DIRECTモードでのみ利用できる命令

DIRECTボタンを押した状態でのみ利用できる命令
※EDITモードでプログラムに書くことはできません

CLEAR	BASIC内部のメモリーを初期化 DIRECTモード専用
書式	CLEAR
例	CLEAR

NEW	プログラムを消去 DIRECTモード専用	
書式	NEW [プログラムSLOT]	
引数	プログラムSLOT	0～3:指定SLOTのみ消去(省略時:全SLOT消去)
例	NEW NEW 3	

LIST	EDITモードへの切り替えと編集開始 ・DIRECTモード専用 ・引数のないLISTは、EDITボタンを押すことと同じ働き	
書式	LIST [行番号/ERR]	
引数	行番号	・省略するとデフォルト行から表示 ・2:120のような書き方でプログラムSLOTを指定
	ERR	直前にエラーの発生した行を指定
例	LIST ERR LIST 1:	

RUN	プログラムの実行 DIRECTモード専用	
書式	RUN [プログラムSLOT]	
引数	プログラムSLOT	実行する指定プログラムSLOT(0～3、省略時=0)
例	RUN RUN 1	

CONT	停止中のプログラムを再開 ・DIRECTモード専用 ・STARTボタン、STOP、エラー等の中断位置から実行再開 ・プログラム編集を行うと再開できない ・入力待ちの途中で中断したときは再開できない ・エラー種類によっても再開できない
書式	CONT
例	CONT

PROJECT (1)	デフォルトプロジェクトの切り替え DIRECTモード専用	
書式	PROJECT "プロジェクト名"	
引数	プロジェクト名	変更するプロジェクト名文字列 ・新規作成はTOP MENUから行う ・プロジェクト名を "" にするとDEFAULT扱いとなる
補足	カレントプロジェクト 3つの状態	1) 起動時のカレントプロジェクト(OPTIONで設定) 2) 非実行時のカレントプロジェクト(PROJECT命令で設定) 3) 実行時のカレントプロジェクト(EXEC等の実行時に設定) それぞれの設定時には下位のプロジェクト設定も更新されます。 例えば、OPTION設定でカレントプロジェクトを変更した場合、非実行時と実行時のプロジェクトも同時に変化します。 また、実行開始(RUN、ツール実行、VIEWERから実行)時には、実行時カレントプロジェクトは非実行時カレントプロジェクトが初期値として設定されます。
例	PROJECT ""	

PROJECT (2)	デフォルトプロジェクトの取得 プログラムの中からも利用可能	
書式	PROJECT OUT PJ$	
引数	なし	
戻り	PJ$	カレントのプロジェクト名
例	PROJECT OUT PJ$	

BACKTRACE	直前の呼び出し元の履歴表示 ・DIRECTモード専用 ・STOP命令等で止めた際に直前までの呼び出し履歴を表示 ・スロット番号と行番号のリストを表示
書式	BACKTRACE
例	BACKTRACE

基本命令（変数と配列）

変数や配列の定義、配列操作に関する命令

=	変数に数値や式の値を代入 ・従来のBASICのLET命令を省略した書き方 ・本製品ではLET命令自体が省略されており代入には'='のみ使用	
書式	=	
例	A=10 A$="HELLO"	

DIM (1)	使用する配列を宣言 ・本製品では配列の宣言は省略できない ・添字は0から始まる ・要素数は必ず[]でくくり()は使用できない ・DIMとVARはどちらを使ってもよい	
書式	DIM 配列変数名[要素数] ,…	
引数	配列変数名[要素数]	・英数字とアンダースコア(_)が使用可能 ・先頭に数字は使用不可 ・配列変数には文字列変数も使用可能
	要素数	・確保する配列の個数を[]でくくって指定 ・要素数はカンマ(,)で区切って4次元まで指定可能
例	DIM ATR[4] DIM DX[5], DY[5], DZ[5] DIM POS[10,5]	

DIM (2)	使用する変数を宣言 ・OPTION STRICT指定時、すべての変数の宣言が必要 ・DIMを変数の定義用として利用する使い方	
書式	DIM 変数名 ,…	
引数	変数名	・英数字とアンダースコア(_)が使用可能 ・先頭に数字は使用不可 ・文字列変数も宣言可能
例	DIM A, ATRB, B$	

VAR (1)	使用する変数を宣言 OPTION STRICT指定時、すべての変数の宣言が必要	
書式	VAR 変数名 ,…	
引数	変数名	・英数字とアンダースコア(_)が使用可能 ・先頭に数字は使用不可 ・文字列変数も宣言可能
例	VAR A, ATRB, B$	

VAR (2)	使用する配列を宣言 ・本製品では配列の宣言は省略できない ・添字は0から始まる ・要素数は必ず[]でくくり()は使用できない ・DIMとVARはどちらを使ってもよい	
書式	VAR 配列変数名[要素数] ,…	
引数	配列変数名[要素数]	・英数字とアンダースコア(_)が使用可能 ・先頭に数字は使用不可 ・配列変数には文字列変数も使用可能
	要素数	・確保する配列の個数を[]でくくって指定 ・要素数はカンマ(,)で区切って4次元まで指定可能
例	VAR ATR[4] VAR DX[5], DY[5], DZ[5] VAR POS[10, 5]	

SWAP	2つの変数の値を交換 文字列と数値の交換は不可	
書式	SWAP 変数1, 変数2	
引数	変数1	交換元の変数
	変数2	交換先の変数
例	SWAP A,B	

INC	変数の値を+1 引数がある場合式の値を加算	
書式	INC 変数 [, 式]	
引数	変数	加算する変数名
	式	加算する値(省略時:1)
文字列参照型についての補足	・文字列変数と配列は参照型です ・INC,DEC,SHIFT,UNSHIFT,PUSH,POPは参照型命令 (例) A$="ABC" '初期値"ABC"のメモリーを確保 B$=A$ '変数は別だが参照先メモリーは同じ ?A$,B$ 'ABCとABCと表示 INC B$,"Z" '参照先に対して"Z"を加算 ?A$,B$ 'ABCZとABCZと表示 B$=B$+"X" '変数への足し算はコピーが発生 ?A$,B$ 'ABCZとABCZXと表示 C$=B$ '変数は異なるが参照先は同じメモリー ?A$,B$,C$ 'ABCZとABCZXとABCZXと表示 INC C$,"W" '参照先に対して"W"を加算 ?A$,B$,C$ 'ABCZとABCZXWとABCZXWと表示	
例	INC X INC X,3	

DEC	変数の値を-1 引数がある場合式の値を減算	
書式	DEC 変数 [, 式]	
引数	変数	減算する変数名
	式	減算する値(省略時:1)
文字列参照型についての補足	・INC命令内の補足説明をご覧ください	
例	DEC X DEC X,3	

COPY (1)	配列を他の配列にコピー ・1次元配列に限りコピー先要素数不足の際自動追加 ・コピー元、コピー先とも次元は無視される	
書式	COPY コピー先配列 [,コピー先オフセット],コピー元配列 [[,コピー元オフセット], コピー要素数]	
引数	コピー先配列	コピー先の配列(コピー元配列の内容で上書きされる)
	コピー先オフセット	上書きされる先頭要素(省略時はコピー先の先頭から)
	コピー元配列	コピー元の配列
	コピー元オフセット	上書きする先頭要素(省略時はコピー元の先頭から)
	コピー要素数	上書きする要素数(省略時はコピー元の末尾まで)
例	DIM SRC[10],DST[10] COPY DST,SRC	

COPY (2)	DATA列を配列に読み込む ・DATA命令に列挙されたデータを配列に読み込む ・1次元配列に限りコピー先要素数不足の際自動追加	
書式	COPY コピー先配列 [,コピー先オフセット], "@ラベル文字列" [,コピーデータ数]	
引数	コピー先配列	コピー先の配列(DATA列の内容で上書きされる)
	コピー先オフセット	上書きされる先頭要素(省略時は配列の先頭から)
	"@ラベル文字列"	読み込むDATA命令に付けられている@ラベル名を文字列で指定
	コピーデータ数	・読み込むデータ数(省略時はコピー先配列の要素数分) ・※コピー先配列数よりDATA数が少ない場合エラー
例	DIM DST[5] COPY DST,"@SRC" @SRC DATA 5,1,1,2,4	

SORT	配列を昇順で並び替える	
書式	SORT [開始位置, 要素数,] 配列1 [,配列2 ,…]	
引数	開始位置	配列1内のソートを開始する位置(0～)
	要素数	配列1内のソートする要素数(1～)
	配列1	ソートする数値の入った配列
	配列2	・配列1のソート結果に従いソートする配列 ・列挙できる配列は配列1～配列8まで
例	DIM WORK[10] SORT 0, 10, WORK	

RSORT	配列を降順で並び替える	
書式	RSORT [開始位置, 要素数,] 配列1 [,配列2 ,…]	
引数	開始位置	配列1内のソート開始位置(0～)
	要素数	配列1内のソートする要素数(1～)
	配列1	ソートする数値の入った配列
	配列2	・配列1のソート結果に従いソートする配列 ・列挙できる配列は配列1～配列8まで
例	DIM WORK[10] RSORT 0, 10, WORK	

PUSH	配列の末尾に要素を追加(要素数が1つ増える)	
書式	PUSH 配列, 式	
引数	配列	要素を追加される配列
	式	追加する要素の値
文字列参照型についての補足	・INC命令内の補足説明をご覧ください	
例	DIM WORK[10] PUSH WORK, 123	

POP	配列の末尾から要素を取り出す(要素数が1つ減る)	
書式	変数=POP(配列)	
引数	配列	要素を取り出される配列
戻り	取り出した要素の値	
文字列参照型についての補足	・INC命令内の補足説明をご覧ください	
例	DIM WORK[10] PUSH WORK, 123 A=POP(WORK)	

UNSHIFT	配列の先頭に要素を追加(要素数が1つ分増える)	
書式	UNSHIFT 配列, 式	
引数	配列	要素を追加される配列
	式	追加する要素の値
文字列参照型についての補足	・INC命令内の補足説明をご覧ください	
例	DIM WORK[10] UNSHIFT WORK, 123	

SHIFT	配列の先頭から要素を取り出す(要素数が1つ分減る)	
書式	変数=SHIFT(配列)	
引数	配列	要素を取り出される配列
文字列参照型についての補足	・INC命令内の補足説明をご覧ください	
例	DIM WORK[10] UNSHIFT WORK, 123 A=SHIFT(WORK)	

FILL	配列内容をすべて指定値に設定 ・オフセットと要素数で部分的な変更も可能 ・配列の型は、整数、実数、文字列いずれでも構わない	
書式	FILL 配列, 値 [,オフセット [,要素数]]	
引数	配列	値を上書きする配列
	値	数値や文字列
	オフセット	値を書き込み始める位置
	要素数	書き込む数
例	DIM WORK[10] FILL WORK,0	

基本命令(制御と分岐)

比較、分岐、くりかえしなどの制御命令

@	プログラムやデータの位置を示す名前 ・GOTOなどで行番号を直接指定することはできない ・分岐先やデータ位置はすべてラベルで指定	
書式	@ラベル名	
引数	@ラベル名	@で始まる英数字とアンダースコア(_)
例	@MAINLOOP	

GOTO (1)	強制分岐	
書式	GOTO @ラベル	
引数	@ラベル	・ジャンプ先の@ラベル名 ・ラベル名を""でくくったラベル文字列(文字列変数も可) ・"1:@ラベル名" の形式でプログラムSLOT指定も可能 ・あらかじめUSE命令で対象SLOTを使用可能にしておくこと
例	GOTO @MAIN JP$="@MAIN":GOTO JP$	

GOSUB (1)	サブルーチンの呼び出し	
書式	GOSUB @ラベル	
引数	@ラベル	・呼び出すサブルーチンの@ラベル名 ・ラベル名を""でくくったラベル文字列(文字列変数も可) ・"1:@ラベル名" の形式でプログラムSLOT指定も可能 ・あらかじめUSE命令で対象SLOTを使用可能にしておくこと
例	GOSUB @SUB	

RETURN (1)	サブルーチンから呼び出し元へ復帰
書式	RETURN
例	RETURN

RETURN (2)	サブルーチンから値を返しつつ呼び出し元へ復帰 関数型として定義されたDEF命令内で値を戻す場合に利用
書式	RETURN
例	DEF CALC(A,B) RETURN A*B END PRINT CALC(2,3)

OUT	複数出力が必要な場合に利用する命令 ・複数の値を返すDEF命令の宣言で利用 ・複数の値を返す組み込み命令でも利用される
書式	OUT
例	DEF SUB A OUT D,M D=A DIV 10 M=A MOD 10 END SUB 34 OUT DV,ML PRINT DV,ML

ON (1)	制御変数の値に合わせてラベル行に分岐 ・従来型BASICとは異なり分岐番号は0からとなる	
書式	ON 制御変数 GOTO @ラベル0, @ラベル1…	
引数	@ラベル0	制御変数が0のときのジャンプ先
	@ラベル1	制御変数が1のときのジャンプ先 : ・必要な数だけ飛び先を用意する ・ON～GOTOのラベルには、ラベル文字列は使えない
例	ON IDX GOTO @JMP_A,@JMP_B PRINT "OVER":END @JMP_A PRINT "IDX=0":END @JMP_B PRINT "IDX=1":END	

GOTO (2)	制御変数の値に合わせてラベル行に分岐 ・従来型BASICとは異なり分岐番号は0からとなる	
書式	ON 制御変数 GOTO @ラベル0, @ラベル1…	
引数	@ラベル0	制御変数が0のときのジャンプ先
	@ラベル1	制御変数が1のときのジャンプ先 ： ・必要な数だけ飛び先を用意する ・ON～GOTOのラベルには、ラベル文字列は使えない
例	ON IDX GOTO @JMP_A,@JMP_B PRINT "OVER":END @JMP_A PRINT "IDX=0":END @JMP_B PRINT "IDX=1":END	

ON (2)	制御変数の値に合わせてサブルーチン呼び出し ・従来型BASICとは異なり分岐番号は0からとなる	
書式	ON 制御変数 GOSUB @ラベル0, @ラベル1…	
引数	@ラベル0	制御変数が0のときのサブルーチン
	@ラベル1	制御変数が1のときのサブルーチン ： ・必要な数だけ飛び先を用意する ・ON～GOSUBのラベルには、ラベル文字列は使えない
例	ON IDX GOSUB @SUB_A,@SUB_B PRINT "EXIT":END @SUB_A PRINT "IDX=0":RETURN @SUB_B PRINT "IDX=1":RETURN	

GOSUB (2)	制御変数の値に合わせてサブルーチン呼び出し ・従来型BASICとは異なり分岐番号は0からとなる	
書式	ON 制御変数 GOSUB @ラベル0, @ラベル1…	
引数	@ラベル0	制御変数が0のときのサブルーチン
	@ラベル1	制御変数が1のときのサブルーチン ： ・必要な数だけ飛び先を用意する ・ON～GOSUBのラベルには、ラベル文字列は使えない
例	ON IDX GOSUB @SUB_A,@SUB_B PRINT "EXIT":END @SUB_A PRINT "IDX=0":RETURN @SUB_B PRINT "IDX=1":RETURN	

IF (1)	条件成立時に処理1を、不成立時に処理2を実行 ・THENやELSE直後のGOTOは、GOTOを省略可能 ・処理が複数行にわたるときはENDIFを使用	
書式	IF 条件式 THEN 成立時処理 [ELSE 不成立時処理] [ENDIF]	
条件式	比較演算子	==　等しい !=　等しくない >　より大きい <　より小さい >=　以上 <=　以下
	論理演算子 （複数条件の比較用）	(条件1 AND 条件2)　同時に満たす (条件1 && 条件2)　同時に満たす (条件1 OR 条件2)　いずれかを満たす (条件1 || 条件2)　いずれかを満たす ※||の文字は、キーボード上で ? の左上にあります

例	IF A==1 THEN PRINT "OK"
	IF A>1 THEN @JMP1 ELSE PRINT DATE$
	IF A==1 THEN
	　PRINT "オメデトウ":BEEP 72
	ELSE
	　PRINT "ザンネン"
	ENDIF
	@JMP1
	END	

THEN	IFによる条件が成立した時の制御先 条件判定については、IFの説明をご覧ください
書式	IF 条件式 THEN 成立時処理 [ELSE 不成立時処理] [ENDIF]

例	IF A==1 THEN PRINT "OK"
	IF A<1 THEN @JMP1 'GOTOの省略
	IF A==1 THEN
	 PRINT "オメデトウ":BEEP 72
	ELSE
	 PRINT "ザンネン"
	ENDIF
	@JMP1
	END

ELSE	IFによる条件が成立しない時の制御先 条件判定については、IFの説明をご覧ください
書式	IF 条件式 THEN 成立時処理 ELSE 不成立時処理 [ENDIF]

例	IF A==1 THEN PRINT "OK"
	IF A<1 THEN @JMP1 ELSE PRINT DATE$
	IF A==1 THEN
	 PRINT "オメデトウ":BEEP 72
	ELSE
	 PRINT "ザンネン"
	ENDIF
	@JMP1
	END

ELSEIF	IFによる条件不成立時の追加条件判定 ・不成立時に続けて条件判断を行う場合に利用 ・条件判定については、IFの説明をご覧ください
書式	IF 条件式 THEN 成立時処理 ELSEIF 条件式 THEN 成立時処理 ENDIF

例	IF A==1 THEN
	 PRINT "オメデトウ":BEEP 0
	ELSEIF A==2 THEN
	 PRINT "ザンネン"
	ELSE IF A==3 THEN
	 PRINT "まあまあ"
	ENDIF '--- ELSE IF の場合は必要
	ENDIF

ENDIF	IFによる制御切替後処理が複数行になる場合の終了 条件判定については、IFの説明をご覧ください
書式	IF 条件式 THEN 成立時処理 ELSE 不成立時処理 [ENDIF]

例	IF A==0 THEN
	 PRINT "A=0"
	ENDIF

IF (2)	条件が成立時@ラベルに分岐 条件判定については、IFの説明をご覧ください
書式	IF 条件式 GOTO @ラベル [ELSE 不成立時処理]
ラベルに文字列を使う場合の注意事項	・ラベルにはラベル文字列も使用可能 ・ELSE直後でのGOTO省略時に文字列は使えない × IF A==0 GOTO "@LABEL1" ELSE "@LABEL2" ○ IF A==0 GOTO "@LABEL1" ELSE @LABEL2 ○ IF A==0 GOTO "@LABEL1" ELSE GOTO "@LABEL2"
例	IF A==1 GOTO @MAIN IF X>0 GOTO @JMP1 ELSE PRINT A$ IF Y==5 GOTO @JMP1 ELSE @JMP2 @JMP1 PRINT "@JMP1" @JMP2 PRINT "@JMP2" END

GOTO (3)	条件が成立時@ラベルに分岐 ・条件判定については、IFの説明をご覧ください
書式	IF 条件式 GOTO @ラベル [ELSE 不成立時処理]
ラベルに文字列を使う場合の注意事項	・ラベルにはラベル文字列も使用可能 ・ELSE直後でのGOTO省略時に文字列は使えない × IF A==0 GOTO "@LABEL1" ELSE "@LABEL2" ○ IF A==0 GOTO "@LABEL1" ELSE @LABEL2 ○ IF A==0 GOTO "@LABEL1" ELSE GOTO "@LABEL2"
例	IF A==1 GOTO @MAIN IF X>0 GOTO @JMP1 ELSE PRINT A$ IF Y==5 GOTO @JMP1 ELSE @JMP2 @JMP1 PRINT "@JMP1" @JMP2 PRINT "@JMP2" END

FOR	処理を指定回数繰り返す ・処理の最後にはNEXT命令を置く ・条件が満たされない場合、1回も実行されないことがある	
書式	FOR ループ変数=初期値 TO 終了値 [STEP 増分]	
引数	ループ変数	ループ回数をカウントする変数(1ループごとに増分が加算)
	初期値	ループ開始時のループ変数の値または式
	TO 終了値	ループ終了時のループ変数の値または式
	STEP 増分	・ループ終わりにループ変数に加算する増分(省略時=1) ・増分が小数の時、演算誤差で意図した回数にならない可能性あり
例	FOR I=0 TO 9 STEP 2 PRINT I;","; NEXT	

TO	ループ数の終わりの値指定 ・FOR～NEXTについてはFOR命令の説明をご覧ください
書式	TO 終了値
例	FOR I=0 TO 9 STEP 2 PRINT I;","; NEXT

STEP	FORループ数の増分の値指定 ・FOR～NEXTについてはFOR命令の説明をご覧ください
書式	STEP 増分
例	FOR I=0 TO 9 STEP 2 PRINT I;","; NEXT

NEXT	FORループの終わりを示す命令 ・FOR～NEXTについてはFOR命令の説明をご覧ください ・FORループ内のIFでNEXTを使うのは非推奨 ・強制的に次のループへ進める時はCONTINUEを使用 ・ループの強制終了はBREAKを使用	
書式	NEXT [制御変数]	
引数	制御変数	・制御変数は記述しても無視され単なるNEXTと同じ働き ・NEXT J,I のような記述は不可
例	FOR I=0 TO 9 STEP 2 PRINT I;","; NEXT	

WHILE	条件が成立している間、WENDまでをくりかえす ・不成立時、またはBREAK命令でループを抜ける	
書式	WHILE 条件式	
条件式	IFと同等の条件式を書くことができます	
	比較演算子	== 等しい != 等しくない > より大きい < より小さい >= 以上 <= 以下
	論理演算子(複数条件の比較用)	(条件1 AND 条件2) 同時に満たす (条件1 && 条件2) 同時に満たす (条件1 OR 条件2) いずれかを満たす (条件1 || 条件2) いずれかを満たす ※||の文字は、キーボード上で ? の左上にあります
例	A=0:B=4 WHILE A<B A=A+1 WEND	

WEND	WHILEループの終わりを示す命令
書式	WEND
例	A=0:B=4 WHILE A<B A=A+1 WEND

REPEAT	REPEATループの開始命令 ・ループの終わりにUNTIL命令と条件式を置く ・WHILE命令とは異なり先に処理を実行してから条件判断 ・成立時、またはBREAK命令でループを抜ける
書式	REPEAT
例	A=0:B=4 REPEAT A=A+1 UNTIL A>B

UNTIL	条件が満たされるまでREPEATからをくりかえす ・ループの始まりにREPEAT命令を置く ・WHILE命令とは異なり先に処理を実行してから条件判断 ・成立時、またはBREAK命令でループを抜ける	
書式	UNTIL 条件式	
条件式	IFと同等の条件式を書くことができます	
	比較演算子	==　等しい !=　等しくない >　より大きい <　より小さい >=　以上 <=　以下
	論理演算子(複数条件の比較用)	(条件1 AND 条件2)　同時に満たす (条件1 && 条件2)　同時に満たす (条件1 OR 条件2)　いずれかを満たす (条件1 || 条件2)　いずれかを満たす ※||の文字は、キーボード上で ? の左上にあります
例	A=0:B=4 REPEAT A=A+1 UNTIL A<B	

CONTINUE	ループを強制的に次に進める ・FOR～NEXT、WHILE～WEND、REPEAT～UNTIL内で使用
書式	CONTINUE
例	FOR I=0 TO 9 IF I==1 THEN CONTINUE IF I==7 THEN BREAK PRINT I;","; NEXT

BREAK	ループから強制的に抜ける ・FOR～NEXT、WHILE～WEND、REPEAT～UNTIL内で使用
書式	BREAK
例	FOR I=0 TO 9 IF I==1 THEN CONTINUE IF I==7 THEN BREAK PRINT I;","; NEXT

END (1)	プログラムを終了
書式	END
例	END

END (2)	ユーザー関数、ユーザー命令のDEF定義を終了
書式	END
例	DEF FUNC PRINT "FUNC" END

STOP	実行中のプログラムを中断 ・中断したプログラムSLOT:行番号が表示される ・CONT命令で続行可能(ただし状況によっては続行できない)
書式	STOP
例	STOP

基本命令(高度な制御)

ユーザー定義関数や拡張機器の制御などの命令

DEF (1)	ユーザー定義命令について 1) USER ※引数なし、戻り値なし 2) USER X,Y ※引数あり、戻り値なし 3) A=USER(X) ※引数あり、戻り値1つ 4) USER(X) OUT A,B ※引数あり、戻り値複数 DEFを使うと上記のような命令を独自定義が可能	
DEF	共通の補足	・DEF～ENDまでが定義範囲となる ・DEF～END範囲で定義された変数やラベルはローカル扱い ・DEF～END範囲を越えるGOTOはできない ・DEF～END範囲内でGOSUBやON GOSUBは使用できない GOSUB "0:@SUB"のようにSLOT指定があれば利用可能 ・COMMON 命令を付けることでSLOTを超えて利用可能
	引数の仕様	・DEFで受け取る引数に関して厳密な型チェックは行わない ・必要な数だけカンマ(,)で区切り受け取る変数名を記述可能 ・文字列変数には変数名の終わりに$を付けることも可能
	戻り値の仕様	・DEFの戻り値に関しての厳密な型チェックは行わない ・出力用変数に最初に書いた値で型が確定 ・数値変数に整数値を代入すると整数型として扱う ・実数型として扱う場合は、A=100.0のように記述 ・DEFからの戻り値を受け取った側の型と異なる場合エラー

DEF (2)	返り値と引数の無いユーザー命令の定義
書式	DEF 定義名
引数	なし
戻り	なし
例	'--- 文字表示 DEF FUNC PRINT "SAMPLE" END '--- 呼び出し FUNC

DEF (3)	引数があり返り値が無いユーザー命令の定義
書式	DEF 定義名 引数 [,引数…]
引数	関数に渡したい引数があれば必要な分の変数名を記述
戻り	なし
例	'--- 指定位置に文字表示 DEF FUNC2 X,Y LOCATE X,Y PRINT "SAMPLE" END '--- 呼び出し FUNC2 10,4

DEF (4)	戻り値が1つだけあるユーザー関数の定義
書式	DEF 関数名([引数 [,引数…]])
引数	関数に渡したい引数があれば必要な分の変数名を記述
戻り	結果として返したい値を、RETURN命令の後に書く ※RETURN ANS のような記述
例	'---足し算 DEF ADD(X,Y) RETURN X+Y END '--- 再帰を使った階乗計算 DEF FACTORIAL(N) IF N==1 THEN RETURN N RETURN N*FACTORIAL(N-1) END '--- 文字列反転 DEF REVERSE$(T$) VAR A$="" 'ローカル文字列 VAR L=LEN(T$) 'ローカル WHILE L>0 A$=A$+MID$(T$,L-1,1) DEC L WEND RETURN A$ END '--- 呼び出し PRINT ADD(10,5) PRINT FACTORIAL(4) PRINT REVERSE$("BASIC")

DEF (5)	複数の返り値を持つユーザー命令の定義
書式	DEF 命令名 [引数 [,引数…]] OUT V1 [,V2…]]
引数	関数に渡したい引数があれば必要な分の変数名を記述
戻り	結果として返したい数分の変数名をOUTの後に記述
例	'--- 足し算と掛け算 DEF CALCPM A,B OUT OP,OM OP=A+B OM=A*B END '--- 呼び出し CALCPM 5,10 OUT P,M PRINT P,M

COMMON	COMMONについて 1) COMMON DEF USER 2) COMMON DEF USER(X) 3) COMMON DEF USER(X) OUT A,B COMMONはSLOTを超えて独自命令を使う場合に利用 異なるSLOT間でプログラムを使う場合はUSEが必要
例	COMMON DEF FOO(X, Y, Z)

CALL (1)	指定名称を持つユーザー定義命令を呼び出す	
書式	CALL "命令名" [,引数…] [OUT 変数1 [,変数2…]]	
引数	命令名	・呼び出すユーザー定義命令名の文字列 ・文字列なので""でくくるか文字列変数を使用
	引数～	指定した命令に必要な引数
戻り	結果として返したい数分の変数名をOUTの後に記述	
例	CALL "USERCD",X,Y OUT A,B ' DEF USERCD X,Y OUT A,B A=X+Y:B=X*Y END	

CALL (2)	指定名称を持つユーザー定義関数を呼び出す	
書式	変数=CALL("関数名" [,引数…])	
引数	関数名	・呼び出すユーザー定義関数名の文字列 ・文字列なので""でくくるか文字列変数を使用
	引数	指定した関数に必要な引数を列挙
例	A=CALL("USERFC",X,Y) ' DEF USERFC(X,Y) RETURN X*Y END	

CALL (3)	SPRITEコールバックの呼び出し SPFUNCで設定されたSPRITE毎の処理を一斉呼び出し	
書式	CALL SPRITE	
例	CALL SPRITE	

CALL (4)	BGコールバックの呼び出し SPFUNCで設定されたSPRITE毎の処理を一斉呼び出し	
書式	CALL BG	
例	CALL BG	

XON	特殊機能の利用宣言 ・これらの機能は宣言しないと利用できない ・XON EXPADが成功するとRESULTにTRUEが返る ・すでにXON状態の場合ダイアログ等は表示されない ・WiiU専用の機能を使う場合には必ず先にXON WIIUが必要（プチコンBIGのみ）	
書式	XON 使用する機能名	
引数	使用する機能名	MOTION: モーションセンサー、ジャイロセンサー EXPAD: 拡張スライドパッド MIC: マイク COMPAT: 3DS/WiiU互換モードへの切替 3DS: 3DS専用モードへの切替 WIIU: WiiU専用モードへの切替 ※WiiUでは、EXPADを指定しなくても入力値取得可能
例	XON MOTION	

XOFF	XONで宣言した特殊機能の使用終了	
書式	XOFF 停止する機能名	
引数	停止する機能名	MOTION: モーションセンサー、ジャイロセンサー EXPAD: 拡張スライドパッド MIC: マイク
例	XOFF MOTION	

基本命令(データ操作・その他)

データ読み込み、垂直同期、コメントなどの命令

READ	DATA命令で列挙した情報を変数に読み込む DATA命令で列挙した型と同じ型で読むこと	
書式	READ 取得変数1 [, 取得変数2…]	
引数	取得変数	・読み込む情報を格納する変数(複数指定可能) ・RESTORE命令で指定した行以降のDATAから取得 ・RESTORE省略時、最初のDATAから取得
例	READ X,Y,Z,G$ DATA 200,120,0,"JAN" DATA 210,120,0,"FEB"	

DATA	READで読み込むデータの定義 ・数値も文字列も混在可能 ・数値定数のみの式は定数として扱われるためDATA文に記述可能 ・#で始まる定数も記述可能 ・&&、||、変数や関数の混ざった式は記述できない ・文字列式は記述できない
書式	DATA データ [, データ…]
データの記述方法	・数値や文字列を ',' で区切って並べる ・文字列は必ず""でくくる(""は省略不可)
例	READ X,Y,Z,ST$ '行末にコメントが書けます DATA 123,345,56,"SAMPLE"

RESTORE	READ命令が読み込む先頭DATAを指定	
書式	RESTORE @ラベル	
引数	@ラベル	・読み込むDATA命令の先頭に付けられた@ラベル名 ・@ラベル名が代入された文字列変数も指定可能 ・RESTORE "1:@ラベル名" で他のSLOTから参照も可能 ・あらかじめUSE 1等で対象SLOTを使用可能にしておく
例	RESTORE @DATATOP @DATATOP DATA 123,345,56,"SAMPLE"	

OPTION	プログラムの動作モードを設定	
書式	OPTION 機能名	
引数	機能名	STRICT: 変数宣言が必須となる(宣言なし参照はエラー) DEFINT: 変数のデフォルト型を整数型にする
例	OPTION STRICT	

WAIT	指定回数分の垂直同期が来るまでプログラム停止	
書式	WAIT [フレーム数]	
引数	フレーム数	現在からの経過フレーム数を指定(0:無視、省略時は1)
例	WAIT 60	

VSYNC	指定回数分の垂直同期が来るまでプログラム停止 WAITとは異なり前回VSYNCからのVSYNC回数	
書式	VSYNC [フレーム数]	
引数	フレーム数	前回のVSYNCからの経過フレーム数を指定(0:無視 省略時=1)
例	VSYNC 1	

'	コメント記述用の記号 ・コメント内容はプログラム実行に影響しない
書式	' [文字列]
例	' ---MAIN ROUTINE---

REM	コメント記述用の命令 ・コメント内容はプログラム実行に影響しない
書式	REM [文字列]
例	REM ---MAIN ROUTINE---

KEY	ファンクションキーに任意の文字列を割り当てる	
書式	KEY 番号,"文字列"	
引数	番号	ファンクションキーの番号(1～5)
	文字列	・割り当てる文字列 ・表示しきれない場合は最後を'…'として表示
例	KEY 1,"CLS"+CHR$(13)	

TMREAD	時間文字列を数値に変換	
書式	TMREAD ["時間文字列"] OUT H,M,S	
引数	時間文字列	"HH:MM:SS"形式の時間文字列(省略時、現在の時間)
戻り	数値格納先変数	H: 時間を受け取る変数(0～23) M: 分を受け取る変数 S: 秒を受け取る変数
例	TMREAD "12:59:31" OUT H,M,S	

DTREAD	日付文字列を数値に変換	
書式	DTREAD ["日付文字列"] OUT Y,M,D [,W]	
引数	日付文字列	"YYYY/MM/DD"形式の日付文字列(省略時現在の日時)
戻り	数値格納先変数	Y: 年を受け取る変数 M: 月を受け取る変数 D: 日を受け取る変数 W: 曜日(日曜日を0とする数値)を受け取る変数
例	DTREAD "2014/10/12" OUT Y,M,D	

CHKLABEL	指定文字列で参照できるラベルの存在確認	
書式	変数 = CHKLABEL("@ラベル文字列"[,フラグ])	
引数	@ラベル文字列	・CHKLABEL "1:@ラベル名" で他のSLOT確認も可能 ・あらかじめUSE 1 等で対象SLOTを使用可能にしておく
	フラグ	0=DEF内だけを検索(省略時=0) 1=DEF内に無ければグローバルラベルを検索
戻り	FALSE=存在しない、TRUE=存在	
例	A=CHKLABEL("@MAIN")	

CHKCALL	指定文字列で参照できる命令・関数の存在確認	
書式	変数 = CHKCALL("文字列")	
引数	文字列	調べたい命令・関数の文字列
戻り	FALSE=存在しない、TRUE=存在	
例	A=CHKCALL("KEYCHECK")	

CHKVAR	指定文字列で参照できる変数の存在確認	
書式	変数 = CHKVAR("文字列")	
引数	文字列	調べたい変数の文字列
戻り	FALSE=存在しない、TRUE=存在	
例	A=CHKVAR("COUNTX")	

DIALOG (1)	ダイアログを表示しボタンが押されるまで待つ ・システム変数 RESULT に結果を返す ・RESULT：1(決定)、-1(キャンセル)、0(タイムアウト)	
書式	DIALOG "テキスト文字列"	
引数	テキスト文字列	ダイアログに表示する文字列
補足(DIALOG命令共通)	・ダイアログは必ず下画面に表示 ・テキスト文字列、キャプション文字列の合計は250文字まで ・テキスト文字列にCHR$(10)かCHR$(13)があると改行 ・タイムアウト時間をマイナス値にするとフレーム単位となる	
例	DIALOG "おはようございます！"	

DIALOG (2)	ダイアログを表示しボタンが押されるまで待つ	
書式	DIALOG "テキスト文字列",[選択タイプ],["キャプション文字列"],[タイムアウト時間]	
引数	テキスト文字列	ダイアログに表示する文字列
	選択タイプ	0: 了解(デフォルト) 5: 次へ
	キャプション文字列	ダイアログ上部のキャプション欄に表示する文字列
	タイムアウト時間	ダイアログを自動的に閉じるまでの秒数(省略時=0:閉じない)
補足(DIALOG命令共通)	・ダイアログは必ず下画面に表示 ・テキスト文字列、キャプション文字列の合計は256文字まで ・テキスト文字列にCHR$(10)がCHR$(13)があると改行 ・タイムアウト時間をマイナス値にするとフレーム単位となる	
例	DIALOG "はじめましょう",5,"シナリオ",-120	

DIALOG (3)	ダイアログを表示し指定したボタンが押されるまで待つ	
書式	変数 = DIALOG("テキスト文字列",[選択タイプ],["キャプション文字列"],[タイムアウト時間])	
引数	テキスト文字列	ダイアログに表示する文字列
	選択タイプ	0: 了解(デフォルト) 1: いいえ/はい 2: 戻る/次へ 3: 中止/決定 4: 中止/実行 5: 次へ
	キャプション文字列	ダイアログ上部のキャプション欄に表示する文字列
	タイムアウト時間	ダイアログを自動的に閉じるまでの秒数(省略時=0:閉じない)
戻り	-1: 否定(左ボタン) 0: タイムアウト 1: 肯定(右ボタン) ※システム変数 RESULT にも値が残ります	
補足(DIALOG命令共通)	・ダイアログは必ず下画面に表示 ・テキスト文字列、キャプション文字列の合計は256文字まで ・テキスト文字列にCHR$(10)がCHR$(13)があると改行 ・タイムアウト時間をマイナス値にするとフレーム単位となる	
例	R=DIALOG("もう一度しますか？",1,"確認",0)	

DIALOG (4)	ダイアログ表示しタッチやハードウェアボタンが押されるまで待つ	
書式	変数 = DIALOG("テキスト文字列",ボタン種類,["キャプション文字列"],[タイムアウト時間])	
引数	テキスト文字列	ダイアログに表示する文字列
	ボタン種類	|b00|ABXYボタン(1) |b01|十字キー(2) |b02|L,Rボタン(4) |b03|タッチパネル(8) ・上記ビット値をORし符号反転したものを指定する ・ZL,ZRボタンは検出不可 ・-1でABXYのみの指定となる ・例えば、ABXYと十字キーを検出する場合は-3を指定
	キャプション文字列	ダイアログ上部のキャプション欄に表示する文字列
	タイムアウト時間	ダイアログを自動的に閉じるまでの秒数(省略時=0:閉じない)
戻り	128: Aボタン押下 129: Bボタン押下 130: Xボタン押下 131: Yボタン押下 132: 十字キー上押下 133: 十字キー下押下 134: 十字キー左押下 135: 十字キー右押下 136: Lボタン押下 137: Rボタン押下 140: タッチパネル押下	
例	R=DIALOG("ABXYLR/十字キー/タッチ",-15,"特別",0)	

DIALOG (5)	ファイル名入力専用のダイアログを表示	
書式	文字列=DIALOG("初期文字列", "キャプション文字列" [,最大文字数])	
引数	初期文字列	最初に入力されている文字列
	キャプション文字列	キャプションに表示する文字列
	最大文字数	最大14文字まで
戻り	取得した文字列が返ります ※RESULT=-1の時は中止(文字列無効)	
補足(DIALOG命令共通)	・ダイアログは必ず下画面に表示 ・テキスト文字列、キャプション文字列の合計は256文字まで ・テキスト文字列にCHR$(10)がCHR$(13)があると改行 ・タイムアウト時間をマイナス値にするとフレーム単位となる	
例	T$=DIALOG("NEWNAME0","SAVE", 14)	

DIALOG (6)	DIALOGへの漢字表示サンプル 漢字を使う場合はUTF-16形式の文字コードをCHR$に渡す ※UTF-16形式については専門書等を参考にしてください
補足(DIALOG命令共通)	・ダイアログは必ず下画面に表示 ・テキスト文字列、キャプション文字列の合計は256文字まで ・テキスト文字列にCHR$(10)がCHR$(13)があると改行 ・タイムアウト時間をマイナス値にするとフレーム単位となる
例	' もう一度？ T$="もう"+CHR$(&H4E00)+CHR$(&H5EA6)+"？" ' 確認 C$=CHR$(&H78BA)+CHR$(&H8A8D) R=DIALOG(T$,1,C$,0)

CLIPBOARD (1)	クリップボードの内容を設定する	
書式	CLIPBOARD 文字列	
引数	文字列	クリップボードに格納する文字列
例	CLIPBOARD "おはようございます"	

CLIPBOARD (2)	クリップボードの内容を取得する
書式	CLIPBOARD()
戻り	クリップボード内の文字列
例	PRINT CLIPBOARD()

DLCOPEN	「カタログIPオープン化プロジェクト」対象IPが利用可能か調べる。対象IPを利用する場合、作品内で使用するすべてのIPを宣言する必要がある。宣言しないで利用している場合、サーバー公開時に公開停止対象となる。 ・「カタログIPオープン化プロジェクト」参加規約に準拠 ・オリジナルと全く同じルールのゲーム公開は禁止 ・非購入者が公開キーで購入者の作品を入手しても実行できません ・非購入者によるカタログIP作品の公開は削除対象となります ・日本国内専用 **COPYRIGHT** (C)BANDAI NAMCO Entertainment Inc.	
書式	DLCOPEN "IP名" [,"IP名2" ,...]	
引数	IP名	利用する対象IP名略語（以下の文字列より指定、配信されていないIP名は無効です） GALAXIAN PACMAN XEVIOUS GALAGA MAPPY DIGDUG DRUAGA B_CITY S_LUSTER SKYKID D_BUSTER GENPEI BABEL A_VALKYRIE YOKAI W_MOMO WAGAN
	IP名2～	同時に複数のIPを利用する場合は含まれるすべてのIP名を引数として渡す
例	DLCOPEN "XEVIOUS"	

コンソール入出力

画面への文字表示や文字列の入力に関する命令

CLS	コンソール画面を消去(DISPLAY命令で指定された画面)
書式	CLS
例	DISPLAY 0 CLS

COLOR	コンソール画面の表示色を指定 テキストカラー用の定数利用可能(#TBLACK～#TWHITE)	
書式	COLOR 描画色 [,背景色]	
引数	描画色	0: 透明色 1: 黒、#TBLACK 2: 暗い赤、#TMAROON 3: 赤、#TRED 4: 暗い緑、#TGREEN 5: 緑、#TLIME 6: 暗い黄色、#TOLIVE 7: 黄色、#TYELLOW 8: 紺色、#TNAVY 9: 青、#TBLUE 10: 暗いマゼンタ、#TPURPLE 11: マゼンタ、#TMAGENTA 12: 暗いシアン、#TTEAL 13: シアン、#TCYAN 14: 灰色、#TGRAY 15: 白、#TWHITE
	背景色	・文字ごとの背景用色番号(0～15:描画色参照) ・背景色だけを変更するときは、描画色の省略可能
例	COLOR 7,4 COLOR #TWHITE COLOR ,0	

LOCATE	コンソール画面への文字表示位置を指定	
書式	LOCATE [座標X],[座標Y] [,座標Z]	
引数	座標X,Y	・文字単位の座標(X:0～49,Y:0～29) ・XSCREEN 4と下画面は(X:0～39) ・WIDTH 16実行時(X:0～24,Y:0～14) ・WIDTH 16のXSCREEN 4と下画面は(X:0～19) ・X,Yを省略すると、それぞれ以前のX,Y座標を維持 ・XSCREEN 5か6の場合は指定した解像度に合わせた範囲となります（プチコンBIGのみ）
	座標Z	・奥行方向の座標(奥:1024<液晶面:0<手前:-256) ・省略すると、以前のZ座標を維持
例	LOCATE 20,15 LOCATE 0,0,-200	

PRINT	コンソール画面への文字表示 ・式を省略すると改行のみ行う ・PRINTは?で代用可能	
書式	PRINT [式 [;または, 式…]]	
引数	式	・表示する変数、文字列変数、数値、文字列 ・四則演算等や関数による計算式も記述可(計算結果が表示される)
	;(セミコロン)	表示後に改行せず、次の表示を密着させる
	,(カンマ)	・表示後に改行せず、次の表示を一定間隔開ける ・表示位置はシステム変数のTABSTEP単位に従う
例	PRINT "RESULT(X,Y)=";DX*4+1,DY+1	

ATTR	コンソール画面に表示する文字の回転・反転属性を設定 テキスト属性用の定数利用可能(#TROT0～270、#TREVH,V)	
書式	ATTR 表示属性	
引数	表示属性	|b00|↑90度単位の回転(b00とb01の2ビットで指定) |b01|↓#TROT0、#TROT90、#TROT180、#TROT270 |b02|横反転(0=OFF、1=ON)、#TREVH |b03|縦反転(0=OFF、1=ON)、#TREVV
例	ATTR 3:PRINT "ABC"	

SCROLL	コンソール画面全体の表示位置調整 ・視点が移動するイメージ(文字が移動する方向は逆) ・画面外に押し出された文字は消える	
書式	SCROLL 文字数X, 文字数Y	
引数	文字数X	横方向の視点移動量(マイナス値で左、プラス値で右)
	文字数Y	縦方向の視点移動量(マイナス値で上、プラス値で下)
例	SCROLL 5,7	

CHKCHR	コンソール画面の文字コードを調べる	
書式	変数 = CHKCHR(座標X,座標Y)	
引数	座標X,Y	文字単位の座標(X:0～49,Y:0～29)
戻り	UTF-16文字コード	
例	CODE=CHKCHR(0,0)	

INPUT	キーボードから数値または文字列を入力 ・ENTERキーが入力されるまで入力待ち ・入力数が不足時「?Redo from start」を表示し再入力	
書式	INPUT ["ガイド文字列";] 変数[,変数2…]	
引数	ガイド文字列	・入力用のガイドメッセージ(省略可能) ・ガイド文字列後の;を,(カンマ)にすると?が表示されない ・;を使うときのみガイド文字列には文字列変数も使用可能
	変数	・入力を受け取る変数(数値または文字列変数) ・変数を複数指定する場合,(カンマ)で区切る
例	INPUT "名前と年齢は";NM$,AG	

LINPUT	キーボードから文字列の取得 ・INPUT命令では入力できない「,」等も受け付ける ・ENTERキーが入力されるまで入力待ち	
書式	LINPUT ["ガイド文字列";] 文字列変数	
引数	ガイド文字列	入力用のガイドメッセージ(省略可能)
	文字列変数	1行分の入力を受け取る文字列変数
例	LINPUT "ADDRESS:";ADR$	

INKEY$	キーボードから1文字取得(入力待ちなし)
書式	文字列変数=INKEY$()
引数	なし
戻り	・キーボードからの1文字(UTF-16) ・入力がない場合 "" が戻る
例	C$=INKEY$()

FONTDEF (1)	指定文字コードのフォントを定義	
書式	FONTDEF 文字コード, "フォント定義文字列"	
引数	文字コード	フォントを定義する文字コード(UTF-16)
	フォント定義文字列	・1ドットはRGBA=5551形式の16ビット色コード ・RGB各色5ビット(0～31) +α1ビット(0:透明、1:不透明) ・色要素を16進数4ケタの文字列で扱う ・例)白:FFFF 黒:0001 赤:F801 ・1文字は8×8=64ドットフォント定義文字列は計256文字
関連	GCOPY、GSAVE、GLOADのページ番号-1でフォント画像操作可能	
例	F$="FFFF":Z$="0000" D$=F$*7+Z$ D$=D$+F$*2+Z$*3+F$*2+Z$ D$=D$+F$+Z$+F$*3+Z$+F$+Z$ D$=D$+F$+Z$*5+F$+Z$ D$=D$+F$+Z$+F$*3+Z$+F$+Z$ D$=D$+F$+Z$+F$*3+Z$+F$+Z$ D$=D$+F$*7+Z$ D$=D$+Z$*8 FONTDEF ASC("A"),D$	

FONTDEF (2)	指定文字コードのフォントを定義	
書式	FONTDEF 文字コード，数値配列	
引数	文字コード	フォントを定義する文字コード(UTF-16)
	フォント定義配列	・1ドット1要素の数値配列を用意(1文字8×8ドット=64要素) ・1ドットはRGBA=5551形式の16ビット色コード ・RGB各色5ビット(0～31) +α1ビット(0:透明、1:不透明) ・例)白:＆HFFFF 黒:＆H0001 赤:＆HF801
関連	GCOPY、GSAVEのページ番号-1でフォント画像操作可能	

例

```
DIM F%[64]
DATA "11111110"
DATA "11000110"
DATA "10111010"
DATA "10000010"
DATA "10111010"
DATA "10111010"
DATA "11111110"
DATA "00000000"
TOP=ASC("A"):CNT=1
FOR I=0 TO CNT-1
 FOR D=0 TO 7
  READ F$
  FOR B=0 TO 7
   C=0:IF MID$(F$,B,1)=="1" THEN C=&HFFFF
   F%[D*8+B]=C
  NEXT
 NEXT
 FONTDEF TOP+I,F%
NEXT
```

FONTDEF (3)	フォントの定義を初期状態に戻す
書式	FONTDEF
例	FONTDEF

WIDTH (1)	コンソール文字サイズの変更 ・文字拡大表示のため滑らかな表示拡大ではありません ・文字が小さくて見えにくい人向けの表示補助機能	
書式	WIDTH フォントサイズ	
引数	フォントサイズ	8: 8x8ドット(標準状態) 16: 16x16ドット(通常の縦横2倍表示)
例	WIDTH 16	

WIDTH (2)	コンソール文字サイズの取得
書式	変数=WIDTH()
引数	なし
戻り	8: 8x8ドット(標準状態) 16: 16x16ドット(通常の縦横2倍表示)
例	FS=WIDTH()

各種入力

ボタン・スティック・タッチ・センサー・マイクの情報取得命令

BUTTON (1)	ハードウェアボタンの状態取得 戻り値に対してボタン用の定数利用可能	
書式	変数=BUTTON([機能ID [,端末ID]])	
引数	機能ID	0: 押され続けている状態 1: 押された瞬間(リピート機能付き) 2: 押された瞬間(リピート機能なし) 3: 放された瞬間
	端末ID(0～3)	ワイヤレス通信で他の端末から取得時に指定
戻り	|b00|十字ボタン上(1)、#UP |b01|十字ボタン下(2)、#DOWN |b02|十字ボタン左(4)、#LEFT |b03|十字ボタン右(8)、#RIGHT |b04|Aボタン(16)、#A |b05|Bボタン(32)、#B |b06|Xボタン(64)、#X |b07|Yボタン(128)、#Y |b08|Lボタン(256)、#L |b09|Rボタン(512)、#R |b10|未使用 |b11|ZRボタン(2048)、#ZL |b12|ZLボタン(4096)、#ZR ・b0～b15のビットに各ボタンが対応(押されたビット=1) ・ボタン名称横の()内は10進数表記 ・ZRとZLボタンは拡張スライドパッド使用時のみ有効	
例	B=BUTTON() B=BUTTON(0,3)	

BUTTON (2)	※BIG固有命令 ハードウェアボタンの状態取得 戻り値に対してボタン用の定数利用可能 3DS版との互換性を維持する場合は(1)を参照	
書式	変数=BUTTON([機能ID [,コントローラID]])	
引数	機能ID	0: 押され続けている状態 1: 押された瞬間(リピート機能付き) 2: 押された瞬間(リピート機能なし) 3: 放された瞬間
	コントローラID(0～4)	0: GamePad 1～4: Wii互換コントローラ ・事前にXON WIIUが必要
戻り	|b00|十字ボタン上(1)、#UP |b01|十字ボタン下(2)、#DOWN |b02|十字ボタン左(4)、#LEFT |b03|十字ボタン右(8)、#RIGHT |b04|Aボタン(16)、#A |b05|Bボタン(32)、#B |b06|Xボタン/Wiiリモコン1ボタン(64)、#X |b07|Yボタン(128)、#Y |b08|Lボタン/ヌンチャクC(256)、#L |b09|Rボタン(512)、#R |b10|未使用 |b11|ZRボタン(2048)、#ZL |b12|ZLボタン/ヌンチャクZ(4096)、#ZR |b13|未使用 |b14|Lスティック押下(16384) |b15|Rスティック押下(32768) ・b0～b15のビットに各ボタンが対応(押されたビット=1) ・ボタン名称横の()内は10進数表記	
例	B=BUTTON() B=BUTTON(0,3)	

BREPEAT	キーリピート機能の設定 ・開始時間とインターバルを省略した場合リピートOFF ・BUTTONのボタンごとのビット値とは異なる管理番号 ・3DSでは、ZRとZLボタンは拡張スライドパッド使用時のみ有効	
書式	BREPEAT ボタンID, 開始時間, インターバル	
引数	ボタンID	0: 十字ボタン上のID 1: 十字ボタン下のID 2: 十字ボタン左のID 3: 十字ボタン右のID 4: AボタンのID 5: BボタンのID 6: XボタンのID 7: YボタンのID 8: LボタンのID 9: RボタンのID 10: 未使用 11: ZRボタンのID 12: ZLボタンのID
	開始時間	最初に押されてからリピート開始までの時間(1/60秒単位)
	インターバル	リピート開始後のリピート間隔(1/60秒単位、0=リピートOFF)
例	BREPEAT 0,15,4	

STICK (1)	スライドパッドの情報取得	
書式	STICK [端末ID] OUT X,Y	
引数	端末ID(0～3)	ワイヤレス通信で他の端末から取得時に指定
戻り	X,Y	・変化量を受け取る変数(X:±1.0, Y:±1.0) ・実際に返る値は±約0.86ぐらい ・Yの値は↑が正↓が負となる
例	STICK OUT X,Y STICK 3 OUT X,Y	

STICK (2)	※BIG固有命令 指定コントローラーの左スティックの情報取得 3DS版との互換性を維持する場合は(1)を参照	
書式	STICK [コントローラID] OUT X,Y	
引数	コントローラID(0～4)	0: GamePad 1～4: Wii互換コントローラ ・事前にXON WIIUが必要
戻り	X,Y	・変化量を受け取る変数(X:±1.0, Y:±1.0) ・Yの値は↑が正↓が負となる ・左スティックが存在しない場合X,Yは常に0
例	STICK 0 OUT X,Y	

STICKEX (1)	・拡張スライドパッドのスティック情報取得 ・事前にXON EXPADで拡張スライドパッドを有効化	
書式	STICKEX [端末ID] OUT X,Y	
引数	端末ID(0～3)	ワイヤレス通信で他の端末の情報を取得する際に指定
戻り	X,Y	変化量を受け取る変数(X:±1.0, Y:±1.0)
例	XON EXPAD STICKEX OUT X,Y	

STICKEX (2)	※BIG固有命令 ・指定コントローラーの右スティック情報取得 ・XON EXPADの宣言無しで利用可能 ・3DS版との互換性を維持する場合は(1)を参照	
書式	STICKEX [コントローラID] OUT X,Y	
引数	コントローラID(0～4)	0: GamePad 1～4: Wii互換コントローラ ・事前にXON WIIUが必要
戻り	X,Y	・変化量を受け取る変数(X:±1.0, Y:±1.0) ・Yの値は↑が正↓が負となる ・右スティックが存在しない場合X,Yは常に0
例	STICKEX 0 OUT X,Y	

ACCEL	加速度情報取得 ・事前にXON MOTIONでモーションセンサーを有効化 ・常に重力方向へ1Gの加速度を検出し続ける事に注意 ・傾けて操作する場合に利用すると便利
書式	ACCEL OUT X,Y,Z
引数	なし
戻り	X,Y,Z　加速度を受け取る変数(単位:G)
例	XON MOTION ACCEL OUT X,Y,Z

GYROV	ジャイロセンサーの角速度情報取得 事前にXON MOTIONでモーションセンサーを有効化
書式	GYROV OUT P,R,Y
引数	なし
戻り	**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)
例	XON MOTION GYROV OUT P,R,Y

GYROA	ジャイロセンサーの角度情報取得 事前にXON MOTIONでモーションセンサーを有効化
書式	GYROA OUT P,R,Y
引数	なし
戻り	**P**　Pitch(X軸角度)を取得する変数(単位:ラジアン)
	R　Roll(Y軸角度)を取得する変数(単位:ラジアン)
	Y　Yaw(Z軸角度)を取得する変数(単位:ラジアン)
例	XON MOTION GYROA OUT P,R,Y

GYROSYNC	ジャイロ情報の更新 ・ジャイロ情報は取得を繰り返すと誤差が蓄積される ・適宜リセットするために呼び出す ・ただし1フレーム間隔以下での呼び出しは禁止
書式	GYROSYNC
例	GYROSYNC

TOUCH (1)	タッチ情報取得 画面周辺の5ドットは読み取れません
書式	TOUCH [端末ID] OUT TT,TX,TY
引数	端末ID(0～3)　ワイヤレス通信で他の端末の情報を取得する際に指定
戻り	TT　タッチされた時間を受け取る変数(0=タッチなし)
	TX,TY　・タッチされた座標を受け取る変数(TX:5～314, TY:5～234) ・下画面のサイズと同じ範囲では返らないことに注意
例	TOUCH OUT TT,TX,TY

TOUCH (2)	※BIG固有命令 GamePadのタッチ情報取得 ・DISPLAY 1の画面非表示状態ではタッチ無効(TT=0) ・3DS版との互換性を維持する場合は(1)を参照
書式	TOUCH 座標変換フラグ OUT TT,TX,TY
引数	座標変換フラグ　0: DISPLAY 1解像度範囲に変換された値 1: GamePadネイティブ値（854x480） ・事前にXON WIIUが必要
戻り	TT　タッチされた時間を受け取る変数(0=タッチなし)
	TX,TY　・タッチされた座標を受け取る変数(DISPLAY 1解像度の範囲) ・タッチパネルの周辺約8ドットは読み取れません
例	XON WIIU TOUCH 1 OUT TT,TX,TY

MICSTART (1)	※3号固有命令 マイクからのサンプリング開始 ・事前にXON MICでマイクを有効化 ・3DSでは XON MIC 直後、および蓋開け直後は1秒間無音となる ・システム内のサンプリング用メモリーに記録	
書式	MICSTART サンプリングレート，ビット数，秒数	
引数	サンプリングレート	0: 8180Hz 1: 10910Hz 2: 16360Hz 3: 32730Hz
	ビット数	0: 8ビット 1: 16ビット 2: 8ビット符号付 3: 16ビット符号付
	秒数	0: ループモード 1〜: サンプリング秒数 ・8180Hz、8bit時最大32秒、16bit時最大16秒 ・10910Hz、8bit時最大24秒、16bit時最大12秒 ・16360Hz、8bit時最大16秒、16bit時最大 8秒 ・32730Hz、8bit時最大 8秒、16bit時最大 4秒 ・ループモード時、最大秒数を超えると先頭から上書き
例	XON MIC MICSTART 0,1,10	

MICSTART (2)	※BIG固有命令 マイクからのサンプリング開始 ・事前にXON WIIUが必要 ・3DSの波形データとの互換性はない ・システム内のサンプリング用メモリーに記録 ・サンプリングレートは32000Hzに固定 ・サンプリングフォーマットは16bit符号付きに固定	
書式	MICSTART [デバイス指定,] 秒数	
引数	デバイス指定	0: GamePadの内蔵マイク/マイク端子 1: USB接続されたWiiUマイク ・GamePadとWiiUマイクを同時にサンプリングする事はできない
	秒数(0〜32)	0: ループモード 1〜32: サンプリング秒数 ・ループモード時、最大秒数を超えると先頭から上書き
例	XON WIIU MICSTART 0,4	

MICSTOP	マイクからのサンプリングを停止
書式	MICSTOP
例	MICSTOP

MICDATA	マイクからのデータ取得 指定位置のサンプリングデータを返す	
書式	変数=MICDATA(取得位置)	
引数	取得位置	・0〜(ビット数と最大秒数で範囲が決まる) ・ループモード時は範囲チェックは行わない
戻り	波形データ	・8ビット時の戻り値は128基準 ・16ビット時の戻りは32768基準
例	D=MICDATA(100)	

MICSAVE	内部サンプリングメモリーから配列へのコピー	
書式	MICSAVE [[取得位置,] 取得サンプル数,] 配列名	
引数	取得位置	取り込み始める位置(0〜)
	サンプル数	・取り込む数(省略時:サンプリングバッファ全体) ・MICSTARTで指定したサンプリングレート×秒数以上はエラー
	配列名	・取り込んだサンプリングデータの格納先 ・1次元配列の場合サンプル数が要素数を超えると自動拡張
例 (1)	XON MIC MICSTART 0,0,1 'rate:8180 bit:8 length:1sec DIM WAVE%[8180] 'MICSIZE MICSAVE 0,8180,WAVE%	
例 (2)	XON WIIU XON MIC MICSTART 0,1 'GamePad(32KHz,16bit) length:1sec DIM WAVE%[32000] 'MICSIZE MICSAVE 0,32000,WAVE%	

CONTROLLER	対象となるコントローラの情報取得 3DS版「プチコン3号」ではコントローラIDが0以外は常に0	
書式	数値=CONTROLLER([コントローラID])	
引数	コントローラID(0～4)	0: 3DS/GamePad 1～4: Wii互換コントローラ ・事前にXON WIIUが必要
戻り	|b00|GamePad/3DS(1) |b01|Wiiリモコン(2) |b02|プロコン(4) |b03|ジャイロ/リモコンプラス(8) |b04|ヌンチャク(16) |b05|クラコン/クラコンPRO(32) 返値が0の場合、該当IDのコントローラーは未接続。 b00-02は、接続されたコントローラーに応じて3bitのいずれか1bitがONになる。 b02-04は、b02が2の時のみORで合成される。 実際に取り得る戻り値のパターンは次の通り 0:未接続 1:GamePad/3DS 2:Wiiリモコン 4:プロコン 10:Wiiリモコンプラス(＋ジャイロ/リモコンプラス) 18:Wiiリモコン(＋ヌンチャク) 26:Wiiリモコンプラス(＋ジャイロ/リモコンプラス＋ヌンチャク) 34:Wiiリモコン(＋クラコン/クラコンPRO) 42:Wiiリモコンプラス(＋ジャイロ/リモコンプラス＋クラコン/クラコンPRO)	
例	V=CONTROLLER()	

VIBRATE	※BIG固有命令 対象となるコントローラを振動させる WiiU専用命令	
書式	VIBRATE コントローラID, 振動の強さ, 時間	
引数	コントローラID(0～4)	0: GamePad 1～4: Wii互換コントローラ
	振動の強さ(0～100)	
	時間(0～10秒)	・少数による指定可能
例	XON WIIU VIBRATE 0,50,2	

ファイル

一覧取得、ファイルへの読み書きなどの命令

FILES (1)	ファイル一覧をコンソール上に表示	
書式	FILES ["ファイル種別"]	
引数	ファイル種別	特定のファイル種別のみ表示させたい場合に以下を指定 "TXT:" テキストおよびプログラム "DAT:" バイナリデータ(グラフィック含む) "//" プロジェクトの一覧 "PROJECT/" プロジェクト名を指定
例	FILES	

FILES (2)	ファイル一覧を配列に取得	
書式	FILES ["ファイル種別",] 文字列配列	
引数	ファイル種別	特定のファイル種別のみ表示させたい場合に以下を指定 "TXT:" テキストおよびプログラム "DAT:" バイナリデータ(グラフィック含む) "//" プロジェクトの一覧 "PROJECT/" プロジェクト名を指定
	文字列配列	ファイル一覧のファイル名が格納される文字列配列 ・1次元配列の場合取得したファイル数に応じて自動拡張
例	DIM NAMETBL$[100] FILES NAMETBL$	

LOAD (1)	ファイルの読み込み ・確認用のダイアログが表示される ・実行中のプログラムSLOTにプログラムはロードできない	
書式	LOAD "[リソース名:]ファイル名"[,ダイアログ表示フラグ]	
引数	リソース名:	省略時: カレントプログラムSLOT PRG0～PRG3: プログラムSLOT (PRG=PRG0) GRP0～GRP5: グラフィックページ GRPF: フォント用の画像ページ
	ファイル名	読み込むファイル名
	ダイアログ表示フラグ	FALSE=確認用ダイアログを表示させない
例	LOAD "PROGNAME" LOAD "GRP0:GRPDATA"	

LOAD (2)	画像ファイルの読み込み ・確認用のダイアログが表示される ・オフセット指定時はみ出した部分は無視されます	
書式	LOAD "[GRPリソース名:]ファイル名"[OX,OY][,ダイアログ表示フラグ]	
引数	GRPリソース名:	GRP0～GRP5: グラフィックページ GRPF: フォント用の画像ページ
	ファイル名	読み込むファイル名
	OX,OY	画像上のオフセット(各0～511)
	ダイアログ表示フラグ	FALSE=確認用ダイアログを表示させない
例	LOAD "GRP0:GRPDATA",0,64	

LOAD (3)	テキストファイルを文字列変数に読み込む	
書式	LOAD "TXT:ファイル名"[,ダイアログ表示フラグ] OUT TX$	
引数	ファイル名	読み込むテキストファイル名(先頭に"TXT:"を付ける)
	ダイアログ表示フラグ	FALSE=確認用ダイアログを表示させない
戻り	TX$	読み込んだテキストファイルが格納される文字列変数
例	LOAD "TXT:MEMOFILE" OUT TX$	

LOAD (4)	テキストファイルを文字列変数に読み込む	
書式	文字列変数 = LOAD("TXT:ファイル名" [,ダイアログ表示フラグ])	
引数	ファイル名	読み込むテキストファイル名(先頭に"TXT:"を付ける)
	文字列変数	読み込んだテキストファイルが格納される文字列変数
	ダイアログ表示フラグ	FALSE=確認用ダイアログを表示させない
例	TX$=LOAD("TXT:MEMOFILE")	

LOAD (5)	バイナリファイルを数値配列に読み込む	
書式	LOAD "DAT:ファイル名", 数値配列[,ダイアログ表示フラグ]	
引数	ファイル名	読み込むバイナリファイル名(先頭に"DAT:"を付ける)
	数値配列	読み込んだバイナリファイルが格納される数値変数
	ダイアログ表示フラグ	FALSE=確認用ダイアログを表示させない
例	DIM MARRAY[100] LOAD "DAT:MDATA", MARRAY	

SAVE (1)	ファイルの保存 ・実行時、確認のダイアログが表示される ・SAVE用の確認ダイアログは非表示にできません	
書式	SAVE "[リソース名:]ファイル名"	
引数	リソース名:	省略時: カレントプログラムSLOT PRG0～PRG3: プログラムSLOT (PRG=PRG0) GRP0～GRP5: グラフィックページ GRPF: フォント用の画像ページ
	ファイル名	保存するファイルに付ける名前
例	SAVE "PRG0:TEST"	

SAVE (2)	文字列変数をテキストファイルに保存	
書式	SAVE "TXT:ファイル名", 文字列変数	
引数	ファイル名	保存するファイルに付ける名前(先頭に"TXT:"を付ける)
	文字列変数	保存するテキストデータの格納された文字列変数(UTF-8)
例	SAVE "TXT:MEMOFILE",TX$	

SAVE (3)	数値配列をバイナリファイルに保存	
書式	SAVE "DAT:ファイル名", 数値配列	
引数	ファイル名	保存するファイルに付ける名前(先頭に"DAT:"を付ける)
	数値配列	保存するデータの格納された数値配列
例	SAVE "DAT:TEST",MARRAY	

RENAME	ファイル名の変更 実行時、確認ダイアログを表示	
書式	RENAME "[ファイル種別:]ファイル名", "[ファイル種別:]新しい名前"	
引数	ファイル種別:	"TXT:" テキストおよびプログラム(省略可能) "DAT:" バイナリデータ(グラフィック含む)
	ファイル名	名前を変更するファイルのファイル名
	新しい名前	新しいファイル名
例	RENAME "SAMPLE","NEWNAME"	

DELETE	ファイルの消去 実行時、確認ダイアログを表示	
書式	DELETE "[ファイル種別:]ファイル名"	
引数	ファイル種別:	"TXT:" テキストおよびプログラム(省略可能) "DAT:" バイナリデータ(グラフィック含む)
例	DELETE "SAMPLE"	

EXEC (1)	プログラムのLOADと実行 ・EXECで実行したプログラムから元のプログラムに戻れない ・他のSLOTでEXECしたプログラムのENDで戻ることができる ・DIRECTモードでは実行できない	
書式	EXEC "[リソース名:]ファイル名"	
引数	リソース名:	PRG0～PRG3: 読み込み先プログラムSLOT
	ファイル名	読み込むプログラムのファイル名
例	EXEC "SAMPLE" EXEC "PRG0:SBGED"	

EXEC (2)	他のSLOTのプログラムを実行 ・EXECで実行したプログラムから元のプログラムに戻れない ・他のSLOTでEXECしたプログラムのENDで戻ることができる ・DIRECTモードでは実行できない	
書式	EXEC プログラムSLOT	
引数	プログラムSLOT	0～3: 実行するプログラムSLOT番号
例	EXEC 2	

USE	指定プログラムSLOTのプログラムを実行可能にする	
書式	USE プログラムSLOT	
引数	プログラムSLOT	0～3: プログラムSLOT
例	USE 2	

CHKFILE	指定したファイルが存在するかどうかチェック	
書式	変数 = CHKFILE("[ファイル種別:]ファイル名")	
引数	ファイル種別	"TXT:" テキストおよびプログラム "DAT:" バイナリデータ(グラフィック含む)
	ファイル名	調べるファイル名
戻り	TRUE=存在、FALSE=存在しない	
例	A=CHKFILE("SBATTACK")	

ワイヤレス通信

最大4台まで接続可能なワイヤレス通信に関する命令
※接続にはプチコン3号が入った接続数分の本体が必要です

MPSTART	ワイヤレス通信セッションの開始 ・セッションはMPSTARTの識別子が同じとき接続可能 ・セッション構築の確認はRESULTシステム変数で取得 ・スリープモードに入ると通信は切断される ・WiiUではワイヤレス命令はサポートされません	
書式	MPSTART 最大接続ユーザー数, "通信識別子文字列"	
引数	最大接続ユーザー数	2～4: 同時に接続するユーザーの数
	通信識別文字列	認証用の任意の文字列
例	MPSTART 4,"ANYSTR"	

MPEND	ワイヤレス通信セッションを終了 ・参加者全員が同期して終了 ・待ち合わせ用ダイアログが表示される ・WiiUではワイヤレス命令はサポートされません
書式	MPEND
例	MPEND

MPSEND	ワイヤレス通信セッション参加者全員へデータ送信 ・送信データは確実に配送されるが遅延が発生する ・短時間に大量のMPSENDを呼ぶとエラーが発生 ※Communication buffer overflow ・スリープモードに入ると通信は切断される ・WiiUではワイヤレス命令はサポートされません	
書式	MPSEND "送信文字列"	
引数	送信文字列	最大256バイトまでの文字列
例	MPSEND "HELLO!"	

MPRECV	MPSENDからのデータ受信 ・受信データがない場合は送信元IDに-1が入る ・スリープモードに入ると通信は切断される ・WiiUではワイヤレス命令はサポートされません	
書式	MPRECV OUT SID,RCV$	
引数	SID	0～3: 文字列を送信する接続先番号
	RCV$	受信されたデータを受け取る文字列変数
例	MPRECV OUT SID,RCV$ PRINT SID;":";RCV$	

MPSTAT	ワイヤレス通信の指定端末の接続状況を取得 ・スリープモードに入ると通信は切断される ・WiiUではワイヤレス命令はサポートされません	
書式	変数 = MPSTAT([端末ID])	
引数	端末ID	0～3: ワイヤレス通信による他の端末ID(省略時:セッション全体)
戻り	0:未接続、1:接続	
例	RET=MPSTAT(2)	

MPNAME$	ワイヤレス通信の指定端末の端末名を取得 ・スリープモードに入ると通信は切断される ・WiiUではワイヤレス命令はサポートされません	
書式	文字列変数 = MPNAME$(端末ID)	
引数	端末ID	0～3: ワイヤレス通信による他の端末ID
戻り	端末名文字列	
例	NAME$=MPNAME$(3)	

MPGET	ワイヤレス通信指定端末のユーザー定義データ取得 ・スリープモードに入ると通信は切断される ・WiiUではワイヤレス命令はサポートされません	
書式	変数=MPGET(端末ID, 内部管理番号)	
引数	端末ID	0～3: ワイヤレス通信による他の端末ID
	内部管理番号	0～8: 対象となるデータの管理番号
戻り	指定されたデータの数値(整数値)	
例	RET=MPGET(0, 5)	

MPSET	ワイヤレス通信のユーザー定義データ書き込み ・スリープモードに入ると通信は切断される ・WiiUではワイヤレス命令はサポートされません	
書式	MPSET 内部管理番号, 数値	
引数	内部管理番号	0～8: 対象となるデータの管理番号
	数値	登録する数値(整数値のみ受け付け)
例	MPSET 5,123	

スクリーン制御

画面の表示モードなどに関する命令

XSCREEN (1)	画面モードの設定 ・画面モード2,3はDIRECTモードでも使用できるが、実行後は下画面がキーボードに切り替わる ・3D指定はペアレンタルコントロールで無効にできる プチコンBIGの場合 ・3DS版との互換性を維持した画面モード ・立体視表示には対応していません ・TV側解像度480x240、GamePad側解像度320x240に固定 ・WiiUで追加された画面モード5,6については(2)以降に記載	
書式	XSCREEN 画面モード [,SPRITE割当数 ,BG割当数]	
引数	画面モード	プチコン3号の場合 0: 上-3D　下-未使用(デフォルト) 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) プチコンBIGの場合 0: TVのみ1画面(立体視非対応) 1: TVのみ1画面 2: TVとGamePadの2画面(立体視非対応) 3: TVとGamePadの2画面 4: TVとGamePadに解像度320x480の画面を同時表示
	SPRITE割当数	・上画面 / TV側に割り当てるSPRITE数: 0〜512 ・下画面 / GamePad側は512-上画面 / TV側のSP数
	BG割当数	・上画面 / TV側に割り当てるBGレイヤー枚数: 0〜4 ・下画面 / GamePad側は4-上画面 / TV側BGレイヤー数
例	XSCREEN 2,128,4	

XSCREEN (2)	※BIG固有命令 ・事前にXON WIIUで専用モードへの切替が必要 ・解像度指定が可能なTV側1画面モード	
書式	XSCREEN 5, TV解像度 [,SPRITE割当数 ,BG割当数]	
引数	TV解像度(0〜6)	0: 256x192 1: 320x200 2: 320x240 3: 400x240 4: 640x400 5: 640x480 6: 854x480
	SPRITE割当数	・割り当てるSPRITE数: 0〜4096
	BG割当数	・BGレイヤー枚数: 0〜4
例	XON WIIU XSCREEN 5,6,4096,4	

XSCREEN (3)	※BIG固有命令 ・事前にXON WIIUで専用モードへの切替が必要 ・解像度指定が可能なTVとGamePadの2画面モード	
書式	XSCREEN 6, TV解像度, GamePad解像度 [,SPRITE割当数 ,BG割当数]	
引数	TV解像度(0〜6)	0: 256x192 1: 320x200 2: 320x240 3: 400x240 4: 640x400 5: 640x480 6: 854x480
	GamePad解像度(0〜6)	・TV解像度の引数を参照
	SPRITE割当数	・割り当てるSPRITE数: 0〜4096 ・GamePad側のSPRITE数は4096-SPRITE割当数となる
	BG割当数	・BGレイヤー枚数: 0〜4 ・GamePad側のBGレイヤー数は4-BG割当数となる
例	XON WIIU XSCREEN 6,6,6,2048,2	

DISPLAY (1)	操作対象の上下画面を選択 ・DISPLAY 1は、XSCREEN 2または3のとき指定可 ・ダイレクトモードでは直接実行できない	
書式	DISPLAY 画面ID	
引数	画面ID	0:上画面 1:下画面
例	DISPLAY 0	

DISPLAY (2)	現在利用中の画面IDを取得 ・DISPLAY 1は、XSCREEN 2または3のとき指定可 ・ダイレクトモードでは直接実行できない
書式	変数=DISPLAY()
戻り	画面ID (0:上画面 1:下画面)
例	A=DISPLAY()

VISIBLE	画面表示要素のON/OFF切り替え	
書式	VISIBLE コンソール,グラフィック,BG,SPRITE	
引数	コンソール	0:非表示(#OFF)、1:表示(#ON)
	グラフィック	0:非表示(#OFF)、1:表示(#ON)
	BG	0:非表示(#OFF)、1:表示(#ON)
	SPRITE	0:非表示(#OFF)、1:表示(#ON)
例	VISIBLE 1,1,1,1	

BACKCOLOR (1)	背景色を指定	
書式	BACKCOLOR 背景色コード	
引数	背景色コード	・通常はRGB関数で指定　例)BACKCOLOR RGB(64,128,128) ・直接数値を指定するときは、RGB各8ビットの色コード
例	BACKCOLOR RGB(64,128,128)	

BACKCOLOR (2)	現在の背景色を取得
書式	変数=BACKCOLOR()
戻り	現在設定されている背景色の色コード
例	C=BACKCOLOR()

ACLS	描画設定をBASIC起動時の状態に戻す ・例のEND以降の処理に相当する動作を実行 ・BGMなどのサウンド設定には影響しません	
書式	ACLS [GR, SP, FN]	
引数	GR	GCLSとSPRITE画像とBG画像初期化(TRUE=しない)
	SP	SPDEF初期化(TRUE=しない)
	FN	フォント初期化(TRUE=しない)
例	ACLS END '--- XSCREEN 0 LOAD "GRP4:SYS/DEFSP.GRP" LOAD "GRP5:SYS/DEFBG.GRP" FONTDEF SPDEF DISPLAY 1 WIDTH 8 BACKCOLOR 0 FADE 0 COLOR 15,0:LOCATE 0,0,0:ATTR 0:CLS GPAGE 1,1:SPPAGE 4:BGPAGE 5 VISIBLE 1,1,1,1 DISPLAY 0 BACKCOLOR 0 FADE 0 WIDTH 8 COLOR 15,0:LOCATE 0,0,0:ATTR 0:CLS FOR I=0 TO 3:GPAGE I,I:GCLS 0:NEXT GPAGE 0,0:GPRIO 1024 SPPAGE 4:SPCLR BGPAGE 5:BGCLR VISIBLE 1,1,1,1	

FADE (1)	画面フェーダの色を設定 ・フェーダは常に最前面に描画 ・画面全体をフェード色(透明色込み)で塗りつぶす	
書式	FADE フェード色 [,時間]	
引数	フェード色	画面を覆う色(RGB(0,0,0,0)を設定すると無効となる)
	フェード時間	指定時間(1/60秒単位)で指定フェード色まで変化
例	FADE RGB(32,64,64,64),60	

FADE (2)	現在の画面フェーダの色を取得
書式	数値=FADE()
戻り	ARGB各8ビットの色コード
例	C=FADE()

FADECHK	フェードアニメーションの状態取得
書式	変数=FADECHK()
戻り	TRUE=アニメーション中、FALSE=停止
例	R=FADECHK()

グラフィック

ピクセル単位で線や円や矩形などを描画するための機能

GPAGE (1)	グラフィック表示ページと操作ページの指定	
書式	GPAGE 表示ページ，操作ページ	
引数	表示ページ	0～5: GRP0～GRP5
	操作ページ	0～5: GRP0～GRP5 ※初期状態はGRP4にSPRITE、GRP5にBG画像が存在
例	GPAGE 0,0	

GPAGE (2)	現在設定されているグラフィックページ情報取得	
書式	GPAGE OUT VP,WP	
引数	なし	
戻り	VP	表示用のページ番号(0～5)
	WP	操作用のページ番号(0～5)
例	GPAGE OUT WP,GP	

GCOLOR (1)	グラフィック描画色の指定	
書式	GCOLOR 色コード	
引数	色コード	・通常はRGB関数で指定　例)GCOLOR RGB(64,255,48) ・直接数値を指定するときはARGB各8ビットの色コード ・A(255:不透明、それ以外:透明)＋RGB各8ビット(0～255)
例	GCOLOR RGB(255,0,0)	

GCOLOR (2)	グラフィック描画色の指定	
書式	GCOLOR OUT C32	
引数	なし	
戻り	C32	ARGB各8ビットの色コード
例	GCOLOR OUT C32	

RGB	8ビットRGB値をもとに色コードを得る ・黒 RGB(0,0,0) ・白 RGB(255,255,255) ・薄いグレー RGB(224,224,224) ・グレー RGB(128,128,128) ・濃いグレー RGB(64,64,64) ・赤 RGB(255,0,0) ・ピンク RGB(255,96,208) ・紫 RGB(160,32,255) ・水色 RGB(80,208,255) ・青 RGB(0,32,255) ・黄緑 RGB(96,255,128) ・緑 RGB(0,192,0) ・黄色 RGB(255,224,32) ・オレンジ RGB(255,160,16) ・茶色 RGB(160,128,96) ・薄紅色 RGB(255,208,160)	
書式	変数 = RGB([透明度,] 赤要素,緑要素,青要素)	
引数	透明度	・透明度情報(255:不透明、それ以外:透明) ・SPCOLORでは0～255で透明度を指定可能
	赤・緑・青要素	各要素 8ビット諧調(各0～255)
戻り	変数=色コード(ARGB各8ビット)　※GCOLORを参照	
例	GPSET 0,0, RGB(255,255,0) 'YELLOW	

RGBREAD	色コードからRGBの各成分を得る	
書式	RGBREAD 色コード OUT [A,] R,G,B	
引数	色コード	ARGB各8ビットの色コード　※GCOLORを参照
戻り	A	透明度情報を受け取る変数(不透明:255～0:透明)
	R,G,B	8ビット色情報を受け取る変数(各0～255)
例	RGBREAD C OUT R,G,B	

GCLIP	グラフィック画面のクリッピング領域を指定 ・表示モードで範囲省略時、画面全体をクリッピング ・書き込みモードで範囲省略時、グラフィックページ全体
書式	GCLIP クリップモード [,始点X,始点Y,終点X, 終点Y]
引数	クリップモード：0:表示クリッピング、1:書き込みクリッピング 始点X,Y：クリップ領域の始点座標 終点X,Y：クリップ領域の終点座標
例	GCLIP 0,100,100,200,200

GPRIO	グラフィック画面の表示順位変更 3Dモードの場合グラフィック画面全体が影響を受ける
書式	GPRIO Z座標
引数	Z座標：奥行方向の座標(奥:1024<液晶面:0<手前:-256)
例	GPRIO -100

GCLS	グラフィック画面の消去 ・黒で画面全体を塗りつぶす命令 ・色コードを指定した塗りつぶしも可能
書式	GCLS [色コード]
引数	色コード：ARGB各8ビットの色コード　※GCOLORを参照
例	GCLS RGB(32,32,32)

GSPOIT	グラフィック画面の指定座標の色を取得 内部で色変換を経由するため描画時と同じ数値が返らない可能性があります
書式	変数 = GSPOIT(座標X,座標Y)
引数	座標X,Y：色を取得する座標(X:0～399、Y:0～239)
戻り	ARGB各8ビットの色コード　※GCOLORを参照
例	C=GSPOIT(100,100)

GPSET	グラフィック画面に点を打つ
書式	GPSET 座標X,座標Y [,色コード]
引数	座標X,Y：点を打つ座標 色コード：ARGB各8ビットの色コード　※GCOLORを参照
例	GPSET 100,50

GLINE	グラフィック画面に直線を引く
書式	GLINE 始点X,始点Y, 終点X,終点Y [,色コード]
引数	始点X,Y：始点座標(X:0～399、Y:0～239) 終点X,Y：終点座標(X:0～399、Y:0～239) 色コード：ARGB各8ビットの色コード　※GCOLORを参照
例	GLINE 0,0,399,239,RGB(0,255,255)

GCIRCLE (1)	グラフィック画面に円を描く
書式	GCIRCLE 中心点X,中心点Y, 半径 [,色コード]
引数	中心点X,Y：中心点座標(X:0～399、Y:0～239) 半径：円の半径(ドット) 1～ 色コード：ARGB各8ビットの色コード　※GCOLORを参照
例	GCIRCLE 200,120,30

GCIRCLE (2)	グラフィック画面に円弧を描く
書式	GCIRCLE 中心点X,中心点Y, 半径, 開始角, 終了角 [,フラグ [,色コード]]
引数	中心点X,Y：中心点座標(X:0～399、Y:0～239) 半径：円の半径(ドット) 1～ 開始角,終了角：円弧の角度0～360 フラグ：描画方法(0=円弧,1=扇形) 色コード：ARGB各8ビットの色コード　※GCOLORを参照
例	GCIRCLE 200,120,30, 0,45, 1

GBOX	グラフィック画面に四角形を描く
書式	GBOX 始点X,始点Y, 終点X,終点Y [,色コード]
引数	始点X,Y：始点座標(X:0～399、Y:0～239) 終点X,Y：終点座標(X:0～399、Y:0～239) 色コード：ARGB各8ビットの色コード　※GCOLORを参照
例	GBOX 0,0,399,239

GFILL	グラフィック画面に四角形を描いて塗りつぶす	
書式	GFILL 始点X,始点Y, 終点X,終点Y [,色コード]	
引数	始点X,Y	始点座標(X:0～399、Y:0～239)
	終点X,Y	終点座標(X:0～399、Y:0～239)
	色コード	ARGB各8ビットの色コード　※GCOLORを参照
例	GFILL 0,0,399,239	

GPAINT	グラフィック画面を塗りつぶす 境界色省略時、開始点座標にある色の範囲を塗る	
書式	GPAINT 開始点X, 開始点Y [,塗りつぶし色 [, 境界色]]	
引数	開始点X,Y	塗りつぶしを開始する座標(X:0～399、Y:0～239)
	塗りつぶし色	ARGB各8ビットの色コード　※GCOLORを参照
	境界色	指定方法は塗りつぶし色と同じ
例	GPAINT 200,120,RGB(255,0,0),RGB(0,0,0)	

GCOPY	他のグラフィックページから画像をコピー	
書式	GCOPY [転送元ページ,] 始点X,始点Y, 終点X,終点Y, 転送先X,転送先Y, コピーモード	
引数	転送元ページ	0～5(GRP0～GRP5)、-1(GRPF)　省略時:現在の描画ページ
	始点X,Y　終点X,Y	コピー元範囲の始点座標と終点座標(X:0～399、Y:0～239)
	転送先X,Y	コピー先範囲の始点座標(X:0～399、Y:0～239)
	コピーモード	TRUE=透明色をコピー、FALSE=透明色をコピーしない
例	GCOPY 0, 0,0,100,100, 200,100 ,1	

GSAVE	画像を配列へコピー(画面全体)	
書式	GSAVE [転送元ページ,] [X,Y,幅,高さ,] 転送先配列, 色変換フラグ	
引数	転送元ページ	0～5(GRP0～GRP5)、-1(GRPF)　省略時:現在の描画ページ
	X,Y,幅,高さ	コピー元範囲の始点X,Y座標,幅,高さ(ドット) 省略時:現在の描画領域
	転送先配列	画像を格納する配列変数 ※配列の要素が不足する場合、1次元配列に限り自動追加される
	色変換フラグ	0: 色変換する(32bitの論理色にする) 1: 物理コードのまま(16bit)
例	DIM WORK[0] GSAVE 0,0,0,512,512,WORK,1	

GLOAD (1)	画像データを配列からグラフィック画面にコピー	
書式	GLOAD [X,Y,幅,高さ,] 画像配列,色変換フラグ,コピーモード	
引数	X,Y,幅,高さ	コピー先範囲の始点X,Y座標,幅,高さ(ドット)
	画像配列	GSAVEによって画像データが格納された数値配列
	色変換フラグ	0: 色変換する(32bitの論理色にする) 1: 物理コードのまま(16bit)
	コピーモード	TRUE=透明色をコピー、FALSE=透明色をコピーしない
例	GLOAD 0,0,512,512, WORK, 1, 0	

GLOAD (2)	画像データを配列からグラフィック画面にコピー パレットを使ったインデックスカラー扱い	
書式	GLOAD [X,Y,幅,高さ,] 画像配列,パレット配列,コピーモード	
引数	X,Y,幅,高さ	コピー先範囲の始点X,Y座標,幅,高さ(ドット)
	画像配列	GSAVEによって画像データが格納された数値配列
	パレット配列	パレットデータが格納された数値配列
	コピーモード	TRUE=透明色をコピー、FALSE=透明色をコピーしない
例	GLOAD 0,0,512,512, WORK, PALETTE, 0	

GTRI	グラフィック画面に三角形を描いて塗りつぶす	
書式	GTRI X1,Y1, X2,Y2, X3,Y3 [,色コード]	
引数	X1,Y1	頂点1(X:0～399、Y:0～239)
	X2,Y2	頂点2(X:0～399、Y:0～239)
	X3,Y3	頂点3(X:0～399、Y:0～239)
	色コード	ARGB各8ビットの色コード　※GCOLORを参照
例	GTRI 200,10,300,200,100,200	

GPUTCHR (1)	グラフィック画面に文字を描く	
書式	GPUTCHR X,Y, "文字列" [,スケールX,スケールY][,色コード]	
引数	X,Y	表示位置(X:0～399、Y:0～239)
	"文字列"	表示する文字列
	スケールX,Y	表示倍率(等倍=1.0)
	色コード	ARGB各8ビットの色コード　※GCOLORを参照
例	GPUTCHR 10,10,"あいう"	

GPUTCHR (2)	グラフィック画面に文字を描く	
書式	GPUTCHR X,Y, 文字コード [,スケールX,スケールY][,色コード]	
引数	X,Y	表示位置(X:0～399、Y:0～239)
	文字コード	表示する文字コード
	スケールX,Y	表示倍率(等倍=1.0)
	色コード	ARGB各8ビットの色コード　※GCOLORを参照
例	GPUTCHR 10,10,ASC("A")	

GPUTCHR16 (1)	※BIG固有命令 グラフィック画面に16ドットの文字を描く WiiU専用命令	
書式	GPUTCHR16 X,Y, "文字列" [,スケールX,スケールY][,色コード]	
引数	X,Y	表示位置
	"文字列"	表示する文字列
	スケールX,Y	表示倍率(等倍=1.0)
	色コード	ARGB各8ビットの色コード　※GCOLORを参照
例	XON WIIU GPUTCHR16 10,10,"あいう"	

GPUTCHR16 (2)	※BIG固有命令 グラフィック画面に16ドットの文字を描く WiiU専用命令	
書式	GPUTCHR16 X,Y, "文字列" [,スケールX,スケールY][,色コード]	
引数	X,Y	表示位置
	文字コード	表示する文字コード
	スケールX,Y	表示倍率(等倍=1.0)
	色コード	ARGB各8ビットの色コード　※GCOLORを参照
例	XON WIIU GPUTCHR16 10,10,ASC("A")	

GOFS (1)	グラフィック画面の座標変更(移動)	
書式	GOFS X,Y	
引数	X,Y	グラフィック画面の表示オフセット
例	GOFS 50,80	

GOFS (2)	グラフィック画面のオフセット値取得	
書式	GOFS OUT X,Y	
戻り	X,Y	現在設定されているグラフィック画面の表示オフセット
例	GOFS OUT GX,GY	

SPRITE（スプライト）

自由に動かせる矩形単位の画像表示に関する機能

SPPAGE (1)	SPRITEに割り当てるグラフィックページの設定	
書式	SPPAGE グラフィックページ	
引数	グラフィックページ	0～5(GRP0～GRP5) 初期状態のSPRITE用ページは4(GRP4)
例	SPPAGE 4	

SPPAGE (2)	SPRITEに割り当てられたグラフィックページの取得
書式	変数=SPPAGE()
戻り	グラフィックページ番号(0～5)
例	P=SPPAGE()

SPCLIP	SPRITEのクリッピング領域を指定 ・範囲省略時画面全体	
書式	SPCLIP [始点X,始点Y,終点X, 終点Y]	
引数	始点X,Y	クリップ領域の始点座標(X:0～399、Y:0～239)
	終点X,Y	クリップ領域の終点座標(X:0～399、Y:0～239)
例	SPCLIP 100,100,200,200	

SPDEF (1)	SPRITEキャラクタ定義用テンプレートを初期状態に戻す
書式	SPDEF
引数	なし
SPDEF共通の補足	・SPRITEの定義用テンプレートは上下画面共通要素 ・SPSETの定義を簡略化するために用意されています
例	SPDEF

SPDEF (2)	SPRITEのキャラクタ定義用テンプレートを作成	
書式	SPDEF 定義番号, U,V [,W,H [,原点X,原点Y]] [,アトリビュート]	
引数	定義番号	テンプレートの定義番号: 0～4095
	U,V	定義する元画像の座標(U:0～511、V:0～511)
	W,H	定義する元画像サイズ 省略時16,16 ※U+WおよびV+Hの値が512を超えるとエラー
	原点X,Y	SPRITEの座標基準点　省略時0,0
	アトリビュート	|b00|表示(0=OFF、1=ON) #SPSHOW |b01|↑90度単位の回転(b01とb02の2ビットで指定) |b02|↓#SPROT0、#SPROT90、#SPROT0180、#SPROT270 |b03|横反転(0=OFF、1=ON)、#SPREVH |b04|縦反転(0=OFF、1=ON)、#SPREVV |b05|加算合成(0=OFF、1=ON)、#SPADD 省略時 0x01(表示のみON)
例	SPDEF 0,192,352,32,32,16,16,1	

SPDEF (3)	SPRITEのキャラクタ定義用テンプレートを配列から一括作成	
書式	SPDEF 数値配列 [,定義番号オフセット [,Uオフセット ,Vオフセット]]	
引数	数値配列	SPRITEのテンプレートデータが格納された数値配列 ・1個分の要素はU,V,W,H,原点X,原点Y,アトリビュートの7つ ・要素数は7の倍数である必要がある ・0から順に要素数/7までのSPRITEテンプレートが定義される
	定義番号オフセット	定義開始番号を指定:0～4095
	U,Vオフセット	画像の定義位置調整用として加算されます(各0～511)
例	SPDEF SRCDATA SPDEF SRCDATA 256, 0,256	

SPDEF (4)	SPRITEのキャラクタ定義用テンプレートをDATA列から一括作成	
書式	SPDEF "@ラベル文字列" [,定義番号オフセット [,Uオフセット ,Vオフセット]]	
引数	@ラベル文字列	SPRITEのテンプレートデータが列挙されたDATA命令のラベル ・@ラベル名は""でくくるか、文字列変数で指定する ・先頭データは定義するSPRITE数とし、続いて各SPRITEのデータを列挙(1個につき7データ) ・1個分のデータはU,V,W,H,原点X,原点Y,アトリビュートの7つ
	定義番号オフセット	定義開始番号を指定:0～4095
	U,Vオフセット	画像の定義位置調整用として加算されます(各0～511)
例	SPDEF "@SRCDATA" SPDEF "@SRCDATA", 256, 0,256	

SPDEF (5)	SPRITEのキャラクタ定義テンプレートの情報を得る	
書式	SPDEF 定義番号 OUT U,V [,W,H [,HX,HY]] [,A]	
引数	定義番号	テンプレートの定義番号: 0～4095
戻り	U,V	画像の座標を受け取る変数
	W,H	画像サイズを受け取る変数
	HX,HY	SPRITEの座標基準点を受け取る変数
	A	アトリビュートを受け取る変数
例	SPDEF 2 OUT U,V,ATR	

SPDEF (6)	SPRITEキャラクタ定義用テンプレートをコピー ・コピー不要な要素は省略可能(区切りの','カンマは必要) ・引数部分はコピーした上で調整する場合に利用	
書式	SPDEF 定義番号,元になる定義番号,[U],[V],[W],[H],[原点X],[原点Y],[アトリビュート]	
引数	定義番号	テンプレートの定義番号:0～4095
	ソース定義番号	コピー元になる定義番号:0～4095
	U,V	定義する元画像の座標(U:0～511、V:0～511)
	W,H	定義する元画像サイズ 省略時16,16 ※U+WおよびV+Hの値が512を超えるとエラー
	原点X,Y	SPRITEの座標基準点　省略時0,0
	アトリビュート	|b00|表示(0=OFF、1=ON) #SPSHOW |b01|↑90度単位の回転(b01とb02の2ビットで指定) |b02|↓#SPROT0、#SPROT90、#SPROT0180、#SPROT270 |b03|横反転(0=OFF、1=ON)、#SPREVH |b04|縦反転(0=OFF、1=ON)、#SPREVV |b05|加算合成(0=OFF、1=ON)、#SPADD 省略時 0x01(表示のみON)
例	SPDEF 0,255,192,352,32,32,16,16,1 SPDEF 1,255,,,32,32,,,	

SPSET (1)	SPRITE作成(定義テンプレートを利用) ・SPSETによりSPRITEは利用可能となる ・SPSET実行により回転等の情報は全て初期化 ・SPVARの値はすべて0になる ・SPHIT系衝突判定を使う時はSPSET後にSPCOLを呼びます	
書式	SPSET 管理番号,定義番号	
引数	管理番号	作成するSPRITEの番号: 0～511
	定義番号	SPDEFで定義したテンプレートの定義番号: 0～4095
例	SPSET 1,500	

SPSET (2)	SPRITE作成(直接画像情報等を指定) SPDEFの値を使わずに独自に設定する場合に利用 ・SPSETによりSPRITEは利用可能となる ・SPSET実行により回転等の情報は全て初期化 ・SPVARの値はすべて0になる ・SPHIT系衝突判定を使う時はSPSET後にSPCOLを呼びます	
書式	SPSET 管理番号 ,U,V [,W,H] ,アトリビュート	
引数	管理番号	作成するSPRITEの番号: 0～511
	U,V	定義する元画像の座標(U:0～511、V:0～511)
	W,H	定義する元画像サイズ(省略時16,16) ※U+WおよびV+Hの値が512を超えるとエラー
	アトリビュート	|b00|表示(0=OFF、1=ON) #SPSHOW |b01|↑90度単位の回転(b01とb02の2ビットで指定) |b02|↓#SPROT0、#SPROT90、#SPROT0180、#SPROT270 |b03|横反転(0=OFF、1=ON)、#SPREVH |b04|縦反転(0=OFF、1=ON)、#SPREVV |b05|加算合成(0=OFF、1=ON)、#SPADD 省略時 0x01(表示のみON)
例	SPSET 54,0,0,32,32,1	

SPSET (3)	SPRITEに空きを探して作成(定義テンプレート利用) SPRITE全体から空きを探す ・SPSETによりSPRITEは利用可能となる ・SPSET実行により回転等の情報は全て初期化 ・SPVARの値はすべて0になる ・SPHIT系衝突判定を使う時はSPSET後にSPCOLを呼びます	
書式	SPSET 定義番号 OUT IX	
引数	定義番号	SPDEFで定義したテンプレートの定義番号: 0～4095
戻り	IX	生成された番号を受け取る変数: 0～511(-1=空きなし)
例	SPSET 500 OUT IX	

SPSET (4)	SPRITEに空きを探して作成(直接画像情報等を指定) SPRITE全体から空きを探す ・SPSETによりSPRITEは利用可能となる ・SPSET実行により回転等の情報は全て初期化 ・SPVARの値はすべて0になる ・SPHIT系衝突判定を使う時はSPSET後にSPCOLを呼びます	
書式	SPSET U,V,W,H,アトリビュート OUT IX	
引数	U,V	定義する元画像の座標(U:0～511、V:0～511)
	W,H	定義する元画像サイズ(省略時16,16) ※U+WおよびV+Hの値が512を超えるとエラー
	アトリビュート	|b00|表示(0=OFF、1=ON) #SPSHOW |b01|↑90度単位の回転(b01とb02の2ビットで指定) |b02|↓#SPROT0、#SPROT90、#SPROT0180、#SPROT270 |b03|横反転(0=OFF、1=ON)、#SPREVH |b04|縦反転(0=OFF、1=ON)、#SPREVV |b05|加算合成(0=OFF、1=ON)、#SPADD
戻り	IX	生成された番号を受け取る変数: 0～511(-1=空きなし)
例	SPSET 0,0,32,32,1 OUT IX	

SPSET (5)	範囲内でSPRITEに空きを探して作成(定義テンプレート利用) 指定範囲内で空きを探す ・SPSETによりSPRITEは利用可能となる ・SPSET実行により回転等の情報は全て初期化 ・SPVARの値はすべて0になる ・SPHIT系衝突判定を使う時はSPSET後にSPCOLを呼びます	
書式	SPSET 上限,下限, 定義番号 OUT IX	
引数	上限,下限	空きを探す範囲(0～511)
	定義番号	SPDEFで定義したテンプレートの定義番号: 0～4095
戻り	IX	生成された番号を受け取る変数: 0～511(-1=空きなし)
例	SPSET 100,120, 500 OUT IX	

SPSET (6)	範囲内でSPRITEに空きを探して作成(直接画像情報等を指定) 指定範囲内で空きを探す ・SPSETによりSPRITEは利用可能となる ・SPSET実行により回転等の情報は全て初期化 ・SPVARの値はすべて0になる ・SPHIT系衝突判定を使う時はSPSET後にSPCOLを呼びます	
書式	SPSET 上限,下限, U,V,W,H,アトリビュート OUT IX	
引数	上限,下限	空きを探す範囲(0～511)
	U,V	定義する元画像の座標(U:0～511、V:0～511)
	W,H	定義する元画像サイズ(省略時16,16) ※U+WおよびV+Hの値が512を超えるとエラー
	アトリビュート	|b00|表示(0=OFF、1=ON) #SPSHOW |b01|↑90度単位の回転(b01とb02の2ビットで指定) |b02|↓#SPROT0、#SPROT90、#SPROT0180、#SPROT270 |b03|横反転(0=OFF、1=ON)、#SPREVH |b04|縦反転(0=OFF、1=ON)、#SPREVV |b05|加算合成(0=OFF、1=ON)、#SPADD 省略時 0x01(表示のみON)
戻り	IX	生成された番号を受け取る変数: 0～511(-1=空きなし)
例	SPSET 100,120, 0,0,32,32,1 OUT IX	

SPCLR	指定SPRITEの使用をやめてメモリーを解放 利用後に開放しないとSPSET用の空きがなくなる	
書式	SPCLR 管理番号	
引数	管理番号	使用をやめるSPRITEの管理番号: 0～511
例	SPCLR 56	

SPSHOW	SPRITEの表示を開始 SPSET前に使うとエラー	
書式	SPSHOW 管理番号	
引数	管理番号	表示するSPRITEの管理番号: 0〜511
例	SPSHOW 43	

SPHIDE	SPRITEの表示を隠す ・表示を隠すだけでありSPRITEは存在 ・SPSET前に使うとエラー	
書式	SPHIDE 管理番号	
引数	管理番号	表示を隠すSPRITEの管理番号: 0〜511
例	SPHIDE 43	

SPHOME (1)	SPRITEの座標基準点(ホーム位置)指定 ・SPOFS命令の位置基準点 ・回転やスケーリングの中心点 ・衝突判定の中心座標 ・SPSET前に使うとエラー	
書式	SPHOME 管理番号,位置X,位置Y	
引数	管理番号	基準点を設定するSPRITEの管理番号: 0〜511
	位置X,Y	SPRITEの左上を原点(0,0)とした相対座標
例	SPHOME 34,16,16	

SPHOME (2)	SPRITEの座標基準点(ホーム位置)取得 SPSET前に使うとエラー	
書式	SPHOME 管理番号 OUT HX,HY	
引数	管理番号	SPRITEの管理番号: 0〜511
戻り	HX,HY	基準点の座標を受け取る変数
例	SPHOME 10 OUT HX,HY	

SPOFS (1)	SPRITE座標の変更(移動) SPSET前に使うとエラー	
書式	SPOFS 管理番号, X, Y [,Z]	
引数	管理番号	対象のSPRITEの管理番号: 0〜511
	X,Y	SPRITEを表示する画面座標
	Z	奥行方向の座標(奥:1024<液晶面:0<手前:-256)
例	SPOFS 23,50,80 SPOFS 23,,,1000 SPOFS 23,150,180,0	

SPOFS (2)	SPRITEの座標を得る SPSET前に使うとエラー	
書式	SPOFS 管理番号 OUT X,Y[,Z]	
引数	管理番号	対象のSPRITEの管理番号: 0〜511
戻り	X,Y	座標を受け取る変数
	Z	奥行情報を受け取る変数
例	SPOFS 12 OUT X,Y,Z	

SPROT (1)	SPRITEの回転角度指定 SPSET前に使うとエラー	
書式	SPROT 管理番号,角度	
引数	管理番号	対象のSPRITEの管理番号: 0〜511
	角度	回転角度: 0〜360(時計回り)
例	SPROT 23,45	

SPROT (2)	SPRITEの回転角度を得る SPSET前に使うとエラー	
書式	SPROT 管理番号 OUT DR	
引数	管理番号	対象のSPRITEの管理番号: 0〜511
戻り	DR	角度を受け取る変数
例	SPROT 23 OUT DR	

SPROT (3)	SPRITEの回転角度を得る(関数タイプ) SPSET前に使うとエラー
書式	変数=SPROT(管理番号)
引数	管理番号 | 対象のSPRITEの管理番号: 0～511
戻り	現在の角度(0～360)
例	A=SPROT(23)

SPSCALE (1)	SPRITEのスケール(表示倍率)の変更 ・スケールを考慮した当り判定は先にSPCOLを実行 SPSET前に使うとエラー
書式	SPSCALE 管理番号, 倍率X, 倍率Y
引数	管理番号 | 対象のSPRITEの管理番号: 0～511
	倍率X,Y | 0.5(50%)～1.0(100%)～2.0(200%)～
例	SPSCALE 56, 0.75, 0.75

SPSCALE (2)	SPRITEの表示倍率を得る SPSET前に使うとエラー
書式	SPSCALE 管理番号 OUT SX,SY
引数	管理番号 | 対象のSPRITEの管理番号: 0～511
戻り	SX,SY | 倍率を受け取る変数
例	SPSCALE 45 OUT SX,SY

SPCOLOR (1)	SPRITEの表示色を設定 SPSET前に使うとエラー
書式	SPCOLOR 管理番号, 色コード
引数	管理番号 | 対象のSPRITEの管理番号: 0～511
	色コード | ARGB=8888形式の32ビット色コード ・Aの値を小さくすると透明度が上がる ・実際の表示色は色コードに元のドット色を乗算した値
例	SPCOLOR 1,RGB(16, 255,0,0) 'A=16,R=255,G=0,B=0

SPCOLOR (2)	SPRITEの表示色を得る SPSET前に使うとエラー
書式	SPCOLOR 管理番号 OUT C32
引数	管理番号 | 対象のSPRITEの管理番号: 0～511
戻り	C32 | 現在の色コードが返る変数(32ビットARGB)
例	SPCOLOR 1 OUT C

SPCHR (1)	SPRITEのキャラクタ定義を変更(テンプレート指定) SPSET前に使うとエラー
書式	SPCHR 管理番号, 定義番号
引数	管理番号 | 定義を変更するSPRITEの管理番号: 0～511
	定義番号 | SPDEF命令で登録したテンプレートの番号: 0～4095
例	SPCHR 0,500

SPCHR (2)	SPRITEのキャラクタ定義を変更(直接定義) ・管理番号以外の引数は省略可能 ・SPSET前に使うとエラー
書式	SPCHR 管理番号,[U],[V],[W],[H],[アトリビュート]
引数	管理番号 | 対象のSPRITEの管理番号: 0～511
	U,V | 定義する元画像の座標(U:0～511、V:0～511)
	W,H | 定義する元画像サイズ(省略時16,16) ※U+WおよびV+Hの値が512を超えるとエラー
	アトリビュート | |b00|無視(直前のSPHIDE/SPSHOWの状態を維持) |b01|↑90度単位の回転(b01とb02の2ビットで指定) |b02|↓#SPROT0、#SPROT90、#SPROT0180、#SPROT270 |b03|横反転(0=OFF、1=ON)、#SPREVH |b04|縦反転(0=OFF、1=ON)、#SPREVV |b05|加算合成(0=OFF、1=ON)、#SPADD 省略時 0x01(表示のみON)
例	SPCHR 5,64,64,16,16,1 SPCHR 6,,,32,32,1 'UV skip

SPCHR (3)	SPRITEのキャラクタ定義情報を得る SPSET前に使うとエラー	
書式	SPCHR 管理番号 OUT U,V [,W,H [,A]]	
引数	管理番号	対象のSPRITEの管理番号: 0～511
戻り	U,V	元画像の座標を格納する変数
	W,H	元画像のサイズを格納する変数
	A	アトリビュートを格納する変数
例	SPCHR 5 OUT U,V,W,H,ATR	

SPCHR (4)	SPRITEのキャラクタ定義番号を得る SPSET前に使うとエラー	
書式	SPCHR 管理番号 OUT DEFNO	
引数	管理番号	対象のSPRITEの管理番号: 0～511
戻り	DEFNO	定義番号を受け取る変数
例	SPCHR 5 OUT DEFNO	

SPLINK (1)	SPRITEを別のSPRITEにリンク ・リンクは座標のみ(回転角度や倍率情報は対象外) ・リンク先(親)指定は自分よりも小さい管理番号のみ ・子の表示座標は、親を基準とした相対座標となる ・この座標系は画面左上が原点とはならない ・リンクの階層に制限はない ・SPSET前に使うとエラー	
書式	SPLINK 管理番号, リンク先管理番号	
引数	管理番号	リンク元(子)のSPRITEの管理番号: 0～511
	リンク先管理番号	リンク先(親)のSPRITEの管理番号: 0～511 ※リンク元より小さい管理番号以外エラー
例	SPLINK 15,4	

SPLINK (2)	指定された管理番号SPRITEのリンク先番号を得る ・SPSET前に使うとエラー	
書式	変数=SPLINK(管理番号)	
引数	管理番号	リンク元(子)のSPRITEの管理番号: 0～511
戻り	リンク先の管理番号:0～511(-1の時リンク無し)	
例	V=SPLINK()	

SPUNLINK	SPRITEのリンクを解除 SPSET前に使うとエラー	
書式	SPUNLINK 管理番号	
引数	管理番号	リンクを解除するSPRITEの管理番号: 0～511
例	SPUNLINK 15	

SPANIM (1)	SPRITEによるアニメ表示(配列で指定) SPSET前に使うとエラー ・アニメは値を設定して指定時間分待つという動作 ・アニメ開始はSPANIMを実行した次フレームから ・対象要素ごとに最大32個のデータを受け付ける ・時間にマイナス値を指定すると直前の値から線形補間を行う	
書式	SPANIM 管理番号,"アニメ対象",データ配列 [,ループ]	
引数	管理番号	アニメーションを設定するSPRITEの管理番号: 0～511
	アニメ対象	変化させる要素を管理する数値または文字列 ・0または"XY": XY座標 ・1または "Z": Z座標 ・2または"UV": UV座標(定義元画像座標) ・3または "I": 定義番号 ・4または "R": 回転角度 ・5または "S": 倍率XY ・6または "C": 表示色 ・7または "V": 変数(SPRITE内部変数7の値) ・対象数値に8を加えると実行時からの相対指定 ・文字列の末尾に"+"を付けた場合も相対指定
	データ配列	アニメデータが格納された1次元数値配列
	ループ	ループ回数: (1～) 0で無限ループ
データ配列	アニメデータは数値配列に次の順で用意(最大32個まで) 時間1, 項目1,[項目2,] 時間2,項目1,[項目2,]…	
例	DIM PANIM[6] PANIM[0] = -60 'frame(-60=smooth) PANIM[1] = 200 'offset X,Y PANIM[2] = 100 PANIM[3] = -30 'frame PANIM[4] = 50 'offset PANIM[5] = 20 SPSET 0,0 SPANIM 0,"XY",PANIM	

SPANIM (2)	SPRITEによるアニメ表示(DATA命令で指定) SPSET前に使うとエラー ・アニメは値を設定して指定時間分待つという動作 ・アニメ開始はSPANIMを実行した次フレームから ・対照要素ごとに最大32個のデータを受け付ける ・時間にマイナス値を指定すると直前の値から線形補間を行う	
書式	SPANIM 管理番号,"アニメ対象","@ラベル文字列" [,ループ]	
引数	管理番号	アニメーションを設定するSPRITEの管理番号: 0～511
	アニメ対象	変化させる要素を管理する数値または文字列 ・0または"XY": XY座標 ・1または "Z": Z座標 ・2または"UV": UV座標(定義元画像座標) ・3または "I": 定義番号 ・4または "R": 回転角度 ・5または "S": 倍率XY ・6または "C": 表示色 ・7または "V": 変数(SPRITE内部変数7の値) ・対象数値に8を加えると実行時からの相対指定 ・文字列の末尾に"+"を付けた場合も相対指定
	@ラベル文字列	・アニメデータが格納されたDATA命令の先頭ラベル ・@ラベル名を""でくくって文字列として指定(または文字変数)
	ループ	ループ回数: (1～) 0で無限ループ
データ	アニメデータはDATA命令に次の順で用意 DATA キーフレーム数(最大32) DATA 時間1,項目1[,項目2] DATA 時間2,項目1[,項目2] :	
例	@MOVDATA DATA 2 'counter DATA -60,200,100 'frame,offset DATA -30,50,20 'frame,offset SPSET 0,0 SPANIM 0,"XY","@MOVDATA"	

SPANIM (3)	SPRITEによるアニメ表示(直接引数として指定) SPSET前に使うとエラー ・アニメは値を設定して指定時間分待つという動作 ・アニメ開始はSPANIMを実行した次フレームから ・対照要素ごとに最大32個のデータを受け付ける ・時間にマイナス値を指定すると直前の値から線形補間を行う	
書式	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]]… [,ループ]	
引数	管理番号	アニメーションを設定するSPRITEの管理番号: 0～511
	アニメ対象	変化させる要素を管理する数値または文字列 ・0または"XY": XY座標 ・1または "Z": Z座標 ・2または"UV": UV座標(定義元画像座標) ・3または "I": 定義番号 ・4または "R": 回転角度 ・5または "S": 倍率XY ・6または "C": 表示色 ・7または "V": 変数(SPRITE内部変数7の値) ・対象数値に8を加えると実行時からの相対指定 ・文字列の末尾に"+"を付けた場合も相対指定
	時間,項目	・アニメデータそのもの(必要な数分並べる、最大32個)
	ループ	ループ回数: (1～) 0で無限ループ
例	SPSET 0,0 SPANIM 0,"XY", -60,200,100, -30,50,20	

SPSTOP	SPRITEのアニメーションを停止 SPSET前に使うとエラー	
書式	SPSTOP [管理番号]	
引数	管理番号	対象のSPRITEの管理番号: 0～511 ※管理番号を省略すると全SPRITEのアニメーションを停止
例	SPSTOP	

SPSTART	SPRITEのアニメーションを開始する ※SPSETの前に使うとエラーとなります	
書式	SPSTART [管理番号]	
引数	管理番号	対象のSPRITEの管理番号: 0～511 ※管理番号を省略すると全SPRITEのアニメーションを開始
例	SPSTART	

SPCHK	SPRITEのアニメーション状態を取得 SPSET前に使うとエラー	
書式	変数=SPCHK(管理番号)	
引数	管理番号	対象のSPRITEの管理番号: 0～511
戻り	|b00|XY座標(1)、#CHKXY |b01|Z座標(2)、#CHKZ |b02|UV座標(4)、#CHKUV |b03|定義番号(8)、#CHKI |b04|回転(16)、#CHKR |b05|倍率XY(32)、#CHKS |b06|表示色(64)、#CHKC |b07|変数(128)、#CHKV ビットごとに対象割り当て(すべて0の時アニメ停止中)	
例	ST=SPCHK(5) '|b00|#CHKXY '|b01|#CHKZ '|b02|#CHKUV '|b03|#CHKI '|b04|#CHKR '|b05|#CHKS '|b06|#CHKC '|b07|#CHKV	

SPVAR (1)	SPRITE用内部変数への書き込み ・SPRITE用内部変数(8個ずつあるユーザー用変数) ・SPSET前でも利用可能(SPSET実行で8個とも0)	
書式	SPVAR 管理番号,内部変数番号,数値	
引数	管理番号	対象のSPRITE管理番号: 0～511
	内部変数番号	内部変数の番号: 0～7
	数値	内部変数に登録する数値(0～
例	SPVAR 0,7,1	

SPVAR (2)	SPRITE用内部変数の読み込み(関数型) ・SPRITE用内部変数(8個ずつあるユーザー用変数) ・SPSET前でも利用可能(SPSET実行で8個とも0)	
書式	変数=SPVAR(管理番号,内部変数番号)	
引数	管理番号	対象のSPRITE管理番号: 0～511
	内部変数番号	内部変数の番号: 0～7
戻り	SPVARで書き込んだ値	
例	V=SPVAR(54,0)	

SPVAR (3)	SPRITE用内部変数の読み込み ・SPRITE用内部変数(8個ずつあるユーザー用変数) ・SPSET前でも利用可能(SPSET実行で8個とも0)	
書式	SPVAR 管理番号,内部変数番号 OUT V	
引数	管理番号	対象のSPRITE管理番号: 0～511
	内部変数番号	内部変数の番号: 0～7
戻り	V	内部変数の値が戻る数値変数
例	SPVAR 54,0 OUT V	

SPCOL (1)	SPRITE衝突判定情報の設定 ・SPHIT系の命令を使う前に必ず呼ぶこと ・SPSET前に使うとエラー	
書式	SPCOL 管理番号 [,スケール対応]	
引数	管理番号	対象のSPRITE管理番号: 0～511
	スケール対応	FALSE=無視(省略時=FALSE) TRUE=SPSCALEに判定領域を同期 ※SPCOL命令後に設定したSPSCALEから有効
例	SPCOL 3,TRUE	

SPCOL (2)	SPRITE衝突判定情報の設定(マスク指定付) ・SPHIT系の命令を使う前に必ず呼ぶこと ・SPSET前に使うとエラー	
書式	SPCOL 管理番号,[スケール対応],マスク	
引数	管理番号	対象のSPRITE管理番号: 0～511
	スケール対応	FALSE=無視(省略時=FALSE) TRUE=SPSCALEに判定領域を同期 ※SPCOL命令後に設定したSPSCALEから有効
	マスク	0～&HFFFFFFFF(32ビット) ※衝突判定時に互いのビットのANDをとり、0であれば衝突していないとみなす (省略時&HFFFFFFFF)
例	SPCOL 3,TRUE,31 SPCOL 3,,31	

SPCOL (3)	SPRITE衝突判定情報の設定(範囲指定付) ・SPHIT系の命令を使う前に必ず呼ぶこと ・SPSET前に使うとエラー	
書式	SPCOL 管理番号,始点X,始点Y,幅,高さ,[スケール対応],マスク	
引数	管理番号	対象のSPRITE管理番号: 0～511
	始点X,Y	・判定領域の始点座標: X,Y(-32768～32767) ・SPHOMEを原点(0,0)とした相対座標
	幅,高さ	判定領域の幅と高さ: W,H(0～65535)
	スケール対応	FALSE=無視(省略時=FALSE) TRUE=SPSCALEに判定領域を同期 ※SPCOL命令後に設定したSPSCALEから有効
	マスク	0～&HFFFFFFFF(32ビット) ※衝突判定時に互いのビットのANDをとり、0であれば衝突していないとみなす (省略時&HFFFFFFFF)
例	SPCOL 3,0,0,32,32,TRUE,255 SPCOL 3,0,0,32,32,,255	

SPCOL (4)	SPRITE衝突判定情報取得(スケール対応とマスク) SPSET前に使うとエラー	
書式	SPCOL 管理番号 OUT スケール対応 [,マスク]	
引数	管理番号	対象のSPRITE管理番号: 0～511
戻り	スケール対応	スケール値を受け取る変数
	マスク	マスク値を受け取る変数
例	SPCOL 3 OUT SC,MSK	

SPCOL (5)	SPRITE衝突判定情報取得(範囲) SPSET前に使うとエラー	
書式	SPCOL 管理番号 OUT 始点X,始点Y,幅,高さ	
引数	管理番号	対象のSPRITE管理番号: 0～511
戻り	始点X,Y	判定領域の始点座標を受け取る変数
	幅,高さ	判定領域の幅と高さを受け取る変数
例	SPCOL 3 OUT X,Y,W,H	

SPCOL (6)	SPRITE衝突判定情報の取得(範囲とスケール対応) SPSET前に使うとエラー	
書式	SPCOL 管理番号 OUT 始点X,始点Y,幅,高さ,スケール対応	
引数	管理番号	対象のSPRITE管理番号: 0～511
戻り	始点X,Y	判定領域の始点座標を受け取る変数
	幅,高さ	判定領域の幅と高さを受け取る変数
	スケール対応	スケール値を受け取る変数
例	SPCOL 3 OUT X,Y,W,H,SC	

SPCOL (7)	SPRITE衝突判定情報の取得(すべて) SPSET前に使うとエラー	
書式	SPCOL 管理番号 OUT 始点X,始点Y,幅,高さ,スケール対応,マスク	
引数	管理番号	対象のSPRITE管理番号: 0～511
戻り	始点X,Y	判定領域の始点座標を受け取る変数
	幅,高さ	判定領域の幅と高さを受け取る変数
	スケール対応	スケール値を受け取る変数
	マスク	マスク値を受け取る変数
例	SPCOL 3 OUT X,Y,W,H,SC,MSK	

SPCOLVEC	SPRITE衝突判定用移動速度の設定 ・SPSET前に使うとエラー	
書式	SPCOLVEC 管理番号 [,移動量X,移動量Y]	
引数	管理番号	対象のSPRITE管理番号: 0～511
	移動量X,移動量Y	・省略時は次の要領で自動計算される ・SPANIMの"XY"を線形補完で実行中:前フレームからの移動距離 ・それ以外の場合:0,0
例	SPCOLVEC 93	

SPHITSP (1)	SPRITEの衝突判定 ・SPCOLを呼び出しておくこと ・SPSET前に使うとエラー	
書式	変数 = SPHITSP(管理番号 [,先頭ID,末尾ID])	
引数	管理番号	判定するSPRITEの管理番号: 0～511
	先頭ID,末尾ID	判定するSPRITEの範囲(0～511)
戻り	衝突したSPRITEの管理番号(衝突のないとき-1)	
例	H=SPHITSP(0)	

SPHITSP (2)	指定SPRITEとのSPRITE衝突判定 ・SPCOLを呼び出しておくこと ・SPSET前に使うとエラー	
書式	変数 = SPHITSP(管理番号 ,相手管理番号)	
引数	管理番号	判定するSPRITEの管理番号: 0～511
	相手管理番号	相手側のSPRITEの管理番号: 0～511
戻り	FALSE=衝突なし、TRUE=衝突	
例	H=SPHITSP(0,34)	

SPHITSP (3)	直前に設定した情報からSPRITEの衝突判定 ・引数を省略しないSPSETSPを呼び出しておくこと ・SPCOLを呼び出しておくこと ・SPSET前に使うとエラー
書式	変数 = SPHITSP()
引数	なし
戻り	衝突したSPRITEの管理番号(衝突のないとき-1)
例	H=SPHITSP()

SPHITRC (1)	動く四角形とすべてのSPRITEの衝突判定 ・SPCOLを呼び出しておくこと ・SPSET前に使うとエラー
書式	SPHITRC(始点X,始点Y,幅,高さ[,[マスク],移動量X,移動量Y])
引数	始点X,Y：判定元の四角形の左上座標 幅,高さ：判定元の四角形の幅と高さ マスク：0～&HFFFFFFFF(32ビット) ※衝突判定時に互いのビットのANDをとり、0であれば衝突していないとみなす (省略時&HFFFFFFFF) 移動量X,Y：判定元の四角形の移動量
戻り	衝突したSPRITEの管理番号(衝突のないとき-1)
例	H=SPHITRC(0,0,16,16)

SPHITRC (2)	指定したSPRITEと四角形の衝突判定 ・SPCOLを呼び出しておくこと ・SPSET前に使うとエラー
書式	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])
引数	管理番号：衝突相手のSPRITEの管理番号: 0～511 始点X,始点Y：判定元の四角形の左上座標 幅,高さ：判定元の四角形の幅と高さ マスク：0～&HFFFFFFFF(32ビット) ※衝突判定時に互いのビットのANDをとり、0であれば衝突していないとみなす (省略時&HFFFFFFFF) 移動量X,Y：判定元の四角形の移動量
戻り	FALSE=衝突なし、TRUE=衝突
例	H=SPHITRC(1,0,0,16,16)

SPHITRC (3)	指定範囲のSPRITEと四角形の衝突判定 ・SPCOLを呼び出しておくこと ・SPSET前に使うとエラー
書式	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])
引数	先頭ID,末尾ID：判定するSPRITEの範囲(0～511) 始点X,Y：判定元の四角形の左上座標 幅,高さ：判定元の四角形の幅と高さ マスク：0～&HFFFFFFFF(32ビット) ※衝突判定時に互いのビットのANDをとり、0であれば衝突していないとみなす (省略時&HFFFFFFFF) 移動量X,Y：判定元の四角形の移動量
戻り	衝突したSPRITEの管理番号(衝突のないとき-1)
例	H=SPHITRC(0,0,16,16)

SPHITRC (4)	直前に設定した情報からSPRITEの衝突判定 ・引数を省略しないSPSETRCを呼び出しておくこと ・SPCOLを呼び出しておくこと ・SPSET前に使うとエラー
書式	変数 = SPHITRC()
引数	なし
戻り	衝突したSPRITEの管理番号(衝突のないとき-1)
例	H=SPHITRC()

SPHITINFO (1)	衝突判定結果の情報取得(衝突時間) SPSET前に使うとエラー
書式	SPHITINFO OUT TM
引数	なし
戻り	TM：・衝突時間が戻る変数: 0～1の実数値 ・判定時の位置＋速度×衝突時間が衝突座標と一致
例	SPHITINFO OUT TM

SPHITINFO (2)	衝突判定結果の情報取得(衝突時間と座標) SPSET前に使うとエラー
書式	SPHITINFO OUT TM,X1,Y1,X2,Y2
引数	なし
戻り	TM：・衝突時間が戻る変数: 0～1の実数値 ・判定時の位置＋速度×衝突時間が衝突座標と一致 X1,Y1：衝突時の物体1の座標が戻る変数 X2,Y2：衝突時の物体2の座標が戻る変数
例	SPHITINFO OUT TM,X1,Y1,X2,Y2

SPHITINFO (3)	衝突判定結果の情報取得(衝突時間と座標と速度) SPSET前に使うとエラー	
書式	SPHITINFO OUT TM,X1,Y1,VX1,VY1,X2,Y2,VX2,VY2	
引数	なし	
戻り	衝突時間	・衝突時間が戻る変数: 0～1の実数値 ・判定時の位置＋速度×衝突時間が衝突座標と一致
	X1,Y1	衝突時の物体1の座標が戻る変数
	VX1,VY1	衝突時の物体1の速度が戻る変数
	X2,Y2	衝突時の物体2の座標が戻る変数
	VX2,VY2	衝突時の物体2の速度が戻る変数
例	SPHITINFO OUT TM,X1,Y1,VX1,VY1,X2,Y2,VX2,VY2	

SPFUNC	SPRITEごとに処理を割り当て ・コールバック処理が必要な上級者向けの命令 ・CALL SPRITE により全SPRITEの処理を実行 ・@ラベルの代わりにDEFで定義したユーザー処理も指定可能 ・処理先ではCALLIDXシステム変数で管理番号取得可能 ・SPSET前に使うとエラー	
書式	SPFUNC 管理番号, @ラベル	
引数	管理番号	対象のSPRITE管理番号: 0～511
	@ラベル	呼び出される処理先のラベル(またはユーザー定義処理)
例	SPFUNC 0,@PROG	

SPUSED	指定されたSPRITEが使われているか調査	
書式	変数=SPUSED(管理番号)	
引数	管理番号	対象のSPRITE管理番号: 0～511
戻り	TRUE=使用中、FALSE=空き	
例	S=SPUSED(4)	

BG

16x16ドットの矩形画像をタイル状に並べて表示する機能

BGPAGE (1)	BGに割り当てるグラフィックページの設定	
書式	BGPAGE グラフィックページ	
引数	グラフィックページ	0～5(GRP0～GRP5) 初期状態のBG用ページは5(GRP5)
例	BGPAGE 5	

BGPAGE (2)	BGに割り当てられたグラフィックページの取得
書式	変数=BGPAGE()
戻り	グラフィックページ番号(0～5)
例	P=BGPAGE()

BGSCREEN	BGスクリーンのサイズをレイヤーごとに設定	
書式	BGSCREEN レイヤー,幅,高さ	
引数	レイヤー	対象のレイヤー番号: 0～3
	幅,高さ	・キャラ単位の幅と高さ(幅×高さが16383以下まで) ・初期状態では25×15(上画面をBGで埋められるサイズ)
	キャラクターサイズ	キャラクターサイズ: 8/16/32(省略時16)
例	BGSCREEN 0,128,127	

BGCLR	BGスクリーンを消去	
書式	BGCLR [レイヤー]	
引数	レイヤー	対象のレイヤー番号: 0～3(省略時すべてのレイヤー)
例	BGCLR	

BGSHOW	BGスクリーンを表示	
書式	BGSHOW レイヤー	
引数	レイヤー	対象のレイヤー番号: 0～3
例	BGSHOW 0	

BGHIDE	BGスクリーンを非表示	
書式	BGHIDE レイヤー	
引数	レイヤー	対象のレイヤー番号: 0～3
例	BGHIDE 0	

BGCLIP	BGスクリーンの表示領域を指定	
書式	BGCLIP レイヤー [,始点X,始点Y,終点X,終点Y]	
引数	レイヤー	対象のレイヤー番号: 0～3
	始点X,Y	表示領域の始点ドット座標
	終点X,Y	・表示領域の終点ドット座標 ・始点と終点を省略するとレイヤー全体が表示領域となる
例	BGCLIP 0,20,20,379,219	

BGHOME (1)	レイヤーの表示原点設定 ・BGスクリーンに対する回転や拡大縮小の原点	
書式	BGHOME レイヤー,位置X,位置Y	
引数	レイヤー	対象のレイヤー番号: 0～3
	位置X,Y	ドット単位の原点座標
例	BGHOME 0,200,120	

BGHOME (2)	レイヤーの表示原点取得	
書式	BGHOME レイヤー OUT HX,HY	
引数	レイヤー	対象のレイヤー番号: 0～3
戻り	HX,HY	基準点の座標を受け取る変数
例	BGHOME 0 OUT HX,HY	

BGOFS (1)	BGスクリーンの表示オフセットを変更
書式	BGOFS レイヤー,X,Y,[Z]
引数	レイヤー：対象レイヤー番号：0～3
	X,Y：表示オフセットのドット座標
	Z：奥行方向の座標(奥:1024<液晶面:0<手前:-256)
例	BGOFS 0,-100,-100

BGOFS (2)	BGの座標を得る
書式	BGOFS レイヤー OUT X,Y[,Z]
引数	レイヤー：対象のレイヤー番号：0～3
戻り	X,Y：座標を受け取る変数
	Z：奥行情報を受け取る変数
例	BGOFS 0 OUT X,Y,Z

BGROT (1)	BGスクリーンの回転
書式	BGROT レイヤー,角度
引数	レイヤー：対象レイヤー番号：0～3
	角度：回転角(時計回り)：0～360
例	BGROT 0,180

BGROT (2)	BGスクリーンの回転情報取得
書式	BGROT レイヤー OUT R
引数	レイヤー：対象レイヤー番号：0～3
戻り	角度：R：0～360
例	BGROT 0 OUT R

BGSCALE (1)	BGスクリーンの拡大縮小 ・縮小時は全体で3600個分以上のBGは表示されません ・表示限界を超えるとBG画面が乱れます
書式	BGSCALE レイヤー,拡大率X,拡大率Y
引数	レイヤー：対象レイヤー番号：0～3
	拡大率X,Y：0.5(50%)～1.0(100%)～2.0(200%)～
例	BGSCALE 0,1.5,2.0

BGSCALE (2)	BGスクリーンの拡大縮小情報取得
書式	BGSCALE レイヤー OUT SX,SY
引数	レイヤー：対象レイヤー番号：0～3
戻り	SX,SY：0.5(50%)～1.0(100%)～2.0(200%)～
例	BGSCALE 0 OUT SX,SY

BGPUT	BGスクリーンへのBGキャラ配置 BGキャラクター番号0番は画像が表示されません
書式	BGPUT レイヤー,X,Y,スクリーンデータ
引数	レイヤー：対象レイヤー番号：0～3
	X,Y：配置先キャラ座標(0～BGSCREENで指定した値-1)
	スクリーンデータ： |b00|↑ | |キャラクター番号(0～4095、1024周期でくりかえし) |b11|↓ |b12|↑90度単位の回転(b12とb13の2ビットで指定) |b13|↓[00=0度、01=90度、10=180度、11=270度] |b14|横反転(0=OFF、1=ON) |b15|縦反転(0=OFF、1=ON) ・キャラクター番号や回転情報を16ビット数値化したもの ・16進数4桁の文字列指定も可能("0000"～"FFFF")
例	BGPUT 0,0,0,5 BGPUT 0,20,15,"80FF"

BGFILL	BGスクリーンをBGキャラで塗りつぶし	
書式	BGFILL レイヤー,始点X,始点Y,終点X,終点Y,スクリーンデータ	
引数	レイヤー	対象レイヤー番号: 0～3
	始点X,Y	始点座標(各 0～BGSCREEN命令で指定した値-1)
	終点X,Y	終点座標(各 0～BGSCREEN命令で指定した値-1)
	スクリーンデータ	b00↑ 　キャラクター番号(0～4095、1024周期でくりかえし) b11↓ b12↑90度単位の回転(b12とb13の2ビットで指定) b13↓[00=0度、01=90度、10=180度、11=270度] b14横反転(0=OFF、1=ON) b15縦反転(0=OFF、1=ON) ・キャラクター番号や回転情報を16ビット数値化したもの ・16進数4桁の文字列指定も可能("0000"～"FFFF")
例	BGFILL 0,0,0,19,15,1024 BGFILL 0,5,5,10,10,"C040"	

BGGET	BGスクリーンのBGキャラ情報を得る	
書式	変数=BGGET(レイヤー, X, Y [,座標系フラグ])	
引数	レイヤー	対象レイヤー番号: 0～3
	X,Y	BGキャラの取得座標(下記座標系フラグによって座標値が異なる)
	座標系フラグ (省略時0)	0: X,Y座標をBGスクリーン座標(キャラ単位)とする 1: X,Y座標を画面座標(ドット単位)とする
戻り	b00↑ 　キャラクター番号(0～4095、1024周期でくりかえし) b11↓ b12↑90度単位の回転(b12とb13の2ビットで指定) b13↓#BGROT0、#BGROT90、#BGROT0180、#BGROT270 b14横反転(0=OFF、1=ON)、#BGREVH b15縦反転(0=OFF、1=ON)、#BGREVV スクリーンデータ	
例	C=BGGET(0,12,14)	

BGANIM (1)	BGによるアニメ表示(配列で指定) ・アニメは値を設定して指定時間分待つという動作 ・アニメ開始はBGANIMを実行した次フレームから ・対象要素ごとに最大32個のデータを受け付ける ・時間にマイナス値を指定すると直前の値から線形補間を行う	
書式	BGANIM レイヤー,"アニメ対象",データ配列 [,ループ]	
引数	レイヤー	アニメーションを設定するレイヤー番号: 0～3
	アニメ対象	変化させる要素を管理する数値または文字列 ・0または"XY": XY座標 ・1または "Z": Z座標 ・4または "R": 回転角度 ・5または "S": 倍率XY ・6または "C": 表示色 ・7または "V": 変数(BG内部変数7の値) ・対象数値に8を加えると実行時からの相対指定 ・文字列の末尾に"+"を付けた場合も相対指定
	データ配列	アニメデータが格納された1次元数値配列
	ループ	ループ回数: (1～)　0で無限ループ
データ配列	アニメデータは数値配列に次の順で用意(最大32個まで) 時間1, 項目1,[項目2,] 時間2,項目1,[項目2,]…	
例	DIM PANIM[6] PANIM[0] = -60 'frame(-60=smooth) PANIM[1] = 200 'offset X,Y PANIM[2] = 100 PANIM[3] = -30 'frame PANIM[4] = 50 'offset PANIM[5] = 20 BGANIM 0,"XY",PANIM	

BGANIM (2)	BGによるアニメ表示(DATAで指定) ・アニメは値を設定して指定時間分待つという動作 ・アニメ開始はBGANIMを実行した次フレームから ・対象要素ごとに最大32個のデータを受け付ける ・時間にマイナス値を指定すると直前の値から線形補間を行う	
書式	BGANIM レイヤー,"アニメ対象","@ラベル文字列" [,ループ]	
引数	レイヤー	アニメーションを設定するレイヤー番号: 0～3
	アニメ対象	変化させる要素を管理する数値または文字列 ・0または"XY": XY座標 ・1または "Z": Z座標 ・4または "R": 回転角度 ・5または "S": 倍率XY ・6または "C": 表示色 ・7または "V": 変数(BG内部変数7の値) ・対象数値に8を加えると実行時からの相対指定 ・文字列の末尾に"+"を付けた場合も相対指定
	@ラベル文字列	・アニメデータが格納されたDATA命令の先頭ラベル ・@ラベル名を""でくくって文字列として指定(または文字変数)
	ループ	ループ回数: (1～)　0で無限ループ
データ	アニメデータはDATA命令に次の順で用意 DATA キーフレーム数(最大32) DATA 時間1,項目1[,項目2] DATA 時間2,項目1[,項目2] 　:	
例	@MOVDATA DATA 2 'counter DATA -60,200,100 'frame,offset DATA -30,50,20 'frame,offset BGANIM 0,"XY","@MOVDATA"	

BGANIM (3)	BGによるアニメ表示(直接引数として指定) ・アニメは値を設定して指定時間分待つという動作 ・アニメ開始はBGANIMを実行した次フレームから ・対象要素ごとに最大32個のデータを受け付ける ・時間にマイナス値を指定すると直前の値から線形補間を行う	
書式	BGANIM レイヤー,"アニメ対象",時間1,項目1[,項目2] [,時間2,項目1[,項目2]]… [,ループ]	
引数	レイヤー	アニメーションを設定するレイヤー番号: 0～3
	アニメ対象	変化させる要素を管理する数値または文字列 ・0または"XY": XY座標 ・1または "Z": Z座標 ・4または "R": 回転角度 ・5または "S": 倍率XY ・6または "C": 表示色 ・7または "V": 変数(BG内部変数7の値) ・対象数値に8を加えると実行時からの相対指定 ・文字列の末尾に"+"を付けた場合も相対指定
	時間,項目	・アニメデータそのもの(必要な数分並べる、最大32個)
	ループ	ループ回数: (1～)　0で無限ループ
例	BGANIM 0,"XY", -60,200,100, -30,50,20	

BGSTOP	BGのアニメーションを停止	
書式	BGSTOP [レイヤー]	
引数	レイヤー	対象のレイヤーの番号: 0～3 ※レイヤー省略時、全レイヤーのアニメーションを停止
例	BGSTOP	

BGSTART	BGのアニメーションを開始	
書式	BGSTART [レイヤー]	
引数	レイヤー	対象のレイヤーの番号: 0～3 ※レイヤー省略時、全レイヤーのアニメーションを開始
例	BGSTART	

BGCHK	BGのアニメーション状態を取得	
書式	変数=BGCHK(レイヤー)	
引数	レイヤー	調べるレイヤーの番号: 0〜3
戻り	|b00|XY座標(1)、#CHKXY |b01|Z座標(2)、#CHKZ |b02| |b03| |b04|回転(16)、#CHKR |b05|倍率XY(32)、#CHKS |b06|表示色(64)、#CHKC |b07|変数(128)、#CHKV ビットごとに対象割り当て(すべて0の時アニメ停止中)	
例	ST=BGCHK(0) '|b00|#CHKXY '|b01|#CHKZ '|b04|#CHKR '|b05|#CHKS '|b06|#CHKC '|b07|#CHKV	

BGVAR (1)	BG用内部変数への書き込み BG用レイヤーごとに8個ずつあるユーザー用変数	
書式	BGVAR レイヤー,内部変数番号,数値	
引数	レイヤー	対象のレイヤー番号: 0〜3
	内部変数番号	内部変数の番号: 0〜7
	数値	内部変数に登録する数値
例	BGVAR 0,7,1	

BGVAR (2)	BG用内部変数の読み込み(関数型) BG用レイヤーごとに8個ずつあるユーザー用変数	
書式	変数=BGVAR(レイヤー番号,内部変数番号)	
引数	レイヤー	対象のレイヤー番号: 0〜3
	内部変数番号	内部変数の番号: 0〜7
戻り	BGVARで書き込んだ値	
例	V=BGVAR(0,5)	

BGVAR (3)	BG用内部変数の読み込み ・BG用レイヤーごとに8個ずつあるユーザー用変数	
書式	BGVAR レイヤー,内部変数番号 OUT V	
引数	レイヤー	対象のレイヤー番号: 0〜3
	内部変数番号	内部変数の番号: 0〜7
戻り	V	内部変数の値が戻る数値変数
例	BGVAR 0,5 OUT V	

BGCOPY	BGスクリーンをキャラ単位でコピー	
書式	BGCOPY レイヤー,始点X,始点Y, 終点X,終点Y, 転送先X,転送先Y	
引数	レイヤー	対象レイヤー番号:0〜3
	始点X,Y 終点X,Y	コピー元の始点座標と終点座標(0〜BGSCREENで指定した値-1)
	転送先X,Y	コピー先の始点座標(0〜BGSCREENで指定した値-1)
例	BGCOPY 2,0,0,32,32,0,0	

BGLOAD (1)	配列からBGデータをBGスクリーンにコピー	
書式	BGLOAD レイヤー, [始点X,始点Y,幅,高さ,] 数値配列	
引数	レイヤー	コピー先範囲のレイヤー番号: 0〜3
	始点X,始点Y	コピー先始点座標(キャラ座標)
	幅, 高さ	・コピー先範囲の幅と高さ(キャラ単位) ・範囲指定を省略時、BGスクリーン全体が対象
	数値配列	BGSAVEによってBGデータが格納された数値配列
例	BGLOAD 0, 0,0,30,10, BGARRAY	

BGLOAD (2)	配列からBGデータをBGスクリーンにコピー	
書式	BGLOAD レイヤー, [始点X,始点Y,幅,高さ,] 数値配列, BGキャラ番号オフセット	
引数	レイヤー	コピー先範囲のレイヤー番号: 0〜3
	始点X,始点Y	コピー先始点座標(キャラ座標)
	幅, 高さ	・コピー先範囲の幅と高さ(キャラ単位) ・範囲指定を省略時、BGスクリーン全体が対象
	数値配列	BGSAVEによってBGデータが格納された数値配列
	BGキャラ番号オフセット	BGキャラ番号に加算してからBGSCREENへ転送される
例	BGLOAD 0, 0,0,30,10, BGARRAY, 256	

BGSAVE	BGスクリーンの内容を数値配列へコピー	
書式	BGSAVE レイヤー, [始点X,始点Y,幅,高さ,] 数値配列	
引数	レイヤー	コピー元のレイヤー番号: 0〜3
	始点X,始点Y	コピー元範囲の始点座標(キャラ座標)
	幅, 高さ	・コピー元範囲の幅と高さ(キャラ単位) ・範囲指定省略時、BGスクリーン全体が対象
	数値配列	・データがコピーされる数値配列 ・配列不足時は1次元配列に限り要素を自動拡張
例	DIM BGARRAY[30*10] BGSAVE 0, 0,0,30,10, BGARRAY	

BGCOORD	ディスプレイ座標とBGスクリーン座標との変換	
書式	BGCOORD レイヤー,元座標X,元座標Y[,モード]OUT DX,DY	
引数	レイヤー	レイヤー番号: 0〜3
	元座標X,Y	変換元座標(BGキャラ座標またはディスプレイ座標)
	モード	変換モード:0〜2 0: BGスクリーン座標からディスプレイ座標への変換 1: ディスプレイ座標からキャラ単位のBGスクリーン座標への変換 2: ディスプレイ座標からドット単位のBGスクリーン座標への変換
	DX,DY	変換後の座標格納先変数(BGキャラ座標またはディスプレイ座標)
例	BGCOORD 0,BGX,BGY,0 OUT DX,DY	

BGCOLOR (1)	BGの表示色を設定	
書式	BGCOLOR レイヤー, 色コード	
引数	管理番号	レイヤー番号: 0〜3
	色コード	・ARGB=8888形式の32ビット色コード ・RGB関数で指定すると便利 RGB(R,G,B) ・SPRITEとは異なりα値は無効(半透明表現は使えない) ・実際の表示色は、色コードに元のドット色を乗算したもの
例	BGCOLOR 1,RGB(255,0,0) 'R=255,G=0,B=0	

BGCOLOR (2)	BGの表示色を得る	
書式	BGCOLOR レイヤー OUT C32	
引数	レイヤー	レイヤー番号: 0〜3
戻り	C32	現在の色コードが返る変数(32ビットARGB)
例	BGCOLOR 1 OUT C	

BGFUNC	BGレイヤーごとに処理を割り当て ・コールバック処理が必要な上級者向けの命令 ・CALL BG により全BGレイヤーの処理を実行 ・@ラベルの代わりにDEFで定義したユーザー処理も指定可能 ・処理先ではCALLIDXシステム変数で管理番号取得可能	
書式	BGFUNC レイヤー, @ラベル	
引数	レイヤー	レイヤー番号: 0〜3
	@ラベル	呼び出される処理先のラベル(またはユーザー定義処理)
例	BGFUNC 0,@LAYERSUB0	

サウンド

音楽や効果音の演奏とエフェクター設定および音声合成

BEEP	単純な警告音・効果音の発生	
書式	BEEP [効果音番号][,周波数][,音量][,パンポット][,出力先（BIGのみ）]	
引数	効果音番号	・鳴らす音の種類: プリセット音0～133 ・SMILEボタンからプリセット音一覧を確認可能
	周波数	・周波数の変更値: -32768～32767(100で半音)
	音量	・再生する音量: 0～127
	パンポット	・ステレオのパンポット指定: 0(左)～64(中央)～127(右)
	出力先 (BIGのみ)	・WiiU専用の引数(3DS版「プチコン3号」では使えません) ・事前にXON WIIUが必要 ・複数の出力先の値をORする事で同時に発声可能 |b00|TV(1) |b01|GamePad(2) |b02|未割当(4) |b03|未割当(8) |b04|リモコン1(16) |b05|リモコン2(32) |b06|リモコン3(64) |b07|リモコン4(128)
例	BEEP 20	

BGMCHK	音楽の演奏状態調査	
書式	変数=BGMCHK([トラック番号])	
引数	トラック番号	トラック番号: 0～7(省略時は0番)
戻り	FALSE=停止中、TRUE=演奏中	
例	RET=BGMCHK(0)	

BGMCLEAR	ユーザー定義音楽の消去	
書式	BGMCLEAR [ユーザー定義曲番号]	
引数	ユーザー定義 曲番号	曲番号: 128～255(省略時すべての定義を消去)
例	BGMCLEAR	

BGMPLAY (1)	音楽演奏(登録済みBGMを再生) ・同時に8曲演奏可能(同時発音数は全体で16音まで) ・MMLを使った演奏方法については (2) をご覧ください	
書式	BGMPLAY [トラック番号,] 曲番号 [,音量][,出力先（BIGのみ）]	
引数	トラック番号	再生するトラック番号: 0～7(省略時は0番)
	曲番号	・プリセット曲(0～42) ・ユーザー定義(128～255) ・SMILEボタンからプリセット曲一覧を確認可能
	音量	再生する音量: 0～127
	出力先 (BIGのみ)	・WiiU専用の引数(3DS版「プチコン3号」では使えません) ・事前にXON WIIUが必要 ・複数の出力先の値をORする事で同時に発声可能 |b00|TV(1) |b01|GamePad(2) |b02|未割当(4) |b03|未割当(8) |b04|リモコン1(16) |b05|リモコン2(32) |b06|リモコン3(64) |b07|リモコン4(128)
例	BGMPLAY 0	

BGMPLAY (2)	音楽演奏(入力したMMLデータを再生) ・MMLによる再生はトラック0で行われる ・ユーザー定義曲番号255がMMLによる曲に書き換わる ・BGMPLAY直後に実行すると約2フレームの遅延が発生	
書式	BGMPLAY "MML文字列" [,出力先（BIGのみ）]	
引数	MML文字列	・"MML"でヘルプボタンを押すとコマンド説明を表示 ・以下の記号を並べることで演奏用文字列を登録 :0～:15　チャンネル指定 T1～T512　テンポ指定 CDEFGAB　音階(C#で半音上がり、C-で半音下がる) N0～N127　キーを数値指定(O4C=60) 1～192　音長の個別指定(C1=全音符、C4.=付点四分音符) L1～L192　デフォルト音長(付点は.を付ける) R　休符 O0～O8　オクターブ数値指定 < >　1オクターブ上げる、下げる V0～V127　音量の数値指定 ( )　音量上げる、下げる @0～@255　音色変更(0～127:GM相当、224～ユーザー波形) P0～P127　パンポット(左:P0～63 中:P64 右:P65～127) [　リピート開始]回数　リピート終了(回数省略時は無限ループ) &　前後の音をつなぐ _　ポルタメント
	出力先 (BIGのみ)	・WiiU専用の引数(3DS版「プチコン3号」では使えません) ・複数の出力先の値をORする事で同時に発声可能 |b00|TV(1) |b01|GamePad(2) |b02|未割当(4) |b03|未割当(8) |b04|リモコン1(16) |b05|リモコン2(32) |b06|リモコン3(64) |b07|リモコン4(128)
例	BGMPLAY "T120O4L4CC8D8EE8F8GA8G8E2"	

BGMSET	ユーザー定義音楽の事前定義 BGMPLAY直後に実行すると約2フレームの遅延が発生	
書式	BGMSET ユーザー定義曲番号,"MML文字列"	
引数	ユーザー定義曲番号	ユーザー定義曲番号: 128～255
MML文字列	"MML"でヘルプボタンを押すとコマンド説明を表示	
例	BGMSET 128,"CDEFG"	

BGMSETD	ユーザー定義曲の事前定義 ・DATA命令を利用してMMLを内部に登録(DATA "CDEFGAB") ・DATAの終端は数値で判断(DATA 0) ・内部的にはRESTOREと同じ扱い ・BGMSETD後にREADする場合はRESTOREを使うこと ・BGMPLAY直後に実行すると約2フレームの遅延が発生	
書式	BGMSETD ユーザー定義曲番号,"@ラベル文字列"	
引数	ユーザー定義曲番号	ユーザー定義曲番号: 128～255
	@ラベル文字列	・DATAでMML文字列が登録されたラベルの文字列 ・""でくくるか、文字列変数に代入して指定 ・"MML"でヘルプボタンを押すとコマンド説明を表示
例	BGMSETD 128,"@MMLTOP"	

BGMVAR (1)	MMLの内部変数への書き込み	
書式	BGMVAR トラック番号, 変数番号, 値	
引数	トラック番号	対象のMMLのトラック番号: 0～7
	変数番号	値を書き込む内部変数: 0～7(MMLの$0～$7)
	値	変数に書き込む値
例	BGMVAR 0,5,10	

BGMVAR (2)	MMLの内部変数の読み込み	
書式	変数=BGMVAR(トラック番号, 変数番号)	
引数	トラック番号	対象のMMLのトラック番号: 0～7
	変数番号	値を読み込む内部変数: 0～7(MMLの$0～$7)
戻り	演奏中の指定された変数の内容(演奏停止中は-1)	
例	MC=BGMVAR(0,5)	

BGMSTOP (1)	音楽演奏停止	
書式	BGMSTOP [トラック番号 [,フェード時間]]	
引数	トラック番号	対象のトラック番号: 0～7(省略時は全トラックを停止)
	フェード時間	秒(小数指定可能、0=即時停止、省略時0扱い)
例	BGMSTOP	

BGMSTOP (2)	音楽演奏停止 ・鳴り続けるリリース音などの強制的な停止 ・実行するとユーザー定義BGMの255番を上書き
書式	BGMSTOP -1
引数	-1 : 強制的に音を止めるための値
例	BGMSTOP -1

BGMVOL	指定トラックの音量を設定	
書式	BGMVOL [トラック番号,] 音量	
引数	トラック番号	対象のトラック番号: 0～7(省略時は0)
	音量	設定する音量: 0～127
例	BGMVOL 0,64	

BGMPAUSE (1)	音楽演奏一時停止(再開にはBGMCONTを利用)	
書式	BGMPAUSE [トラック番号 [,フェード時間]]	
引数	トラック番号	対象のトラック番号: 0～7(省略時は全トラックを一時停止)
	フェード時間	秒(小数指定可能、0=即時一時停止、省略時0扱い)
例	BGMPAUSE	

BGMPAUSE (2)	音楽演奏一時停止状態の確認
書式	BGMPAUSE([トラック番号])
引数	対象のトラック番号: 0～7(省略時は0)
例	A=BGMPAUSE()

BGMCONT	一時停止中の音楽演奏を再開(一時停止にはBGMPAUSEを利用)	
書式	BGMCONT [トラック番号 [,フェード時間]]	
引数	トラック番号	対象のトラック番号: 0～7(省略時は全トラックを再開)
	フェード時間	秒(小数指定可能、0=即時一時停止、省略時0扱い)
例	BGMCONT	

WAVSET	MMLのユーザー定義楽器音を定義	
書式	WAVSET 定義番号,A,D,S,R,"波形文字列" [,基準音程]	
引数	定義番号	・ユーザー定義楽器番号: 224～255 ・MMLの@コマンドで指定する番号
	A,D,S,R	エンベロープ定義パラメータ A: アタック(0～127) D: ディケイ(0～127) S: サスティン(0～127) R: リリース(0～127)
	波形文字列	・16進数文字列 ・2文字で1サンプルの値(8bit)を表す ・&H00 ～ &H80(128) ～ &HFF(255) ・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
	基準音程	省略時は69(O4A)
例	W$="7F7F7F7FFFFFFFFF7F7F7F7FFFFFFFFF"*4 WAVSET 224,3,10,30,5,W$,69	

WAVSETA	MMLのユーザー定義楽器音を配列から定義 ・MICSAVEで得た配列から定義するときに利用 ・サンプリングレート8180Hz、8ビット固定	
書式	WAVSETA 定義番号,A,D,S,R,数値配列 [,基準音程][,先頭添字][,最終添字]	
引数	定義番号	・ユーザー定義楽器番号: 224～255 ・MMLの@コマンドで指定する番号
	A,D,S,R	エンベロープ定義パラメータ A: アタック(0～127) D: ディケイ(0～127) S: サスティン(0～127) R: リリース(0～127)
	数値配列	MICSAVE命令で得た配列(最大16384サンプルまで)
	基準音程	省略時は69(O4A)
	先頭添字	数値配列の読み込み開始要素の添字(省略時は0)
	最終添字	数値配列の読み込み終了要素の添字(省略時は最終要素)
例	WAVSETA 224,0,95,100,20,SMPDATA	

EFCOFF	エフェクター設定をOFFにする
書式	EFCOFF
例	EFCOFF

EFCON	エフェクター設定をONにする エフェクトの種類はEFCSET命令で選択
書式	EFCON
例	EFCON

EFCSET (1)	音楽のエフェクトの種類を選択	
書式	EFCSET 種類番号	
引数	種類番号	0: なし(EFCOFFと同じ) 1: リバーブ(風呂場) 2: リバーブ(洞窟) 3: リバーブ(宇宙)
例	EFCSET 2	

EFCSET (2)	上級者向けエフェクトパラメータの設定	
書式	EFCSET 初期反射時間,残響音ディレイ時間,残響音減衰時間,残響音フィルタ係数1,残響音フィルタ係数2,初期反射音ゲイン,残響音ゲイン	
引数	初期反射時間	0～2000(msec)
	残響音ディレイ時間	0～2000(msec)
	残響音減衰時間	1～10000(msec)
	残響音フィルタ係数1	0.0～1.0
	残響音フィルタ係数2	0.0～1.0
	初期反射音ゲイン	0.0～1.0
	残響音ゲイン	0.0～1.0
例	EFCSET 997,113,1265,0.1,0,0.2,0.1	

EFCWET	BEEP、BGM、TALKそれぞれのエフェクト量を設定	
書式	EFCWET BEEP効果値, BGM効果値, TALK効果値	
引数	BEEP効果値	BEEPに対するエフェクトの大きさ(0～127)
	BGM効果値	BGMに対するエフェクトの大きさ(0～127)
	TALK効果値	・TALKのエフェクト設定(64未満:OFF、64以上:ON) ・TALKに対してはON/OFFのみで量は変化しない
例	EFCWET 0,100,64	

TALK	音声合成による発声 英数字記号はそのまま文字として読み上げます	
書式	TALK "音声文字列" [,出力先（BIGのみ）]	
引数	音声文字列	発音させたい文字列(文字をそのまま発音)
	特殊なコマンド	文字列内に<>で囲んだ特殊コマンドが利用可能 <S速さ>：喋る速さ(速さ0～65535、デフォルト32768) <P高さ>：音の高さ(高さ0～65535、デフォルト32768) ※指定範囲を超えてもエラーになりませんが限界値に補正されます
	出力先 (BIGのみ)	・WiiU専用の引数(3DS版「プチコン3号」では使えません) ・事前にXON WIIUが必要 ・複数の出力先の値をORする事で同時に発声可能 |b00|TV(1) |b01|GamePad(2) |b02|未割当(4) |b03|未割当(8) |b04|リモコン1(16) |b05|リモコン2(32) |b06|リモコン3(64) |b07|リモコン4(128)
例	TALK "コンニチハ<P50000><S20000>タカクテユックリ"	

TALKCHK	音声合成の状態調査
書式	変数=TALKCHK()
戻り	FALSE=停止中、TRUE=再生中
例	RET=TALKCHK()

TALKSTOP	再生中の音声を停止
書式	TALKSTOP
例	TALKSTOP

SNDSTOP	すべての音の発声を停止
書式	SNDSTOP
例	'BGM/BEEP/TALK/EFFECT/WAVE SNDSTOP

CHKMML	MML文字列の内容が正常にMMLとして解釈できるかどうかを返す	
書式	変数=CHKMML("MML文字列")	
引数	MML文字列	・MML文字列の詳細は"MML"でヘルプボタンを押して確認できます
戻り	-1:正常、0～:エラー（解釈エラーになった文字位置）	
例	R=CHKMML("T120O4L4CC8D8EE8F8GA8G8E2") PRINT R	

数学

三角関数や対数などの数学系の命令

FLOOR	整数部を取り出す(小数部切り捨て) ・その数を超えない最大の整数を得る ・FLOOR(12.5)は12、FLOOR(-12.5)は-13となる
書式	変数 = FLOOR(数値)
引数	数値 | 元になる数値
戻り	小数部を切り捨てられた整数値
関連	ROUND:四捨五入、CEIL:切り上げ
例	A=FLOOR(12.345)

ROUND	整数部を取り出す(小数部四捨五入)
書式	変数 = ROUND(数値)
引数	数値 | 元になる数値
戻り	小数部を四捨五入された整数値
関連	FLOOR:切り捨て、CEIL:切り上げ
例	A=ROUND(12.345)

CEIL	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 ・CEIL(12.5)は13、CEIL(-12.5)は-12となる
書式	変数 = CEIL(数値)
引数	数値 | 元になる数値
戻り	小数部を切り上げられた整数値
関連	ROUND:四捨五入、FLOOR:切り捨て
例	A=CEIL(12.345)

ABS	絶対値を得る
書式	変数 = ABS(数値)
引数	数値 | 絶対値を得る数値
戻り	絶対値
例	A=ABS(-12.345)

SGN	符号取得
書式	変数 = SGN(数値)
引数	数値 | 符号を得る数値
戻り	0または、±1
例	A=SGN(12.345)

MIN (1)	指定された数値配列内の一番小さい値を得る
書式	変数 = MIN(数値配列)
引数	数値配列 | 複数の数値の格納された数値配列名
戻り	渡された引数の中で一番小さい数
例	DIM TMP[2] TMP[0]=50:TMP[1]=3 A=MIN(TMP)

MIN (2)	指定された複数の数値から一番小さい値を得る
書式	変数 = MIN(数値1, 数値2 [,数値3 ...])
引数	数値を直接列挙 | カンマで区切って複数の数値を列挙
戻り	渡された引数の中で一番小さい数
例	A=MIN(1,2,3,4)

MAX (1)	指定された数値配列内の一番大きい値を得る	
書式	変数 = MAX(数値配列)	
引数	数値配列	複数の数値の格納された数値配列名
戻り	渡された引数の中で一番大きい数	
例	DIM TMP[2] TMP[0]=50:TMP[1]=3 A=MAX(TMP)	

MAX (2)	指定された複数の数値から一番大きい値を得る	
書式	変数 = MAX(数値1, 数値2 [,数値3 ...])	
引数	数値を直接列挙	カンマで区切って複数の数値を列挙
戻り	渡された引数の中で一番大きい数	
例	A=MAX(1,2,3,4)	

RND	整数の乱数を得る(0～最大値-1まで)	
書式	変数 = RND([シードID,] 最大値)	
引数	シードID	乱数の系列: 0～7
	最大値	取得する乱数の上限
戻り	0～最大値-1までのランダムな整数	
例	A=RND(100)	

RNDF	実数型の乱数を得る(0以上 1.0未満の実数乱数)	
書式	変数 = RNDF([シードID])	
引数	シードID	乱数の系列: 0～7
戻り	0以上1未満のランダムな実数	
例	A=RNDF()	

RANDOMIZE	乱数系列の初期化	
書式	RANDOMIZE シードID [, シード値]	
引数	シードID	乱数系列の種類: 0～7
	シード値	0または省略時、利用できるエントロピー情報を用いて初期化
例	RANDOMIZE 0	

SQR	正の平方根を求める	
書式	変数 = SQR(数値)	
引数	数値	平方根を求める数値
戻り	求めた正の平方根	
例	A=SQR(4)	

EXP	e(自然対数の底)のべき乗を求める	
書式	変数 = EXP([数値])	
引数	数値	指数(※省略時eを返す)
戻り	求めた結果	
例	A=EXP(2)	

LOG	対数を求める	
書式	変数 = LOG(数値 [,底])	
引数	数値	真数
	底	底(※省略時、自然対数を求める)
戻り	求めた結果	
例	A=LOG(2,2)	

POW	べき乗を求める	
書式	変数 = POW(数値, 乗数)	
引数	数値	べき乗を求める数値
	乗数	べき乗の乗数
戻り	求めた結果	
例	A=POW(1,4)	

PI	円周率を得る
書式	変数 = PI()
戻り	円周率の値(3.14159265)
例	A=PI()

RAD	度からラジアンを求める	
書式	変数 = RAD(数値)	
引数	数値	度: 0～360
戻り	度から求めたラジアン	
例	R=RAD(45)	

DEG	ラジアンから度を求める	
書式	変数 = DEG(数値)	
引数	数値	ラジアン
戻り	ラジアンから求めた度	
例	A=DEG(0.5*PI())	

SIN	サイン値を返す	
書式	変数 = SIN(角度)	
引数	角度	ラジアン
戻り	求めた値	
例	A=SIN(RAD(45))	

COS	コサイン値を返す	
書式	変数 = COS(角度)	
引数	角度	ラジアン
戻り	求めた値	
例	A=COS(RAD(45))	

TAN	タンジェント値を返す	
書式	変数 = TAN(角度)	
引数	角度	ラジアン
戻り	求めた値	
例	A=TAN(RAD(45))	

ASIN	アークサイン値を返す	
書式	変数 = ASIN(数値)	
引数	数値	-1.0～1.0
戻り	求めたアークサイン(ラジアン)	
例	A=ASIN(0)	

ACOS	アークコサイン値を返す	
書式	変数 = ACOS(数値)	
引数	数値	-1.0～1.0
戻り	求めたアークコサイン(ラジアン)	
例	A=ACOS(1)	

ATAN (1)	アークタンジェント値を返す(数値から)	
書式	変数 = ATAN(数値)	
引数	数値	角度を求める数値
戻り	求めたアークタンジェント(ラジアン)	
例	A=ATAN(1)	

ATAN (2)	アークタンジェント値を返す(XY座標から)	
書式	変数 = ATAN(座標Y,座標X)	
引数	座標Y,X	・原点からのX,Y座標値 ・入力値はY座標が先
戻り	求めたアークタンジェント(ラジアン)	
例	A=ATAN(1,1)	

SINH	ハイパボリックサイン値を返す
書式	変数 = SINH(数値)
引数	数値 | ハイパボリックサインを求める実数
戻り	求めたハイパボリックサイン
例	A=SINH(1)

COSH	ハイパボリックコサイン値を返す
書式	変数 = COSH(数値)
引数	数値 | ハイパボリックコサインを求める実数
戻り	求めたハイパボリックコサイン
例	A=COSH(1)

TANH	ハイパボリックタンジェント値を返す
書式	変数 = TANH(数値)
引数	数値 | ハイパボリックタンジェントを求める実数
戻り	求めたハイパボリックタンジェント
例	A=TANH(0.5)

CLASSIFY	通常数値、無限大、非数（NaN）の判定
書式	変数 = CLASSIFY(数値)
引数	数値 | チェックしたい実数
戻り	0=通常数値、1=無限大、2=NaN
例	A=CLASSIFY(0.5)

文字列操作

文字列の表示書式指定や抽出などの命令

ASC	指定された文字(または文字列変数)の文字コード取得	
書式	変数 = ASC("文字")	
引数	文字	調べたい文字が入った文字列(または文字列変数)
戻り	指定された文字の文字コード(UTF-16)	
例	A=ASC("A")	

CHR$	指定された文字コードから文字を返す	
書式	文字列変数 = CHR$(文字コード)	
引数	文字コード	文字ごとに対応する番号(UTF-16)
戻り	文字コードに対応する文字	
例	S$=CHR$(65)	

VAL	文字列から数値を得る	
書式	変数 = VAL("文字列")	
引数	文字列	数字を表す文字列("123"など)、または文字列変数
戻り	文字列から解釈された数値	
例	A=VAL("123")	

STR$	数値から文字列を得る	
書式	文字列変数 = STR$(数値 [,桁数])	
引数	数値	文字列に変換したい数値
	桁数	・指定桁数で右揃えしたい場合に指定 ・数値の桁数が指定桁数より大きい場合指定を無視
戻り	数値から生成された文字列(123→"123")	
例	S$=STR$(123)	

HEX$	数値から16進文字列を得る	
書式	文字列変数 = HEX$(数値 [,桁数])	
引数	数値	16進文字列を得たい数値(小数部は切り捨て)
	桁数	・出力する16進文字の桁数 ・指定すると先頭に0を埋めた文字列を返す
戻り	数値から生成された16進文字列(255→"FF")	
例	S$=HEX$(65535,4)	

BIN$	数値から2進文字列を得る	
書式	文字列変数 = BIN$(数値 [,桁数])	
引数	数値	2進文字列を得たい数値(小数部は切り捨て)
	桁数	・出力する2進文字の桁数 ・指定すると先頭に0を埋めた文字列を返す
戻り	数値から生成された2進文字列(255→"11111111")	
例	S$=BIN$(65535,16)	

FORMAT$	表示書式を使って値を整形し文字列化する	
書式	変数$ = FORMAT$("書式文字列", 値 ,…)	
引数	書式文字列(複数列挙可能)	%S: 文字列変数の内容を出力 %D: 整数を10進出力 %X: 整数を16進出力 %F: 実数を出力 %B: 整数を2進出力
	書式文字列の補助指定	%のあとに次の補助指定を行うことにより出力を整形 ・桁数指定:桁数の数値を指定(%8D、%4X) ・小数の桁数指定:整数部.小数部の桁数(%8.2F) ・空白埋め:空白文字＋桁数を指定(% 4D→　　0) ・ゼロ埋め:0＋桁数を指定(%08D→00000000) ・左寄せ:-記号＋桁数を指定(%-8D) ・+符号表示:+記号＋桁数を指定(%+8D)
	値	・整形する元の値 ・書式内で指定された要素分カンマ(,)で区切って列挙
戻り	生成された文字列	
例	S$=FORMAT$("%06D",A)	

LEN	文字列内の文字数を得る/配列の要素数を得る	
書式	変数 = LEN("文字列" または 配列変数)	
引数	文字列の場合	文字数を調べたい文字列または文字列変数名
	配列変数の場合	要素数を調べたい配列変数名
戻り	・文字列のとき: 文字数(すべての文字を1文字として数える) ・配列変数のとき: 要素数	
例	A=LEN("ABC123")	

MID$	文字列の指定位置から指定数分の文字列を取り出す	
書式	文字列変数 = MID$("文字列", 開始位置, 文字数)	
引数	文字列	元になる文字列
	開始位置	文字列を取り出す開始位置(文字単位)
	文字数	取り出す文字数
戻り	取り出した文字列	
例	S$=MID$("ABC",0,2)	

LEFT$	文字列の左端から指定数分の文字列を取り出す	
書式	文字列変数 = LEFT$("文字列", 文字数)	
引数	文字列	元になる文字列
	文字数	取り出す文字数
戻り	取り出した文字列	
例	S$=LEFT$("ABC",2)	

RIGHT$	文字列の右端から指定数分の文字列を取り出す	
書式	変数$ = RIGHT$("文字列", 文字数)	
引数	文字列	元になる文字列
	文字数	取り出す文字数
戻り	取り出した文字列	
例	S$=RIGHT$("ABC",2)	

INSTR	文字列内から対象文字列を検索	
書式	変数 = INSTR([開始位置,] "元文字列", "検索する文字列")	
引数	開始位置	・元文字列内で検索を開始する位置(0～文字単位) ・省略すると先頭から検索
	元文字列	元になる文字列
	検索する文字列	元文字列の中から検索したい文字列
戻り	・見つかった場合: 文字列内の位置(文字単位) ・見つからなかった場合: -1	
例	A=INSTR(0, "ABC","B")	

SUBST$	文字列の置換	
書式	文字列変数 = SUBST$("文字列", 開始位置, [文字数,] "置換文字列")	
引数	文字列	元になる文字列
	開始位置	元になる文字列の置換開始位置(0～文字数-1)
	文字数	・置換する文字数 ・省略時は置換位置以降の全文字を置換文字列に置き換え
	置換文字列	開始位置から文字数分をこの文字列で置き換え
戻り	置換された文字列	
例	A$=SUBST$("ABC",0,2,"XY")	

ソースコード操作

指定したSLOTにプログラム文字列を書き込む機能

PRGEDIT	操作するプログラムSLOTと、カレント行を指定	
書式	PRGEDIT プログラムSLOT [,行番号]	
引数	プログラムSLOT	・操作するプログラムSLOT: 0〜3 ・現在実行中のSLOTを指定するとエラー
	行番号	・操作対象とする行(カレント行) ・省略した場合、先頭行がカレント行となる ・行番号に-1を指定した場合、最終行がカレント行となる
例	PRGEDIT 0	

PRGGET$	カレント行１行分の文字列を取得
書式	文字列変数=PRGGET$()
戻り	カレント行のソース文字列(範囲外の場合、空文字列)
例	A$=PRGGET$()

PRGSET	カレント行の内容を指定文字列に置き換える PRGGET$が空文字列を返す場合は行追加	
書式	PRGSET "文字列"	
引数	文字列	カレント行を置き換える文字列
例	PRGSET "'Comment"	

PRGINS	カレント行への１行挿入 改行コードCHR$(10)を含む文字列は複数行の挿入	
書式	PRGINS "文字列" [,フラグ]	
引数	文字列	挿入するソース文字列
	フラグ	1=カレント行の後方に挿入 0=カレント行の前方へ挿入(省略時=0、前方へ)
例	PRGINS "PRINT "+CHR$(34)+"HELLO"+CHR$(34)	

PRGDEL	カレント行の削除	
書式	PRGDEL [削除行数]	
引数	削除行数	・削除したい行の数(省略時1行) ・マイナス値を入れた場合全体を削除
例	PRGDEL	

PRGSIZE	ソースコードの行数取得	
書式	変数=PRGSIZE([プログラムSLOT [,取得する値のタイプ]])	
引数	プログラムSLOT	行数を取得するプログラムSLOT: 0〜3
	取得する値のタイプ	0=行数、1=文字数、2=空き文字数(デフォルトは0)
戻り	タイプに応じた値	
例	A=PRGSIZE(0)	

PRGNAME$	プログラムのファイル名 LOAD/SAVE命令で扱ったファイル	
書式	文字列変数=PRGNAME$([プログラムSLOT])	
引数	プログラムSLOT	ファイル名を取得するプログラムSLOT: 0〜3
戻り	・プログラムのファイル名 ・プログラム実行中は実行しているSLOT ・実行していない時は「直前に実行していたSLOT」 ・「直前に実行していたSLOT」は通常0 ・STOP命令やSTARTボタンでプログラム中断時、およびエラー発生時はその時のSLOTとなり、以後RUNするまでその状態が維持される	
例	PRINT PRGNAME$(0)	

ビット演算

数値に対してビット単位で演算を行う機能

MOD	数値1を数値2で割った余りの取得	
書式	変数=数値1 MOD 数値2	
引数	数値1	割られる数(または式)
	数値2	割る数(または式、0で割るとエラー)
例	A=200 MOD 5	

DIV	数値1を数値2で割った整数値の取得	
書式	変数=数値1 DIV 数値2	
引数	数値1	割られる数(または式)
	数値2	割る数(または式、0で割るとエラー)
例	A=200 DIV 5	

AND	数値1と数値2の論理積(ビットの掛算)	
書式	変数=数値1 AND 数値2	
引数	数値1	ビット列1
	数値2	ビット列2
例	A=200 AND &HE7	

OR	数値1と数値2の論理和(ビットの足し算)	
書式	変数=数値1 OR 数値2	
引数	数値1	ビット列1
	数値2	ビット列2
例	A=128 OR &HA3	

XOR	数値1と数値2の排他的論理和(同じ時0、異なる時反転)	
書式	変数=数値1 XOR 数値2	
引数	数値1	ビット列1
	数値2	ビット列2
例	A=100 XOR &H4C	

NOT	数値のビット反転(1の補数を求める)	
書式	変数=NOT 数値	
引数	数値	ビット列
例	A=NOT 1	

<<	数値を回数分左へビットシフト	
書式	変数=数値 << 回数	
引数	数値	ビット列1
	回数	ビットシフト数
例	A=100 << 2	

>>	数値を回数分右へビットシフト	
書式	変数=数値 >> 回数	
引数	数値	ビット列1
	回数	ビットシフト数
例	A=100 >> 2	

MML

MusicMacroLanguageのコマンド

MML (1)	曲全体を制御するコマンド	
	チャンネル指定	:0～:15 (コロン:に続けてチャンネル番号指定)
	テンポ指定	T1～T512
例	'--- テンポ120でドミソの和音 BGMPLAY "T120:0CCC:1EEE:2GGG"	

MML (2)	音の長さに関するコマンドや記述方法	
	デフォルト音長の指定	L1～L192　指定すると以降のデフォルト音長が変化
	音長の個別指定◆	デフォルト音長以外の長さで演奏させたい場合に、音程記号に並べて入力することで音長を変更する (例)ドの長さを直接指定する C1(ドの全音符) C2(ドの2分音符) C4(ドの4分音符) C8(ドの8分音符) C16(ドの16分音符) C32(ドの32分音符) C1.～C32.(ドの付点音符表現) ※三連符はC12C12C12、C24C24C24のように指定
	奏法	&　前後の音をつなぐ() _　ポルタメント()
	発音時間の割合(ゲート)設定	Q0～Q8　小さい程音が途切れて聞こえる

MML (3)	音程(音の高さ)に関するコマンド	
	音階指定	C(ド) D(レ) E(ミ) F(ファ) G(ソ) A(ラ) B(シ)
	音階を半音上げる	C# D# E# F# G# A# B#
	音階を半音下げる	C- D- E- F- G- A- B-
	休符◆	R　※音階と同じように利用可能 (例) R4(4分休符)
	オクターブ指定	O0～O8 オクターブ数値指定
	オクターブを1つ上げる	<
	オクターブを1つ下げる	>
	オクターブ指定の反転	!　※指定すると以降<>記号の扱いが逆転
	キーを数値指定◆	N0～N127　※O4C=60、半音ごとに1増減
例	'--- ドドレミファソソラシシシシ BGMPLAY "CCDEFGGABBBB"	

MML (4)	音量や定位に関するコマンド	
	音量指定◆	V0～V127　音量の数値指定
	音量を1つ上げる	(
	音量を1つ下げる	)
	パンポット◆	P0～P127　スピーカーから聞こえる場所(定位)を決める 左:P0～P63　中央:P64　右:P65～P127
	エンベロープ設定	@E数値A,D,S,R　発声から減衰までの音量変化設定 A(Attack time):0～127 D(Decay time) :0～127 S(Sustain level):0～127 R(Release time):0～127　※各timeは小さいほど遅い (例) @E127,100,30,100
	エンベロープリセット	@ER　エンベロープを解除

MML（5）	音色変更コマンド	
	楽器音変更	@0 〜@127　GM音源相当(SMILETOOLで確認可能) @128　標準ドラムセット @129　エレクトリックドラムセット @144〜@150　PSG音源 @151　ノイズ音源 @224〜@255　ユーザー定義波形(WAVSETで登録した波形) @256〜　BEEP用に用意された効果音
	@128ドラムセット(@129)	B1　Acoustic Bass Drum 2(909BD) C2　Acoustic Bass Drum 1(808BDTom) C2# Side Stick(808RimShot) D2　Acoustic Snare(808SD) D2# Hand Clap E2　Electric Snare(909SD) F2　Low Floor Tom(808TomLF) F2# Closed Hi-hat(808CHH) G2　High Floor Tom(808TomF) G2# Pedal Hi-hat(808CHH) A2　Low Tom(808TomL) A2# Open Hi-hat(808OHH) B2　Low-Mid Tom(808TomLM) C3　High Mid Tom(808TomHM) C3# Crash Cymbal 1(808Cymbal) D3　High Tom(808TomH) D3# Ride Cymbal 1 E3　Chinese Cymbal F3　Ride Bell F3# Tambourine G3　Splash Cymbal G3# Cowbell(808Cowbell) A3　Crash Cymbal 2 A3# Vibra-slap B3　Ride Cymbal 2 C4　High Bongo C4# Low Bongo D4　Mute Hi Conga(808CongaMute) D4# Open Hi Conga(808CongaHi) E4　Low Conga(808CongaLo) F4　High Timbale F4# Low Timbale G4　High Agogo G4# Low Agogo A4　Cabasa A4# Maracas(808Maracas) B4　Short Whistle C5　Long Whistle C5# Short Guiro D5　Long Guiro D5# Claves(808Claves) E5　Hi Wood Block F5　Low Wood Block F5# Mute Cuica G5　Open Cuica G5# Mute Triangle A5　Open Triangle

MML（6）	音や音量に微妙な揺れを与える特殊効果コマンド ※@MA、@MP、@MLは同時に使用できません	
	モジュレーションの開始	@MON
	モジュレーションの停止	@MOF
	デチューン(周波数の微調整)設定◆	@D-128〜@D127　(-128で1音低く、+127で1音高くなる)
	トレモロ設定	@MA 数値Depth, Range, Speed, Delay (各0〜127) (例) @MA64,1,16,32
	ビブラート設定	@MP 数値Depth, Range, Speed, Delay (各0〜127) (例) @MP64,1,16,32
	オートパンポット設定	@ML 数値Depth, Range, Speed, Delay (各0〜127) (例) @ML100,1,8,0

MML（7）	特殊な演奏コマンド	
	リピート開始	[
	リピート終了	]回数　※回数省略時は無限ループ◆ (例) 複雑なくりかえし CCC CCC DEF CCC CCC DEF と演奏 BGMPLAY "[[CCC]2DEF]2"
	MML内部変数の指定	$0～$7　MML内部変数 ※◆印のコマンドで、数値の代わりに指定可能 (例) V64 の代わりに $0=64 V$0
	MML内部変数への代入	$0=数値～$7=数値　変数への数値代入(0～255) ※演奏中の変数はBGMVAR命令で代入・参照可能
	マクロの定義	{ラベル名=MML} 同じメロディーやフレーズを再利用したい場合に利用 ・定義MML内でチャンネル指定は禁止 ・ラベル名は8文字までの英数字 ・同じラベル名で再定義はできない
	マクロの利用	{ラベル名}　定義済みのラベルに対応するMMLが展開される
例	'--- マクロを使ったリズム演奏 BGMPLAY "T240@128O2{PT0=CDEDCDE<G}[{PT0}]4"	

高度サウンドユニット

音声信号の解析・加工に興味のある方向けの高度な拡張命令

BIQUAD	BiQuadフィルタ	
書式	BIQUAD OTWK,INWK,FP	
引数	OTWK	出力結果を受け取る配列(1次元,2次元) ・OTWKの配列の要素数がINWKの配列の要素数より少ない場合エラー
	INWK	入力要素を渡す配列(1次元,2次元)
	FP	フィルター配列(要素数は13以上、FP[5..12]の値は変更なし) ・FPの配列の要素数が13より少ない場合エラー
補足	IN,OUT共に2次元配列を指定した場合はステレオ扱い。INで指定される配列の全要素に対して以下の式の演算を行い、OTWKで指定した配列に出力します。 OTWK[t]=FP[0]*INWK[t]+FP[1]*INWK[t-1]+FP[2]*INWK[t-2]-FP[3]*OTWK[t-1]-FP[4]*OTWK[t-2] ※配列のインデックスが負の場合は、モノラル又は左チャンネルはFP[5..8]、右チャンネルはFP[9..12]の値を用います。 FP[5..12]の値は実行ごとに更新されます なお、フィルタ係数配列FPの値の詳細については、http://www.musicdsp.org/files/Audio-EQ-Cookbook.txtを参考にしております。 上記文献に示される係数の b0,b1,b2,a0,a1,a2 を用いて FP[0]=b0/a0 FP[1]=b1/a0 FP[2]=b2/a0 FP[3]=a1/a0 FP[4]=a2/a0 とすると、その特性のフィルタになります。	
例	DIM OD[1000],ID[1000],FP[13] BQPARAM FP,#BQLPF,32730,8000,1.0 BIQUAD OD,ID,FP	

BQPARAM	BiQuadフィルタのフィルタ係数を計算		
書式	BQPARAM FP,k,s,f,q(o) [,g]		
引数	FP		フィルタ係数を受け取る配列（要素数は13以上、FP[5..12]の値は変更なし）
	k		フィルタ種別:0～7 0 #BQAPF BQPARAM FP,k,s,f 1 #BQLPF BQPARAM FP,k,s,f 2 #BQHPF BQPARAM FP,k,s,f 3 #BQBPF BQPARAM FP,k,s,f,o 4 #BQBSF BQPARAM FP,k,s,f,o 5 #BQLSF BQPARAM FP,k,s,f,q,g 6 #BQHSF BQPARAM FP,k,s,f,q,g 7 #BQPEQ BQPARAM FP,k,s,f,o,g
	s		サンプリングレート(Hz)
	f		カットオフ周波数(Hz)
	q		Q値
	o		帯域幅(octave: nでカットオフ周波数を中心とした nオクターブの帯域)
	g		増幅量(db: -40<=g<=40)
補足	フィルタごとの説明	オールパスフィルタ	カットオフ周波数付近の位相のみ変化
		ローパスフィルタ	カットオフ周波数以下を通過
		ハイパスフィルタ	カットオフ周波数以上を通過
		バンドパスフィルタ	カットオフ周波数を中心とした指定の周波数帯域を通過
		バンドストップフィルタ	カットオフ周波数を中心とした指定の周波数帯域以外を通過
		ローシェルフフィルタ	カットオフ周波数以下を増幅
		ハイシェルフフィルタ	カットオフ周波数以上を増幅
		ピーキングイコライザ	カットオフ周波数を中心とした指定の周波数帯域を増幅
例	DIM OD[1000],ID[1000],FP[13] BQPARAM FP,#BQLPF,32730,8000,1.0 BIQUAD OD,ID,FP		

FFT	複素数配列に対してフーリエ変換	
書式	FFT oR,oI,iR,iI[,W]	
引数	oR,oI	結果を格納するoR(実数部),oI(虚数部)の複素数配列
	iR,iI	iR(実数部),iI(虚数部)で表される複素数配列
	W	窓関数値配列。指定した場合、入力の複素数配列に対し同一インデックスの要素値を乗算（窓関数値配列はFFTWFN命令で得ること可能）
補足	iR,iIで表される複素数配列に対してフーリエ変換を行い、結果を oR,oIの複素数配列に返す。 指定する配列は全て同じ要素数で2のn乗であることが必要（条件を満たしていない場合エラー）	
例	DIM iR[1024],iI[1024] DIM oR[1024],oI[1024] FFT oR,oI,iR,iI	

IFFT	複素数配列に対してフーリエ逆変換	
書式	IFFT oR,oI,iR,iI	
引数	oR,oI	結果を格納するoR(実数部),oI(虚数部)の複素数配列
	iR,iI	iR(実数部),iI(虚数部)で表される複素数配列
補足	iR,iIで表される複素数配列に対してフーリエ逆変換を行い、結果を oR,oIの複素数配列に返す。 指定する配列は全て同じ要素数で2のn乗であることが必要（条件を満たしていない場合エラー）	
例	DIM iR[1024],iI[1024] DIM oR[1024],oI[1024] IFFT oR,oI,iR,iI	

FFTWFN	配列 W に nで指定した種類の窓関数値を返す	
書式	FFTWFN W,n	
引数	W	処理を適用する配列（要素数は2のn乗のみ）
	n	窓関数種別 0 #WFRECT 矩形窓 1 #WFHAMM ハミング窓 2 #WFHANN ハニング窓 3 #WFBLKM ブラックマン窓
例	DIM iR[1024],iI[1024],WF[1024] DIM oR[1024],oI[1024] FFTWFN WF,#WFHANN FFT oR,oI,iR,iI,WF	

PCMPOS	PCMSTREAMの内部のFIFOに転送する先頭位置（配列のインデックス）示すシステム変数（FIFOが空になった時にこの位置から転送されます）
書式	PCMPOS
補足	このシステム変数へ書き込むことで、PCMSTREAMの再生位置を設定します。 再生中に設定すると即時で指定の位置から再生（設定値は、PCMSTREAMに設定されている配列の要素数で割った余りとなります）
例	P=PCMPOS PCMPOS=P

PCMCONT	PCMSTREAM停止時の状態から再開
書式	PCMCONT
例	PCMCONT

PCMSTOP	PCMSTREAMを停止
書式	PCMSTOP
例	PCMSTOP

PCMSTREAM (1)	指定配列を左右のチャンネルに割り当てPCM再生を行う ・最初に指定した配列の長さでループ再生	
書式	PCMSTREAM M [,サンプリングレート]	
引数	M	PCMが格納された配列:-32768～32767(符号付き16ビット) MがM[2,N]の2次元配列の場合、[0,x]が左[1,x]を右チャンネルに割り当て
	サンプリングレート	単位はHz、範囲は1～192000(省略時32730)
補足	プログラムの実行中のみ再生（プログラムが終了する／エラーで停止するなどして、ダイレクトモードに戻ると再生が停止します）	
例	DIM MONO[1000] PCMSTREAM MONO	

PCMSTREAM (2)	左右分の配列からPCM再生を行う ・最初に指定した配列の長さでループ再生	
書式	PCMSTREAM L,R [,サンプリングレート]	
引数	L,R	Lに左側、Rに右側のPCMが格納された配列:-32768～32767(符号付き16ビット)
	サンプリングレート	単位はHz、範囲は1～192000(省略時32730)
補足	プログラムの実行中のみ再生（プログラムが終了する／エラーで停止するなどして、ダイレクトモードに戻ると再生が停止します）	
例	DIM LEFT[1000],RIGHT[1000] PCMSTREAM LEFT,RIGHT DIM STEREO[2,1000] PCMSTREAM STEREO	

PCMSTREAM (3)	現在再生中の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）	
書式	PCMSTREAM サンプリングレート	
引数	サンプリングレート	単位はHz、範囲は1～192000(省略時32730)
例	PCMSTREAM 880	

PCMSTREAM (4)	※BIG固有命令 左右分の配列からPCM再生を行う ・最初に指定した配列の長さでループ再生	
書式	PCMSTREAM L,R [,サンプリングレート][,出力先]	
引数	L,R	Lに左側、Rに右側のPCMが格納された配列:-32768～32767(符号付き16ビット)
	サンプリングレート	単位はHz、範囲は1～192000(省略時32730)
	出力先	・WiiU専用の引数(3DS版「プチコン3号」では使えません) ・事前にXON WIIUが必要 ・複数の出力先の値をORする事で同時に発声可能 |b00|TV(1) |b01|GamePad(2) |b02|未割当(4) |b03|未割当(8) |b04|リモコン1(16) |b05|リモコン2(32) |b06|リモコン3(64) |b07|リモコン4(128)
補足	プログラムの実行中のみ再生（プログラムが終了する／エラーで停止するなどして、ダイレクトモードに戻ると再生が停止します）	
例	DIM LEFT[1000],RIGHT[1000] PCMSTREAM LEFT,RIGHT DIM STEREO[2,1000] PCMSTREAM STEREO	

PCMSTREAM (5)	※BIG固有命令 指定配列を左右のチャンネルに割り当てPCM再生を行う ・最初に指定した配列の長さでループ再生	
書式	PCMSTREAM M,サンプリングレート[,出力先]	
引数	M	PCMが格納された配列:-32768～32767(符号付き16ビット) MがM[2,N]の2次元配列の場合、[0,x]が左[1,x]を右チャンネルに割り当て
	サンプリングレート	単位はHz、範囲は1～192000(省略時32730)
	出力先	・WiiU専用の引数(3DS版「プチコン3号」では使えません) ・事前にXON WIIUが必要 ・複数の出力先の値をORする事で同時に発声可能 |b00|TV(1) |b01|GamePad(2) |b02|未割当(4) |b03|未割当(8) |b04|リモコン1(16) |b05|リモコン2(32) |b06|リモコン3(64) |b07|リモコン4(128)
補足	プログラムの実行中のみ再生（プログラムが終了する／エラーで停止するなどして、ダイレクトモードに戻ると再生が停止します）	
例	DIM MONO[1000] PCMSTREAM MONO	

PCMSTREAM (6)	※BIG固有命令 現在再生中の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）	
書式	PCMSTREAM サンプリングレート[,出力先]	
引数	サンプリングレート	単位はHz、範囲は1～192000(省略時32730)
	出力先	・WiiU専用の引数(3DS版「プチコン3号」では使えません) ・事前にXON WIIUが必要 ・複数の出力先の値をORする事で同時に発声可能 |b00|TV(1) |b01|GamePad(2) |b02|未割当(4) |b03|未割当(8) |b04|リモコン1(16) |b05|リモコン2(32) |b06|リモコン3(64) |b07|リモコン4(128)
例	PCMSTREAM 880	

PCMVOL	PCMSTREAMの音量を設定	
書式	PCMVOL [CH,]VOL	
引数	CH	チャンネル:0=左,1=右（省略時は両チャンネル）
	VOL	音量:-32767～32767（負の値で位相が逆になります）
例	PCMVOL 16384	

RINGCOPY (1)	コピー先配列変数をリングバッファとしてデータをコピーする命令 COPY命令のように配列間の値をコピーするが、コピー先よりもコピー元のコピー要素数が多い場合でも要素数を拡張せず、配列の先頭に折り返してデータをコピーする。	
書式	RINGCOPY コピー先配列, コピー先オフセット, コピー元配列 [[,コピー元オフセット],コピー要素数]	
引数	コピー先配列,コピー先オフセット	コピーされる側の配列名とコピー開始位置
	コピー元配列,コピー元オフセット	元になる側の配列名とコピー開始位置
	コピー要素数	実際にコピーする数
補足	コピー先、コピー元共に1,2次元配列を指定可能。2次元配列を指定した場合、第1次元の要素数分チャンネルのあるマルチチャネルデータとして扱われる。 コピー先、コピー元両方に2次元配列を指定した場合、双方のチャンネル数が異なるとエラーとなる。 コピー先が2次元、コピー元が1次元配列の場合、コピー元データがコピー先の全チャンネルに同じようにコピーされる。	
例	DIM BUFF[1024],DT[256] RINGCOPY BUFF,1000,DT	

RINGCOPY (2)	リングバッファとしてデータをコピーする命令 関数型で利用した場合、リングバッファとしてのコピーを行った後の末尾位置を返す。	
書式	変数 = RINGCOPY(コピー先配列, コピー先オフセット, コピー元配列 [[,コピー元オフセット],コピー要素数])	
引数	コピー先配列,コピー先オフセット	コピーされる側の配列名とコピー開始位置
	コピー元配列,コピー元オフセット	元になる側の配列名とコピー開始位置
	コピー要素数	実際にコピーする数
戻り	コピーしたデータの末尾位置（リングバッファとして扱う場合、帰ってきた値が次回のコピー先オフセットとなる）	
補足	コピー先、コピー元共に1,2次元配列を指定可能。2次元配列を指定した場合、第1次元の要素数分チャンネルのあるマルチチャネルデータとして扱われる。 コピー先、コピー元両方に2次元配列を指定した場合、双方のチャンネル数が異なるとエラーとなる。 コピー先が2次元、コピー元が1次元配列の場合、コピー元データがコピー先の全チャンネルに同じようにコピーされる。	
例	DIM BUFF[1024],DT[256] N=1000 N=RINGCOPY(BUFF,N,DT)	

ARYOP	配列間で要素の一括演算を行う 配列の先頭から、結果格納配列の末尾まで、それぞれの要素間で演算タイプで指定された演算を行い、結果格納配列に演算結果を格納する。 パラメータ配列の要素数が結果格納配列の要素数と異なる場合、配列の末尾まで到達したら配列の先頭に戻ってくりかえし参照する。	
書式	ARYOP 演算タイプ，結果格納配列変数，パラメータ1，パラメータ2 [,パラメータ3]	
引数	演算タイプ	以下の数値（または定数）を指定可能 0 #AOPADD 加算(p1+p2) 1 #AOPSUB 減算(p1-p2) 2 #AOPMUL 乗算(p1*p2) 3 #AOPDIV 除算(p1/p2) 4 #AOPMAD 積和(p1*p2+p3) 5 #AOPLIP 線形補間(p1*p3+p2*(1-p3)) 6 #AOPCLP クランプ(p1の値をp2<=x<=p3の範囲に丸める)
	結果格納配列	数値列を指定
	パラメータ 1,2,3	数値配列、または通常数値を指定（通常数値を指定した場合はその項は常にその数値を使用）
例	DIM O[10],I0[10],I1[10] ARYOP #AOPADD,O,I0,I1	

エラー表

エラー用システム変数	エラーが発生すると、システム変数に情報が残ります。 ERRNUM（エラー番号） ERRLINE（発生した行番号）
主なエラーの内容	3:Syntax error(文法が間違っている 4:Illegal function call(命令や関数の引数の数が違う 5:Stack overflow(スタックがあふれた 6:Stack underflow(スタックが不足した 7:Divide by zero(0による除算をした 8:Type mismatch(変数の型が一致しない 9:Overflow(演算結果が許容範囲を超えた 10:Out of range(範囲外の数値を指定した 11:Out of memory(メモリー不足である 12:Out of code memory(コード領域のメモリー不足 13:Out of DATA(READできるDATAが不足 14:Undefined label(指定されたラベルが存在しない 15:Undefined variable(指定された変数が存在しない 16:Undefined function(指定された命令・関数が存在しない 17:Duplicate label(ラベルが二重定義されている 18:Duplicate variable(変数が二重定義されている 19:Duplicate function(命令・関数が二重定義されている 20:FOR without NEXT(NEXTが無いFORがある 21:NEXT without FOR(FORが無いNEXTがある 22:REPEAT without UNTIL(UNTILが無いREPEATがある 23:UNTIL without REPEAT(REPEATが無いUNTILがある 24:WHILE without WEND(WENDが無いWHILEがある 25:WEND without WHILE(WHILEが無いWENDがある 26:THEN without ENDIF(ENDIFが無いTHENがある 27:ELSE without ENDIF(ENDIFが無いELSEがある 28:ENDIF without IF(IFが無いENDIFがある 29:DEF without END(ENDが無いDEFがある 30:RETURN without GOSUB(GOSUBが無いRETURNがある 31:Subscript out of range(配列添字が範囲外 32:Nested DEF(DEF内でDEFを定義した 33:Can't continue(CONTでプログラムを再開できない 34:Illegal symbol string(ラベル文字列の記述方法が間違っている 35:Illegal file format(SMILEBASICでは扱えないファイル形式 36:Mic is not available(XON MICせずにマイク命令を利用した 37:Motion sensor is not available(XON MOTIONせずにモーション命令を利用した 38:Use PRGEDIT before any PRG function(PRGEDITせずにPRG系の命令を使った 39:Animation is too long(アニメーション定義が長すぎる 40:Illegal animation data(不正なアニメーションデータ 41:String too long(文字列が長すぎる 42:Communication buffer overflow(MPSEND用の送信バッファがあふれた 43:Can't use from direct mode(ダイレクトモードでは使えない命令 44:Can't use in program(プログラム内では使えない命令 45:Can't use in tool program(ツールプログラム内からは使えない命令 46:Load failed(ファイル読み込みに失敗 47:Illegal MML(不正なMML(MusicMacroLanguage) 48:Uninitialized variable used(未初期化変数を参照しようとした 49:Protected resource(保護リソースを読み出そうとした 50:Protected file(保護ファイルを操作しようとした 51:DLC not found(DLCを購入していないため該当機能は使用できない 52:Incompatible statement(現在の互換モードで使用できない機能を使おうとした 53:END without CALL(ユーザー定義命令を呼び出していないのにユーザー定義末尾のENDに遭遇した 54:Array is too large(SAVE命令で保存しようとした配列のサイズが大きすぎる

システム変数

システム変数とは?	SmileBASICが管理しているシステムで予約された変数です。 ※基本読み取り専用ですが一部代入できる変数もあります
例	CSRX 'カーソル位置X CSRY 'カーソル位置Y CSRZ 'カーソル位置Z(奥行) FREEMEM '残りユーザーメモリー(キロバイト) VERSION 'システムバージョン(&HXXYYZZZZ) TABSTEP 'TAB移動量(書込み可能) SYSBEEP 'システム効果音制御(書込み可能、TRUE=許可) ERRNUM 'エラー番号 ERRLINE 'エラー発生行 ERRPRG 'エラー発生プログラムSLOT PRGSLOT 'PRG命令のカレントプログラムSLOT RESULT 'ダイアログの結果(TRUE/FALSE/-1=中断) MAINCNT 'SmileBASIC起動時からのフレーム数 MILLISEC 'SmileBASIC起動時からのミリセカンド値 MICPOS 'サンプリングの現在位置 MICSIZE 'サンプリングバッファのサンプリング数 MPCOUNT 'セッション参加人数 MPHOST 'ホストのID MPLOCAL '自分のID TRUE '常に1 FALSE '常に0 TIME$ '時刻文字列(HH:MM:SS) DATE$ '日付文字列(YYYY/MM/DD) HARDWARE '環境判定用システム変数(0=3DS,1=New3DS) CALLIDX 'SPFUNCおよびBGFUNCで呼び出された番号 PCMPOS 'PCMSTREAM内の位置(高度サウンドDLC専用) EXTFEATURE '拡張命令のサポート状況

定数

#	・システムに用意された32bit数値の定義 ・色の指定やボタンを扱う時に数値代わりに利用 ・IF BUTTON() AND(#A OR #B) THEN のように使用
例	'--- 汎用 #ON '1 #OFF '0 #YES '1 #NO '0 #TRUE '1 #FALSE '0 '--- RGB（グラフィック用） #AQUA '&HFF00F8F8 #BLACK '&HFF000000 #BLUE '&HFF0000F8 #CYAN '&HFF00F8F8 #FUCHSIA '&HFFF800F8 #GRAY '&HFF808080 #GREEN '&HFF008000 #LIME '&HFF00F800 #MAGENTA '&HFFF800F8 #MAROON '&HFF800000 #NAVY '&HFF000080 #OLIVE '&HFF808000 #PURPLE '&HFF800080 #RED '&HFFF80000 #SILVER '&HFFC0C0C0 #TEAL '&HFF008080 #WHITE '&HFFF8F8F8 #YELLOW '&HFFF8F800 '--- TEXTCOLOR(文字の色) #TBLACK '1、黒 #TMAROON '2、暗い赤 #TRED '3、赤 #TGREEN '4、緑 #TLIME '5、明るい緑 #TOLIVE '6、暗い黄色 #TYELLOW '7、黄色 #TNAVY '8、暗い青 #TBLUE '9、青 #TPURPLE '10、紫 #TMAGENTA '11、明るい紫 #TTEAL '12、暗い水色 #TCYAN '13、水色 #TGRAY '14、灰 #TWHITE '15、白 '--- BUTTON #UP '&H0001、上 #DOWN '&H0002、下 #LEFT '&H0004、左 #RIGHT '&H0008、右 #A '&H0010 #B '&H0020 #X '&H0040 #Y '&H0080 #L '&H0100 #R '&H0200 #ZL '&H0800 #ZR '&H1000 '--- ATTR #TROT0 '&H00、回転なし #TROT90 '&H01、回転90度 #TROT180 '&H02、回転180度 #TROT270 '&H03、回転270度 #TREVH '&H04、左右反転 #TREVV '&H08、上下反転 '---SPSET/SPCHR ATTR #SPSHOW '&H01、表示 #SPROT0 '&H00、回転なし #SPROT90 '&H02、回転90度 #SPROT180 '&H04、回転180度 #SPROT270 '&H06、回転270度 #SPREVH '&H08、左右反転 #SPREVV '&H16、上下反転 #SPADD '&H32、加算合成 '--- BG ATTR #BGROT0 '&H0000、回転なし #BGROT90 '&H1000、回転90度 #BGROT180 '&H2000、回転180度

```
#BGROT270 '&H3000、回転270度
#BGREVH '&H4000、左右反転
#BGREVV '&H8000、上下反転
'--- SPCHK/BGCHK
#CHKXY '&H01、XY座標
#CHKZ '&H02、Z座標
#CHKUV '&H04、画像位置
#CHKI '&H08、DEF定義
#CHKR '&H16、回転
#CHKS '&H32、スケール
#CHKC '&H64、色
#CHKV '&H128、内部変数
'--- DLC:SOUND:BQPARAM
#BQAPF '0、オールパスフィルタ
#BQLPF '1、ローパスフィルタ
#BQHPF '2、ハイパスフィルタ
#BQBPF '3、バンドパスフィルタ
#BQBSF '4、バンドストップフィルタ
#BQLSF '5、ローシェルフフィルタ
#BQHSF '6、ハイシェルフフィルタ
#BQPEQ '7、ピーキングイコライザ
'--- DLC:SOUND:FFTWIN
#WFRECT '0、矩形窓
#WFHAMM '1、ハミング窓
#WFHANN '2、ハニング窓
#WFBLKM '3、ブラックマン窓
'--- DLC:SOUND:ARYOP
#AOPADD '0、加算(p1+p2)
#AOPSUB '1、減算(p1-p2)
#AOPMUL '2、乗算(p1*p2)
#AOPDIV '3、除算(p1/p2)
#AOPMAD '4、積和(p1*p2+p3)
#AOPLIP '5、線形補間(p1*p3+p2*(1-p3))
#AOPCLP '6、クランプ(p1をp2<=x<=p3に丸める)
```