TS-BASIC for Windows

- reference manual -

目次

TS-BASIC for Windows について

TS-BASIC for Windows は C#で書かれた汎用的に動作する tsBasicCore を Windows 用に拡張した.NET Framework で動作する、BASIC インタープリタです
コードは入力時に中間コードに変換してメモリに格納します
現状ではテキストスクリーンの他に、グラフィックスクリーンを１ページ使用できます
スクリーンサイズはグラフィックスクリーンが８００×６００ドット、テキストスクリーンが８１×３１キャラクターです
（使用できるフォントサイズの関係でテキストスクリーンのサイズが中途半端になっています）
これらのサイズは環境変数で取得できます
また、現状では tsBasicCore は日本語に対応していますが、Windows 用のコンソールが日本語入力に対応できていないので、まだ日本語は使用できません

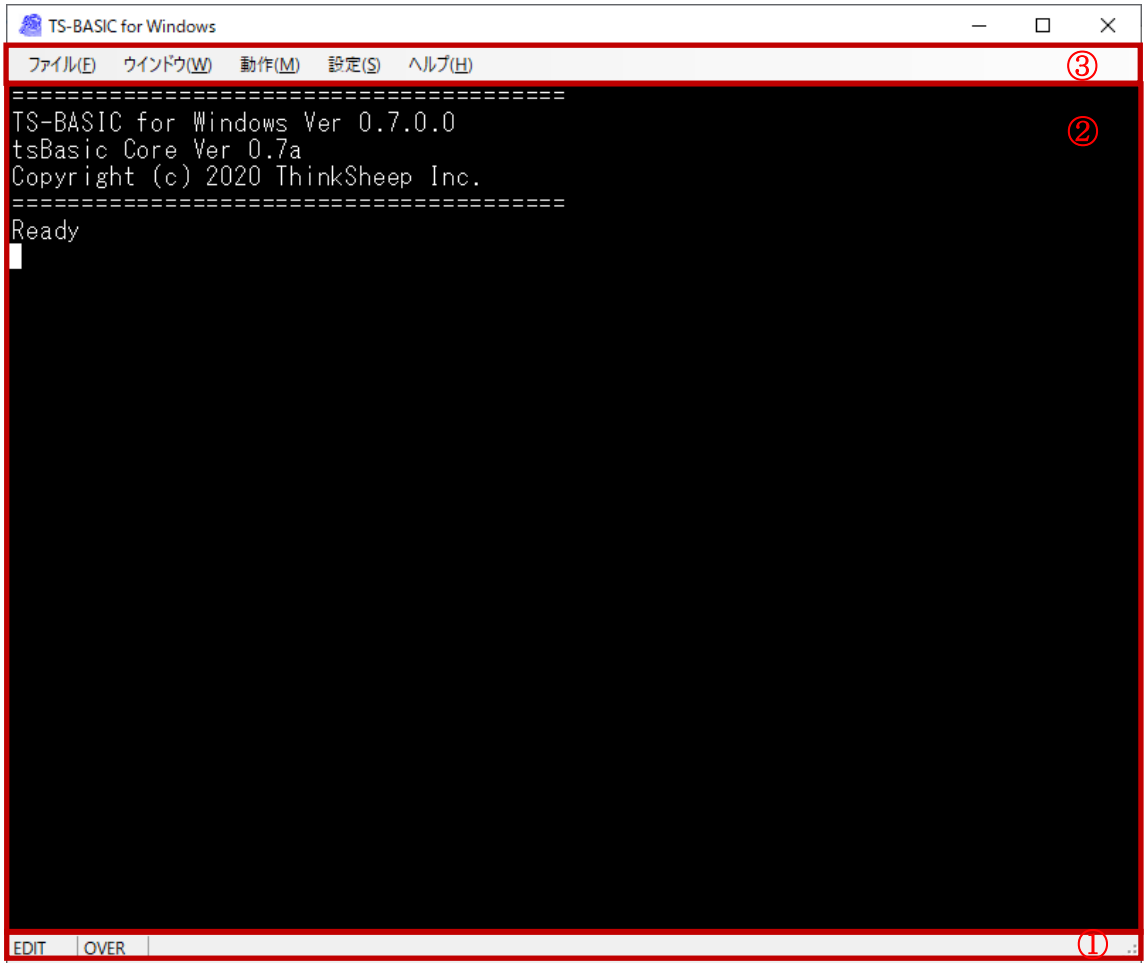
動作環境

windows7、windows10 で動作確認しています
動作には.NET Framework4.6.1 が必要です
.Net Framework はマイクロソフトのページからダウンロードできます
オンラインインストーラ
https://www.microsoft.com/ja-JP/download/details.aspx?id=49981
オフラインインストーラ
https://support.microsoft.com/ja-jp/help/3102436/the-net-framework-4-6-1-offline-installer-for-windows

本アプリにはインストーラは存在しません
ダウンロード後、適当な場所に保存し exe ファイルを実行してください

メイン画面

TsBasicWindows.exe を起動すると下記の画面が表示されます



プレビュー領域、顧客管理領域、見積書管理領域はサイズを自由に変える事ができます

① メインメニュー
② コンソール領域
③ ステータスバー

メインメニュー

ファイル(F)

プログラムを開く

オープンファイルダイアログで BASIC プログラムを開きます

※ＢＡＳＩＣコマンドの『LOAD “ファイル名”』と同じです

プログラムを保存

セーブファイルダイアログで BASIC プログラムを保存します

※BASIC コマンドの『SAVE “ファイル名”』と同じです

標準フォルダを開く

標準の BASIC プログラム保存先フォルダをエクスプローラで開きます

※標準フォルダは実行ファイルと同じフォルダにある SaveData フォルダになります

終了

TS-BASIC for Windows を終了します

ウインドウ(W)

ノーマル

コンソール領域を標準サイズで表示します

x2

コンソール領域を２倍に拡大して表示します

ストレッチ

コンソール領域をウインドウサイズに合わせて拡大、縮小表示します

※ストレッチモードの時はオートサイズの設定は無効になります

オートサイズ

コンソール領域の表示サイズに合わせてウインドウを自動でリサイズします

動作(M)

実行

読み込まれた BASIC プログラムを実行します

※BASIC コマンドの『RUN』と同じです

停止

実行中の BASIC プログラムを停止します

※BREAK キーを押したのと同じです

設定(S)

環境設定

環境設定ウインドウを開き、各種設定をします

ヘルプ(H)

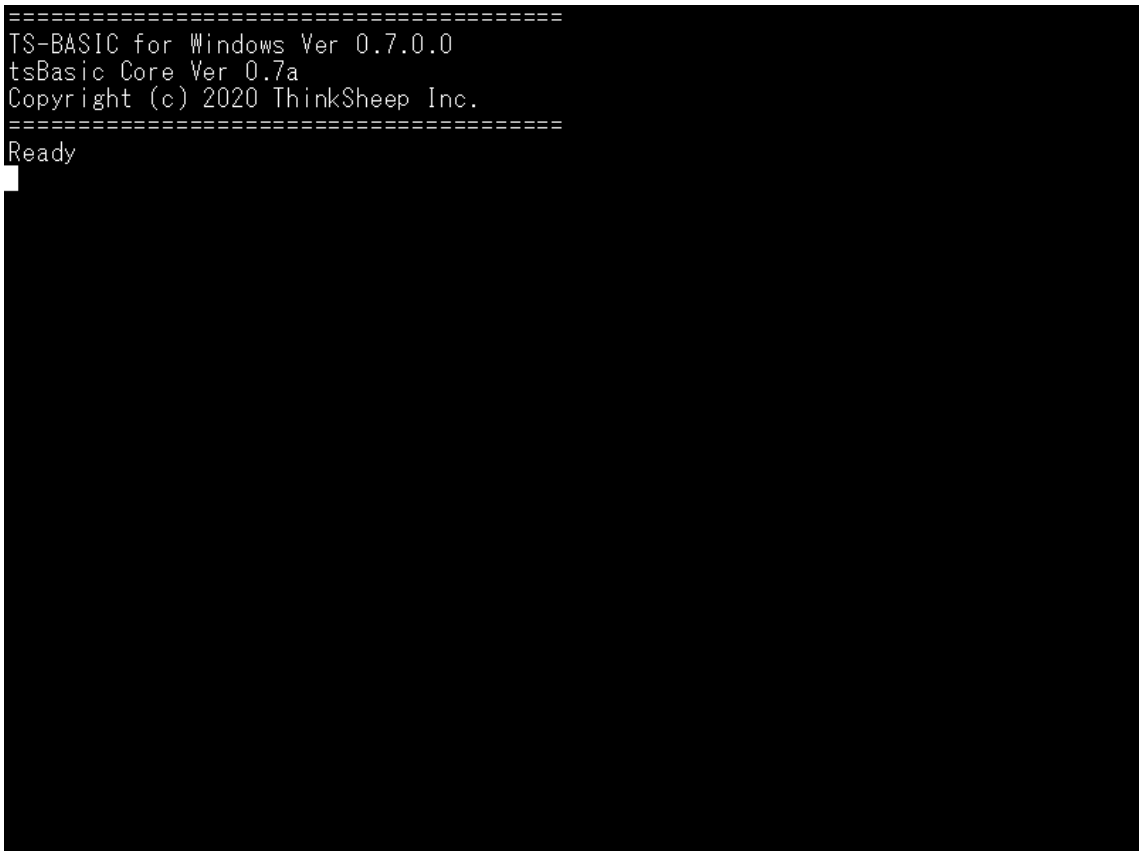
ヘルプ　　F1

ヘルプファイル（このファイル）を開きます

バージョン情報

バージョン情報ウインドウを開きます

コンソール領域



BASIC のプログラムを入力したり、コマンドを実行する画面です
現在はテキストを入力する画面に重ねて、グラフィックを表示する画面が 1 枚使用できます
画面上をクリックするとカーソルが移動します

ステータスバー

① 動作状態

インタープリタの動作状態を表します

EDIT：編集状態です

RUN：BASIC プログラムの実行状態です

DIRECT：コマンドの直接実行状態です

② 編集状態

現在の編集状態です

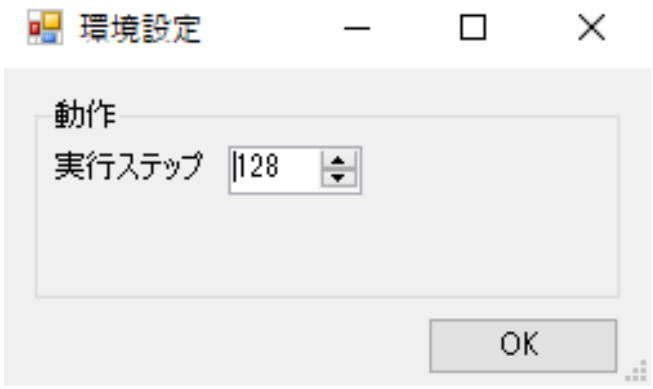
OVER：上書きモード

INS：インサートモード

※編集状態は IINSERT キーを押すと切り替わります

※②の部分をクリックする事でも切り替える事ができます

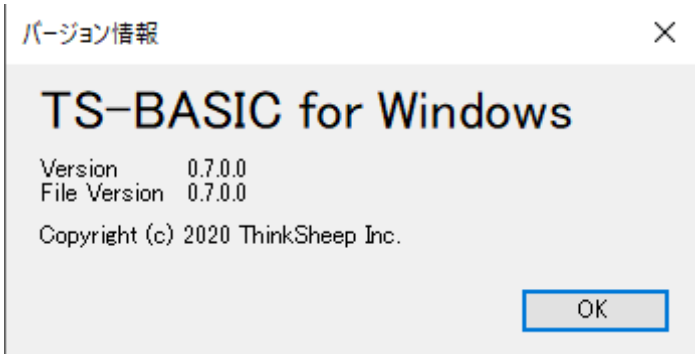
環境設定ウインドウ



自動読込/保存

TS-BASIC for Windows のコアはマルチスレッドで動作していますが、画面の更新をするためにメインスレッドとの同期を行います
この画面更新間で実行される中間コードのステップ数を指定します
設定できる範囲は１～２５６で、数値が大きい程 BASIC プログラムの実行速度が早くなりますが、画面更新の間隔が飛んだ様に見えます

バージョン情報ウインドウ



バージョン情報が表示されます

TS-BASIC for Windows の基本仕様

TS-BASIC for Windows はコンソール領域内での操作により、一般的な BASIC インタープリタの動作をします
コンソール画面で Enter キーを押した時点で、カーソルのある行をコマンド解析して、先頭が行番号ならプログラムとして登録され、行番号で無い場合はコマンドとして実行されます
各コマンドは『:』で区切る事で 1 行に複数書く事ができます
TS-BASIC for Windows はインタープリタの実行部分を tsBasicCore と言うクラスで実装しているので、今後 tsBasicCore にて実行される部分に関しては TS-BASIC と略します

値

TS-BASIC で扱える値は、整数型、実数型、文字列型の 3 種類です

整数型

３２ビットの符号付きの整数値です
先頭に『0x』を付ける事で１６進数として記述できます
使用できる演算子と優先順位については演算子を参照してください

実数型

３２ビットの符号付き実数（FLOAT）です
使用できる演算子と優先順位については演算子を参照してください

文字列型

『”』（ダブルクォート）で囲んだ文字の集合を文字列として扱います
文字列型の値は=と+のみ演算子として使用できます

ラベル

ラベルは『*』で始まる英数字と_（アンダーバー）が使用できます
ラベルは行番号の代わりに使用する事ができる、行の位置を表す物です

変数

変数は英文字で始まる英数字と_（アンダーバー）.（ピリオド）が使用できます
変数には各種の型の値を保存できます

整数型

変数名の後の%を付ける事で実数型変数になります
整数型の変数には３２ビットの符号付きの値を格納できます
例）data%=10+5

実数型

変数名の後の!を付けるか、何も付けない場合は実数型変数になります
実数型変数には３２ビットの符号付き実数（FLOAT）が格納できます
例）data=10.5+2.3

文字列型

変数名の後に$を付ける事で文字列型変数になります
文字列型変数は=と+のみ演算子として使用できます
例）data$=”TEST PROGRAM”+” 001”

拡張型

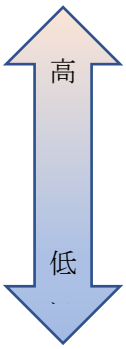
変数名の後に@を付ける事で拡張型変数になります
拡張型変数は特殊な変数で、演算には使用できません
拡張型変数は画像データなど、特殊なデータの集合を記憶して、一部のグラフィック系コマンドなどで使用します

配列変数

各型の変数の後ろに（ ）を付けると配列変数として使用できます
配列の要素は０から『DIM』コマンドで宣言した値までで、１次元から３次元配列まで使用できます
例）dim data(2,3)
　　data(0,3)=10+5

演算子

TS-BASIC で使用できる演算子には以下の物があります



記号	種類	整数型	実数型	文字列型
-,NOT	単項演算子	○	○	×
*,/	二項演算子	○	○	×
+,-	二項演算子	○	○	△
=,<,>	比較演算子	○	○	○
AND,OR,MOD	論理演算子	○	×	×
=	代入	○	○	○

環境変数

TIME$

現在時刻を『TT:MM:SS』のフォーマットで文字列を返します

DATE$

現在日時を『YYYY/MM/DD』のフォーマットで文字列を返します

CONSOLE.WIDTH

テキスト画面の横方向の文字数を返します

CONSOLE.HEIGHT

テキスト画面の縦方向の文字数を返します

GRAPHIC.SCREEN

使用できるグラフィック画面の枚数を返します

SCREEN.WIDTH

グラフィック画面の横方向のドット数を返します

SCREEN.WIDTH

グラフィック画面の横方向のドット数を返します

TOUCH.INFO

マウスの情報を返します

この環境変数は２次元配列になっており TOUCH_INFO(pn,pt)の様に取得します

pn はマルチタッチに対応したポイント番号で０～４の５ポイントまで保持しますが、Windows の場合は 0 のみ有効です

pt は各ポイントの取得したい情報を以下の様に指定します

０：ポイント状態（0:NONE 1:KLICK 2:MOVE）

１：X 座標

２：Y 座標

３：最初にクリックされた X 座標

４：最初にクリックされた Y 座標

５：最初にクリックされた X 座標からのオフセット

６：最初にクリックされた Y 座標からのオフセット

コマンド

制御系コマンド

CLS

テキスト画面をクリアしてカーソルを左上に移動します

DIM 変数名(要素数[,要素数],[,要素数])[,…

配列変数を宣言します

例）DIM A(10,10)

END

プログラムを終了します

FILES ["パス"][,下層フォルダ表示]

ファイルの一覧を表示します

パスを省略した場合は標準フォルダ内のファイルを表示します

下層フォルダ表示を 0 以外に指定すると下層のフォルダ内のファイルも表示します

FOR…TO…[STEP] ～ NEXT[変数名]

FOR で指定された変数を STEP ずつ TO まで増加しながら、NEXT までの処理を実行します

例）FOR I=0 TO 100:PRINT i:NEXT

※NEXT の後ろに変数を指定する事でループの流れが分かりやすくなります

GOTO 行番号 or ラベル

指定行に制御を移します

GOSUB 行番号 or ラベル

指定行のサブルーチンコールを行います

IF 論理式 THEN 行番号 or ラベル or コマンド [ELSE] 行番号 or ラベル or コマンド

論理式が成立した場合、THEN 以下のコマンドを実行する
例）IF A=0 THEN *LABEL

LIST [行番号 or ラベル][-行番号 or ラベル]

指定範囲の BASIC プログラムを表示する

LOAD ”ファイル名”

指定されたファイル名の BASIC プログラムを読み込みます
フルパス指定以外は標準フォルダからの相対パスになります

LOCATE x,y

カーソルを指定位置に移動します

NEW

BASIC を消去し、変数、スタックをクリアします

ON インデックス GOTO(GOSUB) ,...,...,...

インデックスの値に応じて制御を移します
インデックスがジャンプテーブルの範囲外、または指定が無い場合はスルーします

PRINT 値[;]

カーソル位置に値を表示して改行します
最後に『;』が付いている場合は改行しません

RANDOMIZE(値)

指定された値を使って乱数を初期化します

REM コメント

以降、行の最後までをコメントにします

RENUM [新行番号][,旧行番号][,STEP]

行番号を書き換えます
パラメータを省略した場合は全て 10 です

RETURN

サブルーチンから制御を戻します

RND(上限値)

上限値未満のランダムな整数を返します

RUN 行番号 or ラベル

指定行から BASIC プログラムを実行します
パラメータを省略した場合は、先頭行から実行します

SAVE "ファイル名"

指定されたファイル名の BASIC プログラムを保存します
フルパス指定以外は標準フォルダからの相対パスになります
フォルダが存在しない場合はフォルダを作成します

SWAIT 前回からの待ち時間

前回『swait』が呼ばれてから指定された時間が経過するまで処理を待ちます
待ち時間は ms 単位で指定します

WAIT 待ち時間

指定された時間が経過するまで処理を待ちます
待ち時間は ms 単位で指定します

READ 変数

DATA 文から値を読み取り変数に格納する

例）READ a(i)

DATA 値[,値…]

READ 文で読み込まれる値を格納する

例）DATA 10,20,30

RESTORE 行番号 or ラベル

READ 文で読み込む値を指定する

例）RESTORE *DATA

GREAD w h 拡張変数

指定したサイズのグラフォックデータを作成して data 文からデータを読み込みます

入力系コマンド

INPUT [“メッセージ”[;/,]][文字列変数]

メッセージが指定されている場合はメッセージを表示します
メッセージの後に『,』が指定されている時は改行を行い『,』が指定されている時は改行を行いません
文字入力モードに入り Enter が押されるまで制御は戻りません
Enter が押されると指定された文字列変数に値を格納して制御が戻ります
例）input ”INPUT:”;S$

INKEY$

キーが押されている場合はキーを返し、何も押されていない場合は空白文字『 』
を返します
例）k$=inkey$

KEY_STATE(仮想キーコード)

リアルタイムで指定したキーが押されているかを調べます
複数のキーの状態を同時に調べる事ができます
例）print kay_state(32)

ビット演算系コマンド

AND

ビット毎の AND を返します

例）print 0xfc and 0x3f

OR

ビット毎の OR を返します

例）print 0xfc or 0x3f

XOR

ビット毎の xor を返します

例）print 0xfc xor 0x3f

NOT

ビット毎の not を返します

例）print not 0x3f

文字列操作系コマンド

ASC(“文字”)

指定した文字のキャラクターコードを返します

例）s=asc(“A”)

CHR$(キャラクターコード)

指定したキャラクターコードの文字を返します

例）s$=chr$(65)

HEX$(数値)

数値を表す 16 進数文字列を返します

例）s$=hex$(255)

LEFT$(“文字列”,取得する長さ)

文字列の左から任意の長さの文字列を返します

例）s$=left$(“abcdef”,3)

MID$(“文字列”,抜き出す位置[,抜き出す長さ])

文字列の任意の位置から、任意の長さの文字列を返します

例）s$=mid$(“abcdef”,1,3)

RIGHT$

文字列の右から任意の長さの文字列を返します

例）s$=right$(“abcdef”,3)

STR$(数値)

数値を表す文字列を返します

例）s$=str$(1234)

STRING$(繰り返し回数,“文字列”)

指定した回数繰り返した文字列を返します

例）s$=string$(10,“A”)

VAL(“文字列”)

文字列が表す数値を返します

例）a=val(“1234”)

算術演算系コマンド

SIN(角度(ラジアン))

指定された角度のサインを返します

例）print sin(radian(30))

COS(角度(ラジアン))

指定された角度のコサインを返します

例）print cos(radian(30))

TAN(角度(ラジアン))

指定された角度のタンジェントを返します

例）print tan(radian(30))

ASIN(角度)

サインが指定数となる角度（ラジアン）を返します

例）print degree(asin(0.5))

ACOS(角度)

コサインが指定数となる角度（ラジアン）を返します

例）print degree(acos(0.5))

ATAN(角度)

タンジェントが指定数となる角度（ラジアン）を返します

例）print degree(atan(0.5))

SQR(値)

平方根を返します

例）print sqr(9)

RADIAN(角度)

角度をラジアンになおします

DEGREE

ラジアンを角度になおします

SGN

符号を取得します

INT

小数点以下を切り捨てます

ファイル操作系コマンド

CHDIR “フォルダ名”

指定したフォルダに移動します

MKDIR “フォルダ名”

指定した名前でフォルダを作成します

KILL “ファイル名/フォルダ名”

指定したファイル又はフォルダを削除します

フォルダ内にファイルがあってもファイルごと削除します

グラフィック系コマンド

グラフィック系コマンドは全てのコマンドの先頭にスクリーン番号を指定します
省略した場合は『GCLS』は全てのグラフィック画面、それ以外のコマンドはスクリーン番号０に対して操作します
現状ではグラフィック画面は１枚だけなのでスクリーン番号は省略しても問題ないでしょう
またカラー指定は３２ビット整数で指定し、A,R,G,B それぞれ８ビット（０～２５５）で、アルファブレンドをサポートしています
カラーデータは ARGB の形式になるので 0xFFFF0000 の場合は赤になります
各描画コマンドで色指定を省略した場合は白（0xFFFFFFFF）で描画します

GCLS [スクリーン番号]

指定したスクリーン番号のグラフィック画面をクリアします

LINE [スクリーン番号,](x1,y1)-(x2,y2)[,color],[,"B"/"BF"]

指定した座標間で直線を描画します
"B"を指定した場合は指定座標を対角線とした矩形を描画します
"BF"を指定した場合は矩形を塗り潰します
例）line(0,0)-(800,600),0xffff0000
　　Line(0,0)-(800,600),0xffff0000,"BF"

CIRCLE [スクリーン番号,](x1,y1)-(x2,y2)[,color][,"F"][,開始角度,ステップ]

指定した座標の矩形に内接する円を描画します
"F"を指定すると塗り潰し描画を行います
開始角度とステップは度で指定します
省略した場合は完全な縁を描画します
例）circle(0,0)-(100,100),0xffff0000,"F",90,270

PAINT [スクリーン番号,](x,y)[,color]

指定した座標の色を開始色として開始色以外で閉じられた空間を塗り潰します
例）paint(100,100),0xffff0000

PSET [スクリーン番号,](x,y)[,color]

指定した座標に点を描画します

例）pset(100,100),0xffff0000

POINT [スクリーン番号,] (x,y)

指定した座標の色を返します

例）print point (100,100)

GPUT [スクリーン番号,] (x,y),グラフィック型変数

グラフィック型変数に記憶された画像を描画します

アルファチャンネルをサポートします

例）gput(100,100),g@

GGET [スクリーン番号,] (x1,y1)-(x2,y2)

指定した座標の画像データを取得します

アルファチャンネルをサポートします

例）g@=gget(100,100)-(300,300)

GLOAD “画像ファイル名”

bmp,png,jpg などの画像ファイルを読み込みます

アルファチャンネルをサポートします

例）g@=gload”data/sample.png”

GSAVE “画像ファイル名”,グラフィック型変数[,”画像形式”]

グラフィック変数の画像データを指定された画像形式でファイルに保存します

画像形式を省略した場合は png 形式で保存されます

画像形式は”B”:bmp、”P”:png、”J”:jpg です

例）gsave“data/sample.png”,g@,”P”

GRECT (x,y),(w,h)[,color]

指定した矩形を塗り潰します
座標と、幅,高さで指定します
他のグラフィックコマンドはアルファブレンドしながら描画しますが、GRECT はアルファチャンネルの値もそのまま描画します
部分的に透明でクリアしたい時などに使います
色を省略した場合は 0 で描画します
例）grect (100,100),(32,32)

ROLL[スクリーン番号] (x,y),

グラフィック画面をスクロールします
X が正の場合は右から左へ、X が負の場合は左から右へスクロールします
Y が正の場合は下から上へ、Y が負の場合は上から下へスクロールします
スクロールによってできた空白部分はクリアされます

修正履歴

- 2020-08-13　0.7.0.0　β版リリース
- 2020-08-15　0.7.5.0　実数計算時の不具合を修正
  算術関数(sin,cis,tan,asin,acis,atan,sqr,radian,degree)を追加
  キー入力関数(input,inkey$)を追加
  グラフィック関数で色を指定しない場合は白で描画する様に修正
  circle 関数に塗り潰し指定を追加
- 2020-08-17　0.7.6.0　実数型変数の不具合修正
  入力系コマンドの不具合修正
  renum コマンドで飛び先（goto 等）が変更されない不具合を修正
  KEY_STATE 処理を追加
  wait,swait コマンドを追加
  gload,gput コマンドを追加
  その他のバグ修正
- 2020-08-18　0.8.0.0　文字列操作系コマンド（chr$,left$,mid$,right$）を追加
  Line コマンドに矩形、塗り潰し指定を追加
  grect コマンドを追加
- 2020-08-19　0.8.1.0　for コマンドの変数が整数型変数しか使えなかったのを修正
  コマンドが小文字しか認識しなかったのを修正
  実数の記述で誤った表記をすると例外が発生していたのを修正
- 2020-08-19　0.8.2.0　dim コマンドの変数宣言を複数指定できる様に修正
- 2020-08-25　0.8.3.0　省略コマンド「?」「'」に対応
  文字列操作系、グラフィック系の未実装コマンドを実装
  「str$」「hex$」「string$」「asc」「val」「gget」「gsave」
- 2020-08-25　0.8.3.1　「'」によるコメントの不具合を修正
- 2020-09-07　0.8.5.0　細かなバグの修正
  16 進表記で桁が多すぎてオーバーフローするとフリーズしたり、LOCATE 命令で画面範囲外にかーソルを移動して PRINT 命令を実行するとフリーズするなど
- 2020-08-25　0.8.4.0　0.8.5.0 に致命的な不具合があったため、core だけ最新の物にした 0.8.4.0 をリース

- 2020-09-19　0.8.4.1　renum コマンド実行時にパラメータを省略するとフリーズするバグを修正
- 2020-09-20　0.8.4.2　文字列変数の配列を使用するとフリーズする不具合を修正
  DATA,READ,RESTORE コマンドに対応
- 2020-09-21　0.8.4.5　renum コマンドで飛び先などが更新されない事がある不具合を修正しました
  環境変数 TOUCH_INFO を追加
  ROLL コマンドを追加
- 2020-10-05　0.8.4.6　コアの内部処理を大幅に修正し細かなバグの修正
  変数の標準を実数に変更
  環境変数の『_』を『.』に変更
  ファイル操作系コマンド、算術演算系コマンドなどにいくつかのコマンドを追加

連絡先

E-Mail:info@thinksheep.com

URL:http://www.thinksheep.com

最後に

使用中に不具合が発生した場合、操作に関する不明点などのお問い合わせは上記連絡先にご連絡ください