parent(親Widget):Widget作成後、Widgetの親Widgetを参照だけできます。

下図の例で説明します。

- ButtonやGroupBox作成時のtargetWidgetには、Windowのタグ名を指定します。
- ButtonやGroupBox作成後、parentプロパティ値から、Windowのタグ名を参照できます。
- RadioButton1やRadioButton2作成時のtargetWidgetには、GroupBoxのタグ名を指定します。
- RadioButton1やRadioButton2作成後、parentプロパティ値から、GroupBoxのタグ名を参照できます。

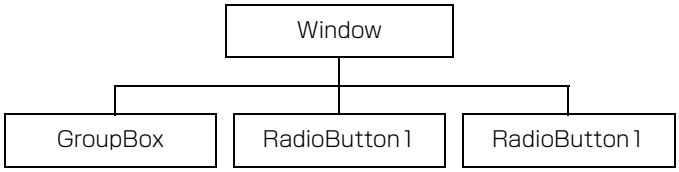


■Widgetの表示／非表示（visibleプロパティ）

Widgetの表示／非表示は、親Widgetと子Widgetの関連で、次のような動作をします。

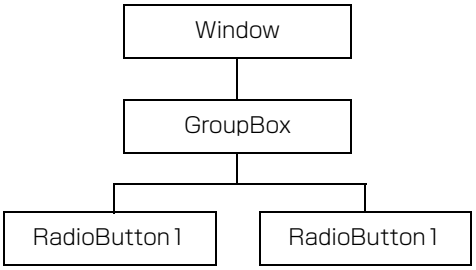
- 親Widgetのvisibleがtrueのとき、子Widgetはvisibleの設定に従って、個別に表示／非表示の状態になります。
- 親Widgetのvisibleがfalseのとき、子Widgetのvisibleの設定にかかわらず、子Widgetはすべて非表示になります。

【例1】



- Windowがvisible=trueのとき、GroupBox、RadioButton1とRadioButton2は、個別に表示／非表示の動作になります。
- Windowがvisible=falseのとき、GroupBox、RadioButton1とRadioButton2は、すべて非表示になります。

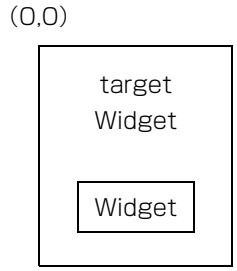
【例2】



- Windowがvisible=true、およびGroupBoxがvisible=trueのとき、RadioButton1とRadioButton2は、個別に表示／非表示の動作になります。
- Windowがvisible=true、およびGroupBoxがvisible=falseのとき、RadioButton1とRadioButton2は非表示になります。
- Windowがvisible=falseのとき、GroupBox、RadioButton1、RadioButton2は、すべて非表示になります。

■座標の指定(left、topプロパティ)

Widgetを作成するときのleftプロパティのX座標、topプロパティのY座標は、targetWidgetに対する相対座標で指定します。



最大X座標＝targetWidgetの幅-1
最大Y座標＝targetWidgetの高さ-1

前述(7-10ページ)の【例１】と【例2】で説明します。

- 【例1】では、RadioButton1とRadioButtun2のleft、topプロパティは、Windowに対する相対座標で指定します。
- 【例2】では、RadioButton1とRadioButtun2のleft、topプロパティは、GroupBoxに対する相対座標で指定します。

■幅と高さ(width、heightプロパティ)

Widgetの幅、高さの最小値は、枠種別によって次のように異なります。

枠種別	幅、高さの最小値
"none"	1
"thin"	3
"raised"	5
"recessed"	5

Window	テキストボックスなどを配置するウィンドウを設定します。 タグ指定コントロールです。

注 記

タグ名として、「ROOT_WINDOW」、「EMULATE_WINDOW」は予約タグなので使用できません。

「ルートウィンドウ、エミュレートウィンドウ」(7-14ページ)

プロパティ

名称	説明	範囲	初期値	代入
parent※	親Widgetのタグ名	Widgetのタグ名	nil	不可
frame※	枠種別	“none”:なし “thin”:線枠 “raised”:3D凸型 “recessed”3D凹型	raised	可
backColor※	背景色	RGBの文字列	“255|255|255”	可
visible※	表示／非表示	true:表示 false:表示しない	false	可
enable※	有効／無効	true:有効 false:無効	false	可
left※	最左X座標	0～239	0	可
top※	最上Y座標	0～1279	0	可
width※	幅	1～240(ドット)	100	可
height※	高さ	1～1280(ドット)	100	可
scrollBarV	縦スクロールバーの表示	true:表示 false:表示しない	false	可
scrollBarH	横スクロールバーの表示	true:表示 false:表示しない	false	可
showPosx	Window内の表示開始位置X座標 「表示位置の指定(showPosx、showPosyプロパティ、Scrollメソッド)」(7-14ページ)	0～239 (Window上のx座標)	0	可
showPosy	Window内の表示開始位置Y座標 「表示位置の指定(showPosx、showPosyプロパティ、Scrollメソッド)」(7-14ページ)	0～1279 (Window上のｙ座標)	0	可

※ Widget共通プロパティです。他のWidget操作コントロールでも使用します。ここでは、Widget共通プロパティの詳細説明を省略しています。
Widget 共通プロパティの詳細説明については、「Widget 共通プロパティ」(7-5 ページ)を参照してください。

イベントプロパティ

名称	発生タイミング	sender
onFocusIn※	Windowにフォーカスが入ったとき	Windowのタグ名
onFocusOut※	Windowからフォーカスが抜けたとき	Windowのタグ名
onTouchUp※	タッチパネルが押されている状態から解放されたとき BT-W100/W80/W70シリーズのみ有効です。	Windowのタグ名

※ Widget共通イベントプロパティです。他のWidget操作コントロールでも使用します。ここでは、Widget共通イベントプロパティの詳細説明を省略しています。
Widget 共通イベントプロパティの詳細説明については、「Widget 共通イベントプロパティ」(7-7ページ)を参照してください。

メソッド

名称	説明	引数	戻り値
Create※	Windowを作成します。	targetWidget:作成するWidgetの所属先になるWindowのタグ名	true:正常終了 false:エラー終了
Copy※	Windowを複製します。	targetWidget:複製Widgetの所属先になるWindowのタグ名 copyWidget:コピー元になるWindowのタグ名	true:正常終了 false:エラー終了
Delete※	Windowを削除します。	なし	true:正常終了 false:エラー終了
Update※	Windowの表示を更新します。	なし	true:正常終了 false:エラー終了
SetFocus※	Windowにフォーカスを当てます。	なし	true:正常終了 false:エラー終了
GetFocus※	現在フォーカスが当たっているWidgetのタグ名を取得します。	なし	Widgetのタグ名
Scroll	Window内の表示をスクロールします。 「表示位置の指定(showPosx、showPosyプロパティ、Scrollメソッド)」(7-14ページ)	x:x座標 (Window上のx座標) y:y座標 (Window上のy座標)	true:正常終了 false:エラー終了

※ Widget共通メソッドです。他のWidget操作コントロールでも使用します。ここでは、Widget共通メソッドの詳細説明を省略しています。
Widget 共通メソッドの詳細説明については、「Widget 共通メソッド」(7-8 ページ)を参照してください。

■ルートウィンドウ、エミュレートウィンドウ

ルートウィンドウ(”ROOT_WINDOW”)、エミュレートウィンドウ(”EMULATE_WINDOW”)は、システムに最初から存在するウィンドウです。

- Windowのタグ名に、”ROOT_WINDOW”と”EMULATE_WINDOW”は使用できません。
- ルートウィンドウのプロパティ値は、参照だけできます。

プロパティ	参照時の値
visible	常にtrue
left	常に0
top	常に0
width	常に240
height	常に296

- エミュレートウィンドウのプロパティ値は、次のようになります。

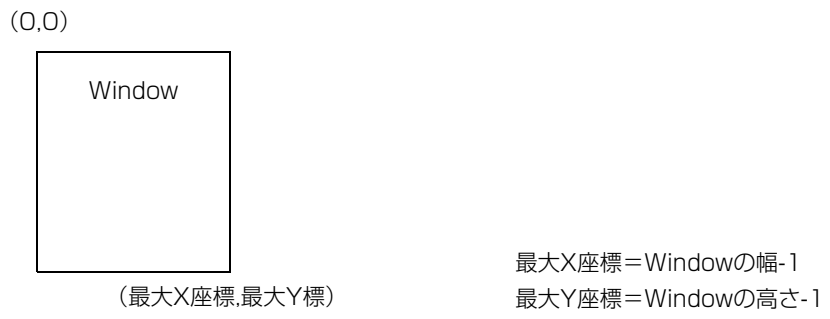
プロパティ	初期値	値の代入
visible	false	○
enable	false	○
left	常に0	×
top	常に0	×
width	常に240	×
height	常に296	×

エミュレートウィンドウの「visible」プロパティと「enable」プロパティは、「false」に設定されているので、エミュレータ表示するときは、「true」にします。

「エミュレータ表示」(6-4ページ)

■表示位置の指定(showPosx、showPosyプロパティ、Scrollメソッド)

次の座標で設定します。



使用例

SubWindow1をROOT_WINDOW上に作成します。

```
Method Job1()
	Begin

	With Window<"SubWindow1">                    /* Windowのタグ名に
                                                    「SubWindow1」を設定 */
		:Create("ROOT_WINDOW")               /* ルートウインドウ上に
                                                    「SubWindow1」を作成 */
		:frame = "thin"                      // 枠なし
		:backColor = "133|142|199"           // 背景色の設定
		:left = 50                           /* ルートウィンドウ上に
                                                    「SubWindow1」
                                                   を表示する最左X座標＝50 */
		:top = 50                            /* ルートウィンドウ上に
                                                    「SubWindow1」
                                                    を表示する最上Y座標=50 */
		:width = 100                         // 幅は100ドット
		:height = 100                        // 高さは100ドット
		:scrollBarV = false                  // 縦スクロールバーの表示をしない
		:scrollBarH = false                  // 横スクロールバーを表示しない
		:showPosx = 0                        /*「SubWindow1」内の
                                                    表示X座標＝0 */
		:showPosy = 0                        /*「SubWindow1」内の
                                                    表示Y座標=0 */
		:onFocusOut = SubWindow1_FocusOut    // イベント登録
		:visible = true
		:enable = true
		:Update()                            // 表示を更新
	End With

	Event:Wait() // イベント取得

EndMethod

// イベント処理メソッドの設定
Method SubWindow1_FocusOut(sender)
	Begin

	Select Case Window<"SubWindow1">:frame
		case "none"
			Window<"SubWindow1">:frame = "thin"
		case "thin"
			Window<"SubWindow1">:frame = "raised"
		case "raised"
			Window<"SubWindow1">:frame = "recessed"
		case "recessed"
			Window<"SubWindow1">:frame = "none"
	EndSelect

	Window<"SubWindow1">:Update()

EndMethod
```

Canvas	背景画像や図形を表示できるウィンドウを設定します。 タグ指定コントロールです。

注 記

- **タグ名として、「ROOT_WINDOW」、「EMULATE_WINDOW」は予約タグなので使用できません。**
- **「Drawing」コントロールを使用して、Canvasに文字や図形を描画します。**

プロパティ

名称	説明	範囲	初期値	代入
parent※	親Widgetのタグ名	Widgetのタグ名	nil	不可
frame※	枠種別	“none”：なし “thin”：線枠 “raised”：3D凸型 “recessed”3D凹型	raised	可
backColor※	背景色	RGBの文字列	“255|255|255”	可
visible※	表示／非表示	true：表示 false：表示しない	false	可
enable※	有効／無効	true：有効 false：無効	false	可
left※	最左X座標	0～239	0	可
top※	最上Y座標	0～1279	0	可
width※	幅	1～240（ドット）	100	可
height※	高さ	1～1280（ドット）	100	可

※ Widget共通プロパティです。他のWidget操作コントロールでも使用します。ここでは、Widget共通プロパティの詳細説明を省略しています。
Widget 共通プロパティの詳細説明については、「Widget 共通プロパティ」（7-5 ページ）を参照してください。

イベントプロパティ

名称	発生タイミング	sender
onFocusIn※	Canvasにフォーカスが入ったとき	Canvasのタグ名
onFocusOut※	Canvasからフォーカスが抜けたとき	Canvasのタグ名
onTouchUp※	タッチパネルが押されている状態から解放されたとき BT-W100/W80/W70シリーズのみ有効です。	Canvasのタグ名

※ Widget共通イベントプロパティです。他のWidget操作コントロールでも使用します。ここでは、Widget共通イベントプロパティの詳細説明を省略しています。
Widget 共通イベントプロパティの詳細説明については、「Widget 共通イベントプロパティ」（7-7ページ）を参照してください。

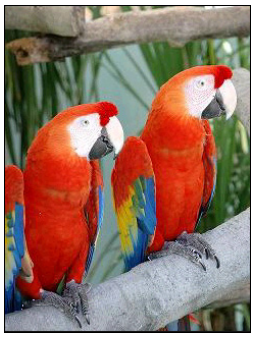
メソッド

名称	説明	引数	戻り値
Create※	Canvasを作成します。	targetWidget:作成するWidgetの所属先になるWindowのタグ名	true:正常終了 false:エラー終了
Copy※	Canvasを複製します。	targetWidget:複製Widgetの所属先になるWindowのタグ名 copyWidget:コピー元になるCanvasのタグ名	true:正常終了 false:エラー終了
Delete※	Canvasを削除します。	なし	true:正常終了 false:エラー終了
Update※	Canvasの表示を更新します。	なし	true:正常終了 false:エラー終了
SetFocus※	Canvasにフォーカスを当てます。	なし	true:正常終了 false:エラー終了
GetFocus※	現在フォーカスが当たっているWidgetのタグ名を取得します。	なし	Widgetのタグ名
SetImage	Canvasの背景に表示する画像を設定します。	fileName:画像ファイル名(ドライブ番号:ファイル名) 「画像の貼り付け(SetImageメソッド)」(7-18ページ)	true:正常終了 false:エラー終了

※ Widget共通メソッドです。他のWidget操作コントロールでも使用します。ここでは、Widget共通メソッドの詳細説明を省略しています。
Widget 共通メソッドの詳細説明については、「Widget 共通メソッド」(7-8 ページ)を参照してください。

■画像の貼り付け(SetImageメソッド)

次のように、Canvasの背景に画像を貼り付けられます。



●画像形式

使用できる画像ファイルの形式は、次の4種類です。

- ビットマップファイル(.BMP)
- JPEGファイル(.JPG)(Exif情報が付加されたJPEGファイルは使用できません)
- PNGファイル(.PNG)
- GIFファイル(.GIF)

●画像の指定

ドライブ番号:ディレクトリ名¥ファイル名(最大255バイトまで)

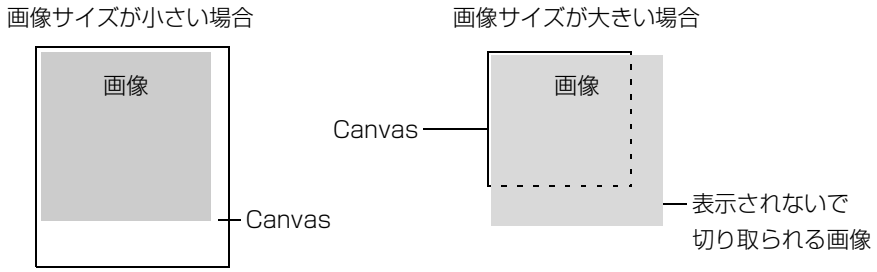
【例】2:AAA¥BBB.BMP

ドライブ番号を指定しない場合は、ドライブ2になります。

●画像の表示

画像はCanvasの左上から貼り付けられます。

画像サイズとCanvasサイズが同じでない場合、下図のようになります。



使用例

Canvas 1の背景に画像を貼り付けます。

```
Method Job1()
    Begin

    With Canvas<"Canvas1">                  // Canvasのタグ名に「Canvas1」を設定
        :Create("ROOT_WINDOW")              // ルートウィンドウ上に「Canvas1」を作成
        :visible = TRUE
        :enable = TRUE
        :backColor="0|102|255"              // 背景色を設定
        :frame = "thin"                     // フレームの設定
        :left = 30                          /* ルートウィンドウ上に「Canvas1」
                                               を表示する最左X座標＝30 */
        :top = 50                           /* ルートウィンドウ上に「Canvas1」
                                               を表示する最上Y座標=50 */
        :width = 120                        // 幅は120ドット
        :height = 150                       // 高さは150ドット
        :SetImage("2:SUN.BMP")              // ビットマップファイルの設定
        :Update()                           // 表示を更新

        Event:Wait()                        // イベント取得

    End With

EndMethod
```

GroupBox	複数のWidgetを1つのグループとして扱います。 タグ指定コントロールです。

プロパティ

名称	説明	範囲	初期値	代入
parent※	親Widgetのタグ名	Widgetのタグ名	nil	不可
foreColor※	背景色	RGBの文字列	"0|0|0"	可
visible※	表示／非表示	true:表示 false:表示しない	false	可
enable※	有効／無効	true:有効 false:無効	false	可
left※	最左X座標	0～239	0	可
top※	最上Y座標	0～1279	0	可
width※	幅	1～240(ドット)	80	可
height※	高さ	1～1280(ドット)	100	可
fontSize※	フォントサイズ	12:12ドットフォント 16:16ドットフォント 24:24ドットフォント	12	可
text※	テキスト	文字列 (最大8192バイト)	""	可
textAlign※	テキストの表示位置	"left":左詰め "center":中央 "right":右詰め	"left"	可

※ Widget共通プロパティです。他のWidget操作コントロールでも使用します。ここでは、Widget共通プロパティの詳細説明を省略しています。
Widget 共通プロパティの詳細説明については、「Widget 共通プロパティ」(7-5 ページ)を参照してください。

イベントプロパティ

名称	発生タイミング	sender
onFocusIn※	GroupBoxにフォーカスが入ったとき	GroupBoxのタグ名
onFocusOut※	GroupBoxからフォーカスが抜けたとき	GroupBoxのタグ名
onTouchUp※	タッチパネルが押されている状態から解放されたとき BT-W100/W80/W70シリーズのみ有効です。	GroupBoxのタグ名

※ Widget共通イベントプロパティです。他のWidget操作コントロールでも使用します。ここでは、Widget共通イベントプロパティの詳細説明を省略しています。
Widget 共通イベントプロパティの詳細説明については、「Widget 共通イベントプロパティ」(7-7ページ)を参照してください。

メソッド

名称	説明	引数	戻り値
Create※	GroupBoxを作成します。	targetWidget:作成するWidgetの所属先になるWindowのタグ名	true:正常終了 false:エラー終了
Copy※	GroupBoxを複製します。	targetWidget:複製Widgetの所属先になるWindowのタグ名 copyWidget:コピー元になるGroupBoxのタグ名	true:正常終了 false:エラー終了
Delete※	GroupBoxを削除します。	なし	true:正常終了 false:エラー終了
Update※	GroupBoxの表示を更新します。	なし	true:正常終了 false:エラー終了
SetFocus※	GroupBoxにフォーカスを当てます。	なし	true:正常終了 false:エラー終了
GetFocus※	現在フォーカスが当たっているWidgetのタグ名を取得します。	なし	Widgetのタグ名

※ Widget共通メソッドです。他のWidget操作コントロールでも使用します。ここでは、Widget共通メソッドの詳細説明を省略しています。
Widget 共通メソッドの詳細説明については、「Widget 共通メソッド」(7-8 ページ)を参照してください。

使用例

GroupBox1の中にRadioButtonを2つ作成します。

```
Method Job1()
    Begin

    With GroupBox<"GroupBox1">              /* 作成するGroupBoxのタグ名は
                                               「GroupBox1」 */
        :Create("ROOT_WINDOW")              /* ルートウィンドウ上に「BroupBox1」を
                                               作成 */
        :foreColor = "0|0|0"                // 前景色は黒
        :left = 10                          /* ルートウィンドウ上に「GroupBox1」を
                                               表示する最左X座標＝10 */
        :top = 20                           /* ルートウィンドウ上に「GroupBox1」を
                                               表示する最上Y座標＝20 */
        :width = 220                        // 幅は220ドット
        :height = 200                       // 高さは200ドット
        :text = "表示･非表示"               // 「表示･非表示」を表示
        :visible = TRUE
        :enable = TRUE
    End With
```

```
    With RadioButton<"item1">                  /* 作成するRadioButtonのタグ名は
                                                  「item1」*/
        :Create("GroupBox1")                   // 「GroupBox1」上に「item1」を作成
        :left = 10                             /* 「GroupBox1」上に「item1」を
                                                  表示する最左X座標＝10 */
        :top = 50                              /* 「GroupBox1」上に「item1」を
                                                  表示する最上Y座標＝50 */
        :width = 125                           // 幅は125ドット
        :height = 30                           // 高さは30ドット
        :groupId = 0                           // GroupIdに0を指定
        :text = "表示する"                      // 「表示する」を表示
        :checked = "item1"                     // 「item1」を選択状態にする
        :visible = TRUE
        :enable = TRUE
    End With

    With RadioButton<"item2">                  /* 作成するRadioButtonのタグ名は
                                                  「item2」 */
        :Create("GroupBox1")                   // 「GroupBox1」上に「item2」を作成
        :left = 10
        :top = 100
        :width = 125
        :height = 30
        :groupId = 0                           // GroupIdに0を指定
        :text = "表示しない"                    // 「表示する」を表示
        :checked = "item1"                     // 「item1」を選択状態にする
        :visible = TRUE
        :enable = TRUE
    End With

    Window<"ROOT_WINDOW">:Update()             // ルートウィンドウの表示更新

    Event:Wait()                               // イベント取得

EndMethod
```

TextField	1行のテキストを表示、入力を設定します。 タグ指定コントロールです。

プロパティ

名称	説明	範囲	初期値	代入
parent※1	親Widgetのタグ名	Widgetのタグ名	nil	不可
frame※1	枠種別	“none”:なし “thin”:線枠 “raised”:3D凸型 “recessed”:3D凹型	"recessed"	可
foreColor※1	前景色	RGBの文字列 (“0~255|0~255|0~255”)	"0|0|0"	可
backColor※1	背景色	RGBの文字列 (“0~255|0~255|0~255”)	"255|255|255"	可
selTextColor※1	選択行の文字色	RGBの文字列 (“0~255|0~255|0~255”)	"255|255|255"	可
selBackColor※1	選択行の背景色	RGBの文字列 (“0~255|0~255|0~255”)	"0|0|184"	可
focusColor※1	フォーカス枠色	RGBの文字列 (“0~255|0~255|0~255”)	"000|255|000"	可
visible※1	表示／非表示	true:表示 false:表示しない	false	可
enable※1	有効／無効	true:有効 false:無効	false	可
left※1	最左X座標	0~239	0	可
top※1	最上Y座標	0~1279	0	可
width※1	幅	1~240(ドット)	80	可
height※1	高さ	1~1280(ドット)	24	可
fontSize※1	フォントサイズ	12:12ドットフォント 16:16ドットフォント 24:24ドットフォント	12	可
text※1	テキスト 入力完了後は、入力文字列が格納されます。	文字列 (最大8192バイト)	“”	可
textAlign※1	テキストの表示位置	“left”:左詰め “center”:中央 “right”:右詰め	"left"	可
maxLength	文字列入力するときの最大文字列長	0~8192(バイト): 半角で8192文字まで、 全角で4096文字まで	8192	可
editDirect	フォーカスインしたときに入力状態にします。	true:有効 false:無効	false	可

名称	説明	範囲	初期値	代入
enableBCR	トリガモードの設定 バーコード読み取りするときのモードを設定します。 バーコードを読み取りしない場合には、「nil」を指定します。	nil:バーコード入力しない 1:オートオフ 2:連続読み取り 3:モーメンタリ 4:ソフトウエアトリガ 5:離して読み 6:ダブルクリック	1	可
inputOption	入力オプションの設定 入力するときの動作を設定します。 「日本語を入力する(inputOptionプロパティ)」(7-26ページ) 「電卓機能(inputOptionプロパティ)」(7-30ページ)	0:なし 1:バーコードリーダのみの入力 2:日本語入力をする 4:電卓から入力する 8:シフトキーで切り替え(初期無効) 16:シフトキーで切り替え(初期有効)	0	可
fontBold	太字設定の有効/無効 BT-W100/W80/W70シリーズのみ有効です。	true/false	true	可
fontSize	フォントサイズ BT-W100/W80/W70シリーズのみ有効です。	6～255	-	可
fontName	フォント名 BT-W100/W80/W70シリーズのみ有効です。	32文字以内のフォント名	""	可
textVAlign	テキストの表示位置(上下)	"top"（上端揃え） "bottom"（下端揃え） "center"（上下中央揃え）	"top"	可
simpleWidget	Simple Widget有効/無効	true/false	false	可
inFocusTextColor※2	フォーカス時の文字色	RGBの文字列 （"0～255 | 0～255 | 0～255"）	"0 | 0 | 0"	可
inFocusBackColor※2	フォーカス時の背景色	RGBの文字列 （"0～255 | 0～255 | 0～255"）	"192 | 192 | 192"	可
outFocusTextColor※2	非フォーカス時の文字色	RGBの文字列 （"0～255 | 0～255 | 0～255"）	"0 | 0 | 0"	可
outFocusBackColor※2	非フォーカス時の背景色	RGBの文字列 （"0～255 | 0～255 | 0～255"）	"255 | 255 | 255"	可

※1 Widget共通プロパティです。他のWidget操作コントロールでも使用します。ここでは、Widget共通プロパティの詳細説明を省略しています。
Widget共通プロパティの詳細説明については、「Widget共通プロパティ」(7-5ページ)を参照してください。

※2 simpleWidgetプロパティがtrue、frameプロパティが"none"の場合のみ有効です。

イベントプロパティ

名称	発生タイミング	sender
onFocusIn※1	TextFieldにフォーカスが入ったとき	TextFieldのタグ名
onFocusOut※1	TextFieldからフォーカスが抜けたとき	TextFieldのタグ名
onTouchUp※1	タッチパネルが押されている状態から解放されたとき BT-W100/W80/W70シリーズのみ有効です。	TextFieldのタグ名
onEditStart	入力を開始したとき	TextFieldのタグ名
onEditEnd	入力を完了したとき キー入力で「ENT」を押して入力を確定したとき、またはトリガーキーを押して自動的にバーコードを読み取ったとき	TextFieldのタグ名
onEditCancel	入力をキャンセルしたとき	TextFieldのタグ名
onTouchOut	タッチパネルを押して入力状態を解除したとき BT-W100/W80/W70シリーズのみ有効です。	TextFieldのタグ名
onScanComplete※2	バーコードの読み取りが完了したとき	TextFieldのタグ名

※1 Widget共通イベントプロパティです。他のWidget操作コントロールでも使用します。ここでは、Widget共通イベントプロパティの詳細説明を省略しています。
Widget共通イベントプロパティの詳細説明については、「Widget共通イベントプロパティ」(7-7ページ)を参照してください。

※2 バーコード読み取りが完了し、TextFieldに読み取りデータが表示される場合に発生します。
読み取り異常の場合は、BCRコントロールのイベントプロパティから取得できます。
- タイムアウト　:onScanTimeout
- 読み取りエラー:onScanError

メソッド

名称	説明	引数	戻り値
Create※	TextFieldを作成します。	targetWidget:作成するWidgetの所属先になるWindowのタグ名	true:正常終了 false:エラー終了
Copy※	TextFieldをコピーします。	targetWidget:複製Widgetの所属先になるWindowのタグ名 copyWidget:コピー元になるTextFieldのタグ名	true:正常終了 false:エラー終了
Delete※	TextFieldを削除します。	なし	true:正常終了 false:エラー終了
Update※	TextFieldの表示を最新情報に更新します。	なし	true:正常終了 false:エラー終了
SetFocus※	TextFieldにフォーカスを当てます。	なし	true:正常終了 false:エラー終了
GetFocus※	現在フォーカスが当たっているWidgetのタグ名を取得します。	なし	Widgetのタグ名
SetEditMode	各状態(フォーカスイン・アウト)から入力状態へ遷移します。逆は入力状態からフォーカスイン状態に遷移させます。	mode: “on”:入力状態へ遷移 “off”:フォーカスイン状態に遷移	true:正常終了 false:エラー終了

※ Widget共通メソッドです。他のWidget操作コントロールでも使用します。ここでは、Widget共通メソッドの詳細説明を省略しています。
Widget 共通メソッドの詳細説明については、「Widget 共通メソッド」(7-8 ページ)を参照してください。

■日本語を入力する(inputOptionプロパティ)

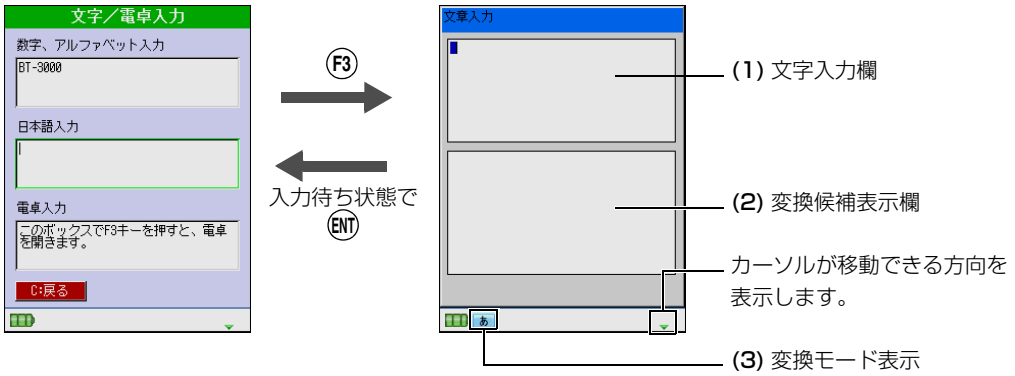
「inputOption」プロパティに日本語入力／シフト切り替えを設定して、シフトキー(F3)を押すと、次の文字が入力できるようになります。

- 日本語入力を設定した場合
  日本語(全角ひらがな／カタカナ)を入力できます。
- シフト切り替えを設定した場合
  日本語(全角ひらがな／カタカナ)、半角および全角英数字(A-Z/a-z/0-9)、記号を入力できます。

●日本語の入力

1 (F3)キーを押します。

日本語入力ダイアログボックスが表示されます。
入力待ち状態で(ENT)キーを押すと元の画面に戻ります。



(1)文字入力欄

入力した文字を表示します。

(2)変換候補表示欄

入力した文字を変換するときに、変換候補を表示します。
また、記号入力するときには一覧を表示します。

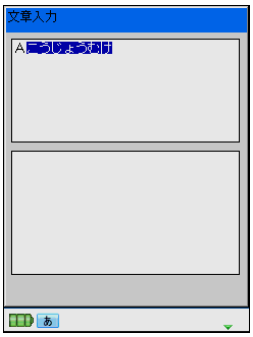
(3)変換モード表示

変換モードを表示します。
シフト切り替えを設定した場合は、(F3)キーを押すたびに、変換モードを切り替えます。

表示	変換モード
あ	日本語(ひらがな)入力モード
AB	半角英数字入力モード
A	全角英数字入力モード

2 キーを押して、文字を入力します。

日本語入力モードの主な操作と、
入力できる文字は以下のようになります。



キー	機能
◀/▶	カーソルの移動
Ⓒ	カーソルの右側の1文字を削除します。

キー	入力できる文字
①	あ→い→う→え→お→ぁ→ぃ→ぅ→ぇ→ぉ→あ･･･
②	か→き→く→け→こ→か･･･
③	さ→し→す→せ→そ→さ･･･
④	た→ち→つ→て→と→っ→た･･･
⑤	な→に→ぬ→ね→の→な･･･
⑥	は→ひ→ふ→へ→ほ→は･･･

キー	入力できる文字
⑦	ま→み→む→め→も→ま･･･
⑧	や→ゆ→よ→ゃ→ゅ→ょ→や･･･
⑨	ら→り→る→れ→ろ→ら･･･
⓪	わ→を→ん→わ･･･
•※	゛→゜→゛･･･
−	―→、→。→！→？→・→―･･･

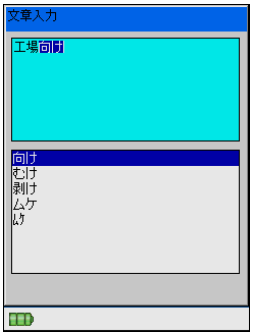
※濁音、半濁音のある文字以外では入力できません。

3 ▲または▼キーを押します。

変換候補欄に変換候補が表示されます。
▲/▼キーを押して候補を選択します。

参考 ◯キー(◀ ▶)を押すと、文節の長さを変更できます。

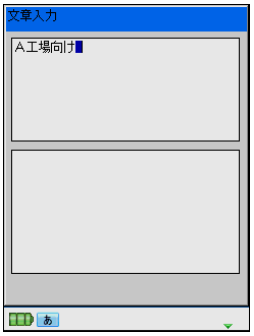
4 ㊎キーを押します。



5 複数の文節がある場合には、手順3、4を繰り返します。

6 **変換が完了したら、(ENT)キーを押します。**

文字入力欄の文字列が画面に表示されます。



● 入力可能な文字

日本語入力モード

キー	入力できる文字
①	あ→い→う→え→お→ぁ→ぃ→ぅ→ぇ→ぉ→あ･･･
②	か→き→く→け→こ→か･･･
③	さ→し→す→せ→そ→さ･･･
④	た→ち→つ→て→と→っ→た･･･
⑤	な→に→ぬ→ね→の→な･･･
⑥	は→ひ→ふ→へ→ほ→は･･･

キー	入力できる文字
⑦	ま→み→む→め→も→ま･･･
⑧	や→ゆ→よ→ゃ→ゅ→ょ→や･･･
⑨	ら→り→る→れ→ろ→ら･･･
⓪	わ→を→ん→わ･･･
•※	゛→゜→゛･･･
−※	ー→、→。→！→？→・→ー･･･

※濁音、半濁音のある文字以外では入力できません。

半角/全角英数字入力モード

全角英数字入力モードの場合、全角文字になります。

キー	入力できる文字
①	(半角スペース)→1→(半角スペース)･･･
②	A→B→C→a→b→c→2→A･･･
③	D→E→F→d→e→f→3→D･･･
④	G→H→I→g→h→i→4→G･･･
⑤	J→K→L→j→k→l→5→J･･･
⑥	M→N→O→m→n→o→6→M･･･

キー	入力できる文字
⑦	P→Q→R→S→p→q→r→s→7→P･･･
⑧	T→U→V→t→u→v→8→T･･･
⑨	W→X→Y→Z→w→x→y→z→9→W･･･
⓪	-→+→/→0→-･･･
•	*→%→$→.→*･･･
−※	(半角入力時)-→(半角スペース)→-･･･ (全角入力時)−

※ (−)キーの入力動作は、「Handy」または「Key」コントロールで設定できます。

「Handy」(7-320ページ)
「Key」(7-315ページ)

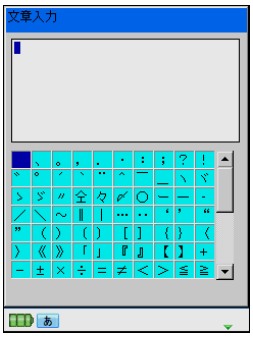
●操作一覧

キー	状態		
	入力待ち	入力中	変換候補選択中
ENT	元の画面に戻ります。文字入力欄の内容が元の画面に入力されます。	入力中の文字を無変換で確定します。	変換を確定して、入力待ち状態に戻ります。文節が残っている場合には、次の候補を表示します。
C	カーソルの右側の1文字を削除します。	カーソルの右側の1文字を削除します。	変換を中止して、入力中の状態に戻ります。
▲	カーソル移動	反転している文節の変換候補を表示します。	候補を選択します。(上方向に移動します)
▼			候補を選択します。(下方向に移動します)
◄		(機能しません)	変換中の文節を1文字減らします。
►		(機能しません)	変換中の文節を1文字増やします。
F1	記号一覧表示	(機能しません)	(機能しません)
F2	(機能しません)	全角かな／全角カナ／半角カナ切り替え	(機能しません)
F3	変換モード切り替え		

●記号の入力

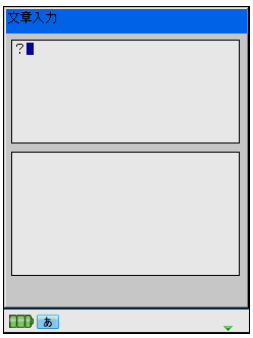
1 **入力待ち状態でF1キーを押します。**

変換候補表示欄に記号一覧が表示されます。



2 **キーで選択して、ENTキーを押します。**

選択した記号が入力されます。



●注意事項

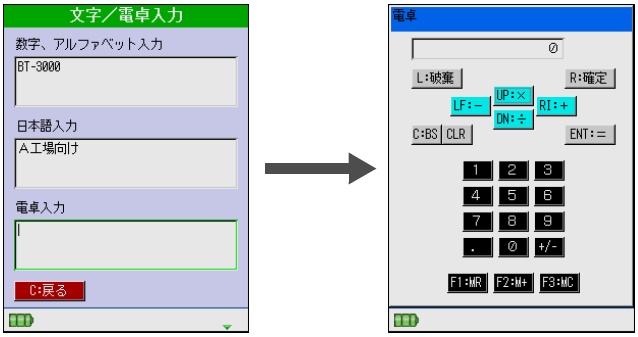
- 変換前に入力できる文字数は全角20文字までになります。
- 区点コードや外字の入力はできません。

■電卓機能(inputOptionプロパティ)

「inputOption」プロパティに電卓入力を設定すると、電卓機能が有効になり、電卓で計算した結果が文字列として入力されます。
電卓からの入力方法は、次のとおりです。

1 (F3)キーを押します。

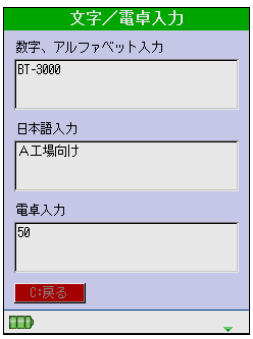
電卓画面が表示されます。



キー	記号	内容
①～⑨	数字	数字を入力します。
(•)	.	小数点を入力します。
◀	−	加減乗除の演算をします。
▲	×	
▶	+	
▼	÷	
(ENT)	=	
(L)	破棄	計算結果を破棄して、元の画面に戻ります。
(R)	確定	計算結果を戻り値として、元の画面に反映します。
(F1)	MR	メモリに記憶した数値を呼び出します。
(F2)	M+	表示している数値をメモリ内の数値に加算します。
(F3)	MC	メモリに記憶した数値を消去します。

2 計算が完了したら、(R)キーを押します。

元の画面に戻ります。計算結果が入力されます。
計算結果を入力しない場合には、(L)キーを押します。



使用例

例1　TextFieldを使用して、バーコード読み取りをおこないます。

```
Method ReadBarcode()
    Begin
        With Label< "表示" >                    // Labelのタグ名に「表示」を設定
            :Create( "ROOT_WINDOW" )            // ルートウィンドウ上にLabelを作成

            // バーコード読み取りのメッセージ表示
            :text = "バーコードを読み取ってください。¥n
                TRGキー押下で読み取り開始¥n(Cキー長押しでキャンセル)"
            :width = 240                        // 幅は240ドット
            :height = 60                        // 高さは60ドット
            :visible = true
            :enable = true
        End With

        With TextField< "入力" >                // TextFieldのタグ名に「入力」を設定
            :Create( "ROOT_WINDOW" )            /* ルートウィンドウ上にTextFieldを
                                                   作成 */
            :top = 70                           /* ルートウィンドウ上にTextFielを
                                                   表示する最上Y座標＝70 */
            :left = 10                          /* ルートウィンドウ上にTextFielを
                                                   表示する最左X座標＝10 */
            :width = 200                        // 幅は200ドット
            :height = 40                        // 高さは40ドット
            :enableBCR = 1                      /* バーコード読み取り時のモードは
                                                   オートオフ */
            :visible = true
            :enable = true
        End With

        With Window< "ROOT_WINDOW" >
            :enable = true
            :Update()                           // ルートウィンドウの表示更新
        End With

        Event:Wait()                            // イベント取得

End Method
```

例2　editDirectプロパティを使用して、フォーカスインしたときに入力状態にするかしないかを変更します。

```
Method main()
Begin

    With Button<"Button1">
        :Create("ROOT_WINDOW")
        :foreColor =  "0¦0¦0"
        :left = 62
        :top = 38
        :height =  24
        :width = 117
        :fontSize =  12
        :textAlign = "center"
        :text =  "editDirectあり"
        :visible=True
        :enable=True
    EndWith

    With Button<"Button2">
        :Create("ROOT_WINDOW")
        :foreColor =  "0¦0¦0"
        :left = 62
        :top = 170
        :height =  24
        :width = 117
        :fontSize =  12
        :textAlign = "center"
        :text =  "editDirectなし"
        :visible=True
        :enable=True
    EndWith

    With TextField<"TextField1">
        :Create("ROOT_WINDOW")
        :foreColor =  "000¦000¦000"
        :backColor =  "255¦255¦192"
        :left = 52
        :top = 95
        :height =  24
        :width = 140
        :fontSize =  16
        :textAlign = "left"
        :editDirect = true                    // editDirectをtrueにする
        :visible = true
        :enable = true
```

```
        EndWith

        With TextField<"TextField2">
            :Create("ROOT_WINDOW")
            :foreColor =  "0¦0¦0"
            :backColor =  "255¦255¦192"
            :left = 52
            :top = 230
            :height =  24
            :width = 140
            :fontSize =  16
            :textAlign = "left"
            :editDirect = false                    // editDirectをfalseにする
            :visible = true
            :enable = true
        EndWith

        Button<"Button1">:onButton = OnButton1
        Button<"Button2">:onButton = OnButton2

        With Window< "ROOT_WINDOW" >
            :enable = true
            :Update()                  // ルートウィンドウの表示更新
        End With

        Event:Wait()
End Method

Method  OnButton1(sender)
Begin
        TextField<"TextField1">:SetFocus()
End Method

Method  OnButton2(sender)
Begin
        TextField<"TextField2">:SetFocus()
End Method
```

TextBox	複数行のテキストを表示、入力を設定します。 タグ指定コントロールです。

プロパティ

名称	説明	範囲	初期値	代入
parent※1	親Widgetのタグ名	Widgetのタグ名	nil	不可
frame※1	枠種別	“none”:なし “thin”:薄型 “raised”:厚型 “recessed”凹型	'recessed'	可
foreColor※1	前景色	RGBの文字列 (“0~255|0~255|0~255”)	'0|0|0'	可
backColor※1	背景色	RGBの文字列 (“0~255|0~255|0~255”)	“255|255|255”	可
selTextColor※1	選択行の文字色	RGBの文字列 (“0~255|0~255|0~255”)	“255|255|255”	可
selBackColor※1	選択行の背景色	RGBの文字列 (“0~255|0~255|0~255”)	“0|0|184”	可
focusColor※1	フォーカス枠色	RGBの文字列 (“0~255|0~255|0~255”)	“000|255|000”	可
visible※1	表示／非表示	true:表示 false:表示しない	false	可
enable※1	有効／無効	true:有効 false:無効	true	可
left※1	最左X座標	0~239	0	可
top※1	最上Y座標	0~1279	0	可
width※1	幅	1~240(ドット)	80	可
height※1	高さ	1~1280(ドット)	24	可
fontSize※1	フォントサイズ	12:12ドットフォント 16:16ドットフォント 24:24ドットフォント	12	可
text※1	テキスト 入力完了後は、入力文字列が格納されます。	文字列 (最大8192バイト)	Widgetのタグ名	可
textAlign※1	テキストの表示位置	“left”:左詰め “center”:中央 “right”:右詰め	“left”	可
showPosy	TextBox内の表示開始位置Y座標 「表示位置の指定(showPosyプロパティ、Scrollメソッド)」(7-37ページ)	0~1279 (TextBox上のy座標)	0	可
scrollBarV	縦スクロールバーの表示	true:表示する false:表示しない	false	可
maxLength	文字列入力するときの最大文字列長	0~8192(バイト): 半角で8192文字まで、 全角で4096文字まで	8192	可
editDirect	フォーカスインしたときに入力状態にします。	true:有効 false:無効	false	可

名称	説明	範囲	初期値	代入
enableBCR	トリガモードの設定 バーコード読み取りするときのモードを設定します。 バーコードを読み取りしない場合には、「nil」を指定します。	nil:バーコード入力しない 1:オートオフ 2:連続読み取り 3:モーメンタリ 4:ソフトウエアトリガ 5:離して読み 6:ダブルクリック	1	可
inputOption	入力オプションの設定 入力するときの動作を設定します。 📖「日本語を入力する(inputOptionプロパティ)」(7-26ページ) 📖「電卓機能(inputOptionプロパティ)」(7-30ページ)	0:なし 1:バーコードリーダのみの入力 2:日本語入力をする 4:電卓から入力する 8:シフトキーで切り替え(初期無効) 16:シフトキーで切り替え(初期有効)	0	可
fontBold	太字設定の有効/無効 BT-W100/W80/W70シリーズのみ有効です。	true/false	true	可
fontSize	フォントサイズ BT-W100/W80/W70シリーズのみ有効です。	6～255	-	可
fontName	フォント名 BT-W100/W80/W70シリーズのみ有効です。	32文字以内のフォント名	""	可
textVAlign	テキストの表示位置(上下)	"top"(上端揃え) "bottom"(下端揃え) "center"(上下中央揃え)	"top"	可
simpleWidget	Simple Widget有効/無効	true/false	false	可
inFocusTextColor※2	フォーカス時の文字色	RGBの文字列 ("0～255 | 0～255 | 0～255")	"0 | 0 | 0"	可
inFocusBackColor※2	フォーカス時の背景色	RGBの文字列 ("0～255 | 0～255 | 0～255")	"192 | 192 | 192"	可
outFocusTextColor※2	非フォーカス時の文字色	RGBの文字列 ("0～255 | 0～255 | 0～255")	"0 | 0 | 0"	可
outFocusBackColor※2	非フォーカス時の背景色	RGBの文字列 ("0～255 | 0～255 | 0～255")	"255 | 255 | 255"	可

※1 Widget共通プロパティです。他のWidget操作コントロールでも使用します。ここでは、Widget共通プロパティの詳細説明を省略しています。
Widget共通プロパティの詳細説明については、📖「Widget共通プロパティ」(7-5ページ)を参照してください。

※2 simpleWidgetプロパティがtrue、frameプロパティが"none"の場合のみ有効です。

イベントプロパティ

名称	発生タイミング	sender
onFocusIn※1	TextBoxにフォーカスが入ったとき	TextBoxのタグ名
onFocusOut※1	TextBoxからフォーカスが抜けたとき	TextBoxのタグ名
onTouchUp※1	タッチパネルからリリースしたとき BT-W100/W80/W70シリーズのみ有効です。	TextBoxのタグ名
onEditStart	入力を開始したとき	TextBoxのタグ名
onEditEnd	入力を完了したとき キー入力で「ENT」を押して入力を確定したとき、またはトリガーキーを押して自動的にバーコードを読み取ったとき	TextBoxのタグ名
onEditCancel	入力をキャンセルしたとき	TextBoxのタグ名
onTouchOut	タッチパネルを押して入力状態を解除したとき BT-W100/W80/W70シリーズのみ有効です。	TextBoxのタグ名
onScanComplete※2	バーコードの読み取りが完了したとき	TextBoxのタグ名

※1 Widget共通イベントプロパティです。他のWidget操作コントロールでも使用します。ここでは、Widget共通イベントプロパティの詳細説明を省略しています。
Widget共通イベントプロパティの詳細説明については、「Widget共通イベントプロパティ」(7-7ページ)を参照してください。

※2 バーコード読み取りが完了し、TextBoxに読み取りデータが表示される場合に発生します。
読み取り異常の場合は、BCRコントロールのイベントプロパティから取得できます。
- タイムアウト　:onScanTimeout
- 読み取りエラー:onScanError

メソッド

名称	説明	引数	戻り値
Create※1	TextBoxを作成します。	targetWidget:作成するWidgetの所属先になるWindowのタグ名	true:正常終了 false:エラー終了
Copy※1	TextBoxをコピーします。	targetWidget:複製Widgetの所属先になるWindowのタグ名 copyWidget:コピー元になるTextBoxのタグ名	true:正常終了 false:エラー終了
Delete※1	TextBoxを削除します。	なし	true:正常終了 false:エラー終了
Update※1	TextBoxの表示を最新情報に更新します。	なし	true:正常終了 false:エラー終了
SetFocus※1	TextBoxにフォーカスを当てます。	なし	true:正常終了 false:エラー終了
GetFocus※1	現在フォーカスが当たっているWidgetのタグ名を取得します。	なし	Widgetのタグ名
Scroll※2	TextBox内の表示を縦スクロールします。 「表示位置の指定(showPosyプロパティ、Scrollメソッド)」(7-37ページ)	x:x座標(0固定) y:y座標 (TextBox上のy座標)	true:正常終了 false:エラー終了

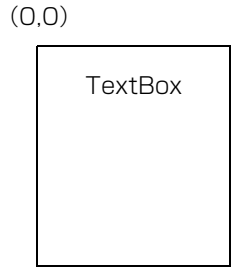
名称	説明	引数	戻り値
SetEditMode	各状態(フォーカスイン･アウト)から入力状態へ遷移します。逆は入力状態からフォーカスイン状態に遷移させます。	mode: “on”:入力状態へ遷移 “off”:フォーカスイン状態に遷移	true:正常終了 false:エラー終了

※1 Widget共通メソッドです。他のWidget操作コントロールでも使用します。ここでは、Widget共通メソッドの詳細説明を省略しています。
Widget 共通メソッドの詳細説明については、「Widget 共通メソッド」(7-8 ページ)を参照してください。

※2 x座標は「0」固定です。0以外の値を代入するとfalseが返ります。
y座標に表示不可能な座標を指定しても、正常終了になります。このときの表示位置は、スクロール最大位置になります。

■表示位置の指定(showPosyプロパティ、Scrollメソッド)

次の座標で指定します。



最大X座標＝TextBoxの幅-1
最大Y座標＝TextBoxの高さ-1

使用例

例1　TextBoxを使用して、バーコード読み取りをおこないます。

```
Method ReadBarcode()
    Begin

    With Label< "表示" >                          // Labelのタグ名に「表示」を設定
        :Create( "ROOT_WINDOW" )                  // ルートウィンドウ上にLabelを作成

        // バーコード読み取りのメッセージ表示
        :text = "バーコードを読み取ってください。¥n
          TRGキー押下で読み取り開始¥n(Cキー長押しでキャンセル)"
        :width = 240                              // 幅は240ドット
        :height = 60                              // 高さは60ドット
        :visible = true
        :enable = true
    End With

    With TextBox< "入力" >                        // TextBoxのタグ名に「入力」を設定
        :Create( "ROOT_WINDOW" )                  /* ルートウィンドウ上にTextBoxを
                                                     作成 */
        :top = 70                                 /* ルートウィンドウ上にTextBoxを
                                                     表示する最上Y座標＝70 */
        :left = 10                                /* ルートウィンドウ上にTextBoxを
                                                     表示する最左X座標＝10 */
        :width = 200                              // 幅は200ドット
        :height = 40                              // 高さは40ドット
        :enableBCR = 1                            /* バーコード読み取り時のモードは
                                                     オートオフ */
        :visible = true
        :enable = true
    End With

    With Window< "ROOT_WINDOW" >
        :enable = true
        :Update()                                 // ルートウィンドウの表示更新
    End With

    Event:Wait()                                  // イベント取得

End Method
```

例2　editDirectプロパティを使用して、フォーカスインしたときに入力状態にするかしないかを変更します。

```
Method main()
Begin

    With Button<"Button1">
        :Create("ROOT_WINDOW")
        :foreColor =  "0¦0¦0"
        :left = 62
        :top = 38
        :height =  24
        :width = 117
        :fontSize =  12
        :textAlign = "center"
        :text =  "editDirectあり"
        :visible=True
        :enable=True
    EndWith

    With Button<"Button2">
        :Create("ROOT_WINDOW")
        :foreColor =  "0¦0¦0"
        :left = 62
        :top = 170
        :height =  24
        :width = 117
        :fontSize =  12
        :textAlign = "center"
        :text =  "editDirectなし"
        :visible=True
        :enable=True
    EndWith

    With TextBox<"TextBox1">
        :Create("ROOT_WINDOW")
        :foreColor =  "000¦000¦000"
        :backColor =  "255¦255¦192"
        :left = 52
        :top = 95
        :height =  24
        :width = 140
        :fontSize =  16
        :textAlign = "left"
        :editDirect = true                    // editDirectをtrueにする
        :visible = true
        :enable = true
```

```
        EndWith

        With TextBox<"TextBox2">
            :Create("ROOT_WINDOW")
            :foreColor =  "0¦0¦0"
            :backColor =  "255¦255¦192"
            :left = 52
            :top = 230
            :height =  24
            :width = 140
            :fontSize =  16
            :textAlign = "left"
            :editDirect = false                    // editDirectをfalseにする
            :visible = true
            :enable = true
        EndWith

        Button<"Button1">:onButton = OnButton1
        Button<"Button2">:onButton = OnButton2

        With Window< "ROOT_WINDOW" >
            :enable = true
            :Update()                  // ルートウィンドウの表示更新
        End With

        Event:Wait()
End Method

Method  OnButton1(sender)
Begin
        TextBox<"TextBox1">:SetFocus()
End Method

Method  OnButton2(sender)
Begin
        TextBox<"TextBox2">:SetFocus()
End Method
```

Label	文字列の表示を設定します。 タグ指定コントロールです。

プロパティ

名称	説明	範囲	初期値	代入
parent※1	親Widgetのタグ名	Widgetのタグ名	nil	不可
frame※1	枠種別	“none”:なし “thin”:線枠 “raised”:3D凸型 “recessed”:3D凹型	“recessed”	可
foreColor※1	前景色	RGBの文字列 (“0~255|0~255|0~255”)	“0|0|0”	可
backColor※1	背景色	RGBの文字列 (“0~255|0~255|0~255”)	“255|255|255”	可
visible※1	表示／非表示	true:表示 false:表示しない	false	可
enable※1	有効／無効	true:有効 false:無効	true	可
left※1	最左X座標	0~239	0	可
top※1	最上Y座標	0~1279	0	可
width※1	幅	1~240(ドット)	90	可
height※1	高さ	1~1280(ドット)	20	可
fontSize※1	フォントサイズ	12:12ドットフォント 16:16ドットフォント 24:24ドットフォント	12	可
text※1	テキスト	文字列 (最大8192バイト)	“”	可
textAlign※1	テキストの表示位置	“left”:左詰め “center”:中央 “right”:右詰め	“left”	可
fontBold	太字設定の有効/無効 BT-W100/W80/W70シリーズのみ有効です。	true/false	true	可
fontSize	フォントサイズ BT-W100/W80/W70シリーズのみ有効です。	6~255	-	可
fontName	フォント名 BT-W100/W80/W70シリーズのみ有効です。	32文字以内のフォント名	“”	可
textVAlign	テキストの表示位置(上下)	top”(上端揃え) “bottom”(下端揃え) “center”(上下中央揃え)	“top”	可
simpleWidget	Simple Widget有効/無効	true/false	false	可
inFocusTextColor※2	フォーカス時の文字色	RGBの文字列 (“0~255|0~255|0~255”)	“0|0|0”	可

名称	説明	範囲	初期値	代入
inFocusBackColor※2	フォーカス時の背景色	RGBの文字列 (“0～255 | 0～255 | 0～255”)	“192	192	192”	可
outFocusTextColor※2	非フォーカス時の文字色	RGBの文字列 (“0～255 | 0～255 | 0～255”)	“0	0	0”	可
outFocusBackColor※2	非フォーカス時の背景色	RGBの文字列 (“0～255 | 0～255 | 0～255”)	“255	255	255”	可

※1 Widget共通プロパティです。他のWidget操作コントロールでも使用します。ここでは、Widget共通プロパティの詳細説明を省略しています。
Widget共通プロパティの詳細説明については、「Widget共通プロパティ」(7-5ページ)を参照してください。

※2 simpleWidgetプロパティがtrue、frameプロパティが"none"の場合のみ有効です。

メソッド

名称	説明	引数	戻り値
Create※	Labelを作成します。	targetWidget:作成するWidgetの所属先になるWindowのタグ名	true:正常終了 false:エラー終了
Copy※	Labelをコピーします。	targetWidget:複製Widgetの所属先になるWindowのタグ名 copyWidget:コピー元になるLabelのタグ名	true:正常終了 false:エラー終了
Delete※	Labelを削除します。	なし	true:正常終了 false:エラー終了
Update※	Labelの表示を最新情報に更新します。	なし	true:正常終了 false:エラー終了

※ Widget共通メソッドです。他のWidget操作コントロールでも使用します。ここでは、Widget共通メソッドの詳細説明を省略しています。
Widget共通メソッドの詳細説明については、「Widget共通メソッド」(7-8ページ)を参照してください。

使用例

SubWindow1上にラベルを作成して、「作業指示選択」の文字列を表示します。

```
Method Job1()
    Begin

    With Label<"Label1">                    // Labelのタグ名に「Label1」を設定
        :Create("ROOT_WINDOW")              // ルートウィンドウ上に「Label1」を作成
        :foreColor = "255|255|255"          // 「Label1」の表示文字列の色を設定
        :backColor = "0|102|255"            // 背景色を設定
        :left = 10
        :top = 10
        :width = 220                        // 幅は220ドット
        :height = 50                        // 高さは50ドット
        :fontSize = 24                      // 「Label1」の表示文字列のフォントは
                                               24ドット
        :text = "作業指示選択"              // 「Label1」の表示文字列を設定
        :textAlign = "center"               // 「Label1」の表示文字列の表示位置を設定
        :visible = true
        :enable = true
    End With

    Window<"ROOT_WINDOW">:Update()          // ルートウィンドウの表示更新

    Event:Wait()                            // イベント取得

EndMethod
```

ListBox	リストから選択入力の設定をします。複数選択はできません。 タグ指定コントロールです。

プロパティ

名称	説明	範囲	初期値	代入
parent※1	親Widgetのタグ名	Widgetのタグ名	nil	不可
frame※1	枠種別	“none”:なし “thin”:線枠 “raised”:3D凸型 “recessed”:3D凹型	“thin”	可
foreColor※1	前景色	RGBの文字列 (“0~255|0~255|0~255”)	“0|0|0”	可
backColor※1	背景色	RGBの文字列 (“0~255|0~255|0~255”)	“255|255|255”	可
selTextColor※1	選択行の文字色	RGBの文字列 (“0~255|0~255|0~255”)	“255|255|255”	可
selBackColor※1	選択行の背景色	RGBの文字列 (“0~255|0~255|0~255”)	“0|0|184”	可
focusColor※1	フォーカス枠色	RGBの文字列 (“0~255|0~255|0~255”)	“000|255|000”	可
visible※1	表示／非表示	true:表示 false:表示しない	false	可
enable※1	有効／無効	true:有効 false:無効	false	可
left※1	最左X座標	0~239	0	可
top※1	最上Y座標	0~1279	0	可
width※1	幅	1~240(ドット)	80	可
height※1	高さ	1~1280(ドット)	120	可
fontSize※1	フォントサイズ	12:12ドットフォント 16:16ドットフォント 24:24ドットフォント	12	可
text※1、※2	テキスト リスト項目に登録する文字列を設定できます。	文字列 (最大8192バイト)	“”	可
textAlign※1	テキストの表示位置	“left”:左詰め “center”:中央 “right”:右詰め	“left”	可
showPosy	ListBox内の表示開始位置Y座標 「表示位置の指定(showPosyプロパティ、Scrollメソッド)」(7-47ページ)	0~1279 (ListBox上のy座標)	0	可
count	登録したリスト項目数	0~1024	0	不可
editDirect	フォーカスインしたときに入力状態にします。	true:有効 false:無効	false	可
fontBold	太字設定の有効/無効 BT-W100/W80/W70シリーズのみ有効です。	true/false	true	可

名称	説明	範囲	初期値	代入
fontSize	フォントサイズ BT-W100/W80/W70シリーズのみ有効です。	6～255	-	可
fontName	フォント名 BT-W100/W80/W70シリーズのみ有効です。	32文字以内のフォント名	“”	可
textVAlign	テキストの表示位置(上下)	“top”（上端揃え） “bottom”（下端揃え） “center”（上下中央揃え）	“top”	可
simpleWidget	Simple Widget有効/無効	true/false	false	可
inFocusTextColor※3	フォーカス時の文字色	RGBの文字列 （“0～255 | 0～255 | 0～255”）	“0 | 0 | 0”	可
inFocusBackColor※3	フォーカス時の背景色	RGBの文字列 （“0～255 | 0～255 | 0～255”）	“192 | 192 | 192”	可
outFocusTextColor※3	非フォーカス時の文字色	RGBの文字列 （“0～255 | 0～255 | 0～255”）	“0 | 0 | 0”	可
outFocusBackColor※3	非フォーカス時の背景色	RGBの文字列 （“0～255 | 0～255 | 0～255”）	“255 | 255 | 255”	可

※1 Widget共通プロパティです。他のWidget操作コントロールでも使用します。ここでは、Widget共通プロパティの詳細説明を省略しています。
Widget共通プロパティの詳細説明については、「Widget共通プロパティ」(7-5ページ)を参照してください。

※2 リスト項目を、次の方法で設定できます。
- リストに表示する文字列を"¥n"で区切り、1024項目まで設定できます。
  1025項目以上を指定した場合は、切り捨てられます。
- 全長8192バイトまでです("¥n"も含みます)。8192バイトを超える文字を指定した場合は、実行時例外となります。

※3 simpleWidgetプロパティがtrue、frameプロパティが"none"の場合のみ有効です。

イベントプロパティ

名称	発生タイミング	sender
onFocusIn※	ListBoxにフォーカスが入ったとき	ListBoxのタグ名
onFocusOut※	ListBoxからフォーカスが抜けたとき	ListBoxのタグ名
onTouchUp※	タッチパネルからリリースしたとき BT-W100/W80/W70シリーズのみ有効です。	ListBoxのタグ名
onSelectChange	選択項目を変更したとき	ListBoxのタグ名
onEditStart	項目選択を開始したとき	ListBoxのタグ名
onEditEnd	「ENT」を押して項目選択を確定したとき	ListBoxのタグ名
onEditCancel	項目選択をキャンセルしたとき	ListBoxのタグ名
onTouchOut	タッチパネルを押して項目選択状態を解除したとき BT-W100/W80/W70シリーズのみ有効です。	ListBoxのタグ名

※ Widget共通イベントプロパティです。他のWidget操作コントロールでも使用します。ここでは、Widget共通イベントプロパティの詳細説明を省略しています。
Widget 共通イベントプロパティの詳細説明については、「Widget 共通イベントプロパティ」(7-7ページ)を参照してください。

メソッド

名称	説明	引数	戻り値
Create※1	ListBoxを作成します。	targetWidget:作成するWidgetの所属先になるWindowのタグ名	true:正常終了 false:エラー終了
Copy※1	ListBoxをコピーします。	targetWidget:複製Widgetの所属先になるWindowのタグ名 copyWidget:コピー元になるListBoxのタグ名	true:正常終了 false:エラー終了
Delete※1	ListBoxを削除します。	なし	true:正常終了 false:エラー終了
Update※1	ListBoxの表示を最新情報に更新します。	なし	true:正常終了 false:エラー終了
SetFocus※1	ListBoxにフォーカスを当てます。	なし	true:正常終了 false:エラー終了
GetFocus※1	現在フォーカスが当たっているWidgetのタグ名を取得します。	なし	Widgetのタグ名
Add	リストの最後に項目を追加します。	item:リストに表示する文字列※3	true:正常終了 false:エラー終了
Insert	指定するリスト位置にアイテムを追加します。	index:リスト位置※4 (0～(リスト項目数-1)) item:リストに表示する文字列※3	true:正常終了 false:エラー終了
Select	指定するリスト位置を選択状態にします。	index:リスト位置 (0～(リスト項目数-1))	true:正常終了 false:エラー終了
Remove	指定するリスト位置にある項目を削除します。	index:リスト位置 (0～(リスト項目数-1)、-1を指定すると、全項目の削除)	true:正常終了 false:エラー終了

名称	説明	引数	戻り値
GetItem	指定するリスト位置の表示文字列を取得します。	index:リスト位置 (0～(リスト項目数-1))	item:表示文字列 false:エラー終了
GetSelectedIndex	選択されたアイテムのリスト位置を取得します。	なし	index:リスト位置 (0～(リスト項目数-1)) false:エラー終了
Scroll※2	ListBox内の表示を縦スクロールします。 「表示位置の指定(showPosyプロパティ、Scrollメソッド)」(7-47ページ)	x:x座標(0固定) y:y座標 (ListBox上のy座標)	true:正常終了 false:エラー終了
SetEditMode	各状態(フォーカスイン・アウト)から入力状態へ遷移します。逆は入力状態からフォーカスイン状態に遷移させます。	mode: "on":入力状態へ遷移 "off":フォーカスイン状態に遷移	true:正常終了 false:エラー終了

※1 Widget共通メソッドです。他のWidget操作コントロールでも使用します。ここでは、Widget共通メソッドの詳細説明を省略しています。
Widget 共通メソッドの詳細説明については、「Widget 共通メソッド」(7-8 ページ)を参照してください。

※2 x座標は「0」固定です。0以外の値を代入するとfalseが返ります。
y座標に表示不可能な座標を指定しても、正常終了になります。このときの表示位置は、スクロール最大位置になります。

※3 itemに設定する個々の文字列は、最大8192バイトまでです。

※4 indexは、最大1024個までです。

■表示位置の指定(showPosyプロパティ、Scrollメソッド)

次の座標で指定します。

最大X座標＝ListBoxの幅-1
最大Y座標＝ListBoxの高さ-1

使用例

例1　SubWindow1上にリストボックスを作成して、リスト項目を登録します。

```
Method Job()
    Begin

    With ListBox<"ListBox1">                    // ListBoxのタグ名に「ListBox1」を設定
        :Create("ROOT_WINDOW")                  // ルートウィンドウ上に「ListBox1」を作成
        :foreColor = "0|0|0"                    // リスト項目の表示文字色を設定
        :backColor = "201|235|67"               // 背景色の設定
        :left = 30                              /* ルートウィンドウ上に「ListBox1」を
                                                   表示する最左X座標＝30 */
        :top = 50                               /* ルートウィンドウ上に「ListBox1」を
                                                   表示する最上Y座標＝50 */
        :width = 150                            // 幅は150ドット
        :height = 80                            // 高さは80ドット
        :fontSize = 16                          // リスト項目のフォント種類を設定
        :textAlign = "left"                     // リスト項目の文字列の表示位置

        // リスト項目の登録
        :Add("はさみ")
        :Add("ステープラ")
        :Add("セロハンテープ")
        :Add("ノート")
        :Add("えんぴつ")
        :Add("けしゴム")
        :visible = true
        :enable = true

        // イベント登録
        /*
        :onFocusIn = ListBox1_OnFocusIn
        :onFocusOut = ListBox1_OnFocusOut
        :onSelectChange = ListBox1_OnSelectChange
        :onEditEnd = ListBox1_OnEditEnd
        :onEditCancel = ListBox1_OnEditCancel
        :onTouchOut = ListBox1_OnTouchOut
        */
    End With

    With Window< "ROOT_WINDOW" >
        :enable = true
        :Update()                               // ルートウィンドウの表示更新
    End With

EndMethod
```

例2　editDirectプロパティを使用して、フォーカスインしたときに入力状態にするかしないかを変更します。

```
Method main()
Begin

    With Button<"Button1">
        :Create("ROOT_WINDOW")
        :foreColor = "0¦0¦0"
        :left = 62
        :top = 38
        :height = 24
        :width = 117
        :fontSize = 12
        :textAlign = "center"
        :text = "editDirectあり"
        :visible=True
        :enable=True
    EndWith

    With Button<"Button2">
        :Create("ROOT_WINDOW")
        :foreColor = "0¦0¦0"
        :left = 62
        :top = 170
        :height = 24
        :width = 117
        :fontSize = 12
        :textAlign = "center"
        :text = "editDirectなし"
        :visible=True
        :enable=True
    EndWith

    With ListBox<"ListBox1">
        :Create("ROOT_WINDOW")
        :foreColor = "0¦0¦0"
        :backColor = "255¦255¦192"
        :left = 62
        :top = 67
        :height = 74
        :width = 117
        :fontSize = 12
        :textAlign = "left"
        :editDirect = true                    // editDirectをtrueにする

        // リスト項目の登録
        :Add("はさみ")
        :Add("ステープラ")
```

```
            :Add("セロハンテープ")

            :visible = true
            :enable = true
        EndWith

        With ListBox<"ListBox2">
            :Create("ROOT_WINDOW")
            :foreColor =  "0¦0¦0"
            :backColor =  "255¦255¦192"
            :left = 62
            :top = 207
            :height =  74
            :width = 117
            :fontSize =  12
            :textAlign = "left"
            :editDirect = false                    // editDirectをfalseにする

            // リスト項目の登録
            :Add("ノート")
            :Add("えんぴつ")
            :Add("けしゴム")

            :visible = true
            :enable = true
        EndWith

        Button<"Button1">:onButton = OnButton1
        Button<"Button2">:onButton = OnButton2

        With Window< "ROOT_WINDOW" >
            :enable = true
            :Update()              // ルートウィンドウの表示更新
        End With

        Event:Wait()
End Method

Method  OnButton1(sender)
Begin
        ListBox<"ListBox1">:SetFocus()
End Method

Method  OnButton2(sender)
Begin
        ListBox<"ListBox2">:SetFocus()
End Method
```

ComboBox	コンボボックスから選択入力の設定をします。 複数選択はできません。タグ指定コントロールです。

プロパティ

名称	説明	範囲	初期値	代入
parent※1	親Widgetのタグ名	Widgetのタグ名	nil	不可
frame※1	枠種別	“none”:なし “thin”:線枠 “raised”:3D凸型 “recessed”:3D凹型	“thin”	可
foreColor※1	前景色	RGBの文字列 (“0~255|0~255|0~255”)	“0|0|0”	可
backColor※1	背景色	RGBの文字列 (“0~255|0~255|0~255”)	“255|255|255”	可
selTextColor※1	選択行の文字色	RGBの文字列 (“0~255|0~255|0~255”)	“255|255|255”	可
selBackColor※1	選択行の背景色	RGBの文字列 (“0~255|0~255|0~255”)	“0|0|184”	可
focusColor※1	フォーカス枠色	RGBの文字列 (“0~255|0~255|0~255”)	“000|255|000”	可
visible※1	表示／非表示	true:表示 false:表示しない	false	可
enable※1	有効／無効	true:有効 false:無効	false	可
left※1	最左X座標	0~239	0	可
top※1	最上Y座標	0~1279	0	可
width※1	幅	1~240(ドット)	80	可
height※1	高さ	1~1280(ドット)	120	可
text※1、※2	テキスト リスト項目に登録する文字列を設定できます。	文字列 (最大8192バイト)	“”	可
fontSize※1	フォントサイズ	12:12ドットフォント 16:16ドットフォント 24:24ドットフォント	12	可
textAlign※1	テキストの表示位置	"left":左詰め "center":中央 "right":右詰め	“left”	可
count	登録したアイテム数	0~1024	0	不可
editDirect	フォーカスインしたときに入力状態にします。	true:有効 false:無効	false	可
fontBold	太字設定の有効/無効 BT-W100/W80/W70シリーズのみ有効です。	true/false	true	可

名称	説明	範囲	初期値	代入
fontSize	フォントサイズ BT-W100/W80/W70シリーズのみ有効です。	6～255	-	可
fontName	フォント名 BT-W100/W80/W70シリーズのみ有効です。	32文字以内のフォント名	""	可
textVAlign	テキストの表示位置(上下)	"top"（上端揃え） "bottom"（下端揃え） "center"（上下中央揃え）	"top"	可
simpleWidget	Simple Widget有効/無効	true/false	false	可
inFocusTextColor※3	フォーカス時の文字色	RGBの文字列 （"0～255 | 0～255 | 0～255"）	"0 | 0 | 0"	可
inFocusBackColor※3	フォーカス時の背景色	RGBの文字列 （"0～255 | 0～255 | 0～255"）	"192 | 192 | 192"	可
outFocusTextColor※3	非フォーカス時の文字色	RGBの文字列 （"0～255 | 0～255 | 0～255"）	"0 | 0 | 0"	可
outFocusBackColor※3	非フォーカス時の背景色	RGBの文字列 （"0～255 | 0～255 | 0～255"）	"255 | 255 | 255"	可

※1 Widget 共通プロパティです。他の Widget 操作コントロールでも使用します。ここでは、Widget 共通プロパティの詳細説明を省略しています。
Widget 共通プロパティの詳細説明については、「Widget 共通プロパティ」(7-5 ページ)を参照してください。

※2 リスト項目を、次の方法で設定できます。
- リストに表示する文字列を"¥n"で区切り、1024項目まで設定できます。
  1025項目以上を指定した場合は、切り捨てられます。
- 全長、8192バイトまでです("¥n"も含みます)。
  8192バイトを超える文字を指定した場合は、実行時例外となります。

※3 simpleWidgetプロパティがtrue、frameプロパティが"none"の場合のみ有効です。

イベントプロパティ

名称	発生タイミング	sender
onFocusIn※	ComboBoxにフォーカスが入ったとき	ComboBoxのタグ名
onFocusOut※	ComboBoxからフォーカスが抜けたとき	ComboBoxのタグ名
onTouchUp※	タッチパネルからリリースしたとき BT-W100/W80/W70シリーズのみ有効です。	ComboBoxのタグ名
onSelectChange	選択項目を変更したとき	ComboBoxのタグ名
onEditStart	項目選択を開始したとき	ComboBoxのタグ名
onEditEnd	「ENT」を押して項目選択を確定したとき	ComboBoxのタグ名
onEditCancel	項目選択をキャンセルしたとき	ComboBoxのタグ名
onTouchOut	タッチパネルを押して項目選択状態を解除したとき BT-W100/W80/W70シリーズのみ有効です。	ComboBoxのタグ名

※ Widget共通イベントプロパティです。他のWidget操作コントロールでも使用します。ここでは、Widget共通イベントプロパティの詳細説明を省略しています。
Widget 共通イベントプロパティの詳細説明については、「Widget 共通イベントプロパティ」(7-7ページ)を参照してください。

メソッド

名称	説明	引数	戻り値
Create※1	ComboBoxを作成します。	targetWidget:作成するWidgetの所属先になるWindowのタグ名	true:正常終了 false:エラー終了
Copy※1	ComboBoxをコピーします。	targetWidget:複製Widgetの所属先になるWindowのタグ名 copyWidget:コピー元になるComboBoxのタグ名	true:正常終了 false:エラー終了
Delete※1	ComboBoxを削除します。	なし	true:正常終了 false:エラー終了
Update※1	ComboBoxの表示を最新情報に更新します。	なし	true:正常終了 false:エラー終了
SetFocus※1	ComboBoxにフォーカスを当てます。	なし	true:正常終了 false:エラー終了
GetFocus※1	現在フォーカスが当たっているWidgetのタグ名を取得します。	なし	Widgetのタグ名
Add	リストの最後にアイテムを追加します。	item:追加アイテムの文字列※2	true:正常終了 false:エラー終了
Insert	リストの指定する位置にアイテムを追加します。	index:リスト位置※3 (0～(アイテム数-1)) item:追加アイテムの文字列※2	true:正常終了 false:エラー終了
Select	指定するリスト位置を選択状態にします。	index:リスト位置 (0～(アイテム数-1))	true:正常終了 false:エラー終了
Remove	指定するリスト位置にあるアイテムを削除します。	index:リスト位置 (0～(アイテム数-1)、-1を指定すると、全アイテムの削除)	true:正常終了 false:エラー終了
GetItem	指定するリスト位置のアイテムを取得します。	index:リスト位置 (0～(アイテム数-1))	item:アイテムの文字列 false:エラー終了
GetSelectedIndex	選択されたアイテムのリスト位置を取得します。	なし	index:リスト位置 (0～(アイテム数-1)) false:エラー終了
SetEditMode	各状態(フォーカスイン･アウト)から入力状態へ遷移します。逆は入力状態からフォーカスイン状態に遷移させます。	mode: “on”:入力状態へ遷移 “off”:フォーカスイン状態に遷移	true:正常終了 false:エラー終了

※1 Widget共通メソッドです。他のWidget操作コントロールでも使用します。ここでは、Widget共通メソッドの詳細説明を省略しています。
Widget共通メソッドの詳細説明については、「Widget共通メソッド」(7-8ページ)を参照してください。

※2 itemに設定する個々の文字列は、最大8192バイトまでです。

※3 indexは、最大1024個までです。

使用例

例1　SubWindow1上にコンボボックスを作成して、リスト項目を登録します。

```
Method Job()
    Begin

    With ComboBox<"ComboBox1">      // ComboBoxのタグ名に「CombotBox1」を設定
        :Create("ROOT_WINDOW")      // ルートウィンドウ上に「CombotBox1」を作成
        :foreColor = "0|0|0"        // リスト項目の表示文字色を設定
        :backColor = "201|235|67"   // 背景色の設定
        :left = 30                  /* ルートウィンドウ上に「CombotBox1」を
                                       表示する最左X座標＝30 */
        :top = 50                   /* ルートウィンドウ上に「CombotBox1」を
                                       表示する最上Y座標＝50 */
        :width = 150                // 幅は150ドット
        :height = 80                // 高さは80ドット
        :fontSize = 16              // リスト項目のフォント種類を設定
        :textAlign = "left"         // リスト項目の文字列の表示位置

        // リスト項目の登録
        :Add("はさみ")
        :Add("ステープラ")
        :Add("セロハンテープ")
        :Add("ノート")
        :Add("えんぴつ")
        :Add("けしゴム")
        :visible = true
        :enable = true

        // イベント登録
        /*
        :onFocusIn = ComboBox1_OnFocusIn
        :onFocusOut = ComboBox1_OnFocusOut
        :onSelectChange = ComboBox1_OnSelectChange
        :onEditEnd = ComboBox1_OnEditEnd
        :onEditCancel = ComboBox1_OnEditCancel
        :onTouchOut = ComboBox1_OnTouchOut
        */
    End With

    With Window< "ROOT_WINDOW" >
        :enable = true
        :Update()                   // ルートウィンドウの表示更新
    End With

EndMethod
```

例2 editDirectプロパティを使用して、フォーカスインしたときに入力状態にするかしないかを変更します。

```
Method main()
Begin

    With Button<"Button1">
        :Create("ROOT_WINDOW")
        :foreColor = "0¦0¦0"
        :left = 62
        :top = 38
        :height = 24
        :width = 117
        :fontSize = 12
        :textAlign = "center"
        :text = "editDirectあり"
        :visible=True
        :enable=True
    EndWith

    With Button<"Button2">
        :Create("ROOT_WINDOW")
        :foreColor = "0¦0¦0"
        :left = 62
        :top = 170
        :height = 24
        :width = 117
        :fontSize = 12
        :textAlign = "center"
        :text = "editDirectなし"
        :visible=True
        :enable=True
    EndWith

    With ComboBox<"ComboBox1">
        :Create("ROOT_WINDOW")
        :foreColor = "0¦0¦0"
        :backColor = "255¦255¦192"
        :left = 62
        :top = 67
        :height = 74
        :width = 117
        :fontSize = 12
        :textAlign = "left"
        :editDirect = true                    // editDirectをtrueにする

        // リスト項目の登録
        :Add("はさみ")
        :Add("ステープラ")
```

```
        :Add("セロハンテープ")

        :visible = true
        :enable = true
    EndWith

    With ComboBox<"ComboBox2">
        :Create("ROOT_WINDOW")
        :foreColor =  "0¦0¦0"
        :backColor =  "255¦255¦192"
        :left = 62
        :top = 207
        :height =  74
        :width = 117
        :fontSize =  12
        :textAlign = "left"
        :editDirect = false                    // editDirectをfalseにする

        // リスト項目の登録
        :Add("ノート")
        :Add("えんぴつ")
        :Add("けしゴム")

        :visible = true
        :enable = true
    EndWith

    Button<"Button1">:onButton = OnButton1
    Button<"Button2">:onButton = OnButton2

    With Window< "ROOT_WINDOW" >
        :enable = true
        :Update()              // ルートウィンドウの表示更新
    End With

    Event:Wait()
End Method

Method  OnButton1(sender)
Begin
    ComboBox<"ComboBox1">:SetFocus()
End Method

Method  OnButton2(sender)
Begin
    ComboBox<"ComboBox2">:SetFocus()
End Method
```

GridBox	表形式のテキスト表示、入力を設定します。

表形式のテキスト表示、入力を行います。タグ指定コントロールです。

プロパティ

名称	説明	範囲	初期値	代入
parentt※1	親Widgetのタグ名	Widgetのタグ名	nil	不可
frame※1	枠種別	"none"：なし "thin"：線枠 "raised"：3D凸型 "recessed"：3D凸型	raised	可
foreColor※1	前景色	RGBの文字列 ("0～255|0～255|0～255")	"255|255|255"	可
selTextColor※1	選択セルの文字色	RGBの文字列 ("0～255|0～255|0～255")	"255|255|255"	可
backColor※1	背景色	RGBの文字列 ("0～255|0～255|0～255")	"255|255|255"	可
selBackColor※1	選択セルの背景色	RGBの文字列 ("0～255|0～255|0～255")	"255|255|255"	可
focusColor※1	フォーカス枠色	RGBの文字列 ("0～255|0～255|0～255")	"000|255|000"	可
visible※1	表示/非表示	true：表示 false：表示しない	false	可
enable※1	有効/無効	true：有効 false：無効	true	可
left※1	最左X座標	0～239	0	可
top※1	最上Y座標	0～1279	0	可
width※1	幅	1～240(ドット)	80	可
height※1	高さ	1～1280(ドット)	80	可
fontSize※1	フォントサイズ	12:12ドットフォント 16:12ドットフォント 24:12ドットフォント	12	可
text※1	テキスト	文字列	""	可
textLen	テキスト長	0～8192	0	不可
textAlign※1	テキストの表示位置	"left" "center" "right"	"left"	可
editDirect	フォーカスインした時にセルの選択可能状態にします。	true：有効 false：無効	false	可
rHeaderVisible	行ヘッダの表示非表示	true/false	false	可
cHeaderVisible	列ヘッダの表示非表示	true/false	false	可

名称	説明	範囲	初期値	代入
oddTextColor※3	奇数行の文字色	RGBの文字列 (“0~255 | 0~255 | 0~255”)	“0 | 0 | 0”	可
evnTextColor※3	偶数行の文字色	RGBの文字列 (“0~255 | 0~255 | 0~255”)	“0 | 0 | 0”	可
rHeaderTextColor	行ヘッダの文字色	RGBの文字列 (“0~255 | 0~255 | 0~255”)	“0 | 0 | 0”	可
cHeaderTextColor	列ヘッダの文字色	RGBの文字列 (“0~255 | 0~255 | 0~255”)	“0 | 0 | 0”	可
oddBackColor※3	奇数行の背景色	RGBの文字列 (“0~255 | 0~255 | 0~255”)	“255 | 255 | 255”	可
evnBackColor※3	偶数行の背景色	RGBの文字列 (“0~255 | 0~255 | 0~255”)	“255 | 255 | 255”	可
rHeaderBackColor	行ヘッダの背景色	RGBの文字列 (“0~255 | 0~255 | 0~255”)	“192 | 192 | 192”	可
cHeaderBackColor	列ヘッダの背景色	RGBの文字列 (“0~255 | 0~255 | 0~255”)	“192 | 192 | 192”	可
rHeaderTextAlign	行ヘッダのテキスト配置	“left” “center” “right”	“left”	可
cHeaderTextAlign	列ヘッダのテキスト配置	“left” “center” “right”	“left”	可
gridColor	GridBoxの枠線色	RGBの文字列 (“0~255 | 0~255 | 0~255”)	“192 | 192 | 192”	可
cellFrame※2	セルの枠形状	"none”:なし "thin":線枠 “raised”:3D凸型 “recessed”:3D凸型	“thin”	可
cellColorMode	セルの背景色/文字色の設定モード	“altrow”:偶数行/奇数行で切替 “altcol”:偶数列/奇数列で切替 “manual”:各行/各列設定	“altrow”	可
showPosx	GridBox内の表示開始位置Y座標 (セル単位)	表示開始セルの列番号(0~)	0	可
showPosy	GridBox内の表示開始位置X座標 (セル単位)	表示開始セルの行番号(0~)	0	可
simpleWidget	Simple Widget有効/無効	true/false	false	可
inFocusTextColor※4	フォーカス時の文字色	RGBの文字列 (“0~255 | 0~255 | 0~255”)	“0 | 0 | 0”	可
inFocusBackColor※4	フォーカス時の背景色	RGBの文字列 (“0~255 | 0~255 | 0~255”)	“192 | 192 | 192”	可
outFocusTextColor※4	非フォーカス時の文字色	RGBの文字列 (“0~255 | 0~255 | 0~255”)	“0 | 0 | 0”	可

名称	説明	範囲	初期値	代入
outFocusBackColor※4	非フォーカス時の背景色	RGBの文字列（“0～255｜0～255｜0～255”）	“255｜255｜255”	可

※1 Widget共通プロパティです。他のWidget操作コントロールでも使用します。ここでは、Widget共通プロパティの詳細説明を省略しています。
Widget共通プロパティの詳細説明については、「Widget共通プロパティ」(7-5ページ)を参照してください。

※2 cellFrameプロパティの“raised”/“recessed”設定はsimpleWidgetプロパティがfalseの場合のみ有効です。

※3 simpleWidgetプロパティがtrue、cellColorModeプロパティが”altrow”, “altcol”設定の場合のみ有効です。

※4 simpleWidgetプロパティがtrue、frameプロパティが”none”の場合のみ有効です

イベントプロパティ

名称	発生タイミング	Sender
onFocusIn※	GridBoxにフォーカスが入ったとき	GridBoxのタグ名
onFocusOut※	GridBoxにフォーカスが抜けたとき	GridBoxのタグ名
onTouchUp※	タッチパネルが押されている状態から解放されたとき	GridBoxのタグ名
onEditStart	入力を開始したとき	GridBoxのタグ名
onEditEnd	入力を完了したとき キー入力で「ENT」を押して入力確定したとき、またはトリガーキーを押して自動的にバーコードを読取った時	GridBoxのタグ名
onEditCancel	入力をキャンセルしたとき	GridBoxのタグ名
onTouchOut	タッチパネルを押して入力状態を解除したとき	GridBoxのタグ名
onSelectChange	選択項目を変更したとき。	GridBoxのタグ名

※ Widget 共通イベントプロパティです。他の Widget 操作コントロールでも使用します。ここでは、Widget共通イベントプロパティの詳細説明を省略しています。
Widget 共通イベントプロパティの詳細説明については、「Widget 共通プロパティ」(7-5 ページ)を参照してください。

メソッド

名称	説明	引数	戻り値
CreateGrid	GridBoxを作成します。	targetWidget:作成するWidgetの所属先になるWindowのタグ名 row:行数 col:列数	true:正常終了 false:エラー終了
Copy※	GridBoxをコピーします。	targetWidget:複製Widgetの所属先になるWindowのタグ名 copyWidget:コピー元になるGridBoxのタグ名	true:正常終了 false:エラー終了
Delete※	GridBoxを削除します。	なし	true:正常終了 false:エラー終了
Update※	GridBoxの表示を最新情報に更新します。	なし	true:正常終了 false:エラー終了
SetFocus※	GridBoxにフォーカスを当てます。	なし	true:正常終了 false:エラー終了
GetFocus※	現在フォーカスが当たっているWidgetのタグ名を取得します。	なし	Widgetのタグ名
GetText	指定セルのデータを取得します。	row:指定セルの行番号 col:指定セルの列番号	指定セルのデータ
SetText	指定セルのデータを設定します。	row:指定セルの行番号 col:指定セルの列番号 textdata:データ	true:正常終了 false:エラー終了
GetTextLen	指定セルのデータサイズを取得します。	row:指定セルの行番号 col:指定セルの列番号	指定セルのデータサイズ false:エラー終了
GetCount	Gridの行数または列数を取得します。	target:行/列を指定 “row”, “col”	行数または列数 false:エラー終了
Add	行または列を最後尾に追加します。	target:行/列を指定 “row”, “col”	true:正常終了 false:エラー終了
Insert	行または列を挿入します。	target:行/列を指定 “row”, “col” index:挿入する行/列数を指定	true:正常終了 false:エラー終了
Remove	行または列を削除します。	target:行/列を指定 “row”, “col” index::削除する行/列数を指定	true:正常終了 false:エラー終了
Select	指定したセルを選択します。	row:指定セルの行番号 col:指定セルの列番号	true:正常終了 false:エラー終了
GetSelectedItem	選択セルの位置を取得します。	target:行/列を指定 “row”, “col”	選択セルの行番号または列番号
GetWidth	選択した列の幅を取得します。	index:幅を取得する行/列数を指定	行幅または列幅
SetWidth	選択した列の幅を設定します。	index:幅を設定する行/列数を指定 size:行/列幅	true:正常終了 false:エラー終了

名称	説明	引数	戻り値
GetCellParam	指定したセルの設定を取得します。	row:指定セルの行番号 col:指定セルの列番号 id:設定ID(※) ※端末ライブラリと同じ	value false:エラー終了
SetParam	指定した行または列の設定を変更します。	target:行/列を指定 “row”, “col” index:行/列番号 id:設定ID value:設定する値	true:正常終了 false:エラー終了
SetEditMode	セル選択状態と入力状態を遷移します。	mode: “on”:入力状態へ遷移 “off”:選択状態へ遷移	true:正常終了 false:エラー終了

※ Widget共通メソッドです。他のWidget操作コントロールでも使用します。ここでは、Widget共通メソッドの詳細説明を省略しています。
Widget 共通メソッドの詳細説明については、「Widget 共通メソッド」(7-8 ページ)を参照してください。

■SetParam, GetCellParamで指定可能な設定IDと設定値について

設定ID	説明	設定値	初期値
“textAlign”	テキストの表示位置	“left” “center” “right”	“left”
“backColor”	背景色	RGBの文字列 (“0~255 | 0~255 | 0~255”)	“255 | 255 | 255”
“foreColor”	前景色(文字色)	RGBの文字列 (“0~255 | 0~255 | 0~255”)	“0 | 0 | 0”

使用例

例1　GridBoxの表示･操作のサンプル

```
Method sample_GridBox()
        cnt
        val
        pos_1
        pos_2
Begin
        With GridBox<"grid1">
                // 生成
                :CreateGrid("ROOT_WINDOW", 1, 3)

                // Grid設定
                :top = 20
                :left = 5
                :width = 210
                :height = 100
                :enable = true
                :visible = true
                // フォーカスイン時､セル選択可能状態ON/OFF
                :editDirect = false
                // 枠種別
                :frame = "none"
                // セルの枠形状
                :cellFrame = "thin"
                // GridBoxの枠線色
                :gridColor = "0¦0¦0"
                // GridBox内の表示開始位置Y座標
                :showPosx = 0
                // GridBox内の表示開始位置X座標
                :showPosy = 0
                // 列幅設定
                :SetWidth(0, 40)
                :SetWidth(1, 80)
                :SetWidth(2, 60)

                // 列幅取得(80)
                val = :GetWidth(1)

                // イベント設定
                // 選択項目変更イベント
                :onSelectChange = onSelectChange_sample_GridBox
                // 入力完了イベント
                :onEditEnd = onEditEnd_GridBox
```

```
// 行ヘッダ設定
:rHeaderVisible = false
:rHeaderTextColor = "0¦0¦0"
:rHeaderBackColor = "128¦255¦255"
:rHeaderTextAlign = "left"

// 列ヘッダ設定
:cHeaderVisible = true
:cHeaderTextColor = "0¦0¦0"
:cHeaderBackColor = "128¦255¦255"
:cHeaderTextAlign = "left"
:SetText(-1, 0, "番号")
:SetText(-1, 1, "名前")
:SetText(-1, 2, "在庫数")

// Simple Widget
:simpleWidget = false
:inFocusTextColor = "255¦255¦255"
:inFocusBackColor = "192¦192¦192"
:outFocusTextColor = "255¦255¦255"
:outFocusBackColor = "128¦128¦128"
:cellColorMode = "altrow"
:oddTextColor = "0¦0¦0"
:evnTextColor = "0¦0¦0"
:oddBackColor = "128¦255¦128"
:evnBackColor = "255¦255¦128"

// データ設定
:SetText(0, 0, "2")
:SetText(0, 1, "りんご")
:SetText(0, 2, "50")

// データ追加
:Add("row")
:SetText(1, 0, "3")
:SetText(1, 1, "みかん")
:SetText(1, 2, "30")

// データ挿入
:Insert("row", 0)
:SetText(0, 0, "1")
:SetText(0, 1, "イチゴ")
:SetText(0, 2, "20")

// データ取得("イチゴ")
val = :GetText(0, 1)
```

```
        // データサイズ取得(6)
        val = :GetTextLen(0, 1)

        // 行削除
        :Remove("row", 1)

        // 行数取得(2)
        cnt = :GetCount("row")

        // セル選択
        :Select(0, 1)

        // 選択セル取得
        pos_1 = :GetSelectedItem("row")
        pos_2 = :GetSelectedItem("col")

        // データ取得("イチゴ")
        val = :GetText(pos_1, pos_2)

        // 列ごとの設定
        :SetParam("col", 0, "textAlign", "right")
        :SetParam("col", 0, "backColor", "255¦0¦0")
        :SetParam("col", 0, "foreColor", "255¦255¦255")

        // 列ごとの設定取得
        val = :GetCellParam(1, 0, "textAlign")

        // セル選択状態設定
        :SetEditMode("on")
    End With

    Window<"ROOT_WINDOW">:Update()
    Event:Wait()
End Method

//---------------------------------------
// 選択項目変更イベント
Method onSelectChange_sample_GridBox(sender)
    pos_1
    pos_2
Begin
    With GridBox<sender>
        // "番号"列にカーソルが移動しないように抑制
        pos_1 = :GetSelectedItem("row")
        pos_2 = :GetSelectedItem("col")
        If pos_2 == 0 Then
```

```
                :Select(pos_1, 1)
            EndIf
        End With
End Method

//---------------------------------------
// 入力完了イベント
Method onEditEnd_GridBox(sender)
        pos_1
        pos_2
        val
Begin
        With GridBox<sender>
            // 選択データ表示
            pos_1 = :GetSelectedItem("row")
            pos_2 = :GetSelectedItem("col")
            val = :GetText(pos_1, pos_2)
            Handy:ShowMessageBox(val, "ok" )
        End With
End Method
```

CheckBox	チェックボックスから入力の設定をします。 タグ指定コントロールです。

プロパティ

名称	説明	範囲	初期値	代入
parent※1	親Widgetのタグ名	Widgetのタグ名	nil	不可
foreColor※1	前景色	RGBの文字列 (“0~255 | 0~255 | 0~255”)	"0|0|0"	可
focusColor※1	フォーカス枠色	RGBの文字列 (“0~255 | 0~255 | 0~255”)	"128|128|128"	可
visible※1	表示／非表示	true:表示 false:表示しない	false	可
enable※1	有効／無効	true:有効 false:無効	true	可
left※1	最左X座標	0~239	0	可
top※1	最上Y座標	0~1279	0	可
width※1	幅	1~240(ドット)	90	可
height※1	高さ	1~1280(ドット)	20	可
text※1	テキスト	文字列 (最大8192バイト)	“”	可
fontSize※1	フォントサイズ	12:12ドットフォント 16:16ドットフォント 24:24ドットフォント	12	可
textAlign※1	テキストの表示位置	"left":左詰め "center":中央 "right":右詰め	"left"	可
checked	選択状態	true:チェックあり false:チェックなし	false	可
fontBold	太字設定の有効/無効 BT-W100/W80/W70シリーズのみ有効です。	true/false	true	可
fontSize	フォントサイズ BT-W100/W80/W70シリーズのみ有効です。	6~255	-	可
fontName	フォント名 BT-W100/W80/W70シリーズのみ有効です。	32文字以内のフォント名	“”	可
textVAlign	テキストの表示位置(上下)	“top”(上端揃え) “bottom”(下端揃え) “center”(上下中央揃え)	“top”	可
simpleWidget	Simple Widget有効/無効	true/false	false	可
inFocusTextColor※2	フォーカス時の文字色	RGBの文字列 (“0~255 | 0~255 | 0~255”)	"0|0|0"	可
inFocusBackColor※2	フォーカス時の背景色	RGBの文字列 (“0~255 | 0~255 | 0~255”)	"192|192|192"	可

名称	説明	範囲	初期値	代入
outFocusTextColor※2	非フォーカス時の文字色	RGBの文字列 （“0～255｜0～255｜0～255”）	“0｜0｜0”	可
outFocusBackColor※2	非フォーカス時の背景色	RGBの文字列 （“0～255｜0～255｜0～255”）	“255｜255｜255”	可

※1 Widget共通プロパティです。他のWidget操作コントロールでも使用します。ここでは、Widget共通プロパティの詳細説明を省略しています。
Widget共通プロパティの詳細説明については、「Widget共通プロパティ」(7-5ページ)を参照してください。

※2 simpleWidgetプロパティがtrueの場合のみ有効です。

イベントプロパティ

名称	発生タイミング	sender
onFocusIn※	CheckBoxにフォーカスが入ったとき	CheckBoxのタグ名
onFocusOut※	CheckBoxからフォーカスが抜けたとき	CheckBoxのタグ名
onTouchUp※	タッチパネルからリリースしたとき BT-W100/W80/W70シリーズのみ有効です。	CheckBoxのタグ名
onCheckChange	チェック状態を変更したとき	CheckBoxのタグ名

※ Widget共通イベントプロパティです。他のWidget操作コントロールでも使用します。ここでは、Widget共通イベントプロパティの詳細説明を省略しています。
Widget共通イベントプロパティの詳細説明については、「Widget共通イベントプロパティ」(7-7ページ)を参照してください。

メソッド

名称	説明	引数	戻り値
Create※	CheckBoxを作成します。	targetWidget:作成するWidgetの所属先になるWindowのタグ名	true:正常終了 false:エラー終了
Copy※	CheckBoxをコピーします。	targetWidget:複製Widgetの所属先になるWindowのタグ名 copyWidget:コピー元になるCheckBoxのタグ名	true:正常終了 false:エラー終了
Delete※	CheckBoxを削除します。	なし	true:正常終了 false:エラー終了
Update※	CheckBoxの表示を最新情報に更新します。	なし	true:正常終了 false:エラー終了
SetFocus※	CheckBoxにフォーカスを当てます。	なし	true:正常終了 false:エラー終了
GetFocus※	現在フォーカスが当たっているWidgetのタグ名を取得します。	なし	Widgetのタグ名

※ Widget共通メソッドです。他のWidget操作コントロールでも使用します。ここでは、Widget共通メソッドの詳細説明を省略しています。
Widget共通メソッドの詳細説明については、「Widget共通メソッド」(7-8ページ)を参照してください。

使用例

SubWindow1上にチェックボックスを2つ作成します。

```
Method Job1()
    Begin

    With CheckBox<"CheckBox1">// CheckBoxのタグ名に「CheckBox1」を設定
        :Create("ROOT_WINDOW")          // ルートウィンドウ上に「CheckBox1」を作成
        :foreColor = "0|0|0"            // 文字列の表示色を設定
        :left = 50                      /* ルートウィンドウ上に「CheckBox1」を
                                           表示する最左X座標＝50 */
        :top = 50                       /* ルートウィンドウ上に「CheckBox1」を
                                           表示する最上Y座標＝50 */
        :width = 105                    // 幅は105ドット
        :height = 80                    // 高さは80ドット
        :fontSize = 16                  // 表示文字列のフォント種類を設定
        :text = "全角英数字"            // 「CheckBox1」に表示する文字列
        :textAlign = "left"             // 表示文字列の表示位置
        :visible = true
        :enable = true

        // イベント登録
        //:onCheckChange = CheckBox1_OnCheckChange
    End With

    With CheckBox<"CheckBox2">          // CheckBoxのタグ名に「CheckBox2」を設定
        :Create("ROOT_WINDOW")          // ルートウィンドウ上に「CheckBox2」を作成
        :foreColor = "0|0|0"
        :left = 50
        :top = 150
        :width = 105                    // 幅は105ドット
        :height = 80                    // 高さは80ドット
        :fontSize = 16
        :text = "半角英数字"            // 「CheckBox1」に表示する文字列
        :textAlign = "left"             // 表示文字列の表示位置
        :visible = true
        :enable = true

        // イベント登録
        //:onCheckChange = CheckBox2_OnCheckChange
    End With

    Window<"ROOT_WINDOW">:Update()  // ルートウィンドウの表示更新

    Event:Wait()                        //イベント取得
EndMethod
```

RadioButton	ラジオボタンから入力の設定をします。 タグ指定コントロールです。

プロパティ

名称	説明	範囲	初期値	代入
parent※1	親Widgetのタグ名	Widgetのタグ名	nil	不可
foreColor※1	前景色	RGBの文字列 (“0～255 | 0～255 | 0～255”)	“0	0	0”	可
focusColor※1	フォーカス枠色	RGBの文字列 (“0～255 | 0～255 | 0～255”)	“128	128	128”	可
visible※1	表示／非表示	true:表示 false:表示しない	false	可
enable※1	有効／無効	true:有効 false:無効	true	可
left※1	最左X座標	0～239	0	可
top※1	最上Y座標	0～1279	0	可
width※1	幅	1～240(ドット)	90	可
height※1	高さ	1～1280(ドット)	20	可
text※1	テキスト	文字列(最大8192バイト)	“”	可
fontSize※1	フォントサイズ	12:12ドットフォント 16:16ドットフォント 24:24ドットフォント	12	可
textAlign※1	テキストの表示位置	"left":左詰め "center":中央 "right":右詰め	“left”	可
checked	選択状態	true:チェックあり false:チェックなし	nil	可
groupId	排他グループのID 「排他選択(groupIDプロパティ)」(7-71ページ)	0～32767 (16グループまで指定可能)	0	可
fontBold	太字設定の有効/無効 BT-W100/W80/W70シリーズのみ有効です。	true/false	true	可
fontSize	フォントサイズ BT-W100/W80/W70シリーズのみ有効です。	6～255	-	可
fontName	フォント名 BT-W100/W80/W70シリーズのみ有効です。	32文字以内のフォント名	“”	可
textVAlign	テキストの表示位置(上下)	“top”(上端揃え) “bottom”(下端揃え) “center”(上下中央揃え)	“top”	可
simpleWidget	Simple Widget有効/無効	true/false	false	可
inFocusTextColor※2	フォーカス時の文字色	RGBの文字列 (“0～255 | 0～255 | 0～255”)	“0	0	0”	可
inFocusBackColor※2	フォーカス時の背景色	RGBの文字列 (“0～255 | 0～255 | 0～255”)	“192	192	192”	可

名称	説明	範囲	初期値	代入
outFocusTextColor※2	非フォーカス時の文字色	RGBの文字列 (“0~255 | 0~255 | 0~255”)	“0	0	0”	可
outFocusBackColor※2	非フォーカス時の背景色	RGBの文字列 (“0~255 | 0~255 | 0~255”)	“255	255	255”	可

※1 Widget共通プロパティです。他のWidget操作コントロールでも使用します。ここでは、Widget共通プロパティの詳細説明を省略しています。
Widget共通プロパティの詳細説明については、「Widget共通プロパティ」(7-5ページ)を参照してください。
※2 simpleWidgetプロパティがtrueの場合のみ有効です。

イベントプロパティ

名称	発生タイミング	sender
onFocusIn※	RadioButtonにフォーカスが入ったとき	RadioButtonのタグ名
onFocusOut※	RadioButtonからフォーカスが抜けたとき	RadioButtonのタグ名
onTouchUp※	タッチパネルからリリースしたとき BT-W100/W80/W70シリーズのみ有効です。	RadioButtonのタグ名
onCheckChange	チェック状態を変更したとき	RadioButtonのタグ名

※ Widget共通イベントプロパティです。他のWidget操作コントロールでも使用します。ここでは、Widget共通イベントプロパティの詳細説明を省略しています。
Widget共通イベントプロパティの詳細説明については、「Widget共通イベントプロパティ」(7-7ページ)を参照してください。

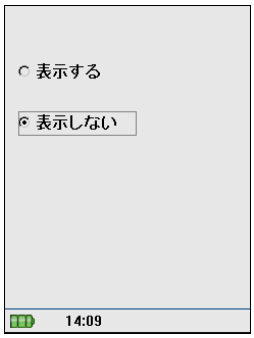
メソッド

名称	説明	引数	戻り値
Create※	RadioButtonを作成します。	targetWidget:作成するWidgetの所属先になるWindowのタグ名	true:正常終了 false:エラー終了
Copy※	RadioButtonをコピーします。	targetWidget:複製Widgetの所属先になるWindowのタグ名 copyWidget:コピー元になるRadioButtonのタグ名	true:正常終了 false:エラー終了
Delete※	RadioButtonを削除します。	なし	true:正常終了 false:エラー終了
Update※	RadioButtonの表示を最新情報に更新します。	なし	true:正常終了 false:エラー終了
SetFocus※	RadioButton にフォーカスを当てます。	なし	true:正常終了 false:エラー終了
GetFocus※	現在フォーカスが当たっているWidgetのタグ名を取得します。	なし	Widgetのタグ名

※ Widget共通メソッドです。他のWidget操作コントロールでも使用します。ここでは、Widget共通メソッドの詳細説明を省略しています。
Widget共通メソッドの詳細説明については、「Widget共通メソッド」(7-8ページ)を参照してください。

■排他選択(groupIDプロパティ)

ラジオボタンを複数作成して、同一グループとして取り扱うことにより、グループ内で排他選択ができます。



上記画面は、次のように設定します。

```
With RadioButton<"item1">          // item1のタグ名
      :groupId = 0                 // item2と同じgroupId
      :checked = "item2"           // 選択状態はitem2
EndWith

With RadioButton<"item2">          // item2のタグ名
      :groupId = 0                 // item1と同じgroupId
EndWith
```

プログラミングの方法は、「使用例」(7-72ページ)を参照してください。

使用例

ROOT_WINDOW上に、同じグループ0の中で排他選択できるラジオボタンを2つ作成します。

```
Method Job1()
    Begin

    With RadioButton<"RadioButton1">          /* RadioButtonのタグ名に
                                                 「RadioButton1」を設定 */
        :Create("ROOT_WINDOW")                /* ルートウィンドウ上に
                                                 「RadioButton1」を作成 */
        :foreColor = "0|0|0"                  // 表示文字列の色を設定
        :left = 10                            /* ルートウィンドウ上に「RadioButton1」
                                                 を表示する最左X座標=10 */
        :top = 50                             /* ルートウィンドウ上に「RadioButton1」
                                                 を表示する最上Y座標=50 */
        :width = 122                          // 幅は122ドット
        :height = 30                          // 高は30ドット
        :fontSize = 16                        // 表示文字列のフォントを設定
        :text = "表示する"                    // 表示文字列を設定
        :textAlign = "left"                   // 表示文字列の表示位置を設定
        :checked = "RadioButton2"             // チェックするWidgetのタグ名を設定
        :groupId = 0                          // グループIDを設定
        :visible = true
        :enable = true
    End With

    With RadioButton<"RadioButton2">          /* RadioButtonのタグ名に
                                                 「RadioButton2」を設定 */
        :Create("ROOT_WINDOW")
        :foreColor = "0|0|0"
        :left = 10
        :top = 100
        :width = 122                          // 幅は122ドット
        :height = 30                          // 高さは30ドット
        :fontSize = 16
        :text = "表示しない"                  // 表示文字列を設定
        :textAlign = "left"
        :checked = "RadioButton2"             // チェックするWidgetのタグ名を設定
        :groupId = 0                          // グループIDを設定
        :visible = true
        :enable = true
    End With

    Window<"ROOT_WINDOW">:Update()            // ルートウィンドウの表示更新

    Event:Wait()                              // イベント取得

EndMethod
```

Button	ボタン動作の設定をします。 タグ指定コントロールです。

プロパティ

名称	説明	範囲	初期値	代入
parent※1	親Widgetのタグ名	Widgetのタグ名	nil	不可
frame	枠種別	“none”:なし “thin”:線枠	“thin”	可
foreColor※1	前景色	RGBの文字列 (“0~255 | 0~255 | 0~255”)	“0 | 0 | 0”	可
backColor※1	背景色	RGBの文字列 (“0~255 | 0~255 | 0~255”)	“224 | 224 | 224”	可
focusColor※1	フォーカス枠色	RGBの文字列 (“0~255 | 0~255 | 0~255”)	“128 | 128 | 128”	可
visible※1	表示／非表示	true:表示 false:表示しない	false	可
enable※1	有効／無効	true:有効 false:無効	true	可
left※1	最左X座標	0~239	0	可
top※1	最上Y座標	0~1279	0	可
width※1	幅	1~240(ドット)	90	可
height※1	高さ	1~1280(ドット)	34	可
text※1	テキスト	文字列 (最大8192バイト)	“”	可
fontSize※1	フォントサイズ	12:12ドットフォント 16:16ドットフォント 24:24ドットフォント	12	可
textAlign※1	テキストの表示位置	"left":左詰め "center":中央 "right":右詰め	“center”	可
frame	枠種別	“none”:なし “thin”:線枠 “raised”:厚型	“thin”	可
fontBold	太字設定の有効/無効	true/false	true	可
fontSize	フォントサイズ	6~255	-	可
fontName	フォント名	32文字以内のフォント名	“”	可
textVAlign	テキストの表示位置(上下)	“top”(上端揃え) “bottom”(下端揃え) “center”(上下中央揃え)	“top”	可
simpleWidget	Simple Widget有効/無効	true/false	false	可
inFocusTextColor※2	フォーカス時の文字色	RGBの文字列 (“0~255 | 0~255 | 0~255”)	“0 | 0 | 0”	可
inFocusBackColor※2	フォーカス時の背景色	RGBの文字列 (“0~255 | 0~255 | 0~255”)	“192 | 192 | 192”	可
outFocusTextColor※2	非フォーカス時の文字色	RGBの文字列 (“0~255 | 0~255 | 0~255”)	“0 | 0 | 0”	可

名称	説明	範囲	初期値	代入
outFocusBackColor※2	非フォーカス時の背景色	RGBの文字列 （“0～255｜0～255｜0～255”）	“255｜255｜255”	可
pushDownTextColor※2	ボタン押下状態の文字色	RGBの文字列 （“0～255｜0～255｜0～255”）	“0｜0｜0”	可
pushDownBackColor※2	ボタン押下状態の文字色	RGBの文字列 （“0～255｜0～255｜0～255”）	“128｜128｜128”	可

※1 Widget共通プロパティです。他のWidget操作コントロールでも使用します。ここでは、Widget共通プロパティの詳細説明を省略しています。
Widget共通プロパティの詳細説明については、「Widget共通プロパティ」（7-5ページ）を参照してください。

※2 simpleWidgetプロパティがtrue、frameプロパティが"none"の場合のみ有効です。

イベントプロパティ

名称	発生タイミング	sender
onFocusIn※	Buttonにフォーカスが入ったとき	Buttonのタグ名
onFocusOut※	Buttonからフォーカスが抜けたとき	Buttonのタグ名
onTouchUp※	タッチパネルからリリースしたとき BT-W100/W80/W70シリーズのみ有効です。	Buttonのタグ名
onButton	ボタンを押したとき	Buttonのタグ名

※ Widget共通イベントプロパティです。他のWidget操作コントロールでも使用します。ここでは、Widget共通イベントプロパティの詳細説明を省略しています。
Widget共通イベントプロパティの詳細説明については、「Widget共通イベントプロパティ」（7-7ページ）を参照してください。

メソッド

名称	説明	引数	戻り値
Create※	Buttonを作成します。	targetWidget：作成するWidgetの所属先になるWindowのタグ名	true：正常終了 false：エラー終了
Copy※	Buttonをコピーします。	targetWidget：複製Widgetの所属先になるWindowのタグ名 copyWidget：コピー元になるButtonのタグ名	true：正常終了 false：エラー終了
Delete※	Buttonを削除します。	なし	true：正常終了 false：エラー終了
Update※	Buttonの表示を最新情報に更新します。	なし	true：正常終了 false：エラー終了
SetFocus※	RadioButtonにフォーカスを当てます。	なし	true：正常終了 false：エラー終了
GetFocus※	現在フォーカスが当たっているWidgetのタグ名を取得します。	なし	Widgetのタグ名

名称	説明	引数	戻り値
SetImage	ボタン上に表示する画像を設定します。	fileName:画像ファイル名(ドライブ番号:ファイル名) 📖「アイコンボタン(SetImageメソッド)」(7-75ページ)	true:正常終了 false:エラー終了

※ Widget共通メソッドです。他のWidget操作コントロールでも使用します。ここでは、Widget共通メソッドの詳細説明を省略しています。
Widget共通メソッドの詳細説明については、📖「Widget共通メソッド」(7-8ページ)を参照してください。

■アイコンボタン(SetImageメソッド)

ボタンに画像を表示することにより、アイコンボタンとして使用できます。
画像は、次の4種類が必要です。

- 通常のとき
- 押したとき
- 無効のとき
- フォーカスが入ったとき

4種類の画像を設定すると、操作の状態にあわせて表示画像が切り替わります。

●画像形式

使用できる画像ファイルの形式は、次の4種類です。

- ビットマップファイル(.BMP)
- JPEGファイル(.JPG)(Exif情報が付加されたJPEGファイルは使用できません)
- PNGファイル(.PNG)
- GIFファイル(.GIF)

●画像の指定方法

ドライブ番号:ディレクトリ名¥ファイル名(最大255バイトまで)
【例】2:AAA¥BBB.BMP
ドライブ番号を指定しない場合は、ドライブ2になります。

●「fileName」の設定

1 ボタンサイズと同じ画像ファイルを下図の順番に連結して、1つのファイルにします。

【例】

通常時　押下時　無効時　フォーカス時

2 「fileName」に、パス付きのファイル名(最大255バイトまで)を指定します。

【例】 2:aaa¥bbb.BMP

使用例

例1　ROOT_WINDOW上にボタンを2つ表示します。

```
Method JOB()
    Begin

        With Button<"Button1">              // Buttonのタグ名に「Button1」を設定
            :Create("ROOT_WINDOW")          // ルートウィンドウ上に「Button1」を作成
            :foreColor = "0|0|0"            // 表示文字列の色を設定
            :left = 10                      /* ルートウィンドウ上に「Button1」を
                                               表示する最左X座標＝10 */
            :top = 50                       /* ルートウィンドウ上に「Button1」を
                                               表示する最上Y座標＝50 */
            :width = 122                    // 幅は122ドット
            :height = 20                    // 高さは20ドット
            :fontSize = 16                  // 表示文字列のフォントを設定
            :text = "表示する"              // 表示文字列を設定
            :textAlign = "left"             // 表示文字列の表示位置を設定
            :visible = true
            :enable = true
        End With

        With Button<"Button2">              // Buttonのタグ名に「Button2」を設定
            :Create("ROOT_WINDOW")
            :foreColor = "0|0|0"
            :left = 10
            :top = 150
            :width = 122
            :height = 20
            :fontSize = 16
            :text = "表示しない"            // 表示文字列を設定
            :textAlign = "left"
            :visible = true
            :enable = true
        End With

        With Window< "ROOT_WINDOW" >
            :enable = true
            :Update()                       // ルートウィンドウの表示更新
        End With

        Event:Wait()                        // イベント取得

End Method
```

使用例

例2　ROOT_WINDOW上にアイコンボタンを作成します。

```
Method ICON_SAMPLE()
Begin

    Window<"ROOT_WINDOW">:enable = true

    With Button< "TEST" >                       // Buttonのタグ名に「TEST」を設定
        :Create( "ROOT_WINDOW" )                // ルートウィンドウ上に「TEST」を作成
        :foreColor = "200|200|200"
        :backColor = "255|0|0"
        :visible = TRUE
        :enable = TRUE
        :left = 50
        :top = 50
        :width = 75
        :height = 45
        :fontSize = 12
        :text = "テスト"
        :SetImage( "2:ICON.BMP" )               // ビットマップファイルを設定

        Window<"ROOT_WINDOW">:Update()  // ルートウィンドウの表示更新

        Handy:KeyWait()                         // 入力待ち
        :Delete()                               // 「TEST」の削除
        :Update()
    End With

End Method
```

ListTable	文字列とデータを紐づけるテーブルを作成します。

プロパティ

なし

メソッド

名称	説明	引数	戻り値
CreateList	空のリストを作成します。	なし	リストID(1～) false:エラー終了
DeleteList	指定リストを削除します。子リストも削除されます。	id:リストID	true:正常終了 false:エラー終了
Add	リストにアイテムを登録します。	id:リストID key:設定キー val:設定データ	true:正常終了 false:エラー終了
AddList	リストにリストを登録します。	id:リストID key:設定キー add_id:登録するリストID	true:正常終了 false:エラー終了
Remove	指定アイテムを削除します。	id:リストID idx:アイテムインデックス (0～255)(0～1023)	true:正常終了 false:エラー終了
Count	リストに含まれるアイテム数を取得します。	id:リストID	true:正常終了 false:エラー終了
SetValue	アイテムのキーとデータを設定します。	id:リストID key:設定キー val:設定データ	true:正常終了 false:エラー終了
GetValue	指定した設定キーを持つアイテムのデータを取得します。	id:リストID key:検索用設定キー	true:正常終了 false:エラー終了
Set	指定インデックスのアイテムのキーまたはデータを設定します。	id:リストID idx:アイテムインデックス type:設定対象("key" / "val") val:設定値	true:正常終了 false:エラー終了
Get	指定インデックスのアイテムのキーまたはデータを取得します。	id:リストID idx:アイテムインデックス type:設定対象("key" / "val")	true:正常終了 false:エラー終了
Find	リストから指定されたキーまたはデータを持つアイテムのインデックスを取得します。	id:リストID type:設定対象("key" / "val") val:検索用データ	アイテムインデックス(0～) false:エラー終了

使用例

例1

```
Method sample_ListTable()
        id_1
        id_2
        id_3
        cnt
        val
        idx
Begin
        // リスト作成
        id_1 = ListTable:CreateList()
        id_2 = ListTable:CreateList()

        // 親リストに子リスト登録
        ListTable:AddList(id_1, "野菜", id_2)

        // アイテム登録
        ListTable:Add(id_2, "大根", "100")
        ListTable:Add(id_2, "キャベツ", "200")
        ListTable:Add(id_2, "トマト", "150")

        // アイテム数取得
        cnt = ListTable:Count(id_2)

        // データ設定("200"→"180")
        ListTable:SetValue(id_2, "キャベツ", 180)

        // データ取得("180")
        val = ListTable:GetValue(id_2, "キャベツ")

        // アイテム検索
        idx = ListTable:Find(id_2, "key", "トマト")

        // データ設定("150"→"140")
        ListTable:Set(id_2, idx, "val", 140)

        // データ取得("トマト")
        val = ListTable:Get(id_2, idx, "key")

        // データ取得("140")
        val = ListTable:Get(id_2, idx, "val")

        // アイテム削除
```

```
        ListTable:Remove(id_2, idx)

        // 親リストから子リストID取得
        id_3 = ListTable:Get(id_1, 0, "val")

        // リスト削除
        ListTable:DeleteList(id_1)
        ListTable:DeleteList(id_2)
End Method
```

7-3 イベントの取得

Widget操作コントロールなどでイベントを登録した場合は、イベントを取得してイベント処理をおこないます。

Event	イベント取得の設定をします。

メソッド

名称	説明	引数	戻り値
Initialize	各コントロールのイベントプロパティを初期化します。	なし	なし
Wait	イベントを取得します。 「イベントの取得と終了(Wait、Exitメソッド)」(7-81ページ)	なし	なし
Exit	イベントの取得を終了します。 「イベントの取得と終了(Wait、Exitメソッド)」(7-81ページ)	なし	なし
Clear	イベントキューをクリアします。	なし	なし

■イベントの取得と終了(Wait、Exitメソッド)

●イベントの取得(「Wait」メソッド)

「Wait」メソッドを実行すると、イベントを待ち続け、イベント発生時にイベントを取得できます。

「Wait」メソッドを実行している状態で、「Wait」メソッドを実行しないでください。アプリケーションが正常に動作しなくなるおそれがあります。

●イベント取得の終了

「Wait」メソッドを終了するには、「Exit」メソッドを実行します。

「Exit」メソッドは、プログラム中のどのメソッド内でも実行できます。

■イベントプロパティ一覧

イベントが取得できるコントロールのイベントプロパティを一覧で示します。
イベントを取得するには、それぞれのコントロールであらかじめイベントプロパティの設定が必要です。

コントロール	プロパティ	sender※1
Key	onPress	キーテーブル
Window	onFocusIn	Widgetのタグ名
	onFocusOut	Widgetのタグ名
	onTouchUp	Widgetのタグ名
TextField	onFocusIn	Widgetのタグ名
	onFocusOut	Widgetのタグ名
	onTouchUp	Widgetのタグ名
	onEditStart	Widgetのタグ名
	onEditEnd	Widgetのタグ名
	onEditCancel	Widgetのタグ名
	onTouchOut	Widgetのタグ名
	onScanComplete	Widgetのタグ名
TextBox	onFocusIn	Widgetのタグ名
	onFocusOut	Widgetのタグ名
	onTouchUp	Widgetのタグ名
	onEditStart	Widgetのタグ名
	onEditEnd	Widgetのタグ名
	onEditCancel	Widgetのタグ名
	onTouchOut	Widgetのタグ名
	onScanComplete	Widgetのタグ名
ListBox	onFocusIn	Widgetのタグ名
	onFocusOut	Widgetのタグ名
	onTouchUp	Widgetのタグ名
	onSelectChange	Widgetのタグ名
	onEditStart	Widgetのタグ名
	onEditEnd	Widgetのタグ名
	onEditCancel	Widgetのタグ名
	onTouchOut	Widgetのタグ名
ComboBox	onFocusIn	Widgetのタグ名
	onFocusOut	Widgetのタグ名
	onTouchUp	Widgetのタグ名
	onSelectChange	Widgetのタグ名
	onEditStart	Widgetのタグ名
	onEditEnd	Widgetのタグ名
	onEditCancel	Widgetのタグ名
	onTouchOut	Widgetのタグ名
CheckBox	onFocusIn	Widgetのタグ名
	onFocusOut	Widgetのタグ名
	onTouchUp	Widgetのタグ名
	onCheckChange	Widgetのタグ名
RadioButton	onFocusIn	Widgetのタグ名
	onFocusOut	Widgetのタグ名
	onTouchUp	Widgetのタグ名
	onCheckChange	Widgetのタグ名

コントロール	プロパティ	sender※1
Button	onFocusIn	Widgetのタグ名
	onFocusOut	Widgetのタグ名
	onTouchUp	Widgetのタグ名
	onButton	Widgetのタグ名
BCR	onScanComplete	nil
	onScanTimeout	nil
	onScanError	nil
Network	onConnect	Openメソッドで指定したtype ※2
	onRefused	Openメソッドで指定したtype ※2
	onOutrange	Openメソッドで指定したtype ※2
	onDHCPIpChanged	Openメソッドで指定したtype ※2
WLAN	onConnect	"WLAN"固定
	onRefused	"WLAN"固定
	onOutrange	"WLAN"固定
Bluetooth	onConnect	"Bluetooth"固定
	onRefused	"Bluetooth"固定
	onOutrange	"Bluetooth"固定
Handy	onResume	1(固定)
	onCradle	1:ON
		0:OFF
	onBatteryLevelChange	1～4:電池残量
Timer	onTimer	1～8:タイマーID

※1 senderは、イベントが発生したとき、イベント処理メソッドに渡す引数の内容になります。
「使用例」(7-84ページ)

※2 Network:Openの引数(type)により、値は異なります。

- infrastructure "infra"
- adhoc "adhoc"
- Bluetooth SPP "SPP"
- Bluetooth PPP "PPP"
- Modem "MODEM"
- Modem PPP "MODEM_PPP"

使用例

Widgetイベントの登録と取得

```
Method main()
Begin
    With Button<"ボタン1">                    // Buttonのタグ名に「ボタン1」を指定
        :Create("ROOT_WINDOW")                // ルートウィンドウ上に「ボタン1」を作成
        :foreColor = "0|0|0"
        :left = 10
        :top = 50
        :width = 122
        :height = 20
        :fontSize = 16
        :text = "表示する"
        :textAlign = "left"
        :visible = true
        :enable = true

        :onButton=OnButton1                   // イベント登録
        // イベント発生時のイベント処理を指定
        // イベント処理メソッドの引数は記述しません。
    EndWith

    With Window< "ROOT_WINDOW" >
        :enable = true
        :Update()                             // ルートウィンドウの表示更新
    End With

    Event:Wait()                              // イベント発生時にイベントを取得します。
EndMethod

// イベント処理のメソッド
Method OnButton1(sender)                      /* イベント理のメソッドは、senderを引数に
                                                 します。*/
Begin
    Handy:ShowMessageBox(sender & " が押されました","ok")
    //「 ボタン 1 が押されました」というメッセージが表示されます。
EndMethod
```

7-4 データの入力

データ入力のコントロールには次の4種類があります。
- InputString：キーとバーコード読み取りの両方から入力します。
- InputDecimal：キーから数値入力します。
- InputDate：日付データを入力します。
- InputByList：選択マスタからデーター覧を表示して選択入力します。

InputString	キーやバーコード読み取りから数値と文字データを入力します。 BT-W100/W80/W70のエミュレータ表示、BT-600/1000/1500のCUIプログラムで使用できます。

プロパティ

名称	説明	範囲	初期値	代入
data※1	初期値と入力結果 初期値を設定します。また、入力された文字列が格納されます。	文字列 (最大8192バイト)※2	""	可
posx※1	X座標 入力開始位置のX座標を指定します。	テキスト座標(半角)の場合※3 (例)BT-W80/W70/1000/1500シリーズ 16dot：1～30 24dot：1～20 32dot：1～15	1	可
		グラフィック座標の場合 (例)BT-W80/W70/1000/1500シリーズ 16dot：0～232 24dot：0～228 32dot：0～224	0	
posy※1	Y座標 入力開始位置のY座標を指定します。	テキスト座標(半角)の場合※3 (例)BT-W80/W70/1000/1500シリーズ 16dot：1～37 24dot：1～24 32dot：1～18	1	可
		グラフィック座標の場合 (例)BT-W80/W70/1000/1500シリーズ 16dot：0～303 24dot：0～295 32dot：0～287	0	
multiColumn※1	行折り返し設定 「6-2 画面表示(BT-600/1000/1500シリーズ)」(6-12ページ)	1：折り返ししない 2：折り返し対応インデントなし 3：折り返し対応インデントあり	1	可
inputAreaSize※1	入力領域幅 カーソルを含めた入力欄の文字数を設定します。	2～40文字： N文字の文字列を表示するときはN+1を設定します。	20	可
maxLength	入力文字数を制限するときの最大文字列長を指定します。	1～8192バイト(半角)※4	8192	可
minLength	入力文字数を制限するときの最小文字列長を指定します。	0～8192バイト(半角)※4	0	可
enableKeyInput	キー入力を有効にする、しないを設定します。	true：有効にする false：無効にする	true	可
scanMode	トリガモードの設定 バーコードを読み取りしない場合には、「nil」に設定します。	nil：バーコード入力しない 1：オートオフ 2：連続読み 3：モーメンタリ 4：ソフトウェアトリガ 5：離して読み 6：ダブルクリック	1	可

名称	説明	範囲	初期値	代入
font※1	フォント属性 「6-2 画面表示(BT-600/1000/1500シリーズ)」(6-12ページ)	1:normal 2:横倍角 3:縦倍角 4:4倍角 9:9倍角 16:16倍角 25:25倍角	1	可
reverse※1	入力文字列の反転表示	true:反転表示する false:反転表示しない	false	可
shift	シフトキーの設定 半角英字入力時に使用します。 「シフトキーを使用した半角英字入力(shiftプロパティ)」(7-87ページ)	nil:無効 0:offで開始 1:onで開始	nil	可
shiftPattern※7	シフト入力時のキーパターン	"BT500" "BT1000"	"BT1000"	可
useEdit	日本語入力をする、しないを設定します。 システムメニューの言語設定を「日本語」にしている場合のみ使用可能です。	nil:しない 1:使用する 2:Editのみ※5	nil	可
editTitle	日本語入力画面のタイトル設定 useEdit=2のときのみ有効です。	(例)BT-W80/W70/1000/1500シリーズ 文字列(24dotフォントの場合、最大20バイト)	nil	可
initErase	初期値入力状態からカーソル移動したときに、初期値をクリアする、しないを設定します。	true:クリアする false:クリアしない	true	可
escapeInput	データ入力中、F1/F2/L/Rキーで入力を終了する、しないを設定します。 true(入力終了する)に設定した場合、"F1","F2","L","R"※6の戻り値を返し、入力途中のデータをキャンセルします。※7	true:入力終了する false:入力終了しない	false	可
passwordMode※8	有効にすると入力済み文字を「*」文字で隠すことができます。	true:「*」文字で隠す false:「*」文字で隠さない	false	可
coordinate※1	posx , posyの座標系を変更します。	"text": テキスト座標 "graphic": グラフィック座標	"text"	可
initBufClear※8	開始時点のキーバッファをクリアします。	true:クリアする false:クリアしない	true	可

※1 データ入力コントロールの共通プロパティです。他のコントロールでも使用します。
※2 maxLength以上の文字列は切り捨てられます。
※3 文字間ギャップ「0」、行間ギャップ「1」のときの値です。文字間ギャップや行間ギャップを大きくすると、指定できる座標は小さくなります。
フォントサイズ12×12、14×14、16×16、12×20ドットの切り替え、文字間ギャップ、行間ギャップは、あらかじめScreenコントロールで設定しておきます。
※4 漢字やひらがなは1文字2バイトと数えます。
画面をはみ出す値を入力すると、実行時例外になります。
※5 useEdit=2 にした後、InputString:Exec を実行すると、日本語入力の画面が起動します。システムメニューの言語設定を「日本語」以外にしていると、実行時例外が発生しますのでご注意ください。
※6 BT-W100/W80/W70 シリーズで "L"/"R" キーを使用する場合は、システムメニュー「131.キー」でキーの割り付けをしてください。Lキーは0x8e、Rキーは0x8fを割り付けます。
※7 shift=nil、useEdit=nilのとき、F3キーについても入力途中のデータがキャンセルされ、"F3"の戻り値が返ります。
※8 BT-600シリーズでのみ使用可能です。

メソッド

名称	説明	引数	戻り値
Initialize	各プロパティを初期化します。	なし	なし
Exec	処理を開始します。	なし	"ENT":入力確定した場合 " TRG":バーコードが読み取れた場合 "C"(長押し):キャンセルの場合 "F1"/"F2"/"F3"/"L"/"R":各キーを押した場合※1 "ERR_BCR_INPUT":読み取りエラー "ERR_RANGE_LOWER":※2

※1 "ENT"/"TRG"/"C"/"F1"～"R"以外のキー戻り値は取得できません。
BT-W100/W80/W70シリーズは、"L"/"R"キーは、それぞれ"Tab"/"文字"キーに対応しています。
escapeInput=Falseのとき、データ入力中は"ENT"以外のキー戻り値は取得できません。
ただし、Handy:inputFixKeyでデータ入力中に入力確定するキーを指定することも可能です。

※2 「minLength」プロパティで設定した文字列長より短いデータを入力すると、「ERR_RANGE_LOWER」が返ります。また、「maxLength」プロパティで設定した文字列長より長いデータは入力できません。

■シフトキーを使用した半角英字入力(shiftプロパティ)

「shift」プロパティを使用して、シフトキー(F3)を押すと、半角英字が入力できるように設定できます。「off で開始」を設定すると、実行時にシフトキー(F3)を押すとシフト状態になり、「on で開始」を設定すると、実行と同時にシフト状態になります。

- 入力状態のときにシフトキー(F3)を押すと、ステータスエリアにシフト状態を示すアイコンが表示されます。この状態で例えばBT-600シリーズで「1」を繰り返し押すと、右のように表示が変化します。
  "A"→"B"→"C"→"a"→"b"→"c"→"1"…
- 「1」以外のキー、または「◀」、「▶」キーを押すと入力が確定して、次の位置にカーソルが移動します。
- 入力モードでシフト状態のときの各キーの変化は次の表のようになります。

●BT-W100/W80/W70/1000/1500シリーズ

キー	入力できる文字	キー	入力できる文字
①	(半角スペース)→1→(半角スペース)･･･	⑦	P→Q→R→S→p→q→r→s→7→P･･･
②	A→B→C→a→b→c→2→A･･･	⑧	T→U→V→t→u→v→8→T･･･
③	D→E→F→d→e→f→3→D･･･	⑨	W→X→Y→Z→w→x→y→z→9→W･･･
④	G→H→I→g→h→i→4→G･･･	⓪	-→+→/→0→-･･･
⑤	J→K→L→j→k→l→5→J･･･	●	*→%→$→.→*･･･
⑥	M→N→O→m→n→o→6→M･･･	⊖※	(半角スペース)→-→(半角スペース)

●BT-600シリーズ

「shiftPattern」プロパティの設定値により、入力できる文字が変わります。

“BT-1000”の場合

キー	入力できる文字	キー	入力できる文字
①	A→B→C→a→b→c→1→A･･･	⑦	S→T→U→s→t→u→7→S･･･
②	D→E→F→d→e→f→2→D･･･	⑧	V→W→X→v→w→x→8→V･･･
③	G→H→I→g→h→i→3→G･･･	⑨	Y→Z→y→z→(半角スペース)→9→Y･･･
④	J→K→L→j→k→l→4→J･･･	⓪	- →+→/→0→-･･･
⑤	M→N→O→m→n→o→5→M･･･	・	＊→%→$→.→＊
⑥	P→Q→R→p→q→r→6→P･･･	−※	(半角スペース)→ - →(半角スペース)

“BT-500”の場合

キー	入力できる文字	キー	入力できる文字
①	1→A→B→C→1･･･	⑦	7→S→T→U→7･･･
②	2→D→E→F→2･･･	⑧	8→V→W→X→8･･･
③	3→G→H→I→3･･･	⑨	9→Y→Z→(半角スペース)→9･･･
④	4→J→K→L→4･･･	⓪	0→- →+→/→0･･･
⑤	5→M→N→O→5･･･	・	.→* →%→$→.･･･
⑥	6→P→Q→R→6･･･	−※	(半角スペース)→ - →(半角スペース)

※ −キーの入力動作は、「Handy」または「Key」コントロールで設定できます。

「Handy」(7-320ページ)
「Key」(7-315ページ)

■シフトキーを使用した半角英字入力(passwordModeプロパティ)※BT-600シリーズのみ

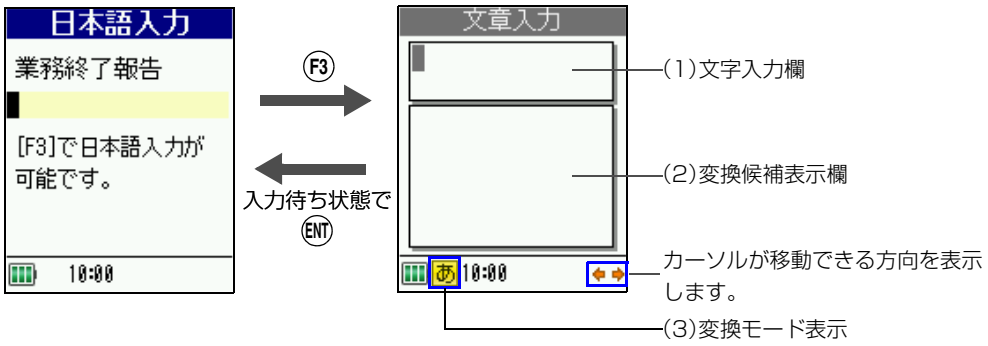
有効にすると入力済み文字を「*」文字で隠すことができます。

■日本語を入力する(useEditプロパティ)

「useEdit」プロパティを使用して、シフトキー(F3)を押すと、日本語が入力できるように設定できます。日本語(全角ひらがな/カタカナ)、半角および全角英数字(A-Z/a-z/0-9)、記号を入力できます。

1 (F3)キーを押します。

文章入力画面に切り替わります。入力待ち状態で(ENT)キーを押すと元の画面に戻ります。



(1)文字入力欄

入力した文字を表示します。

(2)変換候補表示欄

入力した文字を変換するときに、変換候補を表示します。また、記号入力するときには一覧を表示します。

(3)変換モード表示

変換モードを表示します。(F3)キーを押すたびに、変換モードを切り替えます。

表示	変換モード
あ	日本語(ひらがな)入力モード
AB	半角英数字入力モード
A	全角英数字入力モード

2 キーを押すと、文字入力欄に文字が表示されます。

日本語入力モードで入力できる文字と主な操作は以下のようになります。

キー	機能
◀/▶	カーソルの移動
(C)	カーソルの右側の1文字を削除します。

キー	入力できる文字
(1)	あ→い→う→え→お→ぁ→ぃ→ぅ→ぇ→ぉ→あ･･･
(2)	か→き→く→け→こ→か･･･
(3)	さ→し→す→せ→そ→さ･･･
(4)	た→ち→つ→て→と→っ→た･･･
(5)	な→に→ぬ→ね→の→な･･･
(6)	は→ひ→ふ→へ→ほ→は･･･

キー	入力できる文字
(7)	ま→み→む→め→も→ま･･･
(8)	や→ゆ→よ→ゃ→ゅ→ょ→や･･･
(9)	ら→り→る→れ→ろ→ら･･･
(0)	わ→を→ん→わ･･･
(・)※	゛→゜→゛･･･
(−)	ーー→、→。→！→？→･→ー･･･

※濁音、半濁音のある文字以外では入力できません。

3 **▲または▼キーを押します。**

変換候補欄に変換候補が表示されます。

▲/▼キーを押して候補を選択します。

参考　○キー(◄ ►)を押すと、文節の長さを変更できます。

4 **(ENT)キーを押します。**

変換された文字が文字入力欄に表示されます。

複数の文節の場合、次の文節の変換候補が表示されます。

5 **複数の文節がある場合には、手順3、4を繰り返します。**

6 **変換が完了したら、(ENT)キーを押します。**

元の画面に戻ります。

文字入力欄の文字列が画面に表示されます。

● 入力可能な文字

日本語入力モード

キー	入力できる文字
①	あ→い→う→え→お→ぁ→ぃ→ぅ→ぇ→ぉ→あ･･･
②	か→き→く→け→こ→か･･･
③	さ→し→す→せ→そ→さ･･･
④	た→ち→つ→て→と→っ→た･･･
⑤	な→に→ぬ→ね→の→な･･･
⑥	は→ひ→ふ→へ→ほ→は･･･

キー	入力できる文字
⑦	ま→み→む→め→も→ま･･･
⑧	や→ゆ→よ→ゃ→ゅ→ょ→や･･･
⑨	ら→り→る→れ→ろ→ら･･･
⓪	わ→を→ん→わ･･･
・※	゛→゜→゛･･･
−	ー→、→。→！→？→・→―･･･

※濁音、半濁音のある文字以外では入力できません。

半角/全角英数字入力モード

全角英数字入力モードの場合、全角文字になります。

●BT-W100/W80/W70/1000/1500シリーズ

キー	入力できる文字
①	(半角スペース)→1→(半角スペース)･･･
②	A→B→C→a→b→c→2→A･･･
③	D→E→F→d→e→f→3→D･･･
④	G→H→I→g→h→i→4→G･･･
⑤	J→K→L→j→k→l→5→J･･･
⑥	M→N→O→m→n→o→6→M･･･

キー	入力できる文字
⑦	P→Q→R→S→p→q→r→s→7→P･･･
⑧	T→U→V→t→u→v→8→T･･･
⑨	W→X→Y→Z→w→x→y→z→9→W･･･
⓪	-→+→/→0→-･･･
・	*→%→$→.→*･･･
−※	(半角入力時)-→(半角スペース)→-･･･(全角入力時)−

●BT-600シリーズ

キー	入力できる文字
①	A→B→C→a→b→c→1→A･･･
②	D→E→F→d→e→f→2→D･･･
③	G→H→I→g→h→i→3→G･･･
④	J→K→L→j→k→l→4→J･･･
⑤	M→N→O→m→n→o→5→M･･･
⑥	P→Q→R→p→q→r→6→P･･･

キー	入力できる文字
⑦	S→T→U→s→t→u→7→S･･･
⑧	V→W→X→v→w→x→8→V･･･
⑨	Y→Z→y→z→(半角スペース)→9→Y･･･
⓪	– →+→/→0→–･･･
•	＊→%→$→.→＊
−※	(半角スペース)→ – →(半角スペース)

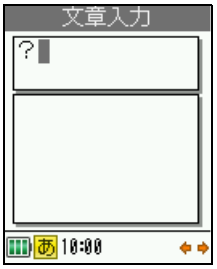
●操作一覧

キー	状態		
	入力待ち	入力中	変換候補選択中
ENT	元の画面に戻ります。文字入力欄の内容が元の画面に入力されます。	入力中の文字を無変換で確定します。	変換を確定して、入力待ち状態に戻ります。文節が残っている場合には、次の候補を表示します。
C	カーソルの右側の1文字を削除します。	カーソルの右側の1文字を削除します。	変換を中止して、入力中の状態に戻ります。
L	(機能しません)	カーソルを左に移動します。	変換中の文節を1文字減らします。
R	(機能しません)	カーソルを右に移動します。	変換中の文節を1文字増やします。
▲	カーソル移動	反転している文節の変換候補を表示します。	候補を選択します。(上方向に移動します)
▼			候補を選択します。(下方向に移動します)
◄		カーソルを左に移動します。	変換中の文節を1文字減らします。
►		カーソルを右に移動します。	変換中の文節を1文字増やします。
F1	記号一覧表示	(機能しません)	(機能しません)
F2	(機能しません)	全角かな/全角カナ/半角カナ切り替え	全角かな/全角カナ/半角カナ切り替え
F3	変換モード切り替え		

●記号の入力

1 **入力待ち状態でF1キーを押します。**
変換候補表示欄に記号一覧が表示されます。

2 ○キーで選択して、ENTキーを押します。
選択した記号が入力されます。



●注意事項

- 区点コードや外字の入力はできません。

使用例

例1　バーコード入力の処理をおこないます。

```
Method main()
Begin
    // バーコード入力の文字列を表示
    With Screen
        :fontSize = "middle"
        :multiColumn = 3
        :OutputText("バーコードを読み取ってください")
        :posy = 8
        :OutputText("(C長押し:キャンセル)")
    End With

    // バーコード入力
    With InputString
        :Initialize()
        :posy = 10

        while true
            // Cキーが入力されたら、入力から抜ける
            If :Exec() eq "C" Then Wbreak End If
        Wend
    End With
End Method
```

例2　入力されたキー(F1～3､ENTキー)によって該当する処理をおこないます。

```
Method Job2()
    Begin
        Screen:fontSize="middle"
        //入力されたキー(F1～3､ENTキー)によって該当する処理をおこないます。
        With InputString
            :data=""                    //｢data｣プロパティの初期化
            :multiColumn=2              // 折り返しインデントなし

            Select Case :Exec()
            Case "F1"
            //OnF1:Exec()               //｢F1｣キーが押されたときの処理
            Case "F2"
            //OnF2:Exec()               //｢F2｣キーが押されたときの処理
            Case "F3"
            return("F3Called")          //｢F3｣キーが押されたときの処理
            Case "ENT"
            Return(InputString:data) //入力データを戻り値として抜ける
            End Select
        End With

        Return(nil)
EndMethod
```

InputDecimal	キーから整数、小数を入力します。 BT-W100/W80/W70のエミュレータ表示、BT-600/1000/1500のCUIプログラムで使用できます。

プロパティ

名称	説明	範囲	初期値	代入
data※2	初期値と入力結果	数値	0	可
posx※2	X座標 入力開始位置のX座標を指定します。	テキスト座標(半角)の場合 (例)BT-W80/W70/1000/1500シリーズ 16dot:1～30 24dot:1～20 32dot:1～15	1	可
		グラフィック座標の場合 (例)BT-W80/W70/1000/1500シリーズ 16dot:0～232 24dot:0～228 32dot:0～224	0	
posy※2	Y座標 入力開始位置のY座標を指定します。	テキスト座標(半角)の場合 (例)BT-W80/W70/1000/1500シリーズ 16dot:1～37 24dot:1～24 32dot:1～18	1	可
		グラフィック座標の場合 (例)BT-W80/W70/1000/1500シリーズ 16dot:0～303 24dot:0～295 32dot:0～287	0	
multiColumn※2	行折り返し設定	1:折り返ししない 2:折り返し対応インデントなし 3:折り返し対応インデントあり	1	可
inputAreaSize※2	入力領域幅	3～40文字(半角): N桁の数字を入力するときは、N+2を設定します。	20	可
maxValue	最大値 入力数値を制限するときの最大値を指定します。	整数:−2147483647～2147483647 小数:−2147483.647～2147483.647 nil:最大値を指定しない	nil	可
minValue	最小値 入力数値を制限するときの最小値を指定します。	整数:−2147483647～2147483647 小数:−2147483.647～2147483.647 nil:最小値を指定しない	nil	可
font※2	フォント属性 「6-2 画面表示(BT-600/1000/1500シリーズ)」(6-12ページ)	1:normal 2:横倍角 3:縦倍角 4:4倍角 9:9倍角 16:16倍角 25:25倍角	1	可
reverse※2	入力文字列の反転表示	true:反転表示する false:反転表示しない	false	可
useCalculator	電卓機能を使用する、しないを設定します。 「電卓機能(useCalculatorプロパティ)」(7-97ページ)	nil:使用しない 1:使用可能 2:電卓入力のみ	nil	可

名称	説明	範囲	初期値	代入
decimal	小数点以下の桁数を設定します。 小数点以下を含む場合、小数点以下の桁数に関係なく、使用できる範囲は、−2147483.647~2147483.647になります。	0~3	0	可
initErase	初期値入力状態からカーソル移動したときに、初期値をクリアする、しないを設定します。	true:クリアする false:クリアしない	true	可
escapeInput	データ入力中、F1/F2/L/Rキーで入力を終了する、しないを設定します。 true(入力終了する)に設定した場合、"F1", "F2", "L", "R"※1の戻り値を返し、入力途中のデータをキャンセルします。	true:入力終了する false:入力終了しない	false	可
coordinate※2	posx , posyの座標系を変更します。	"text": テキスト座標 "graphic": グラフィック座標	"text"	可

※1 BT-W100/W80/W70 シリーズで "L"/"R" キーを使用する場合は、システムメニュー「131.キー」でキーの割り付けをしてください。Lキーは0x8e、Rキーは0x8fを割り付けます。

※2 データ入力コントロールの共通プロパティです。他のコントロールでも使用します。ここでは、共通プロパティの詳しい説明を省略しています。
詳細については、「InputString」(7-85ページ)を参照してください。

メソッド

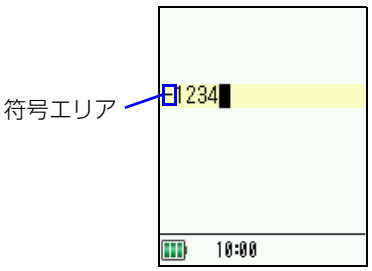
名称	説明	引数	戻り値
Initialize	各プロパティを初期化します。	なし	なし
Exec	処理を開始します。	なし	"ENT":入力確定した場合 "C"(長押し):キャンセルの場合 "F1"/"F2"/"F3"/"L"/"R":各キーを押した場合※1 "ERR_RANGE_UPPER"※2 "ERR_RANGE_LOWER"※2

※1 "ENT"/"C"(長押し)/"F1"/"F2"/"F3"/"L"/"R"以外のキー戻り値は取得できません。
BT-W100/W80/W70シリーズで"L"/"R"キーを使用する場合は、システムメニュー「131.キー」でキーの割り付けをしてください。Lキーは0x8e、Rキーは0x8fを割り付けます。
escapeInput=Falseのとき、データ入力中は"ENT"以外のキー戻り値は取得できません。
ただし、Handy:inputFixKeyでデータ入力中に入力確定するキーを指定することも可能です。

※2 「maxValue」プロパティで設定した最大値より大きいデータを入力すると、「ERR_RANGE_UPPER」が返ります。また、「minValue」プロパティで設定した最小値より小さいデータを入力すると、「ERR_RANGE_LOWER」が返ります。

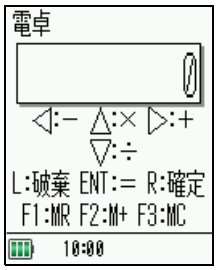
実行動作

- 入力欄の1文字目は常に符号エリアとして使用されています。
- 負の数値を入力したいときは、カーソル位置にかかわらず、「-」キーを押すと符号エリアに「-」が表示されます。
再度「-」キーを押すと符号が消えます。



■電卓機能(useCalculatorプロパティ)

「useCalculator」プロパティで「1」または「2」を設定すると、電卓機能が有効になります。



各キーの動作は次のようになります。

キー	記号	内容
①～⑨	数字	数字を入力します。
⊙	.	小数点を入力します。
◀	−	加減乗除の演算をします。
▲	×	
▶	+	
▼	÷	
ENT	=	
L	破棄	計算結果を破棄して、元の画面に戻ります。
R	確定	計算結果を戻り値として、元の画面に反映します。
F1	MR	メモリに記憶した数値を呼び出します。
F2	M+	表示している数値をメモリ内の数値に加算します。
F3	MC	メモリに記憶した数値を消去します。

InputDate	日付データを入力します。日付はカーソルキーで増加減します。 BT-W100/W80/W70のエミュレータ表示、BT-600/1000/1500のCUIプログラムで使用できます。

プロパティ

名称	説明	範囲	初期値	代入
data※	初期値と入力結果	YYYY/MM/DD (年/月/日) nil:システム日付	BTシリーズの設定値	可
posx※	X座標 入力開始位置のX座標を指定します。	テキスト座標(半角)の場合 (例)BT-W80/W70/1000/1500シリーズ 16dot:1～30 24dot:1～20 32dot:1～15	1	可
		グラフィック座標の場合 (例)BT-W80/W70/1000/1500シリーズ 16dot:0～232 24dot:0～228 32dot:0～224	0	
posy※	Y座標 入力開始位置のY座標を指定します。	テキスト座標(半角)の場合 (例)BT-W80/W70/1000/1500シリーズ 16dot:1～37 24dot:1～24 32dot:1～18	1	可
		グラフィック座標の場合 (例)BT-W80/W70/1000/1500シリーズ 16dot:0～303 24dot:0～295 32dot:0～287	0	
font※	フォント属性 「6-2 画面表示(BT-600/1000/1500シリーズ)」(6-12ページ)	1:normal 2:横倍角 3:縦倍角 4:4倍角 9:9倍角 16:16倍角 25:25倍角	1	可
step	刻み値 「実行動作」(7-99ページ)	1～100(日)	1	可
coordinate※	posx , posyの座標系を変更します。	"text": テキスト座標 "graphic": グラフィック座標	"text"	可

※ データ入力コントロールの共通プロパティです。他のコントロールでも使用します。ここでは、共通プロパティの詳しい説明を省略しています。
詳細については、「InputString」(7-85ページ)を参照してください。

メソッド

名称	説明	引数	戻り値
Initialize	各プロパティを初期化します。	なし	なし
Exec	処理を開始します。	なし	"ENT":入力確定した場合 "C"(長押し):キャンセルの場合 "F1"/"F2"/"F3"/"L"/"R":各キーを押した場合※

※ "ENT"/"C"(長押し)/"F1"/"F2"/"F3"/"L"/"R"以外のキー戻り値は取得できません。
BT-W100/W80/W70シリーズで"L"/"R"キーを使用する場合は、システムメニュー「131.キー」でキーの割り付けをしてください。Lキーは0x8e、Rキーは0x8fを割り付けます。
データ入力中は、"ENT"以外のキー戻り値は取得できません。

実行動作

- 初期化すると、日付はシステム時間が表示されます。
  「使用例」(7-99ページ)

- 日付を変更するときは、十字キーの「▲」、「▼」キーを押すごとに「step」プロパティで指定したステップ値(日数)単位で増減します。また十字キーの「◀」、「▶」キーを押すごとに1月単位で増減します。

【日付変更例】

初期値 "2006/05/09"、刻み値"10"のとき
初期値表示時2006/05/09

「▲」を1回押す	2006/04/29	10日前の日付になります。
「▼」を1回押す	2006/05/19	10日後の日付になります。

【月変更例】

初期値 "2006/05/31"のとき

「▶」を1回押す	2006/06/30	6月は30日までしかないため日付が変更されます。
さらに「▶」を押す	2006/07/30	30日のままになります。

ポイント

月単位移動の場合は、月によって存在する日、しない日があります。

使用例

下図のように表示されます。

```
Screen:font=1
Screen:OutputText("日付入力")
InputDate:data=nil  //システム時間を代入
InputDate:posy=3  //テキスト座標でY軸を3に設定
InputDate:Exec()  //日付入力コントロールを実行
```

InputByList	選択マスタのデータを液晶表示部(LCD)に表示して、選択入力します。 BT-W100/W80/W70のエミュレータ表示、BT-600/1000/1500のCUIプログラムで使用できます。

プロパティ

名称	説明	範囲	初期値	代入
data※	初期値と入力結果 📖「初期値について」(7-102ページ)	文字列 (最大256バイト)	""	可
posx※	X座標 入力開始位置のX座標を指定します。	テキスト座標(半角)の場合 (例)BT-W80/W70/ 1000/1500シリーズ 16dot:1～30 24dot:1～20 32dot:1～15	1	可
		グラフィック座標の場合 (例)BT-W80/W70/ 1000/1500シリーズ 16dot:0～232 24dot:0～228 32dot:0～224	0	
posy※	Y座標 入力開始位置のY座標を指定します。	テキスト座標(半角)の場合 (例)BT-W80/W70/ 1000/1500シリーズ 16dot:1～37 24dot:1～24 32dot:1～18	1	可
		グラフィック座標の場合 (例)BT-W80/W70/ 1000/1500シリーズ 16dot:0～303 24dot:0～295 32dot:0～287	0	
font※	フォント属性 📖「6-2 画面表示(BT-600/1000/1500シリーズ)」(6-12ページ)	1:normal　2:横倍角 3:縦倍角　4:4倍角 9:9倍角　16:16倍角 25:25倍角	1	可
reverse※	入力文字列の反転表示	true:反転表示する false:反転表示しない	false	可
row	初期値と入力結果の行番号が格納されます。 📖「初期値について」(7-102ページ)	行番号(1～2046)	1	可
fileName	選択マスタファイル名を指定します。	ドライブ番号:ファイル名	""	可
separator	ファイル中の区切り文字を設定します。	文字列(1バイト)	","	可
coordinate※	posx , posyの座標系を変更します。	"text": テキスト座標 "graphic": グラフィック座標	"text"	可

※ データ入力コントロールの共通プロパティです。他のコントロールでも使用します。ここでは、共通プロパティの詳しい説明を省略しています。
詳細については、📖「InputString」(7-85ページ)を参照してください。

メソッド

名称	説明	引数	戻り値
Initialize	各プロパティを初期化します。	なし	なし
Exec	処理を開始します。	なし	"ENT":入力確定した場合 "C":キャンセルの場合 "F1"/"F2"/"F3"/"L"/"R":各キーを押した場合※1 "ERR_FILE_NOT_FOUND"※2 "ERR_FILE_FORMAT"※3

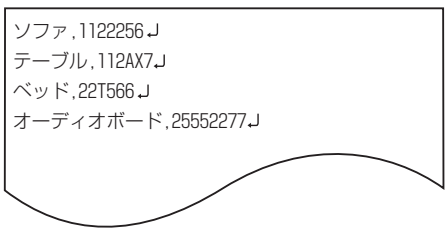
※1 "ENT"/"C"/"F1"/"F2"/"F3"/"L"/"R"以外のキー戻り値は取得できません。
BT-W100/W80/W70シリーズで"L"/"R"キーを使用する場合は、システムメニュー「131.キー」でキーの割り付けをしてください。Lキーは0x8e、Rキーは0x8fを割り付けます。
データ入力中は、"ENT"以外のキー戻り値は取得できません。

※2 該当する選択マスタファイルが見つからないときに返します。

※3 該当する選択マスタファイルのフォーマットが不正なときに返します。

実行動作

次のような選択マスタ(セパレータは",")、改行コードは<CR><LF>)を使用したときの画面と選択方法は以下のようになります。

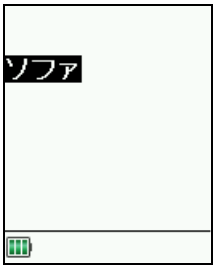


- 「ソファ」、「テーブル」、「ベッド」、「オーディオボード」などが画面に表示され選択できます。「▲」、「▼」キーの操作で選択データが変わります。
- 終端まで表示されると、先頭データが表示されます。「ENT」キーで選択が確定します。
選択されたデータは、「row」プロパティと、「data」プロパティに格納されます。たとえば「ソファ」を選択したとき、「row」プロパティには「1」が、「data」プロパティには「1122256」という文字列が格納されます。

使用例

下図のように表示されます。

```
InputByList:posx = 1
InputByList:posy = 5
InputByList:reverse = true
InputByList:fileName="1:select.txt"
InputByList:Exec()
```



■初期値について

「row」プロパティと「data」プロパティに初期値を入力した場合、下記の動作になります。

- **「data」プロパティが「""」以外の場合**
  「data」プロパティに格納されたデータを表示します。
- **「data」プロパティが「""」の場合**
  「row」プロパティで指定されたレコードを表示します。
  「row」プロパティの値が範囲外の場合には、先頭レコードを表示します。

7-5 液晶表示部(LCD)の設定

液晶表示部(LCD)とVRAMに関する設定をおこないます。

Screen	液晶表示部(LCD)に、文字、線、四角形、円弧、画像を表示します。 BT-W100/W80/W70のエミュレータ表示、BT-600/1000/1500のCUIプログラムで使用できます。

プロパティ

名称	説明	範囲	初期値	代入
topPos	液晶表示部(LCD)に表示する先頭のVRAMのY座標を指定します。 「表示データの指定(topPosプロパティ)」(7-107ページ)	テキスト座標(半角)の場合 (例)BT-W80/W70/1000/1500シリーズ 16dot:1～121※1 24dot:1～81 32dot:1～61	1	可
		グラフィック座標の場合 (例)BT-W80/W70/1000/1500シリーズ 0～960	0	
posx	X座標 VRAMに書き込むX座標を設定します。	テキスト座標(半角)の場合 (例)BT-W80/W70/1000/1500シリーズ 16dot:1～30 24dot:1～20 32dot:1～15	1	可
		グラフィック座標の場合 (例)BT-W80/W70/1000/1500シリーズ 0～239	0	
posy	Y座標 VRAMに書き込むY座標を設定します。	テキスト座標(半角)の場合 (例)BT-W80/W70/1000/1500シリーズ 16dot:1～160 24dot:1～107 32dot:1～80	1	可
		グラフィック座標の場合 (例)BT-W80/W70/1000/1500シリーズ 0～1279	0	
font	フォント属性 「6-2 画面表示(BT-600/1000/1500シリーズ)」(6-12ページ)	1:normal 2:横倍角 3:縦倍角 4:4倍角 9:9倍角 16:16倍角 25:25倍角	1	可
reverse	文字列の反転表示	true:反転表示する false:反転表示しない	false	可
foreColor	線色・文字色※6	RGBの文字列 (“0～255|0～255|0～255”)	"0|0|0"	可
backColor	背景色※6		"255|255|255"	可
selectForeColor	選択時の文字色※6		"255|255|255"	可
selectBackColor	選択時の背景色※6		"0|0|0"	可
compatibleMode※10	グラフィック座標での記述に関して、X,Y座標を2倍に扱います。※8	true:互換モード false:通常モード	false	可
compatibleType※5	互換表示する機種を設定します BT-W100/W80/W70シリーズのみ有効です。	"BT-910" ※9 "BT-1000"	"BT-910"	可

名称	説明	範囲	初期値	代入
multiColumn	行折り返し設定	1:1行のみ 2:複数行(インデントなし) 3:複数行(インデントあり)	1	可
fontSize	フォントサイズ※7	●BT-W100/W80/W70/1000/1500シリーズ "small":16×16ドットフォント "middle":24×24ドットフォント "large":32×32ドットフォント 12:"middle"として扱う(互換用) 16:"large"として扱う(互換用)	"middle"	可
		●BT-600シリーズ "small":12×12ドットフォント "middle":14×14ドットフォント "large":16×16ドットフォント 12:12×20ドットフォント(互換用) 16:"large"として扱います。	12	可
fontGapx	文字間ギャップ 文字列を表示するときの文字間隔を設定します。 📖「文字間ギャップ、行間ギャップ(fontGapx、fontGapyプロパティ)」(7-108ページ)	0～フォントサイズ－1(ドット)	0	可
fontGapy	行間ギャップ 文字列を表示するときの行間を設定します。 📖「文字間ギャップ、行間ギャップ(fontGapx、fontGapyプロパティ)」(7-108ページ)	0～フォントサイズ－1(ドット)	1	可
status	液晶表示部(LCD)のステータスエリアを表示する、しないを設定します。 📖「6-2 画面表示(BT-600/1000/1500シリーズ)」(6-12ページ)	true:表示する(下表示) false:表示しない "bottom" :下表示 "top" :上表示	"bottom"	可
update	表示更新 VRAMの内容を液晶表示部(LCD)に反映させる、させないを設定します。 📖「表示更新(updateプロパティ)」(7-110ページ)	true:表示更新する false:表示更新しない	true	可
statusClock	ステータスエリア表示内に時計を表示する、しないを設定します。 📖「6-2 画面表示(BT-600/1000/1500シリーズ)」(6-12ページ)	true:表示する false:表示しない	false	可

名称	説明	範囲	初期値	代入
statusInfo	液晶表示部(LCD)のステータスエリア表示の設定 「ステータスエリア(statusInfoプロパティ)」(7-110ページ)	表示させるアイコンの数字の和を代入します。(0～255) 1:↑アイコンの表示 2:↓アイコンの表示 4:←アイコンの表示 BT-600シリーズのみ、→アイコンの表示 8:→アイコンの表示 BT-600シリーズのみ、←アイコンの表示 16:シフト「S」※2, ※3 32:シフト「あ」※3 64:シフト「A」※3 128:シフト「AB」※3	0	可
coordinate※4	topPos, posx, posyの座標系を変更します。	"text":テキスト座標 "graphic":グラフィック座標	"text"	可

※1 文字間ギャップ「0」、行間ギャップ「0」のときの値です。文字間ギャップや行間ギャップを大きくすると、指定できる座標は小さくなります。

※2 シフトに変更すると英字の入力が可能になります。

※3 アイコン表示のみで、文字入力可能状態にはなりません。

※4 coordinateプロパティの影響範囲は、Screenコントロール内と、Handy:ShowLineMessageメソッドのみです。

※5 compatibleTypeを指定した時に各種設定が初期値に戻ります。

※6 次の関数に影響があります。
Screen:OutputText/Viewer/Clear/DrawLine/DrawBox/Arc
InputString:Exec、InputDecimal:Exec、InputDate:Exec、InputByList:Exec、
Handy:ShowMenu/ShowMenu2/ShowScreenPassword

※7 過去機種との互換のために12および16という値の設定を可能にしています。
fontSize=16 と指定した場合、取得できる値は16です。"large"ではありません。

※8 BT-W100/W80/W70/1000/1500シリーズでのみ使用可能です。
フォント設定の文字間・行間ギャップは設定の変更に連動し、無効→有効ではそれぞれが倍になり、有効→無効ではそれぞれが半分になります。設定値が奇数値の場合、小数点以下は切り捨てとなります。

※9 BT-910互換の場合、グラフィック座標での記述に関して常にX、Y座標を2倍に扱います。

※10「compatibleType = “ BT-910” 」のとき、実行時例外となります。

メソッド

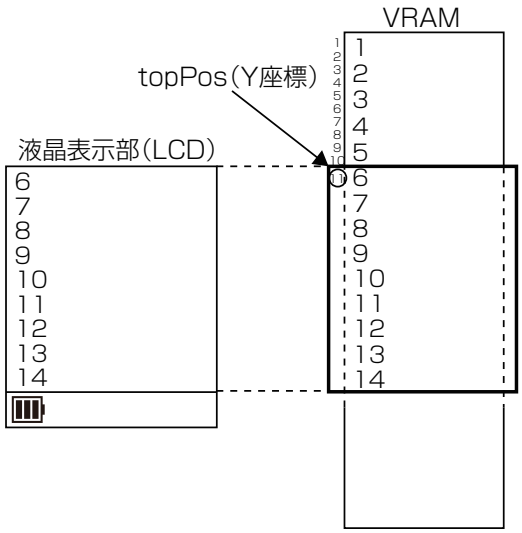
名称	説明	引数	戻り値
Initialize	各プロパティを初期化します。compatibleModeのプロパティ値は初期化されません 。	なし	なし
Clear※1	VRAMをクリアします。	なし	なし
OutputText	文字列をVRAMに書き込みます。	strOutput:出力する文字列※2	なし
DrawLine	直線をVRAMに書き込みます。	(例)BT-W80/W70/1000/1500シリーズ x1:始点のx座標(0～239)※3 y1:始点のy座標(0～1279) x2:終点のx座標(0～239) y2:終点のy座標(0～1279) lineColor:線の描画色 ("white":白、"black":黒、"reverse":反転、nil:「foreColor」で指定したRGB)	なし
DrawBox	四角形をVRAMに書き込みます。	(例)BT-W80/W70/1000/1500シリーズ left:最左のx座標(0～239)※3 top:最上のy座標(0～1279) right:最右のx座標(0～239) bottom:最下のy座標(0～1279) lineColor:線の描画色 ("white":白、"black":黒、"reverse":反転、nil:「foreColor」で指定したRGB) fill:true(四角形を塗りつぶす) false(四角形を塗りつぶさない)	なし
Viewer	液晶表示部(LCD)をスクロール表示します。 「スクロール表示(Viewerメソッド)」(7-110ページ)	title:タイトル文字列(省略不可) top:スクロール開始位置(テキスト座標)※4 height:スクロールする範囲(テキスト座標)※4 step:一度に上下するスクロール量 (1:1行、2:0.5ページ、3:1ページ)	終了時に押されたキー("ENT"/"C"/"RIGHT":「◀」キー/"LEFT":「▶」キー)※5
Read	VRAMの情報をファイルに保存します。	ドライブ番号:ファイル名	true:正常終了 false:エラー終了
Write	ファイルの画面情報をVRAMに展開します。	ドライブ番号:ファイル名	true:正常終了 false:エラー終了
DrawCursor	カーソルをVRAMに書き込みます。	(例)BT-W80/W70/1000/1500シリーズ left:最左のx座標(0～239)※3 top:最上のy座標(0～1279) right:最右のx座標(0～239) bottom:最下のy座標(0～1279) flag:"blink"(点灯) "show"(点灯) "hide"(非表示)	なし
ConvertCoordinate	座標系の計算をします。	val:変換する値 type:変換方式 "text2graphic(x)" "graphic2text(x)" "text2graphic(y)" "graphic2text(y)"	変換されたデータ nil:エラー発生時
Arc	円弧を描画します。	(例)BT-W80/W70/1000/1500シリーズ x:中心x座標(0～239) y:中心y座標(0～1279) start:開始角度(0～719)※6 end:終了角度(1～720)※6 radius:半径(1～240) fill:true(円弧を塗りつぶす) false(円弧を塗りつぶさない)	true:正常終了 false:エラー終了

名称	説明	引数	戻り値
Image	画像ファイルを表示します。(bmp/jpg/gif形式に対応) X2,y2ともにnilを指定すると画像を最大限表示できる座標に設定します。	x1:左上x座標 y1:左上y座標 x2:右下x座標 y2: 右下y座標 filename:ファイル名 (ドライブ番号:ファイル名) transparentColor:透過色	true:正常終了 false:エラー終了

※1 カーソル表示はクリアされません。
※2 文字列に改行コードが含まれていると、「multiColumn」プロパティが「1」以外のときは改行されますが、「1」のときは改行前の文字列のみ表示されます。
※3 グラフィック座標を使用します。
「グラフィック座標」(6-20ページ)
※4 テキスト座標でVRAM のY座標を指定します。
「6-2 画面表示(BT-600/1000/1500シリーズ)」(6-12ページ)
※5 "ENT"/"C"/"RIGHT"/"LEFT"以外のキー戻り値は取得できません。
※6 角度は、start<=endで指定します。

■表示データの指定(topPosプロパティ)

液晶表示部(LCD)に表示をおこなうときに、「topPos」プロパティで、VRAMのどこから表示対象とするかをY座標で指定します。下図は「topPos」プロパティを「11」にしたときの例(テキスト座標)です。



「topPos」プロパティを変更すると、自動的に表示が更新されます。

▶ 重 要

「topPos」プロパティにVRAMの領域外の数値を設定すると実行時例外になります。行間ギャップやフォントサイズを変更するときは注意が必要です。

■文字間ギャップ、行間ギャップ(fontGapx、fontGapyプロパティ)

表示する文字間隔の調整は、文字間ギャップと行間ギャップでおこないます。
文字間ギャップ: X軸方向の座標と座標の間隔
行間ギャップ : Y軸方向の座標と座標の間隔

- 文字間ギャップ、行間ギャップは1座標ごとにドット単位で付加されます。
- 行間ギャップは、設定した値の1/2が各座標に付加されます。

行間ギャップは、設定値が偶数、奇数かで付加のされ方が異なります。
また、文字を拡大表示するときは、文字間ギャップ、行間ギャップも拡大されます。

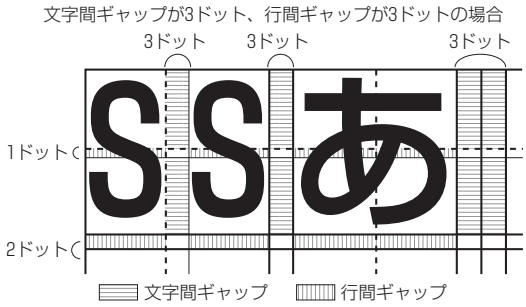
●行間ギャップの設定値が偶数のとき

1行ごとに「行間ギャップ／2」が付加されます。
たとえば、行間ギャップが2ドットのとき、1ドットずつが、各テキスト座標の下部に付加されます。

●行間ギャップの設定値が奇数のとき

- 行間ギャップが奇数値で、テキストのY座標が奇数のとき
  各奇数Y座標の下部に、「(行間ギャップ値-1)/2」の間隔が付加されます。
  たとえば、行間ギャップが3ドットのとき、1ドットずつが付加されます。

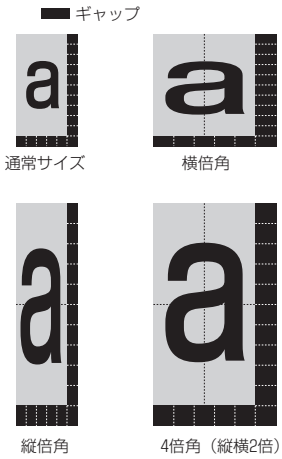
- 行間ギャップが奇数値で、テキストのY座標が偶数のとき
  各偶数Y座標の下部に、「(行間ギャップ値+1)/2」の間隔が付加されます。
  たとえば、行間ギャップが3ドットのとき、2ドットずつが付加されます。



●文字間ギャップ、行間ギャップの拡大

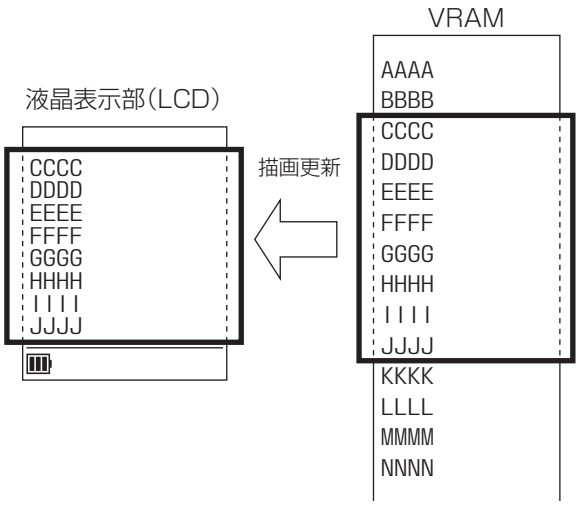
文字を拡大表示するときには、文字間ギャップ、行間ギャップが設定値よりも拡大されます。

文字を拡大表示すると、拡大率に応じて、行間ギャップ、文字間ギャップも拡大されます。
たとえば、4倍角(縦横2倍)文字を表示するときには、行間ギャップが2倍に、文字間ギャップが2倍に拡大されます。



■表示更新(updateプロパティ)

描画処理は、VRAM に対しておこないます。VRAMの内容を液晶表示部(LCD)に反映させるためには表示更新をする必要があります。
「update」プロパティを「false」にすると表示更新しません。



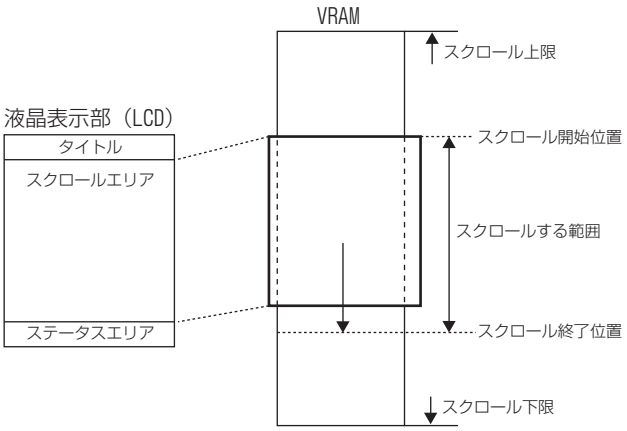
■ステータスエリア(statusInfoプロパティ)

BTシリーズは、ステータスエリア(液晶表示部の最終行)にステータス情報が表示できます。
「statusInfo」プロパティでステータス領域に表示させたいアイコンを設定できます。
たとえば、「◀」「▶」と表示させたいときは「12」、「▲」「▶」と表示させたいときは「9」を設定します。
「statusInfo」プロパティを変更すると、自動的に表示が更新されます。

■スクロール表示(Viewerメソッド)

VRAMのデータを液晶表示部(LCD)に表示させるとき、「Viewer」メソッドを使って画面をスクロールさせることができます。

- スクロールエリアが画面スクロールします。
- 「▲」「▼」キーを押すと「step」数分スクロールします。「ENT」、「C」、「◀」、「▶」キーを押すと終了します。



■表示できない文字の扱い

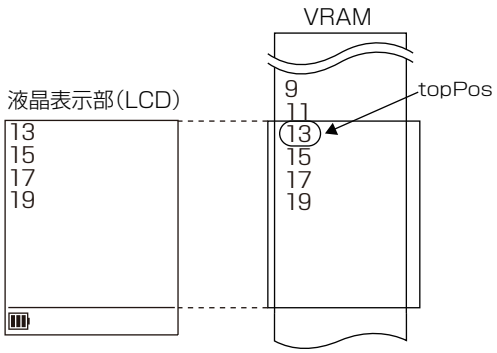
全角文字(2 バイト文字)の 2 バイト目から表示するなど、文字列表示時に不正な指定をした場合には、該当する文字の部分は"@"で表示されます。

使用例

例1　「OutputText」を使用して文字列を表示します。この例ではtopPos=13としています。
実際に液晶表示部(LCD)に表示されるのはVRAM Y座標(topPos=13)以降になります。

```
Method sample1()
Begin
    With Screen
        :Clear()                                    // 初期化
        :coordinate = "text"                        // テキスト座標
        :update = true                              // 描画更新する
        :reverse = false                            // 反転表示しない
        :status = true                              // ステータスエリア表示
        :font = 1 :fontSize = "middle"              // フォントサイズ
        :fontGapx = 0 :fontGapy = 1                 // 文字間･行間ギャップ
        :topPos = 13                                // 先頭VRAM Y座標13
        :posx = 1 :posy = 1 :OutputText("1")
        :posx = 1 :posy = 3 :OutputText("3")
        :posx = 1 :posy = 5 :OutputText("5")
        :posx = 1 :posy = 7 :OutputText("7")
        :posx = 1 :posy = 9 :OutputText("9")
        :posx = 1 :posy = 11 :OutputText("11")
        :posx = 1 :posy = 13 :OutputText("13")
        :posx = 1 :posy = 15 :OutputText("15")
        :posx = 1 :posy = 17 :OutputText("17")
        :posx = 1 :posy = 19 :OutputText("19")
    EndWith

    Handy:KeyWait()
EndMethod
```

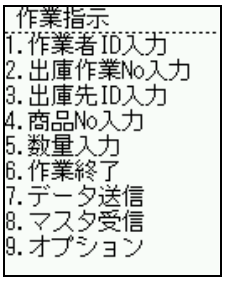
実行結果



例2　「Viewer」を使用してスクロール表示します。

```
Method sample2()
pos[ ] = {1,3,5,7,9,11,13,15,17}
string[ ] = {"1.作業者ID入力","2.出庫作業No入力","3.出庫先ID入力","4.商品No入力",
             "5.数量入力","6.作業終了","7.データ送信","8.マスタ受信","9.オプション"}
i
Begin
    Screen:status = false
    Screen:fontSize = "middle"
    Screen:coordinate = "text"
    Screen:posx=1
    For i=0 to 8
        Screen:posy=pos[i]
        Screen:OutputText(string[i])
    Next

    Screen:Viewer(" 作業指示",1,18,1)
    //タイトル文字列を表示して1行ごとにスクロールさせる

    Handy:KeyWait()      // 入力待ち

EndMethod
```

実行結果



LCD	液晶表示部(LCD)の動作を設定します。

プロパティ

名称	説明	範囲	初期値	代入
status	液晶表示部(LCD)のステータスエリアを表示する、しないを設定します。 📖「6-2 画面表示(BT-600/1000/1500シリーズ)」(6-12ページ)	true:表示する(下表示) false:表示しない “bottom”:下表示 “top”:上表示	“bottom” :下表示	可
update	表示更新 液晶表示部(LCD)の表示を更新する、しないを設定します。	true:表示更新する false:表示更新しない	true	可
statusClock	ステータスエリア表示内に時計を表示する、しないを設定します。 📖「ステータスエリアの表示」(6-14ページ)	true:表示する false:表示しない	false	可
statusInfo	液晶表示部(LCD)のステータスエリア表示の設定 📖「ステータスエリア(statusInfoプロパティ)」(7-114ページ)	表示させるアイコンの数字の和を代入します。(1～255) 1:↑アイコンの表示 2:↓アイコンの表示 4:←アイコンの表示 BT-600シリーズのみ、→アイコンの表示 8:→アイコンの表示 BT-600シリーズのみ、←アイコンの表示 16:シフト「S」 32:シフト「あ」 64:シフト「A」 128:シフト「AB」	0	可
backLightMode	バックライトの動作設定 BT-1000/1500/600シリーズのみ有効です。	“now”:設定直後に点灯。 設定後は、外部要因で点灯します。 “event”:設定直後は点灯しません。外部要因発生時に点灯します。	“event”	可
backLightNormal	動作時のバックライトの明るさを設定します。	“poweroff”:消灯 ※BT-600シリーズのみ “off”:薄灯(オフ) “low”:点灯(暗い) “mid”:点灯(明るい) “high”:点灯(非常に明るい) “auto”:自動 ※BT-1000/1500シリーズのみ	“mid”	可

名称	説明	範囲	初期値	代入
backLightStandby	スタンバイ時のバックライトの明るさを設定します。	“poweroff”:消灯 ※BT-600シリーズのみ “off”:薄灯(オフ) “low”:点灯(暗い) “mid”:点灯(明るい) “high”:点灯(非常に明るい)	BT-W100/W80/W70/1000/1500シリーズ:“low” BT-600シリーズ:“off”	可
backLightTimer	動作時からスタンバイ時に移行するまでの時間	1～127(秒)	10	可
onTime	バックライト点灯時間(Exec呼び出し時に参照される)	1～50(×100ms)	1	可
offTime	バックライト消灯時間(Exec呼び出し時に参照される)	0～50(×100ms)	0	可
loopCount	バックライト点灯･消灯の繰り返し回数(Exec呼び出し時に参照される)	1～100回	1	可
synchronization	バックライト点灯･消灯の完了までの同期･非同期を設定します。	true:同期 false:非同期	true	可

メソッド

名称	説明	引数	戻り値
Initialize	各プロパティを初期化します。	なし	なし
Blink	バックライトのブリンク(点滅)を開始します。 BT-1000/1500/600シリーズのみ有効です。	なし	なし
BlinkStop	バックライトのブリンク(点滅)を停止します。 BT-1000/1500/600シリーズのみ有効です。	なし	なし

■ステータスエリア(statusInfoプロパティ)

BTシリーズではステータスエリア(液晶表示部の最終行)にステータス情報を表示できます。
「statusInfo」プロパティでステータス領域に表示させたいアイコンを設定できます。
たとえば、「◀」「▶」と表示させたいときは「12」、「▲」「▶」と表示させたいときは「9」を設定します。
「statusInfo」プロパティを変更すると、自動的に表示が更新されます。

使用例(BT-600シリーズ向け)

例1 バックライト、タッチパネルの動作を変更します。

```
Method main()
	btvMenupos = 1
	btvItem1 = "動作時消灯|動作時暗い|動作時明るい|動作時最も明るい"
	btvItem2 = "スタンバイ時消灯|スタンバイ時暗い|スタンバイ時明るい|スタンバイ時最も明るい"
	btvItem3 = nil
	btvItem4 = nil
Begin
	While 1
		Screen:fontSize = "middle"
		// メニューを表示
		btvMenupos = 1
		btvMenupos=Handy:ShowMenu("明るさ",btvItem1, btvItem2, btvItem3, btvItem4,
		btvMenupos)
		With LCD
			:backLightTimer = 5                    // スタンバイ時までの時間を5秒に設定します
			Select Case btvMenupos
			Case 1
				:backLightNormal = "poweroff"  // 動作時消灯です
			Case 2
				:backLightNormal = "low"       // 動作時暗いです
			Case 3
				:backLightNormal = "mid"       // 動作時明るいです
			Case 4
				:backLightNormal = "high"      //動作時最も明るいです
			Case 5
				:backLightStandby = "poweroff" // スタンバイ時消灯です
			Case 6
				:backLightStandby = "low"      // スタンバイ時暗いです
			Case 7
				:backLightStandby = "mid"      // スタンバイ時明るいです
			Case 8
				:backLightStandby = "high"     // スタンバイ時最も明るいです
			End Select
		End With
	Wend
End Method
```

例2 ブリンクします。

```
Method main()
        btvMenupos = 1
        btvItem1 = "ブリンク開始|ブリンク終了"
Begin
        While 1
                Screen:fontSize = "middle"
                // メニューを表示
                btvMenupos = 1
                btvMenupos = Handy:ShowMenu("ブリンク",btvItem1,nil,nil,nil,btvMenupos)
                With LCD
                        :onTime = 2
                        :offTime = 2
                        :loopCount = 100
                        :synchronization = false
                        Select Case btvMenupos
                        Case 1
                                :Blink()                        // ブリンクを開始します。
                        Case 2
                                :BlinkStop()                    // ブリンクを終了します。
                        End Select
                End With
        Wend
End Method
```

Drawing	Canvasコントロールで作成したキャンバスに文字列、図形を描画します。

プロパティ

名称	説明	範囲	初期値	代入
lineColor	線色･文字色 線色と文字色をRGBで設定します。 📖「色の指定(lineColor、fillColorプロパティ)」(7-120ページ)	RGBの文字列 (“0～255¦0～255¦0～255”)	“0¦0¦0”	可
fillColor	文字の背景色、または図形領域内の塗りつぶし色 📖「色の指定(lineColor、fillColorプロパティ)」(7-120ページ)	RGBの文字列 (“0～255¦0～255¦0～255”)	“255¦255¦255”	可
fontSize	フォントサイズ 表示文字列のフォントサイズを設定します。	12:12ドットフォント 16:16ドットフォント 24:24ドットフォント	12	可
underline	下線付きの設定 下線付き文字列にするかどうかの設定をします。	true:下線を付ける false:下線を付けない	false	可
strikeThrough	取り消し線の設定 取り消し線付き文字列にするかどうかの設定をします。	nil:なし 1:一重取り消し線を付ける 2:二重取り消し線を付ける	nil	可
bold	太字の設定 表示文字列を太字にするかどうかを設定します。	true:太字にする false:太字にしない	false	可
transparent	背景透過 文字列の背景色を透明にするかどうかを設定します。	true:背景透過にする false:背景透過にしない	true	可
reverse	反転表示 文字色と背景色を反転表示するか、しないかを設定します。	true:反転表示する false:反転表示しない	false	可
magnificate	文字の表示倍率 📖「文字と図形の表示－タッチパネル表示－」(6-5ページ)	1:標準 2:横倍角 3:縦倍角 4:4倍角 9:9倍角 16:16倍角 25:25倍角	1	可
multicolumn	文字列の行折り返し設定 📖「文字と図形の表示－タッチパネル表示－」(6-5ページ)	1:1 行のみ 2:複数行(インデントなし) 3:複数行(インデントあり)	1	可
fontGapx	文字間ギャップ 文字列を表示させるときの文字間隔を設定します。 📖「文字間ギャップ、行間ギャップ(fontGapx、fontGapyプロパティ)」(7-121ページ)	0～5	0	可

名称	説明	範囲	初期値	代入
fontGapy	文字列の行間ギャップ 文字列を表示させるときの行間隔を設定します。 📖「文字間ギャップ、行間ギャップ(fontGapx、fontGapyプロパティ)」(7-121ページ)	0～5	1	可
targetCanvas	描画対象のCanvasを設定します。	Canvasのタグ名	nil	可

メソッド

名称	説明	引数	戻り値
Initialize	各プロパティを初期化します。	なし	なし
Text	文字を描画します。 📖「描画の指定(Text、String、Line、Rectangle、Arcメソッド)」(7-119ページ)	text:文字列 x:描画開始x座標(0～239) y:描画開始y座標(0～1279)	true:正常終了 false:エラー終了
String	下線付き文字、太文字などの装飾文字を描画します。 📖「描画の指定(Text、String、Line、Rectangle、Arcメソッド)」(7-119ページ)	text:文字列 x:描画開始x座標(0～239) y:描画開始y座標(0～1279)	true:正常終了 false:エラー終了
Line	線を描画します。 📖「描画の指定(Text、String、Line、Rectangle、Arcメソッド)」(7-119ページ)	x1:始点のx座標(0～239) y1:始点のy座標(0～1279) x2:終点のx座標(0～239) y2:終点のy座標(0～1279)	true:正常終了 false:エラー終了
Rectangle	矩形を描画します。 📖「描画の指定(Text、String、Line、Rectangle、Arcメソッド)」(7-119ページ)	x:左上x座標(0～239) y:左上y座標(0～1279) height:高さ(1～1280) width:幅(1～240)	true:正常終了 false:エラー終了
Arc	円弧を描画します。 📖「描画の指定(Text、String、Line、Rectangle、Arcメソッド)」(7-119ページ)	x:中心x座標(0～239) y:中心y座標(0～1279) start:開始角度(0～719°)※ end:終了角度(1～720°)※ ratius:半径(1～240)	true:正常終了 false:エラー終了
Clear	キャンバス内の描画図形をすべて削除します。	なし	なし

※ start<=endで指定します。

■ 描画の指定(Text、String、Line、Rectangle、Arcメソッド)

●文字列表示に対応するプロパティ(Text、Stringメソッド)

文字列表示メソッドに対応するプロパティを一覧で示します。

プロパティ	Textメソッド	Stringメソッド
lineColor	○	○
fillColor	○	○
fontSize	○	○
underline	×	○※1
strikethrough	×	○
bold	×	○※2
transparent	×	○
reverse	×	○
magnificate	×	○
multicolumn	×	○
fontGapx	×	○
fontGapy	×	○

※1　fontGapyが2以上のとき
※2　fontGapxが1以上のとき

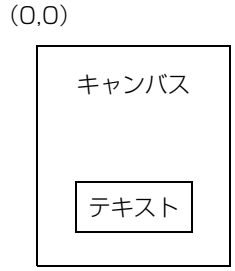
●図形描画に対応するプロパティ(Line、Rectangle、Arcメソッド)

図形描画メソッドに対応するプロパティを一覧で示します。

プロパティ	Lineメソッド	Rectangleメソッド	Arcメソッド
lineColor	○	○	○
fillColor	×	○	○

●座標の指定(Text、String、Line、Rectangle、Arcメソッド)

文字列、図形描画で指定できる座標は、下図のように、描画対象キャンバスに対する相対座標で指定します。



最大X座標＝キャンバスの幅-1
最大Y座標＝キャンバスの高さ-1

文字列の表示位置の指定方法

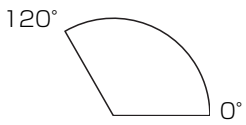
文字列の表示位置は、下図の基準点の座標で指定します。

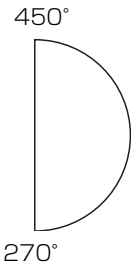
●角度の指定(Arcメソッド)

- 円弧の角度は、下図の例のように0°～720°で指定します。
- 開始角度≦終了角度で指定します。

例1 開始角度＝0°
終了角度＝120°

例2 開始角度＝270°
終了角度＝450°

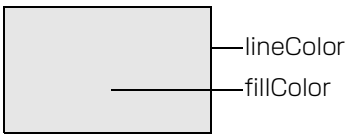




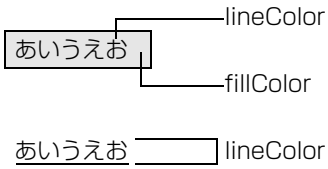
■色の指定(lineColor、fillColorプロパティ)

- 「lineColor」プロパティで、表示文字列と図形の線色を、次のフォーマットで設定できます。
- 「fillColor」プロパティで、図形領域内の色を、次のフォーマットで設定できます。

四角形の場合



文字列の場合



●フォーマット

"Rの値|Gの値|Bの値"

【例】赤色の例:"255|0|0"
緑色の例:"0|255|0"
青色の例:"0|0|255"

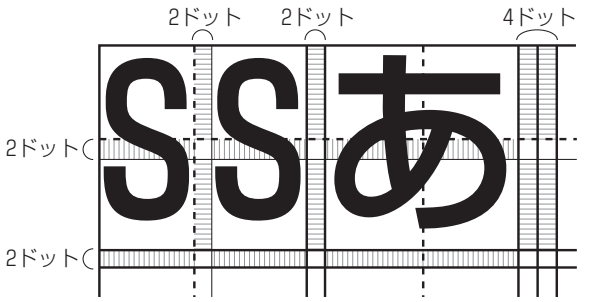
"|"と数字の間にスペースは不要です。

■文字間ギャップ、行間ギャップ(fontGapx、fontGapyプロパティ)

表示する文字間隔の調整は、文字間ギャップと行間ギャップでおこないます。
文字間ギャップ:X軸方向の座標と座標の間隔
行間ギャップ:　Y軸方向の座標と座標の間隔

- 文字間ギャップ、行間キャップは1座標ごとにドット単位で付加されます。
- 行間ギャップは、設定した値の1/2が各座標に付加されます。

文字間ギャップが2ドット、行間ギャップが4ドットの場合



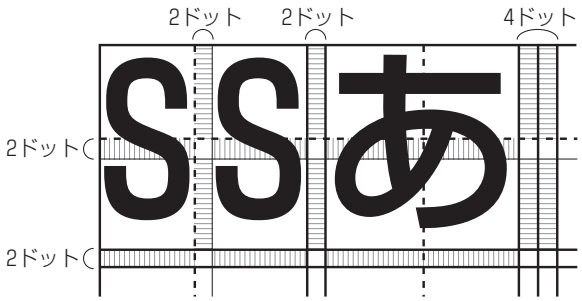
行間ギャップは、設定値が偶数、奇数かで付加のされ方が異なります。
また、文字を拡大表示するときは、文字間ギャップ、行間ギャップも拡大されます。

●行間ギャップの設定値が偶数のとき

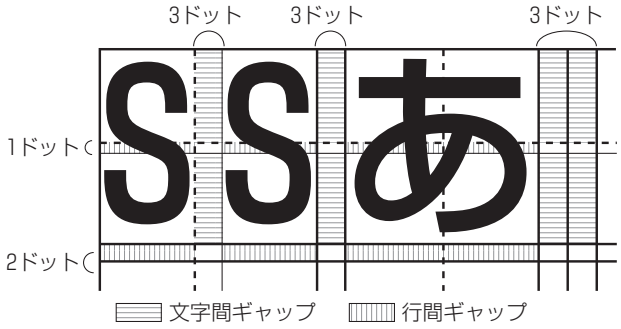
1行ごとに「行間ギャップ／2」が付加されます。
たとえば、行間ギャップが2ドットのとき、1ドットずつが、各テキスト座標の下部に付加されます。

●行間ギャップの設定値が奇数のとき

- 行間ギャップが奇数値で、テキストのY座標が奇数のとき
  各奇数Y座標の下部に、「(行間ギャップ値-1)／2」の間隔が付加されます。
  たとえば、行間ギャップが3ドットのとき、1ドットずつが付加されます。
- 行間ギャップが奇数値で、テキストのY座標が偶数のとき
  各偶数Y座標の下部に、「(行間ギャップ値+1)／2」の間隔が付加されます。
  たとえば、行間ギャップが3ドットのとき、2ドットずつが付加されます。

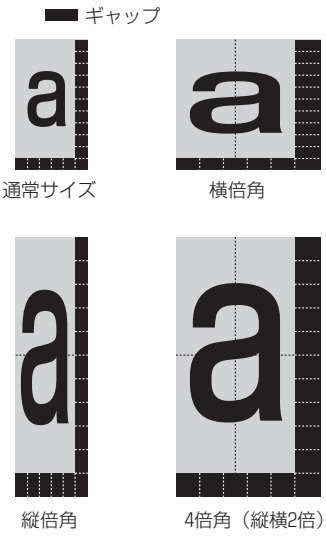


文字間ギャップが3ドット、行間ギャップが3ドットの場合



●文字間ギャップ、行間ギャップの拡大

文字を拡大表示するときには、文字間ギャップ、行間ギャップが設定値よりも拡大されます。



文字を拡大表示すると、拡大率に応じて、行間ギャップ、文字間ギャップも拡大されます。
たとえば、4倍角(縦横2倍)文字を表示するときには、行間ギャップが2倍に、文字間ギャップが2倍に拡大されます。

使用例

例1　キャンバスに文字列を表示します。

```
Method Sample_Text()
Begin

    With Canvas<"canvas1">              // 「canvas1」というタグ名のキャンバスを作成
        :Create("ROOT_WINDOW")
        :top = 0
        :left = 0
        :width = 240
        :height = 320
        :visible = True
        :enable = False
    End With

    With Drawing
        :Initialize()                   // Drawingコントロールのプロパティ値を初期化
        :targetCanvas = "canvas1"       // 文字列を表示するキャンバスは「canvas1」
        :fontSize = 12                  // フォントサイズは12
        :lineColor = "255|0|0"          // 文字列の表示色は赤
        :fillColor = "255|255|255"      // 文字列の背景色は白
        :Text( "SAMPLE TEXT", 0, 0 )    /* 「canvas1」の座標(0,0)に「SAMPLE TEXT」
                                           の文字列が表示されます。*/
    End With

    Window< "ROOT_WINDOW" >:Update()    // ルートウィンドウの表示更新
    Handy:KeyWait()                     // 入力待ち

End Method
```

例2　キャンバスに装飾文字列を表示します。

```
Method Sample_String()
Begin
    With Drawing
        :Initialize()                   // Drawingコントロールのプロパティ値を初期化
        :targetCanvas = "canvas1"       // 文字列を表示するキャンバスは「canvas1」
        :fontSize = 12                  // フォントサイズは12

        Utl:ChangeColor()               // Utl:ChangeColorメソッドの呼び出し
        :fontGapy = 2                   // 文字列の行間ギャップは2
        :underLine = True               // 下線付き
        :String( "SAMPLE STRING", 0, 0 )  /* 「canvas1」の座標(0,0)に
                                             「SAMPLE STRING」が表示されます。*/
```

```
Utl:ChangeColor()
:underLine = False              // 下線付きは無効
:strikeThrough = 1              // 一重取り消し線付き
:String( "SAMPLE STRING", 0, 15 ) /* 「canvas1」の座標(0,15)に
                                    「SAMPLE STRING」が表示されます。*/

Utl:ChangeColor()
:strikeThrough = 2              // 二重取り消し線付き
:bold = True                    // 太字
:fontGapx = 1                   // 文字列間キャップは1
:String( "SAMPLE STRING", 0, 30 ) /*「canvas1」の座標(0,30)に
                                    「SAMPLE STRING」が表示されます。*/

Utl:ChangeColor()
:strikeThrough = nil            // 取り消し線は無効
:bold = True                    // 太字
:fontGapx = 1                   // 文字列間キャップは1
:String( "SAMPLE STRING", 0, 45 ) /*「canvas1」の座標(0,45)に
                                    「SAMPLE STRING」が表示されます。*/

Utl:ChangeColor()
:bold = False                   // 太字は無効
:transparent = True             // 背景は透明
:String( "SAMPLE STRING", 0, 60 ) /*「canvas1」の座標(0,60)に背景が透明な
                                    状態で「SAMPLE STRING」が表示されま
                                    す。*/

Utl:ChangeColor()
:transparent = False            // 背景の透明は無効
:magnificate = 4                // 4倍角文字列
:String( "SAMPLE STRING", 0, 75 ) /*「canvas1」の座標(0,75)に4倍角の
                                    大きさで「SAMPLE STRING」が表示
                                    されます。*/

Utl:ChangeColor()
:magnificate = 1                // 標準の大きさ
:reverse = True                 // 反転表示
:String( "SAMPLE STRING", 0, 120 ) /*「canvas1」の座標(0,120)に
                                    「SAMPLE STRING」が反転表示され
                                    ます。*/

Utl:ChangeColor()
:reverse = False                // 反転表示は無効
:magnificate = 1                // 横倍角文字列
:multiColumn = 2                // 折り返し表示
:String( "SAMPLE STRING0123456789012345", 0, 180 )
```

```
                /* 「canvas1」の座標(0,180)に横倍角の大きさで「SAMPLE
                  STRING0123456789012345」が折り返し表示されます。*/
        End With

        Window< "ROOT_WINDOW" >:Update()    // ルートウィンドウの表示更新
        Handy:KeyWait()                     // 入力待ち

End Method
```

例3　キャンバスに線を描画します。

```
Method Sample_Line()
i
Begin
        With Drawing
                :Initialize()                       // Drawingコントロールのプロパティ値を初期化
                :targetCanvas = "canvas1"           // 線を描画するキャンバスは「canvas1」
                For i = 10 to 200 step 10
                Utl:ChangeColor()                   // Utl:ChangeColorメソッドの呼び出し
                :Line( i, 0, i, 300 )               // 縦線をx方向に10ドットきざみで描画
                Utl:ChangeColor()
                :Line( 0, i, 230, i )               // 横線をy方向に10ドットきざみで描画
                Next
        End With
        Window< "ROOT_WINDOW" >:Update()    // ルートウィンドウの表示更新
        Handy:KeyWait()                     // 入力待ち
End Method
```

例4　キャンバスに四角形を描画します。

```
Method Sample_Rectangle()
Begin
        With Drawing
                :Initialize()                       // Drawingコントロールのプロパティ値を初期化
                :targetCanvas = "canvas1"           // 四角形を描画するキャンバスは「canvas1」
                Utl:ChangeColor()                   // Utl:ChangeColorメソッドの呼び出し
                :Rectangle( 10, 10, 100, 100 )      // 四角形を描画
        End With
        Window< "ROOT_WINDOW" >:Update()    // ルートウィンドウの表示更新
        Handy:KeyWait()                     // 入力待ち
End Method
```

例5　キャンバスに指定する位置に、指定する色で円弧を描画します。

```
Method Sample_Arc()
Begin
    With Drawing
        :Initialize()                   // Drawingコントロールのプロパティ値を初期化
        :targetCanvas = "canvas1"       // 円弧を描画するキャンバスは「canvas1」

        Utl:ChangeColor()               // Utl:ChangeColorメソッドの呼び出し
        :Arc( 10, 10, 0, 90, 10 )       // 円弧を描画

        Utl:ChangeColor()
        :Arc( 30, 10, 90, 180, 10 )

        Utl:ChangeColor()
        :Arc( 50, 10, 180, 270, 10 )

        Utl:ChangeColor()
        :Arc( 70, 10, 270, 360, 10 )

        Utl:ChangeColor()
        :Arc( 10, 30, 90, 360, 10 )

        Utl:ChangeColor()
        :Arc( 10, 50, 180, 450, 10 )

        Utl:ChangeColor()
        :Arc( 10, 70, 270, 540, 10 )
        Utl:ChangeColor()
        :Arc( 10, 90, 360, 630, 10 )
    End With

    Window< "ROOT_WINDOW" >:Update()    // ルートウィンドウの表示更新
    Handy:KeyWait()                     // 入力待ち

End Method

Package Utl
flg = 0
// 色を変更するメソッド
Method ChangeColor()                                /* 描画する線色と塗りつぶし色を
                                                       設定するメソッド */

Begin
    Select Case flg
        case 1
            Drawing:lineColor = "0|0|0"             // BLACK
            Drawing:fillColor = "0|0|0"
        case 2
            Drawing:lineColor = "255|0|0"           // RED
            Drawing:fillColor = "255|0|0"
```

```
            case 3
                Drawing:lineColor = "0|255|0"          // LIME
                Drawing:fillColor = "0|255|0"
            case 4
                Drawing:lineColor = "0|0|255"          // BLUE
                Drawing:fillColor = "0|0|255"
            case 5
                Drawing:lineColor = "255|255|0"        // YELLOW
                Drawing:fillColor = "255|255|0"
            case 6
                Drawing:lineColor = "0|255|255"        // CYAN
                Drawing:fillColor = "0|255|255"
            case 7
                Drawing:lineColor = "255|0|255"        // MAGENTA
                Drawing:fillColor = "255|0|255"
            case 8
                Drawing:lineColor = "255|255|255"      // WHITE
                Drawing:fillColor = "255|255|255"
            case 9
                Drawing:lineColor = "0|128|0"          // GREEN
                Drawing:fillColor = "0|128|0"
            case 10
                Drawing:lineColor = "128|0|128"        // PURPLE
                Drawing:fillColor = "128|0|128"
            case 11
                Drawing:lineColor = "128|0|0"          // MAROON
                Drawing:fillColor = "128|0|0"
            case 12
                Drawing:lineColor = "0|0|128"          // NAVY
                Drawing:fillColor = "0|0|128"
            case 13
                Drawing:lineColor = "192|192|192"      // SILVER
                Drawing:fillColor = "192|192|192"
            case 14
                Drawing:lineColor = "135|206|235"      // SKYBLE
                Drawing:fillColor = "135|206|235"
            case 15
                Drawing:lineColor = "255|20|147"       // DEEPPINK
                Drawing:fillColor = "255|20|147"
            case 16
                Drawing:lineColor = "127|255|0"        // CHARTREUSE
                Drawing:fillColor = "127|255|0"
        End Select

        flg = flg + 1
        If flg == 17 Then flg = 1 End If
End Method

End Package
```

Dialog	メニュー、メッセージを表示するダイアログボックスを設定します。

メソッド

名称	説明	引数	戻り値
Menu	ダイアログボックスにメニューを表示します。 「メニュー表示(Menuメソッド)」(7-130ページ)	title:メニューのタイトル (最大8192バイト) items:メニュー内容の文字列 (最大8192バイト、1アイテム最大1023バイト)※1 initItem:表示時に選択するアイテム (1～16) line:メニュー表示列(1～2)	選択されたメニュー項目番号※2 "C":「C」キー長押し "ERR":引数エラー
AsyncMessage	非同期メッセージボックスを表示します。※5 「メッセージボックス表示(Messageメソッド)」(7-131ページ)	title:タイトル文字列(最大127バイト) str:表示する文字列(最大1023バイト) show:true(表示) false(非表示) frm:""(未使用)※4 icon:""(未使用)※4 font:フォントサイズ ●BT-W100/W80/W70/1000/1500シリーズ "small":16×16フォント "middle":24×24フォント "large":32×32フォント 12:"middle"として扱います。 16:"large"として扱います。 ●BT-600シリーズ "small":12×12ドットフォント "middle":14×14ドットフォント "large":16×16ドットフォント 12:12×20ドットフォント(互換用) 16:"large"として扱います。	true:正常終了 false:エラー終了

名称	説明	引数	戻り値
Message	アイコン付きメッセージボックスを表示します。 「メッセージボックス表示(Messageメソッド)」(7-131ページ)	title:タイトル文字列(最大127バイト) str:表示する文字列(最大1023バイト) btn:表示ボタンと初期値 “ok”※3 “confirm”※3 “okcancel|ok”※3 “okcancel|cancel”※3 “yesno|yes”※3 “yesno|no”※3 frm:“”(未使用)※4 icon:“”(未使用)※4 font:フォントサイズ ●BT-W100/W80/W70/1000/1500シリーズ “small”:16×16フォント “middle”:24×24フォント “large”:32×32フォント 12:“middle”として扱います。 16:"large"として扱います。 ●BT-600シリーズ "small":12×12ドットフォント "middle":14×14ドットフォント "large":16×16ドットフォント 12:12×20ドットフォント(互換用) 16:"large"として扱います。	true:“ok” “yes” “confirm” のどれかが選択されたとき false:“no” “cancel” のどれかが選択されたとき

※1 表示できるメニューは16項目です。各項目間を“|”で区切って設定します。
※2 “1～16”が返ります。キャンセル時には“C”が返ります。
※3 引数「btn」の内容は次のようになります。
“ok” :OKボタンのみ
“confirm” :確認ボタンのみ
“okcancel|ok” :“OK/キャンセル”、初期値は“OK”
“okcancel|cancel” :“OK/キャンセル”、初期値は“キャンセル”
“yesno|yes” :“はい/いいえ”、初期値は“はい”
“yesno|no” :“はい/いいえ”、初期値は“いいえ”
※4 旧互換用なので「frm」、「icon」に設定した値は無視されます。
※5 以下のプロパティの設定、メソッド呼び出しをすると、非同期メッセージボックス表示が消えます。
Screen:fontSize
Screen:fontGapx/fontGapy
Screen:Viewer
Screen:OutputText
Screen:Clear
Screen:topPos
Screen:Read/Write
Screen:update
Screen:DrawLine/DrawBox/Arc/Image/DrawCurSor
Screen:compatibleMode(※BT-1000/1500シリーズのみ)
InputString/InputDecimal/InputDate/InputByList:Exec
Handy:ShowMenu/ShowMenu2
Handy:ShowMessageBox
Handy:LineMessageBox
Handy:ShowClickVolumeDialog
Handy:ShowScreenPassword
Dialog:全関数

■メニュー表示(Menuメソッド)

「Menu」メソッドを使うと、ダイアログボックスにメニューを表示させて、メニュー項目から選択入力することができます。

- メニュー項目数は、最大16個です。
- itemsに、16個までのメニュー項目を設定できます。
  メニュー項目がないときは、必ずnilに設定する必要があります。
  各項目間は“|”で区切ります。したがって、“|”は表示できません。

例1　下図のような1列のメニューを表示します。

各引数は、次のように設定します。

title=“メニュー”
items=“ジョブ1|ジョブ2|ジョブ3|ジョブ4|ジョブ5|ジョブ6|ジョブ7|ジョブ8|ジョブ9|ジョブ10”
initItem=1
line=1

たとえば、「ジョブ5」を選択すると、「5」という値が戻り値になります。

例2　下図のような2列のメニューを表示します。

各引数は、次のように設定します。
title=“メニュー”
items=“ジョブ1|ジョブ2|ジョブ3|ジョブ4|ジョブ5|ジョブ6|ジョブ7|ジョブ8|ジョブ9|ジョブ10”
initItem=1
line=2

■メッセージボックス表示(Messageメソッド)

「Message」メソッドを使用すると、メッセージボックスを表示できます。

●ボタンの設定

引数「btn」の設定したときの動作は、次のようになります。

- 「ok」または「confirm」を設定した場合、「OK」または「確認」ボタンだけが表示されます。
- 「okcancel|ok」を設定した場合、「OK」と「キャンセル」ボタンが表示されて、カーソル位置は「OK」になります。
- 「okcancel|cancel」を設定した場合、「OK」と「キャンセル」ボタンが表示されて、カーソル位置は「キャンセル」になります。

■非同期メッセージボックス表示(AsyncMessageメソッド)

「AsyncMessage」メソッドを使用すると、非同期のメッセージボックスを表示できます。

使用例

メニューで選択するダイアログボックスを表示します。

```
Method main()
        ret
        date
Begin
        While(1)

        // メニューの表示
                ret = Dialog:Menu("Dialogサンプル",
                                "1.Message1|2.Message2|3.Message3","1","1")

        // 選択したメニューのダイアログボックスを表示
                Select Case ret
```

```
        case 1
            Dialog:Message("Message1","ダイアログのサンプル1",
                           "okcancel|ok","","","small")
        case 2
            Dialog:Message("Message2","ダイアログのサンプル2",
                           "confirm","","","middle")
        case 3
            Dialog:Message("Message3","ダイアログのサンプル3",
                           "yesno|no","","","large")
        EndSelect
    Wend

End Method
```


case 1

case 2

case 3

7-6 読み取り動作の設定

読み取り動作の設定をおこないます。
読み取り実行前に設定します。

JAN	JAN、EAN、UPCの読み取り動作を設定します。

チェックディジットは常に有効です。
読み取ったバーコードはチェックディジットを含めて画面表示されます。

プロパティ

名称	説明	範囲	初期値	代入
useCD	JANのチェックディジット判定の有効にするかどうかを設定します。 カメラタイプでは、常に有効設定になり ます。	true: 有効にする false: 無効にする	true	可
addOn	アドオン桁数の設定 アドオンコードを読み取るときは、桁数「2」または「5」に設定します。 両方読みとるときは、「7」に設定します。※2	0:設定しない 2:2桁 5:5桁 7:2桁および5桁	BT-W100/W80/W70/1000/600シリーズ:5 BT-1500シリーズ:7	可
addNS	NSを付加する、しないを設定します。 UPC-E対応 が有効のときのみ(NS)を付加します。	true:付加する false:付加しない	true	可
gtin	GTINに対応する、しないを設定します。 GTINに対応すると、先頭に0を追加して、14桁に整形されます。	true:対応する false:対応しない	false	可
maxAddOnGap	アドオンギャップ最大値	0～255	224	可
minAddOnGap	アドオンギャップ最小値	0～255	96	可
marginMode	マージンの処理モードを指定します。 カメラタイプのみ有効です。	0:通常マージン 1:ショートマージン	0	可
enableUPCE	UPC-E対応	true:読み込む false:読まない	true	可
enableJAN8	JAN8対応	true:読み込む false:読まない	true	可
enableJAN13	JAN13対応	true:読み込む false:読まない	true	可
prefix	プレフィックスを付加するかどうかを指定します。	true:付加する false:付加しない	true	可
enable※1	このバーコードを有効にする、しないを設定します。 trueに設定すると、InputString、BCRコントロールでバーコードを読み取ることができます。	true:有効にする false:無効にする	true	可

※1 バーコード/2次元コード読み取り動作設定コントロールの共通プロパティです。他のコントロールでも使用します。

※2 BT-1000シリーズでは「7」を設定した時、「5」の時と同じになります。2桁と5桁を両方読み取る設定はできません。

メソッド

名称	説明	引数	戻り値
Initialize	各プロパティを初期化します。	なし	なし

CODE39	CODE39の読み取り動作を設定します。

プロパティ

名称	説明	範囲	初期値	代入
useCD※1	チェックディジットの有効、無効を設定します。 trueに設定すると、バーコード読み取りにチェックディジットを使用できます。	true：有効にする false：無効にする	false	可
includeCD※1	チェックディジットをデータに含める、含めないを設定します。 useCDプロパティがtrueのときのみ有効です。	true：含める false：含めない	true	可
includeStartChar※1	スタートアップキャラクタを含める、含めないを設定します。 trueに設定すると、バーコードのはじめと終わりに、「＊」(スタート/ストップコード)が付加されます。	true：含める false：含めない	false	可
min※1	最小桁数※2 読み取り桁数の最小桁数を設定します。	3～50	3	可
max※1	最大桁数※2 読み取り桁数の最大桁数を設定します。	3～50	50	可
marginMode	マージンの処理モードを指定します。 カメラタイプのみ有効です。	0：通常マージン 1：ショートマージン	0	可
enable※1	このバーコードを有効にする、しないを設定します。	true：有効にする false：無効にする	true	可

※1 バーコード/2次元コード読み取り動作設定コントロールの共通プロパティです。他のコントロールでも使用します。

※2 最小桁数≦最大桁数になるように設定します。他の場合は実行時例外になります。

メソッド

名称	説明	引数	戻り値
Initialize	各プロパティを初期化します。	なし	なし

EAN128 / CODE128	GS1-128(EAN128)、CODE128の読み取り動作を設定します。

プロパティ

名称	説明	範囲	初期値	代入
useCD※1※4	チェックディジットの有効、無効を設定します。	true:有効にする false:無効にする	true	可
includeCD※1※4	チェックディジットをデータに含める、含めないを設定します。 useCDプロパティがtrueのときのみ有効です。	true:含める false:含めない	false	可
separator※2	可変長データの区切りを表す[FNC1]を変換する文字を設定します。	0x1～0xFF (ASCIIコード)	0x20	可
min※1	最小桁数※3	1～100	1	可
max※1	最大桁数※3	1～100	100	可
marginMode	マージンの処理モードを指定します。 カメラタイプのみ有効です。	0:通常マージン 1:ショートマージン	0	可
enable※1	このバーコードを有効にする、しないを設定します。	true:有効にする false:無効にする	true	可

※1 バーコード/2次元コード読み取り動作設定コントロールの共通プロパティです。他のコントロールでも使用します。ここでは、共通プロパティの説明を省略しています。
共通プロパティについては、「CODE39」(7-135ページ)を参照してください。

※2 「separator」はGS1-128のみで使用します。

※3 スタートコードがCODE-Cの場合、数字2桁を1桁として認識します。

※4 BT-1500 シリーズでは、useCD プロパティの設定に関係なく、常にチェックディジットが有効になります。また、includeCDプロパティの設定に関係なく、チェックディジットをデータに含めません。BT-W100/W80/W70シリーズのカメラタイプでは、CODE128のみ上記の動作になります。

メソッド

名称	説明	引数	戻り値
Initialize	各プロパティを初期化します。	なし	なし

ITF	ITFの読み取り動作を設定します。

プロパティ

名称	説明	範囲	初期値	代入
useCD※	チェックディジットの有効、無効を設定します。	true：有効にする false：無効にする	false	可
includeCD※	チェックディジットをデータに含める、含めないを設定します。 useCDプロパティがtrueのときのみ有効です。	true：含める false：含めない	true	可
min※	最小桁数（偶数）	2～50	4	可
max※	最大桁数（偶数）	2～50	50	可
marginMode	マージンの処理モードを指定します。 カメラタイプのみ有効です。	0：通常マージン 1：ショートマージン	0	可
enable※	このバーコードを有効にする、しないを設定します。	true：有効にする false：無効にする	true	可

※ バーコード/2次元コード読み取り動作設定コントロールの共通プロパティです。他のコントロールでも使用します。ここでは、共通プロパティの説明を省略しています。
共通プロパティについては、「CODE39」（7-135ページ）を参照してください。

メソッド

名称	説明	引数	戻り値
Initialize	各プロパティを初期化します。	なし	なし

NW7	NW7(Codabar)の読み取り動作を設定します。

プロパティ

名称	説明	範囲	初期値	代入
typeofCD	チェックディジット形式の設定 NW7コントロールには7種類のチェックディジット形式があります。	nil:なし 0:モジュラス16 1:モジュラス11 2:モジュラス10/2ウェイト 3:モジュラス10/3ウェイト 4:7チェックDR 5:加重モジュラス11 6:ルーンズ	nil	可
includeCD※	チェックディジットをデータに含める、含めないを設定します。 typeofCDプロパティがnil以外のときのみ有効です。	true:含める false:含めない	true	可
includeStartChar※	スタートアップキャラクタを設定します。	nil:なし 0:小文字 1:大文字	0	可
min※	最小桁数	3～50	4	可
max※	最大桁数	3～50	50	可
marginMode	マージンの処理モードを指定します。 カメラタイプのみ有効です。	0:通常マージン 1:ショートマージン	0	可
enable※	このバーコードを有効にする、しないを設定します。	true:有効にする false:無効にする	true	可

※ バーコード/2次元コード読み取り動作設定コントロールの共通プロパティです。他のコントロールでも使用します。ここでは、共通プロパティの説明を省略しています。
共通プロパティについては、「CODE39」(7-135ページ)を参照してください。

メソッド

名称	説明	引数	戻り値
Initialize	各プロパティを初期化します。	なし	なし

CODE93	CODE93の読み取り動作を設定します。

プロパティ

名称	説明	範囲	初期値	代入
useCD※	チェックディジットの有効、無効を設定します。 レーザータイプのみ有効です。	true：有効にする false：無効にする	true	可
includeCD※	チェックディジットをデータに含める、含めないを設定します。 useCDプロパティがtrueのときのみ有効です。 レーザータイプのみ有効です。	true：含める false：含めない	false	可
min※	最小桁数	1～50	1	可
max※	最大桁数	1～50	50	可
marginMode	マージンの処理モードを指定します。 カメラタイプのみ有効です。	0：通常マージン 1：ショートマージン	0	可
enable※	このバーコードを有効にする、しないを設定します。	true：有効にする false：無効にする	true	可

※ バーコード/2次元コード読み取り動作設定コントロールの共通プロパティです。他のコントロールでも使用します。ここでは、共通プロパティの説明を省略しています。
共通プロパティについては、「CODE39」(7-135ページ)を参照してください。

メソッド

名称	説明	引数	戻り値
Initialize	各プロパティを初期化します。	なし	なし

TOF / COOP

インダストリアル2of5、COOPの読み取り動作を設定します。

インダストリアル2of5は、TOFコントロールを使用します。

プロパティ

名称	説明	範囲	初期値	代入
min※	最小桁数	1～50	4	可
max※	最大桁数	1～50	50	可
enable※	このバーコードを有効にする、しないを設定します。	true：有効にする false：無効にする	true	可

※ バーコード/2次元コード読み取り動作設定コントロールの共通プロパティです。他のコントロールでも使用します。ここでは、共通プロパティの説明を省略しています。
共通プロパティについては、「CODE39」(7-135ページ)を参照してください。

メソッド

名称	説明	引数	戻り値
Initialize	各プロパティを初期化します。	なし	なし

RSS	GS1 DataBar(RSSコード)の読み取り動作を設定します。

プロパティ

名称	説明	範囲	初期値	代入
min※	最小桁数	1～74	1	可
max※	最大桁数	1～74	74	可
symbolType	GS1 DataBarのタイプを設定します。	読み取り設定したタイプの数字を足した値を代入します。 〈1～31〉 1:GS1 DataBar Expanded 2:GS1 DataBar Expanded Stacked 4:GS1 DataBar Limited 8:GS1 DataBar 16:GS1 DataBar Stacked 31:すべて読み取る	31	可
marginMode	マージンの処理モードを設定します。 BT-1000/600シリーズでのみ有効です。	0:マージン必要 1:マージン不要、通常コードのみ 2:マージン不要、白黒反転コードのみ	0	可
enable※	このコードを有効にする、しないを設定します。	true:有効にする false:無効にする	true	可

※ バーコード/2次元コード読み取り動作設定コントロールの共通プロパティです。他のコントロールでも使用します。ここでは、共通プロパティの説明を省略しています。
共通プロパティについては、「CODE39」(7-135ページ)を参照してください。

メソッド

名称	説明	引数	戻り値
Initialize	各プロパティを初期化します。	なし	なし

QR	QRコードの読み取り動作を設定します。

ポイント

レーザータイプでは、設定は可能ですが、読み取りできません。

プロパティ

名称	説明	範囲	初期値	代入
min※	最小桁数	1～7089	1	可
max※	最大桁数	1～7089	7089	可
enableMicro	マイクロQRコードを有効にする、しないを設定します。	true：有効にする false：無効にする	true	可
scanDataSize	読み取りコードのデータサイズを設定します。 BT-1500シリーズのみ有効です。	0：通常データサイズ 1：小データサイズ	0	可
enable※	この2次元コードを有効にする、しないを設定します。	true：有効にする false：無効にする	true	可

※ バーコード/2次元コード読み取り動作設定コントロールの共通プロパティです。他のコントロールでも使用します。ここでは、共通プロパティの説明を省略しています。
共通プロパティについては、「CODE39」(7-135ページ)を参照してください。

ポイント

連結QRコードを読み取った場合には、それぞれ個別のQRコードとして読み取ります。データを連結させたい場合は、BCRコントロールのGetData, GetQRDataメソッドなどを利用して、読み取った各データを連結させてください。

メソッド

名称	説明	引数	戻り値
Initialize	各プロパティを初期化します。	なし	なし

DataMatrix	DataMatrixの読み取り動作を設定します。

ポイント

- レーザータイプでは、設定は可能ですが、読み取りできません。
- DataMatrix(ECC200)にのみ対応しています。

プロパティ

名称	説明	範囲	初期値	代入
min※	最小桁数	1～3116	1	可
max※	最大桁数	1～3116	3116	可
symbolType	DataMatrix(ECC200)の正方形タイプ、長方形タイプの読み取りを設定します。	"both":両方 "square":正方形のみ	"both"	可
scanDataSize	読み取りコードのデータサイズを設定します。BT-1500シリーズのみ有効です。	0:通常データサイズ 1:小データサイズ	0	可
enable※	この2次元コードを有効にする、しないを設定します。	true:有効にする false:無効にする	true	可

※ バーコード/2次元コード読み取り動作設定コントロールの共通プロパティです。他のコントロールでも使用します。ここでは、共通プロパティの説明を省略しています。
共通プロパティについては、「CODE39」(7-135ページ)を参照してください。

メソッド

名称	説明	引数	戻り値
Initialize	各プロパティを初期化します。	なし	なし

PDF417	PDF417コードの読み取り動作を設定します。

ポイント

レーザータイプでは、設定は可能ですが、読み取りできません。

プロパティ

名称	説明	範囲	初期値	代入
min※	最小桁数	1～2710	1	可
max※	最大桁数	1～2710	2710	可
enableMicro	マイクロPDFを有効にする、しないを設定します。	true：有効にする false：無効にする	true	可
enable※	この2次元コードを有効にする、しないを設定します。	true：有効にする false：無効にする	true	可

※ バーコード/2次元コード読み取り動作設定コントロールの共通プロパティです。他のコントロールでも使用します。ここでは、共通プロパティの説明を省略しています。
共通プロパティについては、「CODE39」(7-135ページ)を参照してください。

メソッド

名称	説明	引数	戻り値
Initialize	各プロパティを初期化します。	なし	なし

Composite	合成シンボル(GS1(EAN/UCC)Composite)の読み取り動作を設定します。

ポイント

• レーザータイプでは、設定は可能ですが、読み取りできません。
• 合成シンボルとして合成するバーコード部(GS1 DataBar(RSS)、JAN、GS1-128(EAN128))の読み取りを有効にする必要があります。

プロパティ

名称	説明	範囲	初期値	代入
separator	バーコード部と2次元コード部の間に入れる文字を設定します。	0x1～0xFF(ASCIIコード)	0x20	可
enable※	この2次元コードを有効にする、しないを設定します。	true:有効にするfalse:無効にする	true	可

※ バーコード/2次元コード読み取り動作設定コントロールの共通プロパティです。他のコントロールでも使用します。ここでは、共通プロパティの説明を省略しています。
共通プロパティについては、「CODE39」(7-135ページ)を参照してください。

メソッド

名称	説明	引数	戻り値
Initialize	各プロパティを初期化します。	なし	なし

BCR	バーコードを読み取るときの動作設定をおこないます。

バーコードの読み取りに関する各種設定をおこないます。

プロパティ

名称	説明	範囲	初期値	代入
trigger	トリガ状態 レーザのON/OFFを制御します。	true:On false:Off	false	可
data	読み取りデータ	文字列(最大8192バイト)	nil	不可
convertedData	ヘッダ/フッタ付加、文字置換実行後の読み取りデータ	文字列(最大8192バイト)	nil	不可
header	読み取りデータにヘッダ文字列を付加します。	true:On false:Off	false	可
terminator	読み取りデータにターミネータ文字列を付加します。	true:On false:Off	false	可
replace	読み取りデータに含まれる制御コードの文字列置換を行います。	true:On false:Off	false	可
reverse	白黒反転コードの読み取り設定 カメラタイプのみ有効です。	"negative":白黒反転 "positive":通常 "both":両方読み取る	"positive"	可
scanType2D	読み取りモード カメラタイプのみ有効です。	"first":最速 "center":中心 "full":多段読み "line":水平ライン "limited":ラインスキャン (BT-W350/W370/W250のみ有効)	"first"	可
count	読み取りデータ数 カメラタイプで、scanType2Dが"full"(多段読み)のときに、読み取れたバーコード/2次元コードの数が返されます。	0～4(QRコード以外) 0～16(QRコード)	0	不可
inverse	表裏反転コードの読み取り設定 カメラタイプで2次元コードを読み取る場合のみ有効です。	"normal":通常 "mirror":表裏反転	"normal"	可
lastType	最後に読み取ったコード種別が格納されます。	"JAN" "CODE39" "EAN128" "CODE128" "ITF" "RSS" "NW7" "CODE93" "2of5" "COOP" "QR" "DataMatrix" "PDF417" "Maxi" "Composite" "OCR"	nil	不可
compareCount	読み取り一致回数 読み取りOKと判定するために、同じバーコードを読み取る回数を設定します。	1～100(回)	2	可
timeOut※1	タイムアウト時間 読み取り開始後、読み取り終了するまでのタイムアウト時間を設定します。	1～255(×100ミリ秒)	100	可
scanTimeOut※2	スキャンタイムアウト時間 レーザ/ターゲットポインタ点灯後、読み取り開始するまでのタイムアウト時間を設定します。	1～255(×100ミリ秒)	100	可
ignoreIdentical※3	二度読み防止時間 読み取り後、設定した時間内に同じ内容のバーコードを読み取っても無効とします。二度読み防止機能を無効にする場合は、「0」に設定します。	0～255(×100ミリ秒)	20	可

名称	説明	範囲	初期値	代入
quality	PMI※5 カメラタイプでは、2次元コード読み取り時のみ有効です。	0～100	0	不可
codeLink	バーコード合成対象のコード レーザータイプのみ有効です。	"JAN"/"CODE39"/"EAN128"/"CODE128"/"ITF"/"NW7"/"CODE93"/nil	nil	可
composite	読み取ったコードがコンポジットコードの一部であるか	"not_linked" コンポジットコードではない "linked_c128" コンポジットコードの一部 "linked_rss" コンポジットコードの一部であるGS1 Databar "linked_cc_a" コンポジットCC_Aを構成する2Dコード "linked_cc_b" コンポジットCC_Bを構成する2Dコード "linked_cc_c" コンポジットCC_Cを構成する2Dコード	nil	不可
isCodeLink※4	読み取ったコードがバーコード合成か否か レーザータイプのみ有効です。	true：合成 false：合成ではない	false	不可
scanMode2D	読み取り動作モードを指定します。 カメラタイプのみ有効です。	"balance":バランスモード "blackErosion"：黒収縮フィルタモード "blackDilation"：黒膨張フィルタモード "smooth"：平滑化フィルタモード "high1"：熟考1モード "high2"：熟考2モード "distance":高分解能(BT-W350/W370/W250のみ有効) "dpm":DPMモード(BT-W370のみ有効)	BTシリーズの設定値	可
scanIllumination	読み取り時にLED照明を点灯するかどうかを設定します。 カメラタイプのみ有効です。	true:照明ON false:照明OFF	true	可
focusMode	読み取り時の焦点モード設定します。 BT-W350/W370/W250のみ有効です。	"auto"：オートフォーカス "fixed"：固定焦点	BT-W350/W370/W250の設定値	可
focusPosAF ※6	オートフォーカス設定時に読み取り開始時の焦点位置を設定します。 BT-W350/W370/W250のみ有効です。	60：近距離 180：標準 450：遠距離 -180：前回値	BT-W350/W370/W250の設定値	可
focusPosFixed ※7	固定焦点時の焦点位置を設定します。 BT-W350/W370/W250のみ有効です。	60：近距離 180：標準 450：遠距離	BT-W350/W370/W250の設定値	可
laserMode ※8･9	レーザ照射幅を設定します。 BT-W300/W200のみ有効です。	"wide"：標準 "narrow"：遠距離 "auto"：自動	BT-W300/W200の設定値	可
liveViewDecoding	読取中ライブビュー制御を指定します。 BT-W370のみ有効です。	true: ライブビューON false: ライブビューOFF	true	可
distanceLED	読み取り距離表示LED制御を指定します。 BT-W370のみ有効です。	true: 近距離でLED表示する false: 近距離でLED表示しない	true	可

名称	説明	範囲	初期値	代入
scanExtIllumination	照明パターンを指定します。 BT-W370のみ有効です。	("0～1 | 0～1 | 0～1 | 0～1")※ ※ "|"と数値の間にスペースは不要です。 ※ "|"区切りで左から、内部照明(0:無効、1:有効)、マルチアングル照明(0:無効、1:有効)、ローアングル照明(0:無効、1:有効)、3D照明(0:無効、1:有効)の順に指定します。	"1|1|1|0"	可
scanExtTargetBrightness	ターゲット輝度を指定します。照明パターンごとに撮像画像の明るさと、輝度調整の対象とする領域の広さを設定します。 BT-W370のみ有効です。	("0～255 | 0～255 | 0～255 | 0～255 | 0～2") ※ ※ "|"と数値の間にスペースは不要です。 ※ "|"区切りで左から、ターゲット輝度値(内部照明)、ターゲット輝度値(マルチアングル照明)、ターゲット輝度値(ローアングル照明)、ターゲット輝度値(3D照明)、ターゲットエリア(0:エリア小、1:エリア中、2:エリア大)　の順に指定します。	"128|128|180|0|1"	可
extAttachment	照明拡張アタッチメントの接続有無を指定します。 BT-W370のみ有効です。	true:On false:Off	FALSE	可
threeDMode	3D照明設定。凹凸設定を指定します。 BT-W370のみ有効です。	"recessed":凹設定 "raised":凸設定 "both":凹凸設定	"both"	可
scanExtLowDirection	ローアングル照明の照明方向を指定します。 ※BT-W370のみ有効です。	("0～1 | 0～1 | 0～1 | 0～1")※ ※ "|"区切りで左から、ローアングル照明の上方向、右方向、左方向、下方向の順に指定します。	"0 | 0 | 0 | 1"	可

※1 「timeOut」は、「Input」メソッドを使用した場合、または、「トリガモード」が、「オートオフ」「離して読み」「ダブルクリック読み」「ソフトウェアトリガ」の場合に有効です。
※2 「scanTimeOut」は、「trigger」プロパティを使用した場合、または、「トリガモード」が、「離して読み」「ダブルクリック読み」の場合に有効です。
※3 「ignoreIdentical」は、「トリガモード」が、「連続読み取り」「ソフトウェアトリガ」の場合のみ有効です。
※4 「isCodeLink」の設定は、バーコードが読めたタイミングでおこないます。
※5 ·レーザータイプの場合
　読み取り品質＝(成功時の読み取り一致回数／スキャン回数)×100
　·カメラタイプの場合
　読み取り品質＝(100－誤り訂正符号の使用率)
※6 「focusPosAF」は、「focusMode」が「オートフォーカス」の場合のみ有効です。
※7 「focusPosFixed」は、「focusMode」が「固定焦点」の場合のみ有効です。
※8 遠距離／自動の設定は、ナロー幅が 0.5mm 以上のバーコード、かつ距離が 1m 以上離れている場合に特に有効です。
※9 自動設定では反射光量で距離を判断しているため、想定とは異なるレーザ幅になる可能性もあります。事前に動作をご確認ください。

イベントプロパティ

名称	発生タイミング	sender
onScanComplete	バーコード読み取りが成功したとき	nil
onScanTimeout	読み取りしないでタイムアウトになったとき	nil
onScanError	バーコード読み取りがエラーになったとき	nil

メソッド

名称	説明	引数	戻り値
Initialize	各プロパティを初期化します。	なし	なし
GetPict	直近で撮影した画面をファイル(ビットマップ形式)に保存します。BT-1500シリーズでのみ有効です。	filename:保存ファイル名 (ドライブ番号:ファイル名)	true:正常終了 false:エラー終了
GetData※1	多段読みしたときに、ｎ番目に読み取れたデータを取得します。カメラタイプで、scanType2Dが"full"(多段読み)のときのみ有効です。	n:データ番号 (0～3:QRコード以外) (0～15:QRコード)	true:正常終了 false:エラー終了
GetQRData	ｎ番目に読み取れたデータを取得します。(連結QRコード用)カメラタイプのみ有効です。	n:データ番号(0～15)	x,y,p x: 連結コード番号 (0～15) y: 連結データ総数 (1～16) p: パリティ (0～255)※2
SetFullDetection	多段読みをするコードのデータ形式を設定します。カメラタイプで、scanType2Dが"full"(多段読み)のときのみ有効です。	index:多段設定番号(0～3) code: "JAN"、"CODE39"、 "EAN128"、 "CODE128"、"ITF"、"NW7"、"CODE93"、 "RSS"、"QR" column-param:桁数の最小、最大値設定※3	なし
Input	バーコード読み取りを実行します。	なし	0:正常終了 1:読み取り中 −1:読み取りエラー −2:レーザオフ −4:バーコードデータ不正 −5:バーコード入力失敗
SetAction	バーコード読み取り時の各デバイス動作を設定します。	device: "led":LEDの動作設定 "buzzer":ブザーの動作設定 "vibration":バイブレーションの動作設定 case: "ok":読み取り成功時の動作設定 "timeout":読み取りタイムアウト時の動作設定 "error":読み取りエラー時の動作設定 nil:設定解除	true:正常終了 false:エラー終了
GetHeader	ヘッダ文字列を取得します。	なし	ヘッダ文字列 false:エラー終了
SetHeader	ヘッダ文字列を設定します。	len:有効文字数 string:設定する文字列	true:正常終了 false:エラー終了

名称	説明	引数	戻り値
GetTerminator	ターミネータ文字列を取得します。	なし	ターミネータ文字列 false:エラー終了
SetTerminator	ターミネータ文字列を設定します。	len:有効文字数 string:設定する文字列	true:正常終了 false:エラー終了
GetReplace	制御コードの変換ルールを取得します。	srcdata:置換え対象文字 (0x00～0x1F, 0x7F)	置換え文字 false:エラー終了
SetReplace	制御コードの変換ルールを設定します。	srcdata:置換え対象文字 (0x00～0x1F, 0x7F) dstdata: 置換え文字	true:正常終了 false:エラー終了

※1 「多段読み」の場合は、GetDataメソッドを実行した時点で、「data」「quality」の値が更新されます。
※2 連結 QR コードを読み取る場合、連結の順番、連結コードの総数、パリティが返されます。同一の連結コード内であれば、パリティは同じ値になります。
※3 column-param のフォーマットは、最小桁と最大桁をカンマで区切ったグループを、多段読みしたい個数(2～4)だけ、"｜"で区切ります。
"min1,max1｜min2,max2｜min3,max3｜min4,max4"

上記以外のフォーマットでは実行時例外となります。
設定可能なコードは、以下のとおりです。
JAN、CODE39、GS1-128(EAN128)、CODE128、ITF、NW7、CODE93、GS1 DataBar(RSS)
例1 「1段目:10～14桁、2段目:15～16桁、3段目:8桁固定」の場合
"10,14｜15,16｜8,8"

■ バーコード読み取り制御(「trigger」プロパティ、「Input」メソッドなど)

「Input」メソッドを使用して、バーコード読み取りを制御する場合には、以下のようにプログラミングしてください。

ポイント **通常は、InputStringコントロールを使用してください。**

① BCR:Initializeで、初期化します。
② BCR:triggerで、トリガON(ターゲットポインタ/レーザ点光)します。
③ BCR:Inputで、バーコードを読み取ります。
④ 手順③の戻り値を確認します。
「0:正常終了」の場合 … ⑤に進みます。
「1:読み取り中」の場合 … ③に戻ります。
「－1:読み取りエラー」の場合… ⑤に進みます。
⑤ BCR:triggerで、トリガOFF(ターゲットポインタ/レーザ消灯)します。
⑥ BCR:dataで、読み取りデータを取得します。

使用例

例1　「Input」メソッドを使用したバーコード読み取り時のサンプル

```
Method main()
	key=""
	ret=0
Begin
	Screen:fontSize = "middle"
	// 画面表示をクリアし、メッセージを画面出力する
	Screen:Clear()
	Screen:posx = 1
	Screen:posy = 1
	Screen:OutputText("ﾊﾞｰｺｰﾄﾞ入力ｻﾝﾌﾟﾙ2")
	Screen:posy = 15
	Screen:OutputText("ﾄﾘｶﾞｷｰでｽｷｬﾝ開始")
	Screen:posy = 17
	Screen:OutputText("Cｷｰで終了")

	// 各プロパティを初期化
	BCR:Initialize()

	// TRGキーが押されたらスキャン開始、Cキーが押されたら終了
	While(1)
		key = Handy:KeyWait()
		If key eq "TRG" Then
			Wbreak
		Elself ( key eq "C" ) Or ( key eq "BS" ) Then
			Return()
		End If
	Wend

	// レーザーを発光させ,スキャンを開始させる
	BCR:trigger = True

	// 読み取りが終わったらループを抜ける
	While(1)
		ret = BCR:Input()
		If ret <> 1 Then
			Wbreak
		End If
	Wend

	// レーザーを消灯させ,スキャンを停止させる
	BCR:trigger = False

	// 戻り値が0の場合は正常終了なので、読み取ったデータをメッセージボックス表示
	If ret == 0 Then
		Handy:ShowMessageBox( BCR:data & "を読み取りました","confirm" )

		// 0でない場合は読み取りエラーなので、エラーメッセージをメッセージボックス表示
	Else
		Handy:ShowMessageBox("読み取りに失敗しました(" & ret & ")","confirm")
	End If

End Method
```

例2　読み取り成功時、タイムアウト時、エラー時のLEDの動作設定をおこなっています。
ブザー、バイブレーションの使用方法についても同様です。

```
Method SetDeviceAction()
    Begin
        With Led
            :onTime = 10                        // 点灯時間10ミリ秒
            :offTime = 0                        // 消灯時間0ミリ秒
            :loopCount = 1                      // 点灯1回
            :color = "green"                    // 点灯色は緑
            BCR:SetAction("led", "ok")

            :onTime = 2                         // 点灯時間2ミリ秒
            :offTime = 2                        // 消灯時間2ミリ秒
            :loopCount = 3                      // 繰り返し回数3回
            :color = "orange"                   // 点灯色は赤
            BCR:SetAction("led","timeout")

            :onTime = 30                        // 点灯時間30ミリ秒
            :offTime = 0                        // 消灯時間0ミリ秒
            :loopCount = 1                      // 点灯1回
            :color = "red"                      // 点灯色は赤
            BCR:SetAction("led","error")
        End With
End Method
```

OCR	OCR文字列を読み取るときの動作設定をおこないます。

OCR文字列の読み取りに関する各種設定をおこないます。

ポイント

このコントロールはBT-W85TとOCRアクティベート済みのBT-W350/W370/W250でのみ有効です。
一部プロパティでは実行時例外が発生します。
isAvailableなどでOCR対応機器か判別した上で、使用してください。

プロパティ

名称	説明	範囲	初期値	代入
enableType	有効な読取組合せを設定します。	1～3 1:OCR + 1D/2D 2:OCRのみ 3:1D/2Dのみ	1	可
pattern	OCRの認識パターンを設定します。	"string":文字列設定 "date":日付設定	"string"	可
adDigit	西暦桁の読取設定を行います。	"both":2桁または4桁 "2digit":2桁のみ "4digit":4桁のみ	"both"	可
dayExist	日にち表記の読取設定を行います。	"both":日にち有り/無し "exist":日にち有りのみ "notExist":日にち無しのみ	"both"	可
oneDigitMD	1桁の月･日を読取るかどうかを設定します。	true:読み取る false:読み取らない	true	可
outDateFormat	読み取りした日付の出力形式を設定します。 認識パターンが日付設定の場合のみ有効です。	"YYYYMMDD,X" "YYYYMM,X" "YYMMDD,X" "YYMM,X" Xには年月日間のセパレータを指定します※1。	"YYYYMMDD,"	可
enableDispOnAlert	アラート発生時の読取確認画面の表示/非表示を設定します。	true:表示する false:表示しない	true	可
enableBuzzerOnAlert	アラート発生時のブザー動作の有効/無効を設定します。	true:有効にする false:無効にする	true	可
enableVibrationOnAlert	アラート発生時のバイブレーション動作の有効/無効を設定します。	true:有効にする false:無効にする	true	可
enableLedOnAlert	アラート発生時のLED動作の有効/無効を設定します。	true:有効にする false:無効にする	true	可
isOCRAvailable	BT-W350/W370/W250において、OCR機能が有効であるか無効であるかを取得します。	true: OCR利用可能 (BT-W85T、およびOCRアクティベート済み端末) false: OCR利用不可	–	不可

※1 セパレータには次の4記号が使用できます。
スラッシュ　'/'
ピリオド　'.'
ハイフン　'-'
スペース　' '

セパレータを使用しない場合はXは省略できますが、',' は必ず必要です。
例:　"YYMMDD,-" → 15-10-10
　　"YYYYMM," → 201510

メソッド

名称	説明	引数	戻り値
Initialize	各プロパティを初期化します。	なし	なし
SetActionOnAlert	アラート発生時の各デバイス動作を設定します。	device: "buzzer":ブザーの動作設定 ※他のデバイスは未対応 ※設定方法はBCR:SetActionを参照してください。	true:正常終了 false:エラー終了
GetActionOnAlert	アラート発生時の各デバイス動作を取得します。	device: "buzzer":ブザーの動作設定 ※他のデバイスは未対応 ※実行後対象デバイスのコントロールに現在の設定が反映されます。	true:正常終了 false:エラー終了
SetFormatInfo	フォーマット登録情報を設定します。	index:登録番号 1～50 enable:フォーマット有効/無効 true/false length:フォーマット桁数 1～30 format:フォーマット指定文字列 書式は下記参照※1	true:正常終了 false:エラー終了
GetFormatInfo	フォーマット登録情報を取得します。	index:登録番号 1～50	登録情報文字列※2 false:エラー終了
SaveConfig	現在のOCR読取設定をOCR読取設定ファイルとして出力します。	dstfile:ファイル名 overwrite:上書き許可 true/false	true:正常終了 false:エラー終了
LoadConfig	OCR読取設定ファイルを読み込み設定に反映します。	srcfile:ファイル名	true:正常終了 false:エラー終了

※1 フォーマット指定文字列 書式
<d1>^<d2>^<d3>^…^<d30> （<dn>:文字桁単位での有効文字指定）

例)1文字目=英字(文字種)、2桁目=12AB、3桁目=数字(文字種)の場合
フォーマット指定文字列: "$^12AB^!"

※文字種の省略表記については、以下の通りです。
英数字 '&'
英字 '$'
数字 '!'
記号 '@' (スペースは含まれません)
すべて '?'

※2 登録情報文字列 書式
X|Y|フォーマット指定文字列

X:フォーマット有効/無効(1/0)
Y:フォーマット桁数

例) 1文字目=英字(文字種)、2桁目=12AB、3桁目=数字(文字種)の有効フォーマット
登録情報文字列:"1|3|$^12AB^!"

使用例

例1　OCR読取設定を行い、読取結果をテキストボックスに表示する

```
Method main()
Begin
    SetOCRSettings()

    With TextBox< "入力" > // TextBoxのタグ名に「入力」を設定
        :Create( "ROOT_WINDOW" ) /* ルートウィンドウ上にTextBoxを作成 */
        :top = 70 /* ルートウィンドウ上にTextBoxを表示する最上Y座標＝70 */
        :left = 10 /* ルートウィンドウ上にTextBoxを表示する最左X座標＝10 */
        :width = 200 // 幅は200ドット
        :height = 40 // 高さは40ドット
        :enableBCR = 1 /* バーコード読み取り時のモードはオートオフ */
        :visible = true
        :enable = true
    End With

    With Window< "ROOT_WINDOW" >
        :enable = true
        :Update() // ルートウィンドウの表示更新
    End With

    Event:Wait() // イベント取得
End Method

Method SetOCRSettings()
    i
    format
    dat[3]
Begin
    //OCR対象機種以外は終了
    If Handy:model ne "BT-W85T" Then Return() EndIf
    //OCR設定ファイルがある場合は反映して終了
    If OCR:LoadConfig("1:OcrSample") Then Return() EndIf

    //OCRのみ読み取り
    OCR:enableType = 2
    //日付読み取り
    OCR:pattern = "string"

    //アラート時のブザー動作変更
    With Buzzer
        :onTime = 5
        :offTime = 5
```

```
        :loopCount = 3
        OCR:SetActionOnAlert("buzzer")
    End With

    //全てのOCRフォーマットを無効に設定
    For i = 1 to 50
        format = OCR:GetFormatInfo(i)
        dat = format.split("¦")
        OCR:SetFormatInfo(i, false, dat[1], dat[2])
    Next

    //1文字目=英字(文字種)、2桁目=12AB、3桁目=数字(文字種)のフォーマットを登録
    OCR:SetFormatInfo(1, true, 3, "$^12AB^!")

    //OCR設定を設定ファイルに保存
    OCR:SaveConfig("1:OcrSample", true)
End Method
```

7-7 デバイスの動作設定

デバイスの動作(LED、バイブレータ、ブザー)に関する設定をおこないます。バーコード読み取り時、OK/NGの確認などのために使用します。

Led	プロパティの設定値にしたがって、LEDを点灯させます。

プロパティ

名称	説明	範囲	初期値	代入
onTime※	点灯時間 LEDを点灯させる時間を設定します。	1～50 (単位:×100ミリ秒)	1	可
offTime※	消灯時間 LEDを消灯させる時間を設定します。	0～50 (単位:×100ミリ秒)	0	可
loopCount※	点灯・消灯の繰り返し回数 LEDの点灯と消灯を何回繰り返すかを設定します。	1～100(回)	1	可
synchronization※	点灯・消灯の完了までの同期/非同期を設定します。「6-6 デバイス」(6-39ページ)	true:同期 false:非同期	true	可
color	点灯色 LEDをどの色で点灯させるかを設定します。	nil:点灯しない "green":緑 "red":赤 "orange":オレンジ	nil	可

※ デバイス動作コントロールの共通プロパティです。他のコントロールでも使用します。

メソッド

名称	説明	引数	戻り値
Initialize	各プロパティを初期化します。	なし	なし
Exec	処理を開始します。	なし	なし
Stop	処理を停止します。	なし	なし

使用例

緑色で1秒間点灯し0.5秒間消灯、を5回繰り返します。

```
With Led
    :onTime = 10:offTime = 5:loopCount = 5
    :synchronization = true
    :color = "green"
    :Exec()
EndWith
```

Vibration	プロパティの設定値にしたがって、バイブレーションを動作させます。

プロパティ

名称	説明	範囲	初期値	代入
onTime※	振動時間 バイブレーションを振動させる時間を設定します。	1～50(単位:×100ミリ秒)	1	可
offTime※	振動停止時間 バイブレーションの振動を停止させる時間を設定します。	0～50(単位:×100ミリ秒)	0	可
loopCount※	振動･振動停止の繰り返し回数 バイブレーションの振動と振動停止を何回繰り返すかを設定します。	1～100(回)	1	可
synchronization※	振動･振動停止の完了までの同期/非同期を設定します。	true:同期 false:非同期	true	可

※ デバイス動作コントロールの共通プロパティです。他のコントロールでも使用します。

メソッド

名称	説明	引数	戻り値
Initialize	各プロパティを初期化します。	なし	なし
Exec	処理を開始します。	なし	なし
Stop	処理を停止します。	なし	なし

使用例

1秒間振動し1秒間停止、を3回繰り返します。

```
With Vibration
    :onTime = 10:offTime = 10:loopCount = 3
    :synchronization = true
    :Exec()
EndWith
```

Buzzer	プロパティの設定値にしたがって、ブザーを鳴らします。

プロパティ

名称	説明	範囲	初期値	代入
onTime※1	鳴動時間 ブザーを鳴らす時間を設定します。	1～50 (単位:×100ミリ秒)	1	可
offTime※1	鳴動停止時間 ブザーを鳴らさない時間を設定します。	0～50 (単位:×100ミリ秒)	0	可
loopCount※1	鳴動･鳴動停止の繰り返し回数 ブザーの鳴動と鳴動停止を何回繰り返すかを設定します。	1～100(回)	1	可
pitch	音高 ブザーの音の高さを設定します。	1～16(低～高)	1	可
volume※2	音量 ブザーの音量を設定します。	1～3(小～大) nil:消音	3	可
synchronization※1	鳴動･鳴動停止の完了までの同期/非同期を設定します。	true:同期 false:非同期	true	可

※1 デバイス動作コントロールの共通プロパティです。他のコントロールでも使用します。

※2 シミュレータでは音量が変化しない場合があります。

メソッド

名称	説明	引数	戻り値
Initialize	各プロパティを初期化します。	なし	なし
Exec	処理を開始します。	なし	なし
Stop	処理を停止します。	なし	なし

使用例

0.5秒間ブザーを鳴らし0.5秒間停止、を3回繰り返します。

```
With Buzzer
    :onTime = 5:offTime = 5:loopCount = 3
    :synchronization = true
    :Exec()
EndWith
```

7-8 ファイルの操作

ファイルの読み書き、ファイル情報の参照をおこないます。

FileStream	BTシリーズ内(ドライブ1、2)にある任意のファイルを読み書きします。 タグ指定コントロールです。

ポイント

- **File コントロールとの相違点は、複数ファイルを同時に最大 8 つまで取り扱えることです。**
- **タグ名には、ファイル名を次の形式で指定します。**
  BT-W100/W80/W70/1000/1500シリーズ･･･ドライブ番号:パス名¥ファイル名(最大255バイト以内)
  BT-600シリーズ･･･ドライブ番号:ファイル名(最大255バイト以内)

サーバ上のファイルの読み書きについては、「RemoteFile」(7-229ページ)を参照してください。

プロパティ

名称	説明	範囲	初期値	代入
currentPos	ファイルの中の位置(シークポインタ)を設定します。 「シークポインタ(currentPosプロパティ)」(7-162ページ)	-1、0～(バイト)※1	0	可
fileCount※2	現在オープンしているファイル総数	0～8	0	不可
error	エラー情報 エラー原因が格納されます。	文字列データ 「エラー情報(errorプロパティ)」(7-164ページ)	""	不可

※1 ファイルの終端位置を指定するときは、-1を代入します。
-1を代入後、currentPosの値を参照すると、ファイルの終端位置を取得できます。

※2 どのタグ名のFileStreamコントロールから参照しても、同じ値を取得できます。

メソッド

名称	説明	引数	戻り値
Open	ファイルをオープンします。	mode:オープンモード（“r”/“w”/“a”）※1	true:正常終了 false:エラー終了
Gets	ファイルからテキストデータを読み込みます。 📖「テキストデータの読み込み、書き込み(Gets、Putsメソッド)」(7-162ページ)	maxLength:最大読み込みバイト数(1～8192)	読み込み文字列※2 false:読み込みエラー
Puts	ファイルにテキストデータを書き込みます。 📖「テキストデータの読み込み、書き込み(Gets、Putsメソッド)」(7-162ページ)	string:書き込み文字列(最大8192バイト)	書き込んだ文字列のバイト数 false:エラー終了
Read	ファイルからバイナリデータを読み込みます。 📖「バイナリデータの読み込み、書き込み(Read、Writeメソッド)」(7-163ページ)	maxLength:最大読み込みバイト数(1～8192)	読み込み文字列※3 false:読み込みエラー
Write	ファイルにバイナリデータを書き込みます。 📖「バイナリデータの読み込み、書き込み(Read、Writeメソッド)」(7-163ページ)	string:書き込み文字列(最大8192バイト)	書き込んだ文字列のバイト数※3 false:エラー終了
Close	ファイルをクローズします。	なし	なし

※1 “r”:読み込みモード
すでに作成されているファイルを読み込みモードでオープンします。
このモードでファイルをオープンして、「Puts」、または「Write」メソッドを実行するとエラーになります。
“w”:読み書きモード
ファイルを新規作成して、データを読み書きします。
同名のファイルが存在するときは、上書きされます。
“a”:追加モード
すでに作成されているファイルの最語尾に追加書き込みできます。

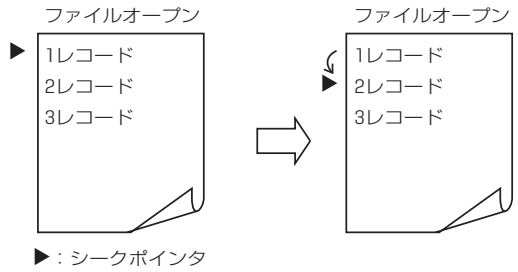
※2 ファイル終端でGetsメソッドを実行すると、“”(空文字)が返ります。

※3 File Stream : Read/WriteとFile : Read/Writeではバイナリデータの扱いに差異があります。

■シークポインタ(currentPosプロパティ)

ファイルを読み書きするとき、読み書きを開始する位置をシークポインタといいます。初めてファイルをオープンしたときは先頭にあります。

例えば、ファイルをオープンし最初の 1 レコード読み込むと、シークポインタは 2 レコード目の先頭に移動します。続けてレコードを読み込むと、2レコード目が読み込まれます。



シークポインタの位置を指定したいときは、ファイルを読み書きする前に「currentPos」プロパティでシークポインタの設定をする必要があります。

■テキストデータの読み込み、書き込み(Gets、Putsメソッド)

「Gets」、「Puts」はASCII文字や漢字を扱う場合に適しています。

ポイント

- **改行、TABなど基本的な制御コードを含むことは問題ありません。**
- **NULL(0x00)を含むデータは扱えません。**

●読み込み(「Gets」メソッド)

「Gets」メソッドを使用して、読み込みモードでオープンしたファイルを、シークポインタの位置から以下のいずれかの条件を満たすまで文字列を読み込みます。

- **読み込みデータにNULL(0x00)があらわれた。**
- **読み込みデータに改行コードがあらわれた。**
- **ファイルの終端に達した。**
- **読み込みデータの長さが(最大読み込みバイト数−1)に達した。**

「Gets」メソッドを実行すると、読み込みデータのバイト数分シークポインタが進みます。
(「currentPos」プロパティに設定されます。)

●書き込み(「Puts」メソッド)

「Puts」メソッドを使用して、読み書きモードまたは追加モードでオープンしたファイルにデータを書き込みます。

「Puts」メソッドを実行すると、ファイルに書き込んだバイト数分シークポインタが進みます。
(「currentPos」プロパティに設定されます。)

■バイナリデータの読み込み、書き込み(Read、Writeメソッド)

「Read」、「Write」メソッドは、NULL(0x00)を含むバイナリデータを扱う場合に適しています。
バイナリ表記については、「バイナリ表記について」(2-6ページ)を参照してください。

●読み込み(「Read」メソッド)

「Read」メソッドを使用して、読み込みモードでオープンしたファイルを、シークポインタの位置から以下のいずれかの条件を満たすまで文字列を読み込みます。

- **ファイルの終端に達した。**
- **読み込みデータの長さが最大読み込みバイト数に達した。**

「Read」メソッドを実行すると、読み込みデータのバイト数分シークポインタが進みます。
(「currentPos」プロパティに設定されます。)

●書き込み(「Write」メソッド)

「Write」メソッドを使用して、読み書きモードまたは追加モードでオープンしたファイルへバイナリデータを書き込みます。

- 書き込む前にドライブの空き容量をチェックして、空きがない場合は書き込みをおこなわずに、戻り値にエラーを返します。
- 「Write」メソッドを実行すると、ファイルに書き込んだバイト数分シークポインタが進みます。(「currentPos」プロパティに設定されます。)

■ドライブへのアクセス

各メソッドの、ドライブに対するアクセス状況は次のようになります。

●BT-W100/W80/W70/1000/1500シリーズ

※BT-W70シリーズは、microSDカードスロットがないため、ドライブ5に対応していません。

メソッド	ドライブ0	ドライブ1	ドライブ2	ドライブ3	ドライブ5
Gets	×	○	○	×	○
Puts	×	○	○	×	○
Read	×	○	○	×	○
Write	×	○	○	×	○

●BT-600シリーズ

メソッド	ドライブ0	ドライブ1	ドライブ2	ドライブ3	ドライブ4
Gets	×	○	○	×	×
Puts	×	○	○	×	×
Read	×	○	○	×	×
Write	×	○	○	×	×

ポイント

BT-600シリーズのドライブ4は、BT-500シリーズとの互換性を保つために用意されています。
そのため、新規開発では、使用しないことをお勧めいたします。

■エラー動作

次のような場合には、メソッドを実行するとエラーとなります。

- メソッド実行中に、ドライブ容量不足などのためにファイル書き込みができないときは、戻り値に「false」が返ります。
- 「currentPos」プロパティの値が負またはファイルサイズを超えているときにメソッドを実行すると、「Gets」、「Puts」、「Read」、「Write」の戻り値に「false」が返ります。

■エラー情報（errorプロパティ）

エラー情報	説明
"ERR_FILE_DRIVENAME"	ドライブ名エラー
"ERR_FILE_FILENAME"	ファイル名エラー
"ERR_FILE_NONAME"	ファイル名を指定していない
"ERR_FILE_MODE"	該当モードではオープンできない
"ERR_FILE_NOT_EXIST"	ファイルは存在しない
"ERR_FILE_ALREADY_EXIST"	ファイルは既に存在している
"ERR_FILE_ALREADY_OPEN"	ファイルは既にオープンしている
"ERR_FILE_NOT_OPEN"	ファイルをオープンしていない
"ERR_FILE_FORMAT"	ファイルフォーマットエラー
"ERR_FILE_ENTRY_OVER"	ファイルエントリー数オーバー
"ERR_FILE_NOSPACE"	空き容量不足
"ERR_FILE_MAXOPEN"	ファイルオープン数オーバー
"ERR_FILE_SIZE"	ファイルサイズを超えるアドレスを指定した
"ERR_FILE_NOT_FOUND"	該当ファイルが見つからない
"ERR_FILEREC_OVERFLOW"	ファイルレコードが最大数を超えた
"ERR_FILE_DIRECTORYNAME"	ディレクトリ名エラー
"ERR_FILE_DRIVEACCESS"	ドライブアクセスエラー

使用例

例1　File1.txtとFile2.txtを同時にオープンして、File1のデータを読み込んでFile2に書き込みます。

```
Method main()
        str
Begin

        FileStream<"2:File1.txt">:Open("r")
        FileStream<"2:File2.txt">:Open("w")

        str = FileStream<"2:File1.txt">:Read(16)
        FileStream<"2:File2.txt">:Write(str)

        FileStream<"2:File1.txt">:Close()
        FileStream<"2:File2.txt">:Close()

End Method
```

例2　Write、Readメソッドを使用して、バイナリデータの読み書きをおこないます。

```
Method main()
        dat
Begin
        FileStream<"2:SAMPLE.BIN">:Open("W")

        //ファイルを新規に作成し{0x01,0x23,0x45,0x67,0x89}を書き込みます
        FileStream<"2:SAMPLE.BIN">:Write("¥x01¥x23¥x45¥x67¥x89")

        //シークポインタを3バイト目に移動して0xABを書き込みます
        FileStream<"2:SAMPLE.BIN">:currentPos = 3
        FileStream<"2:SAMPLE.BIN">:Write("AB")

        //ファイルの終端に0xEFと書き込みます
        FileStream<"2:SAMPLE.BIN">:Puts("EF")

        // ファイルの先頭から5バイト読み込みます
        FileStream<"2:SAMPLE.BIN">:currentPos = 0
        dat = FileStream<"2:SAMPLE.BIN">:Read(5)

        //dat ={ 0x01, 0x23, 0x45, 0x67, 0x89}
        // ファイルの4バイト目から2バイト読み込みます
        FileStream<"2:SAMPLE.BIN">:currentPos = 4
        dat = FileStream<"2:SAMPLE.BIN">:Read(2) //dat ={ 0x89, EOF}

End Method
```

File	BTシリーズ内(ドライブ1、2)にある任意のファイルを読み書きします。

サーバ上のファイルの読み書きについては、「RemoteFile」(7-229ページ)を参照してください。

プロパティ

名称	説明	範囲	初期値	代入
name	読み書きするファイル名を指定します。	ドライブ番号:ファイル名※	nil	可
currentPos	ファイルの中の位置(シークポインタ)を設定します。「シークポインタ(currentPosプロパティ)」(7-167ページ)	0～(バイト)	0	可
error	エラー情報 エラー原因が格納されます。	文字列データ 「エラー情報(errorプロパティ)」(7-169ページ)	""	不可

※ ドライブを指定しない場合はドライブ2になります。ファイル名はパス付きで最大255バイトまでで指定できます。指定できるファイルは1つのみです。

メソッド

名称	説明	引数	戻り値
Initialize	各プロパティを初期化します。	なし	なし
Gets	ファイルからテキストデータを読み込みます。 「テキストデータの読み込み、書き込み(Gets、Putsメソッド)」(7-167ページ)	maxLength:最大読み込みバイト数 (1～8192)	読み込み文字列※2 false:読み込みエラー
Puts	ファイルにテキストデータを書き込みます。 「テキストデータの読み込み、書き込み(Gets、Putsメソッド)」(7-167ページ)	string:書き込み文字列 (最大8192バイト) mode:書き込みモード ("write"/"append"/"create")※1	true:正常終了 false:エラー終了
Read	ファイルからバイナリデータを読み込みます。 「バイナリデータの読み込み、書き込み(Read、Writeメソッド)」(7-168ページ)	maxLength:最大読み込みバイト数 (1～4096)	読み込み文字列※3 「バイナリ表現の特殊形式」(7-168ページ)を参照してください。 false:読み込みエラー
Write	ファイルにバイナリデータを書き込みます。 「バイナリデータの読み込み、書き込み(Read、Writeメソッド)」(7-168ページ)	string:書き込み文字列 (バイナリ表現の特殊形式、最大4096バイト) mode:書き込みモード ("write"/"append"/"create")※1	true:正常終了※3 false:エラー終了
Close	ファイルをクローズします。	なし	なし

※1 "write": 「currentPos」プロパティで指定する位置から上書きします。
"append": 「currentPos」のプロパティ値にかかわらずシークポインタをファイル終端に移動して、その位置から追加書き込みをします。
"create": 既にあるファイルを削除して、「currentPos」のプロパティ値をファイル先頭位置に設定します。

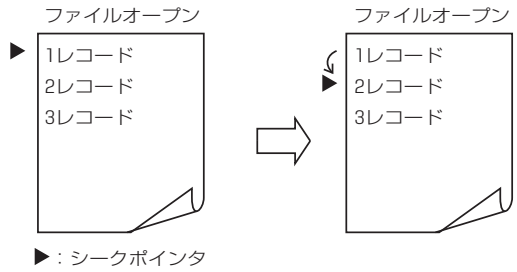
※2 ファイル終端でGetsメソッドを実行すると、falseが返ります。

※3 File Stream : Read/WriteとFile : Read/Writeでは、バイナリデータの扱いに差異があります。

■シークポインタ(currentPosプロパティ)

ファイルを読み書きするとき、読み書きを開始する位置をシークポインタといいます。初めてファイルをオープンしたときは先頭にあります。

例えば、ファイルをオープンし最初の 1 レコード読み込むと、シークポインタは 2 レコード目の先頭に移動します。続けてレコードを読み込むと、2レコード目が読み込まれます。



シークポインタの位置を指定したいときは、ファイルを読み書きする前に「currentPos」プロパティでシークポインタの設定をする必要があります。

■テキストデータの読み込み、書き込み(Gets、Putsメソッド)

「Gets」、「Puts」はASCII文字や漢字を扱う場合に適しています。

ポイント

- **改行、TABなど基本的な制御コードを含むことは問題ありません。**
- **NULL(0x00)を含むデータは扱えません。**

●読み込み(「Gets」メソッド)

「Gets」メソッドを使用して、「name」プロパティで指定するファイルを読み込みモードでオープンして、シークポインタの位置から以下のいずれかの条件を満たすまで文字列を読み込みます。

- **読み込みデータにNULL(0x00)があらわれた。**
- **読み込みデータに改行コードがあらわれた。**
- **ファイルの終端に達した。**
- **読み込みデータの長さが(最大読み込みバイト数−1)に達した。**

「Gets」メソッドを実行すると、読み込みデータのバイト数分シークポインタが進みます。
(「currentPos」プロパティに設定されます。)

●書き込み(「Puts」メソッド)

「Puts」メソッドを使用して、「name」プロパティで指定するファイルにデータを書き込みます。

「Puts」メソッドを実行すると、ファイルに書き込んだバイト数分シークポインタが進みます。
(「currentPos」プロパティに設定されます。)

■バイナリデータの読み込み、書き込み(Read、Writeメソッド)

「Read」、「Write」メソッドは、NULL(0x00)を含むバイナリデータを扱う場合に適しています。
バイナリ表記¥x**は、「Write」メソッドでは、使用できません。

●バイナリ表現の特殊形式

0x12、0x34、0x56、0x78のような16進表現のデータを、「Read」、「Write」メソッドで使用するときは、"12345678"と記述します。

▶ 重 要

- **文字数が奇数の場合は、最後の1文字は無視されます。**
- **変換できない文字列の場合は、実行時例外が発生します。**

●読み込み(「Read」メソッド)

「Read」メソッドを使用して、「name」プロパティで指定するファイルを読み込みモードでオープンして、シークポインタの位置から以下のいずれかの条件を満たすまで文字列を読み込みます。

- **ファイルの終端に達した。**
- **読み込みデータの長さが最大読み込みバイト数に達した。**

「Read」メソッドを実行すると、読み込みデータのバイト数分シークポインタが進みます。
(「currentPos」プロパティに設定されます。)

●書き込み(「Write」メソッド)

「Write」メソッドを使用して、「name」プロパティで指定するファイルへバイナリデータを書き込みます。

- 書き込む前にドライブの空き容量をチェックして、空きがない場合は書き込みをおこなわずに、戻り値にエラーを返します。
- 「Write」メソッドを実行すると、ファイルに書き込んだバイト数分シークポインタが進みます。(「currentPos」プロパティに設定されます。)

■ドライブへのアクセス

各メソッドの、ドライブに対するアクセス状況は次のようになります。

●BT-W100/W80/W70/1000/1500シリーズ

※BT-W70シリーズは、microSDカードスロットがないため、ドライブ5に対応していません。

メソッド	ドライブ0	ドライブ1	ドライブ2	ドライブ3	ドライブ5
Gets	×	○	○	×	○
Puts	×	○	○	×	○
Read	×	○	○	×	○
Write	×	○	○	×	○

●BT-600シリーズ

メソッド	ドライブ0	ドライブ1	ドライブ2	ドライブ3	ドライブ4
Gets	×	○	○	×	×
Puts	×	○	○	×	×
Read	×	○	○	×	×
Write	×	○	○	×	×

ポイント

BT-600シリーズのドライブ4は、BT-500シリーズとの互換性を保つために用意されています。
そのため、新規開発では、使用しないことをお勧めいたします。

■エラー動作

次のような場合には、メソッドを実行するとエラーとなります。

- メソッド実行中に、ドライブ容量不足などのためにファイル書き込みができないときは、戻り値に「false」が返ります。
- 「currentPos」プロパティの値が負またはファイルサイズを超えているときにメソッドを実行すると、「Gets」、「Puts」、「Read」、「Write」の戻り値に「false」が返ります。

■エラー情報(errorプロパティ)

エラー情報	説明
"ERR_FILE_DRIVENAME"	ドライブ名エラー
"ERR_FILE_FILENAME"	ファイル名エラー
"ERR_FILE_NONAME"	ファイル名を指定していない
"ERR_FILE_MODE"	該当モードではオープンできない
"ERR_FILE_NOT_EXIST"	ファイルは存在しない
"ERR_FILE_ALREADY_EXIST"	ファイルは既に存在している
"ERR_FILE_ALREADY_OPEN"	ファイルは既にオープンしている
"ERR_FILE_NOT_OPEN"	ファイルをオープンしていない
"ERR_FILE_FORMAT"	ファイルフォーマットエラー
"ERR_FILE_ENTRY_OVER"	ファイルエントリー数オーバー
"ERR_FILE_NOSPACE"	空き容量不足
"ERR_FILE_MAXOPEN"	ファイルオープン数オーバー
"ERR_FILE_SIZE"	ファイルサイズを超えるアドレスを指定した
"ERR_FILE_NOT_FOUND"	該当ファイルが見つからない
"ERR_FILEREC_OVERFLOW"	ファイルレコードが最大数を超えた
"ERR_FILE_DIRECTORYNAME"	ディレクトリ名エラー
"ERR_FILE_DRIVEACCESS"	ドライブアクセスエラー

使用例

例1　「Gets」、「Puts」メソッドを使ってファイルを読み書きします。

```
Method main()
        dat
Begin

        File:name = "2:SAMPLE.TXT"

        //ファイルを新規に作成し"01234567890"と書き込みます
        File:Puts("01234567890", "create")

        //シークポインタを3バイト目に移動して"ABC"と書き込みます
        File:currentPos = 3
        File:Puts("ABC","write")

        //ファイルの終端に"XYZ"と書き込みます
        File:Puts("XYZ","append")

        // ファイルの先頭から5バイト読み込みます
        File:currentPos = 0
        dat = File:Gets(5) //dat = "012AB"

        // ファイルの8バイト目から6バイト読み込みます
        File:currentPos = 8
        dat = File:Gets(6) //dat = "890XYZ"

End Method
```

例2　Write、Readメソッドを使用して、バイナリデータの読み書きをおこないます。

```
Method main()
    dat
Begin

    File:name = "2:SAMPLE.BIN"

    //ファイルを新規に作成し{0x01,0x23,0x45,0x67,0x89}を書き込みます
    File:Write("0123456789", "create")

    //シークポインタを3バイト目に移動して0xABを書き込みます
    File:currentPos = 3
    File:Write("AB","write")

    //ファイルの終端に0xEFと書き込みます
    File:Write("EF","append")

    // ファイルの先頭から5バイト読み込みます
    File:currentPos = 0
    dat = File:Read(5) //dat = "012345AB89"

    // ファイルの4バイト目から2バイト読み込みます
    File:currentPos = 4
    dat = File:Read(2) //dat = "89EF"

End Method
```

FileSystem	BTシリーズ内のファイル詳細情報(ファイルプロパティ)の参照、コピー、結合、削除などをおこないます。

▶ 重 要 **このコントロールで操作するファイルを、既に「File」コントロールで操作している場合は、メソッドを実行する前に「File:Initialize」メソッドを実行して、「File」コントロールのプロパティを初期化してください。**

プロパティ

名称	説明	範囲	初期値	代入
isDirectory	FindFirst、FindNextで取得できるファイル名がディレクトリか、ファイルかが返されます。	true:ディレクトリ false:ファイル	なし	不可
findFileName	FindFirst、FindNextで取得できるディレクトリ名またはファイル名が返されます。	●BT-W100/W80/W70/1000/1500シリーズ ドライブ番号:ディレクトリ名、または ドライブ番号:パス付きファイル名 ●BT-600シリーズ ドライブ番号: ドライブ番号:ファイル名	""	不可
findFileSize	FindFirst、FindNextで取得できるディレクトリのサイズ、またはファイルのサイズが返されます。	0～	0	不可
findDate	FindFirst、FindNextで取得できるディレクトリまたはファイルの日付が返されます。	YYYY/MM/DD(年/月/日)	""	不可
findTime	FindFirst、FindNextで取得できるディレクトリまたはファイルの時刻が返されます。	HH:MM(時:分)	""	不可
driveTotalSize	DriveInfoで取得できるドライブの全容量が返されます。※1	0～	0	不可
driveFreeSize	DriveInfoで取得できるドライブの空き容量が返されます。※2	0～	0	不可
driveInvalidSize	DriveInfoで取得できるドライブ無効容量が返されます。	0※4	0	不可
driveFileCount	DriveInfoで取得できるドライブのファイル数が返されます。※3	0～	0	不可
fileInfoMode	ファイル情報の取得モードを設定します。	0:通常 1:高速(ファイルサイズを取得しない)※5	0	可
zipPassword	ZIP圧縮・解凍用のパスワードを指定します。	63バイトまでの文字列	""	可
error	エラー情報 エラー原因が格納されます。	文字列データ 「エラー情報(errorプロパティ)」(7-179ページ)	""	不可

※1 ドライブのサイズによって上限値、下限値は変化します。
※2 ドライブのサイズによって上限値は変化します。
※3 ドライブによって上限値が変化します。
●BT-W100/W80/1000/1500シリーズ
ドライブ 0:1、ドライブ 1、3:上限なし、ドライブ 2:128、ドライブ 5:256(ルートディレクトリ)(フォルダを作成すると、その中には512個のファイルが格納可能です。)

●BT-W70シリーズ
ドライブ0：1、ドライブ1、3：上限なし、ドライブ2：128
●BT-600シリーズ
ドライブ0：1、ドライブ1、3、4：上限なし、ドライブ2：128

※4 プロパティ値は、必ず0になります。
driveInvalidSizeプロパティは、BT-910/951互換動作時のために用意しています。

※5「fileInfoMode ＝ 1」にした場合、FindFirst、FindNextメソッドの処理速度が向上します。ただし、同メソッドにてディレクトリまたはファイルを検索した場合、findFileSizeが常に「-1」となり取得できませんのでご注意ください。

メソッド

名称	説明	引数	戻り値
Initialize	各プロパティを初期化します。	なし	なし
CreateDirectory	ディレクトリを作成します。 BT-600シリーズは対応していません。	directoryName:ディレクトリ名※2	true:正常終了 false:エラー終了
FindFirst※1	指定するディレクトリまたはファイルの検索を開始します。 「ファイル検索(FindFirst、FindNextメソッド)」(7-177ページ)	●BT-W100/W80/W70/1000/1500シリーズ fileName:検索ディレクトリ名、またはファイル名(ドライブ番号:ファイル名)※2 ●BT-600シリーズ fileName:ファイル名(ドライブ番号:ファイル名)※2	true:正常終了 false:エラー終了
FindNext※12	ディレクトリまたはファイルの検索を継続します。 「ファイル検索(FindFirst、FindNextメソッド)」(7-177ページ)	なし	true:正常終了 false:エラー終了
DriveInfo※1	指定するドライブのドライブ情報を取得します。 「ドライブ情報の取得(DriveInfoメソッド)」(7-177ページ)	driveNo:ドライブ番号	true:正常終了 false:エラー終了
Delete※3	指定するディレクトリまたはファイルを削除します。	●BT-W100/W80/W70/1000/1500シリーズ fileName:削除ディレクトリ名、またはファイル名(ドライブ番号:ディレクトリ名、またはドライブ番号:パス付きファイル名)※7 ●BT-600シリーズ fileName:削除ファイル名(ドライブ番号:ファイル名)※7	true:正常終了 false:エラー終了
Rename	「oldFileName」で指定するディレクトリまたはファイルを、「newFileName」で指定するディレクトリ名またはファイル名に変更します。	●BT-W100/W80/W70/1000/1500シリーズ oldFileName:古いディレクトリ名、またはファイル名(ドライブ番号:ディレクトリ名、またはドライブ番号:パス付きファイル名)※7 newFileName:新しいディレクトリ名、またはファイル名(ドライブ番号:ディレクトリ名、またはドライブ番号:パス付きファイル名)※4、※7 ●BT-600シリーズ oldFileName:古いファイル名(ドライブ番号:ファイル名)※7 newFileName:新しいファイル名(ドライブ番号:ファイル名)※4、※7	true:正常終了 false:エラー終了

名称	説明	引数	戻り値
Copy※5	ファイルコピー 「srcFileName」で指定するディレクトリまたはファイルを、「newFileName」で指定するディレクトリ名またはファイル名でコピーします。	●BT-W100/W80/W70/1000/1500シリーズ srcFileName:コピー元ディレクトリ名、またはファイル名(ドライブ番号:ディレクトリ名、またはドライブ番号:パス付きファイル名)※7 newFileName:新しいディレクトリ名、またはファイル名(ドライブ番号:ディレクトリ名、またはドライブ番号:パス付きファイル名)※7 overWrite: true(上書きを許可する) false(許可しない) ●BT-600シリーズ srcFileName:コピー元ファイル名(ドライブ番号:ファイル名)※7 newFileName:新しいファイル名(ドライブ番号:ファイル名)※7 overWrite: true(上書きを許可する) false(許可しない)	true:正常終了 false:エラー終了
Cat	ファイル連結 2つのファイルを連結した結果を新ファイルに書き込みます。 「ファイルの連結(Catメソッド)」(7-177ページ)	srcFileName1:コピー元ファイル名1(ドライブ番号:ファイル名)※7 srcFileName2:コピー元ファイル名2(ドライブ番号:ファイル名)※7 newFileName:新しいファイル名※6(ドライブ番号:ファイル名)	true:正常終了 false:エラー終了
Format	ドライブフォーマット ドライブをフォーマットします。	driveNo:ドライブ番号	true:正常終了 false:エラー終了
Deflag	ドライブ1、3、4をデフラグします。 ※ドライブ4はBT-600シリーズのみ。	なし	true:正常終了 false:エラー終了
Mount	microSDカードをマウントします。BT-W100/W80/1000/1500シリーズのみ有効です。	driveNo:ドライブ番号(5固定)	true:正常終了 false:エラー終了
UnMount	microSDカードをマウント解除します。BT-W100/W80/1000/1500シリーズのみ有効です。	driveNo:ドライブ番号(5固定)	true:正常終了 false:エラー終了
IsMount	microSDカードのマウント状況を確認します。 BT-W100/W80/1000/1500シリーズのみ有効です。	driveNo: 5固定	マウントされている:"ON" マウントされていない:"OFF" カードが挿入されていない:"NOT EXIST" false:エラー終了
Compress	ファイルを圧縮します。※8	srcFile:元のファイル名(ドライブ番号:ファイル名) dstFile:圧縮後のファイル名(ドライブ番号:ファイル名) format:"ZIP"固定 overWrite:上書き(true/false)	true:正常終了 false:エラー終了

名称	説明	引数	戻り値
Uncompress	ファイルを解凍します。※9	srcFile:圧縮したファイル名(ドライブ番号:ファイル名) format:"ZIP"固定 overWrite:上書き(true/false)	ディレクトリまたはファイル名 false:エラー終了
CountLine	ファイルの行数を取得します。	fileName:ファイル名(ドライブ番号:ファイル名))	行数 false:エラー終了
SearchCreate	置き換えマスタを作成します。※10	masterFile:マスタファイル(ドライブ番号:ファイル名) indexFile:インデックス情報定義ファイル(ドライブ番号:ファイル名) searchFile:置き換えマスタ(ドライブ番号:ファイル名)※11	true:正常終了 false:エラー終了

※1 シミュレーション機能では、ドライブ 1 に「.sb3」ファイルが存在しないため、実機のドライブ 1 と空き容量、ファイル数などが異なります。

※2 ドライブ番号を指定しない場合はドライブ2になります。

※3 スクリプトファイルを削除したときは、その後の動作は保障されません。

※4 変更後のファイル名が既に存在するときはファイル名の変更をおこないません。

※5 コピー元はすべてのドライブ番号を指定できます。コピー先はドライブ番号1、2が指定できます。BT-W100/W80/1000/1500シリーズのみドライブ番号5も指定できます。

※6 新しいファイルと同じ名前のファイルが既に存在するときは上書きされます。

※7 ディレクトリ名またはファイル名は、フルパスで最大255バイトまで指定できます。
〔例〕2:aaa

※8 パスワード付 ZIP ファイルを作成する場合は、zipPassword プロパティでパスワードを設定後、ZIP圧縮してください。
ドライブ1 へ圧縮する場合、ドライブ1 の状態によってはファイルの再配置が発生し、時間がかかる事が有ります。
・ srcFileにファイル名を指定する場合
指定されたファイルを圧縮します。
・ srcFileにディレクトリ名を指定する場合
指定されたディレクトリの直下にあるファイルを圧縮します。複数ファイルの圧縮が可能です。

※9 パスワード付 ZIP ファイルを解凍する場合は、zipPassword プロパティでパスワードを設定後、解凍してください。
ファイルは解凍元ZIPファイルが存在するフォルダの直下に展開されます。

※10 マスタファイルとインデックス情報定義ファイルからインデックス情報付き端末用マスタファイル(置き換えマスタ)を作成します。置き換えマスタはSearchコントロールで検索することができます。圧縮置き換えマスタの作成には対応していません。

※11 同名のファイルがすでに存在するときは、置き換えマスタの作成に失敗します。置き換えマスタを更新する場合は、前のファイルを削除するか名前変更を先におこなってください。

※12 FindFirstメソッドを実行してからFindNextメソッドを実行するまでの間に、ファイル情報を取得するドライブに対してファイル操作(書き込み / 削除など)をおこなうと、ファイルの管理情報が変化するため、次ファイル情報取得が正しくおこなわれないことがあります。

■ファイル検索(FindFirst、FindNextメソッド)

●「FindFirst」メソッド

- 指定するディレクトリ情報またはファイル情報を取得します。
- 取得した内容は「findFileName」、「findFileSize」、「findDate」、「findTime」プロパティに格納されます。
- ドライブ 3 については、「findFileSize」プロパティにはレコード件数、「findDate」プロパティには"0000/00/00"、「findTime」プロパティには"00:00"が格納されます。
- ドライブ5にmicroSDHCカードを使用している場合、Handy:driveinfoModeを"true"に設定してからドライブ情報取得をおこなってください。

●「FindNext」メソッド

- 直前の「FindFirst」、「FindNext」で指定するディレクトリ情報またはファイルのファイル情報を取得します。
- 取得した内容は「findFileName」、「findFileSize」、「findDate」、「findTime」プロパティに格納されます。
- ドライブ 3 については、「findFileSize」プロパティにはレコード件数、「findDate」プロパティには"0000/00/00"、「findTime」プロパティには"00:00"が格納されます。
- ドライブ5にmicroSDHCカードを使用している場合、Handy:driveinfoModeを"true"に設定してからドライブ情報取得をおこなってください。

▶ 重 要

「FindFirst」から「FindNext」の間に、ファイル情報を取得するファイルに対してファイル操作をおこなうと、情報が正しく取得できないことがあります。

■ドライブ情報の取得(DriveInfoメソッド)

指定するドライブ番号の情報を取得します。
取得した内容は、「driveTotalSize」、「driveInvalidSize」、「driveFileCount」プロパティに格納されます。
ドライブ5にmicroSDHCカードを使用している場合、Handy:driveinfoModeを"true"に設定してからドライブ情報取得をおこなってください。

■ファイルの連結(Catメソッド)

コピー元ファイル1とコピー元ファイル2を連結した結果を新ファイルに書き込みます。

ポイント

- **コピー元のドライブ番号には 1、2 を、新しいファイルの保存先にはドライブ番号 1、2 を指定できます。BT-W100/W80/1000/1500シリーズのみドライブ番号5も指定できます。**
- **新しいファイル名は、コピー元の2つのファイル名と異なる必要があります。**

■ドライブへのアクセス

各メソッドの、ドライブに対するアクセス状況は次のようになります。

●BT-W100/W80/W70/1000/1500シリーズ

※BT-W70シリーズは、microSDカードスロットがないため、ドライブ5に対応していません。

メソッド	ドライブ0	ドライブ1	ドライブ2	ドライブ3	ドライブ5
CreateDirectory	×	×	×	×	○
FindFirst	×	○	○	○	○
FindNext	×	○	○	○	○
DriveInfo	○	○	○	○	○
Delete	×	○	○	○	○
Rename	×	○	○	×	○
Copy※	×	○	○	×	○
Cat	×	○	○	×	○
Format	×	○	○	○	○
Deflag	×	○	×	○	×
Mount	×	×	×	×	○
UnMount	×	×	×	×	○
Compress	×	○	○	×	○
Uncompress	×	○	○	×	○
CountLine	×	○	○	×	○
IsMount	×	×	×	×	○
SearchCreate	×	○	○	×	○

※ドライブ0、3のファイルをドライブ1、2、5にコピーすることは可能です。

●BT-600シリーズ

メソッド	ドライブ0	ドライブ1	ドライブ2	ドライブ3	ドライブ4
FindFirst	×	○	○	○	○
FindNext	×	○	○	○	○
DriveInfo	○	○	○	○	○
Delete	×	○	○	○	○
Rename	×	○	○	×	×
Copy※	×	○	○	×	×
Cat	×	○	○	×	×
Format	×	○	○	○	○
Deflag	×	○	×	○	○
Compress	×	○	○	×	×
Uncompress	×	○	○	×	×
CountLine	×	○	○	×	×
SearchCreate	×	○	○	×	×

※ドライブ0、3のファイルをドライブ1、2にコピーすることは可能です。

ポイント

BT-600シリーズのドライブ4は、BT-500シリーズとの互換性を保つために用意されています。
そのため、新規開発では、使用しないことをお勧めいたします。

■エラー情報(errorプロパティ)

エラー情報	説明
"ERR_FILE_DRIVENAME"	ドライブ名エラー
"ERR_FILE_FILENAME"	ファイル名エラー
"ERR_FILE_NONAME"	ファイル名を指定していない
"ERR_FILE_MODE"	該当モードではオープンできない
"ERR_FILE_NOT_EXIST"	ファイルは存在しない
"ERR_FILE_ALREADY_EXIST"	ファイルは既に存在している
"ERR_FILE_ALREADY_OPEN"	ファイルは既にオープンしている
"ERR_FILE_NOT_OPEN"	ファイルをオープンしていない
"ERR_FILE_FORMAT"	ファイルフォーマットエラー
"ERR_FILE_ENTRY_OVER"	ファイルエントリー数オーバー
"ERR_FILE_NOSPACE"	空き容量不足
"ERR_FILE_MAXOPEN"	ファイルオープン数オーバー
"ERR_FILE_SIZE"	ファイルサイズを超えるアドレスを指定した
"ERR_FILE_NOT_FOUND"	該当ファイルが見つからない
"ERR_FILEREC_OVERFLOW"	ファイルレコードが最大数を超えた
"ERR_FILE_DIRECTORYNAME"	ディレクトリ名エラー
"ERR_FILE_DRIVEACCESS"	ドライブアクセスエラー

使用例

例1 FileSystemコントロールの各メソッドの使用例

```
Method main()
        strItem
        menupos
Begin
        Screen:fontSize = "middle"
        strItem="1:ドライブ情報取得|2:ファイルリネーム|3:ファイルコピー|4:ファイル圧縮|
                5:ファイル解凍|6:ファイル削除|7:ファイル連結|8:デフラグ"
        menupos = 1
        While( 1 )
                menupos = 1
                menupos = Handy:ShowMenu("ファイル操作", strItem, nil, nil, nil, menupos)
                Select Case menupos
                case 1
                        // ドライブ情報取得メソッドを実行
                        _DriveInfo()
                case 2
                        // ファイルリネームメソッドを実行
                        _Rename()
                case 3
                        // ファイルコピーメソッドを実行
                        _Copy()
                case 4
                        // ファイル圧縮メソッドを実行
                        _Compress()
                case 5
                        // ファイル解凍メソッドを実行
                        _UnCompress()
                case 6
                        // ファイル削除メソッドを実行
                        _Delete()
                case 7
                        // ファイル連結メソッドを実行
                        _Cat()
                case 8
                        // ファイルデフラグメソッドを実行
                        _Deflag()
                End Select
        Wend
End Method
```

```
Method _DriveInfo()
Begin
	Handy:ShowMessageBox("ドライブ2の情報を取得します","confirm")
	// ドライブ情報を取得します。
	FileSystem:DriveInfo(2)
	Handy:ShowMessageBox("ドライブサイズ:" & FileSystem:driveTotalSize ,"confirm")
	Handy:ShowMessageBox("ドライブ空き容量:" & FileSystem:driveFreeSize ,"confirm")
	Handy:ShowMessageBox("ドライブ内ファイル数:" & FileSystem:driveFileCount ,"confirm")
End Method

Method _Rename()
Begin
	Handy:ShowMessageBox("ファイル名を変更します","confirm")
	// ドライブ情報を取得します。
	FileSystem:Rename("2:sample.txt","2:sample2.txt")
End Method

Method _Copy()
Begin
	Handy:ShowMessageBox("ファイルをコピーします","confirm")
	// ファイル(コピー先)が存在しないので、ファイルをコピーする
	FileSystem:Copy("2:sample.txt","2:sample2.txt",true)
End Method

Method _Compress()
Begin
	Handy:ShowMessageBox("ファイルを圧縮します","confirm")
	// ファイルを圧縮します
	FileSystem:Compress("2:sample.txt", "2:sample.zip", "ZIP", true)
End Method

Method _UnCompress()
Begin
	Handy:ShowMessageBox("圧縮ファイルを解凍します","confirm")
	// 圧縮ファイルを解凍します
	FileSystem:Uncompress("2:sample.zip", "ZIP", true)
End Method

Method _Delete()
Begin
	Handy:ShowMessageBox("ファイルを削除します","confirm")
	// ドライブ2のファイル削除する
	FileSystem:Delete("2:sample.txt")
End Method

Method _Cat()
Begin
	Handy:ShowMessageBox("ファイルを連結します","confirm")
	// ドライブ2のファイルを連結します
	FileSystem:Cat("2:sample.txt","2:sample2.txt","2:cat.txt")
End Method
Method _Deflag()
Begin
	Handy:ShowMessageBox("ドライブ1, 3をデフラグします","confirm")
	// ドライブ1, 3をデフラグします
	FileSystem:Deflag()
End Method
```

例2 置き換えマスタを作成します。

```
Method main()
	ret
Begin
	// マスタファイルを作成
	With File
		:Initialize()
		:name = "2:replace.txt"
		:Puts("111,aaa\r\n","create")
		:Puts("222,bbb\r\n","append")
		:Puts("333,ccc\r\n","append")
		:Close()
	EndWith
	// インデックスファイルを作成
	With File
		:Initialize()
		:name = "2:replace.idx"
		:Puts("[FILE_DEFINITION]\r\n","create")
		:Puts("EXTENSION1 = csv\r\n","append")
		:Puts("EXTENSION2 = txt\r\n","append")
		:Puts("[INDEX_MODE]\r\n","append")
		:Puts("INDEXMODE = 3\r\n","append")
		:Puts("[RECORD_DEFINITION]\r\n","append")
		:Puts("RECORDFORMAT = 1\r\n","append")
		:Puts("RECORDLENGTH = 0\r\n","append")
		:Puts("SEPARATOR = 44\r\n","append")
		:Puts("[SORT DEFINITION]\r\n","append")
		:Puts("SORTTYPE = 0\r\n","append")
		:Puts("[INDEX_1]\r\n","append")
		:Puts("POSITION = 1\r\n","append")
		:Puts("LENGTH = 0\r\n","append")
		:Close()
	EndWith
	FileSystem:Delete("2:replace2.txt")
	Handy:ShowMessageBox("置き換えマスタを作成します","confirm")
	// 置き換えマスタ作成("2:replace2.txt")
	ret = FileSystem:SearchCreate("2:replace.txt","2:replace.idx","2:replace2.txt")
	If ret is false Then
		Handy:ShowMessageBox("置き換えマスタ作成に失敗しました","confirm")
	Else
		Handy:ShowMessageBox("置き換えマスタ作成に成功しました","confirm")
	With Search
		:Initialize()
		:name = "2:replace2.txt"
		ret = :GetFirst(1,"222")
		If ret is false Then
			Handy:ShowMessageBox("検索に失敗しました","confirm")
		Else
			Handy:ShowMessageBox("検索完了:" & ret,"confirm")
		EndIf
	EndWith
	EndIf
End Method
```

SQLiteデータベースのOpen/Closeを行います。簡単なコマンド実行も可能です。

SQLite	SQLiteデータベースのOpen/Closeを行います。

SQLiteデータベースのOpen/Closeを行います。簡単なコマンド実行も可能です。

プロパティ

なし

メソッド

名称	説明	引数	戻り値
Open	データベースをオープンします。	filename:DBファイル名	db_idx: DB番号※1 false:エラー終了
Close	データベースをクローズします。	db_idx:DB番号※1	true:正常終了 false:エラー終了
Execute	SQLコマンドを実行します。 「SQLコマンドの実行方法」(7-186ページ)参照 (SQLiteCommandコントロール)	db_idx:DB番号※1 cmd:コマンド文字列	true:正常終了 false:エラー終了
GetErrorStr	エラー文字列を取得します。	db_idx:DB番号※1	エラー文字列 false:エラー終了
GetErrorCode	エラーコードを取得します。	db_idx:DB番号※1	エラーコード false:エラー終了

※1　DBを指定するための番号です。Openメソッドの戻り値として取得できます。
Close, Execute, GetErrorStr, GetErrorCodeメソッドで必要となります。

SQLiteCommand	SQLiteデータベースに対するコマンドを実行します。

SQLiteデータベースに対するコマンドを実行します。データの取得も可能です。

プロパティ

なし

メソッド

名称	説明	引数	戻り値
Create	SQLコマンド設定の開始処理です。 「SQLコマンドの実行方法」(7-186ページ)参照	db_idx:DB番号※1	cmd_idx: コマンド番号※2 false:エラー終了
Delete	SQLコマンド設定の終了処理です。 「SQLコマンドの実行方法」(7-186ページ)参照	cmd_idx:コマンド番号※2	true:正常終了 false:エラー終了
SetCommandText	SQLコマンド文字列を設定します。プレースホルダ※3を埋め込むことが出来ます。 「SQLコマンドの実行方法」(7-186ページ)参照 「プレースホルダの使用方法」(7-186ページ)参照	cmd_idx:コマンド番号※2 cmd:コマンド文字列	true:正常終了 false:エラー終了
ExecuteNonQuery	設定されたSQLコマンドを実行します※4。 「SQLコマンドの実行方法」(7-186ページ)参照	cmd_idx:コマンド番号※2	true:正常終了 false:エラー終了
ExecuteReader	設定されたSQLコマンドを実行します※4。 「SQLコマンドの実行方法」(7-186ページ)参照	cmd_idx:コマンド番号※2	true:正常終了 false:エラー終了
SetParam	プレースホルダにパラメータを設定します。 「「SQLコマンドの実行方法」(7-186ページ)参照 「プレースホルダの使用方法」(7-186ページ)参照	cmd_idx:コマンド番号※2 idx:プレースホルダ番号 data:パラメータ値	true:正常終了 false:エラー終了
ResetParam	プレースホルダをリセットします。 「SQLコマンドの実行方法」(7-186ページ)参照 「プレースホルダの使用方法」(7-186ページ)参照	cmd_idx:コマンド番号※2	true:正常終了 false:エラー終了

名称	説明	引数	戻り値
Read	次のレコードを取得します。 ExecuteReader実行後に使用してください。 📖「Read/Valueメソッドによるデータ取得」(7-187ページ)参照	cmd_idx :コマンド番号※2	1:次のレコードに移動 0:次のレコードが存在しない false:エラー終了
Value	Readメソッド実行後、指定したフィールドデータを取得します。 📖「Read/Valueメソッドによるデータ取得」(7-187ページ)参照	cmd_idx :コマンド番号※2 idx :フィールド番号	フィールドデータ false:エラー終了
ValueCount	Readメソッド実行後、フィールド数を取得します。 📖「Read/Valueメソッドによるデータ取得」(7-187ページ)参照	cmd_idx :コマンド番号※2	フィールド数 false:エラー終了

※1　SQLiteコントロールのOpenメソッドの戻り値を指定してください。

※2　コマンドを指定するための番号です。Createメソッドの戻り値として取得できます。
Createを除く全てのメソッドで必要となります。

※3　埋め込んだプレースホルダには、後から文字列を設定することが出来ます(SetParam)。
プレースホルダをリセットする場合はResetParamを使用してください。

※4　ExecuteNonQueryとExecuteReaderの違い

●ExecuteNonQueryメソッド
結果を取得しない場合に使用します。

●ExecuteReaderメソッド
結果を取得する場合に使用します。
詳細は📖「Read_XXXメソッドによるデータ取得」を参照してください。

▶ 重 要

このコントロールは必ずSQLiteコントロールのOpenメソッド実行後に使用してください。
使用後はCloseメソッドでデータベースをクローズしてください。

■SQLコマンドの実行方法

SQLコマンドの実行には次の2つの方法があります。

●Executeメソッドで直接コマンドを実行する

次の様に1メソッドでコマンドを実行することが出来ます。
ただし、次のCmd_XXXメソッドとは異なり、データ取得やプレースホルダの使用ができません。

（例）Execute(db_idx, “INSERT INTO TEST_TBL VALUES(‘AAA’, ‘dat_1’, ‘1’);”)

●Cmd_XXXメソッドを使用してコマンドを実行する。

次の様に複数のメソッドを組み合わせてコマンド実行することができます。
プレースホルダの使用やDBからのデータ取得は、こちらの方法のみが対応しています。
（プレースホルダやデータ取得に関しては次節で説明します）

① コマンド設定を開始
comm_idx = :Create(db_idx)
② SQLコマンドをセット
:SetCommandText(comm_idx, "INSERT INTO TEST_TBL VALUES('AAA', 'dat_1', '1');")
③ SQLコマンドを実行
- 結果が返ってくるコマンドの場合(SELECT等)…:ExecuteReader(comm_idx)
- 結果が返ってこないコマンドの場合(INSERT等)…:ExecuteNonQuery(comm_idx)

④ コマンド設定を終了
:Delete(comm_idx)

■プレースホルダの使用方法

次の手順でプレースホルダを使用できます。

●SetParamメソッドでプレースホルダに値をセットする。

① プレースホルダを含むコマンドをセット
:SetCommandText(comm_idx, "INSERT INTO TEST_TBL VALUES(?, ?, ?);")
② プレースホルダに値をセット
:SetParam(comm_idx, 0, "BBB")
:SetParam(comm_idx, 1, "dat_2")
:SetParam(comm_idx, 2, "2")
③ コマンド実行
:ExecuteNonQuery(comm_idx)

●ResetParamメソッドでプレースホルダに値をリセットする。

さらにプレースホルダをリセットして再度設定することも可能です。
④ ResetParamメソッドでプレースホルダの値をリセット
:ResetParam(comm_idx)
⑤ SetParamメソッドでプレースホルダを再セット
:SetParam(comm_idx, 0, "CCC")
:SetParam(comm_idx, 1, "dat_3")
:SetParam(comm_idx, 2, "3")
⑥ コマンド実行
:ExecuteNonQuery(comm_idx)

■Read/Valueメソッドによるデータ取得

データ取得のために3つのメソッドが用意されています。
どのメソッドもExecuteReaderメソッドを実行後に使用してください。
① Readメソッド… 実行するたびにデータ読出しの対象を次のレコードにします。
　　データはValueメソッドで取得してください。
　　次のレコードが存在しない場合は0を返します。
② Valueメソッド…取得したレコードの指定したフィールドデータを取得します。
③ ValueCountメソッド…取得したレコードのフィールド数を取得します。

(例) 全レコードのデータをメッセージボックスで表示する。

```
comm_idx = :Create(db_idx)
:SetCommandText(comm_idx, "SELECT * FROM TEST_TBL;")
:ExecuteReader(comm_idx)
While (1)
    If :Read(comm_idx) == 0 Then
        Wbreak
    EndIf
    data1 = :Value(comm_idx, 0)
    data2 = :Value(comm_idx, 1)
    data3 = :Value(comm_idx, 2)
    Handy:ShowMessageBox("data:\n" & data1 & "\n" & data2 & "\n" & data3 & "\n", "ok" )
Wend
:Delete(comm_idx)
```

使用例

例1 データ追加後、全データを取得します。

```
Method main()
        db_idx
        comm_idx
        data1
        data2
        data3
Begin
        // DBオープン
        db_idx = SQLite:Open("2:test.db")

        With  SQLiteCommand
                comm_idx = :Create(db_idx)

                // テーブル作成
                :SetCommandText(comm_idx, "CREATE TABLE TEST_TBL(name, val_1, val_2);")
                :ExecuteNonQuery(comm_idx)

                // 1件目登録
                :SetCommandText(comm_idx, "INSERT INTO TEST_TBL VALUES('AAA', 'dat_1', '1');")
                :ExecuteNonQuery(comm_idx)
                // 2件目登録
                :SetCommandText(comm_idx, "INSERT INTO TEST_TBL VALUES(?, ?, ?);")
                :SetParam(comm_idx, 0, "BBB")
                :SetParam(comm_idx, 1, "dat_2")
                :SetParam(comm_idx, 2, "2")
                :ExecuteNonQuery(comm_idx)
                // 3件目登録
                :ResetParam(comm_idx)
                :SetParam(comm_idx, 0, "CCC")
                :SetParam(comm_idx, 1, "dat_3")
                :SetParam(comm_idx, 2, "3")
                :ExecuteNonQuery(comm_idx)

                // 登録件数取得
                :SetCommandText(comm_idx, "SELECT COUNT(*) FROM TEST_TBL;")
                :ExecuteReader(comm_idx)
                :Read(comm_idx)
                data1 = :Value(comm_idx, 0)
                Handy:ShowMessageBox("登録件数:"& data1, "ok" )

                // データ取得
                :SetCommandText(comm_idx, "SELECT * FROM TEST_TBL;")
                :ExecuteReader(comm_idx)
                While (1)
                        If :Read(comm_idx) == 0 Then
                                Wbreak
                        EndIf

                        data1 = :Value(comm_idx, 0)
                        data2 = :Value(comm_idx, 1)
                        data3 = :Value(comm_idx, 2)
                        Handy:ShowMessageBox("data:\n" & data1 & "\n" & data2 & "\n" & data3 & "\n", "ok" )
                Wend
```

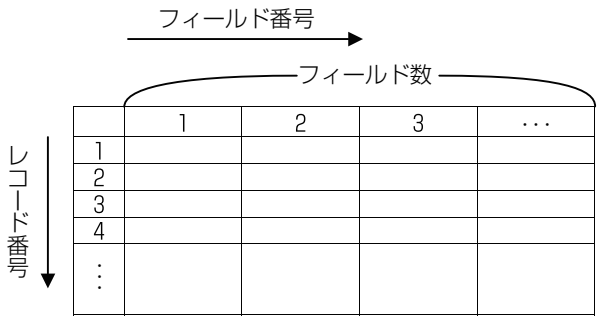
```
        // テーブル削除
        :SetCommandText(comm_idx, "DROP TABLE TEST_TBL;")
        :ExecuteNonQuery(comm_idx)

        :Delete(comm_idx)
    EndWith

    // DBクローズ
    SQLite:Close(db_idx)

    Handy:ShowMessageBox("END", "ok" )
End Method
```

7-9 マスタファイルとの照合

入力したデータを、指示マスタと照合します。
指示マスタの作成方法については、「6-7 マスタファイル」(6-40ページ)を参照してください。

Master	BTシリーズ内(ドライブ1、2、5)にある指示マスタと照合します。 ※ドライブ5はBT-W100/W80/1000/1500シリーズのみ。

マスタファイルのレコード番号、フィールド番号は、以下のように定義されています。



プロパティ

名称	説明	範囲	初期値	代入
keyNum	検索キーの数を指定します。 「検索キーの指定(keyNumプロパティ)」(7-193ページ)	0～10(BT-1000/1500シリーズ) 0～32(BT-W100/W80/W70/600シリーズ)	0	可
tmpFilename	照合用ワークファイル(インデックス情報用ファイル)名を指定します。 nilの場合には、ドライブ2に、「マスタファイル名.MTP」ファイルを作成します。	ドライブ番号:ファイル名※	nil	可
addFilename	追加情報用ワークファイル名を指定します。 nilの場合には、マスタファイルと同じドライブに、「マスタファイル名.ADD」ファイルを作成します。	ドライブ番号:ファイル名※	nil	可
error	エラー情報 エラー原因が格納されます。	文字列 「エラー情報(errorプロパティ)」(7-196ページ)	nil	不可

※ ファイル名は、拡張子を含めて32バイト以内になります。
Openメソッドのfile Nameに拡張子がない場合には、ファイル名+(.MTP/.ADD)のファイル名を使用するため、fileNameの文字数には注意してください。

メソッド

名称	説明	引数	戻り値
Initialize	各プロパティを初期化します。※10	なし	なし
Open	マスタ処理準備 指定するマスタファイルから、照合用ワークファイルを生成して、開きます。※12 📖「ファイルオープン・クローズ（Open、Closeメソッド）」（7-193ページ）	fileName： マスタファイル名 （ドライブ番号：ファイル名）※1 type：マスタ種別 （可変長のときはセパレータ、固定長のときは各項目長）※2 modifyColumn： 変更するフィールド番号 nil（変更しない） 1～10（BT-1000/1500シリーズ）※3 1～32（BT-W100/W80/W70/600シリーズ）※3	true：正常終了 false：エラー終了
Close	照合用ワークファイルをクローズします。 📖「ファイルオープン・クローズ（Open、Closeメソッド）」（7-193ページ）	なし	なし
SetKey	キー文字列設定 検索するフィールド番号と、キー文字列を設定します。 📖「マスタファイル照合（SetKey、GetFirst、GetNext、GetDataメソッド）」（7-194ページ）	itemIndex：フィールド番号 keyData：キーとするデータ（キー文字列）	true：正常終了 false：エラー終了
GetKey	指定するフィールド番号のキー文字列を取得します。	itemIndex：フィールド番号	キー文字列 false：フィールド番号エラー
GetFirst	キー文字列設定した内容に、最初に一致するレコードを検索します。 📖「マスタファイル照合（SetKey、GetFirst、GetNext、GetDataメソッド）」（7-194ページ）	sequence：nil（無効） 1～10（BT-1000/1500シリーズ）※3 1～32（BT-W100/W80/W70/600シリーズ）※3 📖「順序指定の有無（GetFirstメソッド）」（7-194ページ） incDelData：true（削除含む）※4 false（削除含まない）	レコードハンドル false：エラー終了
GetNext	「GetFirst」と同じ内容に、次に一致するレコードを検索します。 📖「マスタファイル照合（SetKey、GetFirst、GetNext、GetDataメソッド）」（7-194ページ）	なし	レコードハンドル false：エラー終了
GetAt	指定するレコード番号のレコードを取得します。 📖「レコード番号を指定して照合（GetAtメソッド）」（7-195ページ）	index：レコード番号 （0～9999）	レコードハンドル false：エラー終了
GetData	データ文字列取得 「GetFirst」、「GetNext」で検索したデータを取得します。 📖「マスタファイル照合（SetKey、GetFirst、GetNext、GetDataメソッド）」（7-194ページ）	column：フィールド番号	データ文字列 false：フィールド番号エラー

名称	説明	引数	戻り値
SetData	「Append」、「Modify」メソッドで使用するレコードのデータを内部バッファに格納します。	column：フィールド番号 value：フィールド値	true：正常終了 false：エラー終了
SetDeleteFlag	キー文字列設定に一致するレコードに、削除フラグを立てます。 📖「レコードの削除、復帰（SetDeleteFlagメソッド）」(7-195ページ)	delete：true（削除） false（復帰） allProc：true（全レコード対象） false（最初のレコードが対象）	true：正常終了 false：エラー終了
ModifyItem※5	キー文字列設定に一致する最初のレコードのフィールドを変更します。	modifyData：変更文字列（最大10バイト） 10バイトを超えるデータのときは11バイト以降を切り捨てて設定します。	true：正常終了 false：エラー終了
GetCount	キー文字列設定に一致したレコード数を取得します。 検索キーを0（keyNum＝0）にして、GetCount( )を実行すると、レコードの総数を取得できます。	incDelData：true（削除含む）※6 false（削除含まない）	取得レコード数（0～10000）※11 false：エラー終了
Append	追加情報用ワークファイルにレコードを追加します。 📖「レコードの追加・修正（Append、Modifyメソッド）」(7-196ページ)	なし	true：正常終了 false：エラー終了
Read	n番目のレコードを読み取ります。 📖「GetAtメソッドとReadメソッド」(7-195ページ)	index：n番目 incDelData：true（削除含む）※7 false（削除含まない）	レコードハンドル false：エラー終了
Delete	レコードハンドルで指定するレコードを削除します。 また、削除したレコードを復活できます。	handle：レコードハンドル※8 flag：true（削除する） false（復活させる）	true：正常終了 false：エラー終了
Modify	レコードハンドルで指定するレコードを修正します。 📖「レコードの追加・修正（Append、Modifyメソッド）」(7-196ページ)	handle：レコードハンドル※8	true：正常終了 false：エラー終了
Restore	追加情報用ワークファイル（addFilename）の情報を反映した、照合用マスタファイルを作成します。	fileName：生成するマスタファイル名 （ドライブ番号：ファイル名）	true：正常終了 false：エラー終了
GetTotalCount	指示マスタの全件数を取得します GetCountのkeyNum=0に設定したときと同じ結果が得られます。	incDelData： true（削除含む） false（削除含まない）	取得レコード数 （0～32767：BT-1000/1500シリーズ） （0～160000：BT-W100/W80/W70/600シリーズ） false：エラー終了
GetRecordStatus	指示マスタの消しこみ状況を取得します。	index:n番目のデータ	成功時：※9 false：エラー終了

※1　ドライブを指定しない場合はドライブ2になります。
　　指定できるマスタファイルは、1つのマスタ識別番号につき1つです。

※2 可変長のときのセパレータは1バイトです。可変長のマスタファイルを新規作成する場合は、セパレータのあとにフィールド数を指定します。

例 三つのフィールドのマスタファイルならば「,3」となります。
また、固定長のときは各項目長を「|」をセパレータとして設定します。

例 "●●●▲▲ ×××××○○○○○○○○○○" というデータならば「3|2|5|10」となります。

可変長、固定長については、「6-7 マスタファイル」(6-40ページ)を参照してください。

※3 ModifyColumnで設定したフィールドのみModifyItemで修正できます。
※4 true(削除含む)のとき、削除フラグがついた状態のレコードも含めて照合します。
※5 「ModifyItem」は、BT-H55W(BT-500用)/H91W(BT-900用)のスクリプトとの互換用です。アプリケーションを新規作成する場合には、「Modify」を使用してください。
※6 true(削除含む)のとき、削除フラグがついた状態のレコードも含めてカウントされます。
※7 incDelDataをfalseに指定した場合、削除されたレコードのindexを指定すると、falseが返ります。
※8 Readの戻り値を使用します。
※9 成功時には以下の値が返ります。失敗時にはエラー情報がerrorプロパティに書き込まれます。
"ORIGINAL": マスタファイル上に存在する有効なレコード
"MODIFIED": 修正され追加マスタファイル上に存在する有効なレコード
"DELETED": 削除されたレコード
※10 Closeメソッド後、Openメソッドでマスタファイルのオープンをおこなう場合には、その間にInitializeメソッドを実行して各プロパティの初期化をおこなってください。
※11 該当レコード数が10000より多い場合でも、戻り値として10000が取得されます。
※12 Open時にプロパティで指定する照合用ワークファイルとは別に、高速に検索するための検索補助ファイルを作成します。検索補助ファイルはtmpFilename(補助マスタファイル名)で指定したファイル名の先頭に"__"(アンダーバー2つ)を追加したファイル名でドライブ2に作成されます。

■検索キーの指定(keyNumプロパティ)

検索キーの数を指定します。

例 フィールド1、フィールド3を検索する場合には以下のようになります。
SetKey(1,"key"),Setkey(2,nil),SetKey(3,"key"),keyNum=2

ポイント
- **有効な(nilではない)キーの数がkeyNumよりも少ない状態でGetFirstを呼び出すと、実行時例外が発生します。**
- **有効なキーの数がkeyNumよりも多い状態でGetFirstを呼び出すと、フィールド1～フィールド(keyNum)まで順番のキーで検索します。**

■ファイルオープン・クローズ(Open、Closeメソッド)

●ファイルオープン(「Open」メソッド)

マスタ処理の準備をします。
マスタファイルをもとに、システムが照合用ワークファイルを作成します。
照合用ワークファイルは、削除したフラグ等を管理するためのファイルです。
ファイル名は次のようになります。
ファイル名:マスタファイル名の拡張子を「.MTP」にして作成。

例 1:MASTER.DAT→2:MASTER.MTP

参考 既に同じ名前の照合用ワークファイルが存在するときは、既存のファイルをオープンします。
存在しないマスタファイルを指定した場合は、指定したファイル名で新規作成されます。

●ファイルクローズ(「Close」メソッド)

照合用ワークファイルをクローズします。
照合用ワークファイルは、照合終了後も削除されませんので、必要に応じて削除してください。

■マスタファイル照合(SetKey、GetFirst、GetNext、GetDataメソッド)

照合手順は次のようになります。

1 **「SetKey」メソッドで、照合するデータのフィールド番号とキー文字列を指定します。**

複数の検索フィールドを設定する場合は、フィールド番号が小さい順になるように「SetKey」を必要な個数分記述します。

参考 設定したデータを消すときは、""を代入します。

2 **「GetFirst」メソッドで一致する最初のレコードを検索します。**

「incDelData」が「true(削除含む)」のとき、削除状態のレコードも含めて検索します。
一致するデータがあったとき、「GetData」メソッドでフィールドごとにデータを取得します。(レコードハンドルは「GetFirst」の戻り値で取得します。)

3 **「GetNext」メソッドで、「GetFirst」の条件に一致する次のレコードを取得します。**

一致するデータがあったとき、「GetData」メソッドでフィールドごとにデータを取得します。(レコードハンドルは「GetNext」の戻り値で取得します。)

■順序指定の有無(GetFirstメソッド)

- 引数「Sequence」には、「nil」または「1」～「10」を設定できます。
  「nil」を設定すると順序指定が無効、「1」～「10」を設定すると順序指定が有効になります。
- 設定値「1」～「10」はフィールド番号を示します。「1」～「10」を設定すると、設定したフィールド番号が順序指定の対象になって、レコードされている順序どおりに照合しないとエラーとなります。

たとえば、"AAA"、"222"、"ddd"と入力されたデータを、次のような指示マスタと照合します(④のレコードと一致します)。

BBB,222,aaa …①
BBB,111,bbb …②
BBB,111,ccc …③
AAA,222,ddd …④
AAA,111,eee …⑤
AAA,222,fff …⑥

入力されたデータから、「SetKey」に(1,"AAA")、(2,"222")、(3,"ddd")を設定します。
「Sequence」に「nil」、「1」～「3」を指定すると、次のようになります。

- 「nil」を指定したときは、順序指定が無効なので、照合OKになります。
- 「1」を指定したときは、「SetKey」に設定したデータが、指示マスタの第1フィールドのデータと先頭から順番に照合チェックされます。その結果、指示マスタの第1フィールド、第1番目のデータが"AAA"と一致しないので、照合エラーになります。(①、②、③のレコードがないときは照合OKになります)。
- 「2」を指定したときは、指示マスタの第1フィールドの順序チェックはされないで、第1フィールドがAAA"であるレコードの中で、第2フィールドの先頭から順番に照合チェックされます。その結果、照合OKになります。
- 「3」に指定したときは、「1」、「2」を指定したときと同様に、指示マスタの第1、第2フィールドの順序チェックはされないで、第1フィールドが"AAA"で、第2フィールドが"222"のレコードの中で、第3フィールドの先頭から順番に照合チェックされます。その結果、照合OKになります。

■レコード番号を指定して照合(GetAtメソッド)

「GetFirst」、「GetNext」、「GetCount」メソッドを実行した後に、任意のレコード番号を指定してデータを取得することができます。

- 引数「index」が「0」のときは、「GetFirst」メソッドで取得したのと同じ位置のデータを取得します。
- 引数「index」が「n」(1以上)のときは、「GetNext」をn回実行したときに取得したのと同じ位置のデータを取得します。
- 「GetCount」メソッドの後に記述するときは、引数「index」が0～(GetCountの戻り値−1)の範囲になるように指定します。
- 以下のような場合は、「GetAt(0)」メソッドで「GetFirst」と同じレコードを取得して、最後の「GetNext」メソッドで一致するレコードの4番目を取得します。

  GetFirst → GetNext → GetNext → GetAt(0) → GetNext

■レコードの削除、復帰(SetDeleteFlagメソッド)

削除･復帰の手順は次のようになります。

1 **「SetKey」メソッドで、削除･復帰するデータのフィールド番号と設定キーを指定します。**

2 **「SetDeleteFlag」メソッドを使用して一致するレコードの削除状態を変更します。**

- 「delete」が「true(削除)」のとき、条件一致レコードを削除します。(レコードに削除フラグが立ちます。)
- 「delete」が「false(復帰)」のとき、条件一致レコードを復帰します。(レコードの削除フラグを外します。)
- 「allProc」が「true(全レコード対象)」のとき、全レコードを対象に削除/復帰をおこないます。
- 「allProc」が「false(最初のレコードが対象)」のとき、条件に一致する最初のレコードを対象に削除 / 復帰をおこないます。

■GetAtメソッドとReadメソッド

GetAtメソッドとReadメソッドの機能の違いは以下のようになります。

GetAt

検索キーに一致したN番目のレコードを取得します。

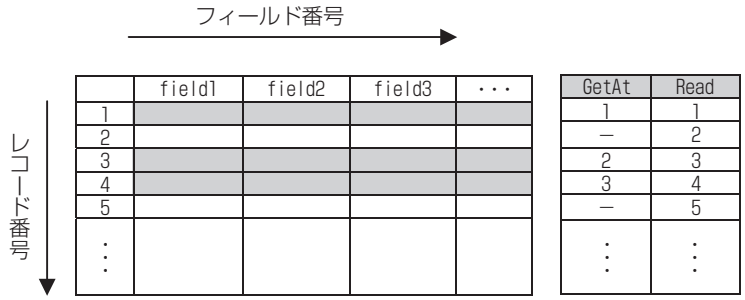
GetFirst＋GetNextまたはGetCountを実行した後に使用できます。

Read

マスタファイルのN番目のレコードを取得します。

GetFirst、GetCountを実行する前にも実行できます。

例　GetAtメソッド、Readメソッドを使用した場合のレコード取得例



■:GetFirst＋GetNextメソッドまたはGetCountメソッドを実行して、検索キーと一致したレコード

■レコードの追加・修正（Append、Modifyメソッド）

Append、Modify、Deleteメソッドで追加、修正、削除したデータは、addFilenameプロパティで指定した追加情報用ワークファイルに蓄積されます。

蓄積した追加情報用ワークファイルの情報を反映したマスタファイルを生成するには、Restoreメソッドを使用します。

■エラー情報（errorプロパティ）

エラー情報	説明
"ERR_FILE_DRIVENAME"	ドライブ名エラー
"ERR_FILE_FILENAME"	ファイル名エラー
"ERR_FILE_NONAME"	ファイル名を指定していない
"ERR_FILE_MODE"	該当モードではオープンできない
"ERR_FILE_NOT_EXIST"	ファイルは存在しない
"ERR_FILE_ALREADY_EXIST"	ファイルは既に存在している
"ERR_FILE_ALREADY_OPEN"	ファイルは既にオープンしている
"ERR_FILE_NOT_OPEN"	ファイルをオープンしていない
"ERR_FILE_FORMAT"	ファイルフォーマットエラー
"ERR_FILE_ENTRY_OVER"	ファイルエントリー数オーバー
"ERR_FILE_NOSPACE"	空き容量不足
"ERR_FILE_MAXOPEN"	ファイルオープン数オーバー
"ERR_FILE_SIZE"	ファイルサイズを超えるアドレスを指定した
"ERR_FILE_NOT_FOUND"	該当ファイルが見つからない
"ERR_FILEREC_OVERFLOW"	ファイルレコードが最大数を超えた

注記	**照合用ワークファイル (.MTP)、追加情報用ワークファイル (.ADD) には、オープンしたマスターファイルのインデックス情報・更新情報が保存されます。そのため、マスターファイルで新規に処理をおこなう場合、ワークファイルを削除する必要があります。 マスターファイル処理中にワークファイルを削除した場合は、更新内容が削除されてしまいます。**

注意事項

- Restore メソッドを使用して、マスタファイルを更新する場合には、一度別のファイル名を指定してください。
  マスタファイルおよび照合用ワークファイルと同じファイル名を指定するとエラーになります。
- Openメソッドは内部で照合用ワークファイルを作成するため、処理に時間がかかることがあります。
- GetCountは内部でGetFirst、GetNextを繰り返すため、処理に時間がかかることがあります。
- レコードハンドルは、マスタファイルのN件目の場合、Nが返ります。従って、Readの第1引数と戻り値は同じ値になります。

使用例

例1 フィールド番号「1」が「111」、フィールド番号「2」が「333444555」のレコードを、順序を指定しないで照合します。

```
Method sample1()
    strItem[10]                                    // 一致レコードのデータ格納用
    recnum                                         // レコード数格納用
    itemnum                                        // 一致レコード数格納用
    cnt1
    cnt2
    Begin
        Handy:ShowMessageBox("sample1","confirm")
        FileSystem:Delete("MASTER.MTP")
        Master:Open("MASTER.TXT",",",nil)          // 指示マスタオープン
        Master:SetKey(1,"111")                     // キーを設定
        Master:SetKey(2,"333444555")               // キーを設定
        Master:keyNum = 2
        recnum = Master:GetCount(false)            // レコード数取得

        If recnum is not false Then
            For cnt1 = 1 to recnum                 // レコード照合、取得
                If cnt1 == 1 Then
                    // 最初の一致レコード照合
                    itemnum = Master:GetFirst(nil, false)
                Else
                    itemnum = Master:GetNext()     // 次の一致レコード照合
                EndIf

                If itemnum is false Then
                    Fbreak
                EndIf

                strItem[cnt1-1] = Master:GetData(1)
```

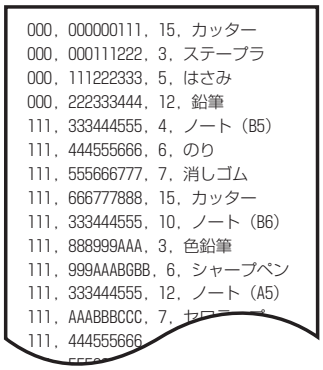
```
                For cnt2 = 2 to itemnum
                    // 一致レコードのデータ取得
                    strItem[cnt1-1] = strItem[cnt1-1] & "," & Master:GetData(cnt2)
                Next
                Handy:DebugOut(strItem[cnt1-1] & "¥n")
            Next
            Handy:ShowMessageBox((cnt1-1) &
                                    "件のデータが見つかりました","confirm")
        EndIf
        Master:Close()
End Method
```


例2 フィールド番号「1」が「111」、フィールド番号「2」が「333444555」レコードを、順序を指定して照合します。

```
Method sample2()
    strItem[10]                                 // 一致レコードのデータ格納用
    recnum                                      // レコード数格納用
    itemnum                                     // 一致レコード数格納用
    cnt1
    cnt2

    Begin

        Handy:ShowMessageBox("sample2","confirm")
        FileSystem:Delete("MASTER.MTP")
        Master:Open("MASTER.TXT",",",nil)       //指示マスタオープン
        Master:SetKey(1,"111")                  //キーを設定
        Master:SetKey(2,"333444555")            //キーを設定
        Master:keyNum = 2
        recnum = Master:GetCount(false)         //レコード数取得

    If recnum is not false Then
        For cnt1 = 1 to recnum                  //レコード照合、取得
            If cnt1 == 1 Then
                                                // 最初の一致レコード照合
               itemnum = Master:GetFirst(nil, true)
            Else
               itemnum = Master:GetNext()       // 次の一致レコード照合
            EndIf

            If itemnum is false Then
               Fbreak
            EndIf

            strItem[cnt1-1] = Master:GetData(1)
```

```
            For cnt2 = 2 to itemnum
                // 一致レコードのデータ取得
                strItem[cnt1-1] = strItem[cnt1-1] & "," & Master:GetData(cnt2)
            Next

            Master:SetDeleteFlag(true, false)
        Next
        Handy:ShowMessageBox((cnt1-1) & "件のデータが見つかりました","confirm")
    EndIf
    Master:Close()
End Method
```

例3　下図のような指示マスタに対して、削除処理を実行します。



フィールド番号「1」が「111」で、フィールド番号「2」が「333444555」のレコードを「strItem」に格納します。また、その条件に一致する最初のレコードを削除状態にします。

```
Method sample3()
    strItem[10]
    itemnum
    cnt1

    Begin

        Handy:ShowMessageBox("sample3","confirm")

        Master:Open("MASTER.TXT", ",",nil)          // 指示マスタオープン
        Master:SetKey(1,"111")                      // キーを設定
        Master:SetKey(2,"333444555")                // キーを設定
        Master:keyNum = 2

        While true                                  //レコード照合
            itemnum = Master:GetFirst(nil, false)

            If( itemnum is false ) Then Wbreak EndIf
```

```
        For cnt1 = 1 to itemnum
            strItem[cnt1-1] = Master:GetData(cnt1)
        Next

        Master:SetDeleteFlag(true, false)      // 一致レコードを削除状態
    Wend

    Master:Close()
```

実行後、「strItem」に「111,333444555,4,ノート(B5)」というレコードがセットされ、削除状態になります。

例4　例2の指示マスタに対して復帰処理をします。実行後、全レコードが復帰します。

```
    Master:Open("MASTER.TXT", ",",nil)              // 指示マスタオープン
    Master:SetKey(1,"111")                          // キーを設定
    Master:SetKey(2,"333444555")                    // キーを設定
    Master:keyNum = 2
    Master:SetDeleteFlag( false, true)              // 全レコード復帰
    Master:Close()

End Method
```

例5　「GetAt」メソッドでレコード番号を指定して、レコードを変更します。

```
Method sample4()
    strItem[10]                                     // 一致レコードのデータ格納用
    recnum                                          //レコード数格納用
    itemnum                                         //一致レコード数格納用
    cnt1
    cnt2

    Begin

        Handy:ShowMessageBox("sample4","confirm")
        FileSystem:Delete("MASTER.MTP")
        Master:Open("MASTER.TXT",",",1)             // 指示マスタオープン
        Master:SetKey(1,"111")                      // キーを設定
        Master:SetKey(2,"333444555")                // キーを設定
        Master:keyNum = 2
        recnum = Master:GetCount(false)             // レコード数取得

        If recnum Then
            For cnt1 = 1 to recnum                  // レコード照合、取得
                itemnum = Master:GetAt(cnt1-1)
                strItem[cnt1-1] = Master:GetData(1)
```

```
                For cnt2 = 2 to itemnum
                    // 一致レコードのデータ取得
                    strItem[cnt1-1] = strItem[cnt1-1] & ",
                    " & Master:GetData(cnt2)
                Next
            Next
        EndIf

        Handy:ShowMessageBox(recnum & "件のデータが見つかりました","confirm")
        itemnum = Master:GetFirst(nil, false)          // 一致レコード照合
        Master:ModifyItem("999")
        recnum = Master:GetCount(false)                // レコード数取得
        If recnum Then
            For cnt1 = 1 to recnum                     // レコード照合、取得
            itemnum = Master:GetAt(cnt1-1)
            strItem[cnt1-1] = Master:GetData(1)
            For cnt2 = 2 to itemnum
                // 一致レコードのデータ取得
                strItem[cnt1-1] = strItem[cnt1-1] & "," & Master:GetData(cnt2)
            Next
        Next
    EndIf

    Handy:ShowMessageBox(recnum & "件のデータが見つかりました","confirm")
    Master:Close()

End Method
```

例6　「SetData」、「Append」、「Read」、「Modify」、「Delete」、「Restore」メソッドを使用して、マスタファイルを修正します。

```
// マスタサンプル
Method sample5()
    i
    h

    Begin
        Handy:ShowMessageBox("sample5","confirm")

        With Master
            :Initialize()
            :Open("test.dat", ",", nil)

            // 第1フィールドが"data*"というレコードを10件追加します
            For i = 1 To 10
                :SetData(1, "data" & i)
```

```
            :Append()
        Next

        // レコードを読み出します
        For i = 1 To 10
            :Read(i,false)
            Handy:ShowMessageBox(:GetData(1), "ok")
        Next
        // 5件目のデータを修正します
        h = :Read(5,false)
        :SetData(1, "modified")
        :Modify(h)
        // レコードを読み出します
        For i = 1 To 10
            :Read(i,false)
            Handy:ShowMessageBox(:GetData(1), "ok")
        Next

        // 7件目のデータを削除します
        h = :Read(7,false)
        :Delete(h, true)

        // レコードを読み出します
        For i = 1 To 10
            If :Read(i,false) is false Then
            Handy:ShowMessageBox(i&"番目のレコードは削除されました", "ok")
            Else
            Handy:ShowMessageBox(:GetData(1), "ok")
            EndIf
        Next

        // Append/Modify/Deleteした結果をマスタファイルに反映します
        :Restore("test2.dat")
        :Close()
    End With

    With FileSystem
        // 古いファイルを削除します
        :Delete("test.dat")
        :Delete("test.mtp")
        :Delete("test.add")
    End With
End Method
```

7-10 ログファイルの操作

ログファイルのレコードの追加、読み込み、書き込みなどをおこないます。

LogRecord	BTシリーズ内(ドライブ3、4)にあるログファイルを読み込み、書き込みします。ログファイルは1度に1ファイルのみ開くことができます。 ドライブ4は全ログ送信ができません。 ※ドライブ4はBT-600シリーズのみ。

プロパティ

名称	説明	範囲	初期値	代入
name	ログファイル名をドライブ番号を含めて指定します。※1	ドライブ番号:ファイル名	nil	可
separator	レコードのデータを区切るセパレータを設定します。	文字列で指定(0x20～0x7E)※2 カンマ、スペース、TABなど	","	可
recordCount	ログファイル中のレコード数が格納されます。	0～	0	不可
fieldNum	レコード内のフィールド数を設定します。	1～32	0	可
header	レコードのヘッダ情報 各ログレコードのはじめに付加されます。 「ヘッダ情報(headerプロパティ)」(7-206ページ)	yyyymmddhhmmssiiiii (年月日 時分 秒 端末ID) または yyyymmddhhmmssiiiiiiiiii (年月日 時分 秒 IPアドレス)	nil	可

※1 アクセスできるドライブは、ドライブ3、4です。
※2 TAB以外の制御コードは使用できません。

ポイント

BT-600シリーズのドライブ4は、BT-500シリーズとの互換性を保つために用意されています。
そのため、新規開発では、使用しないことをお勧めいたします。

メソッド

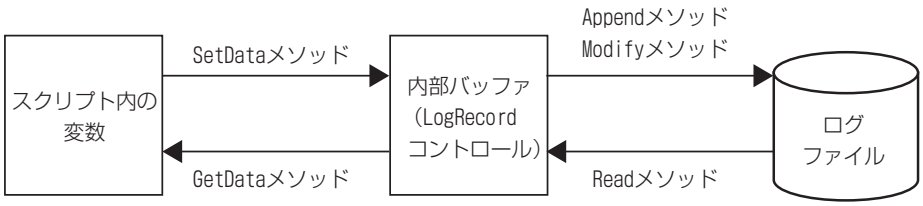
名称	説明	引数	戻り値
Initialize	各プロパティを初期化します。	なし	なし
Read	指定するログファイルのレコードを内部バッファに格納します。 「ログレコードの書き込み、読み込み(SetData、Append、Read、GetDataメソッド)」(7-204ページ)	recNum:レコード番号(1～32767)	レコードハンドル
GetData	「Read」で参照しているLogRecordの内部バッファのレコードから、データを取得します。 「ログレコードの書き込み、読み込み(SetData、Append、Read、GetDataメソッド)」(7-204ページ)	column: フィールド番号(1～32)	フィールド情報 nil
SetData	「Append」、「Modify」するレコードのデータを、LogRecordの内部バッファに格納します。 「ログレコードの書き込み、読み込み(SetData、Append、Read、GetDataメソッド)」(7-204ページ)	column:フィールド番号(1～32) value:フィールド情報※2	true:正常終了 false:エラー終了

名称	説明	引数	戻り値
Append	指定するログファイルの最後にレコードを追加します。※3 📖「ログレコードの書き込み、読み込み（SetData、Append、Read、GetDataメソッド）」（7-204ページ）	なし	true：正常終了 false：エラー終了
Delete	レコードハンドルで指定するレコードを削除します。 📖「ログレコードの削除、修正（Delete、Modifyメソッド）」（7-205ページ）	handle： レコードハンドル※1	true：正常終了 false：エラー終了
Modify	レコードハンドルで指定するレコードを修正します。 📖「ログレコードの削除、修正（Delete、Modifyメソッド）」（7-205ページ）	handle： レコードハンドル※1	true：正常終了 false：エラー終了
SetLogHeaderType	ログのヘッダ情報の種別を指定します。 ファイル送信後またはフォーマット後に呼び出してください。	type： "termid"（端末ID） "ip"（IPアドレス）※4	true：正常終了 false：エラー終了

※1 「Read」メソッドの戻り値を使用します。
※2 データにCR（0x0D）やLF（0x0A）は使用できません。
※3 ログは 32767 件まで書き込めます。それを超えると仕様外動作となり、PC 送信時などにログが上書きされ消える場合があります。
※4 BT-W100/W80/W70/1000/1500シリーズのみ使用可能です。初期値は"ip"です。

■ログレコードの書き込み、読み込み（SetData、Append、Read、GetDataメソッド）

ログファイルへのデータの読み書きは、次のようにLogRecordコントロール内部のバッファ（レコードハンドル）を介しておこないます。



ログレコードの書き込み、読み込み手順は以下のようになります。

●書き込み

1 **フィールド情報の書き込み（「SetData」メソッド）**

「SetData」メソッドで指定するフィールド番号（column）にフィールド情報（value）を書き込みます（バッファに格納します）。

2 **レコードの書き込み（「Append」メソッド）**

「Append」メソッドでバッファに格納されているデータを、セパレータを区切り記号としてログレコードをログの最終に書き込みます。ファイルのオープン・クローズ処理は不要です。
日付情報、時刻情報を自動的に付加し、最終データと比較して同じデータが存在するとデータの圧縮をおこないます。

●読み込み

1 **レコードの読み込み(「Read」メソッド)**

「Read」メソッドで指定するログレコードを読み込みます(読み込んだレコードは内部のバッファに格納されます)。読み込みレコードの終端にNULLが付加されます。

2 **フィールド情報の取得(「GetData」メソッド)**

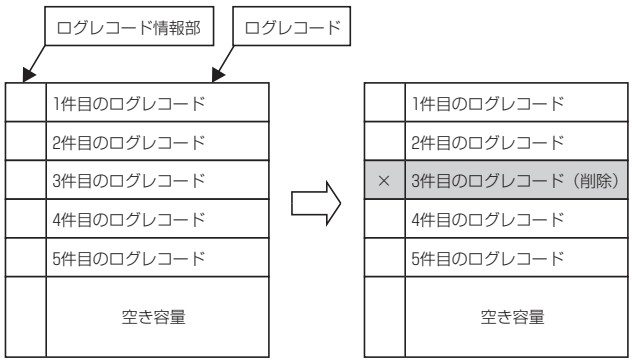
「Read」メソッドで内部バッファに格納したデータから、「GetData」メソッドでフィールド番号(column)の値(フィールド情報)を取得します。

■ログレコードの削除、修正(Delete、Modifyメソッド)

LogRecordコントロールで書き込まれるログレコードは、すべて末尾に書き込まれます。このため、修正・削除時には以下のことに注意する必要があります。

●レコード削除(Deleteメソッド)

ログレコードの削除は、ログレコードそのものを削除しているのではなく、内部的にはレコードの先頭の「ログレコード情報部」に削除したという情報をつけることによって削除処理をしています。
たとえば、5件のレコードが存在している場合に、3件目のレコードを削除する場合を以下に示します。

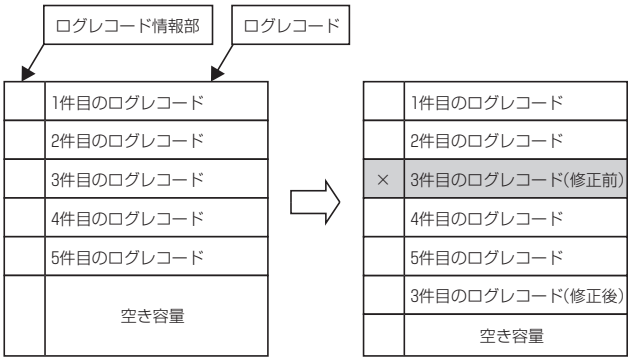


「Delete」メソッドによって3件目のログレコードを削除したとき、コントロール内部では、3件目のログレコード情報部に削除したという情報をつけています。

- ログレコードの領域は削除していないため空き容量は増えません。
- 削除したログレコードは復活することはできません。

●レコード修正(Modifyメソッド)

ログレコードの修正は、修正後のデータをレコードの最後に追加します。内部的にはレコードの先頭の「ログレコード情報部」に削除したという情報をつけて、変更後のデータを末尾に記録します。
たとえば、5件のレコードが存在している場合に、3件目のレコードを修正する場合を以下に示します。



「Modify」メソッドにより3件目のログレコードを修正すると、コントロール内部では、3件目のログレコードに削除したという情報をつけて、修正後のログデータを末尾に記録します。
この場合でも、ログレコード番号は変化しません。
3件目の修正後データのレコード番号は、3番になります。

- ファイルサイズは1レコード分増加します。
- 消去に失敗したときは実行時例外となります。

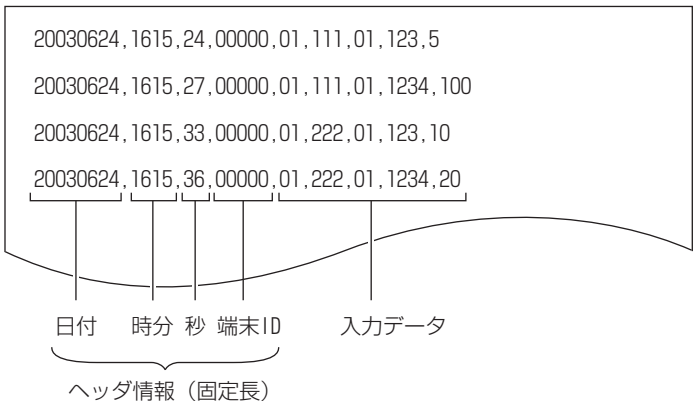
▶ 重 要

ログレコードの削除・修正をおこなうときは、直前に該当するログレコードの読み込みをおこなってください。「Read」メソッドにより「header」プロパティの整合性が取れなくなり正しく動作しなくなります。

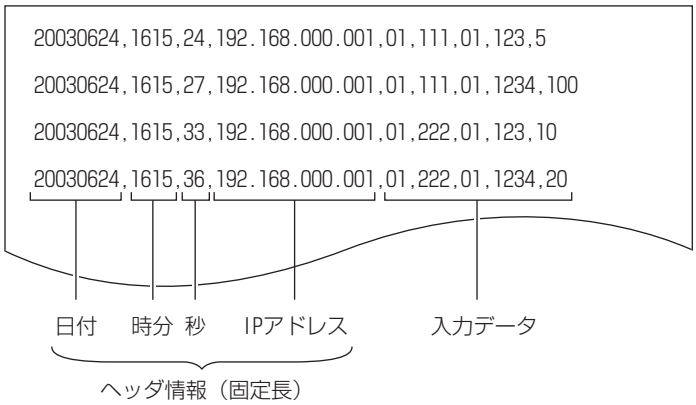
■ヘッダ情報(headerプロパティ)

ログレコードは下図のようなフィールド構成になります。

- **端末IDの場合**



- IPアドレスの場合



各レコードの入力データの前に日付や端末ID/IPアドレスなどのヘッダ情報が付加されます。

▶ 重 要

- 「header」プロパティには、セパレータは入りません。システムが自動的に付加します。
- 「header」プロパティに不正なサイズの文字列を設定して「Modify」を実行すると、実行時例外になります。

使用例

例1　ログレコードを追加します。

```
Method Init()
        Begin
                LogRecord:name = "3:logfile.txt"                    // ファイル名
                LogRecord:separator = ","                           // セパレータの設定
End Method

// 追加するログデータを設定
Method SetData( Const data[] )
        cnt1
        Begin
                LogRecord:fieldNum = data.size                      // フィールド数を設定
                For cnt1 = 1 to LogRecord:fieldNum
                        LogRecord:SetData(cnt1, data[cnt1-1])       // バッファに格納
                Next
End Method

// 新しいレコード書き込み
Method Append( Const data[] )
        Begin
                        SetData( data )
                        Return( LogRecord:Append() )
End Method
```

例2　ログレコードを読み込み、ヘッダ情報、レコード情報を取得します。

```
Method Read( data_idx, header[], data[] )
    recordHandle
    recordIndex
    tmp
    Begin

        // レコードを検索しバッファに格納
        recordHandle = LogRecord:Read(data_idx)
        If( recordHandle is false ) Then
            Return( nil)
        EndIf

        tmp = LogRecord:header                    // ヘッダの取り出し
        If tmp.length < 4 Then
            Return( nil )
        EndIf

        //ヘッダ情報を切り出す
        header[0] = tmp.Left(8)                   // 年月日
        header[1] = tmp.Mid(8,4)                  // 時分
        header[2] = tmp.Mid(12,2)                 // 秒
        header[3] = tmp.Right(15)                 // IPアドレス
        If GetData( data ) is nil Then
            Return( nil )
        EndIf

        Return(recordIndex)

End Method

// 読み込んだデータをフィールドごとに設定
Method GetData( data[] )
    cnt1
    Begin
        If data.size < LogRecord:fieldNum Then
            Return( nil )
        EndIf

        For cnt1 = 1 to LogRecord:fieldNum
            // フィールドのデータを取得
            data[cnt1-1] = LogRecord:GetData(cnt1)
        Next

        Return( LogRecord:fieldNum )
End Method
```

例3　指定するログレコードを書き換えます。

```
Method Modify( recordHandle, header[], Const data[] )
    Begin
        LogRecord:header=header.Merge("")
        SetData( data )
        Return( LogRecord:Modify( recordHandle) )
End Method
```

例4　指定するログレコードを削除します。

```
Method DeleteLog( recordHandle )
    Begin
        Return( LogRecord:Delete(recordHandle) )
End Method
```

7-11 高速検索

高速にデータファイルを検索します。
置き換えマスタに対して使用します。

Search	BTシリーズ内(ドライブ1、2、5)のインデックス情報が付加されている置き換えマスタを高速に検索します。 ※ドライブ5はBT-W100/W80/1000/1500シリーズのみ。

参考
- インデックス情報が付加されていないファイル(BTシリーズ内のファイル)にアクセスする場合は、「File/FileStream」コントロールを使用します。
- 置き換えマスタの作成については、「付録4 置き換えマスタ」(付-6ページ)を参照してください。

プロパティ

名称	説明	範囲	初期値	代入
name	検索するファイル名を指定します。	ドライブ番号:ファイル名※	nil	可
error	エラー情報 エラー原因が格納されます。	文字列 「エラー情報(errorプロパティ)」(7-212ページ)	nil	不可

※ ドライブは1、2のみ有効です。ドライブを指定しない場合はドライブ2になります。
指定できるファイルは1つのみです。

メソッド

名称	説明	引数	戻り値
Initialize	各プロパティを初期化します。	なし	なし
GetFirst	検索を開始して、最初に一致したデータを取得します。 「高速検索(GetFirst、GetNextメソッド)」(7-212ページ)	key:検索キー(インデックス定義の番号1～9)※1 string:検索文字列※2	一致レコード※3 nil:一致レコードなし
GetNext	次に一致したデータを取得します。 「高速検索(GetFirst、GetNextメソッド)」(7-212ページ)	なし	一致レコード※3 nil:一致レコードなし

※1 「アプリケーション作成機能」では検索キーは「1」固定です。
※2 ワイルドカード(*)は使用できません。
※3 レコード終端の改行コード(CR+LF)は取り除かれます。

■高速検索(GetFirst、GetNextメソッド)

●「GetFirst」メソッド

「name」プロパティでファイル名をあらかじめ指定しておき、検索キーと検索文字列をキーに検索を開始します。

一致するレコードが存在するときは該当行の全体(キーも含む)を返し、一致するレコードがないときはnilを返します。

ポイント

- **ファイル名、引数が不正なときもnilを返します。**
- **検索に失敗したときは実行時例外となります。**
- **1行が8192文字を超える場合は、8193文字目以降は切り捨てられ8192文字になります。**

●「GetNext」メソッド

「GetNext」メソッドを使用して、「GetFirst」で検索した条件と一致する別のレコードを検索できます。

一致するレコードが存在する場合は該当行の全体(キーも含む)を返し、一致するレコードがないときはnilを返します。

■エラー情報(errorプロパティ)

エラー情報	説明
"ERR_FILE_DRIVENAME"	ドライブ名エラー
"ERR_FILE_FILENAME"	ファイル名エラー
"ERR_FILE_NONAME"	ファイル名を指定していない
"ERR_FILE_MODE"	該当モードではオープンできない
"ERR_FILE_NOT_EXIST"	ファイルは存在しない
"ERR_FILE_ALREADY_EXIST"	ファイルは既に存在している
"ERR_FILE_ALREADY_OPEN"	ファイルは既にオープンしている
"ERR_FILE_NOT_OPEN"	ファイルをオープンしていない
"ERR_FILE_FORMAT"	ファイルフォーマットエラー
"ERR_FILE_ENTRY_OVER"	ファイルエントリー数オーバー
"ERR_FILE_NOSPACE"	空き容量不足
"ERR_FILE_MAXOPEN"	ファイルオープン数オーバー
"ERR_FILE_SIZE"	ファイルサイズを超えるアドレスを指定した
"ERR_FILE_NOT_FOUND"	該当ファイルが見つからない
"ERR_FILEREC_OVERFLOW"	ファイルレコードが最大数を超えた
"ERR_SRCH_NOT_FOUND"	検索できなかった

使用例

"sample.mst"という置き換えマスタで、フィールド番号1に「foo」という文字列が含まれているデータを高速検索します。

```
Method main()
        record
        const key = 1
        const string = "foo"

        Begin

                Search:name = "2:sample.mst"                    // 置き換えマスタ指定

                // 検索キーと検索文字列を設定して、データを取得
                record = Search:GetFirst(key,string)

                While record
                        Handy:ShowMessageBox(record,"ok")
                        record = Search:GetNext()               // 次のデータを取得
                Wend
EndMethod
```

7-12 サーバとのデータ送受信

サーバ上のマスタデータ(データベース)を参照、ログデータを記録する要求を、サーバの「リクエストマネージャ」へ送信することができます。
BTシリーズ 無線LAN搭載機種には以下のコントロールがあり、これらがサーバの「リクエストマネージャ」と通信します。

- RemoteAccess(7-214ページ)
- RemoteDB(7-217ページ)
- RemoteFile(7-229ページ)
- RemoteUD(7-236ページ)

各コントロールの使用方法については、『通信ライブラリリファレンス(リアルタイムデータ通信編)』を参照してください。

ポイント

- **「RemoteAccess」、「RemoteDB」、「RemoteFile」、「RemoteUD」は、BTシリーズ 無線LAN搭載機種で「リクエストマネージャ」を使用したPCアプリケーションとの通信でのみ、使用できます。**
- **「RemoteDB」、「RemoteFile」、「RemoteUD」を使用する場合、「RemoteAccess：enable」をtrueに設定してください。**

RemoteAccess

無線通信ポートのオープン/クローズ、中間言語ファイルの更新などを実行します。

プロパティ

名称	説明	範囲	初期値	代入
enable※	無線ポートの状態を設定します。	true：無線ポートのオープン false：無線ポートのクローズ	false	可
fileVersion	次に更新される中間言語ファイルのバージョン	「リクエストマネージャ」より取得した値	-	不可
fileTitle	次に更新される中間言語ファイルのタイトル	「リクエストマネージャ」より取得した内容	-	不可
timeOut	通信タイムアウト時間を設定します。	3～300(秒)	7	可
timeOutFile	ファイル転送タイムアウトの時間を設定します。	3～300(秒)	30	可
syncTime	ポートオープン時(RemoteAccess:enable=true)に時刻同期を実施するかどうかを設定します。 ポートオープン前に設定してください。 BT-1000/1500シリーズのみ有効です。	true: 同期する false: 同期しない	true	可
error	エラー情報 エラー原因が格納されます。	文字列データ 「エラー情報(errorプロパティ)」(7-215ページ)	""	不可

※「RemoteAccess」、「RemoteDB」、「RemoteFile」、「RemoteUD」コントロールを使用するときは、trueに設定します。

メソッド

名称	説明	引数	戻り値
Initialize	各プロパティを初期化します。	なし	なし
UpdateApp※1	中間言語ファイルの更新 「無線システム管理ソフト」の「クイックアップデート機能」※2で設定した中間言語ファイルの更新を実行します。 BT-1000/1500シリーズのみ対応です。	なし	-

※1 「UpdateApp」メソッドの実行が完了すると、それ以降のプログラムファイルのステートメントに戻らないで、中間言語ファイルが新しいバージョンに更新されます。更新後、すべてのコントロールのプロパティ値などが初期化されます。実行結果が失敗の場合は、メソッドから戻ってきて次のステートメントが実行されます。

※2 「無線システム管理ソフト」の「クイックアップデート機能」については、『無線LAN環境設定マニュアル』を参照してください。

■エラー情報(errorプロパティ)

エラー情報	説明
"ERR_COM"	通信エラー
"ERR_REQUEST"	要求未送信
"ERR_RESPONSE"	応答未受信
"ERR_TIMEOUT"	タイムアウト
"ERR_REQID"	要求IDエラー
"ERR_RESID"	応答IDエラー
"ERR_FILENAME"	ファイル名エラー
"ERR_DRIVE"	ドライブエラー
"ERR_DATASIZE"	データサイズエラー
"ERR_APPLICATION"	「サーバアプリケーション」エラー
"ERR_TIMESET"	時刻合わせエラー
"ERR_OPEN"	RFオープンエラー
"ERR_OUTRANGE"	圏外エラー
"ERR_RESET"	通信設定エラー
"ERR_LOGINIT"	ログ初期化エラー
"ERR_DIRECTRY"	ディレクトリエラー
"ERR_LCSND"	下位ライブラリ送信エラー
"ERR_RCV_NODATA"	不正データ受信
"ERR_LC_RCVSTOP"	受信停止処理エラー
"ERR_LC_RCVFILESTOP"	ファイル受信停止処理エラー
"ERR_RANGE1"	パラメータエラー
"ERR_RANGE2"	パラメータエラー
"ERR_RANGE3"	パラメータエラー
"ERR_RANGE4"	パラメータエラー
"ERR_RANGE5"	パラメータエラー
"ERR_OTHER"	その他のエラー

使用例

クイックアップデート機能を使用して、中間言語ファイルを更新します。

```
Method main()
Begin
        If RemoteAccess:enable is false Then
                RemoteAccess:enable = true //ポートをオープンします
        EndIf

        //中間言語ファイルを更新します
        RemoteAccess:UpdateApp()

        // 更新が終了すると、新しい中間言語ファイルが起動するため
        // 以降のプログラム内容は実行されません
End Method
```

RemoteDB	サーバ上のマスタデータ(データベース)の検索、削除、更新などを「リクエストマネージャ」へ要求します。

プロパティ

名称	説明	範囲	初期値	代入
databaseName※1	アクセスするデータベース名	文字列(最大128バイト)	nil	可
tableName※1	アクセスするデータベースのテーブル名	文字列(最大128バイト)	nil	可
keyFieldNum※1	keydataの有効フィールド数	0～36	nil	可
newFieldNum※1	newdataの有効フィールド数	0～36	nil	可
resultFieldNum※1	resultdataの有効フィールド数	0～36	nil	不可
htFileName※1、※2	リクエスト種別「Select2(selectintofile)」の実行結果、またはExecSQLメソッドの実行結果を格納するファイル名	ドライブ番号:ファイル名(最大32バイト)	nil	可
result※1	「サーバアプリケーション」の応答結果	数値	nil	不可
index※3	インデックス番号	0～65535	0	可
operator※3	オペレータ演算子	0～255 📖「オプションデータの読み書き(SetOption、GetOptionメソッド)」(7-221ページ)	nil	可
data※3	データ	文字列(最大8192バイト)	nil	可
error	エラー情報 エラー原因が格納されます。	文字列データ 📖「エラー情報(errorプロパティ)」(7-222ページ)	""	不可

※1 プロパティの値は、「クライアントメモリ」に格納されます。
「クライアントメモリ」ついては、📖『通信ライブラリリファレンス(リアルタイムデータ通信編)』を参照してください。

※2 不正なファイル名を設定しても、実行時例外にならないので、注意して設定してください。

※3 「SetData」メソッドを実行すると「クライアントメモリ」に格納されます。
📖「サーバ上のデータベースへのアクセス要求(SetData、GetData、Execメソッド)」(7-219ページ)

メソッド

「RemoteAccess」コントロールの「enable」プロパティをtrueにしてから使用します。
「RemoteAccess」(7-214ページ)

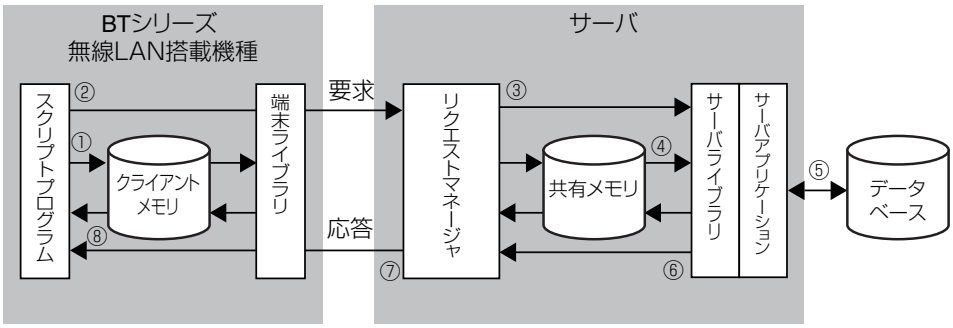
名称	説明	引数	戻り値
Initialize	各プロパティを初期化します。	なし	なし
SetData	BTの「クライアントメモリ」にプロパティ(index、operator、data)の値を格納します。 「サーバ上のデータベースへのアクセス要求(SetData、GetData、Execメソッド)」(7-219ページ)	kind:設定先の「クライアントメモリ」の要素("newdata"/"keydata") dataIndex:要素の配列番号(1～36)	true:正常終了 false:エラー終了
GetData	BTの「クライアントメモリ」から取得した情報をプロパティ(index、operator、data)に格納します。 「サーバ上のデータベースへのアクセス要求(SetData、GetData、Execメソッド)」(7-219ページ)	kind:取得元の「クライアントメモリ」の要素("newdata"/"keydata"/"result") dataIndex:要素の配列番号(1～36)	true:正常終了 false:エラー終了
Exec※1	データベースへのアクセスを要求します。 「サーバ上のデータベースへのアクセス要求(SetData、GetData、Execメソッド)」(7-219ページ)	requestID:リクエスト種別("select"/"count"/"select2"/"selectintofile"/"update"/"insert"/"delete"/"init"/"fin"/"fetch forward"/"fetch backward")	true:通信成功 false:通信失敗
SetOption※2	BTの「クライアントメモリ」へオプションデータを格納します。 「オプションデータの読み書き(SetOption、GetOptionメソッド)」(7-221ページ)	string:格納するデータ(最大8192バイト) optionIndex:要素の配列番号(1～3)	true:正常終了 false:エラー終了
GetOption※2	BTの「クライアントメモリ」からオプションデータを取得します。 「オプションデータの読み書き(SetOption、GetOptionメソッド)」(7-221ページ)	kind:「クライアントメモリ」の要素名("request"/"response") optionIndex:要素の配列番号(1～3)	読み込み文字列(最大8192バイト)
ExecSQL	引数sqlで記述したSQL文をサーバアプリケーションに送信します。実行結果はhtFileNameで指定したファイルに格納します。 「SQL文の送信(ExecSQLメソッド)」(7-221ページ)	sql:SQL文(最大8192バイト)	true:正常終了 false:エラー終了
Retry	通信失敗後に再送要求を出します。	なし	true:正常終了 false:エラー終了

※1 サーバの「リクエストマネージャ」に対して、データベースにアクセスする要求を発行します。「リクエストマネージャ」については、『通信ライブラリリファレンス(リアルタイムデータ通信編)』を参照してください。

※2 サーバとのデータ送受信コントロールの共通メソッドです。他のコントロールでも使用します。

■サーバ上のデータベースへのアクセス要求(SetData、GetData、Execメソッド)

BTからサーバ上のデータベースへアクセスする手順を説明します。



●データベースへのアクセスの要求手順(図の①～②)

①必要な情報を「クライアントメモリ」に格納

データベースのアクセスに必要な情報を、次の方法で「クライアントメモリ」に格納します。

「クライアントメモリ」については、『通信ライブラリリファレンス(リアルタイムデータ通信編)』を参照してください。

- **プロパティに値を設定**
  プロパティにデータベース名、テーブル名、検索条件などを設定します。
  このとき、プロパティの値は、「クライアントメモリ」に格納されます。
  ただし、「index」、「operator」、「data」のプロパティ値は、SetDataメソッドを実行するまで格納されません。
- **「SetData」メソッドを実行**
  「SetData」メソッドを実行して、「index」、「operator」、「data」のプロパティの値を「クライアントメモリ」へ格納します。
  「SetData」メソッドの使用方法については、「例1」(7-223ページ)を参照してください。

ポイント

クライアントメモリに格納できるのは、要求・応答のそれぞれ32KBまでです。

②「リクエストマネージャ」へ要求(「Exec」メソッド)

「Exec」メソッドを実行して、BTから「リクエストマネージャ」へデータベースへのアクセスの要求をします。

Exec の引数には以下の項目があり、サーバアプリケーションに対しては、それぞれの要求コマンドとして送信されます。

RemoteDB:Execの引数	要求コマンドID	
	Visual Basic/.NET	Visual C++
select	REQID_DB_SELECT	HTRQ_DB_SELECT
count	REQID_DB_COUNT	HTRQ_DB_COUNT
select2	REQID_DB_SELECT_INTO_FILE	HTRQ_DB_SELECT _INTO_FILE
selectintofile		
update	REQID_DB_UPDATE	HTRQ_DB_UPDATE
insert	REQID_DB_INSERT	HTRQ_DB_INSERT
delete	REQID_DB_DELETE	HTRQ_DB_DELETE
init	REQID_DB_INIT	HTRQ_DB_INIT
fin	REQID_DB_FIN	HTRQ_DB_FIN
fetch forward	REQID_DB_FETCH_F	HTRQ_DB_FETCH_F
fetch backward	REQID_DB_FETCH_B	HTRQ_DB_FETCH_B

●「リクエストマネージャ」と「サーバアプリケーション」の動作(図の③～⑦)

「リクエストマネージャ」、「共有メモリ」、「サーバアプリケーション」については、📖『通信ライブラリリファレンス(リアルタイムデータ通信編)』を参照してください。

「リクエストマネージャ」の動作

③受信した要求を「サーバアプリケーション」へ送信します。

このとき、受信した「クライアントメモリ」の情報をサーバ上の「共有メモリ」へ格納します。

「サーバアプリケーション」の動作

④端末アプリケーションが送信した要求コマンドは、イベントとしてサーバアプリケーションに通知されます。

「共有メモリ」の情報を読み込んで、要求された内容を確認します。

⑤要求された内容にしたがって、サーバーアプリケーションがデータベースにアクセスします。

例　データベースを検索して、必要な情報を取得

⑥応答(アクセス結果)を、「リクエストマネージャ」に返します。

このとき、データベースから取得した情報を「共有メモリ」に格納します。

「リクエストマネージャ」の動作

⑦「サーバアプリケーション」が応答を返したことをメソッドの戻り値に代入してBTへ返します。

このとき、「共有メモリ」の情報をBTの「クライアントメモリ」へ転送します。

●応答確認、要求情報の取得(図の⑧)

「Exec」メソッドを実行するときのリクエスト種別によって、次のような方法で応答の確認、要求情報の取得をします。

⑧「Exec」メソッドの戻り値で通信の成功を確認後、プロパティ値から必要な情報を取得します(「クライアントメモリ」の情報がプロパティに格納されています)。

「index」、「operator」、「data」のプロパティ値を取得する場合は、「GetData」メソッドを実行すると、「クライアントメモリ」の情報がプロパティに格納されます。

「Exec」メソッドの使用方法については、📖「例1」(7-223ページ)を参照してください。

■オプションデータの読み書き(SetOption、GetOptionメソッド)

BTの「クライアントメモリ」には、自由に使用できるオプションデータ領域があります。
「クライアントメモリ」については、『通信ライブラリリファレンス(リアルタイムデータ通信編)』を参照してください。

- 「SetOption」メソッドで、オプションデータを「クライアントメモリ」へ格納できます。
- 「GetOption」メソッドで、「クライアントメモリ」に格納したオプションデータを取得できます。

■SQL文の送信(ExecSQLメソッド)

ExecSQLメソッドを使用すると、引数sqlで記述したSQL文をサーバアプリケーションに送信できます。SQL文の実行結果は、Exec (select2)またはExec(selectintofile)と同様に、htFileNameプロパティで設定したファイルに格納されます。
詳細については、『通信ライブラリリファレンス(リアルタイムデータ通信編)』を参照してください。

ExecSQLの要求コマンドIDは以下のとおりです。

RemoteDB:ExecSQLの引数	要求コマンドID	
	Visual Basic/.NET	Visual C++
sql	REQID_DB_SQLDIRECT	HTRQ_DB_SQLDIRECT

■オペレータ演算子の読み書き(operatorプロパティ)

サーバアプリケーションで作成するSQL 文において、演算子を端末アプリケーションから指定する場合に設定します。
端末アプリケーションから転送されてきたoperatorの値をサーバアプリケーションが解析し、SQL文に付加します。
詳細については、『通信ライブラリリファレンス(リアルタイムデータ通信編)』を参照してください。

意味	値	備考
"="	0	一致、代入の意味
"NOT"	1	
">"	2	
">="	3	
"<"	4	
"<="	5	
"+"	6	
"+="	7	FIELD=FIELD+dataの意味 UPDATE文で使用
"-"	8	
"-="	9	FIELD=FIELD-dataの意味 UPDATE文で使用
"*"	10	
"/"	11	

■エラー情報(errorプロパティ)

エラー情報	説明
"ERR_RFSET"	通信設定エラー
"ERR_OPEN"	RFオープンエラー
"ERR_ALREADY_OPEN"	RFオープン中エラー
"ERR_LOGINIT"	ログ初期化エラー
"ERR_COM"	通信エラー
"ERR_REQUEST"	リクエスト処理エラー
"ERR_RESPONSE"	レスポンス処理エラー
"ERR_TIMEOUT"	タイムアウト
"ERR_FILE_PUT"	ファイル送信エラー
"ERR_FILE_GET"	ファイル受信エラー
"ERR_FILENAME"	ファイル名エラー
"ERR_UPDATE"	アプリ更新エラー
"ERR_RF_API_PARAM_NG"	RFAPIパラメータエラー
"ERR_RF_API_RF_OPEN_NG"	無線Openエラー
"ERR_RF_API_OUTRANGE"	圏外要因による送信失敗
"ERR_RF_API_SND_NG"	データ送信失敗
"ERR_RF_API_OTHER_ERR"	その他のRFAPIエラー
"ERR_FILE_DRIVENAME"	ドライブ名エラー
"ERR_FILE_NONAME"	ファイル名指定なし
"ERR_FILE_PUT_NGMODE"	ファイルオープン中で送信失敗
"ERR_FILE_NOT_EXIST"	ファイルは存在しない
"ERR_FILE_BIG"	空き容量に対してファイルが大きい
"ERR_RANGE_1"	パラメータエラー
"ERR_RANGE_2"	パラメータエラー
"ERR_RANGE_3"	パラメータエラー
"ERR_RANGE_4"	パラメータエラー
"ERR_RANGE_5"	パラメータエラー
"ERR_OTHER_ERR"	その他のエラー

使用例

例1　サーバ上のデータベースにアクセスして、条件に一致するデータを取得要求します。

```
Method main()
	menu1 = "Select¦Count¦Insert¦Delete¦Update¦SelectIntoFile¦Option"
	menu2 = nil
	menu3 = nil
	menu4 = nil
	ret = 1
Begin
	While RemoteAccess:enable is false
		// RemoteAccessコントロールのenableプロパティにtrueを設定
		RemoteAccess:enable = true
	Wend
	While( 1 )
		// データベースへのリクエスト種別を選択するメニューを表示
		ret = Handy:ShowMenu2( "TEST", menu1, menu2, menu3, menu4, ret)
		Select Case ret
		Case 1
			F_Select() // データ検索のメソッド呼び出し
		Case 2
			F_Count() // データ個数取得のメソッド呼び出し
		Case 3
			F_Insert() // データ挿入のメソッド呼び出し
		Case 4
			F_Delete() // データ削除のメソッド呼び出し
		Case 5
			F_Update() //データ更新のメソッド呼び出し
		Case 6
			F_SelectIntoFile() //
		Case 7
			F_Option()//SetOptionを用いたデータ挿入
		Case Else
		ret = 1
		End Select
	Wend
End Method

Method F_Select()
	index_list[ ]= {0}
	ope_list[ ] = {0}
	data_list[ ] = {123}
	i
Begin
	Screen:Clear()
	With RemoteDB
		// データベースの検索条件を、プロパティと「クライアントメモリ」に格納
		:databaseName = "DBNAME"
		:tableName = "log"
		:keyFieldNum = index_list.size
		:newFieldNum = 0

		// データベースの検索条件を、プロパティと「クライアントメモリ」に格納
		For i = 1 to :keyFieldNum
			:index = index_list[i-1] // プロパティに代入
```

```
                :operator = ope_list[i-1] // プロパティに代入
                :data = data_list[i-1] // プロパティに代入
                :SetData("keydata", i) //「クライアントメモリ」に格納
        Next

        //「リクエストマネージャ」へデータベースの検索を要求して、結果を確認
        :Exec("select")
        Handy:ShowMessageBox("error="&:error, "confirm")
        Handy:ShowMessageBox(:resultFieldNum, "confirm")

        // 検索された個数分のデータを取得
        For i = 1 to :resultFieldNum
                :GetData("result", i) // 検索したデータをプロパティに格納
                With Screen
                        :posx = 1 :posy = i*2+1
                        // 検索データをプロパティから取得して表示
                        :OutputText(RemoteDB:index & ":" & RemoteDB:data)
                EndWith
        Next
        Handy:KeyWait()
    EndWith
End Method
```

例2 サーバ上のデータベースにアクセスして、指定する条件に一致するデータの個数を数えます。

```
Method F_Count()
    index_list[ ] = {0}
    ope_list[ ] = {0}
    data_list[ ] = {123}
    i
Begin
    Screen:Clear()
    With RemoteDB
        // データベースの検索条件を、プロパティと「クライアントメモリ」に格納
        :databaseName = "DBNAME"
        :tableName = "log"
        :keyFieldNum = index_list.size
        :newFieldNum = 0

        // データベースの検索条件を、プロパティと「クライアントメモリ」に格納
        For i = 1 to :keyFieldNum
                :index = index_list[i-1] // プロパティに代入
                :operator = ope_list[i-1] // プロパティに代入
                :data = data_list[i-1] // プロパティに代入
                :SetData("keydata", i) //「クライアントメモリ」に格納
        Next

        //「 リクエストマネージャ」へデータベースの検索を要求して、結果を確認
        :Exec("count")
        Handy:ShowMessageBox("error="&:error, "confirm")
        Handy:ShowMessageBox("count="&:result, "confirm")
    EndWith
End Method
```

例3 サーバ上のデータベースにアクセスして、データを挿入します。

```
Method F_Insert()
	index_list[ ] = {0,1,2}
	ope_list[ ] = {0,0,0}
	data_list[ ] = {123,456,789}
	i
Begin
	Screen:Clear()
	With RemoteDB
		// データベースの検索条件を、プロパティと「クライアントメモリ」に格納
		:databaseName = "DBNAME"
		:tableName = "log"
		:keyFieldNum = 0
		:newFieldNum = index_list.size

		// データベースの検索条件を、プロパティと「クライアントメモリ」に格納
		For i = 1 to :newFieldNum
			:index = index_list[i-1] // プロパティに代入
			:operator = ope_list[i-1] // プロパティに代入
			:data = data_list[i-1] // プロパティに代入
			:SetData("newdata", i) //「クライアントメモリ」に格納
		Next

		//「リクエストマネージャ」へデータベースにデータの挿入を要求して、結果を確認
		:Exec("insert")
		Handy:ShowMessageBox("error="&:error, "confirm")
	EndWith
End Method
```

例4 サーバ上のデータベースにアクセスして、指定する条件に一致するデータを削除します。

```
Method F_Delete()
	index_list[ ] = {0}
	ope_list[ ] = {0}
	data_list[ ] = {123}
	i
Begin
	Screen:Clear()
	With RemoteDB
		// データベースの検索条件を、プロパティと「クライアントメモリ」に格納
		:databaseName = "DBNAME"
		:tableName = "log"
		:keyFieldNum = index_list.size
		:newFieldNum = 0

		// データベースの検索条件を、プロパティと「クライアントメモリ」に格納
		For i = 1 to :keyFieldNum
			:index = index_list[i-1] // プロパティに代入
			:operator = ope_list[i-1] // プロパティに代入
			:data = data_list[i-1] // プロパティに代入
			:SetData("keydata", i) //「クライアントメモリ」に格納
		Next

		//「リクエストマネージャ」へデータベースにデータの削除を要求して、結果を確認
		:Exec("delete")
		Handy:ShowMessageBox("error="&:error, "confirm")
	EndWith
End Method
```

例5 「ExecSQL」、「Retry」メソッドの使用例

```
Method ExecSQLSample()
        ret
Begin
        RemoteAccess:enable = True
        With RemoteDB
                :Initialize()
                :htFileName = "2:dl.csv"
                ret = :ExecSQL("SELECT * FROM master")
                // 上記のSQLは下記の実行と同じ結果を得る
                // :tableName = "master"
                // :Exec("select2")
                While ret is False
                If Handy:ShowMessageBox("再送しますか？","yesno¦yes") is False Then
                   Wbreak
                EndIf
                // 通信に失敗したら再送する
                ret = :Retry()
                Wend
        End With
End Method
```

例6 サーバ上のデータベースにアクセスして、指定する条件に一致するデータを更新します。

```
Method F_Update()
        index_key_list[ ]= {0}
        ope_key_list[ ] = {0}
        keydata_list[] = {123}
        index_new_list[ ]= {0,1,2}
        ope_new_list[ ] = {0,0,0}
        newdata_list[] = {777,777,777}
        i
Begin
        RemoteAccess:enable =true
        Screen:Clear()
        With RemoteDB
                // データベースの検索条件を、プロパティと「クライアントメモリ」に格納
                :databaseName = "DBNAME"
                :tableName = "log"
                :keyFieldNum = index_key_list.size
                :newFieldNum = index_new_list.size

                // データベースの検索条件を、プロパティと「クライアントメモリ」に格納
                For i = 1 to :keyFieldNum
                        :index = index_key_list[i-1] // プロパティに代入
                        :operator = ope_key_list[i-1] // プロパティに代入
                        :data = keydata_list[i-1] // プロパティに代入
                        :SetData("keydata", i) //「 クライアントメモリ」に格納
                Next

                //「リクエストマネージャ」へデータベースの検索を要求して、結果を確認
                For i = 1 to :newFieldNum
                        :index = index_new_list[i-1] // プロパティに代入
                        :operator = ope_new_list[i-1] // プロパティに代入
                        :data = newdata_list[i-1] // プロパティに代入
                        :SetData("newdata", i) //「クライアントメモリ」に格納
                Next

                //「リクエストマネージャ」へデータベースにデータの更新を要求して、結果を確認
                :Exec("update")
                Handy:ShowMessageBox("error="&:error, "confirm")
        EndWith
EndMethod
```

例7 サーバ上のデータベースにアクセスして、指定する条件に一致するデータを検索し、ヒットするレコードをcsvファイルでハンディ上に取得します。

```
Method F_SelectIntoFile()
        index_list[ ]= {0}
        ope_list[ ] = {0}
        data_list[] = {123}
        i
Begin
        RemoteAccess:enable =true
        With RemoteDB
        // データベースの検索条件を、プロパティと「クライアントメモリ」に格納
        :databaseName = "DBNAME"
        :tableName = "log"
        :htFileName = "2:dl.csv"
        :keyFieldNum = index_list.size
        :newFieldNum = 0

        // データベースの検索条件を、プロパティと「クライアントメモリ」に格納
        For i = 1 to :keyFieldNum
                :index = index_list[i-1] // プロパティに代入
                :operator = ope_list[i-1] // プロパティに代入
                :data = data_list[i-1] // プロパティに代入
                :SetData("keydata", i) //「クライアントメモリ」に格納
        Next

        //「リクエストマネージャ」へデータベースにデータの検索を要求して、結果を確認
:Exec("selectintofile")
        Handy:ShowMessageBox("error="&:error, "confirm")
        EndWith
EndMethod
```

例8 オプション領域の使用例

```
Method F_Option()
        index_list[ ] = {0,1,2}
        ope_list[ ] = {0,0,0}
        data_list[ ] = {123,456,789}
        i
        store[ ] = {"","",""}
Begin
        Screen:Clear ()
        With RemoteDB
        // データベースの検索条件を、プロパティと「クライアントメモリ」に格納
        :databaseName = "DBNAME"
        :tableName = "log"
        :keyFieldNum = 0
        :newFieldNum = index_list.size

        // データベースの検索条件を、プロパティと「クライアントメモリ」に格納
        For i = 1 to :newFieldNum
                :index = index_list[i-1] // プロパティに代入
                :operator = ope_list[i-1] // プロパティに代入
                :SetOption(data_list[i-1],i) //「 クライアントメモリ」に格納
        Next
```

```
        //「リクエストマネージャ」へデータベースにデータの挿入を要求して、結果を確認
        :Exec("insert")
        Handy:ShowMessageBox("error="&:error, "confirm")

        //クライアントデータからオプションデータを取得する
        For i = 1 to 3
                store[i-1] = :GetOption("response", i)
                Handy:ShowMessageBox(store[i-1],"confirm")
        Next
        EndWith
EndMethod
```

RemoteFile	サーバ上のファイルの読み込み、書き込みを「リクエストマネージャ」へ要求します。

BT内のファイルの読み込み、書き込みについては、
「File」(7-166ページ)、または「FileStream」(7-160ページ)を参照してください。

ポイント

このコントロールの「GetFile」、「PutFile」メソッドで操作するファイルを、既に「File」「Master」「Search」コントロールで操作している場合は、メソッドを実行する前に使用したコントロールで初期化(Initialize)してください。

プロパティ

名称	説明	範囲	初期値	代入
path	サーバのファイルのパス名	文字列(最大126バイト)	nil	可
name	サーバのファイル名	文字列(最大126バイト)	nil	可
result	「サーバアプリケーション」の応答結果	数値	nil	不可
currentPos	サーバ上のファイルのシークポインタ 「シークポインタ(currentPosプロパティ)」(7-162ページ)	0～(バイト)	0	可
error	エラー情報 エラー原因が格納されます。	文字列データ 「エラー情報(errorプロパティ)」(7-232ページ)	""	不可

※ プロパティの値は、「クライアントメモリ」に格納されます。
「クライアントメモリ」については、『通信ライブラリリファレンス(リアルタイムデータ通信編)』を参照してください。

メソッド

「RemoteAccess」コントロールの「enable」プロパティをtrueにしてから使用します。
「RemoteAccess」(7-214ページ)

名称	説明	引数	戻り値
Initialize	各プロパティを初期化します。	なし	なし
Gets※1	サーバ上のファイルからテキストデータを読み込む要求をします。 「サーバ上のファイルへのアクセス要求(Gets、Puts、Read、Writeメソッド)」(7-231ページ)	maxLength:最大読み込みバイト数 (1～8192)	読み込み文字列 nil
Puts※1	サーバ上のファイルにテキストデータを書き込む要求をします。 「サーバ上のファイルへのアクセス要求(Gets、Puts、Read、Writeメソッド)」(7-231ページ)	string:書き込む文字列 mode:書き込みモード ("write"/"append"/"create")※2	true:正常終了 false:エラー終了
Read※1	サーバ上のファイルからバイナリデータを読み込む要求をします。 「サーバ上のファイルへのアクセス要求(Gets、Puts、Read、Writeメソッド)」(7-231ページ)	maxLength:最大読み込みバイト数 (1～4096)	読み込み文字列 nil 「バイナリ表現の特殊形式」(7-168ページ)

名称	説明	引数	戻り値
Write※1	サーバ上のファイルにバイナリデータを書き込む要求をします。 📖「サーバ上のファイルへのアクセス要求(Gets、Puts、Read、Writeメソッド)」(7-231ページ)	string:書き込む文字列(バイナリ表現の特殊形式) mode:書き込みモード("write"/"append"/"create")※2	true:正常終了 false:エラー終了
GetFile※3	サーバからファイルを受信します。	remoteFileName: 受信元(サーバ)のファイル名 localFileName: 受信先(BT)のファイル名(ドライブ番号:ファイル名)	true:正常終了 false:エラー終了
PutFile※4	サーバへファイルを送信します。	localFileName: 送信元(BT)のファイル名(ドライブ番号:ファイル名) remoteFileName: 送信先(サーバ)のファイル名	true:正常終了 false:エラー終了
SetOption※5	BTの「クライアントメモリ」へオプションデータを格納します。 📖「オプションデータの読み書き(SetOption、GetOptionメソッド)」(7-221ページ)	string:格納するデータ(最大8192バイト) optionIndex:要素の配列番号(1〜3)	true:正常終了 false:エラー終了
GetOption※5	BTの「クライアントメモリ」からオプションデータを取得します。 📖「オプションデータの読み書き(SetOption、GetOptionメソッド)」(7-221ページ)	kind:「クライアントメモリ」の要素名("request"/"response") optionIndex:要素の配列番号(1〜3)	読み込み文字列(最大8192バイト)
Retry	通信失敗後に再送要求を出します。	なし	true:正常終了 false:エラー終了

※1 サーバの「リクエストマネージャ」に対して、ファイルへのアクセス要求を発行します。
ファイルへ実際にアクセスするのは「サーバアプリケーション」です。
「リクエストマネージャ」と「サーバアプリケーション」については、📖『通信ライブラリリファレンス(リアルタイムデータ通信編)』を参照してください。

※2 ベースアプリケーションでは、以下の動作になります。
"write" :「currentPos」プロパティで指定する位置から上書きします。
"append" :「currentPos」のプロパティ値にかかわらずシークポインタをファイル終端に移動して、その位置から追加書き込みをします。
"create" :既にあるファイルを削除して、「currentPos」のプロパティ値をファイル先頭位置に設定します。

※3 サーバから直接ファイルを受信します。
「path」プロパティで設定したフォルダからファイルを受信します。

※4 サーバへ直接ファイルを送信します。
「path」プロパティで設定したフォルダ直下に自動的に作成される IP アドレスと同名のフォルダへファイルを送信します。
ドライブ3 のファイルを送信した場合は、上記フォルダに「drv3」というフォルダが作成され、そのフォルダ内にファイルが送信されます。この場合、送信元のファイル名から変更できません。

※5 サーバとのデータ送受信コントロールの共通メソッドです。他のコントロールでも使用します。

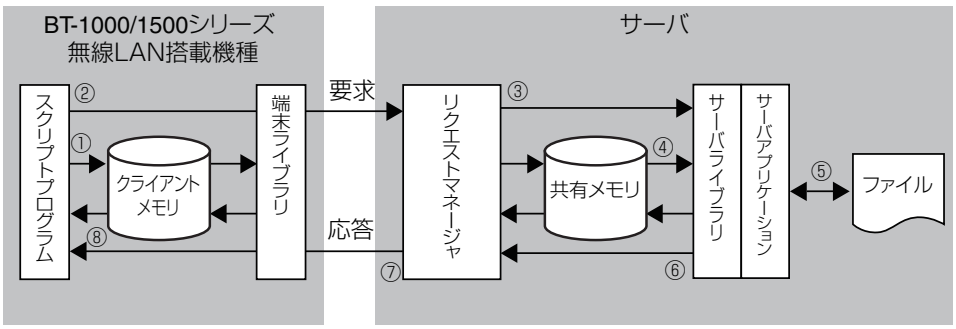
■サーバ上のファイルへのアクセス要求(Gets、Puts、Read、Writeメソッド)

「Gets」、「Puts」メソッドはASCII文字や漢字を扱う場合に適しています。

- 改行、TABなど基本的な制御コードを含むことは問題ありません。
- NULL(0x00)を含んだデータは扱えません。

「Read」、「Write」メソッドは、NULL(0x00)を含むバイナリデータを扱う場合に適しています。
バイナリデータについては、📖「バイナリ表現の特殊形式」(7-168ページ)を参照してください。

次に、「Gets」、「Puts」、「Read」、「Write」メソッドを使ってBTからサーバ上のファイルへアクセスする手順を説明します。



●サーバ上のファイルへのアクセス要求手順(図の①～②)

①必要な情報を「クライアントメモリ」に格納

アクセスするファイルのパス名、ファイル名、読み込み/書き込み位置などをプロパティに設定します。
このとき、「クライアントメモリ」に同じ情報が格納されます。
「クライアントメモリ」については、📖『通信ライブラリリファレンス(リアルタイムデータ通信編)』を参照してください。

②「リクエストマネージャ」へ読み込み、書き込み要求

「Gets」、「Puts」、「Read」、「Write」メソッドのどれかを実行して、BTから「リクエストマネージャ」へファイルのアクセスを要求します。
このとき、「クライアントメモリ」の情報が「リクエストマネージャ」へ転送されます。

●「リクエストマネージャ」と「サーバアプリケーション」の動作(図の③～⑦)

「リクエストマネージャ」、「共有メモリ」、「サーバアプリケーション」については、📖『通信ライブラリリファレンス(リアルタイムデータ通信編)』を参照してください。

「リクエストマネージャ」の動作

③受信した要求を「サーバアプリケーション」へ送信します。

このとき、「クライアントメモリ」のデータをサーバ上の「共有メモリ」へ格納します。

「サーバアプリケーション」の動作

④「共有メモリ」の情報を読み込んで、要求された内容を確認します。

⑤要求された内容にしたがって、ファイルにアクセスします。

⑥応答(アクセス結果)を「リクエストマネージャ」に返します。

必要な情報を「共有メモリ」に格納します。

「リクエストマネージャ」の動作

⑦「サーバアプリケーション」の応答（アクセス結果）をメソッドの戻り値に格納してBTに返します。

このとき、「共有メモリ」の情報をBTの「クライアントメモリ」へ転送します。

●応答確認、要求情報の取得

⑧実行したメソッドによって、次のような方法で応答の確認、要求情報の取得をします。

「Puts」、「Write」メソッドを実行したときは、「サーバアプリケーション」の応答を、メソッドの戻り値で確認します。「Gets」、「Read」メソッドを実行したときは、メソッドの戻り値から要求情報を取得します。

また、エラー情報などをプロパティ値から取得します（「クライアントメモリ」の情報がプロパティに格納されています）。

■エラー情報（errorプロパティ）

エラー情報	説明
"ERR_RFSET"	通信設定エラー
"ERR_OPEN"	RFオープンエラー
"ERR_ALREADY_OPEN"	RFオープン中エラー
"ERR_LOGINIT"	ログ初期化エラー
"ERR_COM"	通信エラー
"ERR_REQUEST"	リクエスト処理エラー
"ERR_RESPONSE"	レスポンス処理エラー
"ERR_TIMEOUT"	タイムアウト
"ERR_FILE_PUT"	ファイル送信エラー
"ERR_FILE_GET"	ファイル受信エラー
"ERR_FILENAME"	ファイル名エラー
"ERR_UPDATE"	アプリ更新エラー
"ERR_RF_API_PARAM_NG"	RFAPIパラメータエラー
"ERR_RF_API_RF_OPEN_NG"	無線Openエラー
"ERR_RF_API_OUTRANGE"	圏外要因による送信失敗
"ERR_RF_API_SND_NG"	データ送信失敗
"ERR_RF_API_OTHER_ERR"	その他のRFAPIエラー
"ERR_FILE_DRIVENAME"	ドライブ名エラー
"ERR_FILE_NONAME"	ファイル名指定なし
"ERR_FILE_PUT_NGMODE"	ファイルオープン中で送信失敗
"ERR_FILE_NOT_EXIST"	ファイルは存在しない
"ERR_FILE_BIG"	空き容量に対してファイルが大きい
"ERR_RANGE_1"	パラメータエラー
"ERR_RANGE_2"	パラメータエラー
"ERR_RANGE_3"	パラメータエラー
"ERR_RANGE_4"	パラメータエラー
"ERR_RANGE_5"	パラメータエラー
"ERR_OTHER_ERR"	その他のエラー

使用例

例1 「RemoteFile」コントロールのプロパティを初期化して、サーバ上のファイルへアクセスするメソッドを呼び出します。

```
Method main() // メインメソッド
	menu1 = "TXT¦BIN¦GETFILE¦PUTFILE"
	menu2 = "OPTION"
	menu3 = nil
	menu4 = nil
	ret = 1
Begin
	While RemoteAccess:enable is False
		Handy:ShowMessageBox("無線ポートをオープンします", "confirm")
		// RemoteAccessコントロールのenableプロパティにtrueを設定
		RemoteAccess:enable = true
	Wend
	// RemoteFileコントロールのプロパティ値を初期化
	RemoteFile:Initialize()
	While(1)
		// サーバ上のファイルにアクセスする種類を選択するメニューを表示
		ret = Handy:ShowMenu2("RemoteFile", menu1, menu2, menu3, menu4, ret)
		Select Case ret
		Case 1
			Sample1() // サーバ上のファイルにアクセスするメソッドの呼び出し
		Case 2
			Sample2() // サーバ上のファイルにアクセスするメソッドの呼び出し
		Case 3
		Sample3() // サーバ上にファイルを送信するメソッドの呼び出し
		Case 4
			Sample4() // サーバ上のファイルを受信するメソッドの呼び出し
		Case 5
			Sample5() // クライアントメモリのオプションデータを読み書きするメソッド
		Case Else
		ret = 1
		End Select
	Wend
End Method
```

例2　サーバ上の新規ファイルにテキストデータを送信します。

```
Method Sample1()
        dat
Begin
        With RemoteFile
                :path = "C:\\temp\\"
                :name = "SAMPLE.TXT"
                // ファイルを新規に作成して"01234567890"と書き込みます
                :Puts("01234567890", "create")
                // シークポインタを3バイト目に移動して"ABC"と書き込みます
                :currentPos = 3
                :Puts("ABC", "write")
                // ファイルの終端に"XYZ"と書き込みます
                :Puts("XYZ", "append")
                // ファイルの先頭から5バイト読み込みます
                :currentPos = 0
                dat = :Gets(5) // datに"012AB"が代入されます
                Handy:DebugOut(dat & "\n")
                // ファイルの8バイト目から6バイト読み込みます
                :currentPos = 8
                dat = :Gets(6) // datに"890XYZ"が代入されます
                Handy:DebugOut(dat & "\n")
        EndWith
End Method
```

例3　サーバ上の新規ファイルにバイナリデータを送信します。

```
Method Sample2()
        dat
Begin
        With RemoteFile
        :path = "C:\\temp\\"
        :name = "SAMPLE.BIN"
        // ファイルを新規に作成し{0x01,0x23,0x45,0x67,0x89}と書き込みます
        :currentPos = 0
        :Write("0123456789", "create")
        // シークポインタを3バイト目に移動して0xABと書き込みます
        :currentPos = 3
        :Write("AB", "write")// ファイルの終端に0xEFと書き込みます
        :Write("EF", "append")
        // ファイルの先頭から5バイト読み込みます
        :currentPos = 0
        dat = :Read(5) // datに"012345AB89"が代入
        されます
        Handy:DebugOut(dat&"\n")
        // ファイルの4バイト目から2バイト読み込みます
        :currentPos = 4
        dat = :Read(2) // datに"89EF"が代入されます
        Handy:DebugOut(dat&"\n")
        EndWith
End Method
```

例4　サーバ上のファイルを受信します。

```
Method Sample3()
Begin
        With RemoteFile
                :path = "C:\\temp\\"
                // サーバPCのC:\\temp内のsample.txtファイルを
                // sample_bt.txtとしてハンディのドライブ2に受信します
                :GetFile("sample.txt", "2:sample.txt")
        EndWith
End Method
```

例5　サーバ上にファイルを送信します。

```
Method Sample4()
Begin
        With RemoteFile
                :path = "C:\\temp\\"
                // ハンディのドライブ2にあるsample_bt.txtを
                // サーバPCのC:\\temp内にsample.txtファイルとして送信します
                :PutFile("2:sample.txt", "sample.txt")
        EndWith
End Method
```

例6　オプション領域のデータの格納･取得を行います。

```
Method Sample5()
        dat
Begin
        With RemoteFile
                :path = "C:\\temp\\"
                // リクエスト構造体の第1オプション領域に"ABC"を格納します
                :SetOption("ABC", 1)
                // リクエスト構造体の第1オプション領域のデータを取得します
                dat = :GetOption("request",1)
                Handy:ShowMessageBox (dat,"confirm")
                // ファイル送信を行い、リクエストマネージャからの応答を取得します
                :PutFile("2:sample_bt.txt", "sample.txt")
                // レスポンス構造体の第1オプション領域のデータを取得します
                dat = :GetOption("response",1)
                Handy:ShowMessageBox(dat,"confirm")
        End With
End Method
```

RemoteUD	あらかじめメッセージを書き込んだファイルをサーバに送信します。

ポイント

このコントロールで操作するファイルを、既に「File」「Master」「Search」コントロールで操作している場合は、メソッドを実行する前に使用したコントロールで初期化(Initialize)してください。

プロパティ

名称	説明	範囲	初期値	代入
type	ユーザID	数値(0～2147483647)	nil	可
pcPath	サーバのパス名	文字列(最大126バイト)	nil	可
pcFileName	サーバのファイル名	文字列(最大126バイト)	nil	可
htFileName※	BTのファイル名	ドライブ番号:ファイル名	nil	可
result	「サーバアプリケーション」の応答結果	数値	nil	不可
error	エラー情報 エラー原因が格納されます。	文字列データ 「エラー情報(errorプロパティ)」(7-238ページ)	""	不可

※ サーバへ送信するメッセージをファイルに書き込んでおきます。不正なファイル名を設定しても、実行時例外にならないので、注意してください。

メソッド

「RemoteAccess」コントロールの「enable」プロパティをtrueにしてから使用します。
「RemoteAccess」(7-214ページ)

名称	説明	引数	戻り値
Initialize	各プロパティを初期化します。	なし	なし
Exec	サーバへユーザデータを送信します。 「サーバへユーザデータの送信要求(Execメソッド)」(7-237ページ)	なし	true:送信成功 false:送信失敗
SetOption※2	BTの「クライアントメモリ」※1にオプションデータを格納します。 「オプションデータの読み書き(SetOption、GetOptionメソッド)」(7-221ページ)	string:格納するデータ(最大8192バイト) optionIndex:要素の配列番号(1～3)	true:正常終了 false:エラー終了
GetOption※2	BTの「クライアントメモリ」からオプションデータを取得します。 「オプションデータの読み書き(SetOption、GetOptionメソッド)」(7-221ページ)	kind:「クライアントメモリ」の要素名("request"/"response") optionIndex:要素の配列番号(1～3)	読み込み文字列(最大8192バイト)
Retry	通信失敗後に再送要求を出します。	なし	true:正常終了 false:エラー終了

※1 「クライアントメモリ」については、『通信ライブラリリファレンス(リアルタイムデータ通信編)』を参照してください。
※2 サーバとのデータ送受信コントロールの共通メソッドです。他のコントロールでも使用します。

■サーバへユーザデータの送信要求(Execメソッド)

「Exec」メソッドを実行すると、「htFileName」プロパティで指定するファイルを、サーバの「リクエストマネージャ」を経由して、サーバ上のフォルダへ送信します。
「リクエストマネージャ」については、『通信ライブラリリファレンス(リアルタイムデータ通信編)』を参照してください。

- ファイル受信先のサーバ上のパス名とファイル名は、「pcPath」プロパティと「pcFilename」で指定できます。
  「pcPath」の下に、BTのIPアドレスと同名のフォルダを作成して、そこにファイルを格納します。
- 「サーバアプリケーション」の実行結果は、「result」プロパティに格納されます。

■エラー情報(errorプロパティ)

エラー情報	説明
"ERR_RFSET"	通信設定エラー
"ERR_OPEN"	RFオープンエラー
"ERR_ALREADY_OPEN"	RFオープン中エラー
"ERR_LOGINIT"	ログ初期化エラー
"ERR_COM"	通信エラー
"ERR_REQUEST"	リクエスト処理エラー
"ERR_RESPONSE"	レスポンス処理エラー
"ERR_TIMEOUT"	タイムアウト
"ERR_FILE_PUT"	ファイル送信エラー
"ERR_FILE_GET"	ファイル受信エラー
"ERR_FILENAME"	ファイル名エラー
"ERR_UPDATE"	アプリ更新エラー
"ERR_RF_API_PARAM_NG"	RFAPIパラメータエラー
"ERR_RF_API_RF_OPEN_NG"	無線Openエラー
"ERR_RF_API_OUTRANGE"	圏外要因による送信失敗
"ERR_RF_API_SND_NG"	データ送信失敗
"ERR_RF_API_OTHER_ERR"	その他のRFAPIエラー
"ERR_FILE_DRIVENAME"	ドライブ名エラー
"ERR_FILE_NONAME"	ファイル名指定なし
"ERR_FILE_PUT_NGMODE"	ファイルオープン中で送信失敗
"ERR_FILE_NOT_EXIST"	ファイルは存在しない
"ERR_FILE_BIG"	空き容量に対してファイルが大きい
"ERR_RANGE_1"	パラメータエラー
"ERR_RANGE_2"	パラメータエラー
"ERR_RANGE_3"	パラメータエラー
"ERR_RANGE_4"	パラメータエラー
"ERR_RANGE_5"	パラメータエラー
"ERR_OTHER_ERR"	その他のエラー

使用例

例1 「RemoteUD」コントロールのプロパティを初期化して、サーバ上のファイルへアクセスするメソッドを呼び出します。

```
Method main() // メインメソッド
        menu1 = "EXEC¦OPTION"
        menu2 = nil
        menu3 = nil
        menu4 = nil
        ret = 1
Begin
        While RemoteAccess:enable is false
                Handy:ShowMessageBox("無線ポートをオープンします", "confirm")
                // RemoteAccessコントロールのenableプロパティにtrueを設定
                RemoteAccess:enable = true
        Wend
        // RemoteUDコントロールのプロパティ値を初期化
        RemoteUD:Initialize()
        While(1)
                // サーバ上のファイルにアクセスする種類を選択するメニューを表示
                ret = Handy:ShowMenu2
```

```
            ("RemoteUD", menu1, menu2, menu3, menu4, ret)
            Select Case ret
            Case 1
                Sample1() // サーバにファイル送信するメソッドの呼び出し
            Case 2
                Sample2() // クライアントメモリのオプションデータを読み書きするメソッド
            Case Else
            ret = 1
            End Select
        Wend
End Method
```

例2　サーバ上にファイルを送信します。

```
Method Sample1()
Begin
        //ハンディ内のファイルをサーバPCに送信します
        With RemoteUD
            :type = 12345 //ユーザIDを設定します
            :pcPath = "C:\\temp\\"
            :pcFileName = "sample.txt"
            :htFileName = "2:sample.txt"
            :Exec()
        End With
End Method
```

例3　オプション領域のデータの格納･取得を行います。

```
Method Sample2()
        dat
Begin
        With RemoteUD
            :type = 12345 //ユーザIDを設定します
            :pcPath = "C:\\temp\\"
            :pcFileName = "sample.txt"
            :htFileName = "2:sample_bt.txt"
            // リクエスト構造体の第１オプション領域に"ABC"を格納します
            :SetOption("ABC", 1)
            // リクエスト構造体の第１オプション領域のデータを取得します
            dat = :GetOption("request",1)
            Handy:ShowMessageBox (dat,"confirm")
            // ファイル送信を行い、リクエストマネージャからの応答を取得します
            :Exec()
            // レスポンス構造体の第１オプション領域のデータを取得します
            dat = :GetOption("response",1)
            Handy:ShowMessageBox(dat,"confirm")
        End With
End Method
```

7-13 サーバPCやプリンタとの通信

無線、赤外線通信(IrDA)ポート、RS-232Cポート、USBポートを使って、BTシリーズどうしの通信、BTシリーズをサーバPCやプリンタなどと通信するための設定をおこないます。

WLAN	BTシリーズ無線LAN搭載機種の無線通信の動作を設定します。

プロパティ

名称	説明	範囲	初期値	代入
isOpen※1	無線ポートの状態を確認します。 「ポートの状態(isOpenプロパティ)」(7-242ページ)	false:無線ポートのクローズ(使用不可) true:無線ポートのオープン(使用可能)	false	不可
essId※2	SSID	文字列(0～32バイト)	BTシリーズの設定値	可
defaultKey	デフォルトで使用されるWEPキーの番号	1～4	1	可
wepKey1	WEPキー(WEP1)	文字列(5バイト/13バイト) nil:指定しない	BTシリーズの設定値	可
wepKey2	WEPキー(WEP2)	文字列(5バイト/13バイト) nil:指定しない	BTシリーズの設定値	可
wepKey3	WEPキー(WEP3)	文字列(5バイト/13バイト) nil:指定しない	BTシリーズの設定値	可
wepKey4	WEPキー(WEP4)	文字列(5バイト/13バイト) nil:指定しない	BTシリーズの設定値	可
macaddr	BTシリーズのMACアドレス	文字列(12桁固定)	固有値	不可
ssid_adhoc	アドホックモード時のSSID	文字列(0～32バイト)	BTシリーズの設定値	可
defaultKey_adhoc	アドホックモード時にデフォルトで使用されるWEPキーの番号	1～4	1	可
wepKey1_adhoc	アドホックモード時のWEPキー(WEP1)	文字列(5バイト/13バイト) nil:指定しない	BTシリーズの設定値	可
channel	アドホックモード時のチャネル	1～14	BTシリーズの設定値	可
wpaMode	WPA/WPA2使用時の認証方式を設定します。	"PSK":WPA-PSK "PEAP":WPA-PEAP "EAP_TLS":WPA-EAP_TLS "PSK2":WPA2-PSK "PEAP2":WPA2-PEAP "EAP_TLS2":WPA2-EAP_TLS nil:WEP	BTシリーズの設定値	可
cipher	WPA/WPA2使用時の暗号化方式を設定します。	"TKIP" "AES"	BTシリーズの設定値	可
networkMod	インフラストラクチャ／アドホックモードを切り替えます。	"infrastructure": インフラストラクチャモード "adhoc": アドホックモード	"infrastructure"	可

名称	説明	範囲	初期値	代入
infra802d11Mode	インフラストラクチャの無線モードを指定します。 3はBT-W100/W155/W80/W85/W85T/W70/W75/1010W/1010WB/1550WBのみ有効です。	1:802.11b only 2:802.11bgn(2.4G) 3:802.11an(5G)	2	可
infraChannel※3	インフラ無線チャネル範囲(2.4GHz帯)を指定します。 14chはBT-W100/W155/W80/W85/W85T/W70/W75/1010W/1010WB/1550WBのみ有効です。	有効にするチャネルの数字の和を代入します。 〈0～16383〉 0 :全チャネル有効 1 : 1ch 2 : 2ch 4 : 3ch 8 : 4ch 16 : 5ch 32 : 6ch 64 : 7ch 128 : 8ch 256 : 9ch 512 :10ch 1024 :11ch 2048 :12ch 4096 :13ch 8192 :14ch	0	可
infraChannel5G※3	インフラ無線チャネル範囲(5GHz帯)を指定します。 BT-W100/W155/W80/W85/W85T/W70/W75/1010W/1010WB/1550WBのみ有効です。	有効にするチャネルの数字の和を代入します。 〈0～524287〉 --- W52---------- 1 :36ch 2 :40ch 4 :44ch 8 :48ch --- W53---------- 16 :52ch 32 :56ch 64 :60ch 128 :64ch --- W56---------- 256 :100ch 512 :104ch 1024 :108ch 2048 :112ch 4096 :116ch 8192 :120ch 16384 :124ch 32768 :128ch 65536 :132ch 131072 :136ch 262144 :140ch	0(W52のみ)	可
error	エラー情報 エラー原因の内容が返されます。	文字列データ 「エラー情報(errorプロパティ)」(7-243ページ)	""	不可

※1 メソッドを実行するときは、trueに設定します。
プロパティに値を格納するときは、falseに設定します。プロパティの参照は、true、falseのどちらでもできます。
※2 出荷時のBTシリーズのSSIDは、””（長さ0の文字列）が設定されています。
※3 これらの無線設定を変更すると、無線通信が不安定になる可能性があります。特に変更が不要な場合は初期値のままで使用してください。

イベントプロパティ

名称	発生タイミング	sender
onConnect	無線LANに接続したとき。ローミング発生時も含みます。	"infra"/"adhoc"
onOutrange	電波が届かない場所で圏外になったとき	"infra"/"adhoc"
onRefused	無線LANへの接続が拒否されたとき	"infra"/"adhoc"

メソッド

名称	説明	引数	戻り値
Initialize	各プロパティを初期化します。	なし	なし
Open※1	無線LANを使用可能にします。	mode:"sync":同期	true：正常終了 false：エラー終了
Close	無線LANを使用不可にします。	なし	なし
GetRecvLevel※3	アクセスポイントとの受信品質を確認します。	なし	0～5：通信品質 0：(通信不可) 5：(通信良好) nil：失敗時
SetSecurityInfo	無線LANの認証について設定します。※2	type:認証方式 value:設定値	true：正常終了 false：エラー終了

※1 同期モード(sync)でOpenメソッドを実行すると、無線LANが使用可能になるまで戻り値を返しません。
※2 設定できる組み合わせは以下のとおりです。
※3 WLAN:Open以外で無線ポートを開いている場合には、実行時にnilが返ります。

type	value
"PSK"	PSKのネットワークキー
"USER_ID"	PEAPのユーザID
"PASSWORD"	PEAPのパスワード
"ROOT_CA"	PEAP、TLSのルート証明書が格納されたファイル名
"CERT"	TLSのクライアント証明書が格納されたファイル名
"PRIVATE_KEY"	TLSの秘密鍵が格納されたファイル名

■ポートの状態(isOpenプロパティ)

無線で通信をおこなうときには、プロパティに通信条件を設定してから、メソッドを実行します。
下表に、「isOpen」プロパティの値によって利用できることと、できないことを示します。

isOpenの値	プロパティの参照	プロパティの設定	メソッドの実行
true	○	×	○
false	○	○	×

○：利用可能です。
×：利用できません。

■エラー情報(errorプロパティ)

エラー情報	説明
"PARAM_ERR"	パラメータエラー
"WLAN_ERR"	無線接続失敗
"WLAN_ALREADYOPEN"	既に無線接続されている
"WLAN_TIMEOUT"	タイムアウト

使用例

例1 無線LAN認証の設定をします。

```
Method SetSecurityInfo()
        password = "keyenceautoid"
Begin
        With WLAN
                :wpaMode = "PSK"
                :cipher = "TKIP"
                :networkMod = "infrastructure"
                // PSKのネットワークキーを設定します
                :SetSecurityInfo("PSK",password)
        EndWith
End Method
```

例2 アクセスポイントとの受信品質を確認します。

```
Method WLAN()
        ret
Begin
        With WLAN
        // ポートオープンします。
        If :Open("sync") is false Then
                Handy:ShowMessageBox( "無線ポートオープンに失敗しました", "confirm" )
        Else
                // アクセスポイントとの受信品質を確認します
                ret = :GetRecvLevel()
                If ret is true Then
                        Handy:ShowMessageBox("通信品質:" & ret,"ok")
                EndIf
                :Close()
        EndIf
        EndWith
End Method
```

Network	BTシリーズのネットワークの設定をします。

プロパティ

名称	説明	範囲	初期値	代入
localIp※1	BTシリーズの IP アドレス	XXX.XXX.XXX.XXX (15バイト固定)	BTシリーズの設定値	可
subnet※2	BTシリーズの サブネットマスク	XXX.XXX.XXX.XXX (15バイト固定)	BTシリーズの設定値	可
gateway※3	BTシリーズの デフォルトゲートウェイ	XXX.XXX.XXX.XXX (15バイト固定)	BTシリーズの設定値	可
hostIp※4	サーバのIP アドレス	XXX.XXX.XXX.XXX (15バイト固定)	BTシリーズの設定値	可
dataPort※5	BTシリーズの データ転送用ポート番号	49152～65535	BTシリーズの設定値	可
ftpPort※6	BTシリーズの ファイル転送用ポート番号	49152～65535	BTシリーズの設定値	可
dhcp※7	BTシリーズの DHCPサーバ設定	true:DHCPサーバを使用する false:DHCPサーバを使用しない	BTシリーズの設定値	可
terminalName	端末名	文字列	BTシリーズの設定値	可
primaryDns	プライマリDNS	XXX.XXX.XXX.XXX (15バイト固定)	BTシリーズの設定値	可
secondaryDns	セカンダリDNS	XXX.XXX.XXX.XXX (15バイト固定)	BTシリーズの設定値	可
dispTransDialog	ファイル通信中ダイアログ表示設定	true:表示 false:非表示	true	可
modemATcommand	接続時に付加するATコマンドです	文字列	BTシリーズの設定値	可
modemTelnumber	モデムの電話番号	文字列(1～16桁)	BTシリーズの設定値	可
modemDialType	回線種別	"PULSE" "TONE"	BTシリーズの設定値	可
modemDialTimeout	ダイアル応答タイムアウト(秒)	1～115	BTシリーズの設定値	可
modemRedialCount	リダイアル回数	0～5	BTシリーズの設定値	可
modemRedialInterval	リダイアル間隔	0～30	BTシリーズの設定値	可
modemCradleBaudrate	通信ユニット経由時のボーレート	9600/19200/38400/ 57600/115200(bps)※8	BTシリーズの設定値	可
rssiThresh	ローミング閾値を設定します。※9 BT-600シリーズは使用できません。	60-90 [(-1)dbm]	BTシリーズの設定値	可

名称	説明	範囲	初期値	代入
blueSppMode	BluetoothSPP の接続モード BT-600/1000/1500シリーズのBluetooth搭載機種のみ有効です。	"Master" "Slave"	"Master"	可
infraChannel	無線LANでの通信時に、IEEE802.11bの使用チャネルを限定します。※9 14chはBT-W100/W155/W80/W85/W85T/W70/W75/1010W/1010WB/1550WBのみ有効です。	有効にするチャネルの数字の和を代入します。 〈0～16383〉 0： 全チャネル有効 1： 1ch 2： 2ch 4： 3ch 8： 4ch 16： 5ch 32： 6ch 64： 7ch 128： 8ch 256： 9ch 512 :10ch 1024 :11ch 2048 :12ch 4096 :13ch 8192 :14ch	BTシリーズの設定値	可
error	エラー情報 エラー原因の内容が返されます。	文字列データ 「エラー情報(errorプロパティ)」(7-247ページ)	" "	不可

※1 出荷時のIPアドレスは、”000.000.000.000”に設定されています。
※2 出荷時のサブネットマスクは、”000.000.000.000”に設定されています。
※3 出荷時のデフォルトゲートウェイは、”000.000.000.000”に設定されています。
※4 出荷時のサーバIPアドレスは、”000.000.000.000”に設定されています。
※5 出荷時のデータ転送用ポート番号は65000に設定されています。
※6 出荷時のファイル転送用ポート番号は65003に設定されています。
※7 dhcp プロパティを true(DHCP サーバを使用する)に設定すると、ローカル IP、デフォルトゲートウェイ、サブネットマスクのプロパティがDHCPサーバから取得した値で上書きされます。
※8 RS-232C通信ユニット(BT-UC10R)使用時は19,200 bit/s以下に設定してください。
※9 このプロパティは通常変更する必要はありません。変更すると無線通信の安定性に影響を与える可能性がありますので、適正条件になるように運用環境にて十分にテスト・調整をおこなってください。

イベントプロパティ

名称	発生タイミング	sender
onConnect	接続状態になった	Openメソッドで指定したtype
onOutrange	圏外状態になった	Openメソッドで指定したtype
onRefused	接続を拒否された	Openメソッドで指定したtype
onDHCPIpChanged	DHCPで割り当てられたIPアドレスが変更になった BT-600シリーズでは使用できません。	Openメソッドで指定したtype

※ NetworkコントロールのイベントとWLANまたはBluetoothコントロールのイベントの両方ともイベントプロパティを登録した場合、後から登録されたイベントプロパティのみが呼び出されます。

メソッド

名称	説明	引数	戻り値
Initialize	各プロパティを初期化します。	なし	なし
Ping	サーバと接続が確立されているか確認します。	ip:送信先IPアドレス XXX.XXX.XXX.XXX(15バイト固定) timeout:タイムアウト1～100(×100ミリ秒)	true:正常終了 false:エラー終了
SntpRequest	SNTPサーバに接続して、BTシリーズの日時を設定します。	ip:SNTPサーバのIPアドレス XXX.XXX.XXX.XXX(15バイト固定) timeout:タイムアウト1～100(×100ミリ秒)	true:正常終了 false:エラー終了
Open	各種ネットワークデバイスをオープンします。※1	mode:"sync"同期 type:"infrastructure":無線 "adhoc":無線アドホック "LanCradle":通信ユニット(mode1) "Bluetooth SPP":Bluetooth SPP接続 "Bluetooth PPP":Bluetooth PPP接続 "Bluetooth HID":Bluetooth HID接続 "Modem":モデムシリアル接続 (RS-232Cポート経由) "Modem PPP":モデムPPP接続 (RS-232Cポート経由) "Modem Cradle":モデムシリアル接続 (通信ユニット経由) "Modem Cradle PPP":モデムPPP接続 (通信ユニット経由)	true:正常終了 false:エラー終了
Close	各種ネットワークデバイスをクローズします。※1	type: Openの引数と同じ nilを指定すると全デバイスクローズします。	true:正常終了 false:エラー終了
IsOpen	各種ネットワークデバイスの状態を確認します。※1	type: Openの引数と同じ	true:正常終了 false:エラー終了
NIResolverRequest	指定されたNIResolverサーバに対して登録要求を実施します。	serverIP: 登録先サーバIP(15バイト固定) timeOut: 登録タイムアウト(1～100(×100ミリ秒))	true:正常終了 false:エラー終了

※1 下表のコントロールのOpen/Enableと同じ意味を持ちます(一方のコントロールでOpenするともう一方のコントロールでもOpenしたことになります)。

Networkコントロールのtype	影響を受けるコントロール名
"infrastructure"	CommRF
	WLAN
"adhoc"	WLAN
"Bluetooth SPP"	Bluetooth
"Bluetooth PPP"	Bluetooth

■ポートの状態(IsOpenメソッド)

各デバイス間で通信をおこなうときには、プロパティに通信条件を設定してからメソッドを実行します。下表に、「IsOpen」メソッドの戻り値によって利用できることと、できないことを示します。

IsOpenの値	プロパティの参照	プロパティの設定	メソッドの実行
false	○	○	×
true	○	△	○

○:利用可能です。
△:設定後一旦falseにして再度trueにしたときに、有効になります。
×:利用できません。

■エラー情報(errorプロパティ)

エラー情報	説明
"PARAM_ERR"	パラメータエラー
"NETWORK_ERR"	ネットワークに接続されていない
"TMOUT"	タイムアウト
"NETWORK_ALREADYCLOSE"	既にポートは閉じています
"NETWORK_ALREADYOPEN"	既にポートはオープンしています。

使用例

例1 Pingを送信します。

```
Method Ping()
	hostIp
Begin
	If Network:Open("sync","LanCradle") is false Then
		Handy:ShowMessageBox("通信ポートをオープンに失敗しました","confirm")
	EndIf
	hostIp = Network:hostIp
	// Pingコマンドを用いてネットワークの疎通を確認します
	Handy:ShowMessageBox("pingを送信します","confirm")
	If Network:Ping(hostIp,20) is true Then
		Handy:ShowMessageBox("サーバとの接続は確立されています","confirm")
	Else
		Handy:ShowMessageBox("サーバとの接続は確立されていません","confirm")
	EndIf
	Network:Close(nil) //通信ポートを閉じます
End Method
```

例2 SNTPサーバに接続して日時を設定します。

```
Method TimeSetting()
	hostIp
Begin
	If Network:Open("sync","LanCradle") is false Then
		Handy:ShowMessageBox("通信ポートをオープンに失敗しました","confirm")
	EndIf
	hostIp = Network:hostIp
	// SNTPサーバに接続して日時を設定します
	Handy:ShowMessageBox("時刻を設定します","confirm")
	If Network:SntpRequest(hostIp,20) is true Then
		Handy:ShowMessageBox("日時が設定されました","confirm")
	Else
		Handy:ShowMessageBox("日時設定に失敗しました","confirm")
	EndIf
	Network:Close(nil) //通信ポートを閉じます
End Method
```

Serial	Bluetooth、RS-232C、赤外線通信(IrDA)、USB、モデムを使って通信をおこないます。タグ指定コントロールです。

▶ 重 要

タグには、使用するデバイス("Bluetooth"/"232C"/"IrDA"/"USB"/"Modem")を指定します。
ただし、指定したデバイスを搭載していない機種では使用できません。
BT-600シリーズでは"232C"、"USB"は使用できません。

プロパティ

名称	説明	範囲	初期値	代入
enable※1	通信ポートの状態を設定します。 📖「ポートの状態(enable プロパティ)」(7-250ページ)	false:通信ポートのクローズ(使用不可) true:通信ポートのオープン(使用可能)	false	可
timeOut	タイムアウト時間を設定します。各メソッドで共通して使用されます。	1～6000(×10ミリ秒)	500	可
baudRate※2	通信速度(bps)を設定します。	9600/19200/38400/57600/115200(bps)	115200	可
stopBits※2	ストップビットを設定します。	1/2(ビット)	1	可
dataLen	データ長を設定します。	7/8(ビット)	8	可
parity	パリティを設定します。	0:なし 1:奇数 2:偶数	0	可
cancelKey	通信をキャンセルするときに押すキーを設定します。	nil:キャンセルキー無効 "C":キャンセルキー "ENT":エンターキー "F1"/"F2"/"F3":ファンクションキー "ANY":任意キー	"C"	可
recvLength	受信データ長	数値	0	不可
error	エラー情報 エラー原因の内容を返します。	文字列データ 📖「エラー情報(errorプロパティ)」(7-250ページ)	" "	不可

※1 メソッドを実行するときは、true に設定します。
プロパティに値を格納するときは、falseに設定します。プロパティの参照は、true、falseのどちらでもできます。
- Bluetooth を使用して通信するときは、Bluetooth または Network コントロールの Open メソッドを実行してBluetoothを使用可能にしてください。
- 赤外線通信(IrDA)、RS-232C を使用して通信するときは、Comm1、Comm2 コントロールのenableプロパティをfalseに設定してください。

※2 赤外線通信(IrDA)、RS-232C、USB、モデムを使用して通信するときのみ設定できます。
Comm1、Comm2コントロールの同名のプロパティとは異なります。使用するコントロールごとに設定してください。

メソッド

名称	説明	引数	戻り値
Initialize	各プロパティを初期化します。	なし	なし
Read※	データを受信します。	maxLength:最大受信バイト数(1～2048)	読み込み文字列 false:読み込みエラー
Write	データを送信します。	string:書き込み文字列(最大8192バイト)	true:正常終了 false:エラー終了
Clear	送受信に使用するバッファをクリアします。	type:クリアするバッファの種類(1:送信、2:受信、3:送受信)	なし

※「enable」プロパティを true に設定してから「Read」メソッドを実行するまでに受信したデータが保存されていることがありますので、必要に応じて「Clear」メソッドで受信バッファをクリアしてください。
指定するバイト数に達した、または「timeOut」プロパティに設定した時間を過ぎると、データ受信処理は終了して、メソッドから戻り、スクリプトプログラムの次のステートメントに移ります。

■ポートの状態(enable プロパティ)

通信を行うときには、プロパティに通信条件を設定してから、「Read」、「Write」などのメソッドを実行します。プロパティ値を設定するには、あらかじめ「enable」プロパティを「false」に設定します。
下表に、「enable」プロパティの値によって利用できることと、できないことを示します。

enable の値	プロパティの参照	プロパティの設定	メソッドの実行
false	○	○	×
true	○	△	○

○:利用可能です。
△:設定後一旦falseにして再度trueにしたときに、有効になります。
×:利用できません。

■エラー情報(errorプロパティ)

エラー情報	説明
"PORT_NOT_OPEN"	シリアルポートが開いていない
"PARAM_ERR"	パラメータエラー
"ERR_ARG_RANGE1"	第1引数不正
"ERR_ARG_RANGE2"	第2引数不正
"ERR_ARG_RANGE3"	第3引数不正
"ERR_ARG_RANGE4"	第4引数不正
"ERR_ARG_RANGE5"	第5引数不正
"ERR_ARG_RANGE6"	第6引数不正
"OTHER"	その他

使用例

例1　データを受信します。

```
Method ReadSample()
    data
    Begin

        With Serial<"Bluetooth">
            // 通信条件を設定
            :timeOut = 1000                    // タイムアウト時間に10 秒を設定
            :baudRate = 9600
            :stopBits = 1
            :dataLen = 8
            :parity = 0
            :cancelKey = "ANY"
            :enable = true                     // 通信ポートのオープン
            :Clear(3)                          // データを送受信するバッファのクリア
            data = :Read(5)                    // 5バイトのデータを受信
            Handy:ShowMessageBox(data, "confirm")
            :enable = false                    // 通信ポートのクローズ
        EndWith
EndMethod
```

Comm	無線/通信ユニット/RS-232Cポート/USBポート/Bluetooth/赤外線ポート/モデム経由でのファイル送受信をおこないます。

▶ 重 要

- **Commコントロールの「PutFile」、「GetFile」、「PutLogFiles」、「PutFileCompress」、「GetFileCompress」メソッドで操作するファイルを、既に「File」、「Master」、「Search」コントロールで操作している場合には、使用したコントロールで初期化（Initialize）してください。**
- **PC側では「データ転送ソフト」、または「通信ライブラリを使用してユーザで開発したPCアプリケーション」が必要です。「通信ライブラリ」を使用してPCアプリケーションを開発する場合には、サーバとのデータ送受信コントロール(RemoteAccess/ RemoteDB/ RemoteFile/RemoteUD)は使用できません。**

プロパティ

名称	説明	範囲	初期値	代入
timeOut	タイムアウト時間を設定します。 メソッドで共通して使用されます。	1～50(秒)	30	可
cancelKey	通信をキャンセルするときに押すキーを設定します。	nil:キャンセルキー無効 "C":キャンセルキー "ENT":エンターキー "F1"/"F2"/"F3": ファンクションキー "ANY":任意キー	"C"	可
resultFileName	Listen時のやりとりのログを記録するファイル名を指定します。 ドライブは、「ドライブ3」固定です。	ドライブ番号:ファイル名	nil	可
deviceId※1	通信経路を設定します。	"CRADLE"または"IRDA": 通信ユニット(232C, USB, LAN通信ユニット(mode2)) "USB" :USBポート "RS232C": RS-232Cポート "MODEM": モデム(シリアル接続) "IRDA2":赤外線通信ポート "RF":無線 "BLUETOOTH": Bluetooth(PPP接続)※2 "BLUETOOTH_SPP": Bluetooth(SPP接続)※2 "SOCK_COMPATIBLE": LAN通信ユニット(mode1)、無線、Bluetooth(PPP接続)、モデム(PPP接続) "SOCK_PASSIVE"※2: LAN通信ユニット(mode1)、無線、Bluetooth(PPP接続)、モデム(PPP接続)	無線LAN搭載機種:"RF" それ以外の機種:"CRADLE"	可
quQueryFileVersion	次に更新される中間言語ファイルのバージョン	QuQueryで取得した値	""	不可
quQueryFileTitle	次に更新される中間言語ファイルのタイトル	QuQueryで取得した値	""	不可

名称	説明	範囲	初期値	代入
quQueryFileName	次に更新される中間言語ファイルのファイル名	QuQueryで取得した値	""	不可
error	エラー情報 エラー原因が格納されます。	文字列データ 📖「エラー情報(errorプロパティ)」(7-256ページ)	""	不可

※1 無線を使用する場合には Network(または CommRF、WLAN)、Bluetooth を使用する場合にはNetwork(またはBluetooth)、モデム通信およびLAN通信ユニット(mode1)の場合にはNetworkの各コントロールでポートを開く必要があります。

※2 Connect時、端末からPCに対して接続を開始します。

メソッド

名称	説明	引数	戻り値
Initialize	各プロパティを初期化します。	なし	なし
Listen	PCからの待ち受けを開始します。 (BTシリーズがサーバ)	なし	true:正常終了 false:エラー終了
Connect	PCとの接続を開始します。 (PCがサーバ)	なし	true:正常に開始 false:エラー終了
Disconnect	PCとの接続を終了します。 (PCがサーバ)	なし	true:正常終了 false:エラー終了
PutFile※1	PCにファイル、ログファイルを送信します。 (PCがサーバ)	localFileName: 送信元(端末)のファイル名 (ドライブ番号:ファイル名) remoteFileName: 送信先のファイル名※2	true:正常終了 false:エラー終了
GetFile※1	PCからファイルを受信します。 (PCがサーバ)※3	remoteFileName: 受信するファイル名 localFileName: 保存するファイル名 (ドライブ番号:ファイル名)	true:正常終了 false:エラー終了
PutLogFiles※1	PCにログファイルを送信します。 (PCがサーバ) ドライブ4の指定はできません。	driveNo:ドライブ番号(3)	true:正常終了 false:エラー終了
DeleteFile	指定するファイルを削除します。 ファイル名は受信フォルダからの相対パスです。※4	FileName: PC上のファイル名	true:正常終了 false:エラー終了
RenameFile	指定するPC上のファイルをリネームします。 ファイル名は受信フォルダからの相対パスです。※4※6	oldFileName:PC上のファイル名 newFileName:PC上のファイル名	true:正常終了 false:エラー終了
CompressFile	指定するPC上のファイルをZIP圧縮します。 ファイル名は送信フォルダからの相対パスです。※4	remoteFileName:PC上のファイル名 compressFileName:PC上の圧縮後ファイル名 overWrite:上書き許可/禁止(true/false)	true:正常終了 false:エラー終了
UnCompressFile	指定するPC上の圧縮ファイルを解凍します。 ファイル名は受信フォルダからの相対パスです。※4※6	remoteFileName:PC上の圧縮ファイル名 overWrite:上書き許可/禁止(true/false)	true:正常終了 false:エラー終了
PutFileCompress※1	ハンディ上でファイルをZIP圧縮後、PCに圧縮ファイルを送信します。 ファイル名は受信フォルダからの相対パスです。※4	localFileName 送信元のファイル名 (ドライブ番号:ファイル名) remoteFileName 送信先のファイル名	true:正常終了 false:エラー終了
GetFileCompress※1	PC上でファイルをZIP圧縮後、PCからファイルを受信します。※3 ファイル名は送信フォルダからの相対パスです。※4※5	remoteFileName:受信するファイル名 localFileName:保存するファイル名 (ドライブ番号:ファイル名)	true:正常終了 false:エラー終了
QuQuery	通信ライブラリを使って、端末アプリのバージョン確認をします。	filePass: PC上のフォルダ名	true:正常終了 false:エラー終了

名称	説明	引数	戻り値
SystemUpdate※1	通信ライブラリを使って、本体システム(ファームウェア)のバージョンアップを実施します。 BT-1000/1500/600のみ有効です。	systemProgram:PC上の本体システム(ファームウェア)ファイル名 drive_No:ファイル受信先のドライブ番号(1/2/5) applicationFile:Update後、再起動するアプリ名 (.APP)※7 (ドライブ番号:ファイル名)	true:正常終了 false:エラー終了
SystemCheck	通信ライブラリを使って、PC上にある本体システム(ファームウェア)ファイルのバージョン確認をします。 BT-1000/1500/600のみ有効です。	systemProgram: PC上の本体システム(ファームウェア)ファイル名	"old":PC上のファイルバージョンが古い "same":PC上のファイルバージョンと同じ "new":PC上のファイルバージョンより新しい nil:エラー終了

※1 送受信時に液晶表示部(LCD)のステータスエリアに転送状態を示すアイコンが表示されます。

※2 ドライブ3のファイルを送信した場合、送信元(端末)のファイル名で送信されます。

※3 サーバ上のファイルを受信する際、ドライブ2 (microSDカードがマウントされているときはドライブ2とドライブ5) に一時ファイルを作成します。その他のドライブにファイルを受信する場合にも、これらのドライブに受信ファイル容量分の空き容量が必要となります。

※4 相対パス指定の際、「..¥」による上階層パス指定は保証できません。

※5 ドライブ2に一時圧縮ファイルを受信した後、指定のドライブに解凍します。受信先がドライブ2の場合、一時圧縮ファイル+解凍後のファイルのサイズがドライブ2の空き容量より小さい必要があります。空き容量が足りない場合、通常（非圧縮）のファイル受信をおこないます。

※6 データ転送ソフトをPC ソフトで使用する場合は、データ転送ソフトの仕様上、ユーザが指定しているフォルダとアプリケーションが内部的に指定しているフォルダが異なるため、スクリプトでの指定フォルダを変更する必要があります。

※7 引数（applicationFile）にnilを指定すると、現在起動しているアプリを再起動します。

ポイント

LAN通信ユニットを用いたデータ送受信
LAN通信ユニットの動作モードの設定には、mode1とmodo2があります。
mode1では、deviceIdに"SOCK_COMPATIBLE"、または"SOCK_PASSIVE" を指定します。"SOCK_PASSIVE"を指定すると、Connect時にセッションを端末からPCへ接続する 動きになります。ただし、この場合待ち受けするPC側もパッシブモードに設定する必要 があります。(データ転送ソフトの場合、システム設定の詳細設定からパッシブモードに 変更することが可能です) "SOCK_COMPATIBLE"を指定した場合、PC側に特別な設定はいりません。
mode2は、BT-500シリーズとの互換性を考慮した動作モードです。

■ファイルの送受信

PCと接続して、BTシリーズからファイルを送受信する場合（PCがサーバ、BTシリーズがクライアント）の手順は次のようになります。

1 「Connect」メソッドで、PCと接続します。

2 「PutFile」、「GetFile」、「PutLogFile」、「PutFileCompress」、「GetFileCompress」メソッドでファイルを送受信します。

3 「Disconnect」メソッドで接続を終了します。

■エラー情報（errorプロパティ）

エラー情報	説明
"CONVERT_FAILED"	マスタファイルまたはアプリケーションへの変換エラー
"INCOMPLETE"	処理が完了しませんでした
"FILENOTFOUND"	ファイルが見つかりません
"BIGDATA"	ファイルサイズが大きすぎます
"NETDOWN"	通信経路が切断されました
"TIMEOUT"	タイムアウトが発生しました
"CANCELED"	キャンセルされました
"REFUSED"	拒絶されました
"NOTCONNECT"	接続されていません
"OTHER"	上記のいずれにも該当しないエラーです

使用例

例1　パソコンと接続して、ログファイルの送信、マスタファイルの受信をおこないます。

```
Method sample1()
Begin
	Comm:Initialize()
	Comm:deviceId = "CRADLE" // デバイスを通信ユニットに指定
	Comm:Connect() // 接続開始

	// ログファイルを送信し、結果をメッセージボックスで表示します。
	If Comm:PutLogFiles(3) Then
		Handy:ShowMessageBox("ログファイルの送信に成功しました","confirm")
	Else
		Handy:ShowMessageBox("ログファイルの送信に失敗しました","confirm")
	EndIf

	// マスタファイルを受信し、結果をメッセージボックスで表示します。
	If Comm:GetFile("MASTER.TXT","MASTER.TXT") Then
		Handy:ShowMessageBox("マスタファイルの受信に成功しました","confirm")
	Else
		Handy:ShowMessageBox("マスタファイルの受信に失敗しました","confirm")
	EndIf
	Comm:Disconnect() // 接続終了
End Method
```

例2　Commコントロールの各メソッドの使用例

```
Method main() // メインメソッド
	strItem ="DeleteFile|RenameFile|CompressFile|UnCompressFile|PutFileCompress|
		GetFileCompress|QuQuery|SystemCheck|SystemUpdate"
	menupos = 1
Begin
	Comm:deviceId = "CRADLE" // デバイスを通信ユニットに指定
	While(1)
		menupos = 1
		// サーバ上のファイルにアクセスする種類を選択するメニューを表示
		menupos = Handy:ShowMenu("Comm",strItem,nil,nil,nil,menupos)
		Select Case menupos
		Case 1
			// サーバ上のファイルを削除するメソッドの呼び出し
			_DeleteFile()
		Case 2
			// サーバ上のファイルをリネームするメソッドの呼び出し
			_RenameFile()
		Case 3
			// サーバ上にファイルを圧縮するメソッドの呼び出し
			_CompressFile()
		Case 4
			// サーバ上のファイルを解凍するメソッドの呼び出し
			_UnCompressFile()
		Case 5
			// ファイルを圧縮して送信するメソッド
			_PutFileCompress()
		Case 6
```

```
                    // ファイルを圧縮して受信するメソッド
                    _GetFileCompress()
                Case 7
                    // 中間言語ファイルのバージョンを確認するメソッド
                    _QuQuery()
                Case 8
                    // ファームウェアのバージョンを確認するメソッド
                    _SystemCheck()
                Case 9
                    // ファームウェアのバージョンアップを実行するメソッド
                    _SystemUpdate()
                End Select
        Wend
End Method
```


例3 サーバ上のファイルを削除します。

```
Method _DeleteFile()
Begin
        Handy:ShowMessageBox("PC上のファイルを削除します","confirm")
        Comm:Connect()
        Comm:DeleteFile("sample.txt")
        Comm:Disconnect()
End Method
```

例4 サーバ上のファイルをリネームします。

```
Method _RenameFile()
Begin
        Handy:ShowMessageBox("PC上のファイルをリネームします","confirm")
        Comm:Connect()
        Comm:RenameFile("sample.txt","sample2.txt")
        Comm:Disconnect()
End Method
```

例5 サーバ上のファイルを圧縮します。

```
Method _CompressFile()
Begin
        Handy:ShowMessageBox("PC上のファイルを圧縮します","confirm")
        Comm:Connect()
        Comm:CompressFile("sample.txt","sample.zip",false)
        Comm:Disconnect()
End Method
```

例6 サーバ上のファイルを解凍します。

```
Method _UnCompressFile()
Begin
        Handy:ShowMessageBox("PC上のファイルを解凍します","confirm")
        Comm:Connect()
        Comm:UnCompressFile("sample.zip",false)
        Comm:Disconnect()
End Method
```

例7 ファイルを圧縮してサーバ上に送信します。

```
Method _PutFileCompress()
Begin
        Handy:ShowMessageBox("ファイルを圧縮してPCに送信します","confirm")
        Comm:Connect()
        Comm:PutFileCompress("2:sample.txt","sample.zip")
        Comm:Disconnect()
End Method
```

例8 サーバ上で圧縮したファイルを受信します。

```
Method _GetFileCompress()
Begin
        Handy:ShowMessageBox("PC上のファイルを圧縮して受信します","confirm")
        Comm:Connect()
        Comm:GetFileCompress("sample.txt","2:sample.zip")
        Comm:Disconnect()
End Method
```

例9 サーバ上の中間言語ファイルのバージョンを確認します。

```
Method _QuQuery()
Begin
        Handy:ShowMessageBox("中間言語ファイルのバージョンを確認します","confirm")
        Comm:Connect()
        Comm:QuQuery("Apps_DT")
        Handy:ShowMessageBox(Comm:quQueryFileVersion,"confirm")
        Handy:ShowMessageBox(Comm:quQueryFileTitle,"confirm")
        Handy:ShowMessageBox(Comm:quQueryFileName,"confirm")
        Comm:Disconnect()
End Method
```

例10 サーバ上のファームウェアのバージョンを確認します。

```
Method _SystemCheck()
        ret
Begin
        Handy:ShowMessageBox("ファームウェアのバージョンを確認します","confirm")
        Comm:Connect()
        ret = Comm:SystemCheck("__systemprogram06.bin")
        If ret is not false Then
                Handy:ShowMessageBox(ret, "confirm")
        EndIf
        Comm:Disconnect()
End Method
```

例11 ファームウェアのバージョンアップを実行します。

```
Method _SystemUpdate()
Begin
        Handy:ShowMessageBox("ファームウェアのバージョンアップを実施します","confirm")
        Comm:Connect()
        Comm:SystemUpdate("__systemprogram06.bin",2,nil)
        Comm:Disconnect()
End Method
```

Socket	ソケット通信の設定をおこないます。

プロパティ

名称	説明	範囲	初期値	代入
retIp	·Recvfromメソッド ·Getpeernameメソッド ·Getsocknameメソッド で取得できるIPアドレス	XXX.XXX.XXX.XXX (15バイト固定)	nil	不可
cancelKey	同期通信時に通信をキャンセルするキーを設定します。	nil:キャンセルキー無効 "C":キャンセルキー "ENT":エンターキー "F1"/"F2"/"F3":ファンクションキー "ANY":任意キー	"C"	可
retPort	·Recvfromメソッド ·Getpeernameメソッド ·Getsocknameメソッド で取得できるポート番号	0～65535	nil	不可
error	エラー情報 エラー原因が格納されます。	文字列データ 「エラー情報(errorプロパティ)」(7-262ページ)	nil	不可

メソッド

名称	説明	引数	戻り値
Open	ソケットハンドルを生成します。	af:"AF_INET"固定 type:ソケット種別 "SOCK_STREAM":TCP "SOCK_DGRAM":UDP protocol:0固定	ハンドル false:エラー終了
Close	ソケットを閉じます。	s:ソケットハンドル※1	true:正常終了 false:エラー終了
Bind	ローカルアドレスとソケットを関連付けます。	s:ソケットハンドル※1 af:"AF_INET"固定 ip:IPアドレス port:ポート番号	true:正常終了 false:エラー終了
Shutdown	ソケットでの送受信を無効にします。	s:ソケットハンドル※1 how:通信種別 "SEND":送信無効 "RECV":受信無効 "BOTH":送受信無効	true:正常終了 false:エラー終了
Connect	指定されたソケットへの接続を確立します。	s:ソケットハンドル※1 af:"AF_INET"固定 ip:相手先IPアドレス port:ポート番号	true:正常終了 false:エラー終了
Listen	ソケットを受動待機モードにして、保留接続キューサイズを確定します。	s:ソケットハンドル※1 backlog:1固定	true:正常終了 false:エラー終了
Accept	待機中のソケットで接続要求を受け付けます。	s:ソケットハンドル※1	ハンドル false:エラー終了
Send	接続されているソケットでデータを送信します。	s:ソケットハンドル※1 data:データ(最大2048バイト) flag:0固定	送信バイト数 false:エラー終了

名称	説明	引数	戻り値
Sendto	指定されたIPアドレスへデータを送信します。	s:ソケットハンドル※1 data:データ af:"AF_INET"固定 ip:相手先IPアドレス port:ポート番号	送信バイト数 nil
Recv	接続されているソケットでデータを受信します。	s:ソケットハンドル※1 flag:(0/"MSG_PEEK")	データ (空文字を含む) nil
Recvfrom	データと送信側のIPアドレスを受信します。※2	s:ソケットハンドル※1	データ nil
Getpeername	ソケットの接続先である相手のIPアドレスを取得します。※2	s:ソケットハンドル※1	true:正常終了 false:エラー終了
Getsockname	ソケットのローカルアドレスを取得します。※2	s:ソケットハンドル※1	true:正常終了 false:エラー終了
GetHostByName	ホスト名からIPアドレス等の情報を取得します。※7	hostname:ホスト名	IPアドレス等の情報※6 false:エラー終了
Select	ソケット状態を調べます。	readfds:読み取り確認ソケットハンドル※1※3 writefds:書き込み確認ソケットハンドル※1※3 exceptfds: エラー確認ソケットハンドル※1※3 timeOut:タイムアウト 0~500(×100ミリ秒)	>0: 条件を満たしたソケット数 0: タイムアウト nil: エラー終了
SyncSend※4	接続されているソケットでデータを送信します(同期通信)。	s:ソケットハンドル data:送信データ(最大8192バイト) timeout:タイムアウト 0~500(×100ミリ秒)	送信バイト数
SyncRecv※5	接続されているソケットでデータを受信します(同期通信)。	s:ソケットハンドル size:受信データサイズ(1~8192バイト) timeout:タイムアウト 0~500(×100ミリ秒)	受信データ

※1 「Open」メソッドの戻り値を使用します。
※2 実行結果のアドレスは、retIp、retPortに格納されます。
※3 複数のソケットを指定する場合には、ソケットハンドルを";"(セミコロン)で連結してください。
例 readfds = s1 & ";" & s2
※4 指定したタイムアウト時間内に送信が完了しないときは、引数に指定した送信データ長よりも小さい値を返します。また、errorプロパティに"SOCK_TIMEOUT"が格納されます。
不正な引数を指定した場合はfalseを返します。
※5 指定したタイムアウト時間内にデータを受信できなかったときは、タイムアウトするまでに受信したデータを返します。また、errorプロパティに"SOCK_TIMEOUT"が格納されます。
送信されたデータが、引数に指定した受信データサイズより大きいときは、繰り返しSyncRecvメソッドを実行してください。
※6 以下の情報がカンマ区切りで取得できます。
"正式名,別名,IPアドレス1, IPアドレス2, IPアドレス3"
IPアドレスは15桁固定の表記になります。例)127.000.000.001
IPアドレス2、IPアドレス3は存在しない場合は、空文字になります。
※7 PPP接続時などで使用します。

■エラー情報(errorプロパティ)

エラー情報	説明
"SOCK_EINVAL"	ソケット番号が不正 不正な引数または手順でコールされた その他のエラー
"SOCK_EWOULDBLOCK"	要求された操作が完了していない
"SOCK_ENOTSOCK"	指定されたソケットは使用できないか、無効である
"SOCK_ENOTCONN"	ソケットは接続されていない
"SOCK_ENETDOWN"	指定されたソケットは切断された
"PARAM_ERR"	不正な引数でコールされた
"SOCK_TIMEOUT"	同期送信／受信がタイムアウトした
"SOCK_CANCEL"	同期送信／受信がキャンセルされた

使用例

Socketコントロールを使用したサンプル

```
Method main()
	strItem = "1:TCP Server|2:TCP Client|3:UDP Server|4:UDP Client"
	menupos = 1
Begin
	While( 1 )
		menupos = 1
		// メニュー画面の表示
		menupos = Handy:ShowMenu("ソケット",strItem,nil,nil,nil,menupos)
		Select Case menupos
		case 1
			// TCP Serverのサンプルメソッドを実行
			TcpServer()
		case 2
			// TCP Clientのサンプルメソッドを実行
			TcpClient()
		case 3
			// UDP Serverのサンプルメソッドを実行
			UdpServer()
		case 4
			// UDP Clientのサンプルメソッドを実行
			UdpClient()
		End Select
	Wend
End Method
```

例1　TCP serverの使用例

```
Method TcpServer()
	strTermIP = "192.168.100.100" // 端末IP
	strHostIP = "192.168.100.001" // ホストIP
	portno = 65004 // ポート番号
	socketListen // ソケットハンドル
	socketRemote // ソケットハンドル
	ret
	count = 0
Begin
	//LANポートをオープンする
	Network:Open("sync", "LanCradle")
	// ソケットをオープンする
	socketListen = Socket:Open("AF_INET", "SOCK_STREAM", 0)
	// ソケットを関連付ける
	Socket:Bind(socketListen, "AF_INET", strTermIP, portno)
	// クライアントからの要求を待機する
	socketListen = Socket:Listen(socketListen, 1)
	// クライアントからの接続要求を待機する
	If socketListen is nil Then return(nil) Endif
	While 1
		// 接続ができるまで待つ
		ret = Socket:Select(socketListen,0,0,10)
		If ret is nil Then
			//クライアントからの接続要求待ちに失敗
			return(nil)
		ElseIf ret == 0 Then
			count = count + 1
			If count > 60 Then
				//クライアントからの接続なし
				return(nil)
			End If
		Else
		Wbreak
		EndIf
	Wend
	// クライアントとの接続を受け付ける
	socketRemote = Socket:Accept(socketListen)
	// 受信データを待ち受ける
	count = 0
	While( 1 )
		// データが受信されるまで待ち受ける
		ret = Socket:Select(socketRemote,0,0,10)
		If ret is nil Then
			//クライアントからの受信待ちに失敗
			return(nil)
		ElseIf ret == 0 Then
			count = count + 1
			If count > 60 Then
			//クライアントからの受信がない
			return(nil)
			End If
		ElseIf ret > 0 Then
			Wbreak
		End If
	Wend
	// データを受信する
	Socket:Recv(socketRemote,0)
	// ソケットを閉じる
	Socket:Close(socketRemote)
	// ソケットを閉じる
	Socket:Close(socketListen)
	// LANポートを閉じる
	Network:Close("LanCradle")
End Method
```

例2 TCP Clientを実行する

```
Method TcpClient()
	strTermIP = "192.168.100.100" // 端末IP
	strHostIP = "192.168.100.001" // ホストIP
	portno = 65004 // ポート番号
	socketClient // ソケットハンドル
	sendData = "12345" //送信データ
	ret
	count = 0
Begin
	// LANポートをオープンする
	Network:Open("sync", "LanCradle")
	// ソケットをオープンする
	socketClient = Socket:Open("AF_INET", "SOCK_STREAM", 0)
	// ソケットを接続する
	Socket:Connect( socketClient, "AF_INET", strHostIP, portno )
	// 接続ができるまで待ち合わせる
	While( 1 )
		ret = Socket:Select(0,socketClient,0,10)
		If ret is nil Then
			//サーバへの接続に失敗
			return(nil)
		ElseIf ret == 0 Then
			count = count + 1
			If count > 60 Then
				//サーバからの応答なし
				return(nil)
			End If
		ElseIf ret > 0 Then
			Wbreak
		End If
	Wend
	// メッセージを送信する
	Socket:Send(socketClient,sendData,0)
	// ソケット送受信を無効にする
	Socket:Shutdown(socketClient, "BOTH")
	// ソケットを閉じる
	Socket:Close(socketClient)
	// LANポートを閉じる
	Network:Close("LanCradle")
End Method
```

例3 UDP Serverを実行する

```
Method UdpServer()
	strTermIP = "192.168.100.100" // 端末IP
	strHostIP = "192.168.100.001" // ホストIP
	portno = 65005 // ポート番号
	socketServer // ソケットハンドル
	ret
	count = 0
Begin
	// LANポートをオープンする
	Network:Open("sync", "LanCradle")
	// ソケットをオープンする
	socketServer = Socket:Open("AF_INET", "SOCK_DGRAM", 0)
	// ソケットを関連付ける
	Socket:Bind(socketServer, "AF_INET", strTermIP, portno)
	// 受信データを待ち受ける
	While( 1 )
		// データが受信されるまで待ち受ける
		ret = Socket:Select(socketServer,0,0,10)
		If ret is nil Then
			//クライアントからの受信待ちに失敗
			return(nil)
		ElseIf ret == 0 Then
			count = count + 1
			If count > 60 Then
				//クライアントからの受信なし
				return(nil)
			End If
		ElseIf ret > 0 Then
			Wbreak
		End If
	Wend
	// データを受信する
	Socket:Recvfrom(socketServer)
	// ソケットを閉じる
	Socket:Close(socketServer)
	// LANポートを閉じる
	Network:Close("LanCradle")
End Method
```

例4 UDP Clientを実行する

```
Method UdpClient()
        strTermIP = "192.168.100.100" // 端末IP
        strHostIP = "192.168.100.001" // ホストIP
        portno = 65005 // ポート番号
        socketClient // ソケットハンドル
        sendData = "12345" //送受信データ
Begin
        // LANポートをオープンする
        Network:Open("sync", "LanCradle")
        // ソケットをオープンする
        socketClient = Socket:Open("AF_INET", "SOCK_DGRAM", 0)
        // メッセージを送信する
        Socket:Sendto(socketClient, sendData, "AF_INET", strHostIP, portno)
        // ソケット送受信を無効にする
        Socket:Shutdown(socketClient, "BOTH")
        // ソケットを閉じる
        Socket:Close(socketClient)
        // LANポートを閉じる
        Network:Close("LanCradle")
End Method
```

FTP	FTP通信を設定します。

プロパティ

名称	説明	範囲	初期値	代入
timeOut	タイムアウト時間を設定します。	1～50(秒)	30	可
cancelKey	通信をキャンセルするときに押すキーを設定します。	nil:キャンセルキー無効 "C":キャンセルキー "ENT":エンターキー "F1"/"F2"/"F3"/:ファンクションキー "ANY":任意キー	"C"	可
resultFileName	Listen時のやりとりのログを記録するファイル名を指定します。	ドライブ番号:ファイル名	nil	可
passiveMode	FTPクライアントでパッシブモードで通信するかを設定します。	true:する false:しない	false	可
dirMode	Dirの実行モードを指定します。	1:NLSTモード 2:LISTモード	1(NLSTモード)	可
putMode	putFileの実行モードを指定します。 BT-1000/1500/600シリーズのみ有効です。	“STOR” “APPE”	"STOR"	可
portMode	FTPクライアントでGetFileしたときにデータポートの切断を待つかを設定します。	true:待つ false:待たない	false	可
enableFTPS	FTPSを有効にするか、さらにFTPS 使用時のプロトコルを設定します。	nil:無効 "TLS"(TLS使用) "SSL"(SSLv2v3)	nil	可
ftpsDataEncrypt	データポートの暗号化を設定します。	true:有効 false無効	false	可
ftpsVerify	証明書認証を利用したセッション確立をおこなうかを設定します。	true:おこなう false:おこなわない	false	可
ftpsCerFilename	証明書認証利用時に使用する、サーバから発行された証明書ファイル名	ドライブ番号:ファイル名	""	可
transMode	FTPクライアントでの転送モード	"BINARY"/"ASCII"	"BINARY"※1	可
port	FTPサーバのポート番号	1～65535	21	可
commandResult	FTP:Commandの戻り値文字列	文字列 (最大256バイトまで。それ以上はカット)	""	不可
tmpDriveNo	ファイル通信時に使用する一時ファイルを作成するドライブを指定します。BT-600シリーズでは使用できません。	2/5	2	可

※1 Initializeメソッドでは設定値は変更されません。

ポイント

LAN通信ユニット経由で通信する場合、LAN通信ユニット本体の設定をmode1に設定しておく必要があります。

メソッド

名称	説明	引数	戻り値
Initialize	各プロパティを初期化します。	なし	なし
Listen	待ち受けを開始します。 FTPS使用時にListenメソッドは利用できません。	なし	なし
Open	FTPセッションを開始します。	remoteIp:接続先のIPアドレス username:ユーザ名※1 password:パスワード	FTPサーバからの応答コード※2
Quit	FTPセッションを終了します。	なし	FTPサーバからの応答コード※2
PutFile	ファイルを送信します。	localfileName: 端末内でのファイル名 (ドライブ番号:ファイル名) remoteFileName: 接続先で保存するファイル名	FTPサーバからの応答コード※2
GetFile	ファイルを受信します。	remotefileName: 接続先のファイル名 localFileName: 端末内でのファイル名 (ドライブ番号:ファイル名)	FTPサーバからの応答コード※2
Rename	接続先のファイル名を変更します。	oldFileName: 変更するファイル名 newFileName: 新しいファイル名	FTPサーバからの応答コード※2
Delete	接続先のファイルを削除します。	fileName:ファイル名	FTPサーバからの応答コード※2
Pwd	接続先の現在のパスを取得します。	なし	パス false:エラー終了
Cwd	接続先でディレクトリを移動します。	path:移動先のパス	FTPサーバからの応答コード※2
Dir	ファイル一覧を取得します。	fileName:取得結果を出力するファイル名 (ドライブ番号:ファイル名)	FTPサーバからの応答コード※2
Command	FTPサーバへのリモートコマンドを実行します。※3 戻り値はcommandResultプロパティで取得します。	command: 実行するコマンド	FTPサーバからの応答コード※2

※1 ユーザ名、パスワードの設定は、ハンディターミナルがFTPサーバになる場合、設定できません。

※2 サーバからの応答コードには、下記のコードも含まれます。

応答コード	説明
-1	btRfOpen関数が実行されていない
-11	ファイル通信に失敗した
-12	タイムアウトした
-13	キャンセルされた
-14	ローカルファイルが見つからない
-15	通信経路が切断された

※3 応答コードのみを返すコマンドと、SIZE コマンドに対応していますが、それ以外のコマンドには対応していません。

■FTPサーバ上のファイルサイズの確認方法

Commandメソッドを用いてリモートコマンドでSIZEを指定することで、FTPサーバ上のファイルサイズを取得することが可能です。ハンディターミナル上のファイルサイズと比較することで、正しくFTP送受信ができたかのチェックとして利用できます。

ret = FTP:Command("SIZE リモートファイル名")
str = FTP:commandResult // FTPサーバからの応答コードとファイルサイズが戻ります。

使用例

FTPコントロールを使用したサンプル

```
Method main()
        strItem = "1:FTP Server|2:FTP Client|3:FTPS"
        menupos = 1
Begin
        While( 1 )
                menupos = 1
                // メニュー画面の表示
                menupos = Handy:ShowMenu("FTP",strItem,nil,nil,nil,menupos)
                Select Case menupos
                case 1
                        // FTP Serverのサンプルメソッドを実行
                        FtpServer()
                case 2
                        // FTP Clientのサンプルメソッドを実行
                        FtpClient()
                case 3
                        // FTPSのサンプルメソッドを実行
                        Ftps()
                End Select
        Wend
End Method
```

例1 FTP Serverを実行する

```
Method FtpServer()
        strTermIP = "192.168.100.100" // 端末IP
        strHostIP = "192.168.100.001" // ホストIP
Begin
        // LANポートをオープンする
        Network:Open("sync", "LanCradle")

        // FTPオプションの設定
        FTP:Initialize()
        FTP:timeOut = 30
        FTP:cancelKey = "C"
        FTP:resultFileName = "2:FTP_Listen.log"

        // FTPサーバを開始する
        FTP:Listen()

        // LANポートを閉じる
        Network:Close("LanCradle")
        Handy:ShowMessageBox("ファイル通信が完了しました。", "confirm")
End Method
```

例2 FTP Clientを実行する

```
Method FtpClient()
	strTermIP = "192.168.100.100" // 端末IP
	strHostIP = "192.168.100.001" // ホストIP
	ret
	remoteFileName1 = "dirinfo.txt"
	remoteFileName2 = "text.txt"
	localFileName = "2:dirinfo.txt"
Begin
	// LANポートをオープンする
	Network:Open("sync", "LanCradle")

	// FTPオプションの設定
	FTP:Initialize()
	FTP:timeOut = 30
	FTP:cancelKey = "C"
	FTP:resultFileName = ""

	// FTPに接続
	ret = FTP:Open(strHostIP, "keyence", "keyence")
	If ret <> 230 Then
		// エラーメッセージを表示する
		Handy:ShowMessageBox("FTPの接続に失敗", "confirm")
		// LANポートを閉じる
		Network:Close("LanCradle")
		Return()
	End If

	// リモートのディレクトリパスを設定
	ret = FTP:Cwd("/")
	If ret <> 250 Then
		Handy:ShowMessageBox("リモートのディレクトリパスの設定に失敗", "confirm")
	End If

	// リモートのディレクトリパスを取得
	ret = FTP:Pwd()
	If ret is False Then
		Handy:ShowMessageBox("リモートのディレクトリパスの取得に失敗", "confirm")
	Else
		Handy:ShowMessageBox(ret, "confirm")
	End If

	// リモートのディレクトリ情報を取得する
	ret = FTP:Dir(localFileName)
	If ret <> 226 Then
		Handy:ShowMessageBox("ディレクトリ情報の取得に失敗", "confirm")
		// FTPを終了する
		FTP:Quit()
		// LANポートを閉じる
		Network:Close("LanCradle")
		Return()
	End If
```

```
        // ファイルを送信する
        ret = FTP:PutFile(localFileName, remoteFileName1)
        If ret <> 226 Then
                Handy:ShowMessageBox("ファイルの送信に失敗", "confirm")
                // FTPを終了する
                FTP:Quit()
                // LANポートを閉じる
                Network:Close("LanCradle")
                Return()
        End If

        // ファイル名を変更する
        ret = FTP:Rename(remoteFileName1, remoteFileName2)
        If (ret <> 250) And (ret <> 450) Then
                Handy:ShowMessageBox("ファイル名の変更に失敗", "confirm")
                // FTPを終了する
                FTP:Quit()
                // LANポートを閉じる
                Network:Close("LanCradle")
                Return()
        End If

        // ファイルを取得する
        ret = FTP:GetFile(remoteFileName2, localFileName)
        If ret <> 226 Then
                Handy:ShowMessageBox("ファイルの受信に失敗", "confirm")
                // FTPを終了する
                FTP:Quit()
                // LANポートを閉じる
                Network:Close("LanCradle")
                Return()
        End If

        // ファイルを削除する
        ret = FTP:Delete( remoteFileName2 )
        If ret <> 226 Then
                Handy:ShowMessageBox("ファイルの削除に失敗", "confirm")
                // FTPを終了する
                FTP:Quit()
                // LANポートを閉じる
                Network:Close("LanCradle")
                Return()
        End If

        // FTP接続を終了する
        FTP:Quit()

        // LANポートを閉じる
        Network:Close("LanCradle")
        Handy:ShowMessageBox("終了しました。", "confirm")
End Method
```

例3 FTPSを実行する

```
Method Ftps()
	strTermIP
	strHostIP
	ret
	localFileName = "2:dirinfo.txt"
Begin
	// FTPSのパラメータを設定する
	FTP:enableFTPS = "TLS"// 暗号方式:TLS
	FTP:ftpsDataEncrypt = true// データチャネル暗号化:ON
	FTP:ftpsVerify = false// 証明書認証:OFF

	/*- 証明書を使用する場合(ここから) -*/
	// FTP:ftpsCerFilename = "1:__rootcertOK"
	/*- 証明書を使用する場合(ここまで) -*/

	// LANポートをオープンする
	Network:Open("sync", "LanCradle")

	// FTPオプションの設定
	FTP:Initialize()
	FTP:timeOut = 30
	FTP:cancelKey = "C"
	FTP:resultFileName = ""

	// FTPに接続
	ret = FTP:Open(strHostIP,"keyence","keyence")
	If ret <> 230 Then
		// エラーメッセージを表示する
		Handy:ShowMessageBox("FTPの接続に失敗", "confirm")
		// LANポートを閉じる
		Network:Close("LanCradle")
		Return()
	End If

	// リモートのディレクトリパスを設定
	ret = FTP:Cwd("/")
	If ret <> 250 Then
		Handy:ShowMessageBox("リモートのディレクトリパスの設定に失敗", "confirm")
	End If

	// リモートのディレクトリパスを取得
	ret = FTP:Pwd()
	If ret is False Then
		Handy:ShowMessageBox("リモートのディレクトリパスの取得に失敗", "confirm")
	Else
		Handy:ShowMessageBox(ret, "confirm")
	End If

	// リモートのディレクトリ情報を取得する
	ret = FTP:Dir(localFileName)
	If ret <> 226 Then
		Handy:ShowMessageBox("ディレクトリ情報の取得に失敗", "confirm")
		// FTPを終了する
		FTP:Quit()
		// LANポートを閉じる
		Network:Close("LanCradle")
		Return()
	End If

	// FTP接続を終了する
	FTP:Quit()

	// LANポートを閉じる
	Network:Close("LanCradle")
	Handy:ShowMessageBox("終了しました。", "confirm")
End Method
```

Bluetooth	Bluetoothを使用した通信の動作を設定します。

プロパティ

名称	説明	範囲	初期値	代入
isOpen	Bluetoothでの通信が可能かどうかを確認します。 📖「ポートの状態(isOpenプロパティ)」(7-275ページ)	true:使用可能 false:使用不可	false	不可
pinCode	PINコードを設定します。	文字列 (ASCII文字1～16バイト)	端末の設定値	可
localDeviceName	Bluetoothのデバイス名を設定します。	文字列 (最大30バイト)	端末の設定値	可
commMod	Bluetooth通信の通信方式を設定します。	"PPP" "SPP" "HID"※1	"PPP"	可
localBDAddr	BDアドレス(Bluetooth Device Address)を取得します。	文字列(ASCII文字12バイト固定)	固有値	不可
remoteBDAddr	接続先のBDアドレス(Bluetooth Device Address)を取得します。	文字列(ASCII文字12バイト固定)	端末の設定値	可
lastTargetBDAddr	最後に接続した(している)相手機器のBD アドレスを取得します。 BT-600シリーズのみ有効です。	文字列(ASCII文字12バイト固定)	—	不可
securityMode	セキュリティモード	"DISABLE":なし "AUTH":認証あり "BOTH":認証･暗号化あり	端末の設定値	可
error	エラー情報 エラー原因が格納されます。	文字列データ 📖「エラー情報(errorプロパティ)」(7-275ページ)	""	不可
hidLocalcod	ローカルクラスオブデバイスを設定します。 BT-1000/1500/600シリーズのみ有効です。	1344:接続時 0:ペアリング時	1344	可
hidMode	HIDモードを設定します。 BT-1000/1500/600シリーズのみ有効です。	1:接続時 2:ペアリング時(Win) 3:ペアリング時(iPad)	0	可

イベントプロパティ

名称	発生タイミング	sender
onConnect	Bluetooth通信で接続したとき	"SPP"/"PPP"
onOutrange	Bluetooth通信で電波の届かない圏外のとき	"SPP"/"PPP"
onRefused	Bluetooth通信で接続が拒否されたとき	"SPP"/"PPP"

メソッド

名称	説明	引数	戻り値
Initialize	各プロパティを初期化します。	なし	なし
Open※	Bluetoothを使用可能に設定します。	mode:"sync":同期	true:正常終了 false:エラー終了
Close	Bluetoothを使用不可に設定します。	なし	なし
Search	周辺のBluetooth機器検索を行います(最大9機器まで)。 ※結果はGetSearchResultメソッドを使用し取得して下さい。	cancelKey: nil:キャンセルキー無効 "C":キャンセルキー "ENT":エンターキー "F1"/"F2"/"F3": ファンクションキー "ANY":任意キー	検索された機器数 (0～9) false:エラー終了 ※失敗要因はerror プロパティ参照
GetSearchResult	Searchメソッドの検索結果を取得します。	index: 取得する機器番号(1～9) kind: 取得したいデータ種 ("addr" / "name")	読込文字列 (最大32バイト) false:エラー終了 ※失敗要因はerror プロパティ参照

※ 同期モードでOpenメソッドを実行すると、Bluetoothが使用できるようになるまで戻り値を返しません。

■ポートの状態(isOpenプロパティ)

無線で通信をおこなうときには、プロパティに通信条件を設定してから、メソッドを実行します。
下表に、「isOpen」プロパティの値によって利用できることと、できないことを示します。

isOpenの値	プロパティの参照	プロパティの設定	メソッドの実行
false	○	○	×
true	○	△	○

○:利用可能です。
△:設定後一旦falseにして再度trueにしたときに、有効になります。
×:利用できません。

■エラー情報(errorプロパティ)

エラー情報	説明
"BLUETOOTH_ERR"	無線接続失敗
"BLUETOOTH_ALREADYOPEN"	既に無線接続されている
"BLUETOOTH_TIMEOUT"	タイムアウト

使用例

例1　SPPを使用したBluetoothの使用方法

```
Method SPP()
    Begin

        // Bluetoothデバイスの設定をおこないます
        With Bluetooth
            :commMod = "SPP"
            If :Open("Sync") is false Then
                Handy:ShowMessageBox( "オープン失敗", "confirm" )
                Return(-1)
            End If
        End With

        // シリアル通信を開始します
        With Serial< "Bluetooth" >
            :timeOut = 100
            :cancelKey = "C"
            :enable = true
            :Write("送信データ")
        End With
        Bluetooth:Close()
End Method
```

ポイント

SPP での通信は厳密にはデータ保証されていませんので、上位アプリ等でデータチェックや再送の仕組みを構築することをお勧めします。

例2　DUNを使用したBluetoothの使用方法

```
Method DUN()
    Begin

        // PPPの設定を行います
        With PPP
            :telNumber = "1234567890"
            :username = "USER"
            :password = "PASS"
            :pap = false
            :chap = false
            :timeOut=10
        End With
```

```
        // Bluetoothデバイスの設定をおこないます
        With Bluetooth
            :commMod = "PPP"

            If :Open( "sync" ) is false Then
                Handy:ShowMessageBox( "オープン失敗", "confirm" )
                return(-1)
            End If
        End With

    // SocketやFTPによる通信をおこないます
    //(省略)
    Bluetooth:Close()
End Method
```


例3

HID接続

ローカルクラスオブデバイス、HIDモードを指定してBluetoothポート、シリアルポートの順に開きます。

```
Bluetooth:hidLocalcod=1344
Bluetooth:hidMode=1
Bluetooth:commMod = "HID"
ret=Bluetooth:Open("sync")
Serial< "Bluetooth" >:enable = True
```

HID切断

シリアルポート、Bluetoothポートの順に閉じます。

```
Serial< "Bluetooth" >:enable = False
Bluetooth:Close()
```

ペアリング

接続前にペアリングをおこないます。ペアリング情報が消えた場合、再ペアリングが必要です。

```
Bluetooth:commMod = "HID"
Bluetooth:hidLocalcod = 0
Bluetooth:hidMode = 3          // iPadの場合 (Windowsの場合、Bluetooth:hidMode = 2)


ret=Bluetooth:Open("sync")
```

```
If ret is True Then
        return(ret)
EndIf
```

データ送信

サンプルプログラム Writeメソッドを使用して、各キーを送る例になります。

“a”キーを指定する場合：Write(0x00,0x04,0,0,0,0,0,True)
Tabキーを指定する場合：Write(0x00,0x2b,0,0,0,0,0,True)
Shift+Tabを指定する場合：Write(0x02,0x2b,0,0,0,0,0,True)

●Writeメソッドの引数説明（詳しくは下記ソースをご参照ください）
第1引数：Alt,Shift,Ctrl設定　※HIDキーコードパケットフォーマット参照
第2～7引数：送信したいキーコード　※キーコード参照
第8引数：True（送信ウェイト有）

```
Method Write(AltShiftCtrl,KEY1,KEY2,KEY3,KEY4,KEY5,KEY6,isWait)
 key_make[12] ={"0x0c","0x00","0xa1","0x01","0x00","0x00","0x00","0x00","0x00","0x00","0x00","0x00"}
        bin_make=""
        bin_break = "¥x0c¥x00¥xa1¥x01¥x00¥x00¥x00¥x00¥x00¥x00¥x00¥x00"

Begin
        // パケットデータの組立て
        key_make[4]＝AltShiftCtrl
        key_make[6]＝KEY1
        key_make[7]＝KEY2
        key_make[8]＝KEY3
        key_make[9]＝KEY4
        key_make[10]＝KEY5
        key_make[11]＝KEY6
        bin_make＝key_make.PackBin
        // データ送信
        Serial＜"Bluetooth"＞:Write(bin_make)
        If isWait is True Then
                Handy:Wait(1,false)
        EndIf

        //キーを離したことを通知する
        Serial＜"Bluetooth"＞:Write(bin_break) End Method
End Method
```

【メモ】キーコード、HIDキーコードパケットフォーマットは、次ページをご参照ください。

■HIDキーコードパケットフォーマット

HIDキーコードパケットフォーマットでBTから、データを送信します。

HIDキーコードパケットフォーマット:

Length		Type	ID	Alt,Shift,Ctrl	Padding	KEY1	KEY2	KEY3	KEY4	KEY5	KEY6
0C	00	A1	01	XX	00	XX	XX	XX	XX	XX	XX

XXについては、送信したいキーコードを設定します。それ以外は固定値になります。

Alt,Shift,Ctrl:

7	6	5	4	3	2	1	0
0	0	0	0	0	Left Alt	Left Shift	Left Control

NULLコードパケットフォーマット

Length		Type	ID	Alt,Shift,Ctrl	Padding	KEY1	KEY2	KEY3	KEY4	KEY5	KEY6
0C	00	A1	01	00	00	00	00	00	00	00	00

キーコード送信後、NULL コードを送る必要があります。NULL コードを送らない場合、通信相手側はキーの長押しと認識します。

使用例

各送信キーについて、指定する値になります。

送信キー	Alt,Shift,Ctrl	KEY1	KEY2～6
“a”	0x00	0x04	0x00
“A”	0x02	0x04	0x00
“1”	0x00	0x1E	0x00
Tab	0x00	0x2B	0x00
Shift+Tab	0x02	0x2B	0x00

■キーコード表

文字	ASCIIｺｰﾄﾞ	ｷｰｺｰﾄﾞ
0	0x30	0x27
1	0x31	0x1E
2	0x32	0x1F
3	0x33	0x20
4	0x34	0x21
5	0x35	0x22
6	0x36	0x23
7	0x37	0x24
8	0x38	0x25
9	0x39	0x26
a	0x61	0x04
b	0x62	0x05
c	0x63	0x06
d	0x64	0x07
e	0x65	0x08
f	0x66	0x09
g	0x67	0x0A
h	0x68	0x0B
i	0x69	0x0C
j	0x6a	0x0D
k	0x6b	0x0E
l	0x6c	0x0F
m	0x6d	0x10
n	0x6e	0x11
o	0x6f	0x12
p	0x70	0x13
q	0x71	0x14
r	0x72	0x15
s	0x73	0x16
t	0x74	0x17
u	0x75	0x18
v	0x76	0x19
w	0x77	0x1A
x	0x78	0x1B
y	0x79	0x1C
z	0x7a	0x1D

キー	キーコード
ENTER	0x28
TAB	0x2B
ESC	0x29
BACKSPACE	0x2A
TAB	0x2B
SPACE	0x2C
DELETE	0x4C
→	0x4F
←	0x50
↓	0x51
↑	0x52

制御キー	
CTRL	0x01
SHIFT	0x02
ALT	0x04
GUI(command)	0x08

▶ 重 要

PPP	Bluetoothを使用した無線通信(DUN)やモデムでのPPP接続の設定をおこないます。

PPPコントロールの設定は、BluetoothコントロールやNetworkコントロールのOpenメソッドを実行する前におこなってください。

プロパティ

名称	説明	範囲	初期値	代入
telNumber	接続先の電話番号を設定します。	文字列(1～16バイト)	BTシリーズの設定値	可
username	認証で使用するユーザー名を設定します。	文字列(最大30バイト)	BTシリーズの設定値	可
password	認証で使用するパスワードを設定します。	文字列(最大20バイト)	BTシリーズの設定値	可
pap	PAP(Password Authentication Protocol)認証を使用するかどうか設定します。	true:使用する false:使用しない	BTシリーズの設定値	可
chap	CHAP(Challenge Handshake Authentication Protocol)認証を使用するかどうかを設定します。	true:使用する false:使用しない	BTシリーズの設定値	可
timeOut	タイムアウト時間 メソッド実行時のタイムアウト時間を設定します。	0～60(秒)	BTシリーズの設定値	可

Printer	キーエンス指定のプリンタと通信するための設定をします。

- 使用できるプリンタは以下のプリンタです。
- BT-PR1/PR2
- サトー社製ラパン、プチラパン
- 東芝テック社製ポータブルプリンタ・ポケプリ
- 東北リコー社製MP-2200(Bluetooth未対応)

▶ 重 要

- **キーエンス指定のプリンタには、Printerコントロールを使用することができます。キーエンス指定以外のプリンタにはこのコントロールは対応していません。**
- **各プリンタとの接続にあたっては、通信速度、印字コマンドなど、各プリンタの仕様書を確認してください。印字コマンドをファイル化して、「SendCommand」でプリンタに送信することで印字します。**
- **このコントロールのメソッドで操作するファイルを「File」コントロールで操作していた場合は、必ず「File:Initialize」メソッドで初期化してください。**

プロパティ

名称	説明	範囲	初期値	代入
printerType	プリンタの種類を設定します。	1:BT-PR1 2:ラパン、プチラパン (PT200/PT400 互換モード)※3 3:ポケプリラベル発行モード 4:ポケプリレシート発行モード 5:ポケプリLEモード 6:MP2200(ラベル) 7:MP2200(レシート) 8:BT-PR2(BPLモード)※4	1	可
portType	ポート種別を設定します。※1	1:赤外線通信(IrDA)ポート 2:RS-232Cポート 3:無線ポート※5 4:Bluetooth※2	1	可
baudRate	通信速度(bps)を設定します。	9600/19200/38400/57600/115200	19200	可
cancelKey	通信をキャンセルするときに押すキーを設定します。	"C":キャンセルキー "ENT":エンターキー "F1"/"F2"/"F3":ファンクションキー "ANY":任意キー	"C"	可
ipAddress	無線プリンタ(アドホックモード)のIPアドレス 無線LAN搭載機種のみ有効です。	XXX.XXX.XXX.XXX (15バイト固定)	nil	可
port	無線プリンタ(アドホックモード)のポート番号 無線LAN搭載機種のみ有効です。	0~65535	nil	可

※1 BT-PR1は、赤外線通信(IrDA)ポートのみです。
※2 「4」(Bluetooth)に設定した場合には、Network または Bluetooth コントロールで、Bluetooth通信の設定が必要です。
※3 送信データの後ろにCRCを付加しています。
※4 送信データの後ろにCRCを付加していません。
※5 「3」(無線ポート)に設定した場合には、Network、WLAN または CommRF コントロールで、無線通信の設定が必要です。アドホックモードのみサポートしています。 無線LAN搭載機種のみ有効です。

メソッド

名称	説明	引数	戻り値
Initialize	各プロパティを初期化します。	なし	なし
SendCommand	プリンタにコマンドを送信します。	fileName： コマンドが記述されているファイル名(ドライブ番号:ファイル名)	true：正常終了 false：エラー終了
CreateBMPCommand	プリンタにビットマップデータを送信するためのコマンドをファイルとして生成します。※1	bmpFileName： ビットマップファイル名※2 (ドライブ番号:ファイル名) dstFileName： コマンドを記述するファイル名 (ドライブ番号:ファイル名※3) posx：ビットマップの表示位置X座標 posy：ビットマップの表示位置Y座標	true：正常終了 false：エラー終了

※1 ビットマップを印字するときは、このコマンドのファイルと、他のコマンドが記述されたファイルをFileSystem：Cat()などで連結して、SendCommandでプリンタへ送信します。
※2 モノクロのビットマップファイルのみ指定可能です。
※3 ドライブ2のファイルのみ指定可能です。

使用例

BT-PR2で印刷する場合の、「Printer」コントロールを使用したパッケージを定義します。

```
Package PrinterSample
    esc = "\x1b"
    Method Init()
    Begin
        With Printer
            // プリンタの種類、ポート、ボーレードなどのプロパティを設定
            :printerType = 8
            :portType = 1
            :baudRate = 115200
            :cancelKey = "C"
            // ビットマップデータを送信するための「sample.bmp」コマンドを
            //「printcmd_bmp.bin」ファイルとして生成
            :CreateBMPCommand("2:sample.bmp", "2:printcmd_bmp.bin", 1, 1)
        EndWith

        With File
            // 開始コマンドを「printcmdhead.bin」ファイルとして作成
            :name = "2:printcmdhead.bin"
            :Puts(esc,"create")
            :Puts("A","append") //開始コマンド
            :Initialize()
            // 終了コマンドを「printcmdend.bin」ファイルとして作成
            :name = "2:printcmdend.bin"
            :Puts(esc,"create")
            :Puts("Q0001","append") //枚数指定コマンド
            :Puts(esc,"append")
            :Puts("Z","append") //終了コマンド
            :Initialize()
        End With
```

```
        // printcmdhead.bin, printcmd_bmp.bin, printcmdend.bin の順で
        // ファイルを連結
        With FileSystem
            :Cat("2:printcmdhead.bin","2:printcmd_bmp.bin","2:printcmd0.bin")
            :Cat("2:printcmd0.bin","2:printcmdend.bin","2:printcmd.bin")
        End With
    End Method

    // コマンドを記述した「printcmd.bin」ファイルを送信
    Method Print()
    Begin
        If Printer:SendCommand("2:printcmd.bin") Then
            Handy:ShowMessageBox("印刷に成功しました","confirm")
        Else
            Handy:ShowMessageBox("印刷に失敗しました","confirm")
        EndIf
    End Method

End Package
```

ZGN	全銀TCP/IP手順でのファイル送受信をおこないます。

ポイント

BT-1000/1500シリーズのみ使用可能です。
全銀TCP/IPを用いたシステムを構築する場合、別途ライセンスが必要となります。

プロパティ

名称	説明	範囲	初期値	代入
cancelKey	通信キャンセルキー	キーコード	"C"	可
maxTextSize	最大伝送テキスト長	32～32768(バイト)	2048	可
timeout	無通信タイマー	0～999(秒)	30	可
destIP	接続先IPアドレス	XXX.XXX.XXX.XXX(15バイト固定)	""	可
destPortNo	接続先ポート番号	1～65535	5020	可
localPortNo	ローカルポート番号	1～65535	0	可
tcpConnectRetry	TCP接続リトライ回数	0～99(回)	0	可
ttcRounduplnit	TTCカウンタラウンドアップ初期値	0～1	1	可
defineMN	連続受信回数	0～15(回)	15	可
zeroByteFile	0バイトファイル送信許可フラグ	true:有効 false:無効	true	可
transDateCheck	回答ファイル制御電文の日付チェック	true:有効 false:無効	false	可
ttcBranch	TTC電文区分	"BASIC"/"PC"	"PC"	可
hostcord	相手センタ確認コード	7バイトまでのHEX文字列 (表記は14バイトになります)	""	可
termcord	当方センタ確認コード	7バイトまでのHEX文字列 (表記は14バイトになります)	""	可
password	パスワード	6バイトまでのHEX文字列 (表記は12バイトになります)	""	可
ext	通信制御拡張エリア	34バイトまでのHEX文字列 (表記は68バイトになります)	""	可
fileName	送受信ファイル名	256バイトまでの文字列	""	可
mode	通信モード	"PUT":送信 "GET":上書き受信 "APPEND":追加受信	"GET"	可
branch	電文区分	"START":開始要求 "RESTART_FILE": 再送要求(ファイル) "RESTART_TEXT": 再送要求(テキスト)	"START"	可
pack	伝送時の圧縮	true:圧縮する false:圧縮しない	false	可
derimita	デリミタ編集	nil/"CR"/"CRLF"/"EOF"/"LF"	nil	可
changeCodePtn	コード変換パターン	nil/"EBCDIC to JIS8"/"JIS8 to EBCDIC"	nil	可
cycle	サイクル番号	0～99	0	可
vLRec	可変長レコード指定	true:圧縮する false:圧縮しない	false	可
recSize	レコード長	1～最大テキスト長-5(バイト)	120	可
zFileName	全銀ファイル名	12バイトまでのHEX文字列 (表記は24バイトになります)	""	可

名称	説明	範囲	初期値	代入
fileAccessKey	ファイルアクセスキー	6バイトまでのHEX文字列 (表記は12バイトになります)	""	可
fileSub	ファイル名補助情報	17バイトまでのHEX文字列 (表記は34バイトになります)	""	可
fileExt	ファイル制御拡張エリア	14バイトまでのHEX文字列 (表記は28バイトになります)	""	可
resultFilename	通信結果記録ファイル名	ドライブ番号:ファイル名	nil	可
blocking	ブロッキング	true:有効 false:無効	true	可
error	エラー情報 エラー原因が格納されます。	文字列データ 「エラー情報(errorプロパティ)」(7-287ページ)	""	不可

メソッド

名称	説明	引数	戻り値
Transfer	全銀TCP/IP手順でファイルを送受信します。	なし	true:正常終了 false:エラー終了

■エラー情報(errorプロパティ)

エラー情報	説明
"REFUSED"	拒否された
"CANCELED"	キャンセルされた
"TIMEOUT"	タイムアウト
"NETDOWN"	通信経路が切断された
"PARAMERROR"	設定値のエラー
-416	[開局･閉局時のエラー]:電文区分エラー
-417	[開局･閉局時のエラー]:相手センタコードエラー
-418	[開局･閉局時のエラー]:当方センタコードエラー
-419	[開局･閉局時のエラー]:サービス時間帯エラー
-420	[開局･閉局時のエラー]:パスワードエラー
-421	[開局･閉局時のエラー]:アプリケーションID エラー
-422	[開局･閉局時のエラー]:モードエラー
-423	[開局･閉局時のエラー]:モード変更不可
-516	[開始･終了時のエラー]:電文区分エラー
-517	[開始･終了時のエラー]:ファイル名エラー
-518	[開始･終了時のエラー]:ファイルアクセスキーエラー
-519	[開始･終了時のエラー]:テキスト数エラー
-520	[開始･終了時のエラー]:レコード数エラー
-521	[開始･終了時のエラー]:レコード長エラー
-522	[開始･終了時のエラー]:二重ファイル伝送
-523	[開始･終了時のエラー]:ファイルなし
-524	[開始･終了時のエラー]:レコードID エラー
-525	[開始･終了時のエラー]:データ圧縮ID エラー
-553	[開局･閉局時のエラー]:その他のエラー
-653	[開始･終了時のエラー]:その他のエラー

使用例

```
Method main()
        strHostIP = "192.168.100.001"
        KEY_ENT = 0x00800000
        ret
        i
Begin
        // サンプルファイルを作成します。
        File:Initialize()
        File:name = "2:sample.txt"
        File:Puts("1234567890abcdefghijklmnopqrstuvwxyg","create")
        For i = 1 to 30
                File:Puts("1234567890abcdefghijklmnopqrstuvwxyg","append")
        Next
        File:Close()

        // 全銀手順のパラメータを設定します。
        With ZGN
                :cancelKey = "C"                        // 通信キャンセルキー
                :maxTextSize = 2048                     // 最大伝送テキスト長
                :timeout = 30                                   // 無通信タイマー
                :destIP = strHostIP             // 接続先IPアドレス
                :destPortNo = 5020                      // 接続先ポート番号
                :localPortNo = 0                        // ローカルポート番号
                :tcpConnectRetry = 0                // TCP接続リトライ回数
                :ttcRoundupInit = 1                     // TTCカウンタラウンドアップ初期値
                :defineMN = 0                                   // 連続受信回数
                :zeroByteFile = true                // 0バイトファイル送信許可フラグ
                :transDateCheck = false             // 回答ファイル制御電文の日付チェック
                :ttcBranch = "PC"                       // TTC電文区分
                :hostcord = "00000000000000"// 相手センタ確認コード
                :termcord = "00000000000000"// 当方センタ確認コード
                :password = "000000000000"// パスワード
                :ext = "OOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOO
                        OOOOOOOOOOOOOOOOOOOOOO"// 通信制御拡張エリア
                :branch = "START"                       // 電文区分
                :pack = false                                   // 伝送時の圧縮
                :derimita = nil                                 // デリミタ編集
                :blocking = true                        // ブロッキング
                :changeCodePtn = "EBCDIC to JIS8"// コード変換パターン
                :cycle = 0                                      // サイクル番号
                :vLRec = true                                   // 可変長レコード指定
                :recSize = 120                                  // レコード長
                :zFileName = "F5F0F2F0F0F0F0F0F0F1F0F0"// 全銀ファイル名
                :fileAccessKey = "000000000000"// ファイルアクセスキー
                :fileSub = "00000000000000000000000000000000000000"// ファイル名補助情報
                :fileExt = "00000000000000000000000000000"// ファイル制御拡張エリア
                :resultFilename = "2:zgn_result.txt"// 通信結果記録ファイル名
        EndWith

        With Screen
                :Clear()
                :posx = 1 :posy = 1 :multiColumn = 2
                :OutputText("受信(GET:Append)します。準備ができたら[ENT]キーを押してください")
        End With
```

```
// キー入力待ち
Handy:KeyWaitEx(KEY_ENT, 0)

// 全銀手順でファイル送信します。
ZGN:mode = "PUT"                                    // 通信モードを送信にする。
ZGN:fileName = "2:sample.txt"

// ポートをオープンします。
// モデムPPPを使用する場合には、この前にPPPの設定を行います。
If Network:Open("sync", "Modem Cradle PPP") is false Then
        return(nil)
EndIf

ret = ZGN:Transfer()                                // 全銀手順でファイル転送します。

If ret is false Then
        Handy:ShowMessageBox("送信(PUT)に失敗しました。(" & ZGN:error & ")", "confirm")
Else
        Handy:ShowMessageBox("送信(PUT)に成功しました", "confirm")
End If
Network:Close(nil)

With Screen
        :Clear()
        :posx = 1 :posy = 1 :multiColumn = 2
        :OutputText("受信(GET:Override)します。準備ができたら[ENT]キーを押してください")
End With

// キー入力待ち
Handy:KeyWaitEx(KEY_ENT, 0)

// 全銀手順でファイル受信します。
ZGN:mode = "GET"                                    // 通信モードを受信(上書き)にします。
ZGN:fileName = "2:sample2.txt"

// ポートをオープンします。
If Network:Open("sync", "Modem Cradle PPP") is false Then
        return(nil)
EndIf

ret = ZGN:Transfer()                                // 全銀手順でファイル転送します。

If ret is false Then
        Handy:ShowMessageBox("受信(GET:Override)に失敗しました。(" & ZGN:error & ")", "confirm")
Else
        Handy:ShowMessageBox("受信(GET:Override)に成功しました", "confirm")
End If
Network:Close(nil)

With Screen
        :Clear()
        :posx = 1 :posy = 1 :multiColumn = 2
        :OutputText("受信(GET:Append)します。準備ができたら[ENT]キーを押してください")
End With

// 全銀手順でファイル受信します。
ZGN:mode = "APPEND"                                 // 通信モードを受信(追加書き込み)にします。
ZGN:fileName = "2:sample2.txt"
```

```
        // ポートをオープンします。
        If Network:Open("sync", "Modem Cradle PPP") is false Then
                return(nil)
        EndIf

        ret = ZGN:Transfer()                    // 全銀手順でファイル転送します。

        If ret is false Then
                Handy:ShowMessageBox("受信(GET:Append)に失敗しました。(" & ZGN:error & ")",
"confirm")
        Else
                Handy:ShowMessageBox("受信(GET:Append)に成功しました", "confirm")
        End If
        Network:Close(nil)
End Method
```

CommRF	無線でサーバPCなどと通信をおこなうときの設定をします。

▶ 重 要

無線LAN搭載機種のみ使用可能です。

- **以下のプロパティ、メソッドは、旧バージョンの「BT-900用通信ライブラリ」と通信するために用意しています。**

プロパティ	retryMax、retryTimer、sendMode、sendStatus、connectTimer、cancelKey
メソッド	Sendstring、RecvString、SendData、RecvData、SetDateTime、PutFile、PutLogFile、Listen

上記のプロパティ、メソッドを使用する場合、「RemoteDB」、「RemoteAccess」、「RemoteFile」、「RemoteUD」を使用して、「リクエストマネージャ」と通信することはできません。

- **このコントロールの「SendData」、「RecvData」、「PutFile」メソッドで操作するファイルを、既に「File」コントロールで操作している場合は、メソッドを実行する前に「File：Initialize」メソッドを実行して、「File」コントロールのプロパティを初期化してください。**

プロパティ

名称	説明	範囲	初期値	代入
enable※1	BTシリーズの無線ポートの状態を設定します。 「ポートの状態(enableプロパティ)」(7-295ページ)	true： 無線ポートのオープン false： 無線ポートのクローズ	false	可
localIp※2	BTシリーズのIPアドレス	XXX.XXX.XXX.XXX (15バイト固定)	BTシリーズの設定値	可
subnet※3	BTシリーズのサブネットマスク	XXX.XXX.XXX.XXX (15バイト固定)	BTシリーズの設定値	可
gateway※4	BTシリーズのデフォルトゲートウェイ	XXX.XXX.XXX.XXX (15バイト固定)	BTシリーズの設定値	可
hostIp※5	サーバのIPアドレス	XXX.XXX.XXX.XXX (15バイト固定)	BTシリーズの設定値	可
dataPort	BTシリーズのデータ転送用ポート番号	49152～65535	BTシリーズの設定値	可
ftpPort	BTシリーズのファイル転送用ポート番号	49152～65535	BTシリーズの設定値	可
essId※6	SSID	文字列(0～32バイト)	BTシリーズの設定値	可
wepKey	WEPキー	文字列(13バイト) nil：指定しない	BTシリーズの設定値	可
macaddr	MACアドレス	文字列(12桁固定)	固有値	不可
ssid_adhoc	アドホックモード時のSSID	文字列(0～32バイト)	BTシリーズの設定値	可
channel	アドホックモード時のチャネル	1～14	BTシリーズの設定値	可
wpaMode	WPA認証方法を設定します。 nilの場合にはWEPを使用します。	"PSK" "PSK2" "PEAP" "PEAP2" "EAP_TLS" "EAP_TLS2" nil：WEP	BTシリーズの設定値	可
cipher	WPA暗号化方式を設定します。	"TKIP" "AES"	BTシリーズの設定値	可

名称	説明	範囲	初期値	代入
networkMod	インフラトストラクチャ/アドホックモードを切り替えます。	"infrastructure":インフラストラクチャ "adhoc":アドホック	"infrastructure"	可
error	エラー情報 エラー原因が格納されます。	文字列データ 📖「エラー情報(errorプロパティ)」(7-296ページ)	""	不可

※1 メソッドを実行するときは、enableをtrueに設定します。
プロパティに値を格納するときは、false に設定します。プロパティの参照は、true、false のどちらでもできます。
※2 出荷時のIPアドレスは、"000.000.000.000"に設定されています。
※3 出荷時のサブネットマスクは、"000.000.000.000"に設定されています。
※4 出荷時のデフォルトゲートウェイは、"000.000.000.000"に設定されています。
※5 出荷時のサーバIPアドレスは、"000.000.000.000"に設定されています。
※6 出荷時のSSIDは、""(長さ0の文字列)が設定されています。

■BT-900通信ライブラリ用プロパティ

名称	説明	範囲	初期値	代入
retryMax※1	通信のリトライ回数を設定します。	0～9(回)	BTシリーズの設定値	可
retryTimer※2	通信タイムアウト時間を設定します。	1～10(×100ミリ秒)	BTシリーズの設定値	可
sendMode※3	BTシリーズの設定値からデータを送信するときの同期／非同期のモードを設定します。	0:送信の完了を待たない(非同期) 1:送信の完了を待つ(同期)	0	可
sendStatus※4	BTシリーズの設定値からデータを非同期で送信したときの通信状態を確認します。	1 :送信完了 0 :未実行 -1:送信中 -2:送信失敗 -3:無線圏外	0	不可
connectTimer※5	サーバとの通信確立に使用される、タイムアウト時間を設定します。	1～30(秒)	10	可
cancelKey※6	通信をキャンセルするときに押すキーを設定します。	nil :キャンセルキー無効 "C":キャンセルキー "ENT":エンターキー "F1"/"F2"/"F3":ファンクションキー "ANY":任意キー	"C"	可

※1 出荷時のリトライ回数は、5回に設定されています。
※2 出荷時のタイムアウト時間は、200ミリ秒に設定されています。
※3 「sendData」メソッドのみに有効です。
※4 「sendMode」で「0」(非同期)を設定して「SendData」メソッドを実行した後に、結果を確認するのに使用できます。一度読み出すと、値が「0」(未実行)になります。
※5 「PutFile」、「PutLogFile」、「Listen」メソッドで、サーバとの通信確立に使用されるタイムアウト時間です。
※6 「PutFile」、「PutLogFile」、「Listen」メソッド実行時に、キャンセルを受け付けるキーを指定します。

メソッド

名称	説明	引数	戻り値
Initialize	各プロパティを初期化します。	なし	なし
Ping※1	サーバと接続が確立されているか確認します。	なし	true:正常終了 false:エラー終了
GetRecvLevel	アクセスポイントとの通信品質を確認します。	なし	0～5:通信品質 0:通信不可 5:通信良好 nil
SetWepInfo	WEPに関する設定をします。※2	valid:有効なWEPキー(1～4)※3 wep1:WEPキー1※4 wep2:WEPキー2※4 wep3:WEPキー3※4 wep4:WEPキー4※4	true:正常終了 false:エラー終了
SetSecurityInfo	無線認証に関する設定をします。	type※5 value※5	true:正常終了 false:エラー終了

※1 RemoteAccessコントロールが有効な場合でも使用できます。

※2 wep1に設定した値は、wepKeyプロパティにも反映されます。
WEPを無効にすると、wepKeyプロパティの値はnilに変更されます。

※3 WEPキーを無効にする場合には、「nil」に設定してください。

※4 5バイトまたは13バイトの文字列か、nil(指定なし)を設定します。

※5 設定できる組み合わせは以下のとおりです。

type	value
"PSK"	PSKのネットワークキー
"USER_ID"	PEAPのユーザID
"PASSWORD"	PEAPのパスワード
"ROOT_CA"	PEAP、TLSのルート証明書が格納されたファイル名
"CERT"	TLSのクライアント証明書が格納されたファイル名
"PRIVATE_KEY"	TLSの秘密鍵が格納されたファイル名

■BT-900通信ライブラリ用メソッド

名称	説明	引数	戻り値
SendString	サーバへテキストデータを送信します。 📖「テキストデータの送信、受信(SendString、RecvStringメソッド)」(7-295ページ)	string:送信する文字列 (最大1440バイト)	true:正常終了 false:エラー終了
RecvString	サーバからテキストデータを受信します。 📖「テキストデータの送信、受信(SendString、RecvStringメソッド)」(7-295ページ)	maxLength:最大受信バイト数 (1～1440)	受信した文字列 nil
SendData	サーバへ、指定するファイルに書き込まれたデータを送信します。バイナリ形式のデータを送信できます。 📖「バイナリデータの送信、受信(SendData、RecvDataメソッド)」(7-295ページ)	fileName:送信するファイル名 (ドライブ番号:ファイル名)	true:正常終了 false:エラー終了
RecvData	サーバからデータを受信して、指定するファイルに保存します。バイナリ形式のデータを受信することができます。 📖「バイナリデータの送信、受信(SendData、RecvDataメソッド)」(7-295ページ)	fileName:受信するファイル名 (ドライブ番号:ファイル名)	true:正常終了 false:エラー終了
SetDateTime	サーバの現在時刻を取得して、BTシリーズに同じ時刻を設定します。	なし	true:正常終了 false:エラー終了
PutFile	サーバへファイルを送信します。	localFile:BTシリーズのファイル名 (ドライブ番号:ファイル名) remoteFile:サーバのファイル名 (フォルダ名+ファイル名、 またはファイル名) 省略すると、localFileと同じファイル名で送信します。	true:正常終了 false:エラー終了
PutLogFile	サーバへBTシリーズのドライブ3にあるログファイルを送信します。	なし	true:正常終了 false:エラー終了
Listen	通信の待ち受けを開始します。	なし	true:正常終了 false:エラー終了

■ポートの状態(enableプロパティ)

無線で通信をおこなうときには、プロパティに通信条件を設定してから、メソッドを実行します。
下表に、「enable」プロパティの値によって利用できることと、できないことを示します。

enableの値	プロパティの参照	プロパティの設定	メソッドの実行
true	○	△	○
false	○	○	×

○:利用可能です。
△:設定後一旦falseにして再度trueにしたときに、有効になります。
×:利用できません。

■テキストデータの送信、受信(SendString、RecvStringメソッド)

「SendString」、「RecvString」メソッドは、ASCII文字や漢字を扱う場合に適しています。

- 改行、TABなど基本的な制御コードを含むことは問題ありません。

送信(「SendString」メソッド)

- NULL(0x00)データは送信できません。
- 引数で指定された文字列をそのまま送信します。

受信(「RecvString」メソッド)

受信したデータは、以下のどれかの条件を満たすときにデータの終端とみなして、メソッドの戻り値に格納されます。

- 受信データにNULL(0x00)があらわれた。
- 受信した文字列の長さが最大読み込みバイト数に達した。

「enable」プロパティをtrueにしたときに、受信バッファに古いデータが残っていることがありますので、「RecvString」メソッドを実行(引数に126を設定)して古いデータを空読みすることによって、受信バッファをクリアしてください。

■バイナリデータの送信、受信(SendData、RecvDataメソッド)

「RecvData」、「SendData」メソッドは、NULL(0x00)を含むバイナリデータを扱う場合に適しています。

送信(「SendData」メソッド)

- 送信するバイナリデータは、「File」コントロールの「Write」メソッドで作成することができます。
- 送信するときに、ファイルの内容を読み出して送信します。ファイルは送信しません。
- 最大1440バイトのデータが送信できます。1440バイト目以降のデータは無視されます。

受信(「RecvData」メソッド)

- 受信したバイナリデータのファイルは、「File」コントロールの「Read」メソッドで読み込むことができます。
- 最大1440バイトのデータが受信できます。1440バイト目以降のデータは無視されます。
- 受信バッファに古いデータが残っていることがありますので、「RecvString」メソッドを実行(引数に1440を設定)して古いデータを空読みすることによって、受信バッファをクリアしてください。

■エラー情報(errorプロパティ)

エラー情報	説明
"RF_ERR"	enableがtrueになっていません
"PARAM_ERR"	パラメータエラー
"SET_ERR"	設定失敗
"OUTRANGE"	無線圏外による通信失敗
"NOT_SND"	データ送信が未実行
"CONTINUE"	データ送信中
"SND_NG"	データ送信失敗
"RCV_WAIT"	データ未受信状態
"INCOMPLETE"	処理が失敗しました
"BIGDATA"	ファイルが大きすぎて相手が受け取れませんでした
"NETDOWN"	ネットワークが異常です
"TIMEOUT"	タイムアウト
"CANCELED"	キャンセルされました
"OTHER"	その他
"CONVERT_FAILED"	マスタファイルまたアプリケーションへの変換エラー
"FILENOTFOUND"	ファイルが見つからない
"NOTFOUND"	通信相手が見つからない
"INUSE"	デバイス使用中
"NOTINITIALISED"	未初期化エラー
"INVALID"	内部処理エラー

使用例

例1　サーバとの接続が確立されているかを確認します。

```
Method PingSample()
    Begin

        With CommRF
            :enable = true              // 無線ポートのオープン

            //サーバとの接続が確立されているかを確認
            If :Ping() Then
                                        Handy:ShowMessageBox("接続OK","ok")
            Else
                                        Handy:ShowMessageBox("接続NG","ok")
            EndIf
            :enable = false             // 無線ポートのクローズ
        EndWith
End Method
```

例2　サーバからデータを受信します。

```
Method ReadSample()
    data
    Begin

        With CommRF
            :enable = true                          // 無線ポートのオープン
            :RecvString(126)
                With Screen
                    :posx = 1 :posy = 1 :OutputText("Push Any Key")
                EndWith
            Handy:KeyWait()                         // 入力待ち
            data = :RecvString(5)                   // 5バイトのデータを受信
            Handy:ShowMessageBox(data, "confirm")
            :enable = false                         // 無線ポートのクローズ
        EndWith
End Method
```

例3　サーバへデータを送信します。

```
Method WriteSample()
    Begin

    With CommRF
        :enable = true                              // 無線ポートのオープン
        :SendString("12345")                        // データ"12345"の送信
        :enable = false                             // 無線ポートのクローズ
    EndWith
End Method
```

例4　サーバからの通信の待ち受けを開始します。

```
Method ListenSample()
    Begin

    With CommRF
        :enable = true                              // 無線ポートのオープン
        :connectTimer = 30                          // タイムアウト30秒に設定
        :cancelKey = "ANY"                          // キャンセルキーの設定
        :Listen()                                   // 待ち受け開始
        :enable = false                             // 無線ポートのクローズ
    EndWith
End Method
```

Comm1	赤外線通信(IrDA)ポートを使うデータ通信をおこないます。

▶ 重 要

過去機種との互換のため用意しています。
新規に開発する際は、Serialコントロールを使用してください。

プロパティ

名称	説明	範囲	初期値	代入
enable※	BTシリーズの赤外線通信ポートの状態を設定します。 「ポートの状態(enableプロパティ)」(7-299ページ)	true:赤外線通信ポートのオープン false:赤外線通信ポートのクローズ	false	可
timeOut	タイムアウト時間を設定します。 各メソッドで共通して使用されます。	1～6000(×10ミリ秒)	500	可
baudRate	通信速度(bps)を設定します。	9600/19200/38400/57600/115200	115200	可
stopBits	ストップビットを設定します。	1/2(ビット)	1	可
dataLen	データ長を設定します。	7/8(ビット)	8	可
parity	パリティを設定します。	0:なし 1:奇数 2:偶数	0	可
cancelKey	通信をキャンセルするときに押すキーを設定します。	nil:キャンセルキー無効 "C":キャンセルキー "ENT":エンターキー "F1"/"F2"/"F3":ファンクションキー "ANY":任意キー	"C"	可
error	エラー情報 エラー原因が格納されます。	文字列データ 「エラー情報(errorプロパティ)」(7-299ページ)	""	不可

※ メソッドを実行するときは、trueに設定します。
プロパティに値を格納するときは、falseに設定します。プロパティの参照は、true、falseのどちらでもできます。

メソッド

名称	説明	引数	戻り値
Initialize	各プロパティを初期化します。	なし	なし
Read※	データを受信します。	maxLength:最大受信バイト数（1～4096）	受信データ（バイナリ表現の特殊形式） nil 「バイナリ表現の特殊形式」(7-168ページ)
Write	データを送信します。	string:送信するデータ（バイナリ表現の特殊形式、最大4096バイト） 「バイナリ表現の特殊形式」(7-168ページ)	true:正常終了 false:エラー終了
Clear	送受信に使用するバッファをクリアします。	type:クリアするバッファの種類（1:送信、2:受信、3:送受信）	なし

※• 「enable」プロパティを true に設定してから「Read」メソッドを実行するまでに受信したデータが保存されていることがありますので、必要に応じて「Clear」メソッドで受信バッファをクリアしてください。

• 指定するバイト数に達した、または「timeOut」プロパティに設定した時間を過ぎると、データ受信処理は終了して、メソッドから戻り、スクリプトプログラムの次のステートメントに移ります。

■ポートの状態（enableプロパティ）

赤外線通信（IrDA）をおこなうときには、プロパティに通信条件を設定してから、「Read」、「Write」などのメソッドを実行します。プロパティ値を設定するには、あらかじめ「enable」プロパティを「false」に設定します。

下表に、「enable」プロパティの値によって利用できることと、できないことを示します。

enableの値	プロパティの参照	プロパティの設定	メソッドの実行
true	○	△	○
false	○	○	×

○:利用可能です。

△:設定後一旦falseにして再度trueにしたときに、有効になります。

×:利用できません。

■エラー情報（errorプロパティ）

エラー情報	説明
"ERR_ARG_RANGE1"	第1引数不正
"ERR_ARG_RANGE2"	第2引数不正
"ERR_ARG_RANGE3"	第3引数不正
"ERR_ARG_RANGE4"	第4引数不正
"ERR_ARG_RANGE5"	第5引数不正
"ERR_ARG_RANGE6"	第6引数不正
"OTHER"	その他

使用例

例1　データを受信します。

```
Method ReadSample()
    data
    Begin

        With Comm1
            //通信条件を設定
            :timeOut = 1000          // タイムアウト時間に10秒を設定
            :baudRate = 9600
            :stopBits = 1
            :dataLen = 8
            :parity = 0
            :cancelKey = "ANY"
            :enable = true           // 赤外線通信(IrDA)ポートのオープン
            :Clear (3)               // データを送受信するバッファのクリア
            data = :Read(5)          // 5バイトのデータを受信
            Handy:ShowMessageBox(data, "confirm")
            :enable = false          // 赤外線通信(IrDA)ポートのクローズ
        EndWith
End Method
```

例2　データを送信します。

```
Method WriteSample()
    Begin

        With Comm1
            //通信条件を設定
            :timeOut = 1000          // タイムアウト時間に10秒を設定
            :baudRate = 9600
            :stopBits = 1
            :dataLen = 8
            :parity = 0
            :cancelKey = "ANY"
            :enable = true           // 赤外線通信(IrDA)ポートのオープン
            :Clear (3)               // データを送受信するバッファのクリア
            :Write("3132333435")     // データ"12345"の送信
            :enable = false          // 赤外線通信(IrDA)ポートのクローズ
        EndWith
End Method
```

Comm2	RS-232Cポートを使うデータ通信をおこないます。

▶ 重 要

BT-1000/1500シリーズのみ使用可能です。
過去機種との互換のため用意しています。
新規に開発する際は、Serialコントロールを使用してください。

プロパティ

名称	説明	範囲	初期値	代入
enable※	BTシリーズのRS-232Cポートの状態を設定します。 「ポートの状態(enable プロパティ)」(7-250ページ)	true:RS-232Cポートのオープン false:RS-232Cポートのクローズ	false	可
timeOut	タイムアウト時間を設定します。 各メソッドで共通して使用されます。	1～60(秒)	5	可
baudRate	通信速度(bps)を設定します。	9600/19200/38400/57600/115200	115200	可
stopBits	ストップビットを設定します。	1/2(ビット)	1	可
dataLen	データ長を設定します。	7/8(ビット)	8	可
parity	パリティを設定します。	0:なし 1:奇数 2:偶数	0	可
cancelKey	通信をキャンセルするときに押すキーを設定します。	nil:キャンセルキー無効 "C":キャンセルキー "ENT":エンターキー "F1"/"F2"/"F3":ファンクションキー "ANY":任意キー	"C"	可
error	エラー情報 エラー原因が格納されます。	文字列データ 「エラー情報(errorプロパティ)」(7-164ページ)	""	不可

※ メソッドを実行するときは、trueに設定します。
プロパティに値を格納するときは、falseに設定します。プロパティの参照は、true、falseのどちらでもできます。

メソッド

名称	説明	引数	戻り値
Initialize	各プロパティを初期化します。	なし	なし
Read※	データを受信します。	maxLength:最大受信バイト数(1～4096)	受信データ (バイナリ表現の特殊形式) nil 📖「バイナリ表現の特殊形式」(7-168ページ)
Write	データを送信します。	string:送信するデータ (バイナリ表現の特殊形式、最大4096バイト) 📖「バイナリ表現の特殊形式」(7-168ページ)	true:正常終了 false:エラー終了
Clear	送受信に使用するバッファをクリアします。	type:クリアするバッファの種類(1:送信、2:受信、3:送受信)	なし

※• 「enable」プロパティを true に設定してから「Read」メソッドを実行するまでに受信したデータが保存されていることがありますので、必要に応じて「Clear」メソッドで受信バッファをクリアしてください。

• 指定するバイト数に達した、または「timeOut」プロパティに設定した時間を過ぎると、データ受信処理は終了して、メソッドから戻り、スクリプトプログラムの次のステートメントに移ります。

■ポートの状態(enableプロパティ)

通信をおこなうときには、プロパティに通信条件を設定してから、「Read」、「Write」などのメソッドを実行します。プロパティ値を設定するには、あらかじめ「enable」プロパティを「false」に設定します。下表に、「enable」プロパティの値によって利用できることと、できないことを示します。

enableの値	プロパティの参照	プロパティの設定	メソッドの実行
true	○	△	○
false	○	○	×

○:利用可能です。

△:設定後一旦falseにして再度trueにしたときに、有効になります。

×:利用できません。

■エラー情報(errorプロパティ)

エラー情報	説明
"ERR_ARG_RANGE1"	第1引数不正
"ERR_ARG_RANGE2"	第2引数不正
"ERR_ARG_RANGE3"	第3引数不正
"ERR_ARG_RANGE4"	第4引数不正
"ERR_ARG_RANGE5"	第5引数不正
"ERR_ARG_RANGE6"	第6引数不正
"OTHER"	その他

使用例

例1　データを受信します。

```
Method ReadSample()
    data
    Begin

        With Comm2
            //通信条件を設定
            :timeOut = 10              // タイムアウト時間に10秒を設定
            :baudRate = 9600
            :stopBits = 1
            :dataLen = 8
            :parity = 0
            :cancelKey = "ANY"
            :enable = true             // RS-232Cポートのオープン
            :Clear (3)                 // データを送受信するバッファのクリア
            data = :Read(5)            // 5バイトのデータを受信
            Handy:ShowMessageBox(data, "confirm")
            :enable = false            // RS-232Cポートのクローズ
        EndWith
End Method
```

例2　データを送信します。

```
Method WriteSample()
    Begin

        With Comm2
            //通信条件を設定
            :timeOut = 10              // タイムアウト時間に10秒を設定
            :baudRate = 9600
            :stopBits = 1
            :dataLen = 8
            :parity = 0
            :cancelKey = "ANY"
            :enable = true             // RS-232Cポートのオープン
            :Clear (3)                 // データを送受信するバッファのクリア
            :Write("3132333435")       // データ"12345"の送信
            :enable = false            // RS-232Cポートのクローズ
        EndWith
End Method
```

BiCom	BT-LR1との通信機能を提供します。

BT-LR1を経由してハンディターミナルから制御機器(PLCなど)を制御できます。

プロパティ

名称	説明	範囲	初期値	代入
header	データ先頭に付加する文字列	文字列(0～5バイト)	"\x00"	可
terminator	データ終端に付加する文字列	文字列(0～5バイト)	"\x0D\x00"	可
error	エラー情報 エラー原因の内容が返されます	文字列データ ※エラー情報参照	""	不可

メソッド

名称	説明	引数	戻り値
Connect	BT-LR1との接続を確立します。	target: 接続先BDアドレス(ASCII)※1 timeout: タイムアウト時間 (1～60000ms) cancelKey: nil:キャンセル無効 "C":キャンセルキー "ENT":エンターキー "F1"/"F2"/"F3"/"F4" :ファンクションキー "ANY":任意キー	true:正常終了 false:エラー終了
Disconnect	BT-LR1との接続を切断します。	なし	true:正常終了 false:エラー終了
Send	指定データをBT-LR1に送信します。出力端子操作のみを行う事も可能です※2。	buffer: 送信文字列(最大8192バイト) status 出力端子用パラメータ(0～7)※3	true:正常終了 false:エラー終了

※1　nilを指定した場合はBluetooth:remoteBDAddrの設定値を使用します

※2　出力端子のみ操作したい場合はdsizeに0を指定してください

※3　出力端子1=1, 出力端子2=2, 出力端子3=4のOR指定となります

■エラー情報(errorプロパティ)

エラー情報	説明
"PARAM_ERR"	引数エラー
"BICOM_NG"	処理失敗
"BICOM_ALREADYOPEN"	排他関係のある通信経路が既に使用されている
"BICOM_ALREADYCLOSE"	接続されていなかった(既に切断されている)
"BICOM_NOTOPEN"	接続されていません
"BICOM_TIMEOUT"	応答がありません
"BICOM_REFUSED"	通信先デバイスから受信を拒否されました
"BICOM_CANCELED"	キャンセルしました

使用例

例1　BT-LR1に接続してデータを送信します。

```
Method sample1()
    ret
    i
    addr
Begin
    Handy:ShowMessageBox("BT-LR1を検索可能モードにしてください。","confirm")
    ret = Bluetooth:Search(nil)

    For i = 0 to ret - 1
        addr = Bluetooth:GetSearchResult(i, "addr")

        If BiCom:Connect(addr, 10000, nil) Then
            Fbreak
        EndIf

        If i == ret - 1 Then
            Handy:ShowMessageBox("BT-LR1が検出されませんでした","confirm")
            Return(nil)
        EndIf
    Next

    Handy:ShowMessageBox("BT-LR1との接続を確立しました","confirm")

    If BiCom:Send("0123456789", 7) Then
        Handy:ShowMessageBox("BT-LR1へのデータ送信に成功しました","confirm")
    Else
        Handy:ShowMessageBox("BT-LR1へのデータ送信に失敗しました","confirm")
    EndIf

    BiCom:Disconnect()
End Method
```

■HIDキーコードパケットフォーマット

HIDキーコードパケットフォーマットでBTから、データを送信します。

HIDキーコードパケットフォーマット:

Length		Type	ID	Alt,Shift,Ctrl	Padding	KEY1	KEY2	KEY3	KEY4	KEY5	KEY6
0C	00	A1	01	XX	00	XX	XX	XX	XX	XX	XX

XXについては、送信したいキーコードを設定します。それ以外は固定値になります。

Alt,Shift,Ctrl:

7	6	5	4	3	2	1	0
0	0	0	0	0	Left Alt	Left Shift	Left Control

NULLコードパケットフォーマット

Length		Type	ID	Alt,Shift,Ctrl	Padding	KEY1	KEY2	KEY3	KEY4	KEY5	KEY6
0C	00	A1	01	00	00	00	00	00	00	00	00

キーコード送信後、NULL コードを送る必要があります。NULL コードを送らない場合、通信相手側はキーの長押しと認識します。

使用例

各送信キーについて、指定する値になります。

送信キー	Alt,Shift,Ctrl	KEY1	KEY2～6
“a”	0x00	0x04	0x00
“A”	0x02	0x04	0x00
“1”	0x00	0x1E	0x00
Tab	0x00	0x2B	0x00
Shift+Tab	0x02	0x2B	0x00

■キーコード表

文字	ASCIIｺｰﾄﾞ	ｷｰｺｰﾄﾞ
0	0x30	0x27
1	0x31	0x1E
2	0x32	0x1F
3	0x33	0x20
4	0x34	0x21
5	0x35	0x22
6	0x36	0x23
7	0x37	0x24
8	0x38	0x25
9	0x39	0x26
a	0x61	0x04
b	0x62	0x05
c	0x63	0x06
d	0x64	0x07
e	0x65	0x08
f	0x66	0x09
g	0x67	0x0A
h	0x68	0x0B
i	0x69	0x0C
j	0x6a	0x0D
k	0x6b	0x0E
l	0x6c	0x0F
m	0x6d	0x10
n	0x6e	0x11
o	0x6f	0x12
p	0x70	0x13
q	0x71	0x14
r	0x72	0x15
s	0x73	0x16
t	0x74	0x17
u	0x75	0x18
v	0x76	0x19
w	0x77	0x1A
x	0x78	0x1B
y	0x79	0x1C
z	0x7a	0x1D

キー	キーコード
ENTER	0x28
TAB	0x2B
ESC	0x29
BACKSPACE	0x2A
TAB	0x2B
SPACE	0x2C
DELETE	0x4C
→	0x4F
←	0x50
↓	0x51
↑	0x52

制御キー	
CTRL	0x01
SHIFT	0x02
ALT	0x04
GUI(command)	0x08

7-14 ユーティリティ

Utility	日付の加算や、文字列変換などのユーティリティです。

プロパティ

なし

メソッド

名称	説明	引数	戻り値
AddDate	日付を加算します。※1	scrDate:加算される日付 (yyyy/mm/dd)※2 addDay:加算する日付 (-100～100)	演算結果 (yyyy/mm/dd)
DiffDate	日付の差分を計算します。※1	srcDate:比較される日付 (yyyy/mm/dd)※2 dstDate:比較する日付 (yyyy/mm/dd)※2	演算結果 (−32767～+32767)※3
Han2Zen	半角文字列をすべて全角に変換します。	src:半角を含む文字列※4	処理結果 (全角文字列)
Sort	配列内のデータをソートします。	ary:配列 type:ソート種別 "num":数値順にソートします。 "string":辞書順にソートします。 order:並び順 "ascending":昇順にソートします。 "descending":降順にソートします。	なし
MbCheck	マルチバイトチェック nバイト目のデータがシングルバイトのデータか、2バイト文字のデータかを判定します。	string:文字列 n:0～(判定位置)	1:シングルバイト 2:マルチバイト1バイト目 3:マルチバイト2バイト目 false:エラー終了
SetGaiji	外字を登録します。 📖「外字の登録(SetGaijiメソッド)」(7-310ページ)	code:文字コード(0xEB40～) dat(バイナリデータ)	true:正常終了 false:エラー終了
StartExe	実行ファイル(EXE)を起動します。 BT-W100/W80/W70シリーズのみ有効です。	filename:実行ファイルパス arg:起動引数文字列 pattern:起動方法 0:通常起動 1:起動後終了 2:EXE終了後再起動	true:正常終了 false:エラー終了
GetHeader	ヘッダ文字列を取得します。	なし	ヘッダ文字列 false:エラー終了
SetHeader	ヘッダ文字列を設定します。	len:有効文字数 string:設定する文字列	true:正常終了 false:エラー終了
GetTerminator	ターミネータ文字列を取得します。	なし	ターミネータ文字列 false:エラー終了
SetTerminator	ターミネータ文字列を設定します。	len:有効文字数 string:設定する文字列	true:正常終了 false:エラー終了

名称	説明	引数	戻り値
GetReplace	制御コードの変換ルールを取得します。	srcdata：置換え対象文字 （0x00～0x1F, 0x7F）	置換え文字 false：エラー終了
SetReplace	制御コードの変換ルールを設定します。	srcdata：置換え対象文字 （0x00～0x1F, 0x7F） dstdata：置換え文字	true：正常終了 false：エラー終了

※1 計算結果は、デバイスの日付には反映されません。

※2 指定できる形式は yyyy/mm/dd のみとなります。設定可能な日付の範囲は 1980 年 1 月 1 日～2079年12月31日までです。

※3 日付の差分で得られる最大値が±32767日を超えたときは±32767日となります。

※4 引数に全角文字や改行コードが入っていると正しく変換できません。

参 考　半角文字列に全角文字列のコードが含まれていたときは、正しく動作しません。

■外字の登録(SetGaijiメソッド)

外字データは、文字コード0xEB40を先頭に登録されます。

●BT-W100/W80/W70/1000/1500シリーズ

16ドットフォントの場合は最大15文字、 24ドットフォントの場合は最大25文字まで登録できます。32ドットフォントの場合は16ドットフォントと同じ形式の指定で、表示が縦横それぞれ倍の4倍角になります。

●BT-600シリーズ

12ドットフォントの場合は最大26文字、14ドットフォントの場合は最大26文字、16ドットフォントの場合は最大16文字、12×20ドットフォントの場合は最大26文字まで登録できます。

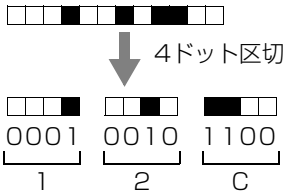
外字データは全角文字のみ登録可能です。

外字データは、現在表示中のフォントタイプに対してのみ有効です。

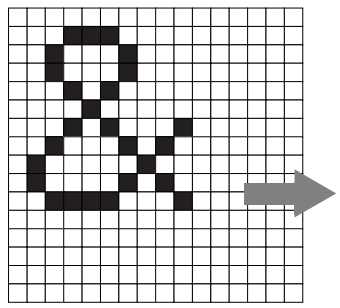
登録した外字データは、Screen:OutputTextで文字コードを指定することによって表示できます。

登録する外字データのドット情報は、最上段の行から1行ずつ、4ドットごとに16進数で表現した数値を組み合わせて表します。ドットがある部分は1、空白部分は0として計算します。

例1 ある外字データで、以下のような行の場合、この行のデータは「12C」になります。



例2 16ドットフォントで以下の外字データの場合、 1行目：0000、 2行目：1C00、3行目：2200、4行目：2200…となり、文字データは"00001C0022002200140008001440228041004280 3C4000000000000000000000" の64バイトのデータになります。



ドット情報				文字データ
0000	0000	0000	0000	0000
0001	1100	0000	0000	1C00
0010	0010	0000	0000	2200
0010	0010	0000	0000	2200
0001	0100	0000	0000	1400
0000	1000	0000	0000	0800
0001	0100	0100	0000	1440
0010	0010	1000	0000	2280
0100	0001	0000	0000	4100
0100	0010	1000	0000	4280
0011	1100	0100	0000	3C40
0000	0000	0000	0000	0000
0000	0000	0000	0000	0000
0000	0000	0000	0000	0000
0000	0000	0000	0000	0000
0000	0000	0000	0000	0000

使用例

例1　Handyコントロールにより現在のシステム時間を取得し、30日後の日付を取得します。

```
Utility:AddDate(Handy:date,30)
```

たとえば、2006年5月19日をシステム時間とすると、実行結果は「6月18日」になります。

例2　半角文字列"ABCDE"を全角に変換します。

```
Utility:Han2Zen("ABCDE")
```

例3　"&"を外字登録し表示します。

```
Method main()
    btvRet = ""
    ret = ""
    code=60224 // 0xeb40
    bin_arr1[3] = {235,64,0} // {0xeb, 0x40, 0 }最後の0はヌル文字
    bin = ""
Begin
    Screen:fontSize = "large"

    // "&"を外字登録
    ret = Utility:SetGaiji
    (code,"00001C00220022001400080014402280410042803C400
    0000000000000000000")
    If ret is true Then
        // 表示
        bin=bin_arr1.pack
        Screen:OutputText(bin)
    Else
        Handy:ShowMessageBox( "外字登録失敗", "confirm" )
    EndIf

    Handy:KeyWait()
End Method
```

例4 配列要素をソートします。

```
Method Sort()
        string[5]= {3,2,1,4,5} // 配列数と格納数が等しくないとソートできない
        i
Begin
        Handy:ShowMessageBox("{3,2,1,4,5}を数値順･昇順にソートします。","confirm")
        Utility:Sort(string,"num","ascending")
        For i = 0 to 4
                Handy:ShowMessageBox(string[i],"confirm")
        Next
End Method
```

例5 マルチバイトチェックをします。

```
Method MbCheck()
        string = "Keyキー"
        ret
Begin
        Handy:ShowMessageBox("「" & string & "」" & "の4バイト目をチェックします","confirm")
        ret = Utility:MbCheck(string,3)
        Select Case ret
        Case 1
                Handy:ShowMessageBox("シングルバイト","confirm")
        Case 2
                Handy:ShowMessageBox("マルチバイト1バイト目","confirm")
        Case 3
                Handy:ShowMessageBox("マルチバイト2バイト目","confirm")
        Case Else
                Handy:ShowMessageBox("マルチバイトチェック失敗","confirm")
        EndSelect
End Method
```

UserObj	任意のデータを保持するオブジェクトを作成します

C言語の構造体のように複数データの一括保持などに使用できます。タグ指定コントロールです。

プロパティ

なし

メソッド

名称	説明	引数	戻り値
Open	指定タグ名※1でオブジェクトを確保します	なし	true:正常終了 false:エラー終了
Close	指定タグ名のオブジェクトを解放します。	なし	true:正常終了 false:エラー終了
Set	データ登録を行います※2。メソッドの登録も可能です。	idx: 登録番号(0～63) val: 登録データ	true:正常終了 false:エラー終了
Get	データ取得を行います	idx: 登録番号(0～63)	登録データ false:エラー終了
Exec	登録されたメソッドを実行します	idx: 登録番号(0～63)	true:正常終了 false:エラー終了
Clear	データをクリアします。	idx: 登録番号(0～63)	true:正常終了 false:エラー終了

※1 確保可能なオブジェクトは256個までとなります。
ただしタグ名の先頭に"__SMALL_OBJ__"を付加した場合は512個まで確保可能です。

※2 登録可能なデータは1オブジェクトにつき64個までとなります。
ただしタグ名の先頭に"__SMALL_OBJ__"を付加した場合は4個までしか登録できません。
その場合、登録番号は0～3までの範囲となります。

使用例

例1

```
Method main()
Begin
    UserObj_Sample()
    Handy:ShowMessageBox("END", "ok" )
End Method

Method UserObj_Sample()
    ret
    str
Begin
    // オープン(変数:64個)
    UserObj<"AAA">:Open()
    // メソッド登録
    UserObj<"AAA">:Set(0, func_1)

    // データ登録
    UserObj<"AAA">:Set(1, "ABCDEFG")
    UserObj<"AAA">:Set(2, "HIJKLMN")

    // メソッド実行(func_1実行)
    ret = UserObj<"AAA">:Exec(0)
    // データ取得("ABCDEFGHIJKLMN")
    str = UserObj<"AAA">:Get(3)

    // クローズ
    UserObj<"AAA">:Close()

    // オープン(変数:4個)
    UserObj<"__SMALL_OBJ__" & "BBB">:Open()
    // データ設定
    UserObj<"BBB">:Set(0, "12345")
    // データ取得("12345")
    str = UserObj<"BBB">:Get(0)
    // クローズ
    UserObj<"BBB">:Close()
End Method

Method func_1(this)
Begin
    // 登録番号1,2のデータを連結して登録番号3に登録
    UserObj<this>:Set(3, UserObj<this>:Get(1) &UserObj<this>:Get(2))
    Return(true)
End Method
```

Key	キーの割り当てについて設定します。

プロパティ

名称	説明	範囲	初期値	代入
leftKey	Lキーに割り当てるキーを設定します。※1	"L"/"ENT"/"TRG"	"L"	可
rightKey	Rキーに割り当てるキーを設定します。※1	"R"/"ENT"/"TRG"	"R"	可
minusKey	マイナスキーにキー文字列を設定します。 入力処理でのみシフトキー有効時に使用可能になります。	3文字の文字列※2	" " (半角3文字分スペース)	可
f1Key	F1キーに割り当てるキーを設定します。	"F1"/"ENT"/"BS"/"C"/"TRG"	"F1"	可
f2Key	F2キーに割り当てるキーを設定します。	"F2"/"ENT"/"BS"/"C"/"TRG"	"F2"	可
f3Key	F3キーに割り当てるキーを設定します。	"F3"/"ENT"/"BS"/"C"/"TRG"	"F3"	可
keyBacklight	キーバックライトを設定します。 BT-W100/W80/1000/1500シリーズのみ有効です。	"AUTO"またはtrue:バックライト連動で点灯 "ALWAYS":常時点灯 false:消灯	false	可
maskKey※3	イベントを取得するキーを設定します。	keyTbl:キーテーブル 「キーテーブル(maskKey、onPressプロパティ)」(7-316ページ)	0	可

※1 BT-W100/W80/W70 シリーズで "L"/"R" キーを使用する場合は、システムメニュー「131. キー」でキーの割り付けをしてください。Lキーは0x8e、Rキーは0x8fを割り付けます。

※2 たとえば、マイナスキーに "xyz" と設定し、シフトキーを有効時に繰り返しマイナスキーを押すと、表示される文字は次のようになります。
"-"→"x" →"y" →"z"

※3 onPress イベントでイベントを取得するキーのキーテーブル値を設定します。複数のキーのイベントを取得したい場合は、すべてのキーテーブル値を合計した値を設定します。

イベントプロパティ

名称	発生タイミング	sender
onPress	キーを押したとき	キーテーブル 「キーテーブル(maskKey、onPressプロパティ)」(7-316ページ)

メソッド

名称	説明	引数	戻り値
Initialize	「Key」コントロールのプロパティを初期化します。	なし	なし
SetVolume	キークリックの音量を設定します。	Volume：音量 0：消音 1：低 2：中 3：大	なし

■キーテーブル（maskKey、onPressプロパティ）

「maskKey」プロパティで、どのキーが押されたときにイベントを取得するのか設定します。
「onPress」イベントプロパティで、現在のキー押下状態を検出します。

以下に、戻り値を一覧で示します。

キー名称	キーテーブル
0	32(0x00000020)
1	64(0x00000040)
2	128(0x00000080)
3	256(0x00000100)
4	512(0x00000200)
5	1024(0x00000400)
6	2048(0x00000800)
7	4096(0x00001000)
8	8192(0x00002000)
9	16384(0x00004000)
・	8(0x00000008)
−	16(0x00000010)
F1	1(0x00000001)
F2	2(0x00000002)
F3	4(0x00000004)
L	2097152(0x00200000)
R	4194304(0x00400000)
▲	65536(0x00010000)
▼	131072(0x00020000)
◀	262144(0x00040000)
▶	524288(0x00080000)
C※	536903680(0x20008000)
C	32768(0x00008000)
ENT	8388608(0x00800000)
TRG	1048576(0x00100000)
スキャン中	268435456(0x10000000)

※ 1秒以上長押し

使用例

バーコード入力時、キャンセルキーが押されたら、入力を終了します。

```
Method sample1()
    Begin

        With TextField<"TextField1">
            :Create("ROOT_WINDOW")
            :top = 70
            :left = 10
            :width = 200
            :enableBCR = 1          // バーコード入力を設定
            :enable = true
            :visible = true
            :Update()
        End With

        Key:maskKey = 0x00008000
        Key:onPress = OnPress1    // キャンセルキーを押されときのイベント処理を登録

        Event:Wait()                    // 入力待ち

End Method

Method OnPress1( sender ) // キャンセルキーを押されたときのイベント処理
    Begin
        //「キャンセルキーが押されました。」が表示されます
        Handy:ShowMessageBox("キャンセルキーが押されました。","ok")
End Method
```

Timer	タイマーの動作を設定します。 タグ指定コントロールです。

▶ 重 要 **タグには、タイマー番号(1～8)を指定します。**

プロパティ

名称	説明	範囲	初期値	代入
timerCount※	動作中のタイマーの数を取得します。	1-8	nil	不可

※ どのタグ(タイマー番号)を指定しても同じ値を取得できます。有効なタグを指定して参照してください。

イベントプロパティ

名称	発生タイミング	sender
onTimer	タイマーがタイムアップしたとき	タイマー番号

メソッド

名称	説明	引数	戻り値
Set※	タイマーを開始します。	timeOut:タイムアウト時間 (×10msec)	true:正常終了 false:エラー終了
Kill※	タイマーを停止します。	なし	なし

※ Setメソッドの引数で指定した時間を経過するごとにタイムアップのイベントが発生します。
タイマーが不要になったときは、必ずKillメソッドを実行してタイマーを停止させてください。

使用例

10秒タイマーを作成します。

```
Method Set_Timer10()
    Begin

        // タイマー番号1を使用します
        With Timer<1>
            :onTimer = OnTimeUp
            :Set(1000)
        End With

        Event:Wait()                    // 入力待ち

End Method

Method OnTimeUp(sender)
    Begin
        //「1がタイムアップしました」が表示されます
        Handy:ShowMessageBox(sender & "がタイムアップしました", "confirm")
End Method
```

Handy	デバイス設定、メッセージ表示、メニュー表示、キー入力などをおこないます。

プロパティ

名称	説明	範囲	初期値	代入
systemVersion	本体システムのバージョン	—	—	不可
version	システムソフトのバージョン	—	—	不可
serialChar	出荷履歴番号が取得できます(9ケタの文字列) BT-1000/1500/600シリーズのみ有効です。	文字列	—	不可
validLcdWidth	LCD内使用幅の指定 BT-600シリーズのみ有効です。	120:BT-500互換 128:LCD 全領域	128	可
sysmenuCompatiMode	次回起動するシステムメニューについてBT-500互換版で初期起動できます。 BT-600シリーズのみ有効です。	true:BT-500 false:BT-600	BTシリーズの設定値	可
commModeM	BT-500通信ライブラリのモデム対応(モードM)指定ができます。 BT-600シリーズのみ有効です。	true:有効 false:無効	BTシリーズの設定値	可
backup	低電圧検知時・完全OFF移行時にRAMデータをROMデータにバックアップし再度電源投入されたときにRAM からROM に書き戻す「ROM⇔RAMバックアップモード」を指定できます。 BT-600シリーズのみ有効です。	true:有効 false:無効	BTシリーズの設定値	可
date	内部時計の日付を設定します。	yyyy/mm/dd(年/月/日)	BTシリーズの設定値	可
time	内部時計の時刻を設定します。	HH:MM:SS(時:分:秒)	BTシリーズの設定値	可
leftKey※1	Lキーに割り当てるキーを設定します。	"L"/"ENT"/"TRG"	"L"	可
rightKey※1	Rキーを割り当てるキーを設定します。	"R"/"ENT"/"TRG"	"R"	可
minusKey	マイナスキーにキー文字列を設定します。入力処理でのみシフトキー有効時に使用可能になります。	3文字の文字列※2	" " (半角3文字分スペース)	可
validKey	有効となるキーをキーテーブルで指定できます。複数指定も可能です。 BT-600シリーズのみ有効です。	キー指定 📖「キー入力待ち(KeyWait、KeyWaitExメソッド)」(7-328ページ)	全キー	可
autoPowerOff※7	電源が自動的にオフになるまでの時間を設定します。	nil:無効 0:即時パワーオフ 1～60:パワーオフ時間(分)	5	可
poweroffMode※7	バックアップモード「省電」に設定します。 BT-600シリーズのみ有効です。	true:有効 false:無効	BTシリーズの設定値	可
autoRamFormat	起動時にRAMフォーマットが必要な場合にRAMフォーマット確認メッセージを表示せずに実行します。 BT-600シリーズのみ有効です。	true:有効 false:無効	BTシリーズの設定値	可
batteryLevel	バッテリーの残量を取得します。	1～4:充電状態(4:満充電)※5	—	不可

名称	説明	範囲	初期値	代入
resumeOn	レジューム機能を有効にする、しないを設定します。 「6-10 省電力機能」(6-61ページ)	true：有効にする false：無効にする	true	可
fileVersion	中間言語ファイルのバージョン	ファイルからの読み込み値	—	不可
fileTitle	中間言語ファイルのタイトル	ファイルからの読み込み値	—	不可
enablePowerKey	PWキーによる電源オフを有効にする、しないを設定します。	true：有効にする false：無効にする	true	可
serialId	シリアル番号を取得します。	文字列	BT-W100/80/70/1000/1500シリーズ: 固有値 BT-600シリーズ:固有値"0"	不可
debugPort	デバッグポートの出力先を設定します。 BT-1000/1500/600シリーズのみ有効です。	●BT-1000/1500シリーズ "232C":RS-232Cポート "USB":USBポート ●BT-600シリーズ "232C"：通信ユニット(互換用) "CRADLE"：通信ユニット	BT-1000/1500シリーズ: "232C" BT-600シリーズ: "CRADLE"	可
wakeupOn	ウェイクアップの有効、無効を制御します。※4 BT-1000/1500/600シリーズのみ有効です。	false：無効 "RF"：有効(無線) "RFALWAYS"：常時有効(無線) "CRADLE"またはtrueまたは "IRDA"：通信ユニット(RS-232C、USB、LAN(mode2)) "LANCRADLE"：LAN通信ユニット(mode1)	無線LAN搭載機種："RF" それ以外の機種： "CRADLE"	可
id	端末のIDを設定します。	1～65534	BTシリーズの設定値	可
model	機種名を取得します。※8 BT-W100/W80/W70/600シリーズで有効です。BT-1000/1500シリーズはmodelCharを使用します。	BTシリーズの型式	固有値	不可
language	言語種別を取得します。	"japanese":日本語 "simplified":中国語簡体字 "english":英語	BTシリーズの設定値	不可
subBatteryLevel	内部電池の残量を取得します。※3	0:満充電でない 1:満充電	BTシリーズの設定値	不可
cradleStatus	通信ユニットからの給電状況	true:通信ユニット上 false:通信ユニット上でない	—	不可
usbStatus	内蔵USBの接続状況 BT-1000/1500シリーズのみ有効です。	true：接続されている false：接続されていない	—	不可
cradleStatW	通信ユニットと通信できることを検出できます。 通信ユニットポートを使用していないときは検出できません BT-600シリーズのみ有効です。	0：通信不可 1：通信可 2：不明	—	不可
inputFixKey	InputString、InputDecimalでデータ編集中にENT以外に入力関数が確定するキーを選択します。※6	有効にするキーテーブルの和を代入します。 0x10000:"UP" 0x20000:"DOWN" 0x200000:"L" 0x400000:"R"	0	可

名称	説明	範囲	初期値	代入
dfinfoMode	FileSystemコントロールのFindFirst/FindNext/DriveInfoメソッドにて、ドライブ5のドライブ容量をブロック単位(512バイト)で取得します。 microSDHCカード(容量が2GBを超えるもの)を使用する場合には有効にしてください。無効にしていると、取得エラーが発生する場合があります。 BT-600シリーズでは使用できません。	true:有効 false:無効	BTシリーズの設定値	可
modelChar	機種名を取得します。 BT-1000/1500シリーズのみ有効です。	"BT-1010"、"BT-1010B"、"BT-1010W"、"BT-1010WB"、"BT-1010WGB"、"BT-1010WBGM"、"BT-1510"、"BT-1510B"、"BT-1510W"、"BT-1510WB"、"BT-1510WBG"、"BT-1510WBGM"、"BT-1550"、"BT-1550WB"、"BT-1550WBG"、"1550WBGM"	固定値	不可

※1　BT-W100/W80/W70シリーズで"L"/"R"キーを使用する場合は、システムメニュー「131.キー」でキーの割り付けをしてください。Lキーは0x8e、Rキーは0x8fを割り付けます。

※2　たとえば、 マイナスキーに"xyz"と設定し、 シフトキーを有効時にマイナスキーを繰返し押すと以下のように表示が変化します。
"-"→"x"→"y"→"z"

※3　パソコン上のシミュレーションでは設定できません。

※4　バックアップモード「省電」で電源をOFFにしている場合は、ウェイクアップ機能を使用できません。

※5　起動直後はバッテリレベルが不安定になり、正確な値が取得できない場合があります。 起動後、1秒以上経過してからプロパティ値を取得するようにしてください。

※6　"UP","DOWN","L","R"を指定できます。これらを指定したとき、InputString/InputDecimal:Execでデータ編集中にこれらのキーを押すと、入力データが確定し、“ENT” が戻り値として返ります。入力確定キーの種類は、各コントロールのExecメソッド直後でHandy:KeySense()でキーセンスを取得することにより判断できます。

※7　以下のプロパティを変更した場合、PropertySaveを実行して、設定をRAMに保存してください。
backup
autoRamFormat
poweroffMode
sysmenuCompatiMode
「ROM保存(PropertySaveメソッド)」(7-330ページ)

※8　旧機種との互換のため、実在しない型式が表示される場合があります。
新規開発ではmodelCharを使用してください。

ポイント

- **端末アプリケーション起動時、開発用通信ポート無効になっています。**
- **開発用通信ポートを使用する場合は、「システムメニュー」の「デバッグ出力」でデバッグ出力を許可する必要があります。**

イベントプロパティ

名称	発生タイミング	sender
onResume	レジューム時に電源をONしたとき	1(固定)
onCradle	置き台に接続したとき／接続しなくなったとき	1:接続 0:未接続
onBatteryLevelChange	バッテリレベルが変更したとき バッテリレベルはbatteryLevelプロパティで取得してください	1～4(電池残量)

メソッド

名称	説明	引数	戻り値
Initialize	各プロパティを初期化します。	なし	なし
FirmUpdate	ファイル名を指定して、本体ファームの読込/出力をおこないます。読み込みで本体ファームのバージョンアップ、出力で現バージョンの本体ファームウェアのファイルを出力します。 BT-600シリーズのみ有効です。	sw:入出力指定(1:読込/2:出力) filename:ファイル名 param:アサイン値 「本体ファームバージョンアップ(FirmUpdateメソッド)※BT-600シリーズのみ有効」(7-330ページ)	true:正常終了 false:エラー終了
ShowClickVolumeDialog	キークリック音量設定画面を表示します。 「キークリック音量設定(ShowClickVolumeDialogメソッド)」(7-326ページ)	title:タイトル	true:正常終了 false:エラー終了
ShowMessageBox	メッセージボックスを表示します。 「メッセージボックス表示(Messageメソッド)」(7-131ページ)	str:表示する文字列 (最大半角1024文字) type:表示ボタンと初期値 "ok"※3 "confirm"※3 "okcancel|ok"※3 "okcancel|cancel"※3 "yesno|yes"※3 "yesno|no"※3	true:正常終了 false:エラー終了
ShowMenu	メニューを表示します。 「メニュー表示(ShowMenu、ShowMenu2メソッド)」(7-327ページ)	title:メニューのタイトル items1:1～4項目のメニュー内容の文字列 items2:5～8項目のメニュー内容の文字列 items3:9～12項目のメニュー内容の文字列 items4:13～16項目のメニュー内容の文字列 initItem:表示時に選択するアイテム(1～16)	選択されたフィールド番号※1 "C":「C」キー長押し
ShowMenu2	メニューを2列で表示します。 「メニュー表示(ShowMenu、ShowMenu2メソッド)」(7-327ページ)	title:メニューのタイトル items1:1～4項目のメニュー内容の文字列 items2:5～8項目のメニュー内容の文字列 items3:9～12項目のメニュー内容の文字列 items4:13～16項目のメニュー内容の文字列 initItem:表示時に選択するアイテム(1～16)	選択されたフィールド番号※1 "C":「C」キー長押し
DebugOut	デバッグ文字列を出力します。 「デバッグ(DebugOut、DebugInメソッド)」(7-328ページ) BT-1000/1500/600シリーズのみ有効です。	str:文字列	なし

名称	説明	引数	戻り値
DebugIn	デバッグ文字列を入力します。 「デバッグ(DebugOut、DebugInメソッド)」(7-328ページ) BT-1000/1500/600シリーズのみ有効です。	size:入力最大バイト数(1～126) timeOut:タイムアウト時間 (1～30秒) nil(タイムアウトなし)※2	文字列
Wait	スクリプトの実行を一時停止します。 「動作の一時停止(Waitメソッド)」(7-328ページ)	time:停止時間 (1～300(×100ミリ秒)) keyCancel: キー押下によるキャンセル ・true(許可する) ・false(許可しない)	なし
KeyWait	キー入力を待ちます。 「キー入力待ち(KeyWait、KeyWaitExメソッド)」(7-328ページ)	なし	入力キーコード 「キー入力待ち(KeyWait、KeyWaitExメソッド)」(7-328ページ)
Reset	BTシリーズのソフトウエアリセットをおこなって、システムを再起動します。	なし	なし
ShowLineMessage	1行メッセージを表示します。	title:タイトル str1:true時の文字列 str2:false時の文字列 default:true/false	true:正常終了 false:エラー終了
KeyWaitEx	タイマー付きキーウェイト 有効キーテーブルで設定したキーを、設定したタイムアウト時間だけ待ちます。 「キー入力待ち(KeyWait、KeyWaitExメソッド)」(7-328ページ)	keyTbl:有効キーテーブル※4 timeOut:タイムアウト(0～300(×100ミリ秒))	入力キーコード タイムアウト値は0
KeySense	キー入力センス 「キー入力センス(KeySenseメソッド)」(7-329ページ)	なし	入力 キーテーブル
GetTickCount	チックカウント取得 システム起動後からのカウントを取得します。 カウント値は約10msで1加算され約485日でオーバーフローします。	なし	チックカウント (10ms単位)
GetTickCount1	1ms精度のチックカウントを取得します。	なし	チックカウント (1ms単位)
SetApplicationSchedule	指定時刻にアプリを起動します。※6	ID: ID(1～8) H:時刻 M:分 fileName:アプリケーション名(ドライブ番号:ファイル名(拡張子「.APP」「.sb3」の形式のみ対応))	true:正常終了 false:エラー終了
KillApplicationSchedule	指定時刻アプリ起動を停止します。	ID: ID(1～8)	true:正常終了 false:エラー終了
GetApplicationSchedule	アプリ起動の設定内容を取得します。	ID: ID(1～8)	true:正常終了※10 false:エラー終了
SetAlarm	指定時刻にLED、ブザーを鳴動させます。※5	ID:アラームID(1～8) H:時刻 M:分 useDev:使用デバイス※7 message:表示メッセージ	true:正常終了 false:エラー終了

名称	説明	引数	戻り値
KillAlarm	アラームを停止します。	ID:アラームID(1～8)	true:正常終了 false:エラー終了
GetAlarm	アラームの設定内容を取得します。	ID:アラームID(1～8)	成功時:※11 false:エラー終了
PropertySave	設定をROMに保存します。	なし	true:正常終了 false:エラー終了
ShowScreenPassword	パスワード入力画面を表示します。 日本語の入力はできません。 ・入力は「*」で表示します。シフト入力の際には、文字選択中はアルファベットを表示し、確定して次の文字へ移行した際に「*」表示に変化します。	title:タイトル password:パスワード※8 maxLength:最大文字数(1～40) scanMode:※9 enableKeyInput:※9 shift:※9	true:正常終了
PhotoSave	撮像した画像を752x480ドットのグレースケールで保存します。 関数実行時に撮像します。 BT-1500シリーズのみ有効です。	fileName:保存ファイル名 (ドライブ番号:ファイル名) type:保存形式(0固定) light:照明設定 true:ON false:OFF	true:正常終了 false:エラー終了
LiveView	ライブビュー機能の開始、一時停止、停止を実行します。 BT-1500シリーズのみ有効です。	0:停止 1:開始・再開 2:一時停止	true:正常終了 false:エラー終了
SetIllumination	ライブビュー機能使用時の照明ON/OFFを設定します。 コード読み取りを実行すると、自動でOFF設定になります。 BT-1500シリーズのみ有効です。	true:ON false:OFF	true:正常終了 false:エラー終了
LoadConfig	共通設定ファイルを読み込みます。	srcfile:ファイル名	true:正常終了 false:エラー終了
SaveConfig	現在の端末設定を共通設定ファイルとして保存します。	dstfile:ファイル名 kind: 保存する設定種別 1: 読取設定 2: 端末設定 4:ネットワーク設定 ※OR指定可能です。 例)読取設定と端末設定を指定する場合は1+2=3、端末設定とネットワーク設定を指定する場合は2+4=6	true:正常終了 false:エラー終了

※1 "1～16"が返ります。キャンセル時には"C"が返ります。
※2 タイムアウトとは、メソッドコール時からの時間です。
※3 引数「type」の内容は次のようになります。
"ok" :OKボタンのみ
"confirm" :確認ボタンのみ
"okcancel|ok" :"OK/キャンセル"、初期値は"OK"
"okcancel|cancel" :"OK/キャンセル"、初期値は"キャンセル"
"yesno|yes" :"はい/いいえ"、初期値は"はい"
"yesno|no" :"はい/いいえ"、初期値は"いいえ"
※4 入力待ちしたいキーテーブルの値(7-329ページ)をすべて足した値を代入してください。
「0」を代入すると、すべてのキーが有効になります。
※5 現在のブザー、LED、バイブレーションコントロールの設定値で動作します。
鳴動時間、鳴動のパターンはブザーコントロールの設定を使用します(端末ライブラリでは 1 種類の鳴動パターンで全デバイスを動作させるため)。
Buzzer:onTime, Buzzer:offTime, Buzzer:loopCount, Buzzer:pitchプロパティ、
Led:colorプロパティが参照されます。
※6 弊社専用アプリケーションを使用してビルド後のXHOファイルから変換されたをAPPファイル、中間言語ファイルを呼び出す場合にはSB3ファイルを、端末上のドライブ1・2・5に置いてください。
※7 "Buzzer", "LED", "Vibration"、"Buzzer+LED"という表記で記述します。

※8 パスワード入力してほしい文字列を入力します。nil を指定するとシステムメニューの管理者パスワードと同じになります。最大32バイトです。
※9 InputStringのプロパティと同じ設定です。
※10 以下のフォーマットの文字列になります。
有効時:"12:34,fileName"(%02d:%02d %s) …区切り文字はカンマ
無効時:"INVALID"
※11 以下のフォーマットの文字列になります。
有効時:"12:34,有効デバイス,message"(%02d:%02d,%s,%s) …区切り文字はカンマ
また、現在のブザー、LED,バイブレーションコントロールの設定値を変更します。
この関数を呼ぶ前にあらかじめこれらのコントロールのプロパティ値をバックアップしておいてください。
変化する値は、SetAlarmで参照しているプロパティだけです。
無効時:"INVALID"

■キークリック音量設定(ShowClickVolumeDialogメソッド)

「ShowClickVolumeDialog」メソッドを使用すると、キー入力音の設定画面を表示させて、キー入力音を設定できます。
引数「title」は、タイトルを表示するときに使用します。液晶表示部(LCD)の最上段の1行のみ表示できます。

ポイント **この関数をコールすると、VRAMに再描画する必要があります。**

参考 キークリック音量はKeyコントロールのSetVolumeメソッドを用いて設定することも可能です。

例
- 「ENT」キーで決定し終了します。
- 「C」キーで変更をキャンセルします。

■メッセージボックス表示(ShowMessageBoxメソッド)

「ShowMesseageBox」メソッドを使用すると、メッセージボックスを画面上に表示できます。
- 引数「type」で「ok」、「confirm」を設定した場合、「OK」または「確認」のみが表示されます。
- 引数「type」で「okcancel |ok」を設定した場合、表示時のカーソル位置は「OK」、「okcancel |cancel」を設定した場合、表示時のカーソル位置は「キャンセル」になります。

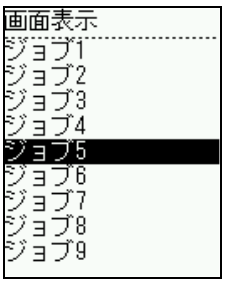
下図は、引数「type」を「okcancel |ok」にしたときの表示例です。

■メニュー表示(ShowMenu、ShowMenu2メソッド)

「ShowMenu」メソッドを使うと、液晶表示部(LCD)にメニューを表示させて、メニュー項目から選択入力することができます。

- メニュー項目数は、最大16個です。
- items1～4に、それぞれ4個までのメニュー項目を設定します。メニュー項目がないときは、必ずnilに設定する必要があります。
- 各項目は"|"で区切ります。したがって、"|"は表示できません。

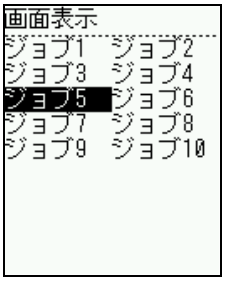
例1　　下図のようなメニューを表示します。




```
title="画面表示"          //タイトルを代入
items1="ジョブ1|ジョブ2|ジョブ3|ジョブ4"
items2="ジョブ5|ジョブ6|ジョブ7|ジョブ8"
items3="ジョブ9|ジョブ10"
items4=nil
//タイトルをつけてメニューを表示
Handy:ShowMenu(title,items1,items2,items3,items4,5)
```

例えば「ジョブ5」を選択すると、「5」という値が戻り値になります。

例2　　下図のような2列のメニューを表示します。




```
title="画面表示"          //タイトルを代入
items1="ジョブ1|ジョブ2|ジョブ3|ジョブ4"
items2="ジョブ5|ジョブ6|ジョブ7|ジョブ8"
items3="ジョブ9|ジョブ10"
items4=nil
//タイトルをつけてメニューを表示
Handy:ShowMenu2(title,items1,items2,items3,items4,5)
```

■デバッグ(DebugOut、DebugInメソッド)

Windowsのハイパーターミナルなどのターミナルソフトを使用して、PCとデバッグ情報を送受信します。BT-1000/1500/600シリーズのみ有効です。

BT-1000/1500シリーズでは、BTシリーズのRS-232C/USBポートとPCのRS-232C/USBポートを開発用ケーブル/USB ケーブルで接続します。

BT-600シリーズでは、BTシリーズのRS-232C/USB通信ユニットで接続します。

PCの通信条件は以下のように設定します。

項目	設定
通信速度	115200bps
データ長	8ビット
パリティ	なし
ストップビット	1ビット
フロー制御	なし

▶ 重 要 **BTシリーズのシステム設定で「デバッグ出力」を「許可」に設定してください。**
設定方法については、『システムメニュー操作マニュアル』を参照してください。

「DebugOut」メソッド

シミュレーション機能の「デバッグ画面」、またはターミナルソフトに「DebugOut」メソッドの引数でわたした文字列を表示します。

改行を指定するときは"¥r"または"¥n"と記述します(ターミナルソフトの設定によって異なります)。

「DebugIn」メソッド

シミュレーション機能の「デバッグ画面」、またはターミナルソフトからの入力を受信します。

受付終了のタイミングは下記のとおりです。

- 受信サイズが入力最大バイト数(size)に達したとき
- 改行コードを受信したとき
- タイムアウト時間に達したとき

例 "step3終了"という文字列を表示します。

```
Handy:DebugOut("step3終了")
```

■動作の一時停止(Waitメソッド)

「Wait」メソッドを使用すると動作を一時停止することができます。

- 「keyCancel」を「true」にすると、この間の割り込みを受け付けます。キー入力されると Wait 状態は解除されます。

■キー入力待ち(KeyWait、KeyWaitExメソッド)

「KeyWait」、「KeyWaitEx」メソッドを使用すると省電力モードでキー入力を待ちます。

キー入力されると、省電力モードは解除され、入力されたキーコードを返します。

「KeyWaitEx」メソッドを使用すると、入力可能なキーの指定および入力タイムアウト時間を設定できます。

■キー入力センス(KeySenseメソッド)

「KeySense」メソッドを使用すると、現在のキー押下状態を検出します。
キー押下状態を取得するとき、長押し、スキャン中の情報も同時に取得されます。また、2個までの同時キー入力も取得できます。同時押しのキー入力を取得した場合、該当するキーの戻り値を足した値として取得します。

以下に、戻り値を一覧で示します。

キー名称	戻り値 (KeyWait、KeyWaitEx)	戻り値(KeySense) キーテーブル
0	"0"	32 (0x00000020)
1	"1"	64 (0x00000040)
2	"2"	128 (0x00000080)
3	"3"	256 (0x00000100)
4	"4"	512 (0x00000200)
5	"5"	1024 (0x00000400)
6	"6"	2048 (0x00000800)
7	"7"	4096 (0x00001000)
8	"8"	8192 (0x00002000)
9	"9"	16384 (0x00004000)
.	"."	8 (0x00000008)
−	"-"	16 (0x00000010)
F1	"F1"	1 (0x00000001)
F2	"F2"	2 (0x00000002)
F3	"F3"	4 (0x00000004)
L	"L"	2097152 (0x00200000)
R	"R"	4194304 (0x00400000)
▲	"UP"	65536 (0x00010000)
▼	"DOWN"	131072 (0x00020000)
◀	"LEFT"	262144 (0x00040000)
▶	"RIGHT"	524288 (0x00080000)
C※	"C"	−
C	"BS"	32768 (0x00008000)
ENT	"ENT"	8388608 (0x00800000)
TRG	"TRG"	1048576 (0x00100000)
スキャン中	−	268435456 (0x10000000)

※ 1秒以上長押し

■有効キーの設定(validKeyプロパティ)※BT-600シリーズのみ有効

有効にするキーテーブルの値を足した値を設定します。
(キーテーブルは、KeySenseメソッドを参照してください。)

全キー有効:16777215 (0xFFFFFF)
全キー無効:0 (0x000000)
TRGのみ無効:15728639 (0xEFFFFF)
"0"、" 1" のみ有効:96 (0x000060)

使用例:全キー無効
Handy:validKey=0

■本体ファームバージョンアップ(FirmUpdateメソッド)※BT-600シリーズのみ有効

アサイン値について

- ファーム出力時のアサイン値は、0(固定)です。それ以外を指定した場合、エラーとなります。
  Handy:FirmUpdate(2,"任意のファイル名",0)

- ファームの読み込み時のアサイン値は以下のパラメータの足し算で指定します。
  1:バージョンアップ開始前の確認画面表示(0の場合表示しない)
  2:バージョンアップ成功後アップデートファイル削除(0の場合削除しない)
  4:バージョンアップ成功後電源OFF(0の場合リセット)

例1) バージョンアップ開始時に確認画面を表示して、成功後、アップデートファイルを削除して電源OFFします。

```
Handy:FirmUpdate(1,"任意のファイル名",7)      //1+2+4
```

例2) 成功後、アップデートファイルを残して電源OFFします。

```
Handy:FirmUpdate(1,"任意のファイル名",4)      //0+0+4
```

■ROM保存(PropertySaveメソッド)

プログラムでプロパティの変更を行った場合、設定は RAM に保存され、プログラム再起動時に設定内容が消去されます。そのため、設定をROMに保存する必要があります。

```
//プロパティを設定します.
Handy:backup=true
Handy:autoRamFormat=true
Handy:poweroffMode=true

//プロパティ設定をROMに保存します.
Handy:PropertySave()
```

使用例

内部時計の日時を表示･変更します。

```
Method main()
        date
        time
Begin
        //内部時計の日時を表示します
        date = Handy:date
        time = Handy:time
        Handy:ShowMessageBox("変更前\n" & date & "\n" & time & "\n","confirm")

        //内部時計の日時を変更します
        //※ ハンディの内部時計が変わります、ご注意ください
        Handy:date = "2000/12/23"
        Handy:time = "12:34:56"
        date = Handy:date
        time = Handy:time
        Handy:ShowMessageBox("変更後\n" & date & "\n" & time,"confirm")
End Method
```

MEMO

付録

スクリプトで使う予約語、エラー一覧などについて説明します。

1　アスキーコード表 …… 付-2
2　予約語一覧 …… 付-3
3　変数の内部型 …… 付-4
4　置き換えマスタ …… 付-6
5　コンパイルエラー一覧 ……付-10
6　実行時例外一覧 ……付-12
7　索　引 ……付-14

1 アスキーコード表

スクリプトで使用できる、1バイト文字のコード一覧です。

			上位4ビット							
	16進		0	1	2	3	4	5	6	7
		2進	0000	0001	0010	0011	0100	0101	0110	0111
下位4ビット	0	0000	NUL	DLE	SP※	0	@	P	`	p
	1	0001	SOH	DC1	!	1	A	Q	a	q
	2	0010	STX	DC2	"	2	B	R	b	r
	3	0011	ETX	DC3	#	3	C	S	c	s
	4	0100	EOT	DC4	$	4	D	T	d	t
	5	0101	ENQ	NAK	%	5	E	U	e	u
	6	0110	ACK	SYN	&	6	F	V	f	v
	7	0111	BEL	ETB	'	7	G	W	g	w
	8	1000	BS	CAN	(	8	H	X	h	x
	9	1001	HT	EM	)	9	I	Y	i	y
	A	1010	LF	SUB	*	:	J	Z	j	z
	B	1011	HM	ESC	+	;	K	[	k	{
	C	1100	CL	FS	,	<	L	¥	l	|
	D	1101	CR	GS	-	=	M	]	m	}
	E	1110	SO	RS	.	>	N	^	n	~
	F	1111	SI	US	/	?	O	_	o	del

※ SPはスペースを示します。

2 予約語一覧

スクリプトで使用している名前一覧です。スクリプトで名前を定義するときは、これらの名前を使用することができません。将来の予約語も含まれています。

A	aeq	G	ge	R	regeq
	And		gt		Rfind
B	Begin	H	Hex		Remove
	byte	I	If		Repeat
C	Case		Include		Return
	CaseElse		int		Right
	Catch		is	S	SelectCase
	Const		isBoolean		size
	Control		isDigit		Split
D	default		isInt		step
	Delete		isString	T	Then
E	Else	L	le		to
	ElseIf		Left		toInt
	EndControl		length		toUpper
	endenum		lt		toLower
	End	M	Match		toMethod
	EndIf		Merge		true
	EndMethod		Method	U	Unpack
	EndPackage		Mid		UnpackBin
	EndSelect		MOD	W	Wbreak
	EndWith	N	ne		Wcontinue
	enum		Next		Wend
	eq		nil		weq
F	false		not		Which
	Fbreak	O	Or		While
	Fcontinue	P	Pack		With
	Find		Package		
	For		PackBin		

3 変数の内部型

変数には、数値、文字列などの型はありませんが、システム内部では型を持っています。ここでは、変数の内部型について説明します。

内部型(数値型、文字列型、論理型)

変数は型を宣言しなくても使用できますが、システム内部では型を持っています。
演算などの実行時には必要に応じて自動的に数値型から文字列型へ、文字列型から数値型へと変換が行われます。

■変数の内部型

数値型

例　12345

文字列型

- 数値変換可能な文字列型

  例　"123"

- 数値変換不可能な文字列型

  例　"ABC"

論理型(true、false、nil)

■変換ルール

●変数が数値型の場合

文字列比較演算子などで評価されるときに、文字列に変換されます。

- 小数値は、末尾の0を省きます。整数のときは、コンマ(.)をつけません。
  たとえば、「0.100」は「0.1」に変換します。「10.0」は「10」に変換します。
- 式の計算結果は、小数点以下4桁目を四捨五入し、最大3桁目まで有効です。
  式の演算途中には、四捨五入、切り捨て処理は行なわれません。

▶ 重 要　**数値型を文字列型に変換する範囲は、−2147483647〜2147483647です。この範囲外の数値を文字列型に変換するときは、実行時例外となります。**

●変数が文字列型の場合

算術演算子、数値比較演算子などで評価されるときに、数値に変換されます。

- 変換された数値が小数のときは、小数第3位(小数点以下4桁目を四捨五入)まで有効です。

▶ 重 要　**空白や数字(符号、小数点記号含む)ではない文字が含まれる文字列を数値変換したときは、実行時例外となります。**

●変数が論理型の場合

算術演算子、文字列比較演算子などで評価されるとき、変換されません。

▶ 重 要　**数値比較演算子、文字列比較(ワイルドカード比較含む)で評価されると、実行時例外となります。**

■変数の内部型の参照

下記のプロパティは、変数の内部型を参照して結果を返します。

名称	説明	参照結果			
		数値型	文字列型		論理型
			数値変換可能	数値変換不可能	
isDigit	変数が数値型かを判定します	true	true	false	false
isBoolean	変数が論理型かを判定します	false	false	false	true
isString	変数が文字列型かを判定します	false	true	true	false

例1　「数値型」のみを抽出するIf文
変数aの内部型が論理型でなく、文字列型でもないとき

```
If (a.isBoolean is false) And (a.isString is false) Then…
```

例2　「数値型」または「文字列型」を抽出するIf文
変数aの内部型が論理型でないとき

```
If (a.isBoolean is false) Then …
```

例3　「数値変換不可能な文字列」のみを抽出するIf文
変数aの内部型が文字列型で、かつ数値型でないとき

```
If (a.isString is true) And (a.isDigit is false) Then …
```

例4　「数値変換可能な文字列」のみを抽出するIf文
変数aの内部型が文字列型で、かつ数値型のとき

```
If (a.isString is true) And (a.isDigit is true) Then …
```

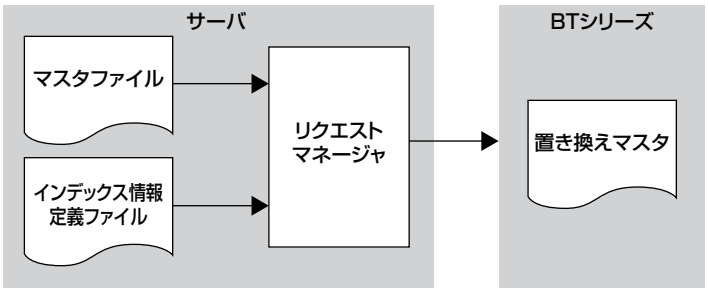
4 置き換えマスタ

BTシリーズ内で使用する置き換えマスタには、インデックス情報が付加されています。
置き換えマスタは、最大数十万件のデータを想定しているため、インデックス情報を使って高速に検索をおこないます。
ここでは、インデックス情報定義ファイルと置き換えマスタについて、説明します。

■置き換えマスタの作成方法

●リクエストマネージャを使用する場合

サーバで作成したマスタファイルとインデックス情報定義ファイルは、「リクエストマネージャ」を経由してBT シリーズが受信するときに、インデックス情報が付加された置き換えマスタに変換されます。



上図は、「リクエストマネージャ」経由で、BTシリーズが「置き換えマスタ」を受信している例です。
BTシリーズと「リクエストマネージャ」との通信については、「RemoteFile」(7-229ページ)を参照してください。

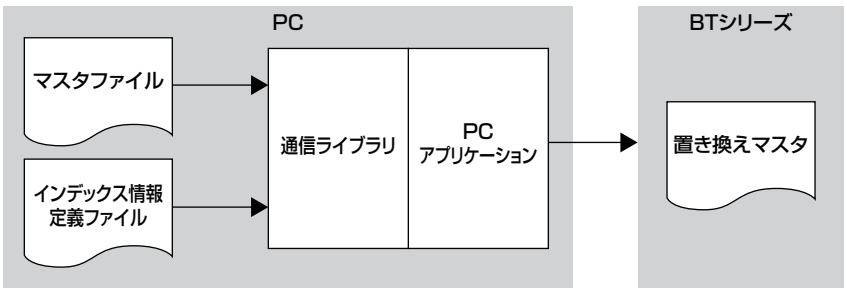
▶ 重 要

インデックス情報定義ファイルの内容が、記述ルールに合っていないときは、下記のようなエラーとなって、マスタファイルはBTシリーズへ送信されません。
•「リクエストマネージャ」から受信時:"ERR_FILE_GET"

記述ルールについては、「インデックス情報定義ファイルのフォーマット」(付-8ページ)を参照してください。

●通信ライブラリを使用して作成したPCアプリケーションを使用する場合

PCで作成したマスタファイルとインデックス情報定義ファイルは、「通信ライブラリ」を経由してBT シリーズが受信するときに、インデックス情報が付加された置き換えマスタに変換されます。



上図は、「通信ライブラリ」を使用して作成したPCアプリケーションを使用している例です。
「通信ライブラリ」については、『通信ライブラリリファレンス』を参照してください。

▶ 重 要

インデックス情報定義ファイルの内容が、記述ルールに合っていないときは、下記のようなエラーが発生し、マスタファイルはBTシリーズへ送信されません。

- **「通信ライブラリ」使用時：変換エラー(-307)**

記述ルールについては、「インデックス情報定義ファイルのフォーマット」（付-8ページ）を参照してください。

●データ転送ソフトを使用する場合

「データ転送ソフト」を使用して、BTシリーズに転送することも可能です。
「データ転送ソフト」の操作方法については、『BT-Navigator アプリケーション開発マニュアル』「12章 データ転送ソフトの操作方法」を参照してください。

●ハンディ上で置き換えマスタを作成する場合

ハンディ上でマスタファイルとインデックス情報定義ファイルから、置き換えマスタを作成することができます。(FileSystem:SearchCreate)
あらかじめマスタファイル、インデックス情報定義ファイルをハンディ上に転送しておくか、File/FileStreamコントロールを使用してこれらのファイルを作成します。
ファイルフォーマットが正しくない場合は、置き換えマスタの作成に失敗し、FileSystem:SearchCreateにエラーが返ります。

ポイント

FileSystem:SearchCreateは圧縮置き換えマスタの作成には対応していません。

■ファイル名の付け方

- **マスタファイル**
  28バイト以内のファイル名+"."+3文字の拡張子(csvまたはtxt)

  例　master.txt

  レコードフォーマットについては、「6-7 マスタファイル」の「置き換えマスタ」(6-40ページ)を参照してください。

- **インデックス情報定義ファイル**
  マスタファイル名(拡張子なし)+"."+拡張子(idx)

  例　マスタファイル名が、replace.txtのとき、replace.idx

  ▶ 重 要

  インデックス情報定義ファイルは、マスタファイルと同じフォルダに置きます。

- **置き換えマスタ(インデックス情報が付加されたマスタファイル)**
  パス付きで251バイト以内のファイル名+"."+3文字の拡張子
  BTシリーズへファイル転送するときに、マスタファイル名と拡張子が指定できます。

■インデックス情報定義ファイルのフォーマット

インデックス情報定義ファイルのフォーマットは、次のようになります。
";"は単行コメントの記号です。

```
; ファイル定義
[FILE_DEFINITION]; 変換を行うマスタファイルの拡張子を指定
EXTENSION1 = csv
EXTENSION2 = txt

; 言語設定(必ずEXTENSIONの後に定義すること 設定がなければ日本語になります)
;LANG = JAPANESE
;LANG = CHINESE_SIMPLE
;LANG = CHINESE_TRADITIONAL

; INDEXテーブル作成レベル
; インデックス情報のサイズを決定します。
; 指定が無い場合は0になります。
; インデックス情報が大きいほど検索速度は速くなりますが、ファイルサイズが大きくなります。
; INDEXMODE = 0 インデックス情報最小
; INDEXMODE = 1 インデックス情報小
; INDEXMODE = 2 インデックス情報大
; INDEXMODE = 3 インデックス情報最大
; DATACOMP
;  マスタデータの圧縮を行うか指定します。
;  指定が無い場合は0(しない)になります。
[INDEX_MODE]
INDEXMODE = 3
; DATACOMP = 1

; レコード定義
[RECORD_DEFINITION]
RECORDFORMAT = 1  ; 0:固定長　1:可変長
RECORDLENGTH = 0  ; 固定長のときレコード長、可変長のときフィールド数
SEPARATOR = 44    ; 可変長のときのみ指定(1～63) 16進数指定可能 ( ,:; 推奨 )

; ソート定義
[SORT DEFINITION]    ; ソート方法定義
SORTTYPE = 0         ; 0:クイックソート、1:バブルソート※1

; インデックス定義
; 最大9つまで定義可能

; POSITION
;         固定長のときレコード先頭からのバイト数(1～255)※2
;         可変長のとき項目番号(1～フィールド数)

; LENGTH
;         固定長のときのみ指定(1～255)※2
;         可変長のとき無視

[INDEX_1]
POSITION = 1
LENGTH = 0
```

※1　バブルソートはクイックソートよりも変換速度は遅いですが、検索データがマスタファイルの並び順になります。クイックソートは変換速度が速いですが、マスタファイルの並び順になりません。

※2　固定長レコードは、「POSITION」で指定する開始位置から「LENGTH」で指定する長さまでのレコード範囲が行末を超えると、または他のインデックス定義と重なると、置き換えマスタに変換されるときにエラーとなります。

行末を超える例：レコード長が64のとき
[INDEX_1]
POSITION=60
LENGTH=5·····65バイト目は行末を超えています。ここで指定できるのは1～4。

他のインデックス定義と重なる例：
[INDEX_1]
POSITION=1
LENGTH=8

[INDEX_2]
POSITION=5·····INDEX_1で1～8バイトを範囲指定しているため、5バイト目は重複。
LENGTH=5　　ここで指定できるのは、9以降。

※3　インデックス定義の [INDEX_*] の番号は、Search コントロール、GetFirst メソッドの第 1 引数（key）に対応します。

ポイント

圧縮置き換えマスターを使用する場合
インデックス情報定義ファイル内のコメントアウトされた記述";DATACOMP = 1"を有効にします。
INDEXMODEに設定した値が小さいほど圧縮率が高くなります。
インデックス定義数が増えるとファイルサイズが大きくなります。

5 コンパイルエラー一覧

スクリプトコンパイル時に表示されるエラーメッセージ一覧です。

ポイント

エラーメッセージが示す行数は、1行ずれることがあります。

No.	エラー内容	対処方法
1	文法に存在しない書式です	該当行を念入りに調べてください。
2	将来のための予約語です	違う変数名、パッケージ名、メソッド名に変更してください。
3	文字列の定義が複数行にわたっています	1行に収めてください。
4	文字列の長さが8192バイトを超えています	8192バイト以内に収めてください。
5	名前の長さが8192文字を超えています	8192バイト以内の名前にしてください。
6	記号の位置が不正です	該当行を念入りに調べてください。
7	Includeの文法が不正です	該当行を念入りに調べてください。
8	Includeの階層が50を超えています	ファイル構成を見直して単純化してください。
9	ファイルのオープンに失敗しました	ファイルの存在、読み込み禁止属性を確認してください。
10	Includeしたファイル数が100を超えています	ファイル構成を見直して単純化してください。
11	Mainメソッドが定義されていません	Mainメソッドを定義してください。
12	プログラムが大きすぎます	ファイルを分割してください。Includeで読み込んだ合計サイズに対してのエラーですので、単純にファイル分割するのではなく、System:Load( ) でアプリケーションを切り換えるように仕様検討してください。
13	0で除算/剰余する式になっています	式を修正してください。
14	If,While,For,SelectCase,Withの階層が100を超えています	プログラムを単純化してください。
15	Ifに対応するEndIfが存在しません	該当行より前の数行を念入りに調べてください。
16	Whileに対応するWendが存在しません	該当行より前の数行を念入りに調べてください。
17	Forに対応するNextが存在しません	該当行より前の数行を念入りに調べてください。
18	SelectCaseに対応するEndSelectが存在しません	該当行より前の数行を念入りに調べてください。
19	Withに対応するEndWithが存在しません	該当行より前の数行を念入りに調べてください。
20	未定義のコントロールまたはパッケージ名が使用されました	該当行を念入りに調べてください。
21	With文の配置位置が不正です	該当行を念入りに調べてください。
22	指定コントロールに指定プロパティ・メソッドが存在しません	該当行を念入りに調べてください。
23	指定パッケージに指定プロパティ・メソッドが存在しません	該当行を念入りに調べてください。
24	With文に囲まれていないか現行パッケージに指定プロパティ・メソッドが存在しません	該当行を念入りに調べてください。
25	ここではコントロール・パッケージ要素へアクセスできません	該当行を念入りに調べてください。
26	変数名が重複しています	違う変数名にしてください。
27	パッケージ名がコントロール名と重複しています	違うパッケージ名にしてください。
28	パッケージ名が重複しています	違うパッケージ名にしてください。
29	パッケージのプロパティ・メソッド名が重複しています	違う名前にしてください。
30	メソッド名が重複しています	違う名前にしてください。
31	未定義名を参照しています	該当行を念入りに調べてください。

No.	エラー内容	対処方法
32	パッケージ数が1000個を超えています	ファイルを分割してください。Includeで読み込んだ合計サイズに対してのエラーですので、単純にファイル分割するのではなく、System:Load( ) でアプリケーションを切り換えるように仕様検討してください。
33	配列要素数が宣言数を超えています	配列の要素数を下げてください。
34	式が複雑です	式を分割するなど単純にしてください。
35	文字列が多すぎます	不要な文字列を削除するか、ファイルを分割してください。ファイル分割ではIncludeで読み込んだすべてのファイルに対してのエラーですので、単純にファイル分割するのではなく、System:Load( ) でアプリケーションを切り換えるように仕様検討してください。
36	Const変数および配列には代入できません	式を変更してください。
37	Const変数および配列の値は変更できません	式を変更してください。
38	メソッドには代入できません	式を変更してください。
39	WbreakがWhile-Wendに囲まれていない場所で使われています	該当行を念入りに調べてください。
40	WcontinueがWhile-Wendに囲まれていない場所で使われています	該当行を念入りに調べてください。
41	FbreakがFor-Nextに囲まれていない場所で使われています	該当行を念入りに調べてください。
42	FcontinueがFor-Nextに囲まれていない場所で使われています	該当行を念入りに調べてください。
43	Wbreakの数が200個を超えています	プログラムを単純化してください。
44	Fbreakの数が200個を超えています	プログラムを単純化してください。
45	Caseの数が200個を超えています	プログラムを単純化してください。
46	Elselfの数が200個を超えています	プログラムを単純化してください。
47	1行内のメソッドの数が50個を超えています	式を分割するなど単純にしてください。
98	*番目の引数のタイプが定義と異なります	該当行を念入りに調べてください。
149	*番目の引数のConstが定義と異なります	該当行を念入りに調べてください。
150	プログラムが複雑すぎます。	プログラムを単純化してください。
151	Caseラベルとして既にTrueが存在します	Caseラベルの順番を見直してください。
152	同じCaseラベルが既に存在します	Caseラベルを見直してください。
153	数値として扱えません	該当行を念入りに調べてください。
154	プロパティの数が1000個を超えています	プログラムを単純化してください。
155	Const変数および配列は宣言時に初期化してください	該当行を修正してください。
156	配列要素数が不明です	該当行を修正してください。
157	配列要素数が127個を超えています	該当行を修正してください。
158	即値の数が127個を超えています	該当行を修正してください。
159	メソッド定義の引数が50個を超えています	該当行を修正してください。
160	メソッド呼び出しの引数が50個を超えています	該当行を修正してください。
161	メソッド引数が定義と異なります	該当行を修正してください。
162	メソッド引数が定義と異なります	該当行を修正してください。
163	Mainメソッドは引数が不要です	該当行を修正してください。
164	書式が不正か式の場所が不正です	該当行を念入りに調べてください。

6 実行時例外一覧

スクリプトプログラム実行時に表示されるエラーメッセージ一覧です。

種類	番号	内容
プログラム異常	1	0除算
	2	文字列長オーバーフロー
	3	演算子のパラメータの型異常
	4	文字列演算(repeat) の繰り返し数が異常
	5	パラメータが許されない負の数
	6	予約
	7	文字列から数値への変換が不可
	8	配列要素外をアクセスした
	9	配列代入のコピー先の配列要素数がコピー元より少ない
	10	splitのセパレータが存在しない
	11	splitの配列要素が足りない
	12	ワイルドカード文字列比較のパラメータ異常
	13	正規表現文字列比較のパラメータ異常
	14	数値オーバーフロー
	15	unpackの配列要素が足りない
	16	packの配列に終端(0)が存在しない
	17	packの配列の値が範囲外である
	18	system:loadの第1引数が文字列でない
	19	不正な配列サイズを指定した
システム異常	60	スタックオーバーフロー
	61	ヒープメモリ不足
	62	スタックアンダーフロー
	63	レジスタPC異常
	64	レジスタEPC異常
	65	レジスタBP異常
	66	コントロール不具合
	67	存在しない単項演算
	68	存在しない2項演算
	69	存在しない3項演算
コントロール異常	1000	JANコントロールで例外発生
	1100	CODE39コントロールで例外発生
	1200	EAN128コントロールで例外発生
	1300	CODE128コントロールで例外発生
	1400	ITFコントロールで例外発生
	1500	NW7コントロールで例外発生
	1600	CODE93コントロールで例外発生
	1700	TOFコントロールで例外発生
	1700	COOPコントロールで例外発生
	1800	QRコントロールで例外発生
	1900	DataMatrixコントロールで例外発生
	2000	PDF417コントロールで例外発生
	2100	Maxiコントロールで例外発生
	2200	RSSコントロールで例外発生
	2300	Compositeコントロールで例外発生
	2400	BCRコントロールで例外発生
	3000	Ledコントロールで例外発生
	3100	Vibrationコントロールで例外発生
	3200	Buzzerコントロールで例外発生
	3300	Keyコントロールで例外発生
	4000	InputStringコントロールで例外発生

種類	番号	内容
コントロール異常	4100	InputDecimalコントロールで例外発生
	4200	InputDateコントロールで例外発生
	4300	InputByListコントロールで例外発生
	4400	Screenコントロールで例外発生
	5000	Fileコントロールで例外発生
	5200	FileSystemコントロールで例外発生
	5300	LogRecordコントロールで例外発生
	5400	Searchコントロールで例外発生
	5600	Masterコントロールで例外発生
	6000	Printerコントロールで例外発生
	6100	Utilityコントロールで例外発生
	6200	Handyコントロールで例外発生
	7000	Comm1コントロールで例外発生
	7100	Comm2コントロールで例外発生
	7200	CommRFコントロールで例外発生
	7300	Commコントロールで例外発生
	7400	Socketコントロールで例外発生
	7500	FTPコントロールで例外発生
	7600	RemoteAccessコントロールで例外発生
	7700	RemoteDBコントロールで例外発生
	7800	RemoteFileコントロールで例外発生
	7900	RemoteUDコントロールで例外発生
	8000	Networkコントロールで例外発生
	8100	WLANコントロールで例外発生
	8200	Bluetoothコントロールで例外発生
	8300	PPPコントロールで例外発生
	9000	Eventコントロールで例外発生
	10000	Dialogコントロールで例外発生
	11000	Drawingコントロールで例外発生
	12000	LCDコントロールで例外発生
	13000	ZGNコントロールで例外発生
	21000	FileStreamコントロールで例外発生
	22000	Timerコントロールで例外発生
	23000	Serialコントロールで例外発生
	24000	Windowコントロールで例外発生
	25000	Canvasコントロールで例外発生
	26000	GroupBoxコントロールで例外発生
	27000	Buttonコントロールで例外発生
	28000	TextFieldコントロールで例外発生
	29000	TextBoxコントロールで例外発生
	30000	Labelコントロールで例外発生
	31000	CheckBoxコントロールで例外発生
	32000	ListBoxコントロールで例外発生
	33000	RadioButtonコントロールで例外発生
	34000	ComboBoxコントロールで例外発生

7 索 引

本書で使用している用語の索引です。アルファベット、五十音順に並んでいます。

記号

& ····· 3-10
# ····· 5-31
/*～*/ ····· 5-31
// ····· 5-31

英数

And ····· 3-9
BCR ····· 6-34, 7-146
BiCom ····· 7-304
Bluetooth ····· 6-54, 7-274
Bluetooth搭載携帯電話を利用した
　遠隔地通信システム ····· 6-54
BT専用アナログモデムBT-MD1 ····· 6-55
Button ····· 5-17, 7-73
Buzzer ····· 6-39, 7-159
Canvas ·· 5-17, 6-5, 6-7, 6-15, 6-17, 7-16
Case ····· 4-4
Case Else ····· 4-4
Catch ····· 5-28
CheckBox ····· 5-17, 6-22, 6-28, 7-66
CODE128 ····· 7-136
CODE39 ····· 7-135
CODE93 ····· 7-139
ComboBox ····· 5-17, 6-22, 6-28, 7-51
Comm ····· 6-51, 6-52, 6-54, 6-60, 7-252
Comm1 ····· 6-60, 7-286, 7-298
Comm2 ····· 6-59, 7-301
CommRF ····· 6-49, 6-60, 7-291
Composite ····· 7-145
Const ····· 2-4, 2-10
COOP ····· 7-140
DataMatrix ····· 7-143
Dialog ····· 6-23, 6-29, 7-128
Drawing ····· 6-5, 6-7, 6-15, 6-17, 7-117
EAN128 ····· 7-136
Else ····· 4-2
ElseIf ····· 4-2
End If ····· 4-2
End Select ····· 4-4
EndWith ····· 5-13
eq ····· 3-5
Event ····· 7-81
false ····· 2-8
Fbreak ····· 4-9
Fcontinue ····· 4-10
File ····· 7-160, 7-166
FileStream ····· 5-15, 7-160
FileSystem ····· 7-172
Find ····· 2-12
For文 ····· 4-9
FTP ····· 6-50, 6-54, 7-267
ge ····· 3-5
GridBox ····· 7-57
GroupBox ····· 5-17, 7-20
gt ····· 3-5
GUIのプログラミング ····· 5-4
Handy ····· 6-26, 6-32, 6-53, 6-61, 7-320
Hex ····· 2-12
If文 ····· 4-2
Include ····· 5-26
InputByList ····· 6-24, 7-100
InputDate ····· 6-24, 7-98
InputDecimal ····· 6-24, 7-95
InputString ····· 6-24
is ····· 3-7
isBoolean ····· 2-11
isDigit ····· 2-11
isInt ····· 2-11
isString ····· 2-11
ITF ····· 7-137
JAN ····· 7-133
Key ····· 6-26, 7-315
Label ····· 5-17, 7-41
LCD ····· 6-61, 7-113
le ····· 3-5
Led ····· 6-39, 7-157
Left ····· 2-12
length ····· 2-11
ListBox ····· 5-17, 6-22, 6-28, 7-44
ListBox/ComboBox/InputByList ····· 6-40
ListTable ····· 7-78
Load ····· 5-27
LogRecord ····· 7-203
lt ····· 3-5
mainメソッド ····· 1-3, 5-3, 5-9
Master ····· 6-40, 7-190
Merge ····· 2-16
Mid ····· 2-12
ne ····· 3-5
Network ····· 6-55, 6-57, 6-60, 7-244
Next ····· 4-9
nil ····· 2-8
not ····· 3-8
NW7 ····· 7-138
OCR ····· 7-153
Or ····· 3-9
Pack ····· 2-16
PackBin ····· 2-16, 2-17
PDF417 ····· 7-144
PPP ····· 6-54, 7-282
Printer ····· 6-58, 7-283
QR ····· 7-142
RadioButton ····· 5-17, 6-22, 6-28, 7-69
RemoteAccess ····· 6-49, 7-214
RemoteDB ····· 6-49, 7-217
RemoteFile ····· 6-49, 7-229
RemoteUD ····· 6-49, 7-236
Remove ····· 2-12
Repeat ····· 2-12
Return ····· 5-10
Rfind ····· 2-12
Right ····· 2-12
RS-232Cポートを使用した通信 ····· 6-59
RSS ····· 7-141
Screen ····· 6-4, 6-8, 6-18
Search ····· 7-211
Select Case ····· 4-4
Select文 ····· 4-4
Serial ····· 5-15, 6-59, 6-60, 7-249

size ····· 2-16
Socket ····· 6-50, 7-260
split ····· 2-19
SQLite ····· 7-183
SQLiteCommand ····· 7-184
SQL文の送信 ····· 7-221
System ····· 5-27, 7-2
TextBox ····· 5-17, 6-22, 6-28, 7-34
TextField ····· 5-17, 6-22, 6-28, 7-23
Timer ····· 5-15, 7-318
TOF ····· 7-140
true ····· 2-8
Unpack ····· 2-19
UnpackBin ····· 2-19
USBポートを使用した通信 ····· 6-59
UserObj ····· 7-313
Utility ····· 7-308
Vibration ····· 6-39, 7-158
VRAM ····· 6-3, 6-13
Wbreak ····· 4-7
Wcontinue ····· 4-7
Wend ····· 4-7
weq ····· 3-5
While文 ····· 4-7
Widget共通イベントプロパティ ····· 7-7
Widget共通プロパティ ····· 7-5
Widget共通メソッド ····· 7-8
Widgetイベント ····· 5-21
Widget共通 ····· 7-5
Widget操作コントロール ····· 5-15, 5-17, 7-5
Widgetの幅と高さ ····· 7-11
Widgetの表示／非表示 ····· 7-10
Window ····· 5-17, 7-12
With ····· 5-13
WLAN ····· 6-60, 7-240

あ

アイコンボタン ····· 7-75
アスキーコード表 ····· 付-2
イベント ····· 5-20
イベント駆動型プログラム ····· 5-4, 5-20
イベント取得の終了 ····· 7-81
イベント対応コントロール ····· 5-20
イベントの取得 ····· 5-22
イベントの取得と終了 ····· 7-81
イベントの処理 ····· 5-22
イベントの登録 ····· 5-21
色の指定 ····· 7-9, 7-120
インデックス情報定義ファイル ····· 6-46
インデックス情報定義ファイルの
フォーマット ····· 付-8
ウィンドウエリア ····· 6-2, 6-12
ウィンドウエリアの表示 ····· 6-3, 6-12
ウェイクアップ機能 ····· 6-53
液晶表示部(LCD)の設定 ····· 7-103
エミュレータ表示 ····· 6-4
エミュレートウィンドウ ····· 5-18, 7-14
エラー情報(BiCom) ····· 7-305
エラー情報(Bluetooth) ····· 7-275
エラー情報(Comm) ····· 7-256
エラー情報(Comm1) ····· 7-299
エラー情報(Comm2) ····· 7-302
エラー情報(CommRF) ····· 7-296
エラー情報(File) ····· 7-169
エラー情報(FileStream) ····· 7-164
エラー情報(FileSystem) ····· 7-179
エラー情報(Master) ····· 7-196
エラー情報(Network) ····· 7-247
エラー情報(RemoteAccess) ····· 7-215
エラー情報(RemoteDB) ····· 7-222
エラー情報(RemoteFile) ····· 7-232
エラー情報(RemoteUD) ····· 7-238
エラー情報(Search) ····· 7-212
エラー情報(Serial) ····· 7-250
エラー情報(Socket) ····· 7-262
エラー情報(WLAN) ····· 7-243
演算子の優先順位 ····· 3-11
オートオフ ····· 6-35
オートパワーオフ機能 ····· 6-61
置き換えマスタ ····· 6-40, 6-44, 付-6
置き換えマスタのフォーマット ····· 6-45
置き台イベント ····· 5-21
オプションデータの読み書き ····· 7-221
オペレータ演算子の読み書き ····· 7-221
親Widget ····· 5-17
親WidgetとtargetWidget ····· 7-9
折り返し表示 ····· 6-6, 6-11, 6-16, 6-21

か

カーソル位置と入力動作 ····· 6-26, 6-32
外字の登録 ····· 7-310
各Widget操作コントロール ····· 6-5, 6-15
画像の貼り付け ····· 7-18
画面表示
（BT-600/1000/1500シリーズ） ···· 6-12
画面表示
（BT-W100/W80/W70シリーズ） ····· 6-2
可変長形式 ····· 6-44
キーイベント ····· 5-21
キークリック音量設定 ····· 7-326
キーテーブル ····· 7-316
キー入力センス ····· 7-329
キー入力待ち ····· 7-328
キーの検出 ····· 6-25, 6-31
キーの割り付け ····· 6-26, 6-32
行間ギャップ ····· 7-108, 7-121
空白文字 ····· 5-30
グラフィック座標 ····· 6-10, 6-20
繰り返し処理 ····· 4-7
検索キーの指定 ····· 7-193
コード読み取りイベント ····· 5-21
子Widget ····· 5-17
高速検索 ····· 7-211
固定長形式 ····· 6-44
コメント ····· 5-31
コントロール ····· 5-3, 5-14
コントロールのタグ名 ····· 2-2
コントロールへのアクセス ····· 5-14
コントロール名の省略 ····· 5-13
コンパイルエラー一覧 ····· 付-10

さ

サーバPCとの通信 ····· 6-48

サーバPCやプリンタとの通信 …… 7-240
サーバアプリケーション …… 6-49
サーバ上のデータベースへのアクセス要求 7-219
サーバ上のファイルへのアクセス要求 … 7-231
サーバとのデータ送受信 …… 7-214
サーバへユーザデータの送信要求 …… 7-237
サーバライブラリ …… 6-49
座標 …… 6-9, 6-19
座標の指定 …… 7-11
算術演算子 …… 3-2
シークポインタ …… 7-162, 7-167
指示マスタ …… 6-41, 6-46
指示マスタのフォーマット …… 6-46
実行時エラー …… 5-29
実行時例外 …… 5-28
実行時例外一覧 …… 付-12
シフトキーを使用した半角英字入力 …… 7-87
順序指定の有無 …… 7-194
白黒反転 …… 6-35
数値比較演算子 …… 3-4
スキャンタイムアウト時間 …… 6-34
スクリプトとは …… 1-2
スクリプトの構成 …… 5-2
スクリプトの特長 …… 1-2
スクリプトファイル名 …… 2-2
スクリプトプログラムの中で定義する名前 … 2-2
スクロール表示 …… 7-110
図形表示 …… 6-7, 6-17, 6-18
ステータスエリア …… 6-2, 6-12, 7-110
ステータスエリア表示 …… 6-4, 6-13, 6-14
制御文字 …… 2-6
線、四角形の表示 …… 6-8
選択マスタ …… 6-42, 6-47
選択マスタのフォーマット …… 6-47
ソケット通信 …… 6-50
ソフトウェアトリガ …… 6-37

た

代入(数値) …… 2-7
代入(文字列) …… 2-6
タイマーイベント …… 5-21
タイムアウト時間 …… 6-34
タグ指定コントロール …… 5-15
タグ名 …… 5-15
タッチパネルの構成 …… 5-17
タッチパネルの表示 …… 5-4
タッチパネル表示 …… 6-3, 6-12
ダブルクリック読み …… 6-35
通信ライブラリ …… 6-52
データ転送ソフト …… 6-51
テキスト座標 …… 6-9, 6-19
テキストデータの送信、受信 …… 7-295
テキストデータの読み込み、
　書き込み …… 7-162, 7-167
デバイス設定、メニュー表示、キー入力 … 7-315
デバイスの動作設定 …… 6-39, 7-157
デバッグ …… 7-328
電卓機能 …… 7-30, 7-97
動作の一時停止 …… 7-328
ドライブ情報の取得 …… 7-177
ドライブへのアクセス …… 7-163, 7-178
トリガモード …… 6-35

な

名前の参照順位 …… 5-24
名前の参照、スコープ …… 5-23
名前の重複 …… 5-24
名前の付け方 …… 2-2
二度読み防止時間 …… 6-34
日本語を入力する …… 7-26, 7-89
入力確定と取り消し …… 6-25, 6-31
入力処理
　(BT-600/1000/1500シリーズ) …… 6-28
入力処理
　(BT-W100/W80/W70シリーズ) …… 6-22

は

バーコード合成読み取り機能 …… 6-38
バーコード読み取り制御 …… 7-150
排他選択 …… 7-71
バイナリデータの送信、受信 …… 7-295
バイナリデータの読み込み、
　書き込み …… 7-163, 7-168
バイナリ表記 …… 2-6
配列変数 …… 2-9
配列変数の参照・代入 …… 2-10
配列変数の初期化 …… 2-10
配列変数の宣言 …… 2-10
配列変数のプロパティ …… 2-16
配列変数のメソッド …… 2-16
配列変数名 …… 2-9
バックライトのオートオフと明るさ調整 … 6-61
パッケージ …… 1-3, 5-3
パッケージの定義 …… 5-11
パッケージへのアクセス …… 5-12
パッケージ名 …… 5-12
パッケージ名の省略 …… 5-13
離して読み …… 6-36
否定演算子 …… 3-8
描画の指定 …… 7-119
表示 …… 7-103
表示位置の指定 …… 7-14, 7-37, 7-47
表示更新 …… 7-110
表示データの指定 …… 7-107
表示できない文字の扱い …… 7-110
表裏反転 …… 6-34
ファイルオープン、クローズ …… 7-193
ファイル検索 …… 7-177
ファイルのインクルード …… 1-3
ファイルのコメント …… 1-3
ファイルの操作 …… 7-160
ファイルの送受信 …… 7-256
ファイルの連結 …… 7-177
プリンタとの通信 …… 6-58
プログラムの起動 …… 5-27
プログラムの実行 …… 5-27
プロパティへのアクセス …… 5-13
分岐条件の入れ子 …… 4-6
分岐処理 …… 4-2
別スクリプトの呼び出し …… 7-2
別スクリプトの読み込み …… 5-26
ヘッダ情報 …… 7-206
別プログラムの実行 …… 5-27
変数 …… 2-3

変数の参照･代入 ･･･2-4
変数の初期化 ･･･2-4
変数の宣言 ･･･2-3
変数の内部型 ･･･ 付-4
変数のプロパティ ･･･ 2-11
変数のメソッド ･･･2-12, 2-19
変数名 ･･･2-3
ポートの状態 ･･･ 7-246
ポートの状態(Bluetooth) ･･･ 7-275
ポートの状態(Comm1) ･･･ 7-299
ポートの状態(Comm2) ･･･ 7-302
ポートの状態(CommRF) ･･･ 7-295
ポートの状態(Serial) ･･･ 7-250
ポートの状態(WLAN) ･･･ 7-242

ま

マスタファイル ･･･ 6-40
マスタファイル照合 ･･･ 7-194
マスタファイルとの照合 ･･･ 7-190
マスタファイルの構成 ･･･ 6-43
マスタファイル名 ･･･ 6-43
無線デバイスイベント ･･･ 5-21
メソッド ･･･5-3, 5-5
メソッド引数 ･･･5-6
メソッドから抜ける ･･･5-10, 5-11
メソッドの定義 ･･･5-5
メソッドの戻り値 ･･･5-6
メソッドの呼び出し ･･･5-7
メソッド引数 ･･･5-7
メソッドへのアクセス ･･･ 5-13
メソッド名 ･･･5-5
メッセージボックス表示 ･･･7-131, 7-326
メニュー表示 ･･･7-130, 7-327
モーメンタリ ･･･ 6-36
文字間ギャップ ･･･7-108, 7-121
文字と図形の表示－タッチパネル表示－ ･･･6-5
文字と図形の表示(CUIプログラム) ･･･ 6-18
文字と図形の表示(GUIプログラム) ･･･ 6-15
文字と図形の表示－エミュレータ表示－ ･･･6-8
文字列比較演算子 ･･･3-5
文字列連結演算子 ･･･ 3-10

や

ユーティリティ ･･･ 7-308
読み取り一致回数 ･･･ 6-34
読み取り動作の設定 ･･･ 6-34, 7-133
読み取り動作モード ･･･ 6-35
読み取りモード ･･･ 6-34
予約語一覧 ･･･ 付-3
予約語の例外規則 ･･･ 5-25

ら

リクエストマネージャ ･･･6-48, 6-49
ルートウィンドウ ･･･5-18, 7-14
レコードの削除､復帰 ･･･ 7-195
レコードの追加､修正 ･･･ 7-196
レコード番号を指定して照合 ･･･ 7-195
連続読み ･･･ 6-36
ログファイルの操作 ･･･ 7-203
ログレコードの書き込み､読み込み ･･･ 7-204
ログレコードの削除､修正 ･･･ 7-205
論理型比較演算子 ･･･ 3-7
論理積 ･･･ 3-9
論理値 ･･･ 2-8
論理和 ･･･ 3-9

わ

枠種別 ･･･ 7-8

改訂履歴

印刷年月日	版　数	改 訂 内 容
2016年10月	初 版	
2018年 1月	2 版	
2018年 5月	3 版	
2019年 4月	4 版	
2019年10月	5 版	

保証について

1. 対象製品

以下に規定する保証は、当社が製造･販売する製品（以下「対象製品」という）に適用します。
なお、対象製品に内蔵されているリレーや電池などの消耗品は対象外とさせていただきます。

2. 保証期間

対象製品の保証期間は、貴社のご指定場所に納入後1年間とします。

3. 保証範囲

（1）上記保証期間内に当社の責任による故障が発生した場合は、無償での代替品との交換または修理をさせていただきます。但し、保証期間内であっても、次に該当する故障の場合は保証対象外とさせていただきます。なお、代替品との交換または修理を行なった場合でも保証期間の起算日は対象製品の当初ご納入日とさせていただきます。

① 取扱説明書、ユーザーズマニュアル、別途取り交わした仕様書などに記載された以外の不適当な条件・環境・取り扱い・使用方法に起因した故障。

② お客様の装置または、ソフトウエアの設計内容など、対象製品以外に起因した故障。

③ 当社以外による改造、修理に起因した故障。

④ 取扱説明書、ユーザーズマニュアルなどに記載している消耗部品が正しく保守、交換されていれば、防止できたと確認できる故障。

⑤ 当社出荷時の科学・技術水準では、予見が不可能だった事由による故障。

⑥ その他、火災、地震、水害などの災害及び電圧異常など当社の責任ではない外部要因による故障。

（2）保証範囲は上記（1）を限度とし、対象製品の故障に起因するお客様での二次損害（装置の損傷、機会損失、逸失利益等）及びいかなる損害も保証の対象外とさせていただきます。

4. 適用用途

当社製品は、一般工業向けの汎用品として設計･製造されております。
従いまして、下記のような用途での使用は意図しておりませんので適用外とさせていただきます。
ただし、事前に当社までご相談いただき、お客様の責任において製品の仕様をご確認のうえ、定格･性能に対してご了承いただき、必要な安全対策を講じていただく場合は適用可能とさせていただきます。
なお、この場合においても保証範囲は上記と同様といたします。

① 原子力発電、航空、鉄道、船舶、車両、医療機器等の人命や財産に多大な影響が予想される設備

② 電気、ガス、水道等の公共設備

③ 屋外での使用および、それに準ずる取扱説明書などで規定していない条件・環境での使用

④ 上記①及び②に準じる安全に関して高度な配慮と注意が要求される用途

全商品、送料無料で
当日出荷

必要な時に、必要な量だけ
在庫不要でトータルコストを削減

■ お問い合わせ 0120-267-911
最寄りの担当営業所に直接つながります。
一部のIP電話からはご利用いただけません。

■ 情報サービス www.keyence.co.jp
カタログ、取扱説明書、マニュアル、CADデータ等をダウンロードできます。

■ 輸出書類サービス www.keyence.co.jp/yushutsu
輸出に必要な書類をその場でダウンロードできます。

株式会社 キーエンス 本社・研究所／自動認識事業部
〒533-8555 大阪市東淀川区東中島1-3-14

自動2-1036

1109-6 136399