プチコンBIG/3号 クイックリファレンス

ver 2016.12.22 / V3.5.5

定義内容

[内容=0]	省略可能、=記載がある場合は省略時の値
ABC	大文字の場合はその単語自身を指定する
値	数値 / 式を記述した場合、計算結果も値
式	数値 / 計算式が解釈され、計算結果が値となる
変数	参照型 / 変数に代入されてる値への参照、変数名を渡す必要がある
文字列変数$	参照型 / 文字を代入する変数。変数に $ を使用
整数値変数%	参照型 / 整数値(16bit, 65535) 変数に % を使用
配列変数	参照型 / DIM/VAR で宣言必須 / 命令に配列変数名を渡すことで配列全体を渡す
スロット	スロットを指定 / 0,1,2,3
行番号	[スロット:]行番号 / 1:123 でスロット1の123行目
frame	フレーム指定 / 1フレームは 1/60秒

キーボードショートカット

エディタ

矢印キー	カーソル移動
Shift + 矢印	範囲選択
BackSpace	カーソルの左1文字削除
Delete	カーソルの右1文字削除
PageUp	1ページ前にスクロール
PageDown	1ページ先にスクロール
Home	行頭移動
End	行末移動
Ctrl + Home	プログラム先頭へ移動
Ctrl + End	プログラム最後へ移動
Ctrl + Enter	1行挿入
Ctrl + Delete	カーソル右から行末を削除
F1	インラインヘルプ表示
F2	検索モード切替 / Ctrl + F
F3	
F4	
F5	スロット0のプログラム実行/停止
F6	
F7	編集中のスロットの保存 / ダイレクトモード:スロット0
F8	編集中のスロットの読込 / ダイレクトモード:スロット0
F9	TOP MENU
F10	スマイルボタン1実行
F11	スマイルボタン2実行
F12	画面モード切り替え
Alt + 0	ダイレクトモード
Alt + 1	スロット0 編集モード
Alt + 2	スロット1 編集モード
Alt + 3	スロット2 編集モード
Alt + 4	スロット3 編集モード
Ctrl + F1	BASIC ファンクションキー1
Ctrl + F2	BASIC ファンクションキー2
Ctrl + F3	BASIC ファンクションキー3
Ctrl + F4	BASIC ファンクションキー4
Ctrl + F5	BASIC ファンクションキー5
Ctrl + F	検索モード / F2
Ctrl + Z	undo
Shift + Ctrl + Z	redo
Ctrl + C	選択範囲コピー
Ctrl + X	cut
Ctrl + V	paste

インラインヘルプ

Alt + F1	インラインヘルプ内容更新
Alt + ←	前のページへ
Alt + →	次のページへ
Alt + PageUp	前の見出しへ
Alt + PageDown	次の見出しへ
Alt + C	ヘルプ内のサンプルプログラムをコピー
Alt + ↑	ヘルプを上へスクロール
Alt + ↓	ヘルプを下へスクロール

プログラム実行中

矢印キー	十字ボタン
Xキー	Aボタン
Bキー	Bボタン
Aキー	Xボタン
Sキー	Yボタン
Enter	Aボタン
Esc	Bボタン
Pause/Brake	プログラム停止
Ctrl + C	プログラム停止
Pad START 長押し	プログラム停止

TOP MENU

1 つくる
2 作品を見る
3 公開キーとサーバー管理
4 プロジェクトとファイル管理
5 サンプルを見る
6 追加機能の購入
7 WEBプチコン入門
8 オプション

FILE MENU

1 作業用のプロジェクトを変更
2 プロジェクトフォルダを作成
3 名前を変更
4 削除
5 コピー

NETWORK MENU

1 公開キーを使ってダウンロード(受信)
2 サーバーへアップロード(送信)
3 二次利用可能な作品を他の人に公開
4 サーバーからのダウンロード(受信)
5 公開とりけし
6 削除
7 受信拒否リストに追加
8 受信拒否リストを削除

基本命令

基本命令:BASIC

REM [文字]	コメント (過去互換)
'[文字]	コメント
KEY 番号, "文字列"	BASICファンクションキーの設定
OPTION [STRICT | DEFINT]	STRICT で変数宣言必須にする / DEFINITで変数宣言不要で新規変数を数値型とする
END	プログラム終了 / DEF 定義がある場合はDEF定義優先

基本命令:変数

変数=値	変数に数値や式の値を代入 / LETは使用不可
DIM 変数	変数宣言 OPTION STRICT指定時
VAR 変数	変数宣言 OPTION STRICT指定時
SWAP 交換元, 交換先	型一致の交換元と交換先の変数値を入れ替え。
INC 変数 [,加算値=1]	変数の値に加算値を加算
DEC 変数 [,減算値=1]	変数の値に減算値を減算
変数=CHKVAR("変数名")	変数存在確認 / 変数=1で存在、=0で未定義

基本命令:配列

要素数	配列を操作する数。宣言した配列は 0始まり の 要素数-1 終わり
DIM 配列変数[要素数]	配列宣言 配列要素は 0 始まり
VAR 配列変数[要素数]	配列宣言 配列要素は 0 始まり
COPY 配列dst [,dstOffset], 配列src [[,srcOffset],要素数]	元配列を先配列にコピー。1次元配列は自動拡張
COPY 配列dst [,dstOffset], "@ラベル" [,データ数]	DATA定義を先配列に読込。1次元配列は自動拡張。DATA不足はエラー
SORT [開始位置, 要素数,] 配列1 [,配列2. ..]	配列を昇順で並び替え
RSORT [開始位置, 要素数,] 配列1 [,配列2 ...]	配列を降順で並び替え
PUSH 配列, 値	配列の末尾に要素を追加 (末尾要素 追加)
変数=POP(配列)	配列の末尾から要素を取出 (末尾要素 削除)
UNSHIFT 配列, 値	配列の先頭に要素を追加 (先頭要素 追加)
変数=SHIFT(配列)	配列の先頭から要素を取出 (先頭要素 削除)
FILL 配列, 値 [,offset [,要素数]]	配列を値で埋める

基本命令:配列(高度処理) [DLC]

RINGCOPY 先配列, 先offset, 元配列 [[,元offset],要素数]	先配列をリングバッファとして配列コピー / PCMSTREAM に便利
次回先offset = RINGCOPY(先配列, 先offset, 元配列 [[,元offset],要素数])	先配列をリングバッファとして配列コピー / 戻値は先配列の次回使用の先offset
ARYOP 演算, 結果配列, p1, p2 [,p3]	結果配列要素分の演算 /演算: 0:加算(p1+p2),1:減算(p1−p2),2:乗算(p1*p2),3:除算(p1/p2), 4:積和(p1*p2+p3),5:線形補間(p1*p3+p2*(1−p3)),6:クランプ(p1を p2<=x<=p3 に丸め)

基本命令:制御

ラベル	[slot:]@ラベル文字 ラベル文字列。"1:@label" でスロット指定
@ラベル	ラベルの定義
GOTO @ラベル	ラベルへ移動
GOSUB @ラベル	ラベルへ遷移 / RETURNで復帰
RETURN	GOSUB遷移から復帰
ON 値 GOTO @ラベル0 [,@ラベル1 ...]	値でラベルへ移動 / 0 始まり整数値
ON 値 GOSUB @ラベル0 [,@ラベル1 ...]	値でラベルへ遷移 / RETURNで復帰 / 0 始まり整数値
変数=CHKLABEL("@ラベル" [,targ=0])	ラベル存在確認 / 変数=1で存在、=0で未定義 / targ=0でDEFのみ、=1で通常ラベル含む

基本命令:分岐

IF 値 THEN 処理1 [[ELSE 処理0] ENDIF]	条件分岐 / 値!=0なら処理1、値=0なら処理0 / ELSE以下は複数行記載可
ELSEIF 値 THEN 処理	条件分岐 / 値!=0で処理 / IF 値 THEN 〜 [ELSE 処理] ENDIF 間のみ記述可
IF 値 GOTO @ラベル [ELSE 処理]	値!=0で @ラベルへ移動 / ELSE 指定時に 値=0で 処理 / GOSUBは使えない

基本命令:繰り返し

FOR 変数=初期値 TO 終了値 [STEP 増分値=1]	FOR繰返開始 / 初回のみ変数に初期値を代入、変数>=終了値なら繰返終了
NEXT [変数]	FOR繰返終端 / FOR指定の変数に増分値を加算、FORへ戻る / FOR − NEXT で繰り返し
WHILE 値	WHILE繰返開始 / 値!=0 で WENDまで処理 / 値=0で繰返終了
WEND	WHILE繰返終端 / WHILEへ戻る / 通常は値にループ中で変化する式を記載
REPEAT	UNTIL繰返開始 / UNTILE戻り先 / ループを最低1回は処理する
UNTIL 値	UNTIL繰返終端 / 値=0 でREPEATへ戻る / 値!=0で次の処理へ
CONTINUE	FOR,WHILE,UNTILループ: 繰返継続 / 繰返開始へ移動 / FOR-NEXTでは増分値加算する
BREAK	FOR,WHILE,UNTILループ: 繰返終了 / 繰返終端の次へ移動

基本命令:関数定義

引数	関数への値渡し変数
変数	関数への参照渡し変数
関数名予約語	既存命令、既存システム変数、SPRITE, BG は定義不可
[COMMON] DEF 関数名 [引数1 [,引数2 ...]] [OUT 変数1 [,変数2 ...]]	関数定義 関数戻値なし / OUT指定時 変数1以降に返却
[COMMON] DEF 関数名([引数1 [,引数2 ...]]) [OUT 変数1 [,変数2 ...]]	関数定義 関数戻値あり / OUT指定可能?
RETURN [値]	DEF 定義から値返却で復帰 / OUT指定時はDEF内から変数名へ代入で返却
END	DEF定義終端 / DEF − END 間はローカルスコープ扱い
COMMON	DEF定義に指定でスロットを超えて使用可能として定義 / USE 実行必要
CALL "関数名" [値1 [,値2 ...]] [OUT 変数1 [,変数2 ...]]	関数呼び出し / DEF定義の関数名に一致する引数と変数の数を渡す
戻値=CALL("関数名" [値1 [,値2 ...]])	関数呼び出し / DEF定義の関数名に一致する引数と変数の数を渡す / OUTは使えない?
変数=CHKCALL("関数名")	呼出可能確認 / 変数=1で可、=0で不可

基本命令:ダイアログ

DIALOG "本文" [,[タイプ=0],["タイトル"=""],[時間=0]]	ダイアログ表示、オプションで詳細指定可能
戻値=DIALOG("本文" [,[タイプ=0],["タイトル"=""],[時間=0]])	ダイアログ表示、オプションで詳細指定可能 / 戻値にダイアログ操作結果を受け取り
本文、タイトル	256文字まで / 本文はCRLFで改行可能 / 漢字はUTF-16で指定
タイプ	0:[了解]/1:[いいえ][はい]/2:[戻る][次へ]/3:[中止][決定]/4:[中止][/実行]/5:[次へ] 負数:|b00|ABXYボタン(−1)|b01|十字キー(−2)|b02|LRボタン(−4)|b03|タッチパネル(−8)
時間	プラス値で秒数、マイナス値でフレーム
戻値 (RESULT変数)	−1: 否定(左) / 0: タイムアウト / 1: 肯定(右) / 128−137:ABXYudlrLR / 140:panel

基本命令:プチコン拡張

[機種] XON [MOTION | EXPAD | MIC | COMPAT | 3DS | WIIU]	拡張機能を有効にする / 一部は記述しているだけでsyntax error (BUG)
[機種] XOFF [MOTION | EXPAD | MIC | COMPAT | 3DS | WIIU]	拡張機能を無効にする / 一部は記述しているだけでsyntax error (BUG)
DLCOPEN "IP名1" [,"IP名2" ...]	有料ダウンロードコンテンツ使用許可指定

基本命令:データ操作

READ 変数1 [,変数2 ...]	DATA命令定義の値を読み込み / RESTRE で読み込みラベル指定
DATA 値1 [,値2 ...]	READ命令で読み込む値 / 計算式は記述できない
RESTORE @ラベル	指定したラベル位置以降のDATA命令をREAD命令が読み込むように指定する

基本命令:wait

WAIT [frame数=1]	frame数経過後まで処理停止 / 必ず停止 ※frame=1/60sec
VSYNC [frame数=1]	前回のVSYNC実行時からframe数経過するまで処理停止 / 停止しないこともある

ファイル

ファイル

種別	TXT:: テキスト, "DAT:":バイナリ, "//":プロジェクト一覧, "prjname/"
リソース	PRGn: プログラムSLOTn, GRPn: グラフィックページn, GRPF: フォント画像
FILES ["種別",] [文字列配列]	ファイル一覧取得 / 文字列配列 未指定時はコンソール出力
RENAME "[種別:]ファイル名", "[種別:]新ファイル名"	ファイル名変更 / ダイアログ非表示はできない
DELETE "[種別:]ファイル名"	ファイル削除 / ダイアログ非表示はできない
戻値= CHKFILE("[種別:]ファイル名")	ファイル存在確認 / 戻値: 1:存在, 0:存在しない

ファイル読込

LOAD "[リソース名:]ファイル名"[,ダイアログ]	ダイアログ: 0 で非表示 リソースにファイルを読み込み
LOAD "[GRPリソース名:]ファイル名" [OX,OY][,ダイアログ]	リソースにオフセットOX/OY指定で読み込み、オフセットはみ出た部分は無視
LOAD "TXT:ファイル名" [,ダイアログ] OUT TX$	テキストファイルをTX$に読み込み
TX$ = LOAD("TXT:ファイル名" [,ダイアログ])	テキストファイルをTX$に読み込み
LOAD "DAT:ファイル名", 数値配列 [,ダイアログ]	バイナリファイルを数値配列に読み込み

ファイル書込

SAVE "[リソース名:]ファイル名"	ダイアログ非表示はできない リソースをファイルに書き込み
SAVE "TXT:ファイル名", TX$	テキストファイルにTX$を保存
SAVE "DAT:ファイル名", 数値配列	バイナリファイルに数値配列を保存

プログラムファイル

EXEC "[リソース名:]ファイル名"	プログラムのLOADと実行 / 他のSLOTでEXECしたプログラムのENDで戻ることができる
EXEC プログラムSLOT	プログラムSLOTを実行 / 他のSLOTでEXECしたプログラムのENDで戻ることができる
USE プログラムSLOT	プログラムSLOTの実行許可を与える

入力

各種入力 / WiiU用は XON WIIU 指定必須

[3DS]	端末ID	3DS ワイヤレス時の端末ID / 3DSのみ / WiiU指定不可 / XON WIIU未指定時
[WiiU]	ctrlID	WiiU コントローラ / 0:pad, 1-4: リモコン系 / XON WIIU
	戻値=BUTTON([機能ID [,端末ID]])	ボタン入力: 共通
[WiiU]	戻値=BUTTON([機能ID [,ctrlID]])	ボタン入力: WiiU 機能ID: 0: key, 1:keyINrepeat, 2:keyIN, 3:keyOUT 戻値: 16bit: ss-lr-RLYXBArldu / ss はL/Rスティック押下 / lr は ZL,ZR
	BREPEAT ボタンID, [frame開始, frameINT=0]	キーリピート機能の設定 / 開始: 押してからリピート開始まで, INT: 間隔 (0:off)
	STICK [端末ID] OUT X,Y	左スティック入力: 共通
	STICKEX [端末ID] OUT X,Y	右スティック入力: 共通 / 3DSは XON EXPAD 必須
[WiiU]	STICK [ctrlID] OUT X,Y	左スティック入力: WiiU
[WiiU]	STICKEX [ctrlID] OUT X,Y	右スティック入力: WiiU 正: Y上 X右, 負 Y下 X左 / 0.86程度
	ACCEL OUT X,Y,Z	加速度 情報取得 / 共通 (3DS本体/Pad) / XON MOTION 必須
	GYROV OUT P,R,Y	角速度 情報取得 / 共通 (3DS本体/Pad) / XON MOTION 必須
	GYROA OUT P,R,Y	角度 情報取得　/ 共通 (3DS本体/Pad) / XON MOTION 必須
[WiiU]	ACCEL [ctrlID] OUT X,Y,Z	加速度 情報取得 / WiiU
[WiiU]	GYROV [ctrlID] OUT P,R,Y	角速度 情報取得 / WiiU
[WiiU]	GYROA [ctrlID] OUT P,R,Y	角度 情報取得　/ WiiU 加速度: (単位: G) / 重力方向に 1G P: Pitch(X軸の角速度)を取得する変数(単位:ラジアン/秒) R: Roll(Y軸の角速度)を取得する変数(単位:ラジアン/秒) Y: Yaw(Z軸の角速度)を取得する変数(単位:ラジアン/秒)
	TOUCH [端末ID] OUT TT,TX,TY	タッチ画面情報取得 / 共通 (3DS下画面/Pad) / 画面端 5 dot 読み取り不可
[WiiU]	TOUCH 座標変換 OUT TT,TX,TY	タッチPad 情報取得 / WiiU / 画面端 8 dot 読み取り不可 座標変換: 1:native(854x480), 0: XSCREEN指定解像度範囲に変換 TT　: タッチされた時間 (0=タッチなし) TX,TY: 共通:TX:5～314, TY:5～234 / WiiU: native or XSCREEN指定解像度範囲
[機種]	戻値=CONTROLLER([ctrlID])	コントローラの情報取得 / 共通(ID 0以外は戻値0) / 0:3DS/Pad, 1-4:WiiU(XON WIIU) 8bit: xxCNGprP: C:クラコン系, N:ヌンチャク, G:ジャイロ系, p:プロコン, r:リモ, P:Pad |b00| type / GamePad/3DS(1) |b01| type / Wiiリモコン(2) extra 拡張可能 |b02| type / プロコン(4) |b03| extra/ ジャイロ/リモコンプラス(8) |b04| extra/ ヌンチャク(16) |b05| extra/ クラコン/クラコンPRO(32)
[WiiU]	VIBRATE ctrlID, 振動, 時間	コントローラの振動 / WiiU / 振動: 弱0-100強 / 時間: 秒(小数指定可能)

入力:マイク

[機種]	MICSTART サンプリングレート, ビット数, 秒数	マイク録音開始 / 共通? / XON MIC 必須 / rate: 0: 8180Hz(32/16), 1: 10910Hz(24/12), 2: 16360Hz(16/8),3: 32730Hz(8/4) bit: 0: 8bit, 1: 16bit, 2: sig8bit, 3: sig16bit 秒数: 0:ループ(ring buff), 1-:録音秒数 / 最大は rate(8/16bit)で決定
[機種]	MICSTART [デバイス,] 秒数	マイク録音開始 / WiiU / XON WIIU 必須 / rate: 32kHz, bit: sgn16bit 固定 デバイス: 0:Pad, 1:USB / Pad,USB同時不可 秒数: 0:ループ(ring buff), 1-32: 秒数 / 最大は 32秒
[機種]	MICSTOP	マイク録音停止
[機種]	波形=MICDATA(取得位置)	マイクデータ取得 / 取得位置: サンプル数 / 波形: 8bit:128, 16bit:32768基準
[機種]	MICSAVE [[取得位置,] 取得数,] 配列変数	マイクデータ取得 / 取得位置,取得数: サンプル数(範囲外はエラー) / 配列は自動拡張

ワイヤレス通信 / 3DS限定

		スリープモードやふたを閉じたら切断される / セッション構築の確認はRESULT変数で取得
[3DS]	MPSTART 最大接続数, "通信識別子文字列"	ワイヤレス通信セッションの開始 / 確認ダイアログ表示 / 最大接続数 : 2-4
[3DS]	MPEND	ワイヤレス通信セッションを終了 / 確認ダイアログ表示
[3DS]	MPSEND "送信文字列"	セッション参加者全員へデータ送信 / 短時間に大量に呼ぶとエラー / 遅延は発生する
[3DS]	MPRECV OUT SID,RCV$	MPSENDからのデータ受信
[3DS]	変数 = MPSTAT([端末ID])	指定端末の接続状況を取得 / 0:未接続、1:接続
[3DS]	文字列変数 = MPNAME$(端末ID)	指定端末の端末名を取得
[3DS]	変数=MPGET(端末ID, 内部管理番号)	指定端末のユーザー定義データ取得 / 内部管理番号 0～8: 対象となるデータの管理番号
[3DS]	MPSET 内部管理番号, 数値	ユーザー定義データ書き込み / 0～8: 対象となるデータの管理番号 / 整数値のみ

スクリーン

スクリーン制御 / WiiU用は XON WIIU 指定必須 / 3D不可

	XSCREEN (0|1|2|3|4) [,SPRITE割当数 ,BG割当数]	画面モード指定 / 3DSの 上-下 は WiiUの TV-Pad / 上:400x240 TV:480?x240 下:320x240 0: 上-3D/下-未使用(起動時) 1: 上-2D/下-未使用 2: 上-3D/下-使用 (INPUT時はキーボード表示) 3: 上-2D/下-使用 (INPUT時はキーボード表示) 4: 上下結合(上画面は2D、INPUTおよびDIRECTモード使用不可) SPRITE割当数 : 上に割り当てる数(0-512) / 下は 512-上 の数 BG割当数 : 上に割り当てる数(0-4) / 下は 4-上 の数
[WiiU]	XSCREEN 5, TV解像度 [,SPRITE割当数 ,BG割当数]	画面モード指定(WiiU) / TVのみ指定 / 解像度は 0-6で指定
[WiiU]	XSCREEN 6, TV解像度, Pad解像度 [,SPRITE割当数 ,BG割当数]	画面モード指定(WiiU) / TV+Pad指定 / 解像度は 0-6で指定 0: 256x192 / 16: 16x12 / 8: 32x24 / 4:3 / ファミコンと同じ 1: 320x200 / 16: 20x12f/ 8: 40x25 / 8:5 2: 320x240 / 16: 20x15 / 8: 40x30 / 4:3 / 3DS下画面と同じ 3: 400x240 / 16: 25x15 / 8: 50x30 / 5:3 / 3DS上画面と同じ 4: 640x400 / 16: 40x25 / 8: 80x50 / 8:5 5: 640x480 / 16: 40x30 / 8: 80x60 / 4:3 6: 854x480 / 16: 53fx30/ 8:106fx60/16:9f/ 848=53*16 SPRITE割当数 : TVに割り当てる数(0-4096)/ Padは 4096-TV の数 BG割当数 : TVに割り当てる数(0-4) / Padは 4-TV の数
	DISPLAY 画面ID	操作対象画面を指定 / 下画面使用不可時は指定できない / DIRECTモード指定不可
	画面ID=DISPLAY()	操作対象画面を取得
	画面ID	0:上画面 / 1:下画面
	VISIBLE console, graph, bg, splite	画面の表示/非表示, 0/#OFF: 非表示、1/#ON: 表示
	BACKCOLOR 背景色	背景色を指定 / RGB888 指定
	背景色=BACKCOLOR()	背景色を取得
	ACLS [GRP系, SPDEF系, フォント系]	リソース定義を起動時にもどす / 1 で初期化しない、0で初期化 GRP系: GRP4,GRP5を起動時の定義に SPDEF系: SPDEF を起動時の定義に フォント系: フォント系定義を起動時の定義に
	FADE 前面色 [,frame]	フェード画面操作 / frame時間かけて前面色を指定 / 前面色:ARGB8888
	前面色=FADE()	フェード画面の前面色を取得 / 前面色:ARGB8888
	戻値=FADECHK()	フェード中か判定 / 1:フェード中、0:フェード済

コンソール

	コンソール入出力 / コンソール画面操作	共通: 上:400x240 = 50x30 / 下:320x240 = 40x30 / Z: 奥:1024 < 液晶面:0 < 手前:-256
	座標	画面解像度準拠 / WIIUは XSCREEN 参照
	CLS	画面を消去(DISPLAY命令で指定された画面)
	COLOR [文字色] [,背景色]	文字の色を指定 / どちらも省略は不可 / 色は0-15 / bit BGRA
	LOCATE [座標X],[座標Y] [,座標Z]	文字位置指定 / 省略時は維持 / すべて省略は不可
	PRINT [値 [;|, 値 ...]]	値(数値・文字列)の表示＋改行 / ; は結合、, は TABSTEP あとに結合
	ATTR 属性	PRINT文字の反転回転属性 / 属性bit: VHtt / V:垂直, H:水平, tt: deg0, 90, 180, 270
	SCROLL 文字数X, 文字数Y	画面のスクロール / 整数:右, 負数:左 / 指定座標がコンソール原点の感じ
	戻値=CHKCHR(座標X, 座標Y)	画面の文字コードを調べる / 戻値: UTF-16文字コード
	INPUT ["文字列"(;|,)] 文字列変数1 [,文字列変数2 ...]	ENTERまでキーボード入力待ち / ;|,で ?あり/なし / 複数変数指定は入力時に , で区切る
	LINPUT ["文字列"(;|,)] 文字列変数	ENTERまでキーボード入力待ち / , も入力可能
	文字列変数=INKEY$()	キーボードからの1文字(UTF-16)
	FONTDEF [文字コード, ("定義文字列"|定義数値配列)]	UTF-16フォント定義 / 文字コード省略で初期状態へ / フォントサイズは 8x8 定義は 16bit:RGBA=5551 / 文字列の場合は bit toHEX 256文字 / 配列は F%[64]
	WIDTH (8|16)	フォントサイズ設定 / 8:通常, 16: 通常の縦横2倍
	変数=WIDTH()	フォントサイズ取得
	色変換	
	変数 = RGB([透明度,] 赤要素,緑要素,青要素)	色変換 / A,R,G,B 8bit から ARGB=8888 / A: 0-254: 透明, 255:不透明 / R,G,B: 0-255
	RGBREAD 色コード OUT [A,] R,G,B	色変換 / ARGB=8888 から A,R,G,B 8bit /

GRP

	グラフィック	共通: 上:400x240 / 下:320x240 / Z: 奥:1024 < 液晶面:0 < 手前:-256 / XSCREEN解像度 色: ARGB=8888 / 特に指定がなければ色は RBG関数で変換したものを使用か&HFFFF直指定
	グラフィック 定義	
	GPAGE 表示ページ, 操作ページ	表示ページと操作ページの指定 / ページ: 0-5 / 4:SPRITE, 5:BG が初期でいる
	GPAGE OUT VP,WP	表示ページと操作ページの取得 / VP: 表示ページ, WP: 操作ページ
	GOFS X,Y	グラフィック表示のオフセット指定
	GOFS OUT X,Y	グラフィック表示のオフセット取得
	GCLIP モード [,始点X,始点Y,終点X, 終点Y]	クリッピング領域指定 / モード: 0:表示時,1:書込時
	GPRIO Z座標	表示順位変更 / Z座標: (奥:1024 < 液晶面:0 < 手前:-256)
	グラフィック 描画定義	
	GCOLOR 色	描画色の指定
	GCOLOR OUT 色C	描画色の取得
	変数 = GSPOIT(座標X, 座標Y)	座標の色を取得 / 変換のため描画時と不一致あり / メモリは RGBA=5551
	グラフィック 描画	
	GCLS [色]	画面消去 / 色: ARGB=8888
	GPSET 座標X, 座標Y [,色]	座標に点を打つ
	GLINE 始点X, 始点Y, 終点X, 終点Y [,色]	始点から終点に線を引く
	GBOX 始点X, 始点Y, 終点X, 終点Y [,色]	始点から終点に四角形を描く
	GFILL 始点X, 始点Y, 終点X, 終点Y [,色]	始点から終点に塗りつぶし四角形を描く
	GCIRCLE 中心点X, 中心点Y, 半径, [,色| ,開始角,終了角 [,扇 [,色]]]	中心点から半径の円を描く / 開始角,終了角: 単位:角度(0-360) / 扇: 0=円弧, 1=扇形
	GTRI X1,Y1, X2,Y2, X3,Y3 [,色]	塗りつぶし三角形を描く
	GPAINT 開始点X, 開始点Y [,色 [, 境界色]]	開始点から周辺を塗りつぶす / 境界色指定時は境界色以外の周辺を塗りつぶす
	GPUTCHR X,Y, ("文字列"|文字コード) [,倍率X, 倍率Y] [,色]	グラフィック領域に文字を描画
[WiiU]	GPUTCHR16 X,Y, ("文字列"|文字コード) [,倍率X, 倍率Y] [,色]	グラフィック領域に文字を描画 / WiiU / 16dot font
	グラフィック 転送	
	GCOPY [転送元,] 始点X, 始点Y, 終点X, 終点Y, 転送先X,転送先Y, 透明	GRPnを操作ページにコピー / 転送元: 0-5: GRPn, -1: GRPF / 透明: 0/1:透明色無視/上書
	GSAVE [転送元,] [X,Y,幅,高,] 転送先配列, 色変換	GRPnを配列へ保存 / 配列は自動拡張 / 色変換: 0: ARGB=8888, 1: RGBA=5551
	GLOAD [X,Y,幅,高さ,] 画像配列,[色変換|色配列],透明	操作ページに画像配列を展開 / 色配列: 画像配列を配列要素にした色 /

SPRITE

スプライト	共通: 上:400x240 / 下:320x240 / Z: 奥:1024 < 液晶面:0 < 手前:-256 / XSCREEN解像度
スプライト初期設定	
SPPAGE グラフィックページ	SPRITEに割り当てるグラフィックページの設定 / 初期値 4
変数=SPPAGE()	SPRITEに割り当てるグラフィックページの取得
SPCLIP [始点X, 始点Y, 終点X, 終点Y]	クリッピング領域を指定
スプライト: 定義	
SPDEF	初期化
SPDEF 定義番号, U,V [,W,H [,原点X,原点Y]] [,attr]	個別定義
SPDEF 数値配列 [,定義番号offset [,U_offset ,V_offset]]	配列から定義
SPDEF "@ラベル文字列" [,定義番号offset [,U_offset ,V_offset]]	DATA命令から定義
SPDEF 定義番号 OUT U,V [,W,H [,HX,HY]] [,A]	定義取得
SPDEF 定義番号,元定義番号,[U],[V],[W],[H],[原点X],[原点Y],[attr]	定義からコピー / 省略でコピーせず
スプライト: 作成	作成前に管理番号を操作するとエラー
SPSET 管理番号,定義番号	定義番号で作成
SPSET 管理番号 ,U,V [,W,H] ,attr	直接指定で作成
SPSET 定義番号 OUT IX	空きを探して定義番号で作成 / 無ければ IX=-1
SPSET U,V,W,H,attr OUT IX	空きを探して直接指定で作成 / 無ければ IX=-1
SPSET 上限,下限, 定義番号 OUT IX	範囲内で空きを探して定義番号で作成 / 無ければ IX=-1
SPSET 上限,下限, U,V,W,H,アトリビュート OUT IX	範囲内で空きを探して直接指定で作成 / 無ければ IX=-1
変数=SPUSED(管理番号)	指定されたSPRITEが使われているか調査
SPCLR 管理番号	スプライト解放
スプライト: 設定	
SPSHOW 管理番号	スプライト表示
SPHIDE 管理番号	スプライト非表示 / 見えないだけで存在はしている
SPHOME 管理番号,位置X,位置Y	座標基準点指定
SPHOME 管理番号 OUT HX,HY	座標基準点(ホーム位置)取得
SPOFS 管理番号, X, Y [,Z]	座標の変更(移動)
SPOFS 管理番号 OUT X,Y[,Z]	座標を取得
SPROT 管理番号,角度	回転角度指定
SPROT 管理番号 OUT DR	回転角度取得
変数=SPROT(管理番号)	回転角度取得
SPSCALE 管理番号, 倍率X, 倍率Y	スケール(表示倍率)の変更
SPSCALE 管理番号 OUT SX,SY	スケール(表示倍率)の取得
SPCOLOR 管理番号, 色	表示色を設定 / 色はスプライトに乗算
SPCOLOR 管理番号 OUT C32	表示色を取得
SPCHR 管理番号, 定義番号	定義を変更(テンプレート指定)
SPCHR 管理番号,[U],[V],[W],[H],[アトリビュート]	定義を変更(直接定義)
SPCHR 管理番号 OUT U,V [,W,H [,A]]	定義情報を得る
SPCHR 管理番号 OUT DEFNO	定義番号を得る
SPLINK 管理番号, リンク先管理番号	別のSPRITEにリンク / リンク先の相対座標となる / リンク先指定は小さい管理番号
変数=SPLINK(管理番号)	リンク先番号を得る
SPUNLINK 管理番号	リンクを解除
スプライト: アニメ	
SPANIM 管理番号,"アニメ対象",データ配列 [,ループ]	アニメ表示(配列で指定)
SPANIM 管理番号,"アニメ対象","@ラベル文字列" [,ループ]	DATA指定
SPANIM 管理番号,"アニメ対象",時間1,項目1[,項目2] [,時間2,項目1[,項目2]]…	直接指定
SPSTART [管理番号]	アニメーション開始
SPSTOP [管理番号]	アニメーション停止
変数=SPCHK(管理番号)	アニメーション状態を取得
スプライト: 変数	
SPVAR 管理番号,内部変数番号,数値	内部変数への書き込み / 内部変数番号: 0-7
変数=SPVAR(管理番号,内部変数番号)	内部変数の読み込み
SPVAR 管理番号,内部変数番号 OUT V	内部変数の読み込み
スプライト: 衝突判定	
SPCOL 管理番号 [,スケール対応]	衝突判定情報の設定
SPCOL 管理番号,[スケール対応],マスク	衝突判定情報の設定(マスク指定付) / マスク 32bit値で同じなら衝突判定
SPCOL 管理番号,始点X,始点Y,幅,高さ,[スケール対応],マスク	衝突判定情報の設定(範囲指定付)
SPCOL 管理番号 OUT スケール対応 [,マスク]	衝突判定情報取得(スケール対応とマスク)
SPCOL 管理番号 OUT 始点X,始点Y,幅,高さ	衝突判定情報取得(範囲)
SPCOL 管理番号 OUT 始点X,始点Y,幅,高さ,スケール対応	衝突判定情報の取得(範囲とスケール対応)
SPCOL 管理番号 OUT 始点X,始点Y,幅,高さ,スケール対応,マスク	衝突判定情報の取得(すべて)
SPCOLVEC 管理番号 [,移動量X,移動量Y]	衝突判定用移動速度の設定
変数 = SPHITSP(管理番号 [,先頭ID,末尾ID])	衝突判定 / SPCOLを呼び出しておくこと
変数 = SPHITSP(管理番号 ,相手管理番号)	衝突判定
変数 = SPHITSP()	直前に設定した情報からSPRITEの衝突判定
SPHITRC(始点X,始点Y,幅,高さ[,[マスク],移動量X,移動量Y])	動く四角形とすべてのSPRITEの衝突判定
SPHITRC(管理番号,始点X,始点Y,幅,高さ[,[マスク],移動量X,移動量Y])	指定したSPRITEと四角形の衝突判定
SPHITRC(先頭ID,末尾ID, 始点x,始点y,幅,高さ[,[マスク],移動量X, 移動量Y])	指定範囲のSPRITEと四角形の衝突判定
変数 = SPHITRC()	直前に設定した情報からSPRITEの衝突判定
SPHITINFO OUT TM	衝突判定結果の情報取得(衝突時間)
SPHITINFO OUT TM,X1,Y1,X2,Y2	衝突判定結果の情報取得(衝突時間と座標)
SPHITINFO OUT TM,X1,Y1,VX1,VY1,X2,Y2,VX2,VY2	衝突判定結果の情報取得(衝突時間と座標と速度)
スプライト: 拡張	
SPFUNC 管理番号, (@ラベル|DEF定義関数)	SPRITEごとに処理を割り当て / 遷移先は CALLIDX で 管理番号取得
CALL SPLITE	SPFUNC で指定した関数を呼び出し

BG

BG	16x16タイルを扱う 共通: 上:400x240 / 下:320x240 / Z: 奥:1024 < 液晶面:0 < 手前:-256 / XSCREEN解像度 レイヤ: 0-3
BG初期設定	
BGPAGE グラフィックページ	BGに割り当てるグラフィックページの設定 / 初期値 5
変数=BGPAGE()	BGに割り当てるグラフィックページの取得
BGSCREEN レイヤ,幅,高,[charsize?=16(8|16|32)]	画面サイズ設定 / BGが何個置けるかの定義 / 幅x高<16383 / 初期 25x15
BG:表示	
BGCLR [レイヤ]	BGスクリーンを消去 / 省略時はすべてのレイヤ
BGSHOW レイヤ	BGスクリーンを表示
BGHIDE レイヤ	BGスクリーンを非表示
BGCLIP レイヤ [,始点X,始点Y,終点X,終点Y]	BGスクリーンの表示領域を指定 / 始点終点省略でレイヤ全体
BG:移動	
BGHOME レイヤ,位置X,位置Y	表示原点設定 / BGスクリーンに対する回転や拡大縮小の原点
BGHOME レイヤ OUT HX,HY	表示原点取得
BGOFS レイヤ,X,Y,[Z]	BGスクリーンの表示オフセットを変更
BGOFS レイヤ OUT X,Y[,Z]	座標を得る
BGROT レイヤ,角度	BGスクリーンの回転 / 角度は 0-360
BGROT レイヤ OUT R	BGスクリーンの回転情報取得
BGSCALE レイヤ,拡大率X,拡大率Y	BGスクリーンの拡大縮小 / 全体で3600個分まで表示 / 拡大率 0.5 - 1.0 - 2.0
BGSCALE レイヤ OUT SX,SY	BGスクリーンの拡大縮小情報取得
BG:キャラ設定	
スクリーンデータ	bit16: VHrrcccccccccccc : V:H: 縦横反転, rr: 回転, c^12: キャラ番号(0-4095:1024周期)
CX, CY	BGキャラ単位の座標
BGPUT レイヤ,CX,CY,スクリーンデータ	BGスクリーンへのBGキャラ配置 / X,Y: BGSCREEN座標
BGFILL レイヤ,始点CX,始点CY,終点CX,終点CY,スクリーンデータ	BGスクリーンをBGキャラで塗りつぶし
変数=BGGET(レイヤ, X, Y [,座標系フラグ=0])	BGスクリーンのBGキャラ情報を得る / 座標系フラグ: 0: BGキャラ単位, 1:画面dot単位
BG:アニメ	
BGANIM レイヤ, "アニメ対象",データ配列 [,ループ]	アニメ表示(配列で指定)
BGANIM レイヤ, "アニメ対象","@ラベル文字列" [,ループ]	DATA指定
BGANIM レイヤ, "アニメ対象",時間1,項目1[,項目2] [,時間2,項目1[,項目2]]… [,ノ	直接指定
BGSTART [レイヤ]	アニメ開始
BGSTOP [レイヤ]	アニメ停止
変数=BGCHK(レイヤ)	アニメ状態を取得
BG:変数	
BGVAR レイヤ, 内部変数番号,数値	内部変数への書き込み / 内部変数番号: 0-7
変数=BGVAR(レイヤ, 内部変数番号)	内部変数の読み込み
BGVAR レイヤ, 内部変数番号 OUT V	内部変数の読み込み
BG:操作	
BGCOPY レイヤ,始点CX,始点CY, 終点CX,終点CY, 転送先CX,転送先CY	BGスクリーンをキャラ単位でコピー
BGLOAD レイヤ, [始点CX,始点CY,幅,高さ,] 数値配列 [,BG番号offset]	配列からBGデータをBGスクリーンにコピー / 数値配列: BGSAVE結果
BGSAVE レイヤ, [始点CX,始点CY,幅,高さ,] 数値配列	BGスクリーンの内容を数値配列へコピー / 数値配列は自動拡張
BGCOORD レイヤ,元座標X,元座標Y [,モード=0?] OUT DX,DY	座標変換 / モード: 0: BGchr?to画面, 1: 画面toBGchr, 2: 画面toBGdot
BGCOLOR レイヤ, 色	表示色を設定 / 色:ARGB=8888 α無効 / 色は乗算
BGCOLOR レイヤ OUT C32	表示色を得る
BG:拡張	
BGFUNC レイヤ, @ラベル	レイヤごとに処理を割り当て / 遷移先は CALLIDX で レイヤ取得
CALL SPLITE	BGFUNC で指定した関数を呼び出し

サウンド

サウンド

```
                                                    BIG出力先 8bit: 4321xxPT : P:Pad, T:TV, 1-4:リモコン / WiiU
                                                    音量: 0-127 / パン:左0-64-127右
                                                    トラック: 0-7 / 曲番号: 1-127: プリセット(変更不可), 128-255:ユーザ定義
```

サウンド 全般

```
    SNDSTOP                                         すべての音を停止 / BGM, BEEP, TALK, EFFECT, PCM
```

サウンド: BEEP

```
    BEEP [効果音番号] [,周波数] [,音量] [,パン]          音を鳴らす / 周波数: -32768 - 32767(100で半音)
[WiiU]  BEEP [効果音番号] [,周波数] [,音量] [,パン] [,BIG出力先]  音を鳴らす / WIIU / 出力先 8bit: 4321xxPT : P:Pad, T:TV, 1-4:リモコン
```

サウンド: 音声合成

```
    TALK "音声文字列" [,BIG出力先]                     音声合成による発声 / 特殊文字: 速さ: <S32768> / 高さ: <P32768> / 0-65535
    変数=TALKCHK()                                  音声合成の状態調査
    TALKSTOP                                        再生中の音声を停止
```

サウンド: BMG

```
    BGMPLAY [トラック番号,] 曲番号 [,音量][,BIG出力先]     BGM再生 /
    BGMPLAY "MML文字列" [,BIG出力先]]                  BGM再生 / 曲番号255 トラック0 に上書き定義して再生

    BGMSTOP [トラック番号 [,フェード時間]]               音楽演奏停止
    BGMSTOP -1                                      音楽演奏停止 / 強制的な停止 / 曲番号255 を上書き

    戻値=BGMCHK( [トラック=0] )                        演奏状態調査 / 戻値: 1:演奏中, 0:停止中
    BGMVOL [トラック番号=0,] 音量                      ボリューム

    BGMPAUSE [トラック番号 [,フェード時間=0]]           一時停止 / トラック省略は全トラック
    変数=BGMPAUSE( [トラック番号=0] )                  一時停止状態の確認
    BGMCONT [トラック番号 [,フェード時間=0]]            一時停止中の音楽演奏を再開 / トラック省略は全トラック
```

サウンド: BMG 定義

```
    BGMSET 曲番号,"MML文字列"                         曲定義
    BGMSETD 曲番号,"@ラベル文字列"                     曲定義 / DATAから定義 / DATA 0 で終端 / RESTORE扱いのため以後READはRESTORE再定義必要
    BGMCLEAR [曲番号]                                ユーザー定義音楽の消去 / 曲番号: 128-255

    WAVSET 定義番号,A,D,S,R,"波形文字列" [,基準音程=69(O4A)]  楽器音を定義 / 定義: 224-255 / ADSR: 0-127 / 波形文字: 16進 2文字1サンプル 8bit
                                                    16,32,64,128,256,512サンプルの指定が可能 / 文字数はサンプル数の2倍
    WAVSETA 定義番号,A,D,S,R,数値配列 [,基準音程][,先頭添字=0][,最終添字=L]  楽器音を定義 / 数値配列 MICSAVE結果 <=16384 / 8180Hz 8bit固定

    BGMVAR トラック番号, 内部変数番号, 値               内部変数への書き込み / 内部変数番号: 0-7 / MML の $0 - $7
    変数=BGMVAR(トラック番号, 内部変数番号 )            内部変数の読み込み
    BGMVAR トラック番号, 内部変数番号 OUT V             内部変数の読み込み / 存在しない？
```

サウンド:エフェクト

```
    EFCON                                           エフェクトON
    EFCOFF                                          エフェクトOFF
    EFCWET BEEP効果値, BGM効果値, TALK効果値            エフェクト量を設定 / BEEP,BGM: 0-127, TALK: 0-63:OFF, 64-:ONM
    EFCSET [種類 | InitRef,Delay,Decay,Filter1,Filter2,InitGain,Gain]  エフェクト設定 / 省略不可 / 種類: 0: 無し, 1-3:リバーブ, 1:風呂場, 2:洞窟, 3:宇宙
                                                    InitRef: 初期反射時間      : 0 - 2000 (msec)
                                                    Delay:   残響音ディレイ時間 : 0 - 2000 (msec)
                                                    Decay:   残響音減衰時間    : 0 - 1000 (msec)
                                                    Filter1: 残響音フィルタ係数1: 0.0 - 1.0
                                                    Filter2: 残響音フィルタ係数2: 0.0 - 1.0
                                                    InitGain:初期反射音ゲイン   : 0.0 - 1.0
```

サウンド:PCM [DLC]

```
                                                    サンプリングデータの配列を再生、RINGCOPYで配列を操作するとよい
    PCMSTREAM [M配列 | L配列,R配列] [,サンプリングレート=32730][,BIG出力先]  PCMループ再生 / 配列は sgn16bit: -32768 - 32767 / Mは2次元でLR / DIRECTモード不可
    PCMSTREAM [サンプリングレート=32730][,BIG出力先]      再生中のPCMSTREAMのサンプリングレートを変更 / 再生していない時は変化なし /1-192000
    PCMPOS=pos                                      PCMSTREAMの再生位置を設定 / PCMSTREAM指定配列要素のMOD値が適用
    PCMCONT                                         PCMSTREAM停止時の状態から再開
    PCMSTOP                                         PCMSTREAMを停止
    PCMVOL [CH,] VOL                                PCMSTREAMの音量を設定 / CH:0=左,1=右 省略で両方 / VOL:-32767 - 32767 負は位相が逆

    RINGCOPY 先配列, 先offset, 元配列 [[,元offset],要素数]  先配列をリングバッファとして配列コピー
    次回先offset = RINGCOPY( 先配列, 先offset, 元配列 [[,元offset],要素数] )  先配列をリングバッファとして配列コピー / 戻値は先配列の次回使用の先offset
    ARYOP 演算, 結果配列, p1, p2 [,p3]                 結果配列要素分の演算 /演算: 0:加算(p1+p2),1:減算(p1-p2),2:乗算(p1*p2),3:除算(p1/p2),
                                                    4:積和(p1*p2+p3),5:線形補間(p1*p3+p2*(1-p3)),6:クランプ(p1を p2<=x<=p3 に丸め)
```

サウンド: 高度サウンドユニット [DLC]

```
                                                    詳細はリファレンス参照
    BIQUAD OTWK,INWK,FP                             BiQuadフィルタ / 出力配列, 入力配列,フィルタ配列 / OTWK<INWK要素数でエラー
    BQPARAM FP,k,s,f,q(o) [,g]                      フィルタ係数を計算 / 出力配列, フィルタ種別, samplrate,cutoff,Q,octave,gain
    FFT oR,oI,iR,iI[,W]                             複素数配列に対してFFT / 出力実数,出力虚数,複素数配列実数,複素数配列虚数,窓関数値配列
    IFFT oR,oI,iR,iI                                複素数配列に対して逆FFT /
    FFTWFN W,n                                      配列 W に窓関数値を返す / n: 0:矩形窓, 1:ハミング窓, 2:ハニング窓, 3:ブラックマン窓
```

MML文字列コマンド

```
    :0 - :15                                        チャンネル指定
    T1 - T512                                       テンポ指定 / BPM
    1 - 192                                         音長の個別指定(C1=全音符、C4.=付点四分音符) / BPM の全音符=1,四分音符=4 / 逆数指定
    L1 - L192                                       音符 規定値音長 / 長さ指定しない場合の音の長さ
    CDEFGAB                                         音符 音階(C#で半音上がり、C-で半音下がる)
    R                                               音符 休符
    C1 C2 C4 C8 C16 C32 C1. C2. C4. C8. C16. C32. C12 C24  音符の長さ指定 / BPM全音符の逆数指定 / 付点は.を付ける(音の長さの半分を加算)
    & _                                             前後の音をつなぐ、ポルタメント
    Q0 - Q8                                         発音時間の割合（ゲート）
    O0 - O8 < > !                                   オクターブ指定、1オクターブ上げる、下げる、オクターブ上げ下げ反転(<>の扱い逆転)
    N0 - N127                                       キーを数値指定(O4C=60)
    V0 - V127 ( )                                   音量の数値指定、音量上げる、下げる
    P0 - P127                                       パンポット(左 P0 - P64 - P127 右)
    @E127,100,30,100                                エンベロープ @E(Attack time),(Decay time),(Sustain level),(Release time) : 0-127
    @0 - @255                                       音色変更 0-127:GM, 128-129:ドラム, 144-150:PSG, 151:ノイズ, 224-255 WAVSET, 256-BEEP
    @MON @MOFF                                      モジュレーション 開始/停止
    @D-128 @D127                                    デチューン(周波数の微調整)設定 / -128で1音低く、+127で1音高くなる
    @MA64,1,16,32  @MP64,1,16,32  @ML100,1,8,0      トレモロ/ビブラート/オートパン / 同時使用不可 / Depth, Range, Speed, Delay: 0-127
    [ ]回数                                         リピート開始/終了 /回数省略時は無限ループ
    $0 - $7  $0=数値 - $7=数値  / 例 $0=64 V$0      MML内部変数 / 数値は 0-255 / BGMVAR トラック, 内部変数, 値 で設定
    {ラベル名=MML} {ラベル名}                        マクロ定義と利用 / チャンネル指定は禁止 / ラベル名は8文字までの英数字
```

汎用命令

数学

変数 = FLOOR(数値)	整数部を取り出す / 小数部切り捨　/ その数を超えない最大の整数
変数 = ROUND(数値)	整数部を取り出す / 小数部四捨五入 /
変数 = CEIL(数値)	整数部を取り出す / 小数部切り上げ / その数を下回らない最小の整数を得る
変数 = ABS(数値)	絶対値
変数 = SGN(数値)	符号取得 / −1, 0, 1
変数 = MIN(数値配列)	指定された数値配列内の一番小さい値を得る
変数 = MAX(数値配列)	指定された数値配列内の一番大きい値を得る
変数 = MIN(数値1 [,数値2 …])	指定された複数の数値から一番小さい値を得る
変数 = MAX(数値1 [,数値2 …])	指定された複数の数値から一番大きい値を得る
変数 = RND([シードID,] 最大値)	整数の乱数を得る(0〜最大値−1まで) / シードID: 乱数の系列: 0〜7
変数 = RNDF([シードID])	実数型の乱数を得る(0以上 1.0未満の実数乱数) / シードID: 乱数の系列: 0〜7
RANDOMIZE シードID [, シード値]	乱数系列の初期化
変数 = SQR(数値)	正の平方根を求める
変数 = EXP([数値])	e(自然対数の底)のべき乗を求める / 省略時eを返す
変数 = LOG(数値 [,底])	対数を求める / 省略時、自然対数を求める
変数 = POW(数値, 乗数)	べき乗を求める
変数 = PI()	円周率を得る
ラジアン = RAD(角度)	度からラジアンを求める
角度 = DEG(ラジアン)	ラジアンから度を求める
三角比 = SIN(ラジアン)	サイン値を返す
三角比 = COS(ラジアン)	コサイン値を返す
三角比 = TAN(ラジアン)	タンジェント値を返す
ラジアン = ASIN(三角比)	アークサイン値を返す
ラジアン = ACOS(三角比)	アークコサイン値を返す
ラジアン = ATAN(三角比)	アークタンジェント値を返す(数値から)
変数 = ATAN(座標Y,座標X)	アークタンジェント値を返す(XY座標から) / 原点からY,Xの値
変数 = SINH(数値)	ハイパボリックサイン値を返す / 双曲線関数
変数 = COSH(数値)	ハイパボリックコサイン値を返す / 双曲線関数
変数 = TANH(数値)	ハイパボリックタンジェント値を返す / 双曲線関数
戻値 = CLASSIFY(数値)	通常数値、無限大、非数（NaN）の判定 / 0=通常数値、1=無限大、2=NaN

文字列操作

数値 = ASC("文字")	文字コード取得 / 文字コード: UTF-16
文字 = CHR$(文字コード)	文字コードから文字を返す / 文字コード: UTF-16
数値 = VAL("文字列")	文字列から解釈された数値を得る
文字 = STR$(数値 [,桁数])	数値から文字列　/ 桁数指定で先頭に空白を埋める / 桁数が超える場合は無視
文字 = HEX$(数値 [,桁数])	数値から16進文字列 / 桁数指定で先頭 0 埋め
文字 = BIN$(数値 [,桁数])	数値から2進文字列 / 桁数指定で先頭 0 埋め
文字 = FORMAT$("書式文字列", 値 [, ...])	書式成形文字 / %S: 文字, %D: 10進, %X: 16進, %F: 実数, %B: 2進 / printf()と同等
数値 = LEN(["文字列" | 配列変数])	文字の長さ, 配列の長さ
文字 = MID$("文字列", 開始位置, 文字数)	文字取り出し / 位置は 0 スタート
文字 = LEFT$("文字列", 文字数)	左端から文字取り出し
文字 = RIGHT$("文字列", 文字数)	右端から文字取り出し
文字 = INSTR([開始位置=0,] "文字列", "検索文字列")	文字列から検索文字列を検索 / 位置は 0 スタート
文字 = SUBST$("文字列", 開始位置, [文字数,] "置換文字列")	文字列の開始位置から文字数分を削って置換文字列に置き換える
TMREAD ["時間文字列"=現在時刻] OUT 変数h,変数m,変数s	時間文字列("HH:MM:SS"形式)を数値に変換し、変数h,m,sへ代入
DTREAD ["日付文字列"=現在月日] OUT 変数y,変数m,変数d,[変数w]	日付文字列("YYYY/MM/DD"形式)を数値に変換し、変数y,m,d,wへ代入 / wは日曜=0の曜日

ビット演算

変数 = 数値1 MOD 数値2	割り算の答のあまり / 0で割るとエラー
変数 = 数値1 DIV 数値2	割り算の答の整数値 / 0で割るとエラー
変数 = 数値1 AND 数値2	論理積 / 各ビットごとで一致したものの結果
変数 = 数値1 OR 数値2	論理和 / 各ビットごとで存在するものの結果
変数 = 数値1 XOR 数値2	排他的論理和 / 各ビットごとで同じなら0で違うなら1の結果
変数 = NOT 数値	ビット反転 / 各ビットごとで0なら1で1なら0の結果
変数 = 数値 << シフト	ビット左シフト / 2のシフト乗で掛け算 / 右側は 0埋め、超えた左側は削除
変数 = 数値 >> シフト	ビット右シフト / 2のシフト乗で割り算 / 左側は 0埋め、超えた右側は削除

エディタ関連

ソースコード操作

PRGEDIT プログラムSLOT [,行番号=1]	操作するSLOTと行を指定 / 行番号=−1 で最終行 / カレントSLOTはエラー
文字 = PRGGET$()	操作行を取得 / 範囲外なら空文字
PRGSET "文字列"	操作行を置換 / 範囲外なら行追加
PRGINS "文字列" [,フラグ=0]	操作行に挿入 / フラグ: 0:操作行の前, 1:操作行の後 / CRLFで複数行
PRGDEL [削除行数=1]	操作行を削除 / 削除行数がマイナス値で全体削除
数値 = PRGSIZE([プログラムSLOT [,タイプ=0]])	行数取得 / タイプ: 0:行数, 1:文字数, 2:空き文字数
文字 =PRGNAME$([プログラムSLOT])	SAVE/LOADでのファイル名取得 / 省略時は実行中/直前実行のスロット
CLIPBOARD "文字列"	クリップボード書き込み / 範囲選択しCtrl+C 結果？
戻値=CLIPBOARD()	クリップボード読み込み

DIRECTモード

DIRECTモード専用

CLEAR	BASIC内部のメモリーを初期化 / slot 指定でスロットを初期化
NEW [slot]	プログラムを消去
LIST [行番号 | ERR]	EDITモードへの切り替えと編集開始
RUN [slot]	プログラムの実行 / slot指定スロットを実行
CONT	停止中のプログラムを再開
PROJECT "プロジェクト名"	デフォルトプロジェクトの切り替え
BACKTRACE	直前の呼び出し元の履歴表示

DIRECTモード

PROJECT OUT PJ$	デフォルトプロジェクトの取得