===============================================================================

アドイン版　Casio Basic Interpreter (& Compiler) for CG ver 0.5x α版　マニュアル

copyright(c)2015/2016/2017/2018 by sentaro21

e-mail sentaro21@pm.matrix.jp

===============================================================================

C.Basic for CGとは？

===============================================================================

純正Casio Basicとfx-5800Pの使い勝手のいいとこ取りを目指した高速互換CasioBasicが

FX版C.Basicというわけで、C.Basic for CGはそのCG10/20/50対応版です。

プログラムソースはストレージメモリ上に置かれます。

サポートコマンドの範囲内では純正のCasioBasicとファイルフォーマット及び動作の互換があります。

===============================================================================（CGシリーズ拡張）

g1mとg3mの実行モード

===============================================================================

C.Basic for CGではFX版C.Basicのプログラムを互換動作させるg1mモードと、

CGシリーズの高解像度カラー画面に対応したg3mモードがあります。

互換動作時にはグラフィックスの解像度が128x64ピクセルになり、

グラフィックスコマンドはFX版C.Basic同様の動作となります。

テキスト表示はCGシリーズのハイレゾフォントとFXフォントが選択できます。

g1mプログラムをg3mモードで動作させた場合は、

テキスト系コマンドのプログラムでは互換性がありますが、

グラフィックスでは互換性の無い部分が出てきますので注意して下さい。

===============================================================================（CGシリーズ拡張）

使用可能メモリ　CG版

===============================================================================

CG10/20/50ではアドインとして自由に使用できるRAMは128KBあります。

さらにCG50ではOSで管理されていない空きRAMが6MB存在しています。

CG50ではそれを使用することで、128KB以上のフリーエリアを実現しています。

（空きRAMの6MBを使うためにはセットアップでUse Hidden RAMをOnにする必要があります。）

===============================================================================（CGシリーズ拡張）

内蔵エディタでの編集可能サイズ

===============================================================================

C.Basicでは標準のエディタと同様の使い勝手を目指したエディタを内蔵しています。

コピー＆ペーストは使えますが一度にコピーできるサイズは16KBが最大です。アンドゥは未実装です。

新規ファイルの初期編集可能サイズは32KBとなっています。（CG50では64KBです。）

プログラム作成時にこのサイズを超えると編集できなくなるので一旦保存後再編集して下さい。

再編集時のプログラム編集可能最大サイズはプログラムサイズ+8KBとなっています。

プログラム実行時にはメインプログラムはメインプログラムサイズ+8KB、サブプログラムはサブプログラム+512Bの編集可能エリアを確保した上で実行されます。

デバッグモード時での再編集可能サイズとなります。

プログラム編集可能最大サイズは、約62KB程度となっています。

===============================================================================

内蔵デバッグモード

===============================================================================

純正には無い機能として[AC]で中断したプログラムの再開、及びデバッグモードを装備しています。

デバッグモードは最初の実行時よりデバッグモードで起動することも出来ます。

トレース実行、１ステップ実行、ステップオーバー、ステップアウトをサポートしています。

===============================================================================

実行モードについて

===============================================================================

C.Basicでは純正CasioBasic同様の実数モードと、整数値しか扱えない整数モード、複素数を扱える複素数モードがあります。

それぞれの実行モードの切り替えはコメント欄で

'#CBasic

'#CBASIC

'#CBDBL

'#CBdbl

と記述するとそれ以降、実数モードに、

'#CBINT

'#CBint

と記述すると整数モードに、

'#CBCPLX

'#CBcplx

と記述すると複素数モードになります。

また、Setupからデフォルトの実行モードが設定できます。

コメント欄での変更はセットアップでの変更に優先します。

整数モードでは得られる結果がすべて整数値となりますが、

整数値しか扱わないプログラムでは実数モードに比べて1.5倍以上速くなります。

整数モードで実数を扱いたい場合は、式の前に＃を付けると＃以降の式を実数モードで計算します。ただし式の結果は整数値に丸められます。

（例）＃100×Frac 1.23 + 5

　整数モードではFlac 1.23は数値エラーになって実行できませんが、＃を付けると

　＃(Frac 1.23×100 + 5)

　＃以降の式を実数計算することが出来るようになるので、結果は実数モードと同じく28になります。

　式の途中に＃がある場合は、その＃以降、式の終わり（区切り記号まで）まで実数計算します。

（例）100×＃Frac 1.23 + 5

　式の解釈は

　100×＃(Frac 1.23 + 5)となるので実数モードでは523、整数モードでは500になります。

===============================================================================

実行時ウエイトについて

===============================================================================

全体の実行速度を調整することが出来ます。

セットアップで設定した場合は全部のプログラムが影響を受けます。

個々のプログラムで調整したい場合はWaitコマンドを使用します。

（例）Wait 100

ウエイト値に100を設定します。

===============================================================================

扱える数値について

===============================================================================

純正CasioBasicでは内部15桁、指数2桁の精度を持つ10進数ですが、

C.BasicではSDKのCの仕様と同じ64ビット倍精度実数型と32ビット整数型が基本になります。

倍精度実数型で扱える数値範囲は

±9.88131291682493e-323～±1.7976931348623151e+308

純正よりは広い範囲の数値が扱えますが、演算は2進数で行われるので小数以下の数値を扱う場合は誤差に注意して下さい。

定数値において、C言語同様のプレフィックス(0Xまたは0B）を付けると32ビットまでの16進数、2進数が使用出来ます。

（例）0XFF, 0B10100110

（例）0xAB, 0b10100110

===============================================================================

変数について

===============================================================================

純正CasioBasic同様に1文字変数としてA～Z、r、θまでの28文字に加えてa～ｚまでの小文字変数が使えます。

変数は実行モードの違いに関係なくそれぞれ独立した実数変数（A～Z、r、θ、a～ｚ）と整数変数（A～Z、r、θ、a～ｚ）が確保されますが、

そのままでは変数名の区別ができないので、％サフィックスを付けると整数変数（A%～Z%、r%、θ%、a%～ｚ%）、＃サフィックスを付けると実数変数（A#～Z#、r#、θ#、a#～ｚ#）となります。

実数モードで％サフィックスを付けた整数変数を使う場合は特に気にすることはないのですが、

整数モードで＃サフィックスを付けた実数変数を使う場合は、変数からの読み出し結果はすべて整数の範囲に丸められます。

実数値が整数範囲に収まらない場合は0になります。

（例）'#CBINT //　整数モード指定

　　　1.2345→A# //　実数変数Aに1.2345を代入したつもりが実際には整数の1が代入されます。

（例）A％→A＃

　整数変数のAより倍精度変数Aにコピーします。

　整数から倍精度へのコピーは無問題ですが、倍精度値から整数値にコピーする場合、整数範囲に収まらない場合は0になります。

変数の初期化では純正同様の0→A～Zの形式に加えて、整数、倍精度、個別に指定できます。

0→A～Z＃

倍精度変数を初期化します。

0→A～Z％

整数変数を初期化します。

===============================================================================

拡張変数について

===============================================================================

ver0.35より、自由な名前で使える変数が26個増えました。

アンダーバーより始まる複数英数字名（８文字まで）の変数が使えます。

従来の１文字変数とは独立したエリアに確保される変数ですがプログラム開始時には常に初期化されます。

使える変数は26個までとなっています。実数変数と整数変数は別の値を保持できますが同じ名前で確保されます。

[SHIFT]+[F1]の変数リストで[F1]を押す度に大文字変数、小文字変数、アンダーバー変数が表示されます。

（例）'#CBDBL

（例）123.456->\_ABC

（例）456->\_ABC%

（例）\_ABC+\_ABC%->\_RESULT

　\_RESULTの値は579.456となります。

===============================================================================

定数変数について

===============================================================================

ver0.35より、変数を定数使用限定にするConstコマンドを使用できます。　[OPTN]-[F5](extd)-[F3](Const)

変数代入時にConstを前置すると変数が定数として扱われ以後代入できなくなります。

（例）123->Const A

（例）123->Const \_ABC

（例）456->A

　同じプログラムで変数の値を変更しようとしたらエラーになります。

===============================================================================（CGシリーズ拡張）

画面描画サイズについて

===============================================================================

CGシリーズで標準的に使用出来る表示範囲は384x216ドットです。

純正のCasioBasicで描画可能な範囲は上部のステータスラインを除いた384x192ですが、

C.Basic for CGでは384x216すべてを描画範囲としてサポートしています。

純正互換コマンドでは基本的に384x192範囲が対象ですが一部コマンドはステータスラインも描画対象になります。

（例）Locate 1,0,"STATUS LINE"

（例）Text -24,1,"STATUS LINE"

　ステータスラインに表示します。

===============================================================================（CGシリーズ拡張）

画面の色について

===============================================================================

CGシリーズでは384x216ドットの画面を1ドットあたり16ビット色で表示することが出来ます。

色指定は16ビットの上位より

R:5ビット

G:6ビット

B:5ビット

となっています。

C.Basicでは標準で16ビットカラーをサポートしています。

色指定方法は純正同様にコマンドの前にカラー指定します。

（例）Blue Locate 1,1,"Blue"

拡張コマンドとして

Back-Color

RGB(

GetRGB(

HSV(

GetHSV(

HSL(

GetHSL(

があります。

（例）Back-Color Red

（例）RGB(255,0,128）Blue Locate 1,1,"Blue"

（例）GetRGB(12345)

（書式1）Back-Color　カラー

（例）Back-Color Green

　背景色を緑に設定します。

（書式2）Back-Color　＠カラー

（例）Back-Color @Green

　描画領域外の枠を緑に設定します。

（書式）RGB(R,G,B)

　16ビット色の任意の色を設定できます。

　値の範囲はそれぞれ8ビット値で0～255までとなりますが

　R:5ビット（0～248）

　G:6ビット（0～252）

　B:5ビット（0～248）

　が有効です。

　Red、Green、Blue等の箇所で使えます。

（例）Plot/LineColor RGB(160，160，0）

　Plot/LineColor Yellowよりちょっと暗いYellowです。

（書式）RGB({R,G,B})

（例）RGB({255,255,0})

（例）{255,255,0}->List 1

　　　RGB(List 1)

（書式）GetRGB(16ビットカラー値）

　カラー値をR、G、Bに分解します。結果はListAnsに返ります。

（例）GetRGB( 65535 )

　ListAns = { 248,252,248 }

R = ListAns[1]

G = ListAns[2]

B = ListAns[3]

===============================================================================

画面描画について

===============================================================================

fx-CGシリーズでは画面表示はVRAMに描かれたデータをLCDに転送することで表示するようになっています。

LCDへの転送はかなりのオーバーヘッドがあるので、描画コマンドの実行の度にLCD転送するのは非効率です。

そのためにC.Basicではコマンド終了時に毎回、VRAMからLCDへの転送（＝リフレッシュ）を行わずに

一定の時間が経ってからリフレッシュするように設定出来ます。（セットアップおよびコマンドで可能です）

この時、次回のリフレッシュされるタイミングよりも描画コマンドが早く終了した場合は、

LCD転送されず画面に表示されることなく次の処理に移ってしまうということが起きる場合がありますので

転送漏れを無くすためには区切りのいいところで強制的にLCD転送させるためにPutDispDDコマンドや\_DispVramコマンドを使用して下さい。

===============================================================================

VRAMについて

===============================================================================

C.Basicでは純正CasioBasic同様にテキスト画面とグラフィックス画面がそれぞれ独立して存在していますが、

コマンド等で画面描画に使われるVRAMとしてはひとつだけであり、これを表示用VRAMとします。

さらに

テキストVRAM＝テキストモードでのVRAM内容を保持しているRAMエリア、

グラフィックVRAM＝グラフィックモードでののVRAM内容を保持しているRAMエリア、

を持っています。

つまり3画面分の同じサイズのVRAMが存在しています。

通常のコマンドで描画対象になるのは表示用VRAMです。

例えば、テキスト系の描画コマンドが実行される場合、

現在の画面モードがテキストモードであれば、そのまま表示用VRAMに描画を行います。

グラフィックモードであれば、表示用VRAMをグラフィックVRAMに退避転送した後、テキストVRAMの内容を表示用VRAMに復帰転送して、表示用VRAMに対してテキスト描画を行います。

この後、テキスト系の描画コマンドが続く場合は、表示用VRAMの退避転送は行われず、そのまま表示用VRAMに対して描画を行います。

グラフィック系コマンドが実行された場合は、表示用VRAMをテキストVRAMに退避転送した後、グラフィックVRAMの内容を表示用VRAMに復帰転送して、表示用VRAMに対してグラフィック描画を行います。

このようにテキストVRAMとグラフィックVRAMは直接描画が行われる領域ではなく単なる保存領域となっていて、テキストモードとグラフィックモードを切り替えが起きると、その都度、表示用VRAMとの入れ替えが行われます。

表示用VRAMに描画しただけではLCDに表示されることはなく、LCD転送されて初めてLCD画面に表示されることになります。

===============================================================================

行列の拡張について

===============================================================================

純正CasioBasicのMatrix行列との違いについて説明します。

通常モードでは純正と同じく実数モードでは実数の行列、整数モードでは整数の行列として使用できますが、

C.Basicでは実数（8バイト）および整数（4バイト）の他に、1ビット、ニブル(4ビット)、バイト（1バイト）、ワード（2バイト）、複素数（16バイト）の型を持つ行列が使えます。

行列確保時にサフィックス指定すると要素の型指定ができます。

[.P]または[.p]で（1ビット）整数バイナリー行列。扱える値の範囲は（0～1）です。

[.N]または[.n]で（4ビット）整数ニブル行列。扱える値の範囲は（-8～7）です。

[.B]または[.b]で（8ビット）整数バイト行列。扱える値の範囲は（-128～127）です。

[.W]または[.w]で（16ビット）整数ワード行列。扱える値の範囲は（-32768～32767）です。

[.L]または[.l]で（32ビット）整数ロングワード行列。扱える値の範囲は（-2147483648～2147483647）です。

[.F]または[.f]で（64ビット）倍精度実数行列。扱える値の範囲は（±9.88131291682493e-323～±1.7976931348623151e+308）です。

[.C]または[.c]で (128ビット) 複素数行列。扱える値の範囲は実部・虚部それぞれ（±9.88131291682493e-323～±1.7976931348623151e+308）です。

行列は純正同様に｛ｍ，ｎ｝型で｛row(行）,column（列）｝での実装となります。

例外として1ビット型、ニブル（4ビット）型は｛X,Y}型の実装で｛column（列,）row(行）｝での実装となります。

16ビット行列は画面のVRAMと同じ構成となるので、

[.V]または[.v]指定するとVRAMを216行x384列の行列に割り当てて読み書きが出来ます。

さらに[.VG]でグラフィックVRAM、[.VT]でテキストVRAMを行列に割り当てることが出来ます。

尚、インデックスベースは0から始まり、行列サイズは{216,384}固定となります。

（例）{216,384}→Dim Mat G.V

　Mat Gにアクセスすることでその時点でのVRAMを直接読み書きできますが表示系コマンドではないので、

　画面表示する場合は、コマンド実行後に強制表示させる必要があります。

（例）{216,384}→Mat G.V

　　　Screen.G //　グラフィック画面選択（他のグラフィックコマンドでも可）

　　　0→Mat G[108,192]

　　　PutDispDD //　画面転送コマンド

　　　グラフィック画面の真ん中に黒点を打ちます。（0は黒、白は65535になります）

　　　\_Pixel 84,192

　　　PxlOn (108-24-2),(192-2)と同じです。

（例）{216,384}→Mat G.V

　　　Screen.T //　テキスト画面選択（他のテキスト系コマンドでも可）

　　　0→Mat G[108,192]

　　　PutDispDD //　画面転送コマンド

　　　テキスト画面の真ん中に点が打てます。

===============================================================================

行列の初期化

===============================================================================

純正同様に、

{m,n}→Dim Mat A形式と、

[[1,2,3][4,5,6]]→Mat A形式の初期化をサポートしています。

さらに、fx-5800Pでの

100→Dim A

形式をサポートしています。

0→Dim A

行列Aを消去します。

行列要素の一括初期化として純正同様のFillコマンドと数値→Mat Aという形式をサポートしています。

（例）Fill(123,Mat A)

（例）123→Mat A

Mat A行列の全要素を123で初期化することができます。

Mat行列初期化コマンド[[ ]]において、改行＆空白を入れることが可能です。

（例）

　[[0B11001100,

　　0B00110011,

　　0B11001100,

　　0B00110011

　]]→Mat A.B

Dim Dimでの行列再確保書式において任意にアドレスを行列に割り当てることが出来ます。

（書式） {m,n}->Dim Dim Mat A(行列実体アドレス)

（例）{128,1}->Dim Mat A.B

　{64,1}->Dim Dim Mat B.W(VarPtr(Mat A))

　　Mat AとMat Bは同じ領域をアクセスしますが、バイトアクセスかワードアクセスというところが違います。

アドレス指定で確保した行列には一覧表示で＊マークが付きます。

===============================================================================

行列へのアクセス

===============================================================================

通常のMat A[1,1]形式に加えて、

Matを省略した

A[1,1]

一列目だけのアクセスでは

A[1]

表記も可能です。

さらに添字が数値の0～9の場合のみ、

A0

A5

表記が出来ます。

この表記の変数は実体となる行列が事前に確保されて無い場合は、自動的に行列が確保されます。

（例）A1+123→B5

　もし、Mat AおよびMat Bが確保されていない場合は、変数がアクセスされた時点で

　9→Dim A

　9→Dim B

　と同じ初期化が自動的に実行され、A1～A9およびB1～B9までが使えます。

（例）'#Mat 0

　A1+123→B5

　の場合、もし、Mat AおよびMat Bが確保されていない場合は、

　10→Dim A

　10→Dim B

　と同じ初期化が自動的に実行され、A0～A9およびB0～B9までが使えます。

（例）'#Mat 0

　2→Dim B

　A1+123→B5

　の場合は、

　10→Dim A

　Mat Aのみ初期化され、Mat Bは再確保されないのでB5にアクセスした時点でエラーになります。

　この場合はA0～A9およびB0～B1までが使えます。

===============================================================================

行列の要素サイズ変更

===============================================================================

行列の型(1ビット、ニブル、バイト、ワード、ロングワード、実数、複素数）は行列確保後でも任意に変更できます。

（例）[[1,2,3,4][5,6,7,8]]→Mat A.B

　　　Mat A→Dim Mat A.W

　バイト型のMat A行列をワード型に変更できます。

　要素数は変更後の型サイズに応じて変化します。

　[[0x12,0x34][0x56,0x78]]→Mat A.W

　と同じ結果になります。

上記の例だと{2,4}のバイト行列を{2,2}のワード行列に変換することしか出来ませんが、

以下の形式だと任意の行列サイズに変更できます。

（例）[[1,2,3,4][5,6,7,8]]→Mat A.B

　　　{4,1}→Dim Dim Mat A.W

　[[0x12][0x34][0x56][0x78]]→Mat A.W

　と同じ結果になります。

（例）[[1,2,3,4][5,6,7,8]]→Mat A.B

　　　{1,4}→Dim Dim Mat A.W

　[[0x12,0x34,0x56,0x78]]→Mat A.W

　と同じ結果になります。

（注意）１ビット型、ニブル（4ビット）型行列からの変更、また１ビット型、ニブル（4ビット）型行列への変更は行と列が反転します。

　これは１ビット型、ニブル（4ビット）型行列はX,Y型で実装しているために通常のm,n型行列とは転置行列の関係になります。

　{127,63}→Dim Mat A.P

　LCD画面の全画面ビットマップデータの仮想メモリとして利用できるバッファとして確保します。

　Mat A→Dim Mat A.B

　バイト型行列に変更すると、

　要素サイズは{63,16}になります。

===============================================================================

行列のインデックス開始値について

===============================================================================

純正CasioBasicでは行列のインデックスは１から始まりますが、C.Basicでは0から始まることができます。

セットアップもしくはコメント欄での記述で行列の始まりを選択できます。

（例）#Mat 1

　　従来通り1から始まります。

（例）#Mat 0

　　{3,2}→Dim Mat A

　確保される行列は A[0,0]～A[2,1]となります。

　同時に画面座標も0まで有効になります。これで全画面が描画対象エリアとなります

（例）Pxlon 0,0

===============================================================================

行列の表示について

===============================================================================

Mat行列エディタにおいて2進数、16進数での表示ができます。

行列編集モードで[F5]を押すと2進数表示、[F6]で16進数表示になります。

10進数表示に戻すには再度[F5][F6]を押します。

ただし、2進数表示はバイト（1バイト）と、ニブル（4ビット）、ワードタイプ（2バイト）の行列までです。

[OPTN]キーを押すとリスト名表示に変わります。

~~[VARS]キーを押すと文字列表示に変わります。~~

行列を表示している状態で[OPTN]キーを押すとドットエディタになります。

g1mモード、及びg3mモードの1ビット行列では、行列の各要素の値が0の場合とそれ以外でドットの白黒が決まります。

g3mモードのニブル行列では4ビットカラー、それ以外の型では16ビットカラーとなります。

行列を表示している状態で[VARS]キーを押すと文字列表示に変わります。

行列のデータをクリップボードにコピー＆ペーストが出来ます。

行列編集モード時に[SHIFT]+[8](CLIP)でコピー、[SHIFT]+[9]でペースト出来ます。

（C.Basic）形式なのでエディタにて行列データをペースト出来ます。

===============================================================================

行列のインデックス拡張および最大インデックス値について

===============================================================================

行列のアクセスは純正同様にMat+英１文字以外に、Listと同じく数値で指定できます。

（θ、ｒ、Ansは数値指定できません。）

Mat A = Mat 1

Mat B = Mat 2

Mat C = Mat 3

Mat Z = Mat 26

Mat a = Mat 27

Mat z = Mat 52

さらに＠を付けるとインデックス値を変数で指定することも出来ます。

（例）3->A

Mat @A （Mat Cと同じです）

Listの拡張で行列も同様に最大1040まで拡張されます。

小文字以降の行列はリストと共用しているために27以降の行列＝小文字行列はリストと同じ領域を使用します。

53以降は数値のみ指定できます。

Mat a[5,1] = Mat 27[5,1] = List 27[5]

Mat b[5,1] = Mat 28[5,1] = List 28[5]

Mat c[5,1] = Mat 29[5,1] = List 29[5]

Mat z[5,1] = Mat 52[5,1] = List 52[5]

List 53[5] = Mat 53[5,1]

List 100[5] = Mat 100[5,1]

List 1000[5] = Mat 1000[5,1]

（例）100->A

Mat @A

Mat @AはMat 100、List 100と同じです。

===============================================================================

リスト機能のサポート範囲について

===============================================================================

リスト機能は行列機能を利用して実装しています。

デフォルトでList 1～52まで使えます。

さらに隠しRAM使用時にはセットアップで1040まで増やすことが出来ます。

Listと行列との対応は以下の通りになっています。

Mat A～ZおよびList 1～26は互いに独立しています。

27～52までは小文字の行列と共有します。

List 27[5] = Mat a[5, 1]

List 28[5] = Mat b[5, 1]

List 29[5] = Mat c[5, 1]

List 52[5] = Mat z[5, 1]

List 100[5] = Mat 100[5, 1]

List 1000[5] = Mat 1000[5, 1]

純正機能の以下のコマンドをサポートしています。

List Ans

10->Dim List 1

{1,2,3,4,5}->List 1

Seq( X^2, X, 1, 10, 2 )->List 2

Augment(List1,List2)->List 3

Mat>List( Mat A, 2) -> List 1

List>Mat( List 1, List 2,..) -> List 9

Fill(3,List 1)

Min(List 1)

Max(List 1)

Mean(List 1）

SortA(List 1)

SortD(List 1)

Sum List 1

Prod List 1

Listを含む四則演算および関数演算。

===============================================================================

文字列について

===============================================================================

純正CasioBasicと同じくStr文字変数や文字列関数が使えますが、文字列の実体は行列として実装しています。

プレフィックスとして$を付けることでMat行列は文字列として認識されます。

（例）["ABCDEF"]→Mat A

　　　Locate 3,3,$Mat A

　画面座標(3,3)に"ABCDEF"が表示されます。

　最初に

　{7,1}→Dim Mat A.B

と同じ初期化が自動的にされます。

　そして文字列は行列内ではバイト型データとして格納されます。

・CasioBasic互換で使える文字列関数

+　（文字列同士の連結）

StrJoin(

StrLen(

StrCmp(

StrSrc(

StrLeft(

StrRight(

StrMid(

Exp>Str(

Exp(

StrUpr(

StrLwr(

StrInv(

StrShift(

StrRotate(

（例）"ABCDEF"→Str 1

　　　Locate 3,3.StrRight(Str 1, 3)

　画面座標(3,3)に"DEF"が表示されます。

fx-CG10/20/50で使われる文字は１バイト文字（英数字、一部のコマンド）と２バイト文字（ほぼすべてのコマンド、特殊文字およびカナ）、GB文字（２バイト）、GB文字が有効なときのエスケープ付き特殊文字（２バイト・４バイト）があり、

内部では１バイトと２バイト、４バイト文字が混在する状態となりますが、文字関数での扱いは純正同様にすべて１文字扱いとなります。

文字列が格納された行列要素に直接アクセスする場合は文字の位置に注意して下さい。

===============================================================================

文字列の初期化について

===============================================================================

純正と同じ文字変数Str 1-20はデフォルトではMat行列の<ｒ>を暗黙的に使用します。

デフォルトでは文字数が255文字までとなっています。

Str文字変数は自動的にMat行列として確保され、次の文字列コマンドにおいては純正互換で使えるので、行列を意識する必要はありません。

Str文字列の初期化は何も指定なければ次のようにMat ｒ行列の初期化が自動的に行われます。

　{20,65}→Dim Mat ｒ

なお、Str文字変数の実体となるMat行列は任意指定することも出来、Str変数を使う前にコメント欄で指定します。

（例）'#Str A

　Str変数をMat Aとして確保します。

　{20,65}→Dim Mat A.B

　と同じです。

（例）'#Str B,30,256

　Str変数をMat Bとして30個、255文字まで可能な変数として確保します。

文字列の代入は次の書式が使えます。

（例）"ABCDEEF"→Str 1

（例）["ABC","TEST","1+3+5"]→Mat A

　バイト型のMat A[3,6]行列を自動的に確保してそれぞれの要素に文字列を配置します。

　Mat A[1,1]="ABC"

　Mat A[2,1]="TEST"

　Mat A[3,1]="1+3+5"

　となります。

　文字列対応コマンドで"ABC"にアクセスするには$Mat A、もしくは$Mat A[1],もしくは$Mat A[1,1]となります。

　"1+3+5"をアクセスするには$Mat A[3],もしくは$Mat A[3,1]となります。

（例）Locate 1,1,$Mat A

　Locate 1,1,"ABC"と同じです。

（例）Locate 1,1,$Mat A[3,3]

　Locate 1,1,"3+5"と同じです。

現バージョンでの文字列対応コマンドです。””の代わりに$Mat形式で使用します。

・””

（例）$Mat A[1]

　　"ABC"と同じです。

・？

（例）？→$Mat A

　行列Mat Aに文字列を入力します。確保している要素数で文字数が制限されます。

・Locate

・Text

・LocateYX

・Exp(

（例）Exp("1+2+3")→6

（例）Exp($Mat[3,3])→8

文字列を数式として評価、値を返します。

===============================================================================

拡張文字列コマンドSprintfについて

===============================================================================

（説明）Cのライブラリ関数sprintfのC.Basic版です。

（書式）Sprintf("書式指定子",引数1[,引数2[,引数3]])

※ 書式指定子はANSI Ｃの sprintf と完全互換ではありません。

※ 引数は３個まで指定可能。

※ 引数が整数の場合はプレフィックス%を、実数にはプレフィックス# を指定します。

※ 引数のプレフィックスを省略した場合は、その時点での実行モード(実数か整数)に従います。

(注) Sprintf( は引数チェックをしていないので、間違った設定によりシステムエラーになる

　　 可能性が有ります。

(例) Sprintf("A=%4d B=%3.2f",%A,#B)

※ A=1234, B=45.68 の場合、変換される文字列は "A=1234 N=45.68" となります。

(例) Sprintf("%g%s",#1.234,"sec")→Str 1

※ Str 1 に代入される文字列は "1.234sec" となります。

(例) Sprintf("Result=%08.3f",#123.45)

※ 変換される文字列は、全８桁、小数点以下３桁でセロ詰めで "Result=0123.450" となります。

(例) Sprintf ("%e",#123456.78)

※ 変換される文字列は、指数表示で "1.234568E+05" となります。

(例) Sprintf("%.3e",#123456.78)→Str 1

※ 変換される文字列は、小数点以下３桁の指数表示で "1.235E+05" となります。

◆ 書式指定子："%[フラグ文字][変換文字数指定][.小数点以下文字数指定]<変換指定子>"

(注) C.Basic for FX での書式指定子はCasio SDK準拠、C.Basic for CG は以下のようになっています。

----------------------------------------------------------------------

・ " " の中に記述する

・ %に続いて以下の[フラグ文字]を指定する。指定しなくても良い。

----------------------------------------------------------------------

0 ゼロ詰めを指定するフラグ。左側に余った空白は全て0になる。

空白 符号付き正の数の場合左側に１文字空白を空ける。

－ 左詰めを指定する。(※[F5](CHAR)よりマイナスキャラクタを選択して下さい。)

----------------------------------------------------------------------

・ 続いて[変換文字数指定]で全文字数を指定する。指定しなくても良い。

・ さらに "." に続いて小数点以下文字数を指定する。指定しなくても良い。

・ 最後に以下の<変換指定子>を指定する。これは必ず指定する。

----------------------------------------------------------------------

d １０進数整数に変換する。

o 符号なし８進数に変換する。

u 符号なし１０進数に変換する。

x 符号無し１６進数に変換する。10～16までは小文字 a, b, c, d, f で表示

X 符号なし１６進数に変換する。10～16までは大文字 A, B, C, D, F で表示

b / B 符号なし２進数に変換する。

f 実数に変換する。

e / E 指数に変換する。変換文字数指定は無視され、小数点以下文字数は有効

g / G 実数か指数に最適化して変換する。

s 文字列に変換する。

===============================================================================

コマンドリファレンス

===============================================================================

フルサポート及び一部サポートされているCacio Basic互換コマンド

===============================================================================

?

▲　(Disps)

→

If ～ Then ～ Else ～ IfEnd

Lbl ～ Goto

= != > < >= <=

Dsz / Isz

=>

Locate

Menu

For ～ To ～ Step ～ Next

While ～ WhileEnd

Do ～ LpWhile

Break

Return

Stop

Getkey

Prog

ClrMat

ClrList

Mat　（配列としての実装で行列演算機能は未サポートです。）

List　（Listを含む演算をサポートしています。）

Dim

Fill(

Seq(

Augment(

Mat>List(

List>Mat(

Min(

Max(

Mean(

SortA(

SortD(

Sum

Prod

Swap / \*Row / \*Row+ / Row+

Trn

Identity

i / Arg / Conjg / ReP / ImP / >r∠θ / a+bi

Real / a+bi / r∠θ

Cls ClrText ClrGraph

ViewWindow

StoV-Win / RclV-Win

Plot / PlotOn / PlotOff / PlotChg

Circle

Line / F-Line

Vertical / Horizontal

PxlOn / PxlOff / PxlChg

PxlTest(

Text

SketchNormal / SketchThick / SketchBroken / SketchDot / SketchThin

S-L-Normal / S-L-Thick / S-L-Broken / S-L-Dot / S-L-Thin

G-Connect / G-Plot

CoordOn / CoordOff

GridOn / GridOff / GridLine

AxesOn / AxesOff / AxesScale

LabelOn / LabelOff

DerivOn / DerivOff

BG-None / BG-Pict

FuncOn / FuncOff

S-WindAuto / S-WindMan

DrawGraph (一部サポート）

Graph Y= (一部サポート）

GraphY (一部サポート）

Graph(X,Y)=( (一部サポート）

DrawStat (一部サポート）

S-Gph1、S-Gph2、S-Gph3、DrawOn、DrawOff、Scatter、xyLine、Square、Cross、Dot

Xmin Ymin Xmax Ymax Xscl Yscl Xfct Yfct Xdot

TθMin TθMax TθPtch

Deg Rad Gra

And Or Not Xor

Abs Int Frac Intg

!

GCD( LCM(

MOD(

Rmdr Int÷

Σ(

log ln logab(

10^x e^x

x^-1

sqr x^2

sqr^-3

sin cos tan

sin^-1 cos^-1 tan^-1

sinh cosh tanh

sinh^-1 cosh^-1 tanh^-1

femto pico nano micro milli Kiro Mega Giga Tera Peta Exa

°(deg) r (rad) g (gra)

° >DMS

Pol( Rec(

fn

Str

StrJoin(

StrLen(

StrCmp(

StrSrc(

StrLeft(

StrRight(

StrMid(

Exp>Str(

Exp(

StrUpr(

StrLwr(

StrInv(

StrShift(

StrRotate(

StoPict / RclPict

StoCapt / RclCapt

Ran# RanInt#( RanNorm#( RanBin#( RanList#(

Rnd RndFix(

Norm / Fix / Sci

Eng / EngOn / EngOff

Black Blue Red Magenta Green Yellow Cyan ~~White~~

Plot/Line-Color

ColorAuto / ColorClr

ColorLighter / ColorNormal

ColorLinkOn / ColorLinkOff / ColorLinkX&Y / ColorLinkOnlyX / ColorLinkOnlyY / ColorLinkX&Freq

===============================================================================

コマンドリファレンス　拡張コマンド

===============================================================================

-------------------------------------------------------------------------------

?

-------------------------------------------------------------------------------

？コマンドにオプション追加で入力形式を選択できます。

（書式）?[##]([@][x][,y][,width][,"空白キャラクタ"][,入力桁数][,R][,M])

　このオプションを付けると直前のカーソル位置から改行されません。入力プロンプトの"？"も表示されません。

　各パラメータは省略できます。

[##]オプションによりGB文字の入力が出来ます。

（例）"A="?()->A

　"A="に続いて入力待ちになります。

（例）"A="?(,,,,,R)->A

　"A="に続いて反転表示で入力待ちになります。

（例）"A="?(,,8,,,R)->A

　"A="に続いて反転入力待ちになります。入力範囲は8桁分のみです。それを超える入力では横スクロールします。

（例）?(3,4,5,">",5)A

　X=3，Y=4の位置でAの値を表示して入力待ちになります。入力範囲と入力桁ともに5桁のみ、空白は">"で埋められます。

（例）?(3,4,5,,5,R)A

　X=3，Y=4の位置でAの値を反転表示して入力待ちになります。入力範囲と入力桁ともに5桁のみです。

（例）?(@30,40,5,">",5)A

　現在のVRAMのピクセル座標(X=30,Y=40)の位置でAの値を表示して入力待ちになります。入力範囲と入力桁ともに5桁のみ、空白は">"で埋められます。

（例）?(@30,40,5,">",5,,M)A

　現在のVRAMのピクセル座標(X=30,Y=40)の位置でAの値をミニフォントで表示して入力待ちになります。空白は">"で埋められます。

　ミニフォント入力の場合は[@]オプションが必須です。

-------------------------------------------------------------------------------

Menu

-------------------------------------------------------------------------------

タイトルの色はMenuコマンド直前の色指定が有効になります。

メニュー項目はPlot/Line-Colorの色設定が有効になります。

メニュー背景はBack-Colorで指定している色になります。

コマンド直後に"@"を付けるとFX版互換表示になります。

タイトルやメニュー項目の直前に"##" を付けるとGB文字を表示できます。

-------------------------------------------------------------------------------

Eng,Norm,Fix,Sci

-------------------------------------------------------------------------------

Eng,Norm,Fix,Sciの状態の読み出しが可能です。

Engモードは変数(Eng)で取得できます。

返り値0：EngOff

返り値1：EngOn

返り値3：EngOn 3 (3桁区切り）

Norm、Fix、Sci

はそれぞれ引数に負数を指定すると現在の設定値を返します。

※引数の()は無くても構いません。

現在の設定がNorm1の場合の返り値は、

Norm (-1)で1

Fix (-1)では-1

Sci (-1)では-1

現在の設定が　Fix 8の場合は、

Norm (-1)で-1

Fix (-1)では 8

Sci (-1)では-1

になります。

-------------------------------------------------------------------------------

EngOn

-------------------------------------------------------------------------------

　3桁区切りも可能になります。

（例）EngOn 3

このコマンド以降の数値出力が3桁区切りになります。

　EngOffで通常モードに戻ります。

-------------------------------------------------------------------------------

StoPict/RclPict

-------------------------------------------------------------------------------

ストレージメモリと内部ヒープメモリで動作するモードを切替できます。

セットアップにてストレージメモリとの切り替え動作になります。

ヒープメモリ指定の場合はPictファイルの互換が無くなる代わりに速度向上しました。

Basic動作上の互換性は確保されていますので、純正CasioBasicのソースはそのままで動きます。

-------------------------------------------------------------------------------

ElseIf

-------------------------------------------------------------------------------

（書式）If～Then～ElseIf～IfEnd

（例）

If A:Then

Locate 1,1,"A"

ElseIf B:Then

Locate 1,2,"B"

EndIf

・Ifに対応するIfEndが無い場合にエラーとするかどうかをセットアップで選択出来ます。

　セットアップでチェックをOnにすれば実行時に対応するIfEndが無いIfコマンドの箇所でエラーになります。

-------------------------------------------------------------------------------

Locate

-------------------------------------------------------------------------------

・Locateコマンドの表示で反転表示や重ね合わせ表示が可能です。

　外部フォントが読み込まれている場合は外部フォント表示になります。

（書式）Locate [@][!]csrX,csrX,[##]"文字列”or数式 [,N/O/R/V][,F][,フォント縦ドット数]

　[@]　現在のVRAMに描画します。

　[!]　外部フォントが導入されていても標準内蔵フォントで描画します。

[##] GBフォントを表示します。

　[,N] Normal

　[,R] Reverse

　[,O] Or

　[,V] reVerse or

　[,F] 通常、GBフォントは24ドット表示ですが、18ドット幅固定ピッチで表示します。

（例）Locate 1,2,"Test",R

　　テキストVRAMのテキスト座標(1,2)に"Test"が反転表示で表示されます。

（例）Locate @!3,4,"Test",O

　　現在のVRAMのテキスト座標(3,4)に"Test"が背景に重ねて表示されます。

・Sprintf、Locate、Text, Dispコマンドでの＃プレフィックスは実数式評価となります。

　整数モードでは常に整数値しか得られないので、＃プレフィックスを付けた式では実数式評価で実数値が返ります。

　整数モードでしか意味は無いのですが整数モードにおけるSprintfやLocate、Textコマンドで実数値を扱うことが出来ます。

（例）'#CBINT

10→A

Locate 1, 2, log 123+A

　整数モードでは整数値に丸められて12となってしまうのですが、

（例）'3CBINT

10→A

Locate 1, 2, #log 123+A

　＃プレフィックスを付けると12.089905111と表示されます。

　＃以降の式評価は実数式として行われますが、変数は整数変数のまま扱われます。％を付ける必要はありません。

-------------------------------------------------------------------------------

Switch

Case

Default

SwitchEnd

-------------------------------------------------------------------------------

（説明）Switchコマンドを追加しました。C言語同様フォールスルー可です。ただしDefaultは最後でないと期待した動作になりません。

　　　　新規コマンドはSwitch、Case、Default、SwitchEndとなります。Breakは従来コマンドです。

（例）

Switch A:

Case 1:Locate 1,1,"A"

Break

Case 2:Locate 1,2,"B"

Case 3:Locate 1,3,"C"

Break

Default

Break

SwitchEnd

-------------------------------------------------------------------------------

ACBreak

-------------------------------------------------------------------------------

（説明）プログラム中で[AC]での一時中止と同じ結果となります。

　セットアップにてACBreakコマンドの有効/無効の切り替えが出来ます。

　無効(off)にした場合はスルーされてプログラムの実行に何も影響を及ぼしません。

（書式）ACBreak

（例）ACBreakStop

　直後にStopコマンドを付けることで、ACBreakStopとなりますが、

　これ以降に実行されるACBreakコマンドをすべてキャンセルします。

　セットアップでのキャンセルと同じです。

-------------------------------------------------------------------------------

Local

-------------------------------------------------------------------------------

（説明）デフォルトでは小文字変数は全てグローバル変数扱いですが、任意の小文字変数をローカル変数とすることが出来ます。

　　　　プログラム内でLocalコマンドで変数指定するとその変数はそのプログラム内だけで使われるローカル変数となります。

（例）Local x,y,z

　ローカル変数としてｘ，ｙ，ｚを指定します。

　指定できるのは小文字変数のみで最大10個まで指定できます。

　またこの変数の順で引数付きのProgとGosubでローカル変数に代入されます。

（例）サブプログラム"TEST"内で、

　　　Local x,y,z

　（記述する位置はサブプログラム内であればどこでも可ですが、同じサブプログラム内で複数のLocalがある場合は最後に記述されたLocalのみが有効になります。）

　　　Prog "TEST",123,456,N

　サブプログラム"TEST"の実行時に、ローカル変数としてx,y,zが確保され、

　123→x、456→y、N→z、それ以外はグローバル変数となります。

ローカル変数が使えるようになったことで再帰プログラムも可能になりますが、

現状ではメモリ制限でネスト200層程度までとなっています。

-------------------------------------------------------------------------------

Prog

-------------------------------------------------------------------------------

（説明）Localコマンドを使うことで引数を記述することが出来ます。

（例）

　"MAIN"プログラム

　 10→X

　 20→Y

X+1→a

Y×2→b

　 Prog "SUB",X,Y

　　Disp Ans

Disp a

Disp b

　"SUB"プログラム

　 Local a,b

Dips a

Disp b

　　Return a+b

-------------実行結果

10

20

30

11

40

-------------

　2つの数値の和を計算するサブプログラムを呼び出し結果を表示します。

　サブプログラム内ではa,bはローカル変数として確保され、サブプログラム内ではグローバル変数a,bにはアクセス出来ません。

　メインプログラムのa,bはグローバル変数であり、サブプログラムのa,bの影響を受けません。

-------------------------------------------------------------------------------

Return

-------------------------------------------------------------------------------

（説明）サブルーチンからメインルーチンに戻ります。返り値を指定することができます。

（例）Return　式

　　返り値をAnsに代入してメインルーチンに戻ります。

　計算結果は常にAnsに代入されるので、

　式

Return

　でも同じ結果になります。

-------------------------------------------------------------------------------

Gosub

-------------------------------------------------------------------------------

（説明）単一のプログラム内でサブルーチンが使えます。

（例）Gosub A

Lbl Aから始まるプログラムを実行しReturnコマンドで復帰します。

（例）Gosub A,123,456

　引数もProgコマンド同様に使えます。

（例）

　 10→X

　 20→Y

X+1→a

Y×2→b

　 Gosub A,X,Y

　　Disp Ans

Disp a

Disp b

Stop

Lbl A

　 Local a,b

Dips a

Disp b

　　Return a+b

-------------実行結果

10

20

30

10

20

-------------

　Progコマンドでのプログラム例を、Gosubを使用するプログラムに書き換えたプログラムです。

　a,bはローカル変数として確保され、このプログラム内ではグローバル変数a,bにはアクセス出来ません。

-------------------------------------------------------------------------------

ElemSize(

RowSize(

ColSize(

-------------------------------------------------------------------------------

（説明）行列のサイズを取得する関数です。

（書式）ElemSize( 行列名 )

　１要素のサイズ（ビット）を返します。

（例）　ElemSize( Mat A )

（書式）RowSize( 行列名 )

　{m,n}のm，行サイズを返します。

（書式）ColSize( 行列名 )

　{m,n}のn，列サイズを返します。

-------------------------------------------------------------------------------

MatBase(

-------------------------------------------------------------------------------

（説明）行列のインデックス開始値を取得する関数です。

（書式）MatBase( 行列名 )

（例）MatBase(Mat A)

　行列Mat Aのインデックス開始値を返します。0か１になります。

-------------------------------------------------------------------------------

Getkey

-------------------------------------------------------------------------------

（説明）純正コマンドの機能に加えて、

　　　　キーが押されるまで待つGetkey1（SDKのGetkeyと同じ）

　　　　キーバッファをクリアしてから入力するGetkey2

　　　　さらに一定時間キー入力待ちをするGetkey3

複数キーの同時読み込みをリストで返すGetkeyM

があります。

　　　　キーコードはSDK互換ではなくCasioBasicのGetkey互換です。

~~SH4A版のfx-9860GIIのGetkeyコマンドにてGetkey1～2と同様に[SHIFT]+[OPTN]でのバックライト制御が可能です。~~

~~[SHIFT]+[AC]での電源OFFには未対応です。~~

ver0.50以降、Getkey1/2において、SDKのキーコードを取得することが出来るようになりました。

（書式）Getkey@1

（書式）Getkey@2

[@]オプション追加でSDKのキーコードが返ります。

（例）Getkey3(128)

　コマンド実行時のタイマーを基準として128Ticksカウント（＝1秒）のウエイト期間中にキー入力があれば最後に押されたキーコードが返ります。

（例）Getkey3(128,A)

　Aをタイマーの初期値として現在のタイマー値が128Ticksカウント経つまでキー待ちをします。

（例）GetkeyM->List 1

[EXE]と[F1]を同時押しした場合、 {79,31}が返ります。

※Exec TimeDsp を有効にしている場合、

・Getkey1/2実行中は時間計測は一時停止しますが、設定によりリセット＆リスタートにすることも出来ます。

　セットアップもしくは以下の制御コマンドで設定できます。

（書式）'#GetkeyC

　デフォルトの計測の一時停止、継続モードです。

（書式）'#GetkeyR

　Getkey1/2実行後に計時がリセット＆リスタートします。

-------------------------------------------------------------------------------

IsError

-------------------------------------------------------------------------------

（説明）数式のエラー値を返します。

（書式）IsErorr(数式)

　数式の最初のエラー値を返します。エラーがない場合は0が返ります。

（例）IsError(12+78)

　エラーが無いので0が返ります。

（例）IsError(12+78+)

　Syntax Errorのエラー値 1が返ります。

（例）IsError(12/0)

　Division By Zeroのエラー値 40が返ります。

※エラーに対応する返り値はErrorCode\_list.txtにあります。

-------------------------------------------------------------------------------

Version （Ver.0.45以降）

-------------------------------------------------------------------------------

（説明）現在のバージョンを3桁の数字で返します。

（例）Version

現在のバージョンが0.45の場合、45が返ります。

-------------------------------------------------------------------------------

System

-------------------------------------------------------------------------------

（説明）SysCallでは取得することが出来ないC.Basic独自の内部値を取得します。

引数は、

0:バージョン　Versionコマンドと同じ

1:VRAMアドレス　（純正）162KB

2:テキストVRAMアドレス （C.Basic）162KB

3:グラフィックVRAMアドレス （C.Basic）162KB

4:ワークRAM （C.Basic)　256KB

10:クリップボードバッファ （C.Basic)　32KB

-4:残りのヒープ領域サイズ（KB単位）

-3:確保したヒープ領域サイズ

-2:OSのバージョンの3桁の整数値（OS3.10なら310）

-1:機種判定（CG10なら10、CG20なら20、CG50なら50）

===============================================================================

コマンドリファレンス　拡張グラフィックコマンド

===============================================================================

CG10/20/50シリーズの画面の実座標は

(0,0)-(383,215)が有効ですが、

C.BasicではFX版の(0,0)-(127,63)から3倍にした(0,0)-(383,191)として、

ステータスラインをマイナス方向に伸ばすことによって、

(0,-24)-(383,191)

の座標値が有効になっています。

ただし、純正互換、および純正系コマンドでは

(1,1)-(379,187)の座標値が有効で、

これはCBasicの座標では

(3,3)-(381,189)となっています。

X,Yそれぞれ2ドットずつずれていますので注意して下さい。

今後、純正系と非純正系（ML系）のグラフィックスコマンドを区別するために

純正座標系と、ML座標系という分け方を使います。

===============================================================================

テキスト系、グラフィック系表示コマンドにおいて

コマンドの直後に[＠]を加えるとテキストモードおよびグラフィックモードに関係なく現在のVRAMが描画対象となります。

-------------------------------------------------------------------------------

（例）Locate 2,2,"String"

　　　Text @16,1,"TextString"

　Textコマンドは現在のVRAM（＝テキストVRAM）に描画します。

-------------------------------------------------------------------------------

コマンド後の区切りが[：]コロンの場合は画面更新を抑制します。

描画結果をLCD表示させるにはPutoDispDD等の転送コマンドを使用して下さい。

-------------------------------------------------------------------------------

-------------------------------------------------------------------------------

Line,F-Line,Vertical,Horizontal,Circle,Rect,FillRect　 （純正座標系）

-------------------------------------------------------------------------------

コマンドの末尾に[,C]を加えると消去します。[,X]で反転します。

　（例）FLine 1,1,30,20,X

-------------------------------------------------------------------------------

Text （純正座標系）

-------------------------------------------------------------------------------

（説明）　コマンドの最後に[,R]で反転表示、[,O]でOr表示、[,V]で反転Or表示、[,F]で固定ピッチ表示が出来ます。

　　　　　外部フォントが読み込まれている場合は外部フォント表示になります。

（書式）Text [@][!]Y,X,[##]"文字列”or数式 [,N/O/R/V][,F/M/MB/FM/FMB/FX/X][,フォント縦ドット数]

　　[@]　現在のVRAMに描画します。

　　[!]　外部フォントが導入されていても標準内蔵フォントで描画します。

　[##] GBフォントを表示します。

　　[,N] Normal

　　[,R] Reverse

　　[,O] Or

　　[,V] reVerse or

　　[,F]　固定ピッチ表示になります。（12ドット間隔固定）

（例）Text 20,80,"Test",R

　　グラフィック座標(80,20)に"Test"が反転表示で表示されます。

（例）Text @!20,80,"Test",O

　　現在のVRAMの座標(80,20)に内蔵フォントで"Test"が背景に重ねて表示されます。

（例）Text 10,1,"Simple Text",,F

　固定ピッチ（12ドット間隔固定）でミニフォント表示されます。

（例）Text 10,20,"ABC",,M

　（X:20，Y:10）座標よりミニミニフォントで"ABC"と表示します。

（例）Text 10,20,"ABC",,MB

　（X:20，Y:10）座標よりミニミニボールドフォントで"ABC"と表示します。

（例）Text 10,20,"ABC",,FM

　（X:20，Y:10）座標より固定ピッチのミニミニフォントで"ABC"と表示します。

（例）Text 10,20,"ABC",,FMB

　（X:20，Y:10）座標より固定ピッチのミニミニボールドフォントで"ABC"と表示します。

（例）Text 10,20,"ABC",,X

（例）Text 10,20,"ABC",,FX

　（X:20，Y:10）座標より超ミニフォントで"ABC"と表示します。

（例）Text 10,1,"Simple Text",,,16

　上端から16ドット分のミニフォント表示されます。

　本来は縦18ドットフォントなので下2ドットは表示されません。

（例）Text 10,1,"Simple Text",,F,20

　固定ピッチ（12ドット間隔固定）で上端から20ドット分のミニフォント表示されます。

　本来は縦18ドットフォントなので下に2ドット分の余白が追加されます。

-------------------------------------------------------------------------------

LocateYX （ML座標系）

-------------------------------------------------------------------------------

（説明）グラフィック画面にLocateと同じサイズの文字表示が出来ます。

　　　　LocateのTextコマンド版です。

（書式）LocateYX [@][!]Y,X,[##]"文字列”or数式 [,N/O/R/V][,F][,フォント縦ドット数]

　　　　座標指定はTextコマンドと同じくY座標が先になります。

　　[@]　現在のVRAMに描画します。

　　[!]　外部フォントが導入されていても標準内蔵フォントで描画します。

　[##] GBフォントを表示します。

　　[.N] Normal

　　[.R] Reverse

　　[,O] Or

　　[,V] reVerse or

　　[,F] 通常、GBフォントは24ドット表示ですが、18ドット幅固定ピッチで表示します。

（例）LocateYX 20,80,"Test",R

　　"Test"がグラフィック座標(80,20)に反転表示で表示されます。

（例）LocateYX 20,80,##"TestGB文字列",,F

　　"Test"がグラフィック座標(80,20)に18ドット固定ピッチでGB文字列が表示されます。

-------------------------------------------------------------------------------

Rect （純正座標系）

-------------------------------------------------------------------------------

（説明）矩形を表示します。座標は実座標指定となります。

（例）Rect 1,1,379,189

　　画面いっぱいに枠を描きます。

-------------------------------------------------------------------------------

FillRect　　　　　　　　　　　　　　　　　　　　　　　　　　　　　 （純正座標系）

-------------------------------------------------------------------------------

（説明）塗りつぶした矩形を表示します。

（例）FillRect 1,1,379,189

　　画面を塗り潰します。

-------------------------------------------------------------------------------

ReadGraph( （ML座標系）

-------------------------------------------------------------------------------

（説明）グラフィック画面のビットマップデータをビット単位で読み取ります。

（書式）ReadGraph(px1,py1,px2,py2)→Mat A

　　グラフィック画面の(px1,py1)-(px2,py2)の範囲をMat A行列に取り込みます。

　　行列は事前に確保しておく必要はありません。自動的に適切な行列サイズで確保します。

　　行列の型指定が無い場合は１ビット型になります。

　　WriteGraphコマンドでそのまま使えるデータとなります。

-------------------------------------------------------------------------------

WriteGraph( （ML座標系）

-------------------------------------------------------------------------------

（説明）ビットマップデータを表示します。

　　　　ビットマップデータは予めMat行列に入力しておきます。

（書式１）WriteGraph x,y,dx,dy,Mat A ,P1[,P2]

　　グラフィック画面の座標(x,y)より幅dx、高さdyの範囲にMat Aのデータを描画します。

　　P1はデータの表示前加工指定で[,N]でそのまま加工無し、[,R]で反転、[,M]でメッシュ加工、

　　P2は指定無しでオーバーライト表示、[,A]でAnd表示、[,O]でOr表示、[,X]でXor表示となります。

（例）[[1,2,4,8,16,32,64,128]]→Mat A

　　WriteGraph 80,20, 8,8, Mat A, N

　　グラフィック画面の座標(80,20)より斜めの線[/]が描画されます。

（書式２）WriteGraph x,y,dx,dy,Mat A[m,n] ,P1[,P2]

　　グラフィック画面の座標(x,y)より幅dx、高さdyの範囲にMat Aの[m,n]からのデータを描画します。

（例）[[16,32,64,128][128,64,32,16]]→Mat A

　　WriteGraph 80,20, 4,4, Mat A[1,1], N

　　WriteGraph 80,24, 4,4, Mat A[2,1], N

　　グラフィック画面の座標(80,20)より[<]の図形が描画されます。

　　WriteGraph 80,20, 4,8, Mat A, N

　　でも同じ結果となります。

　ビットマップデータは8ビット単位で左上が基準になります。

-------------------------------------------------------------------------------

DotGet( 　　　　（ML座標系）

-------------------------------------------------------------------------------

（例）DotGet( px1,py1, px2,py2)→Mat A[x,y]

　　画面上の(px1,py1)-(px2,py2)の範囲をMat A[x,y]を始点として取り込みます。

　　指定範囲よりも行列が足りない場合は行列サイズまで取り込みます。

-------------------------------------------------------------------------------

DotPut( 　　　　　　　　（ML座標系）

-------------------------------------------------------------------------------

（例）DotPut( Mat A, x,y, px1,py1, px2,py2)

　　Mat A行列の(x,y)の要素を始点として、画面上の(px1,py1)-(px2,py2)の範囲に点を打ちます。

（例）DotPut( Mat A[x,y], x1,y1, x2,y2)→Mat B

　　Mat A行列の[x,y]を始点として、Mat Bの[x1,y1]-[x2,y2]の範囲にコピーします。

　　Mat Bがコピー範囲が収まる広さを確保していない場合は、Mat Bに収まる範囲でコピーします。

　　グラフィック画面への転送は行いません。

（例）DotPut( Mat A[x,y], x1,y1, x2,y2)

　　Mat A行列の(x,y)の要素を始点として、画面上の(px1,py1)-(px2,py2)の範囲に点を打ちます。

-------------------------------------------------------------------------------

DotTrim( 　　　　（ML座標系）

-------------------------------------------------------------------------------

（例）DotTrim(Mat A, x1,y1,x2,y2)→Mat B

Mat行列内の[X1,y1]-[x2,y2]範囲の0ではないデータの矩形領域をトリミングしてMat B行列にコピーします。

コピーする行列が確保されてない場合は新たに確保されます。

（例）DotTrim(px1,py1,px2,py2)→Mat A

グラフィック画面からトリミング入力、

グラフィックの(px1,py1)-(px2,py2)の範囲のドットが打たれている矩形領域をMat A行列にコピーします。

どちらの場合も、矩形領域サイズ（Mat行列の確保サイズ）はX,Y変数に自動的に入ります。

0ではないデータの矩形領域の右上X,Y座標は小文字変数x,yに自動的に入ります。

-------------------------------------------------------------------------------

CellSum( 　　　　（ML座標系）

-------------------------------------------------------------------------------

（説明）ライフゲーム専用コマンドです。

（例）CellSum(Mat B[X,Y])→C

　　Mat B[X,Y]周辺8ドットの合計値を計算します。

-------------------------------------------------------------------------------

DotLife( 　　　　（ML座標系）

-------------------------------------------------------------------------------

（説明）ライフゲーム専用コマンドです。

（例）DotLife( Mat A, x1,y1,x2,y2)→Mat B

　Mat Aの座標(x1,y1)-(x2,y2)の範囲をライフゲーム演算してMat Bに格納します。

　Mat AとMat Bは同じサイズの行列である必要があります。

　コマンド実行後に(x1,y1)-(x2,y2)の範囲よりもセルが広がった場合、その矩形範囲より±1の座標が整数変数に自動的に入ります。

　整数変数L : 左端座標-1

　整数変数R : 右端座標+1

　整数変数T : 上端座標-1

　整数変数B : 下端座標+1

===============================================================================

コマンドリファレンス　拡張その他コマンド

===============================================================================

-------------------------------------------------------------------------------

KeyRow(

-------------------------------------------------------------------------------

（説明）キーマトリクスのキースキャン結果を返します。

　　　　返り値対応ビット（2進数）

　 　 bit6 bit5 bit4 bit3 bit2 bit1

　 Row--------------------------------------Row

　　09　 F1 F2 F3 F4 F5 F6 09

　　08　 SHIFT OPTN VARS MENU ← ↑ 08

　　07　 ALPHA ^2 ^ EXIT ↓ → 07

　　06　 XTT log ln sin cos tan 06

　　05　 ab/c F<>D ( ) , → 05

　　04　 7 8 9 DEL 04

　　03　 4 5 6 x div 03

　　02　 1 2 3 + - 02

　　01　 0 . EXP (-) EXE 01

　 Row--------------------------------------Row

（例）KeyRow(9)

　　F4とF6を同時に押した場合は、bit3とbit1が1になり、返り値としては

　　2^3 + 2^1 = 10 が返ります。

-------------------------------------------------------------------------------

Ticks / %

-------------------------------------------------------------------------------

（説明）1/128sのタイマー変数です。

（例）0→Ticks

内部タイマーを初期化します。内蔵RTCは変更されません。

　内部タイマーは1/128s毎にカウントアップされる32ビットカウンタで、24時間で元に戻ります。

最大値は24×60×60×128 -1 = 11059199となります。

-------------------------------------------------------------------------------

Ticks% / %%

-------------------------------------------------------------------------------

（説明）1/32768sのタイマー変数です。

（例）0→Ticks%

内部タイマーを初期化します。内蔵RTCは変更されません。

　内部タイマーは1/32768s毎にカウントアップされる32ビットカウンタです。

最大値は2^32-1 = 4294967295 = -1 となります。

-------------------------------------------------------------------------------

TicksWait / %Wait

-------------------------------------------------------------------------------

（説明）1/128s単位で時間待ちをします。

　（TicksとWaitコマンドの連続合わせ技です。）

（例）TicksWait 128

　1秒間待ちます。

（例）TicksWait -128

　前回のTicksWaitコマンド実行時より1秒間待ちます。すでに1秒を越えていた場合は待ち時間無しで次の処理に移ります。

（例）TicksWait -4

　前回のTicksWaitコマンド実行時より4/128秒待ちます。ループ内にこのコマンドを一つ記述するだけで毎秒32回のループが作れます。

-------------------------------------------------------------------------------

Ticks%Wait / %%Wait

-------------------------------------------------------------------------------

（説明）1/32768s単位で時間待ちをします。

　（TicksとWaitコマンドの間に%を挟みます。）

-------------------------------------------------------------------------------

DATE,TIME

-------------------------------------------------------------------------------

（説明）内蔵RTCの設定、読み出しを行います。文字列型の変数です。

（例）"2017/01/17"→DATE

（例）DATE

　2017/01/17 TUE

　と表示されます。曜日は自動計算されます。

（例）"23:59:59"→TIME

（例）TIME

　23:59:59

　と表示されます。

-------------------------------------------------------------------------------

Disp

-------------------------------------------------------------------------------

（説明）一時停止コマンド▲(Disps)と違い一時停止しない表示コマンドです。

　　　　表示ごとに一行スクロールします。

（書式）Disp 式

（例）Disp A+B

-------------------------------------------------------------------------------

BackLight

-------------------------------------------------------------------------------

バックライトの明るさを設定します。

（書式）Backlight 明るさの値

明るさの値：0～511

0～255：サブレベル

　　+256：メインレベル

（書式）+Backlight

現在のバックライトの明るさの値を読み取ります。

-------------------------------------------------------------------------------

RefrshCtrl/RefrshTime

-------------------------------------------------------------------------------

・画面のリフレッシュをコマンドで設定できるRefrshCtrlとRefrshTimeコマンドを追加しました。

（書式）RefrshCtrl 式

（書式）RefrshTime 式

（例）RefrshCtrl 0

　リフレッシュコントロール無し。Off

（例）RefrshCtrl 1

　リフレッシュコントロール有り。Grpモードに設定します。

（例）RefrshCtrl 2

　すべてのリフレッシュコントロール有り。Allモードに設定します。

（例）RefrshTime 5

　リフレッシュコントール時間を5に設定します。

式の中で使うとRefreshCtrlとRefrshTimeコマンドを使うと現在の値を返します。

（例）+RefrshCtrl

　現在のリフレッシュコントロール値を返します。

　式の先頭にRefreshCtrlが来る場合はエラーになります。

-------------------------------------------------------------------------------

Screen

-------------------------------------------------------------------------------

（説明）VRAMアクセス時の対象VRAMを選択し、画面モードを変更します。

（書式1）Screen

　テキスト/グラフィックの切り替えをします。

（書式2）Screen.T またはScreen.t

　テキストVRAMに切り替え、画面モードをテキストモードに変更し、

（書式3）Screen.G またはScreen.g

　グラフィックVRAMに切り替えます。

（書式4）Screen 式

　式の値が0の場合はテキストモード、1の場合はグラフィックモードに切り替えます。

　数式の中でScreenコマンドを使うとテキストVRAMの場合は0，グラフィックVRAMでは１が返ります。

（例）+Screen

　式の先頭にScreenが来る場合はエラーになります。

・画面全体を反転します。

（書式）Screen.R

・論理座標系から物理座標を求めます。結果はListAnsに返ります。

（書式) Screen#X,Y[,Xmin][,Xmax][,Ymin][,Ymax]

（例)Screen#1,1,-6.3,6.3,-3.1,3.1

　　g1mモードの場合、ListAns:{74,22}

　　g3mモードの場合、ListAns:{220,64}

・物理座標系から論理座標を求めます。結果はListAnsに返ります。

（書式) Screen%X,Y[,Xmin][,Xmax][,Ymin][,Ymax]

（例)Screen%74,22,-6.3,6.3,-3.1,3.1

　　g1mモードの場合、ListAns:{1,1}

（書式）Screen.O

　テキスト画面で実行するコマンドの背景色がグラフィックス画面になります。

　つまり、テキスト画面にグラフィックス画面が合成表示されます。

　LCD出力時にテキスト画面＝VRAMは合成後の画面となります。

（書式）Screen.OT

　テキスト画面で実行するコマンドの背景色がグラフィックス画面になります。

　LCD出力時にもテキスト画面＝VRAMは変化しません。

（書式）Screen.N

　通常モードに戻ります。

-------------------------------------------------------------------------------

PutDispDD　/　\_DispVram

-------------------------------------------------------------------------------

（説明）強制的に画面更新をします。

（書式1）PutDispDD

（書式1）\_DispVram

※注意

　PutDispDDと\_DispVramコマンドで転送エリアが異なります。

　PutDispDD：転送範囲(0,0)-(383,191) 　純正互換仕様です。

　\_DispVram：転送範囲(0,-24)-(383,191)　上部ステータスライン分を含みます。

（書式2）PutDispDD@ 開始行、終了行

（書式2）\_DispVram@ 開始行、終了行

　行の範囲は最上部(-24)から最下部(191)までとなります。

　どちらも同様の動作をします。

（例）PutDispDD@ 0,23

Locateでの１行目だけを転送します。

-------------------------------------------------------------------------------

PopUpWin(

-------------------------------------------------------------------------------

（説明）現在のVRAMにポップアップウインドウ枠を描画したり画面退避復帰を行います。

関数としての実装に変更になります。数式の中でも使えます。

（書式1）PopUpWin( n )

　ｎ行分のポップアップを表示させます。表示のみです。

・exit　ポップアップ。

（書式2）PopUpWin(11,"メッセージ1行目"[,"メッセージ2行目"])

　返り値は常に1です。コマンド終了時に画面が復帰します。

[AC]無効の場合は[AC]でも復帰します。

・Yes No　ポップアップ。

（書式3）PopUpWin(10,"メッセージ1行目"[,"メッセージ2行目"])

　返り値が[Yes]:1 [No]:0となります。コマンド終了時に画面が復帰します。

[F1][F6]以外にも、[EXE]でYes、[AC]および[EXIT]でNoとなります。

・画面退避のみ　ポップアップしません。

（書式4）PopUpWin(0)

・退避した画面の復帰のみ。

（書式5）PopUpWin(9)

　画面退避バッファはシステムと共通なので、退避後にポップアップが出るとバッファが上書きされ元に戻らなくなりますので注意して下さい。

-------------------------------------------------------------------------------

FkeyMenu(

-------------------------------------------------------------------------------

（説明）現在のVRAMがグラフィック画面の場合はグラフィック画面に、

現在のVRAMがテキスト画面の場合にはテキスト画面にファンクションメニューを表示、消去します。

”文字列”かアイコンNoを指定します。

”文字列”はセンタリングして表示します。(ver0.45以降）

（書式1）FkeyMenu( n,"文字列")

　現在のVRAMのファンクションnに文字列を表示します。n:1～6

（書式2）FkeyMenu( n,"文字列",R)

　現在のVRAMのファンクションnに文字列を反転表示します。n:1～6

（書式3）FkeyMenu( n,"",C)

　現在のVRAMのファンクションnを消去します。n:1～6

（書式4）FkeyMenu( n,"文字列",U)

（書式5）FkeyMenu( n,"文字列",RU)

　表示位置を1ドット上にします。

（例）FkeyMenu(2,"test")

　2番めのファンクションメニューエリアにtestと表示されます。

（例）FkeyMenu(1,42)

　一番左側のファンクションメニューエリアに白抜きのEDITが描画されます。

-------------------------------------------------------------------------------

Save

Load(

-------------------------------------------------------------------------------

（説明）行列のデータを読み書きをします。

（例）Save "TEST",Mat A

（例）Load("TEST")→Mat A

　Mat A行列の内容を保存、読み込みます。

　ファイル名は8文字まで、拡張子を省略した場合は[.bin]となります。

　現状、アクセスできるのは行列データのみとなります。

　保存形式はヘッダ無し、内部バイナリーデータそのままのデータファイルとなります。

（例）Save "TEST",Mat A[5,1]

　Mat A[5,1]より最後まで保存します。

（例）Load("TEST",16)→Mat A[10,1]

　保存されたデータの16バイト目のデータからMat A[10,1]に読み込みます

-------------------------------------------------------------------------------

IsExist(

-------------------------------------------------------------------------------

（説明）指定されたファイルが存在しない場合は0が返り値となります。ファイルが存在する場合はファイルサイズが返り値となります。

（例）IsExist("/ABC/TEST")

　"/ABC/TEST.bin"が存在すればファイルサイズが、存在しなければ0となります。

　ファイル名は8文字まで、拡張子を省略した場合は[.bin]となります。

（説明2）ファイルリストを取得します

（書式）IsExist("\*.拡張子")->ファイル数

　Mat Ans：拡張子が"bmp"のファイルがAns行列に作成されます。

　リストはアルファベット順にソートされます。

（例）IsExist("\*.bmp")->N

　$Mat Ans[3]

　リストの3番目のファイルを表示します。

　N：ファイル数が返ります。

-------------------------------------------------------------------------------

Delete

-------------------------------------------------------------------------------

（説明）指定されたファイルを削除します。

（例）Delete "TEST.dat"

　"TEST.dat"が存在すれば削除します。

（例）Delete "TEST.dat",1

　削除前に確認します。"TEST.dat"が存在すれば削除します。

　ファイル名は8文字まで、拡張子を省略した場合は[.bin]となります。

-------------------------------------------------------------------------------

Alias （v0.33αより仕様変更）

-------------------------------------------------------------------------------

（説明）英１文字の変数名やラベル名を別名に置き換えて同様にアクセスできるようにします。

変数と行列、ラベルそれぞれに32個までのエイリアスを利用できます。

（書式1）Alias 置き換える前の変数名＝置き換え後の変数名（一文字キャラクター）

（例）Alias A＝α

　　　α+100→α

　　Aliasコマンド実行後は、変数Aをαとして使用できます。

　　αの実体はAなので

　　A+100→Aと同じです。

（例）Alias Mat A＝α

　　　Mat α[1]+100→Mat α[1]

　　Aliasコマンド実行後は、Mat変数Aをαとして使用できます。

　　αの実体はAなので

　　Mat A[1]+100→Mat A[1]と同じです。

（例）Alias Lbl A＝α

　　Lbl α

　　Goto α

　　Gosub α

（書式2）Alias 置き換える前の変数名＝\_+8文字までの英数字

　　アンダーバーより始まる８文字までも自由な英数字名で同様にアクセスできます。

（例）Alias a=\_abc\_de

（例）Alias Mat a=\_abcXY

（例）\_abc\_de+1->\_abc\_de

（例）Mat \_abcXY[1,2]+1->Mat \_abcXY[1,2]

　小文字変数aを\_abc\_de表記で使えます。

　小文字行列aを\_abcXY表記で使えます。

（例）Alias Lbl A＝\_Sub1

　　Lbl \_Sub1

　　Goto \_Sub1

　　Gosub \_Sub1

-------------------------------------------------------------------------------

Wait

-------------------------------------------------------------------------------

（説明）一定のウエイトをかけてプログラムの実行スピードを遅くします。

（例）Wait 100

　ウエイト値に100を設定します。

　SH3機でもSH4機でも同じ程度の速度低下となっています。

　ウエイト値、10で約1/2，100前後で速度1/10になります。（速度低下の程度はプログラムによって異なります。）

-------------------------------------------------------------------------------

ListCmp(

-------------------------------------------------------------------------------

（説明）リスト同士を比較します。""を追加しました。

　リストとリスト、リストと値の比較ができます。

（例）ListCmp({1,2,3},{1,2,3})

　1が返ります。

（例） ListCmp({1,2,3},{1,3})

　0が返ります。

（例） ListCmp({1,2,3},{1,3,2})

　0が返ります。

（例） ListCmp({1,2,3},2)

　1が返ります。

（例） ListCmp({1,2,3},4)

　0が返ります。

===============================================================================

コマンドリファレンス　文字列系拡張コマンド

===============================================================================

-------------------------------------------------------------------------------

StrLen(

-------------------------------------------------------------------------------

@オプションで文字列の実バイトサイズ結果を返します。

（書式）StrLen(@"ABCDE")

（書式）StrLen(@Str1)

-------------------------------------------------------------------------------

StrRepl(

-------------------------------------------------------------------------------

（説明）文字列を置換をします。

（書式）StrRepl(文字列, 検索文字列, 置換文字列 [,検索開始位置])

（例）StrRepl("Hello World","World","Earth") -> "Hello Earth"

（例）StrRepl("abcabc","ab","ff",3) = "abcffc"

（例）StrRepl("abcabc","ab","ff") = "ffcffc"

-------------------------------------------------------------------------------

StrChar(

-------------------------------------------------------------------------------

（説明）繰り返し文字列を返します。

（例）StrChar("ABC",5)

　文字列、"ABCABCABCABCABC"が得られます。

-------------------------------------------------------------------------------

StrCenter(

-------------------------------------------------------------------------------

（説明）文字列を真ん中に配置します。

（使い方）StrCenter("文字列",長さ[,"周辺を埋めるキャラクタ"])

（例） StrCenter("Test",8)

　文字列、" Test "が得られます。

（例） StrCenter("Test",8,"\*")

　文字列、"\*\*Test\*\*"が得られます。

-------------------------------------------------------------------------------

Hex(

-------------------------------------------------------------------------------

（説明）16進数文字列に変換します。

（例）Hex(12345)->Str 1

　"3039"がStr 1に入ります。

　数値に戻すは、

（例） Exp("0X"+Str 1)->A

-------------------------------------------------------------------------------

Bin(

-------------------------------------------------------------------------------

（説明）2進数文字列に変換します。

（例）Bin(12345)->Str 1

　"11000000111001"がStr 1に入ります。

（例）Exp("0B"+Str 1)->A

-------------------------------------------------------------------------------

StrBase(

-------------------------------------------------------------------------------

（説明）任意の進数変換をします。

（書式) StrBase(　N進数文字列, 変換前の進数N, 変換後の進数M)

進数の範囲は2～64までです。

　　　使用キャラクタは{ 0-9 A-Z a-z <theta><pi>}

N進数の文字列からM進数の文字列に変換します。

（例） StrBase("579",15,12) -> "873"

StrBase("100",13,10) -> "169"

StrBase("123",16,3) -> "101210"

StrBase("43981",10,16) -> "ABCD"

StrBase("12A345AFZ",36,10) -> "2999794422815"

StrBase("AZaz",62,10) -> "2520113"

-------------------------------------------------------------------------------

StrSplit( [SHIFT]+[VARS](PRGM)-[F6]-[F6]-[F3](ExStr)-[F6]-[F6]-[F1](StrSplit)

-------------------------------------------------------------------------------

（説明）文字列を指定文字で分割してMatAnsに返します。

（書式）StrSplit("文字列","区切り文字"[,開始位置])

　結果はMatAnsに文字列として返ります。

（例）StrSplit("123,4567,89",",")

　MatAnsの結果は、["123","4567","89"]となり、個々の要素を読み出すに次のようにアクセスできます。

　$Ans[1]="123"

　$Ans[1]="4567"

　$Ans[1]="89"

-------------------------------------------------------------------------------

StrAsc( [Shift]+[VARS](PRGM)-[F6]-[F6]-[F3](ExStr)-[F6]-[F6]-[F2](StrAsc)

-------------------------------------------------------------------------------

（説明）先頭文字のAsciiコードを返します。

（例）StrAsc("Ascii")

　文字列先頭の'A'のASCII文字コード65(0x41)が返ります。

-------------------------------------------------------------------------------

StoCapt/RclCapt　　　　　　　　　　　　　　　　　　　　　（g3mモードは未対応です）

-------------------------------------------------------------------------------

（説明）Captureファイルをストレージメモリ上への読み書きします。最大20までのファイルが利用できます。

（例）StoCapt 20

（例）RclCapt 20

　関連してPictファイルのファイルサイズを2KBから1KBに半減しました。

===============================================================================

コマンドリファレンス　拡張その他

===============================================================================

「’/」以降はコメントではなくコマンドとして実行します。

C.Basic動作時にのみ実行するコマンドを記述できます。

（例）'/Ticks→S

（例）'/Ticks-S→T

　時間を計測します。

-------------------------------------------------------------------------------

'# コマンド

-------------------------------------------------------------------------------

コメント欄で各種設定が出来ます。

'#CBasic

'#CBASIC

'#CBDBL

'#CBdbl

'#CBINT

'#CBint

'#CBCPLX

'#CBcplx

'#G1M

'#g1m

'#G3M

'#g3m

　動作モードを切り替えます。すべて英文字です。

'#Break0 //　[AC]キー無効

'#Break1 //　[AC]キー有効　（デフォルト）

　[AC]キーでの中断を有効無効切り替えます。

'#Mat 0 //　行列のインデックス開始値を0にします。

'#Mat 1 //　行列のインデックス開始値を1にします。（デフォルト）

'#GetkeyC //　Getkey1/2実行後に時間計測を一時停止、リスタートします。（デフォルト）

'#GetkeyR //　Getkey1/2実行後に時間計測がリセット＆リスタートします。

===============================================================================

===============================================================================

コマンドリファレンス　マシン語関連コマンド

===============================================================================

C.Basicではメモリ上のSH3/SH4Aのマシン語プログラムを実行できます。

現状、自由にアクセス出来るメモリは行列として確保したエリアだけとなっています。

行列の先頭アドレスはVarPtr(~~)~~もしくはアドレス演算子＆で求められます。

C言語同様にアドレス演算子＆、間接演算子＊も使えます。

-------------------------------------------------------------------------------

SysCall(~~)~~

-------------------------------------------------------------------------------

（説明）OSで用意されているシステムルーチンを呼び出す関数です。引数は12までです。引数は省略できます。

（書式）SysCall( システムコール番号 [,引数1][,引数2][,引数3][,引数4][,引数5][,引数6][,引数7][,引数8][,引数9][,引数10][,引数11][,引数12] )

（例）Screen.G

　　　SysCall( 0x1D0,1,8,127,63 )→R

　システムコール0x1D0は画面に直線をFX版互換描画するシステムルーチンです。

　1→R4レジスタ、8→R5レジスタ、127→R6レジスタ、63→R7レジスタ、システムコール0x1D0を呼び出し返り値がRに入ります。

-------------------------------------------------------------------------------

Call(~~)~~

-------------------------------------------------------------------------------

（説明）任意のアドレスのマシン語プログラムを実行する関数です。引数は４つまでです。引数は省略できます。

（書式）Call(マシン語アドレス [,引数1][,引数2][,引数3][,引数4] )

（例）Call( 0x8802F000,A,B,C,D )→R

　A→R4レジスタ、B→R5レジスタ、C→R6レジスタ、D→R7レジスタ、0x8802F000番地にあるマシン語プログラムを呼び出し返り値がRに入ります。

-------------------------------------------------------------------------------

Peek(

-------------------------------------------------------------------------------

（説明）メモリのデータを読み出す関数です。型指定[.B][.W][.L][.F][.C]が無い場合はバイト読み出しです。

間接演算子＊で置き換えることが出来ます。

（書式）Peek( アドレス )[.B][.W][.L][.F][.C]

（書式）＊( アドレス )[.B][.W][.L][.F][.C]

（例）Peek(0x8802E000).B→A%

（例）\*(0x8802E000).B→A%

　0x8802E000番地の内容をバイト(1バイト)で読み出し、整数変数A％に代入します。

（例）Peek(0x8802E000).W→A%

（例）\*(0x8802E000).W→A%

　0x8802E000番地の内容をワード(2バイト)で読み出し、整数変数A％に代入します。

（例）Peek(0x8802E000).L→A%

（例）\*(0x8802E000).L→A%

　0x8802E000番地の内容をロングワード(4バイト)で読み出し、整数変数A％に代入します。

（例）Peek(0x8802E000).F→A

（例）\*(0x8802E000).F→A

　0x8802E000番地の内容を倍精度実数(8バイト)で読み出し、変数Aに代入します。

-------------------------------------------------------------------------------

Poke(

-------------------------------------------------------------------------------

（説明）メモリにデータを書き込みます。型指定[.B][.W][.L][.F][.C]が無い場合はバイト書き込みです。

間接演算子＊で置き換えることが出来ます。

（書式1）Poke( アドレス )[.B][.W][.L][.F][.C] , データ

（書式1）＊( アドレス )[.B][.W][.L][.F][.C] , データ

（書式2）データ→Poke( アドレス )[.B][.W][.L][.F][.C]

（書式2）データ→＊( アドレス )[.B][.W][.L][.F][.C]

（例）Poke(0x8802E000).B,A%

（例）A%→Poke(0x8802E000).B

（例）A%→\*(0x8802E000).B

　整数変数A％の値を、0x8802E000番地にバイト(1バイト)書き込みします。

（例）Poke(0x8802E000).W,A%

（例）A%→Poke(0x8802E000).W

（例）A%→\*(0x8802E000).W

　整数変数A％の値を、0x8802E000番地にワード(2バイト)書き込みします。

（例）Poke(0x8802E000).L,A%

（例）A%→Poke(0x8802E000).L

（例）A%→\*(0x8802E000).L

　整数変数A％の値を、0x8802E000番地にロングワード(4バイト)書き込みします。

（例）Poke(0x8802E000).F,A

(例）A→Poke(0x8802E000).F

(例）A→\*(0x8802E000).F

　実数変数Aの値を、0x8802E000番地に倍精度実数(8バイト)書き込みします。

-------------------------------------------------------------------------------

VarPtr(

-------------------------------------------------------------------------------

（説明）変数、行列のアドレスを返す関数です。

アドレス演算子＆で置き換えることが出来ます。

（書式1）VarPtr(変数)

（書式2）＆(変数)

（例）VarPtr(A%)

（例）&A%

　整数変数A％の変数のアドレスが返ります。

（例）VarPtr(Mat A)

（例）&Mat A

　行列Aの先頭アドレスが返ります。

（例）VarPtr(Mat A[20,10])

（例）&Mat A[20,10]

　行列A[20,10]の値が格納されているアドレスが返ります。

-------------------------------------------------------------------------------

ProgPtr(~~)~~

-------------------------------------------------------------------------------

（説明）プログラムのアドレスを返す関数です。

（例）ProgPtr()

　現在のプログラムが格納されているアドレスを返します。

===============================================================================（CGシリーズ拡張）

コマンドリファレンス　超高速グラフィックライブラリコマンド MonochromeLib　CG版

===============================================================================

PierrotLL氏作成のグラフィックライブラリMonochromeLibに基づくコマンドです。

これらのコマンドは現在のVRAMに作用し、それ以外の処理を一切しません。

座標系はML系となり(0,-24)-(383,191)が有効です。

表示するためにはPutDispDDや\_DispVRAMを使いLCD転送する必要があります。

C.Basicの拡張仕様を利用しているのでCasioBasic互換コマンドとは若干使い勝手が違います。

※注意

引数のColor指定は色の指定ではなく描画モードの指定となっています。

色指定は純正同様にコマンドの前に色指定コマンドを置くことで指定します。

Color:-1 描画しません。

Color: 0 描画色で描画します。

Color: 1 背景色で描画します。

Color: 2 現在の画面色を反転します。

Color: 3 描画色と背景色のチェック模様で描画します。

Color: 4 乱数に基づく確率で描画します。

Color: 5 描画色と画面色の間でブレンドします。

Colorの値:10 11　　　12　　　13　　　14 （1ドット描画）

Colorの値:20　　　21　　　22　　　23　　　24 （2x2ドット描画)

　　　　　□□　　■□　　■□　　■■　　■■

　　　　　□□ □□ □■ ■□ ■■

（例）\_FillCircle 64,32,30,21

　Colorの値に４を指定した場合、Colorに続くパラメータは黒ドットとなる確率(0～1),(％0～％100)を指定できます。

（例） \_Rectangle 0,0,127,63,0,0,4,0.5

（例） \_Rectangle 0,0,127,63,0,0,4,%50

　画面全体が半分の確率でランダムに黒ドットで埋まります。

（例） \_Rectangle 0,0,127,63,0,0,4,0.1

（例） \_Rectangle 0,0,127,63,0,0,4,%10

　画面全体が10％の確率でランダムに黒ドットで埋まります。

（例）\_FillCircle 64,32,30,4,0.5

（例）\_FillCircle 64,32,30,4,%50

　円の内側が50％の確率で黒ドットで埋められます。

（例） \_FillCircle #0,0,3,4,0.5

（例） \_FillCircle #0,0,3,4,%50

　円の内側が半分の確率でランダムに黒ドットで埋まります。

-------------------------------------------------------------------------------

-------------------------------------------------------------------------------

・Monochromeライブラリ系のコマンド直後に"#"を付けると論理座標で使うことのできます。

support ML command,

\_Point

\_Pixel

\_PixelTest(

\_Line

\_Rect

\_Horizontal

\_Vertical

\_Circle

\_FillCircle

\_Elips

\_FillElips

\_ElipsInRct

\_FElipsInRct

(例) ViewWindow -6.3,6.3,0,-3.1,3.1,0

Screen.G

\_Circle #0.5,1.2,1.2.1

-------------------------------------------------------------------------------

-------------------------------------------------------------------------------

\_ClrVRAM

（説明）現在のVRAMを消去します。

-------------------------------------------------------------------------------

\_ClrScreen

（説明）現在のLCD画面を消去します。

-------------------------------------------------------------------------------

\_DispVRAM

（説明）現在のVRAMの内容をLCD画面に転送します。

-------------------------------------------------------------------------------

\_Pixel

（説明）VRAMに点を描画します。

（書式）\_Pixel X, Y, Color[,確率/透過率]

（例）\_Pixel 64, 32, 2

　画面中央の点を反転します。

-------------------------------------------------------------------------------

\_Point

（説明）VRAMに四角形を描画します。

（書式）\_Point X, Y, Width, Color[,確率/透過率]

　(X,Y)を中心にWidth幅の四角形を描画します。

（例）\_Point 10, 10, 3, 1

Rect 9,9,11,11と同じです。

-------------------------------------------------------------------------------

\_PixelTest(

（説明）VRAMの一点の状態を返します。

（書式）\_PixelTest(X,Y）

返り値

-1 ：範囲外

0 ：白色

1 ：黒色

※g3mモードでは16ビットカラー値を返します。

現在のVRAM以外にもテキストVRAMとグラフィックスVRAMを選択出来ます。

（例）\_PixelTest(10,20,T)

　テキストVRAMの(10,20)のカラー値を読み取ります。

（例）\_PixelTest(10,20,G)

　グラフィックスVRAMの(10,20)のカラー値を読み取ります。

-------------------------------------------------------------------------------

\_Line

（説明）VRAMにラインを描画します。

（書式）\_Line x1,y1,x2,y2,Color[,確率/透過率][,幅]

（例）\_Line 0,0,127,63,1,,10

　画面を斜めに10ドットの太さの直線が描かれます。

-------------------------------------------------------------------------------

\_Horizontal

（説明）VRAMに水平線を描画します。

（書式）\_Horizontal Y, X1, X2, Color[,確率/透過率]

　X1～X2の範囲のYラインを描画します。

-------------------------------------------------------------------------------

\_Vertical

（説明）VRAMに垂直線を描画します。

（書式）\_Vertical X, Y1, Y2, Color[,確率/透過率]

　Y1～Y2の範囲のXラインを描画します。

-------------------------------------------------------------------------------

\_Rectangle

（説明）VRAMに四角を描画します。

（書式）\_Rectangle X1, Y1, X2, Y2, border\_width, border\_color, Color[,確率/透過率]

　(X1,Y1)-(X2,Y2)の範囲で、境界幅border\_widthの枠を境界色border\_colorで描画し、四角の内側をColorで塗りつぶします。

（書式）\_Rectangle 50, 20, 80, 40, 2, 1, -1

　画面中央に幅２ドットの四角枠を描画します。

（書式）\_Rectangle 50, 20, 80, 40, 0, -1, 3

　画面中央にチェッカー模様の四角を描画します。

-------------------------------------------------------------------------------

\_Polygon

（説明）VRAMに多角形を描画します。

（書式）\_Polygon X座標のint行列へのポインタ, Y座標のint行列へのポインタ, 頂点数, Color[,確率/透過率]

（例）

　[[60, 75, 70, 50, 45]]→Mat X.L // 整数型指定必須

[[20, 30, 45, 45, 30]]→Mat Y.L // 整数型指定必須

Screen.G

\_ClrVRAM

\_Polygon VarPtr(Mat X), VarPtr(Mat Y), 5, 1);　（または　\_Polygon &Mat X, &Mat Y, 5, 1);

\_DispVRAM （またはPutDispDD）

　画面に五角形を描画します。

-------------------------------------------------------------------------------

\_FillPolygon

（説明）VRAMに多角形を描画し中を塗りつぶします。

（書式）\_FillPolygon X座標のint行列へのポインタ, Y座標のint行列へのポインタ, 頂点数, Color[,確率/透過率]

-------------------------------------------------------------------------------

\_Circle

（説明）VRAMに円を描画します。

(書式)\_Circle x,y,radius,Color[,確率/透過率][,width][,角度初期値][,角度終了値][,頂点数]

　(x,y)を中心に半径radiusの円を描画します。

（例） \_Circle 64,32,20,4,0.5,3

3x3dotで50％の確率で描画します。

（例） \_Circle 64,32,20,1,,3,0,90

3x3dotで円弧を描画します。

（例）\_Circle 64,32,20,1,,,0,360,5

　五角形を描画します。

（例） \_Circle 64,32,20,1,,,20,380,5

　開始位置をずらした五角形を描画します。

-------------------------------------------------------------------------------

\_FillCircle

（説明）VRAMに円を描画し中を塗りつぶします。

（書式）\_FillCircle X, Y, R, Color[,確率/透過率]

　(X,Y)を中心に半径Rの円を描画し、Colorで塗りつぶします。

-------------------------------------------------------------------------------

\_Elips

（説明）VRAMに楕円を描画します。

（書式）\_Elips X, Y, Rwidth, Rheight, Color[,確率/透過率] [,回転角(度)]

　(X,Y)を中心に横罫Rwidth、縦径Rheightの楕円を回転角度分傾けて描画します。

-------------------------------------------------------------------------------

\_FillElips

（説明）VRAMに楕円を描画し中を塗りつぶします。

（書式）\_FillElips X, Y, Rwidth, Rheight, Color[,確率/透過率] [,回転角(度)]

　(X,Y)を中心に横罫Rwidth、縦径Rheightの楕円を回転角度分傾けて描画し、Colorで塗りつぶします。

-------------------------------------------------------------------------------

\_ElipsInRct

（説明）VRAMに楕円を描画します。

（書式）\_ElipsInRct X1, Y1, X2, Y2, Color[,確率/透過率]

　(X1,Y1)-(X2,Y2)の範囲に内接する楕円を描画します。

-------------------------------------------------------------------------------

\_FElipsInRct

（説明）VRAMに楕円を描画し中を塗りつぶします。

（書式）\_FElipsInRct X1, Y1, X2, Y2, Color[,確率/透過率]

　(X1,Y1)-(X2,Y2)の範囲に内接する楕円を描画し、Colorで塗りつぶします。

-------------------------------------------------------------------------------

\_Hscroll

（説明）水平方向にスクロールします。

（書式）\_\_Hscroll n

　nドット分スクロールします。

　+で右方向、-で左方向にスクロールします。溢れた分は反対側に出現します。

-------------------------------------------------------------------------------

\_Vscroll

（説明）垂直方向にスクロールします。

（書式）\_\_Vscroll n

　nドット分スクロールします。

　+で↓方向、-で上方向にスクロールします。溢れた分は反対側に出現します。

-------------------------------------------------------------------------------

\_Bmp

（説明）VRAMにビットマップデータを描画します。

（書式）\_Bmp ビットマップint行列へのポインタ, X, Y, width, height [,O/A/X] [,C]

　行列内のビットマップデータを、(X,Y)より横幅width　縦幅heightで描画します。

　[.O] Or

　[.A] And

　[,X] Xor

　[,C] クリッピングした後に描画します。

（例）

　[[60, 126, 251, 253, 253, 255, 126, 60]]→Mat B.B //　バイト行列

　\_Bmp VarPtr(Mat B), 8, 16, 4, 8, O, C

-------------------------------------------------------------------------------

\_Bmp8

（説明）VRAMに8x8のビットマップデータを描画します。

（書式）\_Bmp8 ビットマップint行列へのポインタ, X, Y [,O/A/X] [,C]

　行列内の8x8のビットマップデータを、(X,Y)に描画します。

（例）

　[[60, 126, 251, 253, 253, 255, 126, 60]]→Mat B.B //　バイト行列

　\_Bmp &Mat B, 8, 16, A

-------------------------------------------------------------------------------

\_Bmp16

（説明）VRAMに16x16のビットマップデータを描画します。

（書式）\_Bmp16 ビットマップint行列へのポインタ, X, Y [,O/A/X] [,C]

　行列内の16x16のビットマップデータを、(X,Y)に描画します。

-------------------------------------------------------------------------------

BmpLoad(　/　BmpSave

-------------------------------------------------------------------------------

・BMP形式の読み書きができます。（1ビットモノクロおよび24ビットフルカラーのみ）

（書式）BmpLoad("BMPファイル名"),X,Y,1,透過率（0.0～1.0）

（書式）BmpLoad("BMPファイル名"),X,Y,1,%透過率（%0～%100）

　(X,Y)座標を始点として背景と透過率で合成してロードします。

（書式）BmpLoad("BMPファイル名"),X,Y,0,透過率

　(X,Y)座標を始点として背景だけにロードします。

（例）BmpLoad("TEST"),100,50,1,%50

（例）BmpLoad("TEST"),100,50,1,0.5

　（100，50）座標よりTEST.bmpファイルを透過率50％でロードします。

（例）BmpLoad("TEST"),100,50,0,%25

（例）BmpLoad("TEST"),100,50,0,0.25

　（100，50）座標よりTEST.bmpファイルを透過率25％で背景部分だけにロードします。

（例）BmpLoad("TEST")->Mat A

　"TEST.bmp"のデータを行列Aに読み込みます。

（例）BmpSave "TEST",100,50,200,90

（100，50）座標より（200,90）座標までの範囲を24ビットフルカラーBMPファイル"TEST.bmp"としてセーブします。

　指定可能な座標値は(0,-24)-(383,191)の範囲です。

　（フル画面の保存には1分以上かかる場合があります。）

（例）BmpSave "TEST",Mat A

　行列AのデータをBMP形式で"TEST.bmp"に保存します。

-------------------------------------------------------------------------------

・DrawMat　（DotPutコマンドのMLコマンド仕様です。）

-------------------------------------------------------------------------------

（書式）DrawMat Mat A[px,py], x, y, width, height [,横拡大幅][,縦拡大幅][,color][,確率]

（例） DrawMat Mat A, 0, 0, 128, 64

（例） DrawMat Mat A[10,5], 0, 0, 40, 30, %150, %250 , 4, %50

※行列の指定で＆やVartPtrは必要ありません。

-------------------------------------------------------------------------------

\_BmpZoom

\_BmpRotate

\_BmpZoomRotate

-------------------------------------------------------------------------------

（Planet-CasioよりNinestars氏がMonochromeライブラリに機能追加したプログラムを利用させて頂いています。）

（書式）\_BmpZoom &Mat , x, y, width, height [,横拡大幅] [,縦拡大幅] [,color][,確率]

（書式）\_BmpRotate &Mat , x, y, width, height, 回転角(度) [,color][,確率]

（書式）\_BmpZoomRotate &Mat , x, y, width, height [,横拡大幅] [,縦拡大幅] [,回転角(度)] [,color][,確率]

（例）\_BmpZoom &Mat A, 0, 0, 20, 10, 1.5, 2.5

（例）\_BmpZoom &Mat A, 0, 0, 20, 10, %150, %250 , 4, %50

（例）\_BmpRotate &Mat A, 0, 0, 20, 10, 45

（例）\_BmpZoomRotate &Mat A, 0, 0, 20, 10, 1.5, 2.5, 90

（例）\_BmpZoomRotate &Mat A, 0, 0, 20, 10, %150, %250 , 15, 4, %50

-------------------------------------------------------------------------------

・\_Paint

-------------------------------------------------------------------------------

（説明）領域を塗りつぶします。

（書式）\_Paint x,y,Color[,確率/透過率]

（例）\_Paint 100,100,1

　座標(100,100)を始点にして描画色で塗りつぶします。

（例）\_Paint 100,100,0

　座標(100,100)を始点にして背景色で塗りつぶします。

-------------------------------------------------------------------------------

・Monochromeライブラリ系コマンドに図形の内部の黒ドット数をカウントする"\_Test"コマンドオプションを追加しました。

-------------------------------------------------------------------------------

(サポートコマンド) \_Point \_Line \_Rectangle \_Circle \_Polygon

　サポートコマンドの前に"\_Test"を付けると機能します。

（例）\_Test\_Rectangle 0,0,127,63

　(0,0)-(127,63)エリアにおける黒ドットの数を返します。

（例）\_Test\_Circle 62,32,20

（例）\_Test\_Point 10,10,5

（例）\_Test\_Line 10,10,100,30

===============================================================================（CGシリーズ拡張）

フォントの拡張について

===============================================================================

外部フォントファイルを用意することで、ASCII文字と（カナ+外字）文字で外部フォントが使えます。

　カレントフォルダ、もしくは＠Fontフォルダにある

　以下のフォントファイルをファイルリスト更新時に読み込みフォントが切り替わります。

　これらのファイルが無ければデフォルトの内蔵フォントになります。

　　@LFONTA.bmp　（ASCII標準フォント）

　　@MFONTA.bmp　（ASCIIミニフォント）

　　@LFONTG.bmp　（外字標準フォント）

　　@MFONTG.bmp　（外字ミニフォント）

　　@LFONTK.bmp　（カナ標準フォント）

　　@MFONTK.bmp　（カナミニフォント）

　同梱のデフォルト外部フォントはIPAフォントを使用しています。

　　@LFONTK0.bmp　カタカナフォント（標準）

　　@MFONTK0.bmp　カタカナフォント（ミニ）

　　@LFONTK1.bmp　ひらがなフォント（標準）

　　@MFONTK1.bmp　ひらがなフォント（ミニ）

　　@LFONTG0.bmp　外字サンプルフォント（標準）

　　@MFONTG0.bmp　外字サンプルフォント（ミニ）

これらのフォントをカレントフォルダに置いて、

BmpLoad(@K1

とすれば、それ以降のプログラム実行ではひらがなフォントに切り替わります。

BmpLoad(@K0

でカナフォントになります。

-------------------------------------------------------------------------------

GetFont(

GetFontMini(

-------------------------------------------------------------------------------

（説明）フォントデータを読み出します。

（書式1）GetFont(フォントコード)->Mat A

（書式1）GetFontMini(フォントコード)->Mat A

（例）GetFont(0x41)->Mat A

　フォント"A"のビットマップデータ(18x24)が1ビット行列Mat A[18,24]に入力されます。

（例）GetFontMini(0xFF80)->Mat A

　外字ミニフォントの一番目のビットマップデータ(16x24)が1ビット行列Mat A[16,24]に入力されます。

（書式2）GetFont(”フォントキャラ”)->Mat A

（書式2）GetFontMini(”フォントキャラ”)->Mat A

（例）GetFont("A")->Mat A

　フォント"A"のビットマップデータ(18x24)が1ビット行列Mat A[18,24]に入力されます。

（例）GetFontMini("A")->Mat B

　フォント"A"の~~の~~ビットマップデータ(16x24)が1ビット行列Mat B[16,24]に入力されます。

（書式3）GetFont(@フォントコード)->Mat A

（書式3）GetFontMini(@フォントコード)->Mat A

（書式4）GetFont(@”フォントキャラ”)->Mat A

（書式4）GetFontMini(@”フォントキャラ”)->Mat A

（例）GetFont(@0x41)->Mat A

（例）GetFont(@"A")->Mat A

　内蔵フォント"A"のビットマップデータ(18x24)が1ビット行列Mat A[18,24]に入力されます。

GetFont(、GetFontMini(コマンドの引数の直前に##を置くことでFull設定でのGBフォントデータの取得が出来ます。

通常、GBノーマルフォントは24ドット表示ですが、18ドット幅固定ピッチで取得します。

（書式5）GetFont(##フォントコード)->Mat A

（書式5）GetFontMini(##フォントコード)->Mat A

（書式6）GetFont(##”フォントキャラ”)->Mat A

（書式6）GetFontMini(##”フォントキャラ”)->Mat A

（例）GetFont(##0xA4A2)->Mat A

ver0.52以降、GetFontMini(コマンドにオプション追加でMiniMiniフォントとMiniMiniBoldフォントに対応しました。

（書式7）GetFontMini(フォントコード)M->Mat A

（書式8）GetFontMini(”フォントキャラ”)M->Mat A

（書式9）GetFontMini(フォントコード)MB->Mat A

（書式10）GetFontMini(”フォントキャラ”)MB->Mat A

（例）GetFont("A")M->Mat A

MiniMiniフォント"A"のビットマップデータ(8x10)が1ビット行列Mat Aに入力されます。

（例）GetFont("A")MB->Mat A

MiniMiniBoldフォント"A"のビットマップデータ(9x10)が1ビット行列Mat Aに入力されます。

（説明）現在の外部フォント状況を読み出します。

（書式）GetFont()

（書式）GetMiniFont()

　返り値は整数値でbit2が外部カナ 、bit1が外部外字 、bit0が外部Asciiフォントとなります。

（例）標準サイズの外部カナフォントのみ導入されている場合。

　GetFont()は4が返ります。

　GetMiniFont()は0が返ります。

（例）ミニフォントのAsciiフォントと外字フォントのみ導入されている場合。

　GetFont()は0が返ります。

　GetMiniFont()は3が返ります。

-------------------------------------------------------------------------------

SetFont

SetFontMini

-------------------------------------------------------------------------------

（説明）フォントデータを設定します。

（書式1）SetFont　フォントコード,Mat A

（書式1）SetFontMini　フォントコード,Mat A

（例）SetFont 0xFF80,Mat A

　Mat Aのフォントデータが外字フォントの一番目ののデータとして設定されます。

（例）SetFontMini 0xFF80,Mat A

　Mat Aのフォントデータが外字ミニフォントの一番目のデータとして設定されます。

（書式2）SetFont　”フォントキャラ”,Mat A

（書式2）SetFontMini　”フォントキャラ”,Mat A

（例）SetFont "@",Mat A

　Mat Aのフォントデータがフォント"@"のデータとして設定されます。

（例）SetFontMini "@",Mat B

　Mat Bのフォントデータがミニフォント"@"のデータとして設定されます。

（※指定するフォントコード/フォントキャラはASCII、外字またはカナフォントのみ可能です。）

外部フォントを使用しないように設定出来ます。

（書式）SetFont 0

このコマンド以降、外部フォントを使用出来なくなり初期状態にリセットされます。

（書式）SetFont 1

　このコマンド以降、外部フォントが使用できるようになります。

-------------------------------------------------------------------------------

BmpLoad(@A

BmpLoad(@AL

BmpLoad(@AM

BmpLoad(@G

BmpLoad(@GL

BmpLoad(@GM

BmpLoad(@K

BmpLoad(@KL

BmpLoad(@KM

-------------------------------------------------------------------------------

（説明）フォントデータをBMPファイルから読み込みます。

（書式）BmpLoad(@G[フォントファイル番号])

（書式）BmpLoad(@GL[フォントファイル番号])

（書式）BmpLoad(@GM[フォントファイル番号])

（書式）BmpLoad(@K[フォントファイル番号])

（書式）BmpLoad(@KL[フォントファイル番号])

（書式）BmpLoad(@KM[フォントファイル番号])

（書式）BmpLoad(@A[フォントファイル番号])

（書式）BmpLoad(@AL[フォントファイル番号])

（書式）BmpLoad(@AM[フォントファイル番号])

　フォントファイル番号は0～9が指定できます。

（例）BmpLoad(@A1)

　@LFONTA1.bmp　（ASCII標準フォント）

　@MFONTA1.bmp　（ASCIIミニフォント）

　のフォントファイルが外部ASCIIフォントとして読み込まれます。

（例）BmpLoad(@AL1)

　@LFONTA1.bmp　（ASCII標準フォント）

　のフォントファイルが外部ASCIIフォントとして読み込まれます。

（例）BmpLoad(@AM1)

　@MFONTA1.bmp　（ASCIIミニフォント）

　のフォントファイルが外部ASCIIフォントとして読み込まれます。

（書式）BmpLoad(@GL"ファイル名")

（書式）BmpLoad(@GM"ファイル名")

（書式）BmpLoad(@KL"ファイル名")

（書式）BmpLoad(@KM"ファイル名")

（書式）BmpLoad(@AL"ファイル名")

（書式）BmpLoad(@AM"ファイル名")

（例）BmpLoad(@GL"MYFONTL")

　MYFONTL.bmpが外字標準フォントとして読み込まれます。

（例）BmpLoad(@GM"MYFONTM")

　MYFONTM.bmpが外字ミニフォントとして読み込まれます。

-------------------------------------------------------------------------------

BmpSave @A

BmpSave @AL

BmpSave @AM

BmpSave @G

BmpSave @GL

BmpSave @GM

BmpSave @K

BmpSave @KL

BmpSave @KM

-------------------------------------------------------------------------------

（説明）フォントデータをBMPファイルに出力します。

（書式）BmpSave @G[フォントファイル番号]

　カレントフォルダに外字フォント（標準、ミニ）をセーブします。

　ファイル名は@LFONTG.bmp　（標準）

　ファイル名は@MFONTG.bmp　（ミニ）

（書式）BmpSave @K[フォントファイル番号]

　カレントフォルダにカナフォント（標準、ミニ）をセーブします。

　ファイル名は@LFONTK.bmp　（標準）

　ファイル名は@MFONTK.bmp　（ミニ）

（書式）BmpSave @A[フォントファイル番号]

　カレントフォルダにASCIIフォント（標準、ミニ）をセーブします。

　ファイル名は@LFONTA.bmp　（標準）

　ファイル名は@MFONTA.bmp　（ミニ）

（書式）BmpSave @@G

（書式）BmpSave @@K

（書式）BmpSave @@A

　@Fontフォルダにセーブします。

（書式）BmpSave @GL[フォントファイル番号]

（書式）BmpSave @GM[フォントファイル番号]

（書式）BmpSave @KL[フォントファイル番号]

（書式）BmpSave @KM[フォントファイル番号]

（書式）BmpSave @AL[フォントファイル番号]

（書式）BmpSave @AM[フォントファイル番号]

（例）BmpSave @G2

　@LFONTG2.bmp　（外字標準フォント）

　@MFONTG2.bmp　（外字ミニフォント）

　現在の外字標準フォントと外字ミニフォントファイルを外字フォント2としてセーブします。

（例）BmpSave @GL3

　@LFONTG2.bmp　（外字標準フォント）

　現在の外字標準フォントを外字フォント3としてセーブします。

（例）BmpSave @GM3

　@LFONTG2.bmp　（外字ミニフォント）

　現在の外字ミニフォントを外字フォント3としてセーブします。

（書式）BmpSave @GL"ファイル名"

（書式）BmpSave @GM"ファイル名"

（書式）BmpSave @KL"ファイル名"

（書式）BmpSave @KM"ファイル名"

（書式）BmpSave @AL"ファイル名"

（書式）BmpSave @AM"ファイル名"

（例）BmpSave @GL"MYFONTL"

　現在の外字標準フォントを"MYFONTL.bmp"としてセーブします。

（例）BmpSave @GM"MYFONTM"

　現在の外字ミニフォントを"MYFONTM.bmp"としてセーブします。

===============================================================================

シリアル通信コマンド（純正と同名コマンドですが純正との互換はありません。）

===============================================================================

（書式）Send( 変数[,ボーレート])

（書式）Recv( 変数[,ボーレート])

（書式）Send38K 変数[,転送バイト数]

（書式）Receive38k 変数[,転送バイト数]

（書式）OpneComport38K[,ボーレート][,転送モード]

（書式）CloseComport38K

転送できるデータは変数、行列、リストとなります。

転送元と転送先でデータの型を合わせる必要があります。

Send38K/Receive38kはOpneComport38Kにて[転送モード]に0を指定するとバイナリデータの転送になります。

[ボーレート値]：0=300, 1=600, 2=1200, 3=2400, 4=4800, 5=9600, 6=19200, 7=38400(デフォルト), 8=57600, 9=115200 baud

[転送モード] 0:バイナリ転送モード　　1:データ形式自動判別モード（デフォルト）

Send(とRecv(はコマンド単独で転送出来ますが、Send38K/Receive38kはOpneComport38K/CloseComport38Kが必要です。

（例）Send(A)

（例）Recv(A)

（例）Send(Mat A)

（例）Recv(Mat A)

（例）OpneComport38K,9,0

　　　Send38K List 1

　　　CloseComport38K

（例）OpneComport38K,9,0

　　　Receive38k List 1

　　　CloseComport38K

-------------------------------------------------------------------------------

Beep　　　　　　　　　　　　[SHIFT]+[VARS](PRGN)-[F6]-[F4](I/O)-[F6]-[F5](Beep)

-------------------------------------------------------------------------------

（書式）Beep　[周波数][、継続時間(ms)]

　3Pin出力より指定周波数の矩形波を出力します。

（例） Beep

　1KHzの矩形波を0.5秒間、3Pinより出力します。

（例） Beep 440,1000

　440Hzの矩形波を1秒間、3Pinより出力します。

===============================================================================

セットアップでの設定

===============================================================================

Angle :Deg/Rad/Gra

Complex Mode:Real/a+bi/r∠θ

Draw Type :Connect/Plot

Coord :on/off

Grid :on/off/Line

Axes :on/off/Scale

Label :on/off

Derivative :on/off

Background :None/Pict1～20

Plot/LineCol:Black/Blue/Red/Magenta/Green/Cyan/Yellow/White/16ビットカラー(C.Basic拡張)

Sketch Line :Normal/Thick/Broken/Dot/Thin

~~Angle :Rad/Deg/Grad~~

----------------------------ここまで純正CasioBasic互換

-------------------------------------------------------------------------------

Display : Fix/Sci/Nrm/Eng

小数以下の桁数は15桁まで。0を指定した場合は16桁となります。

Nrmに関してはNrm1とNrm2のみCasioBasic互換10桁表示仕様です。

Nrm1：0.01以下は指数表示

Nrm2：0.000000001以下は指数表示

それ以外は桁数指定となります。

Eng ：ENGオフ、ENGオン(/E)、3桁区切り(/3)の切り替えとなります。

・関連コマンド Eng,Norm,Fix,Sci,EngOn,EngOff

-------------------------------------------------------------------------------

Exp Display : E / Stdx10 / Allx10

　　　　指数表示の記号を選択します。

　　　E: -1.23e99　従来の表示形式です。

Stdx10: -1.23(x10)99 スタンダードフォントのみ切り替わります。

Allx10: -1.23(x10)99 すべてのフォントサイズで切り替わります。

・関連コマンド '#exp 0 //　従来の表示形式です。

'#exp 1 //　スタンダードフォントのみ切り替わります。

'#exp 2 //　すべてのフォントサイズで切り替わります。

-------------------------------------------------------------------------------

SetupRecover : On/Off

すでにセットアップで設定してある上記の純正CasioBasic互換設定をプログラム終了後に復帰するかどうかを設定します。

-------------------------------------------------------------------------------

Command Inpt : C.Basic/Standard

コマンド入力方法をC.Basic(fx-5800P)方式とStandard(fx-9860G)方式を選択できます。

-------------------------------------------------------------------------------

[X]character : [0x90]/[X]

[X,θ,T]キーで入力できる文字を選択します。

-------------------------------------------------------------------------------

EnableExFont : on/off

外部フォントの使用を設定します。

・関連コマンド SetFont 0 //　外部フォント無効

SetFont 1 //　外部フォント有効

-------------------------------------------------------------------------------

Edit ExtFont : On/Off

エディタ上で外部フォント表示を使用します。

-------------------------------------------------------------------------------

Edit GB Font : On/Off/Full

エディタ上でのGBフォントの表示を設定します。

Off:GBフォントを表示しません。通常のモードです。

On:コマンド＆特殊文字と重なるF7xx,F9xx,E5xx,E6xx,E7xxのGBコードを除いて表示します。

（※文字列に0x03A8コードがあればそれ以後の文字列はFull表示になります。文字列コマンドの終了でOn表示に戻ります。）

Full:内蔵のすべてのGBフォントを表示します。（GBコードと重なる一部コマンドがGBフォントに化けます。）

（※この設定はエディタとテキスト変換で有効になります。プログラム内での制御は以下のコマンドを使用して下さい。）

・関連コマンド '#GB0 GBフォントを表示しません。従来のモードです。

'#GB1 以後、コマンド＆特殊文字と重なるF7xx,F9xx,E5xx,E6xx,E7xxを除いてGBコードとして表示します。

'#GB2 以後、内蔵のすべてのGBコードを表示します。

-------------------------------------------------------------------------------

EditFontSize : Standard/Mini/MiniMini/MiniMiniFX /+Fixed pitch/+Gap

表示文字の大きさおよび行間隔を変更します。

-------------------------------------------------------------------------------

Edit LineNum : On/Off

行番号表示を選択します。

-------------------------------------------------------------------------------

EditListChar : List / 反転L / ボールドL

ListとMat表示を短縮表示（1文字の反転とボールド）に切り替えます。

-------------------------------------------------------------------------------

Edit-backCol : Black/Blue/Red/Magenta/Green/Cyan/Yellow/White/16ビットカラー

エディタの背景色を設定します。

-------------------------------------------------------------------------------

Edit-baseCol : Black/Blue/Red/Magenta/Green/Cyan/Yellow/White/16ビットカラー

エディタの基本色を設定します。

-------------------------------------------------------------------------------

E-NumericCol : Black/Blue/Red/Magenta/Green/Cyan/Yellow/White/16ビットカラー

エディタの数値表示色を設定します。

-------------------------------------------------------------------------------

E-CommandCol : Black/Blue/Red/Magenta/Green/Cyan/Yellow/White/16ビットカラー

エディタのコマンド表示色を設定します。

-------------------------------------------------------------------------------

Ed-QuteColor : Black/Blue/Red/Magenta/Green/Cyan/Yellow/White/16ビットカラー

エディタの文字列表示色を設定します。

-------------------------------------------------------------------------------

E-CommentCol : Black/Blue/Red/Magenta/Green/Cyan/Yellow/White/16ビットカラー

エディタのコメント表示色を設定します。

-------------------------------------------------------------------------------

E-LineNumCol : Black/Blue/Red/Magenta/Green/Cyan/Yellow/White/16ビットカラー

エディタの行番号表示色を設定します。

-------------------------------------------------------------------------------

Auto OverClk : on/off

起動時、自動オーバークロックが必要な場合に確認します。

-------------------------------------------------------------------------------

Heap RAM Size : 96KB/117KB/127KB/size

C.Basicで使用する最大ヒープメモリサイズを指定します。

-------------------------------------------------------------------------------

Use Hidden RAM : on/off

隠しRAMをC.Basicで使用するかどうかを設定します。（fx-CG10/20は未対応です。）

fx-CG50では8MBのRAMが搭載されており、そのうち6MBはOS未使用領域となっているためにこれをプログラム領域と行列領域として利用できるようにします。

-------------------------------------------------------------------------------

HiddenRAM Init : on/off

隠しRAM使用時に、Mat行列&ListをC.Basic起動時に初期化するかどうかを設定します。（fx-CG10/20は未対応です。）

初期化しない場合は、Mat行列&Listの結果が消えないで残ります。

（※注意　ただし、CG50では電源オフから１時間でメモリがクリアされてしまうので、C.Basicが再起動します。）

-------------------------------------------------------------------------------

Max Pict No : 20 ～20

CGシリーズでは無効です。20から変更できません。

-------------------------------------------------------------------------------

Max List No : 52 ～1040

隠しRAM使用時に、Listの上限を増やせます。

設定を変更するとPict＆Mat＆Listデータが初期化されます。

-------------------------------------------------------------------------------

AT DebugMode : on / off

[AC]で中断した時にデバッグモードの有効/無効を設定します。

-------------------------------------------------------------------------------

Break Stop :on/off

[AC]キーによる中断の有無。

offの場合は[AC]キーによる中断が出来ません。

・関連コマンド '#Break0 //　[AC]キー無効

'#Break1 //　[AC]キー有効　（デフォルト）

-------------------------------------------------------------------------------

Exec TimeDsp : on/off/%HR/on reset

プログラム終了時にプログラムの実行時間を表示します。

計測開始はプログラムの最初の行からですが、途中？コマンドや一時停止コマンドがあった場合はそれまでの計測時間はリセットされます。

Getkey1/2の場合は計測が一時停止になり、コマンド実行後に再スタートします。

on resetは、Getkey1/2実行後に計測がリセットされ再スタートします。

%HR:計測に1/32768sタイマーを使います。

・関連コマンド '#GetkeyC //　Getkey1/2実行後に時間計測を一時停止、リスタートします。（デフォルト）

'#GetkeyR //　Getkey1/2実行後に時間計測がリセット＆リスタートします。

-------------------------------------------------------------------------------

IfEnd Check : on/off

IfとIfEndが一対一で対応しているかどうかをチェックします。

-------------------------------------------------------------------------------

ACBreak : on/off

ACBreakコマンドの有効/無効を切り替えます。

・関連コマンド ACBreak

-------------------------------------------------------------------------------

Force Return : None/ F1/ EXE/ F1&EXE

プログラム実行中の中断でエディタに戻るかファイルリストに戻るかを選択できます。

-------------------------------------------------------------------------------

Key 1st time : 25ms～1000ms(デフォルト値 500ms)

リピート開始までの時間 (25ms単位)

Key Rep time : 25ms～500ms(デフォルト値 125ms)

リピート継続時間 (25ms単位)

-------------------------------------------------------------------------------

SkipUp/Down : 1～9999

SkipUp/SkipDownでの一回分の改ページ数を指定します。

※改行ごとに一行増える論理行ではなく画面上での見た目上での物理行数です。

-------------------------------------------------------------------------------

Mat Dsp mode : [m,n]/[X,Y]

行列エディタ画面での表示形式を指定します。

それぞれ転置行列の関係になりますが、行列内部は何も変化なく表示上だけの指定です。

Matrix base : 0/1

行列インデックスの初期値を0か1が指定します。

初期値に0を指定した場合は、グラフィック座標が左上端の(0,0)から有効になります。

・関連コマンド '#Mat 0 //　行列のインデックス開始値を0にします。

'#Mat 1 //　行列のインデックス開始値を1にします。（デフォルト）

-------------------------------------------------------------------------------

Pict mode : S.Mem/Heap/Both

(S.Mem/SD)　Pictファイルをストレージメモリ上に作ります。若干時間がかかります。

(Heap)　Pictファイルとしては作成されずにメモリ上に互換ファイルが作成されます。（推奨設定です）

(Both)　HeapモードでもPictファイルが常にストレージメモリ上に作成されます。

-------------------------------------------------------------------------------

DATE : 2017/01/17 MON

TIME : 23:59:59

内蔵RTCを設定します。曜日は自動設定されます。

・関連コマンド DATE、TIME

-------------------------------------------------------------------------------

Storage mode : S.Mem/Main Mem

ファイルの保存先、ファイル操作の対象を選択します。

-------------------------------------------------------------------------------

Auto file save : on/off

編集後、自動的に保存するかどうか設定します。

-------------------------------------------------------------------------------

Force g1m save : on/off

テキストファイル編集／実行後に同時にg1m形式でも保存するかどうかを設定します。

-------------------------------------------------------------------------------

Direct GB cnvt : on/off

プログラム内のGBコードをテキスト変換無しでそのままテキストファイルに保存するかどうかを設定します。

（※変換無しでテキストファイルにした場合はGB2312対応エディタではそのまま表示できます。）

（※変換対象GBコードはエディタのGBコード表示モードに対応しています。）

-------------------------------------------------------------------------------

RefrshCtl DD : off/Grp/All

描画コマンドにおける画面の更新を設定できます。

Off：リフレッシュコントロール無し。表示においては純正互換仕様です。

Grp：グラフィックコマンドのみリフレッシュコントロール有り。ClrText,Locate,Text,LocateYX、””は無しで以前のバージョン互換です。

All：すべての描画コマンドがリフレッシュコントロールされた表示となります。必要に応じてPutDispDDコマンドが必須となります。

Time：リフレッシュコントール時間を設定します。n/128秒単位で画面更新します。デフォルトは3で約1/42秒での更新となっています。

　　　0を指定するとリフレッシュされなくなります。

・関連コマンド RefrshCtrl/RefrshTime

-------------------------------------------------------------------------------

Wait count : 0～9999

C.Basic全体の実行速度を調整します。

・関連コマンド Wait

-------------------------------------------------------------------------------

G1M/G3M mode : Auto/g3m/g1m

デフォルトのプログラムの実行モードを切り替えます。

・関連コマンド '#G1M

'#g1m

'#G3M

'#g3m

-------------------------------------------------------------------------------

Execute mode : DBL#/INT%/CPLX

デフォルトのプログラムの実行モードを切り替えます。

・関連コマンド '#CBasic

'#CBASIC

'#CBDBL

'#CBdbl

'#CBINT

'#CBint

'#CBCPLX

'#CBcplx

===============================================================================