Serenade

Version 2
プログラマーズガイド

2004/02/14　末井真一郎

目次

１．はじめに

Serenade（セレナーデ）は画面にベクトル描画を行なうためのダイナミックリンクライブラリー（DLL）である．使用できるルーチンはカルコンプ社のプロットルーチンのサブセットになっている（表１）．Windowsの面倒な描画手続きをプログラマーに意識させないことを目的としたので，低水準の描画ルーチンのみのサポートとした．高機能とは言えないがコンパクトで簡単な使用手順を特徴としている．

ルーチン名	意味	備考
plsize	描画ページの開始を指示する	
plexit	描画ページの終了を指示する	
plot	ペンを移動する	
newpen	ペンの種類を変える	
plfont	フォントを変える	拡張機能
symbol	文字列を表示する（左詰め）	
mdsymb	文字列を表示する（中揃え）	
rtsymb	文字列を表示する（右詰め）	拡張機能
number	数値を表示する	
genten	原点を変更する	
factor	拡大率を指定する	
circle	円を描く	
dash	破線を描く	
pladdp	パスに座標を追加する	拡張機能
plclpt	パスを閉じる	拡張機能
plrgbp	パス内に色を塗る	拡張機能
plhalt	ユーザーのアクションを待つ	拡張機能

表１．サポートルーチン一覧

【必要環境】

OS	Windows 95/98/ME/NT4/2000/XP
CPU	Pentiumとその上位互換CPU（Athlon含む）
メモリー	64MB以上
ハードディスク	10MB以上の空き領域が必要

これまでのVersion 1はカスタマイズ版も含めてこれまでに多くのプログラムに使用されてきたが，メモリーの使用効率が悪いという根本的な問題により機能向上が困難であった．今回のバージョンアップでは画面表示ロジックを一新してメモリーの使用効率を改善した．旧バージョンとの相違点は表２のとおりである．

	Version 1	Version 2
描画サイズ	面積が A4 相当以下	面積が A0 相当以下
最大ページ数	A4 サイズ 20 ページ程度でパフォーマンス低下	A4 サイズ 100 程度ではパフォーマンスの低下はほとんどなし
拡大縮小	50%，100%，200%	制限なし
ファイル出力	PostScript，BMP（180dpi 固定）	PostScript，BMP，JPEG（BMP/JPEG は解像度指定可）
フォント	MS P 明朝	複数フォント使用可能（フォントの割当はユーザーが変更可能）
ペンの種類	８色（割当は内部固定，太さは変更不可）	色と太さの組み合わせで 10 種類（割当はユーザーが変更可能）
ペイント機能	なし	あり

表２．Version 1との主な相違

本書には「Serenadeをリンクしたプログラムの使い方」と「Serenadeをリンクするプログラムの作成方法」を記述する．以降では「エンドユーザー」と「プログラマー」という表現を意識的に区別して使用するが，「エンドユーザー」とはSerenadeをリンクして作成したアプリケーションの使用者のことで，「プログラマー」とはSerenadeをリンクしたアプリケーションの作成者のことである．

２．Serenadeをリンクしたプログラムの使い方

一般的にはSerenadeをリンクしたプログラムは実行が終わるとウィンドウを表示したままエンドユーザーからのアクションを待つアイドル状態になる（図１）．

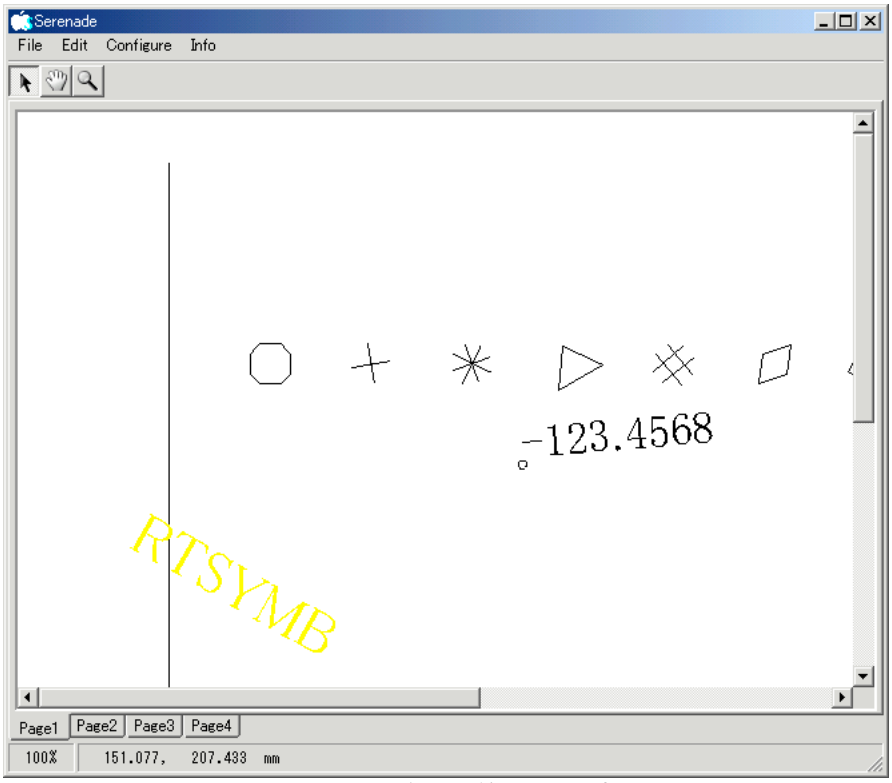


図１．プログラム終了時の表示

エンドユーザーはこのとき「表示エリアの操作」，「ファイル出力」，「ペンスタイル／フォントの変更」，「印刷」が行なえる．

２．１　表示エリアの操作

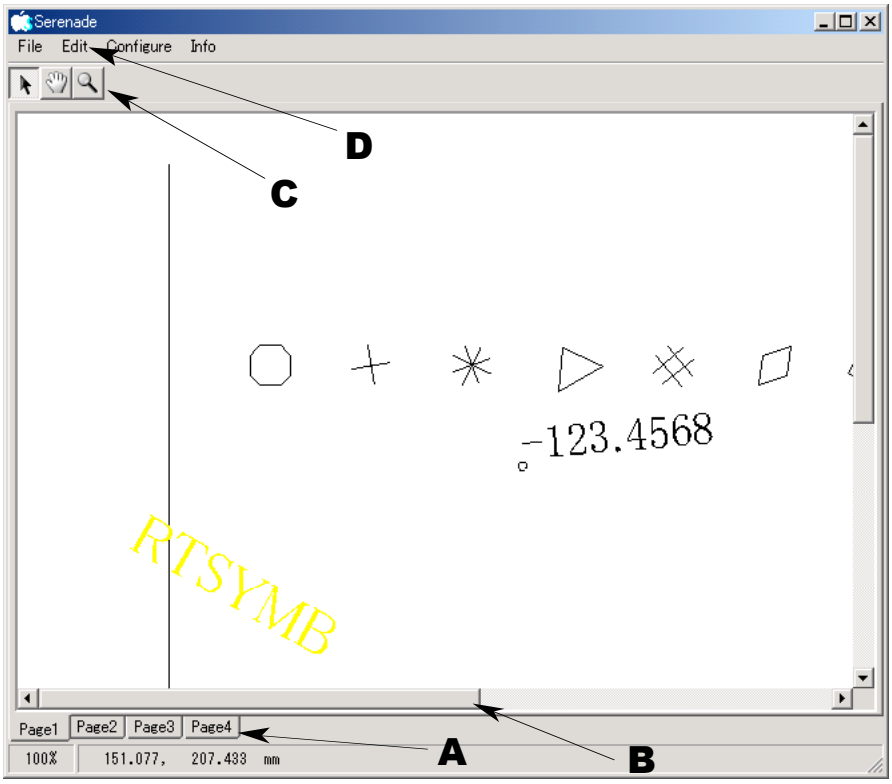


図２．表示エリアの操作

２ページ以上の表示内容がある場合は画面下のタブ（**A**）をクリックして表示したいページを表示させることができる．また，各々のページではスクロールバー（**B**）が使用できる．ツールボタン（**C**）を使うと次の操作が可能．

(1)選択ツール 

・表示画面内の任意の矩形領域を選択できる．

・選択された矩形領域は青い破線枠で示される．

・画面全体を選択する場合にはEditメニュー（**D**）のSelect Allコマンドを選択してもよい．

・選択された状態ではEditメニュー（**D**）のCopyコマンドが使用できる．これは選択領域のグラフィックをクリップボードにコピーする．

(2)ドラッグツール 

・グラフィックをつかんでドラッグすることでウィンドウ内の可視領域を変更することができる．

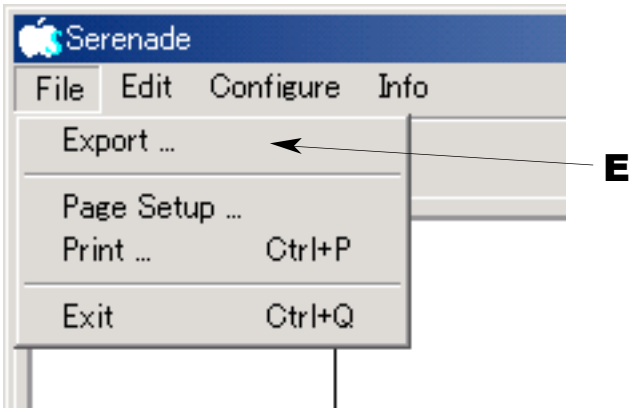
(3)拡大・縮小ツール 

・クリックすると拡大される．

・シフトキーを押した状態でクリックすると縮小される．

・ドラッグして領域を選択すると選択した領域が拡大表示される．

２．２　ファイル出力



FileメニューのExport Fileコマンド（**E**）でファイル出力ができる．

(1)ビットマップファイル（拡張子BMP）

選択されているページのグラフィックを指定された解像度のビットマップファイルにして保存する．Serenade自体には指定できる解像度の制限はないが，実装メモリーや使用できるリソースエリアの大きさで実際に指定できる解像度が制限される．

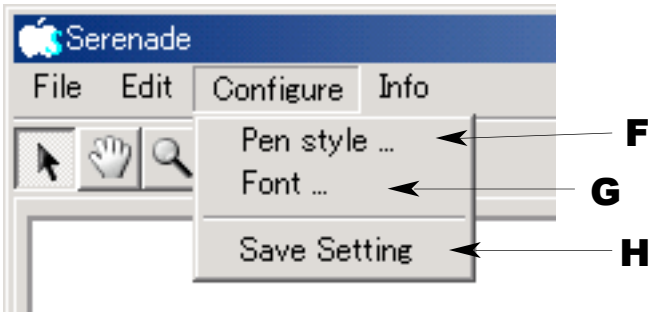
(2)JPEGファイル（拡張子JPG）

選択されているページのグラフィックを指定された解像度のJPEGファイルにして保存する．Serenade自体には指定できる解像度の制限はないが，実装メモリーや使用できるリソースエリアの大きさで実際に指定できる解像度が制限される．

(3)ポストスクリプトファイル（拡張子PS）

選択されているページかあるいは全てのページをポストスクリプトファイルにして保存する．ビットマップファイルやJPEGファイルと異なって解像度に依存しない形式なので可搬性はもっとも優れた形式である．

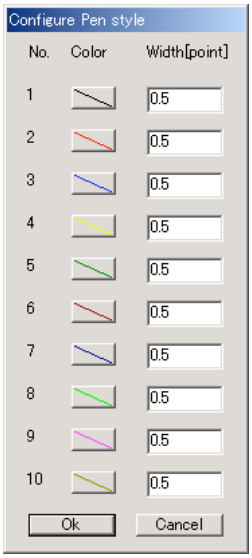
２．３　ペンスタイル／フォントの変更



Configureメニューではペンスタイル（**F**）とフォント（**G**）の設定変更が行なえる．変更した設定は保存（**H**）することができる．

(1)ペンスタイル

次のような画面でペンの色と太さを設定する．太さの単位はpoint[1] である．



No.とはSerenadeをリンクしたアプリケーションが指定したペン番号のことである[2]．ペン番号を決めると色と太さが一意に決まる規則になっている．したがってエンドユーザーが任意の線の色や太さを変更できるわけではない．エンドユーザーはアプリケーションが使用しているペン番号を知っている必要がある．ペン番号は文字の色にも影響する[3]．

色の変更はボタンをクリックして選択画面を使用する．太さの変更はテキストボックス内の数値を直接変更する．

---

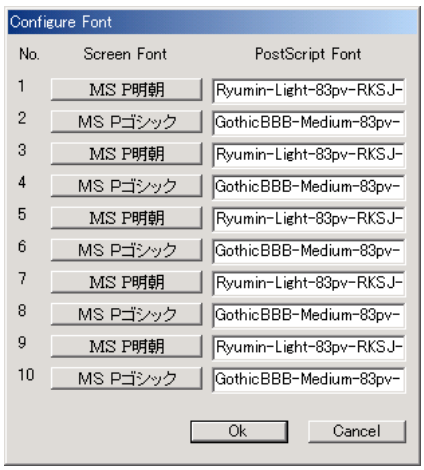
[1] 1 point は 1/72 インチ．

[2] 正確に言うとサブルーチン newpen で指定した番号のことである．

[3] 太さは文字には影響を及ぼさない．

(2)フォント

次のような画面でフォントを変更する．



No.とはSerenadeをリンクしたアプリケーションが指定したフォント番号のことである[1)]．通常「MS明朝」などのスクリーンフォントはPostScriptでは使用できないので画面用とPostScript用を別々に指定できるようになっている．したがってPostScriptファイル出力を行なわないエンドユーザーはPostScript Fontは無視してよい．スクリーンフォントは画面表示のみならずBMP／JPEGファイル出力にも使用される．エンドユーザーがフォントの変更を行なうためにはアプリケーションが使用しているフォント番号を知っている必要がある．

スクリーンフォントの変更はボタンをクリックして選択画面を使用する．PostScriptファイル用フォントの変更はテキストボックス内のフォント名を直接変更する．

---

1) 正確に言うとサブルーチン plfont で指定した番号のことである．

２．４　印刷

(1)用紙設定

FileメニューのPage Setupコマンドでプリンターの用紙設定ができる．設定内容は使用するプリンターによって異なる．

(2)印刷

FileメニューのPrintコマンドで任意のページの印刷ができる．

３．Serenadeをリンクするプログラムの作成方法

３．１　概要

(1)Serenadeを使ったプログラムの作成に使用するファイル

Serenade.dll	ライブラリー本体
Serenade.lib	インポートlibファイル（Microsoft形式[1]）
Serenade.inc	Fortran用関数宣言ファイル[2]
Serenade.bas	Visual BASIC用関数宣言ファイル
Serenade_export.h	C++用関数宣言ファイル

このうちエンドユーザーに配付してもよいファイルはSerenade.dllだけである．他のファイルはエンドユーザーに配付してはならない．

(2)引用方法

Serenadeの関数は機能によって次のような分類ができる．

・ページコントロール関数（描画ページの使用開始と使用終了などを行なう）

plsize，plexit，plhalt

・線描画関数（描画ページに線を引いたり文字を描いたりするもの）

plot，symbol，mdsymb，rtsymb，number，circle，dash

・面描画関数（描画ページの指定領域を塗りつぶすもの）

plrgbp，pladdp，plclpt

・ペン／フォント変更関数

newpen，plfont

・原点／スケールファクター設定関数

genten，factor

・ユーザーインターフェース関数

plmsg，plread

上記の関数群のうち「ユーザーインターフェース関数」だけは特別な手順なしにどのような状況でも引用可能である[3]．その他の関数は，「ページコントロール関数」によって描画ページが有効になっている状態でのみ引用可能である．

描画ページを有効にするにはplsizeを引用し，描画が終わったらplexitを引用して描画ページを無効にする．ある描画ページが有効になったままで別の描画ページを有効にすることはできない．

---

[1] C++ Builder や Delphi のような Borland 製品の統合開発環境で使用する場合は implib コマンドで Borland 形式の lib ファイルを作成しなければならない．

[2] MS Fortran，DEC Fortran，Compaq Fortran などで使用可能．

[3] 「ユーザーインターフェース関数」は Serenade の本来の機能とは関係がなく，Fortran で作成された Win32 アプリケーションに簡単な入出力画面を提供する目的で追加してある．

バッチ型のプログラムの場合に限り，プログラムの終了直前にplhaltを引用する必要がある．plhaltはプログラムの進行を保留し，エンドユーザーからSerenadeに対する操作が行なえるようにする．バッチ型のプログラムでplhaltの引用を忘れると，エンドユーザーが描画ページを見る暇もなくプログラムが終了してしまう．注意が必要なのは，今のところplhaltの引用以降のプログラムコードを実行する手段がないということである．したがってplhaltはプログラム終了直前に引用されなければならない．

表３に正しい引用の例を示す．

流れ	描画ページ	引用可能な Serenade 関数
	無効	ユーザーインターフェース関数のみ
	無効→有効	plsize
	ページ１有効	ユーザーインターフェース関数 ページコントロール関数以外の Serenade 関数
	有効→無効	plexit
	無効	ユーザーインターフェース関数のみ
	無効→有効	plsize
	ページ２有効	ユーザーインターフェース関数 ページコントロール関数以外の Serenade 関数
	有効→無効	plexit
	無効	…
	無効	plhalt
	無効	ここに記述されたプログラムは実行されない

表３．ページコントロールの概念

「ペン／フォント変更関数」を使ってペン番号／フォント番号を変更すると次に変更されるまではその番号を使って描画が行なわれる．

(3)座標系

座標系は左から右，下から上に向かって座標が大きくなる直交座標系である．

(4)面描画関数に関する補足

特定の領域に色を塗る場合は「リージョンの作成」と「塗りつぶし」の二つの手順を行なう必要がある．リージョンとは実際に塗りつぶしたい領域を表現するもので，一個以上の閉じた折れ線（パス）で構成される．

・リージョンの作成方法

①新規パスを作成しそのパスを構成する座標を指定する（pladdp）．パスを閉じるまではいくつでも座標追加が可能である[1]．pladdpで指定した座標が新規パスなのかカレントパスへの追加なのかは

[1] 閉じる前の座標追加が可能なパスをカレントパスと呼ぶ．

Serenade内部で自動判断する．

②カレントパスを閉じてパスを確定する（plclpt）．

上記の①②を繰り返すことでいくつでもパスを作成することができる．このようにして塗りつぶしたい領域を一個以上のパスで指定していくことでリージョンが確定する．

確定したリージョンに対していよいよ塗りつぶしを行なう．これは単純に塗りつぶし関数（plrgbp）を引用するだけである．このときリージョン全体が指定した色で塗りつぶされる．

・リージョンの内部判定について

複数のパスでリージョンを構成している場合や一つのパスでも複雑に折れ線が交差している場合は，リージョンの内部かどうかの判定は非常に重要である．Serenadeではワインディング規則を使って内部判定を行なっているので，各パスが右回りか左回りかはプログラマーが正確に把握していなければならない．

(5)エラー処理

実行効率の低下を避けるために，プログラマーがSerenadeの規則に従わない引用を行なった場合のエラー処理は行なっていない．たとえばサポートしている単位以外の単位を使って描画ページのサイズを指定するとプログラムが暴走する可能性が高い．エラー処理の不備を指摘するより呼び出し側プログラムのデバッグに専念するほうが得策である．

３．２　関数リファレンス

次ページ以降に関数仕様の詳細を記述する．引用例はFortranとVBの場合どのような書式になるかを示しただけなので引数名や数値に深い意味はない．特に記述していない共通の事項は次のとおりである．

◇長さの単位はすべてplsizeで指定した単位となる．

◇引数の説明に《出力》という記述があるもの以外はすべて参照されるだけの入力引数である．

PLSIZE

```
void plsize(double *x, double *y, char *unit, int n)
```

【引数】

double *x	描画ページ横サイズ
double *y	描画ページ縦サイズ
char *unit	サイズの単位（"mm"，"cm"，"m"，"in"，"ft"）
int n	サイズ単位の文字数

【機能】

・指定したサイズの描画ページを新規に作成し有効にする.

・有効にした描画ページの座標指定の単位はunitになる.

・ペン番号／フォント番号はどちらとも 1 に初期化される.

・原点は（0，0）に初期化される.

・拡大率は 1 に初期化される.

・単位指定では大文字／小文字の区別はしない.

【制限】

・有効な描画ページがある状態でplsizeを引用してはならない.

・対応していない単位を指定してはならない.

【引用例】

Fortran	call plsize(200.0d0,100.0d0,'mm')
VB	Call plsize(200#, 100#, "mm", 2)

PLEXIT

```
void plexit()
```

【引数】なし

【機能】

・有効な描画ページを終了して無効にする.

【制限】

・有効な描画ページがない状態でplexitを引用してはならない.

【引用例】

Fortran	call plexit
VB	Call plexit

PLHALT

```
void plhalt()
```

【引数】なし

【機能】

・プログラムを一時停止状態にしてエンドユーザーからの操作を待つ.

【制限】

・plhaltを引用すると以降のプログラムに処理が移される機会はない.

【引用例】

Fortran	call plhalt
VB	Call plhalt

PLOT

```
void plot(double *x, double *y, int *n)
```

【引数】

double *x	ペンの移動先座標
double *y	ペンの移動先座標
int *n	移動モード

【機能】

・座標（*x, *y）にペンを移動する.

・*nの絶対値が２なら現在の座標から移動先までを直線で描画する.

・*nの絶対値が３なら移動先までペンを移動する（描画はしない）.

・*nが負の場合，移動先を新しく原点とする.

【制限】

ペンモードの絶対値が２か３のいずれでもない場合は何もしない.

【引用例】

Fortran　call plot(x, y, 2)

VB　　Call plot(x, y, 2)

SYMBOL

```
void symbol
(double *x, double *y, double *h, char *s, int ns1, double *deg, int *ns2)
```

【引数】

double *x	文字列描画原点（左下）座標
double *y	文字列描画原点（左下）座標
double *h	文字列高さ（単位は座標単位と同じ）
char *s	文字列（センターシンボルの場合は未使用）
int ns1	文字列長さ（半角文字を単位とする）
double *deg	角度（x軸から反時計回りの角度．単位は度で-180～180度）
int *ns2	文字列長さ（*ns1と同じ値）または負の整数

【機能】

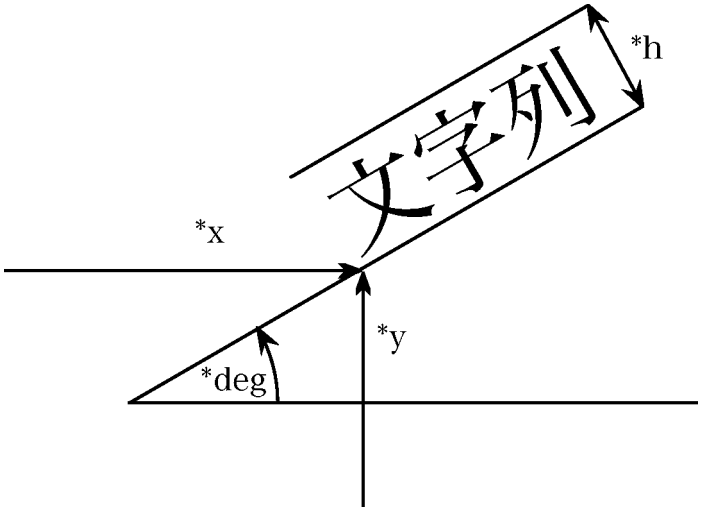
・*ns2が正の場合は通常の文字列表示となる．負の場合はセンターシンボル表示となる．

〔文字列表示の場合〕

・文字高さは*h，文字列角度は*degで指定する．

・座標（*x, *y）を始点とする文字列sを描画する．

・文字列長さは半角一文字を単位とするので全角文字は一文字で長さ２となる．



〔センターシンボルの場合〕

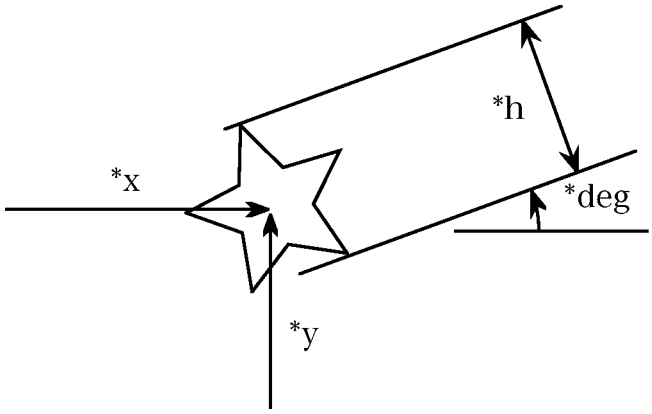
・シンボル高さは*hで指定する．シンボル幅は高さと等しい．

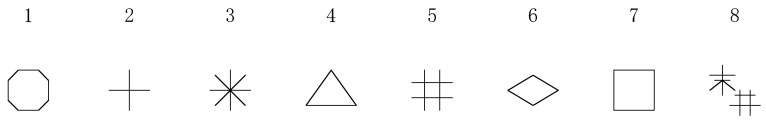
・*sとns1は使用しない．

・座標（*x, *y）をシンボルの中心として*degだけ傾いたシンボルを描画する．

・シンボルパターンは*ns2の絶対値によって決まる．

・*ns2の絶対値が８を超えている場合は８とみなす．





【制限】

文字数ns1は実際に渡される文字列sの文字数でなければならない．一方，文字数*ns2は実際に描画する文字数でよいので*ns2≦ns1ならばいくらでもよい．

【引用例】

Fortran	call symbol(x, y, h, '漢字ABC', deg, 7)
VB	Call symbol(x, y, h, "漢字ABC", 7, deg, 7)

MDSYMB

```
void mdsymb
(double *x, double *y, double *h, char *s, int ns1, double *deg, int *ns2)
```

【引数】

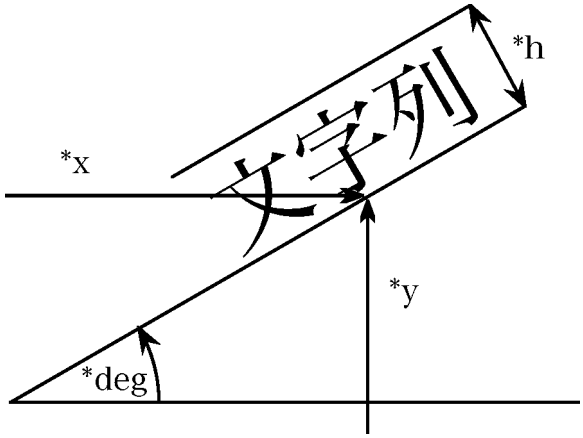
double *x	文字列描画原点（中心下）座標
double *y	文字列描画原点（中心下）座標
double *h	文字列高さ（単位は座標単位と同じ）
char *s	文字列
int ns1	文字列長さ（半角文字を単位とする）
double *deg	角度（x軸から反時計回りの角度．単位は度で-180～180度）
int *ns2	文字列長さ（*ns1と同じ値）

【機能】

・座標（*x, *y）を中心下として左右に均等に振り分けられた文字列sを描画する．

・文字高さは*h，文字列角度は*degで指定する．

・文字列長さは半角一文字を単位とするので全角文字は一文字で長さ２となる．



【制限】

文字数ns1は実際に渡される文字列sの文字数でなければならない．一方，文字数*ns2は実際に描画する文字数でよいので*ns2≦ns1ならばいくらでもよい．

【引用例】

Fortran	call mdsymb(x, y, h, '漢字ABC', deg, 7)
VB	Call mdsymb(x, y, h, "漢字ABC", 7, deg, 7)

RTSYMB

```
void rtsymb
(double *x, double *y, double *h, char *s, int ns1, double *deg, int *ns2)
```

【引数】

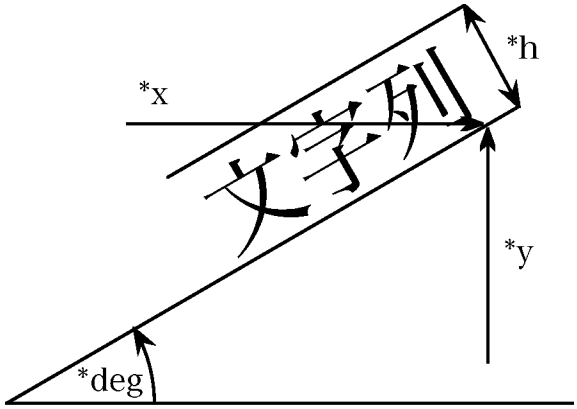
double *x	文字列描画原点（右下）座標
double *y	文字列描画原点（右下）座標
double *h	文字列高さ（単位は座標単位と同じ）
char *s	文字列
int ns1	文字列長さ（半角文字を単位とする）
double *deg	角度（x軸から反時計回りの角度．単位は度で-180～180度）
int *ns2	文字列長さ（*ns1と同じ値）

【機能】

・座標（*x, *y）を右下とする文字列sを描画する．

・文字高さは*h，文字列角度は*degで指定する．

・文字列長さは半角一文字を単位とするので全角文字は一文字で長さ２となる．



【制限】

文字数ns1は実際に渡される文字列sの文字数でなければならない．一方，文字数*ns2は実際に描画する文字数でよいので*ns2≦ns1ならばいくらでもよい．

【引用例】

Fortran　call rtsymb(x, y, h, '漢字ABC', deg, 7)

VB　　Call rtsymb(x, y, h, "漢字ABC", 7, deg, 7)

NUMBER

```
void number(double *x, double *y, double *h, double *v, double *deg, int *n)
```

【引数】

double *x	数字列描画原点座標
double *y	数字列描画原点座標
double *h	数字高さ（単位は座標単位と同じ）
double *v	数値
double *deg	角度（x軸から反時計回りの角度．単位は度で-180～180度）
int *n	表示形式

【機能】

・座標（*x, *y）に数値*vを文字表示する．

・座標（*x, *y）は常に数字列の左下を示す．

・文字高さは*h，文字角度は*degで指定する．

・数値から文字列への変換は*nによって次のように行なわれる．

〔*nが負の場合〕

・数値*vの整数部だけを左詰めで文字列に変換する．

・数値*vの整数部は10の（*nの絶対値－１）乗の桁まで丸める．

〔０≦*n≦９の場合〕

・小数点以下第（*n）位までを左詰めで文字列に変換する．

・*n＝０の場合は小数点までが文字列に加えられる．

〔*n＞９の場合〕

・数値を右詰めで文字列に変換する．

・小数点以下の桁数は*nの一の位の数値で指定する．

・全体の文字数は*nの十の位以上の数値で指定する．

【制限】なし

【引用例】vの値を８文字の右詰め文字列（小数点以下第二位まで）として表示する例

Fortran　call number(x, y, h, v, deg, 82)

VB　　Call number(x, y, h, v, deg, 82)

CIRCLE

```
void circle(double *x, double *y, double *tho, double *thf,
                            double *ro, double *rf, double *dl)
```

【引数】

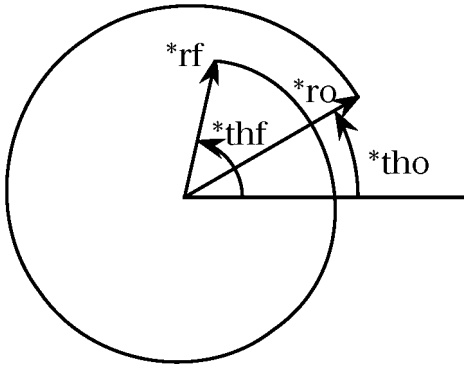
double *x	円弧始点座標
double *y	円弧始点座標
double *tho	円弧開始角度（x軸から反時計回りの角度．単位は度）
double *thf	円弧終了角度（x軸から反時計回りの角度．単位は度）
double *ro	円弧開始の半径
double *rf	円弧終了の半径
double *dl	未使用

【機能】

・座標（*x, *y）を始点とする円弧を描く．

・開始角度は*tho，終了角度は*thfで指定する．したがって円弧の中心座標は（*x-cos(*tho), *y-sin(*tho)）となる．

・開始半径と終了半径を異なる値にすると角度に比例して半径が変化するアルキメデスらせんとなる．



【制限】

・VBから引用する場合，circleという名前はVBの予約語なので注意が必要[1)]．

【引用例】

Fortran call circle(x, y, 0.0d0, 360.0d0, 5.0d0, 5.0d0, dl)

VB Call Sercle(x, y, 0#, 360#, 5#, 5#, dl)

---

1) たとえば添付している serenade.bas では Sercle という名前で宣言している．serenade.bas を使う場合は Sercle というサブルーチン名で引用すればよい．

DASH

```
void dash(double *xs, double *ys, double *xe, double *ye, double *dl)
```

【引数】

double *xs	始点座標
double *ys	始点座標
double *xe	終点座標
double *ye	終点座標
double *dl	破線間隔

【機能】

・座標（*xs, *ys）から座標（*xe, *ye）までの破線を描く.

・線のある部分の長さと線のない部分の長さはどちらも*dlとなる.

【制限】なし

【引用例】

Fortran	call dash(xs, ys, xe, ye, 5.0d0)
VB	Call dash(xs, ys, xe, ye, 5#)

PLADDP

```
void pladdp(double *x, double *y)
```

【引数】

double *x	パスに追加する座標
double *y	パスに追加する座標

【機能】

・カレントパスに指定した座標（*x, *y）を追加する.

・カレントパスが存在しない場合は指定した座標（*x, *y）を始点とする新しいパスを作成し，カレントパスとする.

【制限】なし

【引用例】

Fortran	call pladdp(x, y)
VB	Call pladdp(x, y)

PLCLPT

```
void plclpt(int *n)
```

【引数】

int *n　　パス作成モード（0 または 1）

【機能】

・カレントパスを閉じて閉じたパスをリージョンに追加する.

・*nが 0 の場合は指定された順でパスを作成するが，1 の場合は指定された逆順でパスを作成する[1)].

【制限】

・パスを構成する座標は 3 点以上なければならない.

【引用例】

Fortran　call plclpt(0)

VB　　Call plclpt(0)

1) 逆順にパスを作成するというのは右回りと左回りを逆にしてパスを作成するということ.

PLRGBP

```
void plrgbp(double *r, double *g, double *b, int *n)
```

【引数】

double *r	カラーパラメーター赤（0以上1以下）
double *g	カラーパラメーター緑（0以上1以下）
double *b	カラーパラメーター青（0以上1以下）
int *n	描画モード（現在未使用）

【機能】

・リージョン内部を指定した色で塗りつぶす.

・カラーパラメーターは0が輝度最小で1が輝度最大である．すなわちすべて0なら黒，すべて1なら白となる.

【制限】

・リージョンを作成していない状態で塗りつぶしを行なってはならない.

・カラーパラメーターに負の値や1を越える値を指定してはならない.

【引用例】

Fortran call plrgbp(0.0d0, 1.0d0, 0.0d0, 0)

VB Call plrgbp(0#, 1#, 0#, 0)

NEWPEN

```
void newpen(int *n)
```

【引数】

int *n　　ペン番号（1～10）

【機能】

・指定したペン番号にペンを変更する.

・この後に引用される線描画関数すべてに影響を及ぼす.

・ペンが変更されることによって線の色と線の太さが変わる.

【制限】

・ペン番号は1～10以外を指定してはならない.

【引用例】

Fortran　call newpen(2)

VB　　Call newpen(2)

PLFONT

```
void plfont(int *n)
```

【引数】

int *n　　フォント番号（1～10）

【機能】

・指定したフォント番号にフォントを変更する.

・この後に引用されるsymbol/mdsymb/rtsymb/numberに影響を及ぼす.

【制限】

・フォント番号は1～10以外を指定してはならない.

【引用例】

Fortran　call plfont(5)

VB　　Call plfont(5)

GENTEN

```
void genten(double *x, double *y)
```

【引数】

double *x	新しい原点座標
double *y	新しい原点座標

【機能】

・現在の座標系での座標（*x, *y）を新しい原点とする.

・この後のすべての関数の引用に影響がある.

【制限】なし

【引用例】

Fortran	call genten(x0, y0)
VB	Call genten(x0, y0)

FACTOR

```
void factor(double *r)
```

【引数】

double *r　　スケールファクター（≠０）

【機能】

・この後すべての関数の引用で指定される座標値，および文字の高さに*rをかけて描画する．

【制限】

・スケールファクターに０を指定してはならない．

【引用例】

Fortran　call factor(2.0d0)

VB　　Call factor(2#)

PLMSG

```
void plmsg(char *s, int ns)
```

【引数】

char *s	表示する文字列
int ns	表示する文字列の長さ（半角文字を単位とする）

【機能】

・指定した文字列sをWindow表示する.

・エンドユーザーがOkボタンをクリックするまでプログラムは次に進まない.

・描画ページが有効になっていない状態でも引用できる.

【制限】なし

【引用例】

Fortran　call plmsg('error occurred!')

VB　　Call plmsg("error occurred!", 15)

PLREAD

```
void plread(char *s1, int ns1, char *s2, int ns2, int *ms2)
```

【引数】

char *s1	プロンプト文字列
int ns1	プロンプト文字列長さ（半角文字を単位とする）
char *s2	《出力》エンドユーザー入力文字列
int ns2	エンドユーザー入力文字列用変数のサイズ（半角文字を単位とする）
int *ms2	《出力》実際に入力された文字列の長さ（半角文字を単位とする）

【機能】

・プロンプト文字列が表示された文字列入力用Windowを表示する.

・入力テキストボックスにはデフォールトとしてs2が最初に入っている.

・Okボタンかキャンセルボタンのどちらかをクリックするまでプログラムは次に進まない.

・Okボタンがクリックされたら入力テキストボックスの文字列をs2にセットし，*ms2にはその長さをセットして戻る.

・キャンセルボタンがクリックされたら何もセットしないで戻る.

・描画ページが有効になっていない状態でも引用できる.

【制限】

・エンドユーザーが入力した文字列の長さはns2以下でなければならない.

【引用例】

```
Fortran  call plread('Input number ?', c, n)
VB       Call plread("Input number ?", 14, c, 100, n)
```