TR-AC-0038

C10

Windows版2次元プロットツール
（FD付）

新上 和正　斎藤 茂　北川 美宏 (ATR-I)

2000. 1.20

ATR環境適応通信研究所

目次

1. はじめに

UNIX(X-window)版２次元グラフプロットツールACRPlotTool Ver.2 をWindows98 環境へ移植し、Windows版ACRPlotTool としてリリースしました。

UNIX 版同様、種々のシミュレーションや実験から得られる数値データを、簡易なフォーマットで指定するだけで２次元グラフにプロットできます。また、カラー化・グラフの重ね描き機能を利用すれば、ひとつのグラフ上で複数のデータ系列を比較できます。データファイルのフォーマットは、UNIX版と完全互換です。

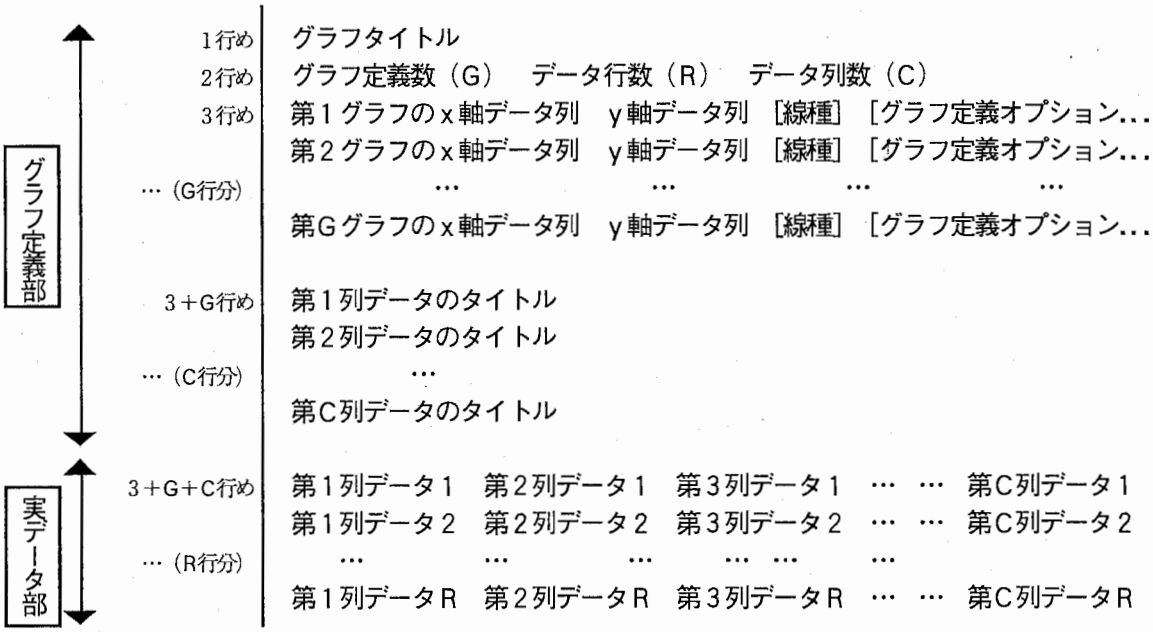
Windows版での新しい機能として、グラフウィンドウからの直接印刷やラベル文字列のフォント変更等が追加されました。

グラフは、ウィンドウへ表示、印刷、またはPS(Post Script),EPS ファイルへ出力されますので、データの視覚化やドキュメント化にご活用ください。

2. 使用法

2.1. データファイルフォーマット

ACRPlotTool のデータファイルは、以下のフォーマットで記述されたASCII ファイルです。
なお、グラフ定義部の［線種］および［グラフ定義オプション］<参照：2.4.線種とグラフ定義オプション>は省略できます。



（注１）空行は無視されます。
（注２）「!」で始まる行は、コメント行となります。
（注３）「@@@@」で始まる行が実データ部にあると、それ以降のデータは無視されます。
（注４）日本語（２バイト文字）には対応していません。

例：

```
sample graph
6   502   6
1  3  0
2  3  1
1  4  1
1  6  1
2  3  0
1  5  0
simulation time
generated packets
routed packets
lost packets
average packet delay
average call delay
100.0   547   362   339    67    66
200.0   598   544   499   100    94
300.0   434   521   545    92    98
400.0   468   437   494    91    94
500.0   375   433   385    92    80
600.0   387   421   458    75    80
700.0   504   439   441    77    81
800.0   486   500   478   106   104
  ...   ...   ...   ...   ...   ...   ...
```

2.2. 実行

ACRPlotTool（以下、plot.exe）を実行するには、主に3つの方法があります。

- plot.exeのアイコンをダブルクリックする
- plot.exeに関連づけたデータファイル（拡張子.apt）をダブルクリックする
- MS-DOSプロンプトからplot.exeを実行する

(1) plot.exeのアイコンをダブルクリックする方法

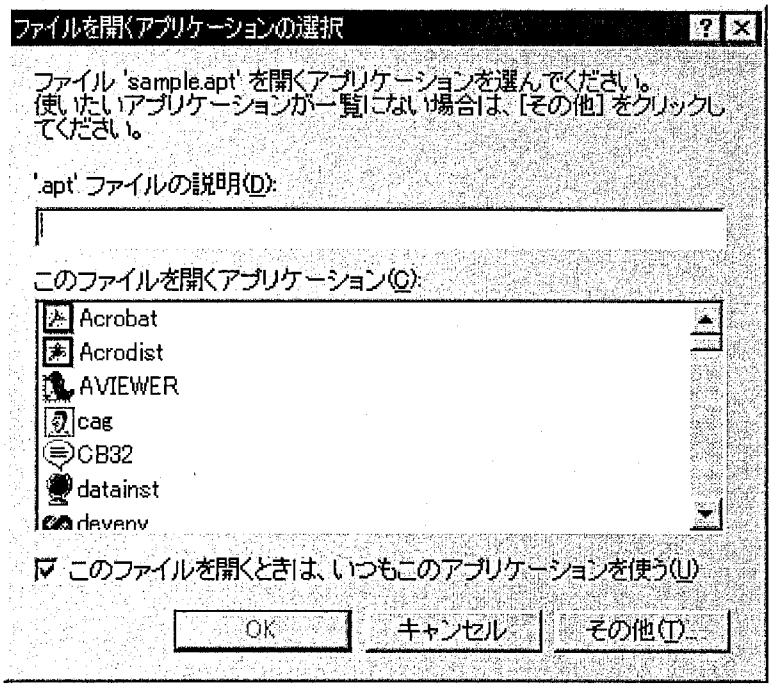
アイコンをダブルクリックすると、plot.exeが起動しウィンドウがひとつ開きます。plot.exeと同じディレクトリの中にfort.99という名前のファイルがある場合は、そのファイルをデータファイルとして読み込み、グラフを表示します。

（注）fort.99は、UNIX版の仕様にあわせたものです。Windows版でのデータファイルの拡張子は.aptです。

(2) データファイル（拡張子.apt）をダブルクリックする方法

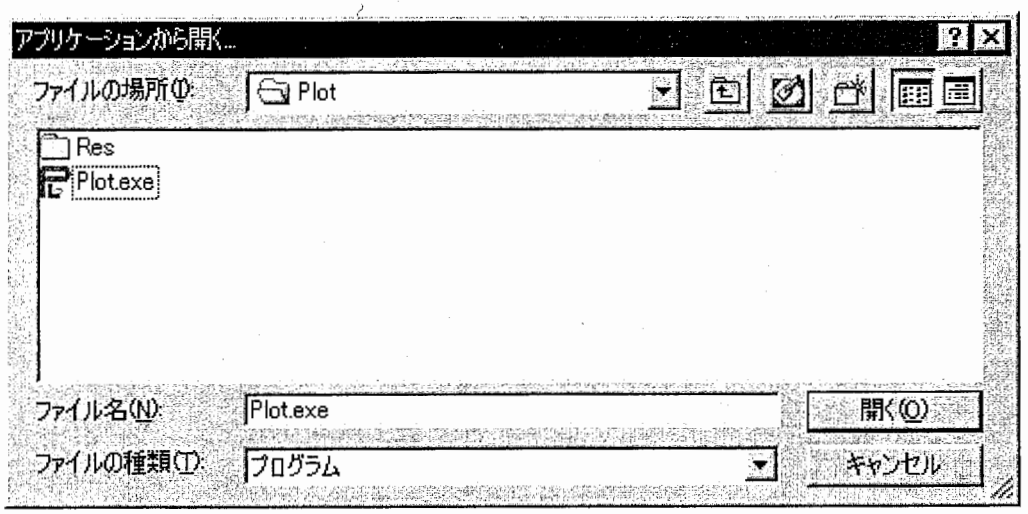
plot.exeと関連づけたデータファイルをダブルクリックすると、plot.exeが起動しウィンドウがひとつ開きます。ダブルクリックしたデータファイルが読み込まれ、グラフが表示されます。

plot.exeとデータファイル（拡張子.apt）の関連づけは、以下の手順により行えます。



アプリケーションとの関連づけをしていないデータファイル（拡張子.apt）をダブルクリックすると、上のようなウィンドウが開きます。このとき、チェックボックス［このファイルを開くときは、いつもこのアプリケーションを使う］がチェックされていることを確認します。

通常、このウィンドウのアプリケーションリストにはplot.exeが入っていません。その場合は、［その他］ボタンをクリックします。



ここで、plot.exeファイルのあるディレクトリからplot.exeを選択、［開く］ボタンをクリックします。
以上により、先にダブルクリックしたデータファイルのグラフが表示されるとともに、plot.exeとデータファイル（拡張子.opt）が関連づけられます。

なお、アプリケーションとファイルの関連づけは、［スタート］メニュー→［設定］→［フォルダオプション］→［ファイルタイプ］タグからも行なえます。詳しくは、Windowsのマニュアル、ヘルプを参照してください。

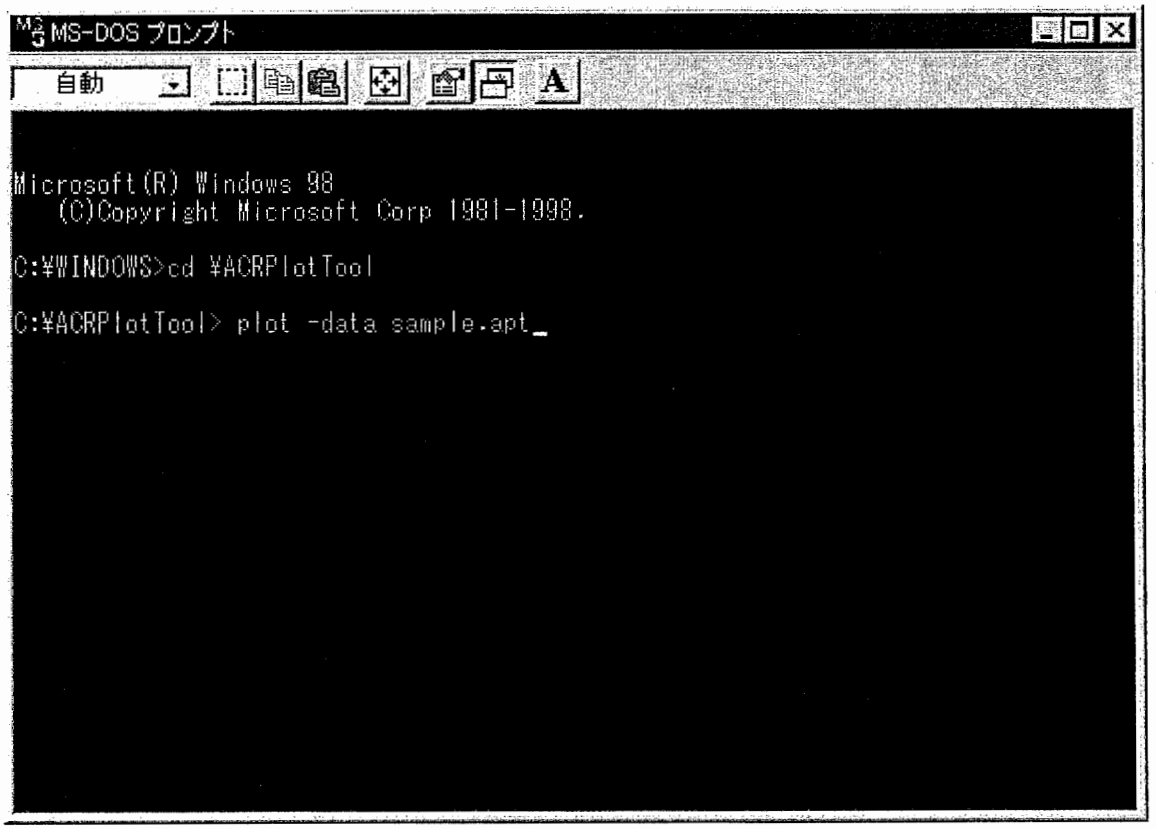
(3) MS-DOSプロンプトから実行する方法

［スタート］メニュー→［プログラム］→［MS-DOS プロンプト］により、MS-DOS プロンプトウィンドウを開きます。

plot.exe ファイルのあるディレクトリへ移動するか、適当なパス名を加えて、plot と入力し<Enter>キーを押すと、plot.exe が起動しウィンドウがひとつ開きます。カレントディレクトリ内に"fort.99"という名前のファイルがある場合は、そのファイルをデータファイルとして読み込み、グラフを表示します。

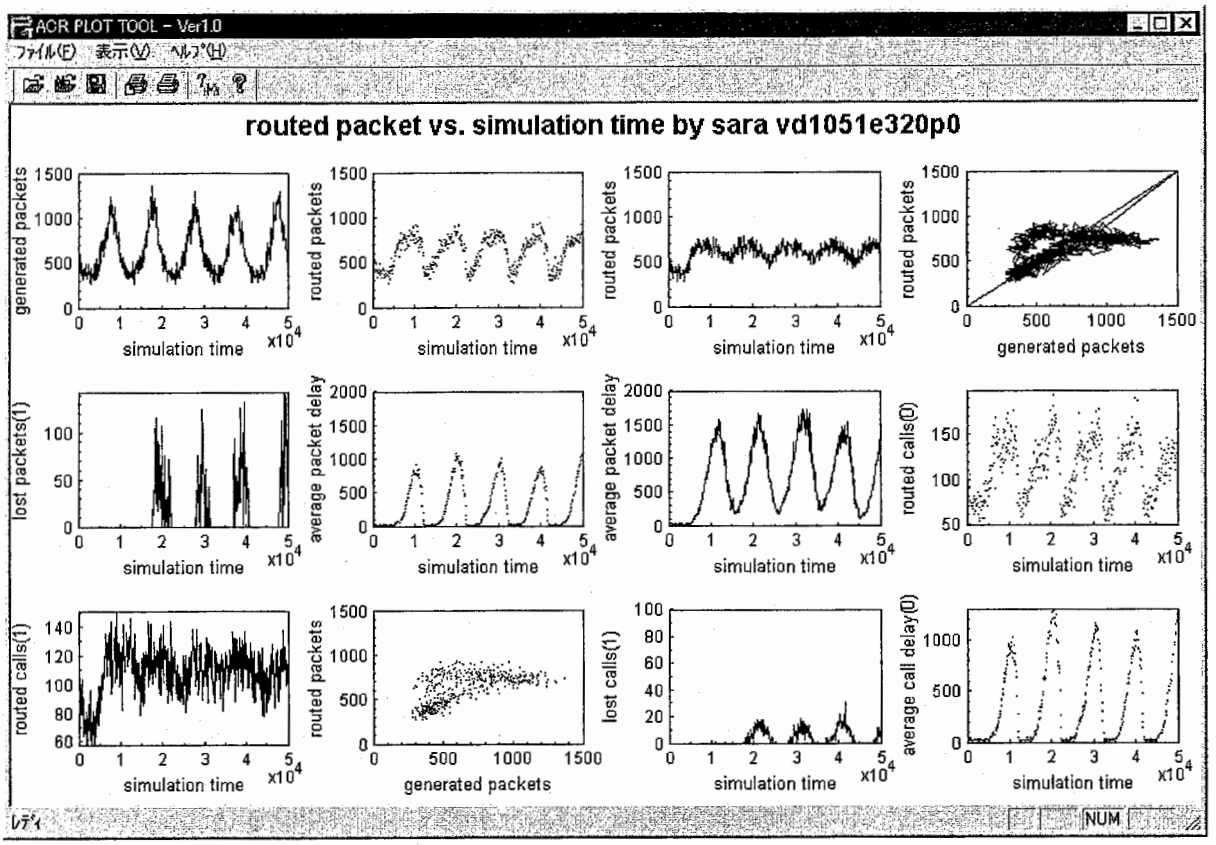
またplotと入力し、続いて空白区切りによりコマンドラインオプションを指定することもできます<参照:2.3. 実行時コマンドラインオプション>。

例えば、実行時に読み込むデータファイルを指定する場合は、オプション -data に続いてファイル名を入力します。



また(1)～(3)の方法以外にも、［スタート］メニュー→［ファイル名を指定して実行］により実行する方法やplot.exeのアイコン上へデータファイルをドラッグ＆ドロップして実行する方法があります。

これらの方法により、新しいウィンドウが開き、該当するデータファイルが存在する場合は次図のようにグラフが表示されます。



plot.exe の起動後は、［ファイル］→［開く］により、データファイルをグラフ表示できます。また、ウィンドウ内へデータファイルをドラッグ＆ドロップし、グラフ表示することもできます。

開いたウィンドウは、ウィンドウの［ファイル］メニュー→［アプリケーションの終了］により閉じられます。

2.3. 実行時コマンドラインオプション

MS-DOSプロンプトからの実行時に以下のオプションが使用できます。

- -data <ファイル名>
  読み込むデータファイル名を指定します。無指定の場合、既定値としてfort.99という名前のファイルが読み込まれます。

- -ps <ファイル名>
  出力するPS (PostScript)ファイル名を指定します。指定した場合、グラフはウィンドウ表示されません。

- -eps <ファイル名>
  出力するEPSファイル名を指定します。指定した場合、グラフはウィンドウ表示されません。
  EPSファイルは、EPS形式に対応したアプリケーションソフトウェアで図として読み込むことができます。

- -landscape
  横置きでウィンドウ表示またはPSまたはEPSファイル出力されます(既定値)。

- -portrait
  縦置きでウィンドウ表示またはPSまたはEPSファイル出力されます。

- -aspectratio <比率>
  グラフの縦横比を実数で指定します。比率が1の場合、グラフが正方になります。

- -dotratio <比率>
  プロットする点の拡大縮小率を指定します。グラフ定義オプションdotsizeの値とこの-dotratioの比率との積がプロットする点の大きさとなります。

- -head <ファイル名>
  グラフ定義部分のみのファイルを指定します。オプション-dataと同時に指定した場合、-dataで指定したデータファイルのグラフ定義部分は無視されます。

- -body <ファイル名>
  実データ部分のみのファイルを指定します。オプション-headと合わせて使用します。

- -geometry <幅>x<高さ>+<座標x>+<座標y>
  ウィンドウのサイズと位置を指定します。指定方法は、X-windowのgeometry指定に準じます。

以上のオプションは、MS-DOSプロンプトから実行する場合の他、［スタート］メニュー→［ファイル名を指定して実行する］でのプログラム名や、ショートカットの［プロパティ］での［リンク先名］に続けて指定することもできます。

2.4. 線種とグラフ定義オプション

データファイル中のグラフ定義部で以下の線種を指定できます。既定値は、0（点描）です。

線種	内容
0	点
1	実線
2	点線
3	1点鎖線
4	長点線
5	長1点鎖線

データファイル中のグラフ定義部で以下のグラフ定義オプションを使用できます。

オプション名	短縮形	内容	例	既定値
dotcolor	dc	色番号またはRGB値(#RRGGBB)でグラフの色を指定 0:black, 1:red, 2:blue, 3:green, 4:pink, 5:violet, 6:yellow, 7:orange, 8:brown, 9:spring-green	dotcolor=1 dotcolor=#00ffff	0
dotsize	ds	点の大きさ(線の太さ)を指定	dotsize=3	1
duplicate	d	直前のグラフへの重ね描きを指定	duplicate=1	-
label	l	ラベル用のデータ列を指定<参照：2.5.ラベルデータ列>	label=8	-
labeldistance	ld	ラベル文字列の中心と点との距離を指定	labeldistance=5	0
labelsize	ls	ラベル文字列の大きさを指定	labelsize=16	10

グラフ定義オプションは、空白文字で区切り、複数指定できます。同一オプションが、複数回指定された場合は、最後の指定が有効となります。

例1：

```
label=3 dotsize=2 dotcolor=1 duplicate=1
```

例2：（短縮形を使用）

```
l=3 ds=2 dc=1 d=1
```

2.5. ラベルデータ列

プロット点にラベル文字列を付けることができます。実データ列の並びにラベル文字列のデータ列を用意し、グラフ定義オプション label=列番号 で指定します。また、数値データと区別するため、データ列タイトル部には「@label」と記述します。

例：

```
sample graph
6   502   7
1  3  0  label=7
2  3  1
1  4  1
1  6  1
2  3  0
1  5  0
simulation time
generated packets
routed packets
lost packets
average packet delay
average call delay
@label
100.0   547   362   339    67    66   min
200.0   598   544   499   100    94   @0
300.0   434   521   545    92    98   max
400.0   468   437   494    91    94   @0
500.0   375   433   385    92    80   @0
600.0   387   421   458    75    80   @0
700.0   504   439   441    77    81   @0
800.0   486   500   478   106   104   @0
  …   …   …   …   …   …   …
```

ラベルデータ列には、以下の予約語を使用できます。

予約語	内容
@0	ラベル無し
@x	点のｘ軸の値をラベルとする
@y	点のｙ軸の値をラベルとする
@000LABEL	ラベル LABEL の表示位置を指定する
～@359LABEL	@に続く３桁の数字は角度、既定値090

ラベルの表示位置は、以下の手順で決定します<参照：2.4.線種とグラフ定義オプション>。

1. ラベルを付ける点Ａからグラフ定義オプションの labeldistance 分ｘ軸の正方向に離れた点Ｂを求める
2. 点Ａを中心として指定角度分だけ点Ｂを回転（左回りにプラス回転）させた点Ｃを求める
3. 点Ｃがラベル文字列の中心点となるようラベルを表示する

2.6. Windows 版で追加された機能

Windows 版では主に2つの機能が新しく追加されました。

- 印刷機能
- グラフのプロパティ設定機能

(1) 印刷機能

plot.exe のウィンドウから［ファイル］メニュー→［印刷］により、ウィンドウに表示しているグラフを直接PSプリンタへ出力できます。

ウィンドウでの表示イメージがそのまま、用紙サイズに調整されて印刷されますので、［印刷プレビュー］で印刷イメージを確認の上、必要に応じて［プリンタの設定］により適当なプリンタのプロパティを設定してください。

(2) グラフのプロパティ設定機能

plot.exe のウィンドウから［ファイル］メニュー→［プロパティ］により、グラフのプロパティを設定できます。

- 縦横比

  各グラフの縦軸と横軸の長さの比を横／縦の値で指定します。1の場合、グラフが正方になります。既定値 -1.000 ではウィンドウのサイズに合わせて縦横比が適当に調整されます。

  この項目は、コマンドラインオプションの -aspectratio と同意です。実行時に -aspectratio を指定した場合は、そちらの値が優先されます。

- ドット比

プロットする点の拡大縮小率を指定します。グラフ定義オプションの dotsize の値とこの値との積が実際にプロットする点の大きさになります。既定値は 1.000 です。

この項目は、コマンドラインオプションの -dotratio と同意です。実行時に -dotratio を指定した場合は、そちらの値が優先されます。

- 表示の向き

既定値は横です。この値を縦にすると、ウィンドウの高さと幅が逆転し、縦長のウィンドウに変わります。ウィンドウ内に複数のグラフを表示している場合は、グラフの配置も変化します。

この項目は、コマンドラインオプションの -landscape/-portrait と同意です。実行時に -landscape または -portrait を指定した場合は、そちらが優先されます。

- タイトルのフォント

ウィンドウ上部に表示されるグラフタイトルのフォントを指定します。

- ラベルのフォント

グラフの縦軸／横軸に表示されるラベル（データ列名）のフォントを指定します。

- 目盛りのフォント

グラフの縦軸／横軸に表示される目盛り数字のフォントを指定します。

- 目盛り（上付）のフォント

グラフの縦軸／横軸に表示される目盛りのうち指数部の数字のフォントを指定します。

フォントの指定は、［フォント］ボタンをクリックし、フォント設定用のウィンドウにより行います。

変更したグラフのプロパティは、［ＯＫ］ボタンまたは［適用］ボタンをクリックすることにより、表示中のグラフに反映されます。

また、［保存］ボタンをクリックすることにより、変更したプロパティをファイルとして保存できます<参照：2.7.初期化ファイル>。ファイルは、カレントディレクトリ内に PLOT.ini という名前で保存され、同じディレクトリから plot.exe を起動するとそのファイルが読み込まれます。既定値に戻すには、ファイル PLOT.ini を削除してから plot.exe を起動してください。

2.7. 初期化ファイルPLOT.ini

グラフのプロパティや実行時コマンドラインオプションを初期化ファイル PLOT.ini に定義しておくと、plot.exeの実行時に既定値より優先させることができます。初期化ファイルは、実行時のカレントディレクトリにあるものが読み込まれます。

[セクション名] キー名=値	内容 ()内は既定値
[ACR_PLOT]	実行時オプション等の指定
DATA = data_file	データファイル名（fort.99）
BODY= body_file	ボディファイル名()
HEAD = head_file	ヘッダーファイル名()
PS = ps_file	PSファイル名()
EPS = eps_file	EPSファイル名()
ORIENTATION = orientation	表示モード：1=横、2=縦(1)
ASPECT = ratio	グラフ軸の縦横比(-1.0)
DOT = ratio	点の拡大縮小率(1.0)
GEOMETRY = geometry	ウィンドウサイズと位置(879x600+100+100)
HELP = help_file	ヘルプファイル名()
[ACR_TITLE_FONT]	タイトル文字フォントの指定
FACE = FaceName	フォントのフェイス名(Arial)
SIZE = FontSize	フォントサイズ(14)
STYLE = FontStyle	フォントスタイル(ﾎﾞｰﾙﾄﾞ)
[ACR_LABEL_FONT]	ラベル文字フォントの指定
FACE = FaceName	フォントのフェイス名(Arial)
SIZE = FontSize	フォントサイズ(10)
STYLE = FontStyle	フォントスタイル(標準)
[ACR_MEM_FONT]	目盛り文字フォントの指定
FACE = FaceName	フォントのフェイス名(Arial)
SIZE = FontSize	フォントサイズ(9)
STYLE = FontStyle	フォントスタイル(標準)
[ACR_MEM2_FONT]	目盛り上付き文字フォントの指定
FACE = FaceName	フォントのフェイス名(Arial)
SIZE = FontSize	フォントサイズ(8)
STYLE = FontStyle	フォントスタイル(標準)

なお、実行時コマンドラインオプションが指定された場合は、そちらが初期化ファイルの指定より優先されます。

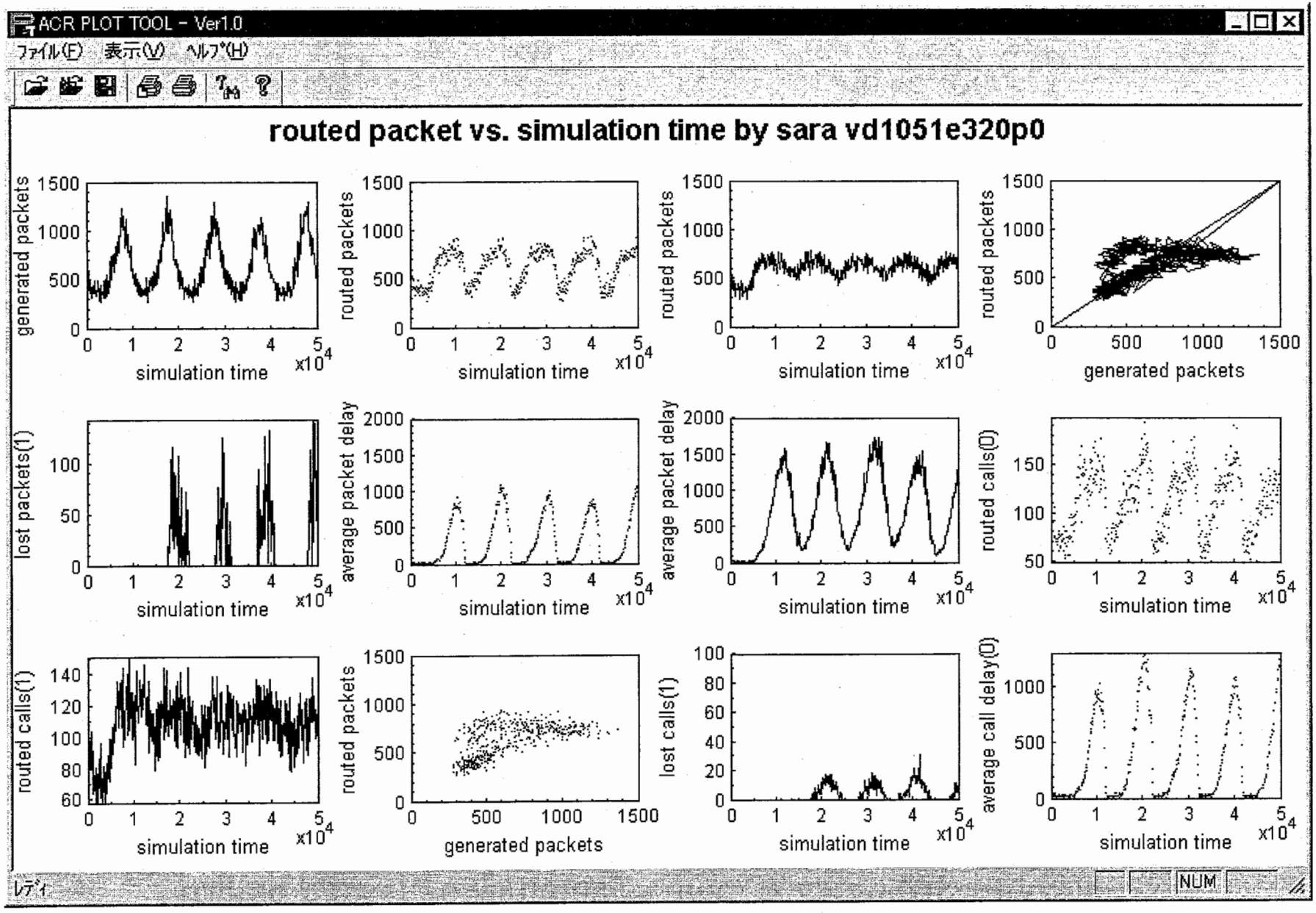
2.8. 稼働環境および開発環境

Windows98 環境があれば、plot.exe をコピーするだけで使用できます。
また、プロットするデータ量に論理的な制限はありませんが、実際に読み込めるデータ量は使用可能なメモリサイズに依存します。

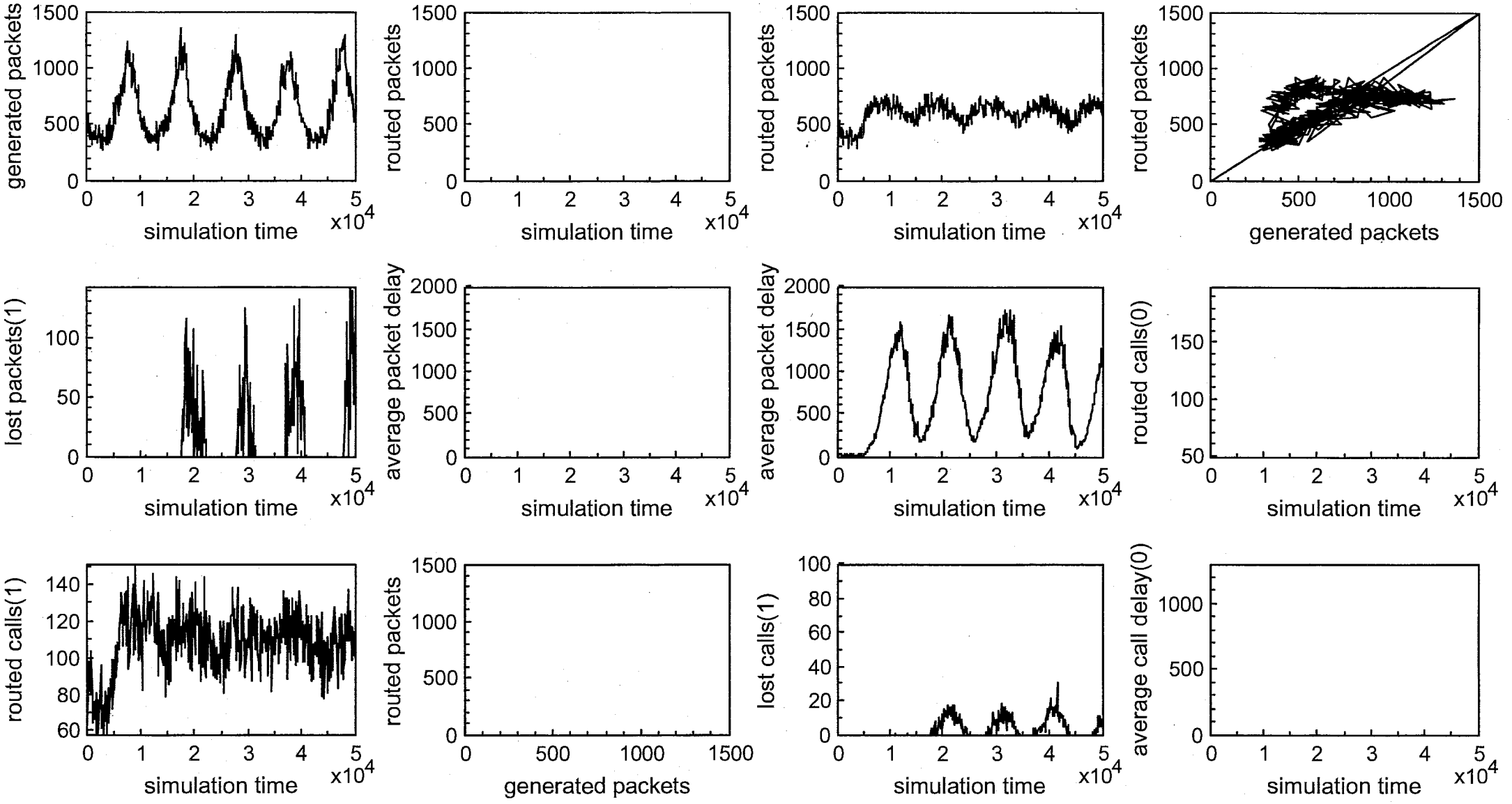
WindowsNT4.x環境では、グラフのプロパティのうち、フォントのスタイルに一部適用不可なものがあります。

本プログラムの開発には、Microsoft 社のVisualC++ Ver6.0 を使用しました。

A1 サンプルグラフおよびデータ


ACR PLOT TOOL - Ver1.0
ファイル(F) 表示(V) ヘルプ(H)
routed packet vs. simulation time by sara vd1051e320p0
generated packets
simulation time
routed packets
generated packets
lost packets(1)
average packet delay
routed calls(0)
routed calls(1)
lost calls(1)
average call delay(0)
x10^4
レディ
NUM

routed packet vs. simulation time by sara vd1051e320p0



routed packet vs. simulation time by sara vd1051e320p0
12 502 14
1 2 1
1 3 0
1 4 1
2 3 1
1 6 1
1 7 0
1 8 1
1 9 0
1 10 1
2 3 0
1 12 1
1 13 0
simulation time
generated packets
routed packets
routed packets
lost packets(0)
lost packets(1)
average packet delay
average packet delay
routed calls(0)
routed calls(1)
lost calls(0)
lost calls(1)
average call delay(0)
average call delay(1)

0.0	0	0	0	0	0	0.0	0.0	67	66	0	0	0.0	0.0
0.0	1500	1500	1500	0	10	2000.0	2000.0	67	66	0	100	200.0	200.0
100.0	547	362	339	0	0	19.34	18.83	67	66	0	0	20.48	20.58
200.0	598	544	499	0	0	25.27	36.48	100	94	0	0	28.91	40.05
300.0	434	521	545	0	0	37.72	49.24	92	98	0	0	40.96	49.73
400.0	468	437	494	0	0	38.43	39.42	91	94	0	0	46.94	44.42
500.0	375	433	385	0	0	43.51	40.74	92	80	0	0	52.06	44.73
600.0	387	421	458	0	0	33.15	43.99	75	80	0	0	39.90	53.65
700.0	504	439	441	0	0	25.91	27.31	77	81	0	0	31.82	35.25
800.0	486	500	478	0	0	24.82	25.01	106	104	0	0	27.15	26.64
900.0	368	417	442	0	0	24.56	26.36	74	80	0	0	27.27	28.50
1000.0	323	333	323	0	0	16.08	16.25	71	69	0	0	19.25	17.43
1100.0	438	408	405	0	0	21.79	23.06	70	65	0	0	24.36	22.40
1200.0	374	385	399	0	0	22.55	27.03	71	76	0	0	26.46	33.07
1300.0	452	437	401	0	0	25.91	24.60	68	67	0	0	30.85	27.02
1400.0	368	382	383	0	0	24.24	36.13	74	70	0	0	27.04	36.13
1500.0	377	366	387	0	0	25.75	26.80	75	78	0	0	26.57	30.12
1600.0	492	420	410	0	0	23.44	27.08	61	58	0	0	29.94	31.68
1700.0	402	499	454	0	0	38.95	48.35	79	77	0	0	41.49	51.53
1800.0	394	401	434	0	0	24.38	40.34	82	81	0	0	25.95	39.62
1900.0	404	384	387	0	0	23.06	27.10	64	67	0	0	25.32	30.84
2000.0	403	415	425	0	0	21.42	27.71	77	77	0	0	24.72	28.19
2100.0	336	338	299	0	0	19.31	30.47	66	59	0	0	21.22	29.17
2200.0	501	441	478	0	0	19.02	28.44	71	77	0	0	25.07	36.16
2300.0	466	439	429	0	0	34.67	39.18	72	72	0	0	35.20	34.41
2400.0	480	450	465	0	0	39.77	44.14	78	78	0	0	45.55	48.16
2500.0	292	420	443	0	0	41.23	39.06	86	96	0	0	44.94	43.54
2600.0	331	338	326	-1.2	0	16.35	13.17	73	69	0	0	16.89	13.77
2700.0	418	370	384	0	0	20.08	20.64	62	62	0	0	22.55	22.31
2800.0	346	390	357	0	0	24.48	24.38	69	67	0	0	26.59	21.84
2900.0	447	361	370	0	0	20.38	22.68	57	60	0	0	20.69	26.13
3000.0	278	386	373	0	0	31.98	38.28	82	74	0	0	34.42	41.09
3100.0	401	352	360	0	0	18.78	22.20	54	58	0	0	20.73	24.90
3200.0	361	411	425	0	0	24.38	26.65	75	78	0	0	25.86	27.37
3300.0	390	366	367	0	0	19.34	19.77	66	66	0	0	23.98	22.85
3400.0	394	410	420	0	0	21.23	21.04	82	82	0	0	26.93	25.24
3500.0	325	350	341	0	0	15.49	13.98	76	75	0	0	17.23	15.75
3600.0	346	285	294	0	0	14.12	13.09	56	59	0	0	16.24	14.66
3700.0	501	408	394	0	0	24.11	21.21	70	69	0	0	30.14	24.00
3800.0	427	459	432	0	0	35.70	34.37	95	84	0	0	38.88	33.83
3900.0	318	424	439	0	0	23.70	37.49	77	84	0	0	26.46	38.00
4000.0	379	379	384	0	0	15.49	18.61	73	72	0	0	16.75	19.10
4100.0	494	332	384	0	0	21.64	18.63	56	65	0	0	23.21	23.39
4200.0	559	552	478	0	0	28.22	29.79	98	84	0	0	34.74	31.87
4300.0	343	420	472	0	0	34.81	37.21	75	89	0	0	35.41	38.64
4400.0	440	462	392	0	0	34.14	33.84	88	79	0	0	35.55	33.29
4500.0	501	430	455	0	0	23.91	33.60	87	87	0	0	29.91	35.81
4600.0	378	479	507	0	0	26.13	29.86	96	99	0	0	32.45	35.36
4700.0	392	390	391	0	0	21.12	24.47	70	72	0	0	21.44	24.99
4800.0	478	466	447	0	0	22.57	23.77	97	89	0	0	28.81	25.28
4900.0	545	478	491	0	0	21.74	23.35	89	99	0	0	24.84	27.51
5000.0	689	526	537	10	0	32.88	34.59	81	83	0	0	41.48	40.19
5100.0	559	614	656	0	0	49.34	47.98	111	107	0	0	61.94	52.22
5200.0	746	584	536	0	0	51.06	42.28	115	107	0	0	58.73	51.36
5300.0	507	589	555	0	0	55.61	51.12	90	98	0	0	55.68	53.77
5400.0	718	635	592	0	0	58.53	66.17	99	90	0	0	67.64	68.14
5500.0	764	642	567	0	0	58.76	62.65	101	104	0	0	68.67	68.89
5600.0	714	715	683	0	0	78.02	88.54	115	98	0	0	88.97	77.68
5700.0	712	683	625	0	0	78.45	93.96	137	109	0	0	92.28	93.66
5800.0	595	656	565	0	0	79.42	119.42	115	105	0	0	86.24	105.17
5900.0	780	730	621	0	0	86.52	111.42	137	108	0	0	104.12	110.33
6000.0	624	693	644	0	0	72.92	136.87	148	116	0	0	92.06	154.64
6100.0	724	743	716	0	0	68.94	138.79	130	127	0	0	76.46	167.45
6200.0	682	730	665	0	0	66.37	137.64	147	128	0	0	80.24	167.47
6300.0	822	575	643	0	0	66.47	160.27	118	119	0	0	81.21	179.86
6400.0	848	759	609	0	0	71.68	140.96	161	136	0	0	86.33	175.94
6500.0	758	642	654	0	0	98.70	157.46	90	125	0	0	118.80	217.06
6600.0	864	752	668	0	0	116.15	176.88	141	107	0	0	135.71	206.30
6700.0	863	842	697	0	0	121.95	168.22	170	129	0	0	142.21	180.30
6800.0	959	752	666	0	0	126.60	198.00	142	118	0	0	162.12	264.87
6900.0	962	732	760	0	0	132.47	216.02	148	130	0	0	141.03	223.57
7000.0	796	731	735	0	0	153.43	270.16	138	135	0	0	173.38	287.73
7100.0	1127	767	716	0	0	160.50	245.45	125	137	0	0	181.91	290.66
7200.0	1071	821	681	0	0	172.89	237.50	155	121	0	0	213.61	300.92
7300.0	1137	785	747	0	0	179.35	221.13	140	107	0	0	207.52	270.29
7400.0	1073	779	672	0	0	208.22	281.57	152	111	0	0	243.77	330.39
7500.0	1227	807	706	0	0	228.35	351.60	145	129	0	0	261.03	399.63
7600.0	1232	643	672	0	0	259.54	342.50	126	144	0	0	287.03	411.55
7700.0	1053	778	606	0	0	300.51	345.19	126	94	0	0	308.37	386.41
7800.0	1165	661	626	0	0	325.05	426.86	114	103	0	0	368.11	462.15
7900.0	1060	769	661	0	0	363.71	475.10	121	116	0	0	414.06	520.19
8000.0	1028	744	683	0	0	394.60	454.29	129	109	0	0	428.10	485.12
8100.0	1041	704	717	0	0	431.84	550.61	124	127	0	0	471.74	593.09
8200.0	1053	695	731	0	0	442.67	536.78	107	123	0	0	526.55	559.31
8300.0	833	800	741	0	0	504.68	592.53	136	127	0	0	475.85	699.96
8400.0	892	864	684	0	0	527.95	672.64	138	140	0	0	607.80	667.76
8500.0	1130	700	722	0	0	537.36	647.12	119	109	0	0	582.98	763.45
8600.0	950	799	637	0	0	608.65	722.87	156	102	0	0	628.84	861.09
8700.0	1003	699	649	0	0	582.39	676.21	153	120	0	0	677.79	753.43
8800.0	974	838	657	0	0	587.62	703.27	121	105	0	0	698.17	848.49
8900.0	788	834	740	0	0	619.40	763.73	154	151	0	0	673.04	860.36
9000.0	830	752	752	0	0	633.49	763.89	146	140	0	0	718.07	757.34
9100.0	806	828	688	0	0	683.92	868.63	131	119	0	0	763.73	985.99
9200.0	763	847	779	0	0	707.09	803.68	136	119	0	0	807.88	993.11
9300.0	748	827	718	0	0	768.57	830.53	168	125	0	0	847.55	975.47
9400.0	785	804	587	0	0	789.14	1007.86	132	99	0	0	858.84	1080.34
9500.0	855	903	632	0	0	814.91	923.91	174	102	0	0	946.44	998.03
9600.0	599	757	694	0	0	791.39	947.35	141	101	0	0	934.45	1052.41
9700.0	631	910	737	0	0	843.63	1023.37	164	129	0	0	964.21	1191.25
9800.0	622	734	658	0	0	792.24	1076.56	145	118	0	0	871.32	1179.80
9900.0	538	799	700	0	0	791.46	1093.95	129	107	0	0	909.03	1170.98
10000.0	593	813	660	0	0	801.55	1167.62	140	104	0	0	927.08	1345.99
10100.0	469	934	602	0	0	791.74	1271.92	144	97	0	0	902.71	1321.49
10200.0	482	724	557	0	0	924.59	1233.91	159	83	0	0	976.51	1388.73
10300.0	549	767	630	0	0	813.01	1168.51	160	113	0	0	941.67	1478.94
10400.0	549	738	684	0	0	855.19	1279.68	154	142	0	0	1024.72	1515.59
10500.0	443	849	767	0	0	757.75	1239.05	122	131	0	0	886.24	1595.07
10600.0	578	917	699	0	0	790.22	1272.33	154	124	0	0	836.97	1485.10
10700.0	463	790	631	0	1	857.49	1359.03	177	111	0	0	956.93	1431.99
10800.0	424	635	633	0	0	778.19	1453.43	125	110	0	0	874.95	1647.48
10900.0	385	754	629	0	1	725.49	1296.58	136	120	0	1	796.27	1515.63
11000.0	471	679	597	0	0	746.23	1317.43	130	112	0	0	798.26	1590.42
11100.0	461	793	765	0	0	623.03	1261.38	145	128	0	0	872.73	1484.11
11200.0	481	716	708	0	0	588.09	1458.81	150	137	0	0	766.61	1663.30
11300.0	377	775	739	0	0	599.51	1340.64	148	121	0	0	794.73	1601.61
11400.0	343	770	619	0	0	605.83	1477.31	134	131	0	0	691.49	1655.01
11500.0	382	748	676	0	0	500.89	1464.11	144	119	0	0	644.78	1717.54
11600.0	449	681	626	0	0	315.74	1359.33	159	114	0	0	456.07	1824.62
11700.0	375	616	694	0	0	207.78	1481.62	127	129	0	0	276.30	1676.26
11800.0	379	595	607	0	0	110.45	1542.25	116	134	0	0	111.36	1750.48
11900.0	338	466	559	0	0	78.11	1583.63	106	118	0	0	104.28	1789.39
12000.0	421	444	582	0	0	40.65	1559.91	73	126	0	0	48.33	1751.07
12100.0	405	408	665	-9	0	22.69	1543.46	84	137	0.00008	0	29.47	1697.27
12200.0	349	388	620	0	0	22.50	1422.40	71	146	0	0	24.43	1765.33
12300.0	383	352	608	0	0	14.81	1400.14	71	142	0	0	18.24	1584.45
12400.0	374	365	602	0	0	17.91	1211.89	65	109	0	0	19.27	1544.79
12500.0	360	381	577	0	0	27.18	1149.18	65	119	0	0	24.24	1735.14
12600.0	421	436	594	0	0	23.36	1200.58	79	114	0	0	27.12	1585.35
12700.0	418	362	539	0	0	22.14	1255.48	59	108	0	0	22.98	1599.56
12800.0	341	349	583	0	0	22.21	1204.27	68	109	0	0	26.22	1533.57
12900.0	525	468	622	0	0	25.74	1070.19	68	118	0	0	29.77	1357.95
13000.0	358	432	595	0	0	26.99	1245.66	84	136	0	0	31.66	1419.91
13100.0	428	386	529	0	0	18.80	898.99	62	91	0	0	22.51	1336.76
13200.0	411	439	614	0	0	24.81	998.40	80	122	0	1	27.35	1320.31
13300.0	279	364	613	0	0	17.21	942.45	74	135	0	0	21.42	1276.87
13400.0	497	332	567	0	0	15.98	982.02	49	101	0	0	17.43	1146.59
13500.0	408	458	580	0	0	26.51	807.06	89	108	0	0	28.09	1144.19
13600.0	488	489	596	0	0	25.35	657.25	85	115	0	0	27.78	990.02
13700.0	443	445	596	0	0	28.18	677.80	85	102	0	0	32.88	872.46
13800.0	526	435	595	0	0	23.46	573.30	72	116	0	0	27.89	849.54
13900.0	466	477	520	0	0	42.28	523.03	70	100	0	0	49.31	695.65
14000.0	538	512	526	0	0	39.99	579.90	78	96	0	0	47.80	958.22

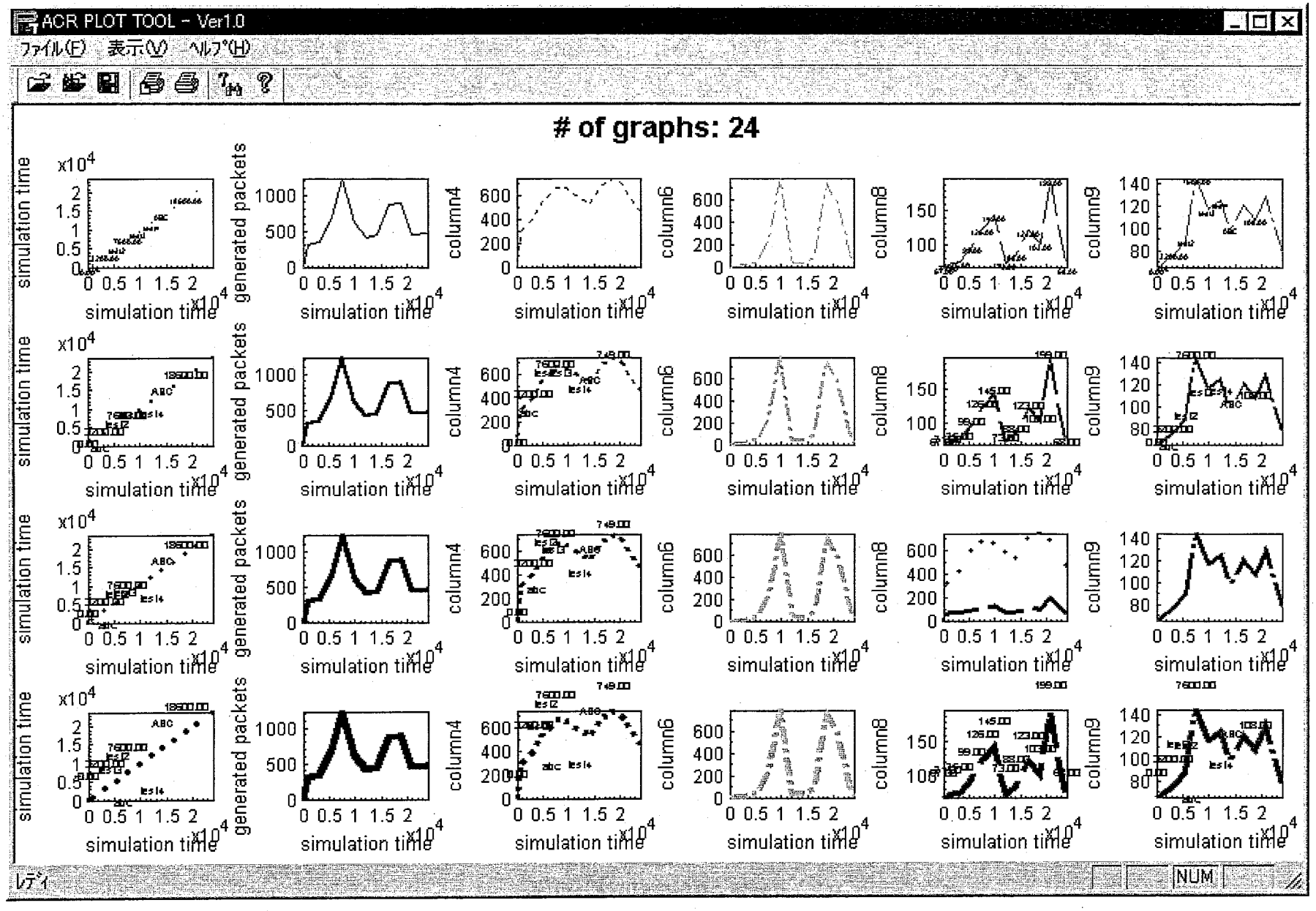
14100.0	452	472	586	0	0	49.32	614.60	76	116	0	0	53.96	831.51
14200.0	460	570	529	0	0	44.94	435.38	88	101	0	0	54.24	531.37
14300.0	395	434	625	0	0	24.47	502.15	81	116	0	0	30.34	666.90
14400.0	466	409	525	0	0	22.53	394.63	76	100	0	0	23.71	549.11
14500.0	489	518	506	0	0	25.26	253.49	100	79	0	0	31.86	329.29
14600.0	588	476	497	0	0	23.12	417.85	86	96	0	0	26.46	509.98
14700.0	527	492	545	0	0	37.31	361.88	89	89	0	0	40.70	354.43
14800.0	570	570	647	0	0	40.94	264.14	99	113	0	0	53.28	298.00
14900.0	451	555	519	0	0	48.25	336.41	113	118	0	0	53.53	439.09
15000.0	466	457	501	0	0	32.30	402.63	92	86	0	0	33.21	467.78
15100.0	513	472	585	0	0	32.90	333.61	91	122	0	0	36.25	341.27
15200.0	644	565	526	0	0	35.50	243.22	107	94	0	0	39.85	222.43
15300.0	530	616	703	0	0	39.36	216.74	109	121	0	0	49.20	242.64
15400.0	689	575	577	0	0	37.48	196.99	104	111	0	0	42.37	233.68
15500.0	575	553	554	0	0	49.28	227.85	98	88	0	0	49.73	286.08
15600.0	676	612	619	0	0	53.05	176.86	91	120	0	0	53.78	205.06
15700.0	780	654	672	0	0	63.27	193.47	129	107	0	0	82.12	180.49
15800.0	811	619	603	0	0	68.53	191.62	113	123	0	0	77.64	182.30
15900.0	679	752	658	0	0	86.42	199.56	131	107	0	0	103.56	217.85
16000.0	789	719	669	0	0	83.94	215.50	144	117	0	0	99.83	229.03
16100.0	694	612	593	0	0	85.61	244.35	99	121	0	0	85.28	317.97
16200.0	799	739	663	0	0	94.74	247.12	114	121	0	0	114.02	319.46
16300.0	936	657	657	0	0	121.57	215.62	120	104	0	0	131.61	230.91
16400.0	868	699	695	0	0	112.98	225.84	123	121	0	0	148.01	222.06
16500.0	1102	699	592	0	0	146.87	289.31	121	99	0	0	174.15	271.79
16600.0	847	701	773	0	0	150.85	269.77	119	144	0	0	154.14	309.18
16700.0	995	632	669	0	0	193.49	353.63	114	119	0	0	245.25	358.16
16800.0	976	757	774	0	0	186.48	369.36	102	104	0	0	230.00	421.92
16900.0	920	758	680	0	0	229.72	312.03	139	125	0	0	303.15	388.13
17000.0	1080	755	769	0	0	231.90	339.94	114	142	0	0	237.41	409.50
17100.0	1061	728	612	0	0	276.52	373.05	126	120	0	0	308.68	379.10
17200.0	1132	699	626	0	0	307.79	346.62	138	108	0	0	350.58	349.06
17300.0	1364	742	646	0	0	329.86	363.69	130	117	0	0	351.26	433.96
17400.0	1118	709	539	0	0	331.06	452.87	128	101	0	0	374.41	455.72
17500.0	997	711	678	0	0	367.68	412.27	101	107	0	0	412.94	483.36
17600.0	1051	714	661	0	40	409.01	482.55	115	120	0	4	463.84	594.33
17700.0	1196	823	660	0	35	409.79	503.68	145	130	0	0	499.76	544.30
17800.0	1078	709	784	0	13	461.30	570.38	130	120	0	0	491.54	578.20
17900.0	994	666	740	0	13	500.65	590.54	129	109	0	0	625.68	722.13
18000.0	1222	767	663	0	59	512.78	472.95	116	118	0	2	611.56	537.35
18100.0	1112	714	697	0	66	563.92	593.51	113	106	0	3	645.10	618.93
18200.0	1081	788	606	0	95	528.71	646.15	111	117	0	4	632.66	685.91
18300.0	1037	761	700	0	117	628.73	705.73	114	124	0	8	630.45	707.32
18400.0	1122	780	650	0	30	635.12	644.26	144	116	0	0	712.83	686.62
18500.0	918	792	680	0	25	644.57	674.95	128	117	0	1	713.29	712.97
18600.0	903	733	749	0	88	746.69	787.94	103	108	0	3	794.97	863.24
18700.0	904	814	723	0	91	705.16	800.78	101	98	0	4	824.54	938.65
18800.0	705	820	749	0	62	720.92	793.54	128	104	0	4	869.20	806.07
18900.0	739	790	741	0	18	816.29	843.88	146	115	0	1	966.06	900.86
19000.0	702	737	744	0	42	832.93	946.75	134	133	0	10	959.19	962.04
19100.0	662	722	617	0	88	859.31	1040.00	126	94	0	5	1013.73	1051.30
19200.0	679	728	624	0	75	861.29	1110.33	111	119	0	9	1070.06	1157.79
19300.0	690	908	693	0	39	834.39	988.84	149	93	0	5	1043.68	1097.37
19400.0	716	897	678	0	20	875.29	941.04	160	131	0	4	1038.21	1041.18
19500.0	676	855	753	0	28	991.76	1146.59	140	136	0	6	1082.38	1146.00
19600.0	571	855	650	0	107	1027.89	1220.16	155	113	0	10	1048.89	1126.26
19700.0	669	820	684	0	48	987.79	1257.57	153	116	0	12	1182.21	1364.85
19800.0	651	831	777	0	61	1087.28	1225.32	151	112	0	7	1174.93	1151.76
19900.0	508	808	691	0	0	1049.94	1271.36	177	125	0	4	1160.67	1239.00
20000.0	481	843	561	0	16	1048.49	1474.14	153	102	0	11	1299.20	1518.59
20100.0	609	840	733	0	49	1018.74	1177.42	161	141	0	8	1276.69	1448.32
20200.0	425	853	657	0	47	1001.89	1363.05	166	120	0	15	1199.08	1408.41
20300.0	594	933	629	0	0	1050.10	1443.91	194	121	0	11	1286.06	1587.21
20400.0	583	905	662	0	77	949.84	1233.20	182	127	0	14	1214.85	1337.49
20500.0	449	772	659	0	43	965.08	1396.49	156	118	0	11	1223.21	1483.77
20600.0	521	765	744	0	6	910.19	1300.12	171	124	0	10	1169.02	1443.56
20700.0	433	763	598	0	29	918.94	1499.40	165	103	0	17	1219.33	1408.80
20800.0	411	796	554	0	14	945.03	1501.37	160	97	0	18	1154.80	1597.62
20900.0	407	716	581	0	7	965.71	1637.82	163	124	0	12	1238.99	1755.59
21000.0	312	746	641	0	13	847.00	1663.22	172	107	0	14	1133.27	1922.15
21100.0	418	802	587	0	33	770.36	1529.73	168	88	0	16	1008.29	1563.34
21200.0	418	697	583	0	0	728.98	1644.46	142	125	0	15	987.58	1853.44
21300.0	438	652	631	0	1	753.54	1649.34	147	110	0	11	983.75	1504.09
21400.0	382	653	565	0	24	627.65	1433.30	120	101	0	17	788.76	1517.40
21500.0	438	629	587	0	72	532.24	1459.50	125	95	0	14	751.07	1493.73
21600.0	350	706	645	0	38	487.40	1530.00	145	124	0	15	718.13	1719.88
21700.0	387	634	689	0	22	467.63	1639.55	120	144	0	18	752.30	1818.13
21800.0	406	711	630	0	12	462.63	1400.90	154	96	0	9	686.21	1683.12
21900.0	444	660	558	0	15	306.42	1385.81	134	90	0	14	503.33	2007.10
22000.0	392	667	660	0	0	165.35	1386.23	142	121	0	18	267.49	1593.61
22100.0	325	514	659	0	0	94.59	1409.78	106	112	0	6	141.35	1716.51
22200.0	345	435	659	0	0	33.27	1264.65	72	126	0	9	40.05	1491.44
22300.0	403	347	548	0	0	21.57	1359.66	59	99	0	9	22.91	1695.96
22400.0	349	367	590	0	0	19.83	1399.32	66	119	0	13	24.06	1705.72
22500.0	339	341	550	0	0	17.28	1221.08	67	106	0	7	19.40	1419.31
22600.0	367	365	576	0	0	17.70	1099.57	70	119	0	10	16.68	1622.44

22700.0	392	348	546	0	0	20.64	1271.85	65	112	0	7	24.74	1575.93
22800.0	373	364	597	0	0	27.16	1258.66	63	123	0	9	28.19	1539.35
22900.0	272	316	494	0	0	22.64	1206.50	70	94	0	12	23.93	1503.97
23000.0	429	342	515	0	0	23.52	1134.12	60	102	0	8	27.24	1478.75
23100.0	310	409	517	0	0	23.53	923.72	82	110	0	8	23.74	1151.96
23200.0	518	433	588	0	0	26.80	792.14	62	107	0	5	28.61	1186.77
23300.0	340	451	508	0	0	30.00	813.40	90	91	0	4	34.51	1076.17
23400.0	377	365	535	0	0	16.75	1033.44	73	112	0	9	19.06	1064.52
23500.0	448	386	580	0	0	15.29	747.02	78	114	0	8	17.99	782.34
23600.0	405	416	552	0	0	21.60	697.84	73	103	0	0	23.16	814.77
23700.0	385	426	580	0	0	24.13	676.09	74	90	0	1	27.96	789.67
23800.0	355	357	431	0	0	17.22	653.47	72	92	0	1	20.61	965.19
23900.0	438	372	516	0	0	16.64	654.95	70	106	0	2	18.47	566.39
24000.0	464	460	469	0	0	25.35	583.59	80	89	0	5	29.01	800.64
24100.0	347	407	517	0	0	25.24	630.63	74	99	0	5	27.86	730.63
24200.0	539	422	519	0	0	19.39	483.97	70	102	0	4	25.09	619.02
24300.0	568	529	586	0	0	33.78	421.32	77	92	0	3	39.17	458.13
24400.0	501	498	557	0	0	40.99	433.88	97	96	0	2	47.71	416.91
24500.0	463	466	493	0	0	41.48	367.99	89	92	0	1	48.61	460.20
24600.0	439	472	489	0	0	48.04	440.11	86	98	0	0	58.32	511.12
24700.0	634	549	539	0	0	33.50	367.67	100	87	0	0	44.42	418.99
24800.0	492	575	590	0	0	42.97	361.13	75	120	0	0	40.57	545.53
24900.0	602	503	587	0	0	40.82	325.29	105	111	0	1	49.91	416.06
25000.0	640	598	551	0	0	39.04	296.11	101	100	0	0	47.25	402.94
25100.0	472	531	590	0	0	46.33	272.15	98	109	0	0	56.94	354.81
25200.0	474	577	658	0	0	47.46	238.55	110	120	0	0	57.60	345.48
25300.0	602	461	512	0	0	33.01	228.89	69	81	0	0	36.06	241.79
25400.0	726	637	676	0	0	49.92	207.07	98	104	0	0	61.90	294.69
25500.0	645	630	581	0	0	57.55	200.96	98	98	0	0	70.20	252.84
25600.0	813	630	611	0	0	54.65	186.40	123	90	0	0	72.23	143.32
25700.0	635	549	600	0	0	80.12	242.41	102	106	0	0	82.92	230.71
25800.0	814	570	662	0	0	98.43	189.89	85	105	0	0	95.29	246.85
25900.0	760	656	682	0	0	107.71	194.22	117	119	0	0	134.97	247.72
26000.0	840	723	702	0	0	121.30	183.50	128	107	0	0	136.11	202.47
26100.0	861	724	593	0	0	133.46	207.67	104	95	0	0	149.44	254.82
26200.0	976	757	599	0	0	148.34	237.46	133	93	0	0	178.33	319.53
26300.0	827	780	638	0	0	156.80	241.89	126	119	0	0	174.44	280.20
26400.0	842	778	606	0	0	184.72	281.49	138	101	0	0	210.62	248.16
26500.0	842	741	632	0	0	175.82	320.22	153	117	0	0	195.29	385.68
26600.0	734	819	693	0	0	186.05	318.52	135	106	0	0	213.15	399.51
26700.0	984	780	624	0	0	209.25	345.88	130	115	0	0	230.23	389.71
26800.0	840	832	732	0	0	207.42	337.65	175	123	0	0	244.25	427.38
26900.0	1164	767	762	0	0	196.08	354.20	136	124	0	0	236.64	421.00
27000.0	885	777	720	0	0	211.08	366.11	139	139	0	0	245.36	377.70
27100.0	933	758	682	0	0	232.16	413.40	144	131	0	0	260.80	458.17
27200.0	969	714	684	0	0	240.88	451.42	126	117	0	0	265.50	476.47
27300.0	1116	734	688	0	0	259.70	397.77	138	127	0	0	301.87	424.82
27400.0	1048	787	674	0	0	287.35	444.54	142	121	0	0	343.23	556.68
27500.0	1300	697	568	0	0	284.06	461.54	122	111	0	0	327.97	446.79
27600.0	1091	709	590	0	0	317.58	486.05	125	99	0	0	349.63	623.12
27700.0	1117	768	678	0	0	357.59	472.17	133	113	0	0	362.37	524.11
27800.0	963	784	595	0	23	372.05	576.22	113	139	0	0	381.95	654.38
27900.0	1114	687	637	0	22	397.63	511.46	108	104	0	2	401.84	656.28
28000.0	1043	873	720	0	3	408.18	572.10	137	111	0	0	471.76	601.68
28100.0	1128	799	696	0	40	451.82	565.49	136	106	0	2	487.77	589.42
28200.0	979	725	708	0	26	491.10	546.08	124	110	0	2	556.11	681.80
28300.0	1063	820	715	0	76	512.81	670.59	114	107	0	5	558.54	645.38
28400.0	1103	692	692	0	61	552.18	718.67	100	93	0	1	617.18	804.66
28500.0	1002	735	677	0	32	602.09	783.89	120	100	0	1	703.43	811.05
28600.0	819	863	702	0	44	585.26	823.18	131	128	0	4	638.15	798.58
28700.0	974	729	687	0	6	637.47	737.94	142	120	0	0	734.78	786.26
28800.0	838	783	690	0	66	630.18	876.10	133	117	0	2	735.35	982.08
28900.0	830	830	655	0	20	708.16	817.10	132	110	0	0	754.22	921.31
29000.0	858	766	684	0	23	717.54	957.49	151	111	0	3	866.11	969.53
29100.0	762	785	687	0	0	700.37	898.11	151	129	0	1	784.81	1004.78
29200.0	639	776	600	0	72	760.52	939.65	133	90	0	11	927.19	1104.48
29300.0	803	811	655	0	69	774.93	1085.26	146	111	0	6	840.42	1012.67
29400.0	800	715	588	0	126	770.25	1231.62	123	94	0	13	885.23	1165.23
29500.0	718	857	728	0	66	745.95	987.52	145	104	0	6	906.01	1112.16
29600.0	759	809	625	0	111	804.64	1101.50	163	102	0	9	989.24	1236.98
29700.0	586	871	699	0	78	809.63	1005.04	153	110	0	8	981.36	1134.55
29800.0	464	829	654	0	36	932.53	1188.12	148	112	0	9	1023.64	1117.52
29900.0	621	811	753	0	16	915.61	1103.91	144	106	0	8	1055.48	1287.74
30000.0	595	787	679	0	8	883.07	1193.70	141	121	0	4	1036.37	1414.02
30100.0	608	841	550	0	0	905.57	1334.28	139	116	0	3	1071.23	1556.44
30200.0	469	794	740	0	56	955.56	1373.65	152	133	0	14	1100.79	1627.94
30300.0	588	743	645	0	62	928.98	1437.42	146	112	0	10	1144.61	1414.40
30400.0	472	845	711	0	26	957.92	1295.53	147	111	0	10	1160.80	1338.77
30500.0	490	767	635	0	18	950.20	1621.71	149	88	0	9	1131.40	1622.44
30600.0	419	830	690	0	11	1009.82	1526.06	176	109	0	10	1143.09	1779.00
30700.0	580	775	709	0	0	817.38	1396.80	136	129	0	8	1081.78	1675.56
30800.0	410	793	673	0	1	869.65	1490.72	162	113	0	19	1111.11	1625.67
30900.0	549	801	657	0	43	818.98	1400.29	164	121	0	12	1098.31	1838.20
31000.0	508	793	695	0	0	825.99	1498.91	143	128	0	8	998.46	1783.42
31100.0	525	745	631	0	9	792.34	1422.55	157	105	0	7	960.19	1602.56
31200.0	498	760	717	0	0	717.23	1659.88	139	124	0	12	970.24	1904.19

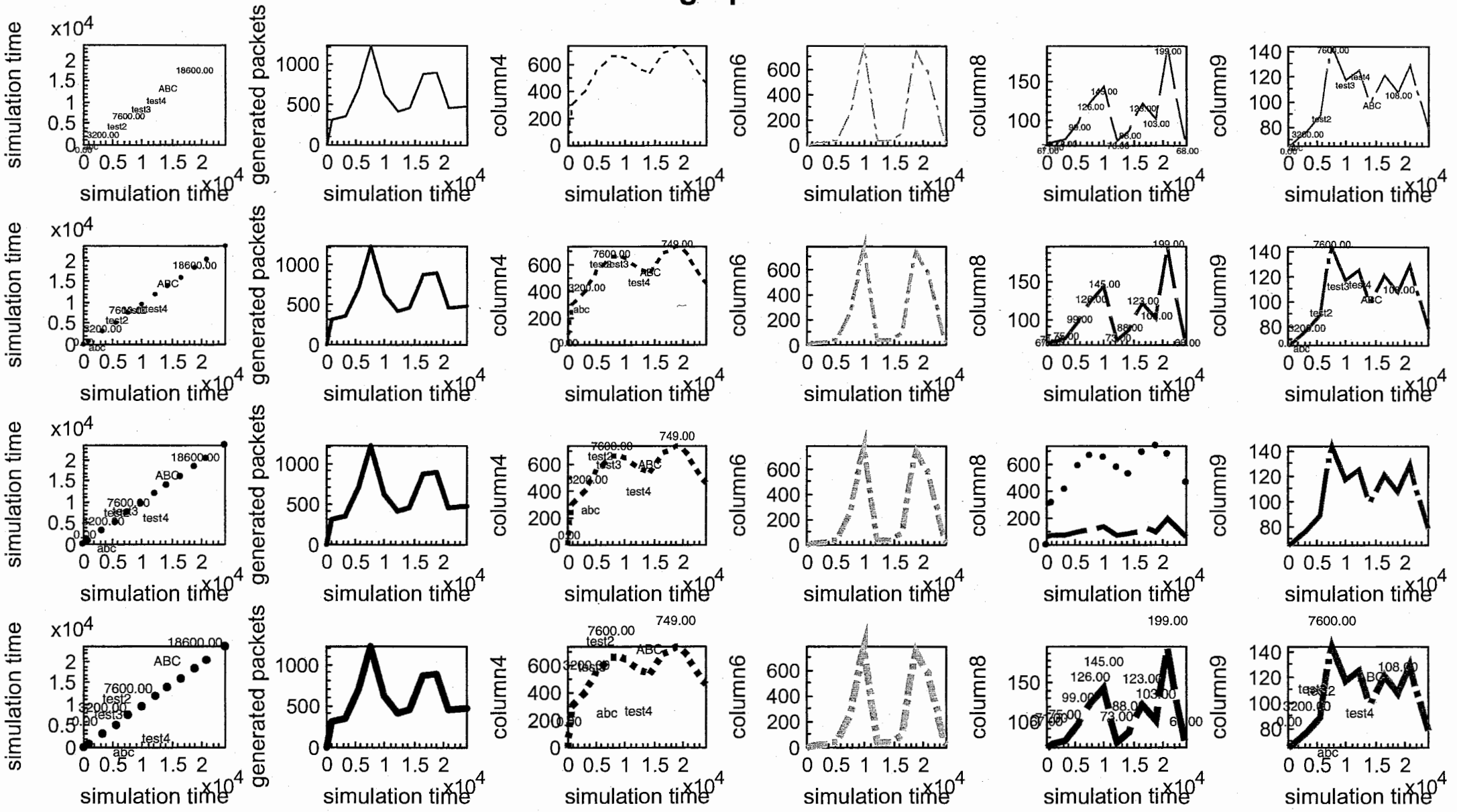
31300.0	350	811	704	0	0	697.52	1492.10	179	111	0	13	903.59	1794.25
31400.0	451	783	667	0	0	620.61	1605.41	168	120	0	17	798.09	1874.66
31500.0	379	728	599	0	0	530.71	1561.64	120	105	0	14	725.60	1648.51
31600.0	415	732	597	0	0	463.02	1726.80	150	126	0	16	563.37	1804.38
31700.0	492	650	566	0	0	396.24	1615.92	127	103	0	11	503.21	1612.83
31800.0	307	645	613	0	0	306.03	1688.58	130	98	0	3	385.52	1793.35
31900.0	438	599	625	0	0	262.76	1506.22	113	98	0	14	307.21	1717.05
32000.0	371	638	584	0	0	175.27	1473.91	145	93	0	12	219.61	1750.65
32100.0	366	536	620	0	0	77.78	1477.89	88	112	0	13	88.71	1927.95
32200.0	477	509	581	0	0	32.56	1370.22	92	88	0	8	43.88	1869.30
32300.0	276	354	613	0	0	20.54	1519.30	69	110	0	10	21.91	1778.20
32400.0	396	358	636	0	0	17.02	1459.38	76	123	0	9	18.63	1814.22
32500.0	418	343	594	0	0	15.50	1742.01	55	103	0	12	19.40	1935.42
32600.0	409	412	590	0	0	26.90	1592.39	66	121	0	12	31.98	1744.55
32700.0	423	445	596	0	0	31.79	1450.55	78	114	0	8	38.39	1895.17
32800.0	308	387	522	0	0	24.59	1603.30	82	110	0	13	27.46	1802.56
32900.0	373	298	532	0	0	16.98	1466.12	53	109	0	5	17.73	1660.58
33000.0	386	425	585	0	0	34.46	1342.79	60	116	0	5	38.14	1973.88
33100.0	299	301	521	0	0	24.29	1674.79	64	104	0	8	28.70	1896.19
33200.0	474	457	619	0	0	22.58	1117.33	61	104	0	6	26.18	1753.62
33300.0	313	372	611	0	0	23.61	1274.06	65	113	0	4	28.01	1617.12
33400.0	299	276	478	0	0	13.78	1382.56	54	117	0	4	14.28	1501.06
33500.0	451	378	540	0	0	16.76	1379.52	61	91	0	1	19.78	1648.15
33600.0	425	400	599	0	0	24.42	1036.96	68	93	0	5	26.56	1363.97
33700.0	480	481	605	0	0	33.83	943.25	75	97	0	1	35.71	1369.38
33800.0	477	516	624	0	0	29.67	1011.87	97	121	0	3	32.48	1528.42
33900.0	390	371	603	0	0	24.38	1062.29	75	120	0	2	23.58	1367.73
34000.0	375	427	633	0	0	28.15	885.90	79	120	0	3	30.81	1370.77
34100.0	443	350	540	0	0	18.23	824.50	60	105	0	0	22.26	1071.10
34200.0	378	396	532	0	0	30.03	908.51	68	117	0	1	30.96	1216.16
34300.0	435	460	473	0	0	35.55	707.98	71	82	0	1	38.21	1081.70
34400.0	428	445	489	0	0	33.86	802.09	88	108	0	0	38.49	1187.01
34500.0	396	446	527	0	0	27.82	698.24	93	105	0	0	33.81	1154.04
34600.0	438	382	543	0	0	21.17	564.65	66	112	0	0	23.42	905.67
34700.0	513	447	543	0	0	26.84	588.97	79	108	0	0	31.82	776.06
34800.0	534	563	564	0	0	30.72	437.92	93	112	0	0	34.40	758.29
34900.0	497	529	598	0	0	32.67	451.17	109	122	0	0	36.11	520.55
35000.0	503	482	552	0	0	31.12	412.43	83	102	0	0	38.36	508.25
35100.0	532	483	617	0	0	28.46	360.68	98	120	0	0	34.32	433.05
35200.0	668	578	622	0	0	37.33	306.84	111	116	0	0	41.89	381.19
35300.0	607	556	625	0	0	48.03	252.23	95	119	0	0	51.66	322.67
35400.0	584	536	555	0	0	61.86	379.51	97	110	0	0	72.15	519.11
35500.0	593	646	526	0	0	53.56	256.37	112	108	0	0	68.78	330.12
35600.0	705	553	569	0	0	60.98	306.50	95	97	0	0	64.63	233.77
35700.0	694	675	717	0	0	65.80	273.46	123	114	0	0	76.12	396.91
35800.0	785	609	570	0	0	74.49	289.22	113	103	0	0	98.23	382.40
35900.0	589	696	549	0	0	84.74	248.46	130	124	0	0	98.68	295.01
36000.0	836	711	665	0	0	81.91	232.75	146	114	0	0	100.13	210.93
36100.0	1029	637	667	0	0	95.62	265.96	106	119	0	0	118.77	382.56
36200.0	902	571	567	0	0	116.54	272.37	109	109	0	0	142.01	327.10
36300.0	852	649	604	0	0	126.65	300.12	85	121	0	0	132.36	349.05
36400.0	1016	767	657	0	0	168.78	287.86	126	112	0	0	174.14	390.24
36500.0	836	660	638	0	0	188.64	315.20	121	137	0	0	193.69	358.75
36600.0	841	695	649	0	0	203.19	388.06	129	133	0	0	222.67	452.58
36700.0	827	682	617	0	0	205.12	310.60	123	108	0	0	202.40	330.86
36800.0	942	715	713	0	0	233.68	325.25	130	122	0	0	248.47	331.90
36900.0	903	741	665	0	0	260.95	392.21	139	126	0	0	261.74	360.62
37000.0	936	724	721	0	78	262.17	420.32	134	133	0	4	291.09	414.28
37100.0	1073	730	714	0	0	311.43	384.55	130	133	0	0	348.41	404.54
37200.0	1033	680	640	0	0	321.65	398.00	137	126	0	0	327.33	437.09
37300.0	1074	733	672	0	95	351.92	478.25	110	89	0	4	393.67	465.19
37400.0	1048	756	630	0	17	378.02	498.63	131	132	0	0	409.80	460.05
37500.0	1062	707	642	0	24	402.29	492.38	112	112	0	1	436.87	514.83
37600.0	967	678	692	0	22	415.38	493.55	109	123	0	2	459.26	507.98
37700.0	1046	700	568	0	39	421.90	517.63	127	99	0	4	468.70	477.46
37800.0	1088	768	682	0	49	422.55	533.48	129	105	0	4	479.58	667.73
37900.0	1145	756	650	0	20	502.92	564.35	139	123	0	4	578.99	540.57
38000.0	930	812	624	0	67	516.30	602.29	128	112	0	4	628.27	664.88
38100.0	967	923	674	0	74	519.69	635.55	161	119	0	5	585.28	586.53
38200.0	1083	723	721	0	43	565.08	723.43	123	133	0	4	575.98	711.93
38300.0	916	901	754	0	26	573.32	709.58	156	127	0	2	670.11	681.04
38400.0	825	760	756	0	27	567.05	658.84	138	102	0	3	688.43	728.22
38500.0	878	708	664	0	127	621.85	703.44	141	126	0	6	694.72	671.17
38600.0	726	799	677	0	66	671.18	797.29	131	125	0	5	677.10	747.33
38700.0	749	760	652	0	74	715.50	854.16	150	96	0	7	847.99	945.49
38800.0	965	787	596	0	80	707.39	857.36	142	100	0	5	771.83	1031.71
38900.0	742	762	735	0	79	720.24	887.21	138	131	0	7	819.88	895.91
39000.0	770	927	745	0	45	749.22	973.21	170	117	0	4	873.51	889.67
39100.0	855	784	699	0	3	716.10	969.77	150	120	0	2	848.38	995.63
39200.0	704	782	658	0	43	767.27	879.79	132	98	0	7	875.32	1033.95
39300.0	701	796	647	0	99	813.06	1002.11	151	121	0	9	925.96	970.11
39400.0	663	800	625	0	103	835.93	1093.49	151	136	0	13	915.05	1202.89
39500.0	593	762	645	0	132	807.19	1106.49	124	112	0	15	903.40	1091.16
39600.0	615	752	779	0	72	837.44	1132.06	141	137	0	9	1000.35	1174.10
39700.0	589	947	709	0	15	806.90	1073.41	192	122	0	17	932.24	1065.13
39800.0	526	799	713	0	29	863.59	1194.29	152	113	0	13	1067.89	1194.18

39900.0	557	775	638	0	64	826.06	1120.90	157	111	0	13	996.20	1268.30
40000.0	533	875	659	0	36	789.45	1331.77	152	120	0	22	1051.50	1310.94
40100.0	558	895	610	0	63	828.54	1177.07	163	105	0	16	1001.51	1271.69
40200.0	521	750	566	0	48	880.40	1314.12	155	98	0	15	1081.17	1375.93
40300.0	503	829	689	0	14	834.14	1191.73	163	118	0	15	998.33	1281.28
40400.0	474	854	600	0	41	777.41	1375.28	188	115	0	14	999.31	1373.82
40500.0	482	855	521	0	15	757.13	1223.61	157	98	0	11	905.24	1178.36
40600.0	498	866	721	0	0	762.08	1158.10	162	105	0	16	924.06	1442.48
40700.0	485	823	648	0	1	663.50	1303.51	157	112	0	11	821.02	1350.21
40800.0	453	829	687	0	0	585.95	1270.46	199	129	0	15	746.20	1349.17
40900.0	407	602	598	0	0	712.09	1461.73	137	109	0	16	882.00	1443.66
41000.0	395	715	705	0	0	593.44	1401.22	135	123	0	18	809.51	1461.76
41100.0	464	694	610	0	0	535.14	1243.72	135	96	0	18	718.43	1415.17
41200.0	297	666	687	0	0	416.61	1316.68	150	111	0	13	592.96	1592.22
41300.0	411	668	646	0	0	386.33	1484.77	128	109	0	31	484.45	1504.77
41400.0	451	677	580	0	0	255.25	1390.25	131	96	0	11	287.26	1471.45
41500.0	291	634	625	0	0	158.63	1314.85	131	118	0	11	202.24	1604.35
41600.0	457	523	505	0	0	90.07	1236.45	114	92	0	13	147.69	1466.14
41700.0	379	474	574	0	0	36.92	1220.61	93	123	0	8	50.26	1410.92
41800.0	348	357	631	0	0	20.84	1339.73	68	116	0	13	22.03	1542.15
41900.0	370	424	541	0	0	22.76	1328.74	73	88	0	13	25.08	1394.72
42000.0	365	312	664	0	0	12.99	1311.95	60	103	0	12	16.26	1257.21
42100.0	336	373	487	0	0	19.81	1539.90	77	89	0	10	22.44	1605.96
42200.0	369	350	507	0	0	14.92	1384.72	72	108	0	12	17.27	1360.52
42300.0	350	305	590	0	0	17.47	1290.03	49	105	0	10	17.05	1659.58
42400.0	373	417	520	0	0	23.52	1219.87	85	102	0	11	25.15	1404.59
42500.0	436	363	599	0	0	22.02	1058.45	62	109	0	5	21.16	1477.30
42600.0	363	437	571	0	0	28.51	1090.66	80	101	0	5	31.43	1367.99
42700.0	298	304	591	0	0	26.85	1318.56	60	113	0	6	31.32	1573.03
42800.0	419	409	515	0	0	23.45	1087.52	72	105	0	9	27.15	1292.50
42900.0	416	356	542	0	0	22.90	1087.36	66	123	0	9	27.61	1399.37
43000.0	375	457	645	0	0	18.28	797.81	81	115	0	2	23.00	1142.21
43100.0	343	343	515	0	0	16.37	1014.50	74	121	0	3	17.13	1391.37
43200.0	350	337	474	0	0	16.27	932.86	58	97	0	1	18.79	1221.94
43300.0	347	330	501	0	0	13.49	855.02	80	113	0	6	14.62	1126.80
43400.0	486	350	433	0	0	21.34	690.28	52	90	0	1	22.36	929.67
43500.0	393	564	638	0	0	31.58	588.25	91	109	0	0	37.06	996.79
43600.0	441	354	530	0	0	18.46	674.33	69	125	0	1	19.80	833.87
43700.0	393	422	532	0	0	26.44	436.66	82	95	0	0	28.53	699.46
43800.0	409	391	492	0	0	25.04	525.70	68	109	0	2	31.77	747.92
43900.0	480	443	465	0	0	25.62	377.39	68	79	0	0	26.58	616.31
44000.0	408	496	548	0	0	29.51	276.93	87	110	0	0	33.18	394.45
44100.0	329	319	447	0	0	20.65	356.73	68	78	0	0	24.53	372.44
44200.0	418	412	447	0	0	27.97	375.44	68	84	0	0	32.33	541.69
44300.0	404	413	534	0	0	19.21	298.53	76	104	0	0	23.06	410.51
44400.0	371	366	491	0	0	15.34	285.72	76	108	0	0	18.10	497.29
44500.0	343	382	476	0	0	19.57	279.87	77	109	0	0	20.53	385.11
44600.0	482	410	549	0	0	18.13	256.40	77	100	0	0	19.20	307.32
44700.0	555	512	492	0	0	28.44	168.91	79	81	0	0	30.95	196.71
44800.0	527	476	548	0	0	34.02	200.36	89	103	0	0	42.68	311.65
44900.0	634	570	629	0	0	34.83	127.73	104	112	0	0	42.62	172.64
45000.0	555	539	514	0	0	47.26	125.05	88	88	0	0	53.04	193.74
45100.0	471	517	508	0	0	53.01	116.67	104	105	0	0	61.11	140.84
45200.0	726	518	499	0	0	43.28	112.99	93	92	0	0	49.45	128.98
45300.0	639	640	567	0	0	63.64	117.99	100	90	0	0	77.48	150.73
45400.0	677	669	622	0	0	67.14	133.68	113	98	0	0	81.96	145.23
45500.0	650	597	618	0	0	69.28	154.45	109	114	0	0	80.97	170.76
45600.0	554	579	532	0	0	84.37	157.52	99	93	0	0	89.75	195.83
45700.0	540	631	602	0	0	88.42	183.43	141	111	0	0	97.60	201.73
45800.0	891	621	690	0	0	64.51	151.18	101	116	0	0	90.41	170.10
45900.0	708	690	667	0	0	76.36	149.87	124	128	0	0	91.23	180.43
46000.0	773	667	596	0	0	84.83	154.56	107	108	0	0	97.21	203.81
46100.0	776	661	652	0	0	107.48	156.23	113	109	0	0	114.15	180.91
46200.0	887	706	694	0	0	107.67	184.13	134	128	0	0	126.61	201.46
46300.0	771	745	724	0	0	115.43	185.23	126	109	0	0	147.45	240.26
46400.0	1175	783	652	0	0	141.75	202.23	127	111	0	0	175.01	211.97
46500.0	1001	744	703	0	0	132.71	194.87	117	113	0	0	154.20	249.30
46600.0	1017	701	572	0	0	162.11	236.76	132	84	0	0	196.96	265.61
46700.0	994	691	682	0	0	173.11	233.27	96	108	0	0	190.10	276.90
46800.0	973	712	655	0	0	204.69	255.18	105	103	0	0	218.98	283.28
46900.0	1106	721	698	0	0	234.48	278.62	107	112	0	0	272.88	310.62
47000.0	990	698	688	0	0	260.85	314.06	117	122	0	0	293.14	311.50
47100.0	1135	739	619	0	0	284.43	284.83	121	106	0	0	303.31	329.38
47200.0	1173	687	644	0	0	327.70	358.09	104	106	0	0	371.47	379.12
47300.0	1171	713	604	0	0	374.12	419.57	129	107	0	0	404.26	433.81
47400.0	1161	738	659	0	0	382.46	432.38	105	92	0	0	420.28	456.23
47500.0	1176	782	776	0	0	400.51	407.42	146	124	0	0	477.46	441.02
47600.0	1245	725	674	0	0	450.77	460.44	134	89	0	0	511.07	484.95
47700.0	1179	766	604	0	0	477.21	489.52	138	113	0	0	543.12	532.51
47800.0	1231	689	717	0	0	481.06	495.15	119	94	0	0	542.67	529.04
47900.0	1295	719	631	0	31	525.22	553.67	98	100	0	0	613.81	633.34
48000.0	1137	767	721	0	36	499.07	590.78	113	119	0	1	578.63	566.89
48100.0	1054	707	638	0	63	597.52	625.78	105	124	0	2	614.77	697.95
48200.0	954	742	756	0	52	606.98	617.53	112	122	0	1	606.50	683.51
48300.0	855	765	678	0	43	622.45	703.47	123	114	0	2	676.36	753.79
48400.0	925	777	693	0	114	670.61	735.52	124	110	0	3	784.07	748.30

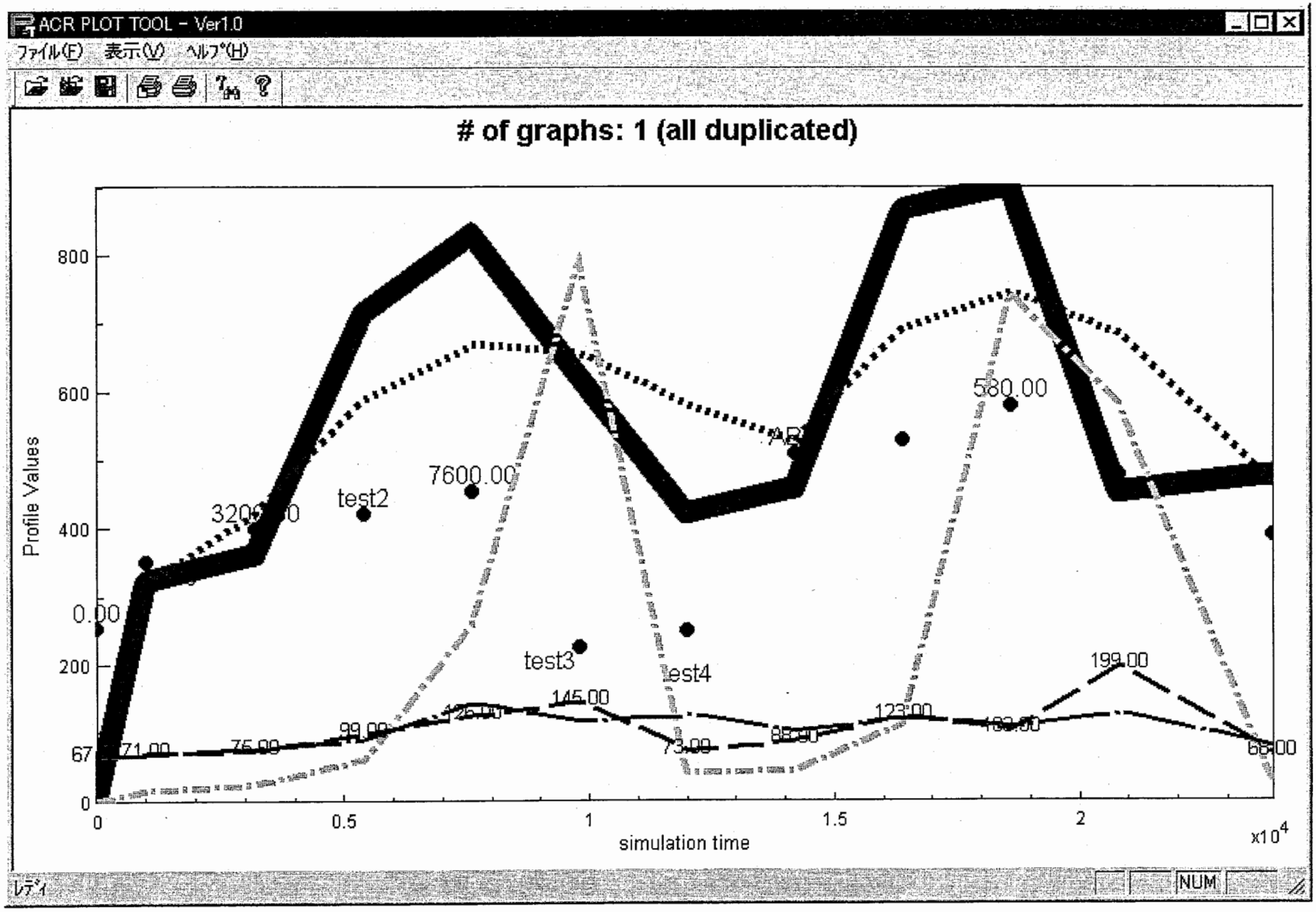
48500.0	897	767	701	0	64	728.72	740.79	130	114	0	2	785.05	735.54
48600.0	886	804	666	0	81	742.10	757.63	136	138	0	4	854.13	819.04
48700.0	952	737	707	0	1	787.52	863.93	144	127	0	0	857.28	862.51
48800.0	759	743	646	0	54	821.52	860.23	129	110	0	4	861.26	809.31
48900.0	770	727	656	0	23	820.96	973.01	118	112	0	2	947.39	931.80
49000.0	934	812	709	0	62	904.06	940.16	134	126	0	5	955.35	1041.07
49100.0	835	746	741	0	143	880.59	914.02	141	117	0	11	981.79	961.03
49200.0	691	801	689	0	87	920.41	1003.03	126	105	0	7	1057.10	1064.49
49300.0	753	768	656	0	109	974.12	1068.71	123	99	0	5	1145.66	999.28
49400.0	738	731	687	0	133	1031.13	1087.27	119	109	0	9	1059.22	1204.85
49500.0	766	770	695	0	140	962.03	1056.64	111	117	0	11	1071.82	1230.94
49600.0	540	829	707	0	48	1054.33	1076.71	130	111	0	6	1187.34	1048.04
49700.0	616	924	689	0	44	992.71	1098.58	141	115	0	9	1177.82	1242.04
49800.0	532	823	612	0	45	1070.58	1269.38	156	102	0	5	1249.25	1369.54
49900.0	644	896	571	0	125	1050.82	1256.83	162	100	0	7	1202.33	1356.84
50000.0	468	782	610	0	73	1167.65	1425.50	146	100	0	12	1281.76	1359.53

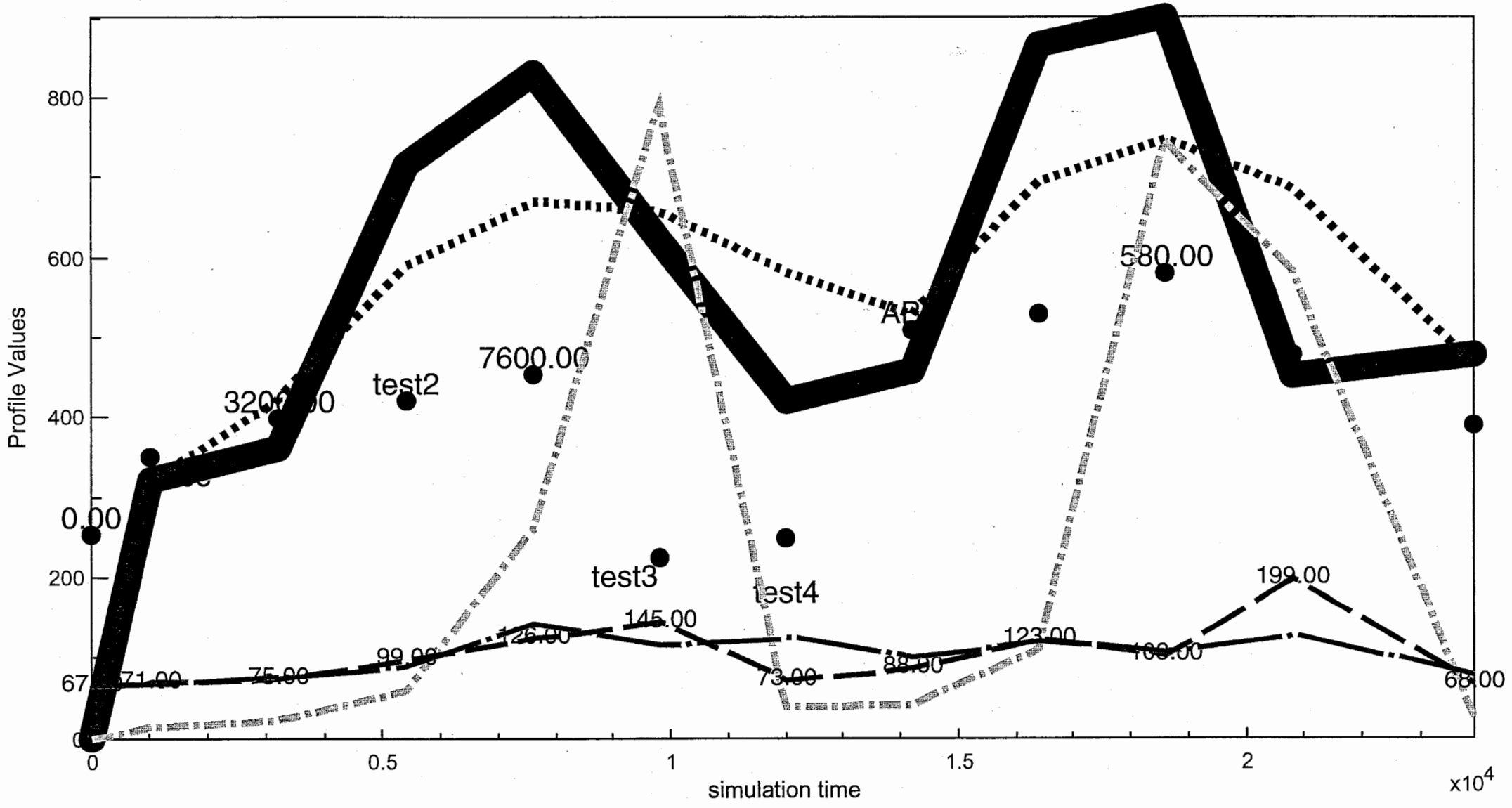

ACR PLOT TOOL - Ver1.0
ファイル(F) 表示(V) ヘルプ(H)
# of graphs: 24
simulation time
generated packets
column4
column6
column8
column9
x10^4
レディ
NUM

of graphs: 24


simulation time
generated packets
column4
column6
column8
column9
x10^4
0 0.5 1 1.5 2
18600.00
ABC
test4
test3
7600.00
test2
3200.00
abc
749.00
145.00
126.00
123.00
103.00
99.00
88.00
73.00
199.00
108.00

```
# of graphs: 24
25 12   11
1 1  0  dotcolor=#105523 dotsize=1 label=11 labelsize=6 labeldistance=0
1 2  1  dotcolor=2 dotsize=1 label=0 labelsize=6 labeldistance=0
1 4  2  dotcolor=8 dotsize=1 labeldistance=0
1 6  3  dotcolor=4 dotsize=1 label=0 labelsize=6 labeldistance=0
1 8  4  dotcolor=1 dotsize=1 label=8 labelsize=6 labeldistance=0
1 9  5  dotcolor=9 dotsize=1 label=11 labelsize=6 labeldistance=0
1 1  0  dotcolor=#105523 dotsize=2 label=11 labelsize=7 labeldistance=5
1 2  1  dotcolor=2 dotsize=2 label=0 labelsize=7 labeldistance=5
1 4  2  dotcolor=8 dotsize=2 label=11 labelsize=7 labeldistance=5
1 6  3  dotcolor=4 dotsize=2 label=0 labelsize=7 labeldistance=5
1 8  4  dotcolor=1 dotsize=2 label=8 labelsize=7 labeldistance=5
1 9  5  dotcolor=9 dotsize=2 label=11 labelsize=7 labeldistance=5
1 1  0  dotcolor=#105523 dotsize=3 label=11 labelsize=8 labeldistance=10
1 2  1  dotcolor=2 dotsize=3 label=0 labelsize=8 labeldistance=10
1 4  2  dotcolor=8 dotsize=3 label=11 labelsize=8 labeldistance=10
1 6  3  dotcolor=4 dotsize=3 label=0 labelsize=8 labeldistance=10
1 8  4  dotcolor=1 dotsize=3
1 4  0  duplicate=1 dotcolor=2 dotsize=3
1 9  5  dotcolor=9 dotsize=3
1 1  0  dotcolor=#105523 dotsize=4 label=11 labelsize=9 labeldistance=20
1 2  1  dotcolor=2 dotsize=4 label=0 labelsize=9 labeldistance=20
1 4  2  dotcolor=8 dotsize=4 label=11 labelsize=9 labeldistance=20
1 6  3  dotcolor=4 dotsize=4 label=0 labelsize=9 labeldistance=20
1 8  4  dotcolor=1 dotsize=4 label=8 labelsize=9 labeldistance=20
1 9  5  dotcolor=9 dotsize=4 label=11 labelsize=9 labeldistance=20
simulation time
generated packets
@label
column4
@label
column6
@label
column8
column9
@label
@label
     0.0     0     0     0     0     0.0      0.0     67    66     0.0    @x
  1000.0   323   333   323     0   16.08    16.25     71    69   17.43    @000abc
  3200.0   361   411   425     0   24.38    26.65     75    78   27.37    @x
  5400.0   718   635   592     0   58.53    66.17     99    90   68.14    @090test2
  7600.0  1232   643   672     0  259.54   342.50    126   144  411.55    @x
  9800.0   622   734   658     0  792.24  1076.56    145   118 1179.80    @180test3
 12000.0   421   444   582     0   40.65  1559.91     73   126 1751.07    @270test4
 14200.0   460   570   529     0   44.94   435.38     88   101  531.37    ABC
 16400.0   868   699   695     0  112.98   225.84    123   121  222.06    @0
 18600.0   903   733   749    88  746.69   787.94    103   108  863.24    @y
 20800.0   453   829   687     0  585.95  1270.46    199   129 1349.17    @0
 23900.0   480   443   465     0   25.62   377.39     68    79  616.31    @0
```


ACR PLOT TOOL - Ver1.0
ファイル(F) 表示(V) ヘルプ(H)
# of graphs: 1 (all duplicated)
Profile Values
simulation time
x10^4
800
600
400
200
67
0
0
0.5
1
1.5
2
test2
7600.00
test3
test4
580.00
199.00
145.00
123.00
99.00
75.00
71.00
73.00
レディ
NUM


# of graphs: 1 (all duplicated)
Profile Values
simulation time
x10^4
0
0.5
1
1.5
2
200
400
600
800
test2
7600.00
test3
test4
580.00
199.00
145.00
126.00
99.00
75.00
73.00
88.00
123.00
0.00

```
# of graphs: 1 (all duplicated)
6 12   12
1 12  0  dotcolor=#105523 dotsize=10 label=11 labelsize=20 labeldistance=20
1 2  1  duplicate=1 dotcolor=2 dotsize=15 label=0 labelsize=16 labeldistance=10
1 4  2  duplicate=1 dotcolor=8 dotsize=5 labeldistance=20
1 6  3  duplicate=1 dotcolor=4 dotsize=4 label=0 labelsize=16 labeldistance=10
1 8  4  duplicate=1 dotcolor=1 dotsize=3 label=8 labelsize=16 labeldistance=10
1 9  5  duplicate=1 dotcolor=9 dotsize=3 label=0 labelsize=20 labeldistance=10
simulation time
generated packets
@label
column4
@label
column6
@label
column8
column9
@label
@label
Profile Values
     0.0     0     0     0   0    0.0     0.0     67     66     0.0   @x          254
  1000.0   323   333   323   0  16.08   16.25     71     69   17.43   @000abc     350
  3200.0   361   411   425   0  24.38   26.65     75     78   27.37   @x          400
  5400.0   718   635   592   0  58.53   66.17     99     90   68.14   @090test2   421
  7600.0   832   643   672   0 259.54  342.50    126    144  411.55   @x          454
  9800.0   622   734   658   0 792.24 1076.56    145    118  879.80   @180test3   225
 12000.0   421   444   582   0  40.65  859.91     73    126  851.07   @270test4   250
 14200.0   460   570   529   0  44.94  435.38     88    101  531.37   ABC         511
 16400.0   868   699   695   0 112.98  225.84    123    121  222.06   @0          530
 18600.0   903   733   749  88 746.69  787.94    103    108  863.24   @y          580
 20800.0   453   829   687   0 585.95  970.46    199    129  849.17   @0          480
 23900.0   480   443   465   0  25.62  377.39     68     79  616.31   @0          390
```

A2 プログラムソースリスト

```
// Line.cpp
//
#include "stdafx.h"
#include "DashLine.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif
//
//////////////////////////////////////////////////////////////////////
// CDashLine

CDashLine::CDashLine(CDC& dc, unsigned* pattern, unsigned count)
        :m_DC(dc), m_Pattern(0)
{
        SetPattern(pattern, count);
}

CDashLine::~CDashLine()
{
        delete[] m_Pattern;
}

void CDashLine::SetPattern(unsigned* pattern, unsigned count)
{
        m_Count = count;
        delete[] m_Pattern;

        if (m_Count) {
                m_Pattern = new UINT[count];
                memcpy(m_Pattern, pattern, count*sizeof(UINT));
        }
        else
                m_Pattern = 0;

        Reset();
}

void CDashLine::Reset()
{
        m_CurPat = 0;
        if (m_Count) {
                m_CurStretch = m_Pattern[0];
        }
}

void CDashLine::Bresenham(LONG x, LONG y)
{
        LONG dx = x - m_CurPos.x;
        LONG dy = y - m_CurPos.y;
        LONG *p1, *p2, *pd1, *pd2;

        if (abs(dx) >= abs(dy)) {
                p1 = &m_CurPos.x;
                p2 = &m_CurPos.y;
                pd1 = &dx;
                pd2 = &dy;
        }
        else {
                p1 = &m_CurPos.y;
                p2 = &m_CurPos.x;
                pd1 = &dy;
                pd2 = &dx;
        }

        int max = abs(*pd1);
        int dec = abs(*pd2);
        int s1 = (*pd1 >= 0) ? 1: -1;
        int s2 = (*pd2 >= 0) ? 1: -1;
        int val = max;
```

```
        for (int i=0; i<max; i++) {
                val -= dec;
                if (val <= 0) {
                        *p2 += s2;
                        val += max;
                }
                *p1 += s1;

                m_CurStretch--;

                if (!m_CurStretch) {
                        m_CurPat = (m_CurPat+1) % m_Count;
                        m_CurStretch = m_Pattern[m_CurPat];

                        if (m_CurPat % 2) m_DC.LineTo(m_CurPos);
                        else              m_DC.MoveTo(m_CurPos);

                        if (i == max-1) return;
                }
        }

        if (!(m_CurPat % 2)) m_DC.LineTo(m_CurPos);
}

void CDashLine::MoveTo(int x, int y)
{
        Reset();

        m_CurPos.x = x;
        m_CurPos.y = y;

        m_DC.MoveTo(m_CurPos);
}


void CDashLine::LineTo(int x, int y)
{
        if (!m_Count) {
                m_DC.LineTo(x, y);
                m_CurPos.x = x; m_CurPos.y = y;
                return;
        }

        Bresenham(x, y);
}

void CDashLine::Polyline(POINT *pos, int num)
{
        if (num < 2) return;

        MoveTo(pos[0]);

        for (int i=1; i<num; i++) LineTo(pos[i]);
}
```

```
// DashLine.h :
////////////////////////////////////////////////////////////////////////////
// CDashLine

class CDashLine
{
public:
        CDashLine(CDC&, unsigned*, unsigned);
        ~CDashLine();
        void SetPattern(unsigned*, unsigned);

protected:
        CDC&  m_DC;
        unsigned int    m_CurPat;
        unsigned int    m_Count;
        unsigned int    m_CurStretch;
        unsigned int*   m_Pattern;
        CPoint                  m_CurPos;

        void Reset();
        void Bresenham(LONG, LONG);

public:
        void MoveTo(const POINT& p) {MoveTo(p.x, p.y);}
        void MoveTo(int, int);
        void LineTo(const POINT& p) {LineTo(p.x, p.y);}
        void LineTo(int, int);
        void Polyline(POINT*, int);
};
```

```
/*
 * ACR PLOT TOOL Version 1.0
 *    (c) Copyright 1999
 *    ATR Adaptive Communications Research Laboratories
 *    All Rights Reserved
 */

/*
 * draw.cpp -
 */
#include "stdafx.h"

#include "plotdef.h"
#include "graph.h"

Graph PsOpen(DATA *, int);

int DrawGraph(DATA *data, int bRdraw)
{
        int   col, colx, coly;
        int   collabel;
        float ratio;
        float size;
        float dotratio;
        Number *lfp;    /* ラベルデータがデータ列の時のポインタ */
        char   *lcp;    /* ラベルデータがラベル列の時のポインタ */

        if (bRdraw == FALSE) {
                /* グラフの初期化 */
                GrInit();
        }

        gPSOnlyMode = (data->mode==WIN_DRAW_MODE)? FALSE: TRUE;

        for (int i=0; i<data->nGraphs; i++) {
                colx = data->colX[i];
                coly = data->colY[i];
                collabel = data->colLabel[i];
                lfp = NULL;
                lcp = NULL;

                if (bRdraw == FALSE) {
                        /* グラフのオープン */
                        if (gPSOnlyMode == FALSE) gGraphNo[i] = GrOpen(data, i);
                        else                      gGraphNo[i] = PsOpen(data, i);
                }
                /* 描画ウィンドウのサイズの設定 */
                GrSetWindowSize(gGraphNo[i], data->winWidth[i], data->winHeight[i]);

                if (bRdraw == FALSE) {
                        /* グラフのプロット */
                        /* ラベルデータへのポインタを指定する */
                        if (collabel > 0) {
                                if (data->colHeader[collabel-1][0] != '@') {
                                        lfp = data->table + (collabel-1)*data->nRows;
                                }
                                else {
                                        col = 0;
                                        for (int j=0; j<collabel; j++ ) {
                                                if (data->colHeader[j][0] == '@') col++;
                                        }
                                        lcp = data->table_label + (col-1)*data->nRows*MAX_
LABELDATA_LEN;
                                }
                        }
                        /* データへのポインタを渡す */
                        GrPlot(gGraphNo[i],
                                data->table + (colx-1)*data->nRows,
                                data->nValidRows,
                                data->table + (coly-1)*data->nRows,
                                data->nValidRows,
                                lfp,
                                lcp);
                        /* グラフのプロパティの設定 */
                        ratio = data->aspectRatio;
                        dotratio = data->dotratio;
                        size = data->dotSize[i]*dotratio;

                        GrSet(gGraphNo[i],
                                GR_MAX_EXP, 3,
                                GR_XLABEL, data->colHeader[colx-1],
                                GR_YLABEL, data->colHeader[coly-1],
                                GR_BOX, TRUE,
                                GR_LINE_STYLE,
                                (data->lineType[i]==0 ? GR_LINE_STYLE_DOT:
                                 (data->lineType[i]==1 ? GR_LINE_STYLE_LINE:
                                      (data->lineType[i]==2 ? GR_LINE_STYLE_DASH:
                                       (data->lineType[i]==3 ? GR_LINE_STYLE_DOTTEDLINE:

                                        (data->lineType[i]==4 ? GR_LINE_STYLE_DASH_LONG:

                                         (data->lineType[i]==5 ? GR_LINE_STYLE_DOTTEDLIN
E_LONG:
                                                GR_LINE_STYLE_LINE))))))),
                                GR_DOT_SIZE, &size,
                                GR_DECORATION_COLOR, "",
                                GR_GRAPH_TYPE, GR_TYPE_XY,
                                GR_ASPECT_RATIO, &ratio,
                                GR_DUPLICATE, data->duplicate[i],
                                GR_DOTCOLOR, data->dot_color[i],
                                GR_DOTLABEL_SIZE, data->dotlabel_fontsize[i],
                                GR_DOTLABEL_DIST, data->dotlabel_dist[i],
                                NULL);
                }
                if (gPSOnlyMode == FALSE) {
                        GrAttr  *p;
                        p = gGraph + gGraphNo[i];
                        if (bRdraw == FALSE) {
                                // 色の設定
                                GrAttr  *p;
                                p = gGraph + gGraphNo[i];
                                if (p->dot_color[0] == '#') {
                                        unsigned int rgb[3];
                                        for (int ii=0,jj=1; ii<3; ii++) {
                                                if (p->dot_color[jj] > 'F')      rgb[ii] =
 16 * (p->dot_color[jj] - 'W');
                                                else if (p->dot_color[jj] > '9') rgb[ii] =
 16 * (p->dot_color[jj] - '7');
                                                else                             rgb[ii] =
 16 * (p->dot_color[jj] - '0');
                                                jj++;
                                                if (p->dot_color[jj] > 'F')      rgb[ii] +
= (p->dot_color[jj] - 'W');
                                                else if (p->dot_color[jj] > '9') rgb[ii] =
 16 * (p->dot_color[jj] - '7');
                                                else                             rgb[ii] +
= (p->dot_color[jj] - '0');
                                                jj++;
                                        }
                                        p->line_col = RGB(rgb[0], rgb[1], rgb[2]);
                                }
                        }
                        // 絶対座標の設定
                        p->absx = (float)(data->winX[i] - data->winX[0]);
                        p->absy = (float)(data->winY[i] - data->winY[0]);
                }
        }

        return 0;
}

int SetPlotWindowSize(DATA *data)
{
        int i;
        ulong wPitch, hPitch;
        int wCount, hCount;
```

```
	int offset;
	int win_num = -1;

	offset = (int)(DEFAULT_TITLE_WINDOW_HEIGHT * data->mainWinHeight);

	/* 横方向表示の場合 */
	if (data->orientation == PAPER_LANDSCAPE) {
		switch (data->nWindows) {
		  case 1:
			wCount = 1; hCount = 1; break;
		  case 2:
			wCount = 2; hCount = 1; break;
		  case 3:
			wCount = 3; hCount = 1; break;
		  case 4:
			wCount = 2; hCount = 2; break;
		  case 5:
		  case 6:
			wCount = 3; hCount = 2; break;
		  case 7:
		  case 8:
			wCount = 4; hCount = 2; break;
		  case 9:
			wCount = 3; hCount = 3; break;
		  case 10:
		  case 11:
		  case 12:
			wCount = 4; hCount = 3; break;
		  case 13:
		  case 14:
		  case 15:
			wCount = 5; hCount = 3; break;
		  case 16:
			wCount = 4; hCount = 4; break;
		  case 17:
		  case 18:
		  case 19:
		  case 20:
			wCount = 5; hCount = 4; break;
		  default:
			wCount = 6; hCount = 4; break;
		}
	}
	/* 縦方向表示の場合 */
	else {
		switch (data->nWindows) {
		  case 1:
			wCount = 1; hCount = 1; break;
		  case 2:
			wCount = 1; hCount = 2; break;
		  case 3:
			wCount = 1; hCount = 3; break;
		  case 4:
			wCount = 1; hCount = 4; break;
		  case 5:
		  case 6:
			wCount = 2; hCount = 3; break;
		  case 7:
		  case 8:
			wCount = 2; hCount = 4; break;
		  case 9:
			wCount = 3; hCount = 3; break;
		  case 10:
		  case 11:
		  case 12:
			wCount = 3; hCount = 4; break;
		  case 13:
		  case 14:
		  case 15:
			wCount = 3; hCount = 5; break;
		  case 16:
		  case 17:
		  case 18:
			wCount = 3; hCount = 6; break;
		  case 19:
		  case 20:
			wCount = 4; hCount = 5; break;
		  default:
			wCount = 4; hCount = 6; break;
		}
	}

	wPitch = (ulong)data->mainWinWidth / wCount;
	hPitch = (ulong)(data->mainWinHeight-offset) / hCount;

	for (i=0; i<data->nGraphs; i++) {
		if (data->duplicate[i] != DUPLICATE_ON) win_num++;
		data->winX[i] = (win_num % wCount) * wPitch;
		data->winY[i] = offset + (int)(win_num / wCount) * hPitch;
		data->winWidth[i] = wPitch;
		data->winHeight[i] = hPitch;
	}

	/* タイトルウィンドウのリサイズ */
	data->titleWinX = 0;
	data->titleWinY = 0;
	data->titleWinWidth = data->mainWinWidth;
	data->titleWinHeight = gTitleHeight = offset;

	return 0;
}
```

```
/*
 * ACR PLOT TOOL Version 1.0
 *               Version 2.0
 *    (c) Copyright 1997,1998
 *    ATR Adaptive Communications Research Laboratories
 *    All Rights Reserved
 */

/*
 * graph.cpp -
 */
#include "stdafx.h"
#include "PLOT.h"
#include <stdio.h>
#include <stdarg.h>
#include <math.h>

#include "plotdef.h"
#include "graph.h"

void GrInit()
{
        gGraph = new GrAttr[GR_MAX_GRAPH];

        for (int i=0; i<GR_MAX_GRAPH; i++) {
                gGraph[i].on = 0;
        }
}

/*****************************************************************************
 *        Name: GrOpen
 *    Function: グラフのオープン
 *    Argument:
 *              Data *data    I       DATA構造体
 *              int   no      I       画面番号
 *
 *      Return:
 *              >=0   割り当てたグラフハンドル
 *              < 0   失敗
 * Description:
 *      1 空いているグラフハンドルを割り当てる。
 *      2 割り当てたグラフ情報の初期化を行なう。
 *
 *****************************************************************************/
Graph GrOpen(DATA *data, int no)
{
        int             i;                      /* ループカウンタ */
        GrAttr  *p;                     /* カレントグラフ情報 */

        /* フォントサイズをグラフの個数が１２より大きいなら１／２にする。
                但しPS出力の時はそのまま。 */
        int             nwindows;
        int             mode;
        CPLOTApp *pApp;

        nwindows = data->nWindows;
        mode = data->mode;

        pApp = (CPLOTApp*)AfxGetApp();

        /* 空いているグラフ情報を捜す */
        for (i=0; i<GR_MAX_GRAPH; i++) {
                if (gGraph[i].on == FALSE) break;
        }
        if (i >= GR_MAX_GRAPH) return -1;

        p = gGraph + i;
        p->on                   = TRUE;
        p->x                    = (float)GR_POS_DEFAULT_X;
        p->y                    = (float)GR_POS_DEFAULT_Y;
        p->width                = (float)GR_POS_DEFAULT_WIDTH;
        p->height               = (float)GR_POS_DEFAULT_HEIGHT;
        strcpy(p->line_style, GR_DEFAULT_LINE_STYLE);
        p->line_width           = GR_DEFAULT_LINE_WIDTH;
        p->dot_size             = 1.0;
        p->xdata                = NULL;
        p->xdatatype            = GR_FLOAT;
        p->xrange[0]            = 0;
        p->xrange[1]            = -1;
        p->ydata                = NULL;
        p->ydatatype            = GR_FLOAT;
        p->datacount            = 0;
        p->yrange[0]            = 0;
        p->yrange[1]            = -1;
        /* ラベルデータへのポインタ (数値(f)、または文字(c)) */
        p->labeldata_fp                 = NULL;
        p->labeldata_cp                 = NULL;
        /* ラベルデータ描画用フォント及びGC設定 */
        p->dotlabel_fontname            = strdup(GR_DEFAULT_FONTNAME);
        p->dotlabel_fontbold            = strdup(GR_DEFAULT_FONTBOLD);
        p->dotlabel_fontitalic          = strdup(GR_DEFAULT_FONTITALIC);
        p->dotlabel_fontsize            = GR_DEFAULT_FONTSIZE;
        if (nwindows > 12)      p->dotlabel_fontsize = GR_DEFAULT_FONTSIZE/2;
        p->dotlabel_font.CreateFont(p->dotlabel_fontsize, 0, 0, 0, FW_NORMAL, 0, 0, 0,
                                                        DEFAULT_CHARSET, OUT_DEFAULT_PRECI
S,
                                                        CLIP_DEFAULT_PRECIS, DEFAULT_QUALI
TY,
                                                        DEFAULT_PITCH,
                                                        p->dotlabel_fontname);

        p->dotlabel_color       = NULL;

        /* 線、点描画用設定（初期化） */
        p->dot_color            = NULL;

        /* label distance */
        p->dotlabel_dist                = 0;
        /* duplicate flag */
        p->duplicate            = DUPLICATE_OFF;

        p->box                  = FALSE;
        p->aspect_ratio         = -1;
        p->xaxis                = TRUE;
        p->yaxis                = TRUE;

        p->fontname             = strdup(GR_DEFAULT_FONTNAME);
        p->fontbold             = strdup(GR_DEFAULT_FONTBOLD);
        p->fontitalic           = strdup(GR_DEFAULT_FONTITALIC);
        p->fontsize             = GR_DEFAULT_FONTSIZE;
        if (nwindows > 12) p->fontsize = GR_DEFAULT_FONTSIZE / 2;

        p->xticks               = NULL;
        p->num_xticks           = 0;
        p->yticks               = NULL;
        p->num_yticks           = 0;
        p->xlabel               = NULL;
        p->ylabel               = NULL;
        p->xtick_marks          = NULL;
        p->num_xtick_marks      = 0;
        p->ytick_marks          = NULL;
        p->num_ytick_marks      = 0;
        p->xticks_minor         = NULL;
        p->num_xticks_minor     = 0;
        p->yticks_minor         = NULL;
        p->num_yticks_minor     = 0;
        p->xtick_scale          = NULL;
        p->ytick_scale          = NULL;
        p->decoration_color     = NULL;
        p->decoration_type      = 1;

        /* タイトルフォント */
        p->title_fontname       = strdup(GR_DEFAULT_TITLE_FONTNAME);
        p->title_fontbold       = strdup(GR_DEFAULT_TITLE_FONTBOLD);
        p->title_fontitalic     = strdup(GR_DEFAULT_TITLE_FONTITALIC);
        p->title_fontsize       = GR_DEFAULT_TITLE_FONTSIZE;
```

```
        if (nwindows > 12) p->title_fontsize = GR_DEFAULT_TITLE_FONTSIZE;
        p->title_color      = NULL;

        /* ラベルフォント */
        p->label_fontname   = strdup(GR_DEFAULT_LABEL_FONTNAME);
        p->label_fontbold   = strdup(GR_DEFAULT_LABEL_FONTBOLD);
        p->label_fontitalic = strdup(GR_DEFAULT_LABEL_FONTITALIC);
        p->label_fontsize   = GR_DEFAULT_LABEL_FONTSIZE;
        if (nwindows > 12) p->label_fontsize = GR_DEFAULT_LABEL_FONTSIZE/2;
        p->label_color      = NULL;

        /* 目盛りフォント */
        p->tick_fontname    = strdup(GR_DEFAULT_TICK_FONTNAME);
        p->tick_fontbold    = strdup(GR_DEFAULT_TICK_FONTBOLD);
        p->tick_fontitalic  = strdup(GR_DEFAULT_TICK_FONTITALIC);
        p->tick_fontsize    = GR_DEFAULT_TICK_FONTSIZE;
        if (nwindows > 12) p->tick_fontsize = GR_DEFAULT_TICK_FONTSIZE/2;
        p->tick_color       = NULL;

        /* 目盛り(上付)フォント */
        p->ticksub_fontname = strdup(GR_DEFAULT_TICKSUB_FONTNAME);
        p->ticksub_fontbold = strdup(GR_DEFAULT_TICKSUB_FONTBOLD);
        p->ticksub_fontitalic = strdup(GR_DEFAULT_TICKSUB_FONTITALIC);
        p->ticksub_fontsize = GR_DEFAULT_TICKSUB_FONTSIZE;
        if (nwindows > 12) p->ticksub_fontsize = GR_DEFAULT_TICKSUB_FONTSIZE/2;
        p->ticksub_color    = NULL;

        p->draw_mode        = GR_MODE_NORMAL;
        p->draw_size        = 0;
        p->draw_ptr         = 0;
        p->draw_buffer      = NULL;
        p->hold             = FALSE;
        p->max_exp          = GR_DEFAULT_MAX_EXP;
        p->graph_type       = GR_DEFAULT_GRAPH_TYPE;

        return i;
}

int GrClose(int gr)
{
        GrAttr  *p;

        if (gr<0 || gr>=GR_MAX_GRAPH || gGraph[gr].on)
          return -1;

        p = gGraph + gr;
        p->on = 0;
        Free(p->xdata);
        Free(p->ydata);
        Free(p->fontname);
        Free(p->fontbold);
        Free(p->fontitalic);
        Free(p->title);
        Free(p->xticks);
        Free(p->yticks);
        Free(p->xlabel);
        Free(p->ylabel);
        if (p->xtick_marks) {
                int     i;
                for (i=0; i<p->num_xtick_marks; i++)
                  Free(p->xtick_marks[i]);
                Free(p->xtick_marks);
        }
        if (p->ytick_marks) {
                int     i;
                for (i=0; i<p->num_ytick_marks; i++)
                  Free(p->ytick_marks[i]);
                Free(p->ytick_marks);
        }
        Free(p->xticks_minor);
        Free(p->yticks_minor);
        Free(p->xtick_scale);
        Free(p->ytick_scale);
        Free(p->decoration_color);

        /* タイトルフォントの解放 */
        Free(p->title_fontname);
        Free(p->title_fontbold);
        Free(p->title_fontitalic);
        Free(p->title_color);

        /* ラベルフォントの解放 */
        Free(p->label_fontname);
        Free(p->label_fontbold);
        Free(p->label_fontitalic);
        Free(p->label_color);

        /* 目盛りフォントの解放 */
        Free(p->tick_fontname);
        Free(p->tick_fontbold);
        Free(p->tick_fontitalic);
        Free(p->tick_color);

        /* ラベルデータ描画用フォント、点、線描画用GCの解放 */
        Free(p->dotlabel_fontname);
        Free(p->dotlabel_fontbold);
        Free(p->dotlabel_fontitalic);
        Free(p->dotlabel_color);
        p->dotlabel_font.DeleteObject();

        /* ラベルデータの解放 */
        Free(p->labeldata_fp);
        Free(p->labeldata_cp);

        return 0;
}

/*****************************************************************************
 *        Name: GrPlot
 *    Function: グラフのプロット
 *    Argument:
 *              Graph gr        I       グラフハンドル
 *      Return:
 *               0  成功
 *              -1  失敗
 * Description:
 *              設定されたリソース値にしたがって描画を行なう。
 *              実際にウィンドウに描画されるのは、次の GrDraw 関数で行なう。
 *
 *****************************************************************************/
int GrPlot(Graph gr, Number *x, int nx, Number *y, int ny, Number *lfp, char *lcp)
{
        GrAttr  *p;
        int             i;

        if (gr<0 || gr>=GR_MAX_GRAPH || !gGraph[gr].on)
                return -1;

        p = gGraph + gr;

        delete p->xdata;
        delete p->ydata;

        /* yデータ */
        p->datacount = ny;
        if (y && ny>0) {
                int     len;
                int     rem;

                switch (p->draw_mode) {
                  case GR_MODE_NORMAL:
                        len = sizeof(Number)*ny;
                        p->ydata = new Number[ny];
                        memcpy(p->ydata, y, len);
                        break;
```

```
                case GR_MODE_FLOW:
                        rem = p->datacount - p->draw_size;
                        if (rem > 0) {
                                p->datacount -= rem;
                                y += rem;
                        }
                        rem = p->datacount - (p->draw_size - p->draw_ptr);
                        if (rem < 0) {
                                len = sizeof(Number)*(p->draw_ptr+1-rem);
                                memcpy(p->draw_buffer, p->draw_buffer+rem, len);
                                p->draw_ptr += rem;
                        }
                        len = sizeof(Number)*p->datacount;
                        memcpy(p->draw_buffer+p->draw_ptr, y, len);
                        p->draw_ptr += p->datacount;
                        break;
                }
        }

        /* xデータ */
        if (x && nx>0) {
                int     len = sizeof(Number)*nx;
                p->xdata = new Number[nx];
                memcpy(p->xdata, x, len);
        }
        else {
                p->xdata = new Number[p->datacount];
                for (i=0; i<p->datacount; i++) p->xdata[i] = (float)i+1;
        }

        /* ドット毎のラベルデータへのポインタ */
        if (lfp) p->labeldata_fp = lfp;
        if (lcp) p->labeldata_cp = lcp;

        /* 保持フラグが立っていない場合、値の範囲をクリア */
        if (p->hold == FALSE) {
                grFreeAxisAttributes(gr);
                p->xrange[0] = p->yrange[0] = 0.0;
                p->xrange[1] = p->yrange[1] = -1.0;
        }

        return 0;
}

int
grMakeTickMarks(float *ticks, int num_ticks, char ***marks,
                                int *num_marks, char **tick_scale, int max_exp)
{
        char    **strs;
        int             i, n;
        float   x, x1, x2, ex;
        char    str[40];
        char    *fmt;
        char    *p;

        if (num_ticks <= 0) {
                return 0;
        }

        x1 = (ticks[0] > 0 ? ticks[0]: -ticks[0]);
        x2 = (ticks[num_ticks-1] > 0 ? ticks[num_ticks-1]: -ticks[num_ticks-1]);
        x = (x1 < x2 ? x2: x1);
        ex = 1;
        strs = new char *[num_ticks];

        fmt = "%3.1e";
        sprintf(str, fmt, x);
        p = strrchr(str, 'e');
        if (p) {
                *p++ = '\0';
                n = atoi(p);
                if (max_exp < n || n < -max_exp) {
                        ex = (float)exp10(n);
                        sprintf(str, "10^%d", n);
                        *tick_scale = strdup(str);
                }
        }

        fmt = "%g";
        for (i=0; i<num_ticks; i++) {
                sprintf(str, fmt, ticks[i]/ex);
                strs[i] = strdup(str);
        }
        *num_marks = num_ticks;
        *marks = strs;

        return 0;
}

int grCalcAxisAttributes(int gr)
{
        GrAttr  *p;
        int             nextg = 0;
        int             num_g;          /* 重ね合わせるグラフ数 */
        Number  maxx, minx;
        Number  maxy, miny;
        int             i;
        int             gr0;            /* 元のポイント位置保存 */

        if (gr<0 || gr>=GR_MAX_GRAPH || !gGraph[gr].on) {
                return -1;
        }

        gr0 = gr;
        p = gGraph+gr;

        if (p->xdata && (p->xrange[1]<p->xrange[0])){
                /* 重ねあわせのベースでないときはまず、ベースまで戻る */
                if (p->duplicate == DUPLICATE_ON){
                        gr--;
                        p = gGraph+gr;
                        while (p->duplicate != DUPLICATE_OFF){
                                gr--;
                                p = gGraph+gr;
                        }
                }
                /* ベースのグラフ */
                p = gGraph+gr;
                /* 重ねて描くグラフも含めてX座標の最大最小を求める */
                maxx = -GR_INF;
                minx = GR_INF;
                while(1){
                        if (p->xdata && (p->xrange[1] < p->xrange[0])) {
                                for (i=0; i<p->datacount; i++) {
                                        if (maxx < p->xdata[i])
                                          maxx = p->xdata[i];
                                        if (p->xdata[i] < minx)
                                          minx = p->xdata[i];
                                }
                        }
                        else if (p->xdata && (p->xrange[1]>p->xrange[0])){
                                maxx = p->xrange[1];
                                minx = p->xrange[0];
                        }
                        nextg++;
                        if (gr+nextg < GR_MAX_GRAPH){
                                p = gGraph+gr+nextg;
                        }
                        else{
                                break;
                        }

                        if (p->duplicate != DUPLICATE_ON) break;
                }
                num_g = nextg-1;
                /* 最大最小値が同じ場合 -1から+1とする */
```

```
                if (minx == maxx) {
                        minx -= 1;
                        maxx += 1;
                }
                nextg = 0;
                p = gGraph+gr;
                while (nextg < num_g+1){
                        p = gGraph+gr+nextg;

                        p->xrange[0] = minx;
                        p->xrange[1] = maxx;
                        nextg++;
                }
        }

        nextg = 0;
        p = gGraph+gr0;

        if (p->ydata && (p->yrange[1]<p->yrange[0])){
                /* ベースのグラフ */
                p = gGraph+gr;
                /* yの最大最小値を、重ねて描くグラフも含めて算出 */
                maxy = -GR_INF;
                miny = GR_INF;
                while (nextg < num_g+1){
                        if (p->ydata && (p->yrange[1] < p->yrange[0])) {
                                for (i=0; i<p->datacount; i++) {
                                        if (maxy < p->ydata[i])
                                          maxy = p->ydata[i];
                                        if (p->ydata[i] < miny)
                                          miny = p->ydata[i];
                                }
                        }
                        else if (p->ydata && (p->yrange[1]>p->yrange[0])){
                                maxy = p->yrange[1];
                                miny = p->yrange[0];
                        }
                        nextg++;
                        p = gGraph+gr+nextg;
                }
                /* 最大最小値が同じ場合 -1から+1とする */
                if (miny == maxy) {
                        miny -= 1;
                        maxy += 1;
                }
                nextg = 0;
                p = gGraph + gr;
                while (nextg < num_g+1){
                        p = gGraph+gr+nextg;
                        p->yrange[0] = miny;
                        p->yrange[1] = maxy;
                        nextg++;
                }
        }

        p = gGraph+gr0;
        /* x目盛りの計算 */
        if (p->num_xticks != -1) {
                grMakeNormalTicks(p->xrange, &(p->xticks), &(p->num_xticks),
                                                &(p->xticks_minor), &(p->num_xticks_mino
r));
                grMakeTickMarks(p->xticks, p->num_xticks,
                                                &(p->xtick_marks), &(p->num_xtick_marks),
                                                &(p->xtick_scale),
                                                p->max_exp);
        }

        /* y目盛りの計算 */
        if (p->num_yticks != -1) {
                grMakeNormalTicks(p->yrange, &(p->yticks), &(p->num_yticks),
                                                &(p->yticks_minor), &(p->num_yticks_mino
r));
                grMakeTickMarks(p->yticks, p->num_yticks,
                                                &(p->ytick_marks), &(p->num_ytick_marks),
                                                &(p->ytick_scale),
                                                p->max_exp);
        }

        return 0;
}

int grFreeAxisAttributes(int gr)
{
        GrAttr  *p;

        if (gr<0 || gr>=GR_MAX_GRAPH || !gGraph[gr].on) {
                return -1;
        }

        p = gGraph+gr;
        /* x目盛りの開放 */
        delete[] p->xticks;
        p->num_xticks = 0;

        /* y目盛りの開放 */
        delete[] p->yticks;
        p->num_yticks = 0;

        /* x目盛り文字列の開放 */
        if (p->xtick_marks) {
                for (int i=0; i<p->num_xtick_marks; i++)
                  delete[] p->xtick_marks[i];
                Free(p->xtick_marks);
        }
        p->num_xtick_marks = 0;

        /* y目盛り文字列の開放 */
        if (p->ytick_marks) {
                for (int i=0; i<p->num_ytick_marks; i++)
                  delete[] p->ytick_marks[i];
                Free(p->ytick_marks);
        }
        p->num_ytick_marks = 0;

        /* x小目盛りの開放 */
        delete[] p->xticks_minor;
        p->num_xticks_minor = 0;

        /* y小目盛りの開放 */
        delete[] p->yticks_minor;
        p->num_yticks_minor = 0;

        /* x,y範囲の初期化 */
        p->xrange[0] = p->yrange[0] = 0;
        p->xrange[1] = p->yrange[1] = -1;

        return 0;
}

int
grMakeNormalTicks(float range[2], float **ticks, int *n,
                                        float **Minor, int *m)
{
        float   w;
        float   tks[GR_TICK_MAX_ALL_TICKS];
        int             i, len;
        int             DIV, N, A, M;
        float   T;
        int             div, c;
        float   t;

        if (!range || range[0]>=range[1]) {
                *ticks = NULL;
                *n = 0;
                *Minor = NULL;
                *m = 0;
```

```
                return -1;
        }

        /*
         * 値の範囲をGR_TICK_NUM_TICKSで割った商の上１桁の数Aと桁数Nを求める。
         *      A == 1 -> M = 1, c=2
         *      A == 2 -> M = 2, c=2
         *      A <= 5 -> M = 5, c=5
         *      A <= 9 -> M = 10,c=2
         * 上記のMとNより、目盛り間隔Tは
         *      T = M*10^(N-1)
         * で算出する。
         */
        w = (range[1]-range[0])/GR_TICK_NUM_TICKS;
        N = (int)floor(log10(w));
        A = (int)ceil((double)(w/exp10((double)N)));
        if      (A == 1) M = 1, c = 2;
        else if (A == 2) M = 2, c = 2;
        else if (A <= 5) M = 5, c = 5;
        else             M = 10,c = 2;
        T = (float)(M*exp10((double)N));
        DIV = (int)ceil(range[0]/T);

        for (i=0; i<GR_TICK_MAX_TICKS; i++) {
                if (range[1] < T*(DIV+i)) break;
                tks[i] = T*(DIV+i);
        }
        *n = i;
        len = sizeof(float)*(*n);
        try {
                *ticks = new float[*n];
        }
        catch (CMemoryException* e) {
                e->Delete();
                return -1;
        }
        memcpy(*ticks, tks, len);

        /* Minor目盛りの算出 */
        t = T/c;
        div = (int)ceil(range[0]/t);
        for (i=0; i<GR_TICK_MAX_ALL_TICKS; i++) {
                if (range[1] < t*(div+i)) break;
                tks[i] = t*(div+i);
        }
        *m = i;
        len = sizeof(float)*(*m);
        try {
                *Minor = new float[*m];
        }
        catch (CMemoryException* e) {
                e->Delete();
                return -1;
        }
        memcpy(*Minor, tks, len);

        return 0;
}

int grCalcAxisOrigin(int gr)
{
        GrAttr  *p;

        if (gr<0 || gr>=GR_MAX_GRAPH || !gGraph[gr].on)
          return -1;

        /* ウィンドウ領域の取得 */
        p = gGraph+gr;

        /* メータ型のグラフ */
        if (p->graph_type == GR_TYPE_METER) {
                /* 描画領域の算出 */
                p->Ax = 0;
                p->Ay = 0;
                p->Aw = p->Ww;
                p->Ah = p->Wh;

                /* グラフ領域の算出 */
                p->Rx = (int)(p->Aw*p->x);
                p->Ry = (int)(p->Ah*p->y);
                p->Rw = (int)(p->Aw*p->width);
                p->Rh = (int)(p->Ah*p->height);

                /* プロット領域の算出 */
                p->Ox = (int)(p->Rx+p->Aw*GR_POS_METER_WIDTH_MARGIN/2);
                p->Oy = (int)(p->Ry+p->Ah*GR_POS_METER_HEIGHT_MARGIN);
                p->Ow = (int)(p->Rw-p->Aw*GR_POS_METER_WIDTH_MARGIN);
                p->Oh = (int)(p->Rh-p->Ah*GR_POS_METER_HEIGHT_MARGIN);
        }

        /* その他通常のグラフ */
        else {
                /* 描画領域の算出 */
                p->Ax = (int)(p->x*p->Ww);
                p->Ay = (int)(p->y*p->Wh);
                p->Aw = (int)(p->width*p->Ww);
                p->Ah = (int)(p->height*p->Wh);

                /* グラフ領域の算出 */
                p->Rx = p->Ax + GR_POS_GRAPH_AREA_LEFT_MARGIN;
                p->Ry = p->Ay + GR_POS_GRAPH_AREA_TOP_MARGIN;
                p->Rw = p->Aw - (GR_POS_GRAPH_AREA_LEFT_MARGIN + GR_POS_GRAPH_AREA_RIGHT_M
ARGIN);
                p->Rh = p->Ah - (GR_POS_GRAPH_AREA_TOP_MARGIN + GR_POS_GRAPH_AREA_BOTTOM_M
ARGIN);

                /* プロット領域の算出 */
                p->Ox = p->Rx + GR_POS_PLOT_AREA_LEFT_MARGIN;
                p->Oy = p->Ry + GR_POS_PLOT_AREA_TOP_MARGIN;
                p->Ow = p->Rw - (GR_POS_PLOT_AREA_LEFT_MARGIN + GR_POS_PLOT_AREA_RIGHT_MAR
GIN);
                p->Oh = p->Rh - (GR_POS_PLOT_AREA_TOP_MARGIN + GR_POS_PLOT_AREA_BOTTOM_MAR
GIN);

                /* 縦横比が指定されていた場合 */
                if (p->aspect_ratio > 0) {
                        float width, height, margin, pitch;

                        width  = (float)p->Oh * p->aspect_ratio;
                        height = (float)p->Ow / p->aspect_ratio;

                        /* 幅を変更する場合 */
                        if (width < p->Ow) {
                                pitch = (float)p->Ow - width;
                                margin = (float)(pitch/2.0f);
                                p->Rx += (int)margin;
                                p->Rw -= (int)pitch;
                                p->Ax += (int)margin;
                                p->Aw -= (int)pitch;
                                p->Ox += (int)margin;
                                p->Ow -= (int)pitch;
                        }
                        /* 高さを変更する場合 */
                        else if (height < p->Oh) {
                                pitch = (float)p->Oh - height;
                                margin = (float)(pitch/2.0f);
                                p->Ry += (int)margin;
                                p->Rh -= (int)pitch;
                                p->Ay += (int)margin;
                                p->Ah -= (int)pitch;
                                p->Oy += (int)margin;
                                p->Oh -= (int)pitch;
                        }
                }
        }
```

```
        p->Ox += (int)p->absx;
        p->Oy += (int)(p->absy + gTitleHeight);
        p->Ax += (int)p->absx;
        p->Ay += (int)(p->absy + gTitleHeight);
        p->Rx += (int)p->absx;
        p->Ry += (int)(p->absy + gTitleHeight);

        return 0;
}

int GrSet(Graph gr, ...)
{
        GrAttr  *p;
        va_list ap;
        int             prop;
        float   *xdata, *ydata;
        float   *xrange, *yrange;
        float   *f;
        int             isfont = FALSE;
        int             istitle_font = FALSE;
        int             islabel_font = FALSE;
        int     istick_font = FALSE;
        int     isticksub_font = FALSE;
        int             is_draw_mode = FALSE;
        int             is_xrange = FALSE;
        int             is_yrange = FALSE;
        char    *str;
        char    *name;

        if (gr<0 || gr>=GR_MAX_GRAPH || !gGraph[gr].on) {
                return -1;
        }

        p = gGraph+gr;
        va_start(ap, gr);
        xdata = ydata = NULL;
        while (1) {
                prop = va_arg(ap, int);
                switch (prop) {
                  case NULL:
                        goto next;
                        break;
                  case GR_X:
                        p->x = va_arg(ap, float);
                        break;
                  case GR_Y:
                        p->y = va_arg(ap, float);
                        break;
                  case GR_WIDTH:
                        p->width = va_arg(ap, float);
                        break;
                  case GR_HEIGHT:
                        p->height = va_arg(ap, float);
                        break;
                  case GR_LINE_STYLE:
                        strcpy(p->line_style, va_arg(ap, char *));
                        break;
                  case GR_LINE_WIDTH:
                        p->line_width = va_arg(ap, float);
                        break;
                  case GR_DOT_SIZE:
                        f = va_arg(ap, float *);
                        if (*f > 0)     p->dot_size = *f;
                        break;
                  case GR_XDATA:
                        xdata = va_arg(ap, float *);
                        break;
                  case GR_XDATATYPE:
                        p->xdatatype = va_arg(ap, int);
                        break;
                  case GR_YDATA:
                        ydata = va_arg(ap, float *);
                        break;
```

```
                  case GR_YDATATYPE:
                        p->ydatatype = va_arg(ap, int);
                        break;
                  case GR_DATACOUNT:
                        p->datacount = va_arg(ap, int);
                        break;
                  case GR_TITLE:
                        p->title = strdup(va_arg(ap, char *));
                        break;
                  case GR_BOX:
                        p->box = va_arg(ap, int);
                        break;
                  case GR_ASPECT_RATIO:
                        f = va_arg(ap, float *);
                        p->aspect_ratio = *f;
                        break;
                  case GR_XTICKS:
                        p->xticks = va_arg(ap, float *);
                        break;
                  case GR_NUM_XTICKS:
                        p->num_xticks = va_arg(ap, int);
                        break;
                  case GR_YTICKS:
                        p->yticks = va_arg(ap, float *);
                        break;
                  case GR_NUM_YTICKS:
                        p->num_yticks = va_arg(ap, int);
                        break;
                  case GR_XLABEL:
                        p->xlabel = strdup(va_arg(ap, char *));
                        break;
                  case GR_YLABEL:
                        p->ylabel = strdup(va_arg(ap, char *));
                        break;
                  case GR_DECORATION_COLOR:
                        str = va_arg(ap, char *);
                        if (str && *str) {
                                p->decoration_color = strdup(str);
#if 0
                                if (gPSOnlyMode == False)
                                  grAllocHighLowColor(p->disp,
                                                                        p->decoration_co
lor,
                                                                        p->decoration_pi
xel);
#endif
                        }
                        break;
                  case GR_XRANGE:
                        xrange = va_arg(ap, float *);
                        is_xrange = TRUE;
                        break;
                  case GR_YRANGE:
                        yrange = va_arg(ap, float *);
                        is_yrange = TRUE;
                        break;
                  case GR_FONTNAME:
                        p->fontname = strdup(va_arg(ap, char *));
                        isfont = TRUE;
                        break;
                  case GR_FONTBOLD:
                        p->fontbold = strdup(va_arg(ap, char *));
                        isfont = TRUE;
                        break;
                  case GR_FONTITALIC:
                        p->fontitalic = strdup(va_arg(ap, char *));
                        isfont = TRUE;
                        break;
                  case GR_FONTSIZE:
                        p->fontsize = va_arg(ap, int);
                        isfont = TRUE;
                        break;
                  case GR_TITLE_FONTNAME:
```

```
                        p->title_fontname = strdup(va_arg(ap, char *));
                        istitle_font = TRUE;
                        break;
                case GR_TITLE_FONTBOLD:
                        p->title_fontbold = strdup(va_arg(ap, char *));
                        istitle_font = TRUE;
                        break;
                case GR_TITLE_FONTITALIC:
                        p->title_fontitalic = strdup(va_arg(ap, char *));
                        istitle_font = TRUE;
                        break;
                case GR_TITLE_FONTSIZE:
                        p->title_fontsize = va_arg(ap, int);
                        istitle_font = TRUE;
                        break;
                case GR_TITLE_COLOR:
#if 0
                        {
                                Colormap        cmap;
                                char            *name;
                                XColor          ci, cr;

                                if (gPSOnlyMode == False)
                                  cmap = DefaultColormap(p->disp, 0);
                                name = strdup(va_arg(ap, char *));
                                if (p->title_color) {
                                        Free(p->title_color);
                                        if (gPSOnlyMode == False)
                                          XFreeColors(p->disp, cmap, &p->title_pixel, 1, 0
);
                                }
                                if (!*name) {
                                        Free(name);
                                        p->title_color = NULL;
                                        if (gPSOnlyMode == False)
                                          p->title_pixel = BlackPixel(p->disp, 0);
                                        break;
                                }
                                p->title_color = name;
                                if (gPSOnlyMode == False) {
                                        XAllocNamedColor(p->disp, cmap, p->title_color, &c
i, &cr);
                                        p->title_pixel = cr.pixel;
                                        XSetForeground(p->disp, p->title_gc, p->title_pixe
l);
                                }
                        }
#else
                        name = strdup(va_arg(ap, char *));
                        if (p->title_color) Free(p->title_color);
                        if (!*name) {
                                Free(name);
                                p->title_color = NULL;
                        }
                        else p->title_color = name;
#endif
                        break;
                case GR_LABEL_FONTNAME:
                        p->label_fontname = strdup(va_arg(ap, char *));
                        islabel_font = TRUE;
                        break;
                case GR_LABEL_FONTBOLD:
                        p->label_fontbold = strdup(va_arg(ap, char *));
                        islabel_font = TRUE;
                        break;
                case GR_LABEL_FONTITALIC:
                        p->label_fontitalic = strdup(va_arg(ap, char *));
                        islabel_font = TRUE;
                        break;
                case GR_LABEL_FONTSIZE:
                        p->label_fontsize = va_arg(ap, int);
                        islabel_font = TRUE;
                        break;
```

```
                case GR_LABEL_COLOR:
#if 0
                        {
                                Colormap        cmap;
                                char            *name;
                                XColor          ci, cr;

                                if (gPSOnlyMode == False)
                                  cmap = DefaultColormap(p->disp, 0);
                                name = strdup(va_arg(ap, char *));
                                if (p->label_color) {
                                        Free(p->label_color);
                                        if (gPSOnlyMode == False)
                                          XFreeColors(p->disp, cmap, &p->label_pixel, 1, 0
);
                                }
                                if (!*name) {
                                        Free(name);
                                        p->label_color = NULL;
                                        if (gPSOnlyMode == False)
                                          p->label_pixel = BlackPixel(p->disp, 0);
                                        break;
                                }
                                p->label_color = name;
                                if (gPSOnlyMode == False) {
                                        XAllocNamedColor(p->disp, cmap, p->label_color, &c
i, &cr);
                                        p->label_pixel = cr.pixel;
                                        XSetForeground(p->disp, p->label_gc, p->label_pixe
l);
                                }
                        }
#else
                        name = strdup(va_arg(ap, char *));
                        if (p->label_color) Free(p->label_color);
                        if (!*name) {
                                Free(name);
                                p->label_color = NULL;
                        }
                        else p->label_color = name;
#endif
                        break;
                case GR_TICK_FONTNAME:
                        p->tick_fontname = strdup(va_arg(ap, char *));
                        istick_font = TRUE;
                        break;
                case GR_TICK_FONTBOLD:
                        p->tick_fontbold = strdup(va_arg(ap, char *));
                        istick_font = TRUE;
                        break;
                case GR_TICK_FONTITALIC:
                        p->tick_fontitalic = strdup(va_arg(ap, char *));
                        istick_font = TRUE;
                        break;
                case GR_TICK_FONTSIZE:
                        p->tick_fontsize = va_arg(ap, int);
                        istick_font = TRUE;
                        break;
                case GR_TICK_COLOR:
#if 0
                        {
                                Colormap        cmap;
                                char            *name;
                                XColor          ci, cr;

                                if (gPSOnlyMode == False)
                                  cmap = DefaultColormap(p->disp, 0);
                                name = strdup(va_arg(ap, char *));
                                if (p->tick_color) {
                                        Free(p->tick_color);
                                        if (gPSOnlyMode == False)
                                          XFreeColors(p->disp, cmap, &p->tick_pixel, 1, 0)
;
```

```
                        }
                        if (!*name) {
                                Free(name);
                                p->tick_color = NULL;
                                if (gPSOnlyMode == False)
                                  p->tick_pixel = BlackPixel(p->disp, 0);
                                break;
                        }
                        p->tick_color = name;
                        if (gPSOnlyMode == False) {
                                XAllocNamedColor(p->disp, cmap, p->tick_color, &ci
, &cr);
                                p->tick_pixel = cr.pixel;
                                XSetForeground(p->disp, p->tick_gc, p->tick_pixel)
;
                        }
                }
#else
                name = strdup(va_arg(ap, char *));
                if (p->tick_color) {
                        Free(p->tick_color);
                }
                if (!*name) {
                        Free(name);
                        p->tick_color = NULL;
                }
                else p->tick_color = name;
#endif
                break;
        case GR_TICKSUB_FONTNAME:
                p->ticksub_fontname = strdup(va_arg(ap, char *));
                isticksub_font = TRUE;
                break;
        case GR_TICKSUB_FONTBOLD:
                p->ticksub_fontbold = strdup(va_arg(ap, char *));
                isticksub_font = TRUE;
                break;
        case GR_TICKSUB_FONTITALIC:
                p->ticksub_fontitalic = strdup(va_arg(ap, char *));
                isticksub_font = TRUE;
                break;
        case GR_TICKSUB_FONTSIZE:
                p->ticksub_fontsize = va_arg(ap, int);
                isticksub_font = TRUE;
                break;
        case GR_TICKSUB_COLOR:
#if 0
                {
                        Colormap        cmap;
                        char            *name;
                        XColor          ci, cr;

                        if (gPSOnlyMode == False)
                          cmap = DefaultColormap(p->disp, 0);
                        name = strdup(va_arg(ap, char *));
                        if (p->ticksub_color) {
                                Free(p->ticksub_color);
                                if (gPSOnlyMode == False)
                                  XFreeColors(p->disp, cmap, &p->ticksub_pixel, 1,
 0);
                        }
                        if (!*name) {
                                Free(name);
                                p->ticksub_color = NULL;
                                if (gPSOnlyMode == False)
                                  p->ticksub_pixel = BlackPixel(p->disp, 0);
                                break;
                        }
                        p->ticksub_color = name;
                        if (gPSOnlyMode == False) {
                                XAllocNamedColor(p->disp, cmap, p->ticksub_color,
&ci, &cr);
                                p->ticksub_pixel = cr.pixel;
                                XSetForeground(p->disp, p->ticksub_gc, p->ticksub_
pixel);
                        }
                }
#else
                name = strdup(va_arg(ap, char *));
                if (p->ticksub_color) {
                        Free(p->ticksub_color);
                }
                if (!*name) {
                        Free(name);
                        p->ticksub_color = NULL;
                }
                else p->ticksub_color = name;
#endif
                break;
        case GR_DRAW_MODE:
                p->draw_mode = va_arg(ap, int);
                is_draw_mode = TRUE;
                break;
        case GR_DRAW_SIZE:
                int     size;
                size = va_arg(ap, int);
                if (size >= 0) {
                        if (p->draw_buffer) Free(p->draw_buffer);
                        p->draw_buffer = new Number[size];
                        p->draw_size = size;
                        p->draw_ptr = 0;
                }
                break;
        case GR_HOLD:
                p->hold = va_arg(ap, int);
                break;
        case GR_MAX_EXP:
                p->max_exp = va_arg(ap, int);
                break;
        case GR_GRAPH_TYPE:
                p->graph_type = va_arg(ap, int);
                break;
        case GR_DUPLICATE:
                /* グラフの重ね合わせフラグの設定 */
                p->duplicate = va_arg( ap, int );
                break;
        case GR_DOTCOLOR:
                char            *colorname;
                char            *subcolname;

                colorname = strdup(va_arg(ap, char *));
                if (colorname[0] == '#'){
                        p->dot_color = strdup(colorname);
                }
                else if ((subcolname = strchr(colorname, ':')) != NULL){
                        p->dot_color = strdup(subcolname);
                        p->dot_color[0] = '#';
                }
                else {
                        p->dot_color = strdup("#000000");
                }
                break;
        case GR_DOTLABEL_SIZE:
                /* 点ラベルのフォントサイズの設定 */
                p->dotlabel_fontsize = va_arg(ap, int);
                if (gPSOnlyMode==FALSE && p->dotlabel_fontsize>0) {
                        p->dotlabel_font.CreateFont(p->dotlabel_fontsize,
                                                    0, 0, 0, FW_NORMAL, 0, 0, 0,
                                                    DEFAULT_CHARSET, OUT_DEFAULT_PRECI
S,
                                                    CLIP_DEFAULT_PRECIS, DEFAULT_QUALI
TY,
                                                    DEFAULT_PITCH,
                                                    p->dotlabel_fontname);
                }
                break;
```

```
                case GR_DOTLABEL_DIST:
                        /* ラベルをてんからどれだけはなして打つか設定 */
                        int dist;
                        dist = va_arg(ap, int);
                        if (dist > -1) p->dotlabel_dist = dist;
                        break;
                case default:
                        break;
                }
        }

  next:
        va_end(ap);

        /* データ変更 */
        if (xdata) {
                int     len = sizeof(Number)*p->datacount;
                Free(p->xdata);
                p->xdata = new Number[p->datacount];
                memcpy(p->xdata, xdata, len);
        }
        if (ydata) {
                int     len;
                int     rem;
                if (p->ydata) Free(p->ydata);

                switch (p->draw_mode) {
                  case GR_MODE_NORMAL:
                        len = sizeof(Number)*p->datacount;
                        p->ydata = new Number[p->datacount];
                        memcpy(p->ydata, ydata, len);
                        break;
                  case GR_MODE_FLOW:
                        rem = p->datacount - p->draw_size;
                        if (rem > 0) {
                                p->datacount -= rem;
                                ydata += rem;
                        }
                        rem = p->datacount - (p->draw_size - p->draw_ptr);
                        if (rem < 0) {
                                len = sizeof(Number)*(p->draw_ptr+1-rem);
                                memcpy(p->draw_buffer, p->draw_buffer+rem, len);
                                p->draw_ptr += rem;
                        }
                        len = sizeof(Number)*p->datacount;
                        memcpy(p->draw_buffer+p->draw_ptr, ydata, len);
                        p->draw_ptr += p->datacount;
                        break;
                }
                if (p->hold == FALSE) {
                        grFreeAxisAttributes(gr);
                }
        }

#if 0
        /* フォント変更 */
        if (gPSOnlyMode == False && isfont == TRUE) {
                if (p->font)
                  XFreeFont(p->disp, p->font);
                if (grLoadFontByType(p->disp, p->fontname,
                                                        p->fontbold, p->fontsize, p->font
italic,
                                                        &p->font)) {
                        fprintf(stderr, "Error: Can't load font\n");
                        return -1;
                }
                XSetFont(p->disp, p->foregc, p->font->fid);
        }
        /* タイトルフォント変更 */
        if (gPSOnlyMode == False && istitle_font == TRUE) {
                XFreeFont(p->disp, p->title_font);
                if (grLoadFontByType(p->disp, p->title_fontname,
                                                        p->title_fontbold,
                                                        p->title_fontsize,
                                                        p->title_fontitalic,
                                                        &p->title_font)) {
                        fprintf(stderr, "Error: Can't load font\n");
                        return -1;
                }
                XSetFont(p->disp, p->title_gc, p->title_font->fid);
        }
        /* ラベルフォント変更 */
        if (gPSOnlyMode == False && islabel_font == TRUE) {
                if (p->label_font)
                  XFreeFont(p->disp, p->label_font);
                if (grLoadFontByType(p->disp, p->label_fontname,
                                                        p->label_fontbold,
                                                        p->label_fontsize,
                                                        p->label_fontitalic,
                                                        &p->label_font)) {
                        fprintf(stderr, "Error: Can't load font\n");
                        return -1;
                }
                XSetFont(p->disp, p->label_gc, p->label_font->fid);
        }
        /* 目盛りフォント変更 */
        if (gPSOnlyMode == False && istick_font == TRUE) {
                if (p->tick_font) XFreeFont(p->disp, p->tick_font);
                if (grLoadFontByType(p->disp, p->tick_fontname,
                                                        p->tick_fontbold,
                                                        p->tick_fontsize,
                                                        p->tick_fontitalic,
                                                        &p->tick_font)) {
                        fprintf(stderr, "Error: Can't load font\n");
                        return -1;
                }
                XSetFont(p->disp, p->tick_gc, p->tick_font->fid);
        }
        /* 目盛り(上付)フォント変更 */
        if (gPSOnlyMode == False && isticksub_font == TRUE) {
                if (p->ticksub_font) XFreeFont(p->disp, p->ticksub_font);
                if (grLoadFontByType(p->disp, p->ticksub_fontname,
                                                        p->ticksub_fontbold,
                                                        p->ticksub_fontsize,
                                                        p->ticksub_fontitalic,
                                                        &p->ticksub_font)) {
                        fprintf(stderr, "Error: Can't load font\n");
                        return -1;
                }
                XSetFont(p->disp, p->ticksub_gc, p->ticksub_font->fid);
        }
#endif

        /* 描画モード変更 */
        if (is_draw_mode == TRUE) {
        }

        if (is_xrange == TRUE) {
                p->xrange[0] = xrange[0];
                p->xrange[1] = xrange[1];
        }
        if (is_yrange == TRUE) {
                p->yrange[0] = yrange[0];
                p->yrange[1] = yrange[1];
        }

        return 0;
}
```

```
/*
 * ACR PLOT TOOL Version 1.0
 *    (c) Copyright 1997
 *    ATR Adaptive Communications Research Laboratories
 *    All Rights Reserved
 */


/*
 * graph.h - グラフ情報の定義ファイル
 */

#ifndef __graph_h__
#define __graph_h__

#define Number  float
#define Graph   int             /* グラフハンドル */

#define GR_MAX_GRAPH            100
#define GR_INF                  1000000
#define GR_EPS                  1.0e-6

#define GR_INT                  1
#define GR_FLOAT                2
#define GR_CHAR                 3
#define GR_STRING               4
#define GR_ADDR                 5
#define GR_DOUBLE               6
#define GR_SHORT                7

#define GR_TYPE_XY              1
#define GR_TYPE_METER       2

#define GR_X                    1
#define GR_Y                    2
#define GR_WIDTH                3
#define GR_HEIGHT               4
#define GR_XDATA                5
#define GR_XDATATYPE        6
#define GR_YDATA                7
#define GR_DATACOUNT            8
#define GR_YDATATYPE            9
#define GR_TITLE                        10
#define GR_WINDOW                       11
#define GR_BOX                          12
#define GR_XTICKS                       13
#define GR_NUM_XTICKS           14
#define GR_YTICKS                       15
#define GR_NUM_YTICKS           16
#define GR_XAXIS                        17
#define GR_YAXIS                        18
#define GR_XLABEL                       19
#define GR_YLABEL                       20
#define GR_DECORATION_COLOR     21
#define GR_XRANGE                       22
#define GR_YRANGE                       23
#define GR_FONTNAME                     24
#define GR_FONTBOLD                     25
#define GR_FONTITALIC           26
#define GR_FONTSIZE                     27
#define GR_TITLE_FONTNAME               28
#define GR_TITLE_FONTBOLD               29
#define GR_TITLE_FONTITALIC             30
#define GR_TITLE_FONTSIZE               31
#define GR_TITLE_COLOR                  32
#define GR_LABEL_FONTNAME               33
#define GR_LABEL_FONTBOLD               34
#define GR_LABEL_FONTITALIC             35
#define GR_LABEL_FONTSIZE               36
#define GR_LABEL_COLOR                  37
#define GR_DRAW_MODE                    38
#define GR_DRAW_SIZE                    39
#define GR_HOLD                                 40
```



```
#define GR_MAX_EXP                              41
#define GR_GRAPH_TYPE                   42
#define GR_DECORATION_TYPE              43
#define GR_LINE_STYLE           44
#define GR_TICK_FONTNAME                45  /* 目盛りフォント名 */
#define GR_TICK_FONTBOLD                46  /* 目盛りフォント太さ */
#define GR_TICK_FONTITALIC              47  /* 目盛りフォント傾斜 */
#define GR_TICK_FONTSIZE                48  /* 目盛りフォントサイズ */
#define GR_TICK_COLOR                   49  /* 目盛りフォント色 */
#define GR_TICKSUB_FONTNAME             50  /* 目盛りフォント名 */
#define GR_TICKSUB_FONTBOLD             51  /* 目盛りフォント太さ */
#define GR_TICKSUB_FONTITALIC   52  /* 目盛りフォント傾斜 */
#define GR_TICKSUB_FONTSIZE             53  /* 目盛りフォントサイズ */
#define GR_TICKSUB_COLOR                54  /* 目盛りフォント色 */
#define GR_LINE_WIDTH           55
#define GR_ASPECT_RATIO         56
#define GR_DOT_SIZE             57
#define GR_DUPLICATE                    58
#define GR_DOTCOLOR                             59
#define GR_DOTLABEL_SIZE                60
#define GR_DOTLABEL_DIST                61

#define GR_LINE_STYLE_LINE              "-"
#define GR_LINE_STYLE_DOT               "."
#define GR_LINE_STYLE_DOTTEDLINE        "-."
#define GR_LINE_STYLE_DASH              "--"
#define GR_LINE_STYLE_ROUND             "o"
#define GR_LINE_STYLE_X                 "x"
#define GR_LINE_STYLE_DOTTEDLINE_LONG   "-.-."
#define GR_LINE_STYLE_DASH_LONG         "----"
/*
 * グラフ情報宣言
 */
typedef struct {
    int         on;                     /* 有効フラグ */
    float       x;                      /* 左上相対座標 */
    float       y;                      /* 左上相対座標 */
    float       width;                  /* 相対幅 */
    float       height;                 /* 相対高さ */
    float       absx;                   /* 左上絶対座標 */
    float       absy;                   /* 左上絶対座標 */
    char    line_style[3];          /* 線種 */
    float   line_width;             /* 線幅 */
    float   dot_size;
    Number      *xdata;                 /* xデータ */
    int         xdatatype;              /*  */
    Number      xrange[2];              /* xデータ範囲 */
    Number      *ydata;                 /* yデータ配列 */
    int     ydatatype;              /*  */
    int         datacount;              /* データ数 */
    float       yrange[2];              /* yデータ範囲 */
    char        *labeldata_cp;          /* ラベルデータ配列
                                           (ラベル列をラベルとするとき) */
    float       *labeldata_fp;          /* ラベルデータ配列
                                           (データ列をラベルとするとき) */
    char        *dotlabel_fontname;     /* ラベルデータフォント名 */
    char        *dotlabel_fontbold;     /* ラベルデータフォント太さ */
    char        *dotlabel_fontitalic;   /* ラベルデータフォント傾斜 */
    int         dotlabel_fontsize;      /* ラベルデータフォントサイズ */
    CFont   dotlabel_font;          /* ラベルデータフォントクラス */
    char        *dotlabel_color;        /* ラベルデータの色 */
    unsigned long dotlabel_pixel;       /* ラベルデータのピクセル値 */
    char        *dot_color;             /* 線、点の色 */
    unsigned long dot_pixel;            /* 線、点のピクセル値 */
    int         dotlabel_dist;          /* ラベルデータ位置 */
    int         duplicate;              /* グラフを重ねるか示すフラグ */
    char        *title;                 /* タイトル文字列 */
    int         box;                    /* ボックスフラグ */
    float   aspect_ratio;           /* 縦横比 */
    char        *fontname;              /* フォント名 */
    char        *fontbold;              /* フォント太さ */
    char        *fontitalic;            /* フォント傾き */
    int         fontsize;               /* フォントサイズ */
```

```
    float       *xticks;                    /* x目盛り配列 */
    int         num_xticks;                 /* x目盛り配列数 */
    float       *yticks;                    /* y目盛り配列 */
    int         num_yticks;                 /* y目盛り配列数 */
    int         xaxis;                      /*  */
    int         yaxis;                      /*  */
    char        *xlabel;                    /* xラベル */
    char        *ylabel;                    /* yラベル */
    char        **xtick_marks;              /* x目盛り文字列 */
    int         num_xtick_marks;            /* x目盛り文字列数 */
    char        **ytick_marks;              /* y目盛り文字列 */
    int         num_ytick_marks;            /* y目盛り文字列数 */
    float       *xticks_minor;              /* x Minor目盛り配列 */
    int         num_xticks_minor;           /* x Minor目盛り配列数 */
    float       *yticks_minor;              /* y Minor目盛り配列 */
    int         num_yticks_minor;           /* y Minor目盛り配列数 */
    char    *xtick_scale;               /* x スケール文字列 */
    char    *ytick_scale;               /* y スケール文字列 */
    int         decoration_type;            /* 装飾タイプ */
    char        *decoration_color;          /* 装飾色 */
    unsigned long decoration_pixel[3];      /* 装飾ピクセル値 */
    char        *title_fontname;            /* タイトルフォント名 */
    char        *title_fontbold;            /* タイトルフォント太さ */
    char        *title_fontitalic;          /* タイトルフォント傾斜 */
    int         title_fontsize;             /* タイトルフォントサイズ */
    char        *title_color;               /* タイトルフォント色 */
    unsigned long title_pixel;              /* タイトルフォントピクセル値 */
    char        *label_fontname;            /* ラベルフォント名 */
    char        *label_fontbold;            /* ラベルフォント太さ */
    char        *label_fontitalic;          /* ラベルフォント傾斜 */
    int         label_fontsize;             /* ラベルフォントサイズ */
    char        *label_color;               /* ラベルフォント色 */
    unsigned long label_pixel;              /* ラベルフォントピクセル値 */
    char        *tick_fontname;             /* 目盛りフォント名 */
    char        *tick_fontbold;             /* 目盛りフォント太さ */
    char        *tick_fontitalic;           /* 目盛りフォント傾斜 */
    int         tick_fontsize;              /* 目盛りフォントサイズ */
    char        *tick_color;                /* 目盛りフォント色 */
    unsigned long tick_pixel;               /* 目盛りフォントピクセル値 */
    char        *ticksub_fontname;          /* 目盛り(上付)フォント名 */
    char        *ticksub_fontbold;          /* 目盛り(上付)フォント太さ */
    char        *ticksub_fontitalic;        /* 目盛り(上付)フォント傾斜 */
    int         ticksub_fontsize;           /* 目盛り(上付)フォントサイズ */
    char        *ticksub_color;             /* 目盛り(上付)フォント色 */
    unsigned long ticksub_pixel;            /* 目盛り(上付)フォントピクセル値 */
    int         draw_mode;                  /* 描画モード */
    int         draw_size;                  /* 描画サイズ */
    float       *draw_buffer;               /* 描画バッファ */
    int         draw_ptr;                   /* 描画ポインタ */
    int         hold;                       /* 設定値の保持フラグ */
    int         max_exp;                    /* 目盛りのスケール値 */
    int         graph_type;                 /* グラフの型 */
    /* private */
    int         Ww;                         /* ウィンドウ幅 */
    int         Wh;                         /* ウィンドウ高 */
    int         Ax;                         /* 描画領域x座標 */
    int         Ay;                         /* 描画領域y座標 */
    int         Aw;                         /* 描画領域幅 */
    int         Ah;                         /* 描画領域高 */
    int         Rx;                         /* グラフ領域x座標 */
    int         Ry;                         /* グラフ領域y座標 */
    int         Rw;                         /* グラフ領域幅 */
    int         Rh;                         /* グラフ領域高 */
    int         Ox;                         /* プロット領域x座標 */
    int         Oy;                         /* プロット領域y座標 */
    int         Ow;                         /* プロット領域幅 */
    int         Oh;                         /* プロット領域高 */
        COLORREF  line_col;                 // 線の色
} GrAttr;

#define GR_MODE_NORMAL                  1
#define GR_MODE_ADD                         2
#define GR_MODE_FLOW                    3
```

```
//#define       GR_FONT_FORMAT "-*-%s-%s-%c-normal--*-%d-*-*-*-*-*-*"
#define GR_FONT_NAME_RANGE { "helvetica", "times", "lucida", "courier", "symbol", NULL}

#define GR_DEFAULT_FONTNAME                     "helvetica"
#define GR_DEFAULT_FONTBOLD                     "medium"
#define GR_DEFAULT_FONTITALIC                   "roman"
#define GR_DEFAULT_FONTSIZE                     14
#define GR_DEFAULT_LABEL_FONTNAME               GR_DEFAULT_FONTNAME
#define GR_DEFAULT_LABEL_FONTBOLD               GR_DEFAULT_FONTBOLD
#define GR_DEFAULT_LABEL_FONTITALIC     GR_DEFAULT_FONTITALIC
#define GR_DEFAULT_LABEL_FONTSIZE               14
#define GR_DEFAULT_TITLE_FONTNAME               GR_DEFAULT_FONTNAME
#define GR_DEFAULT_TITLE_FONTBOLD               "bold"
#define GR_DEFAULT_TITLE_FONTITALIC     GR_DEFAULT_FONTITALIC
#define GR_DEFAULT_TITLE_FONTSIZE               20
#define GR_DEFAULT_TICK_FONTNAME                GR_DEFAULT_FONTNAME
#define GR_DEFAULT_TICK_FONTBOLD                "medium"
#define GR_DEFAULT_TICK_FONTITALIC              "roman"
#define GR_DEFAULT_TICK_FONTSIZE                12
#define GR_DEFAULT_TICKSUB_FONTNAME             GR_DEFAULT_FONTNAME
#define GR_DEFAULT_TICKSUB_FONTBOLD     "medium"
#define GR_DEFAULT_TICKSUB_FONTITALIC   "roman"
#define GR_DEFAULT_TICKSUB_FONTSIZE     10
#define GR_DEFAULT_DRAW_MODE                    GR_MODE_NORMAL
#define GR_DEFAULT_MAX_EXP                              4
#define GR_DEFAULT_LINE_STYLE           "-"
#define GR_DEFAULT_LINE_WIDTH           0

#define GR_DEFAULT_GRAPH_TYPE           GR_TYPE_XY

// PSおよびEPS用のフォント名、フォントサイズ
#define GR_PS_FONTNAME                          "helvetica"
#define GR_PS_LABEL_FONTNAME            GR_PS_FONTNAME
#define GR_PS_TITLE_FONTNAME            GR_PS_FONTNAME
#define GR_PS_TICK_FONTNAME                     GR_PS_FONTNAME
#define GR_PS_TICKSUB_FONTNAME          GR_PS_FONTNAME
#define GR_PS_FONTSIZE                          10
#define GR_PS_LABEL_FONTSIZE            10
#define GR_PS_TITLE_FONTSIZE            16
#define GR_PS_TICK_FONTSIZE             10
#define GR_PS_TICKSUB_FONTSIZE           8

/* ＰＳでのグラフの各線の長さを定義 */
#define PS_DASH_PATTERN_DASH_LEN            2.0
#define PS_DASH_PATTERN_DOTTEDLINE_LEN      12.0
#define PS_DASH_PATTERN_DASH_LONG_LEN       20.0
#define PS_DASH_PATTERN_DOTTEDLINE_LONG_LEN 12.0*4.0

#ifndef EXTRN
#define EXTRN extern
#endif

EXTRN GrAttr    *gGraph;            /* グラフ情報 */
EXTRN int gPSOnlyMode;                      /* PS出力モード */
EXTRN int gGraphNo[MAX_GRAPH_COUNT];
EXTRN int gTitleHeight;                 // タイトル領域の高さ

#define GrGetWindow(gr) (gGraph[(gr)].win)
#define GrSetWindowSize(gr, w, h) (gGraph[(gr)].Ww = (w), gGraph[(gr)].Wh = (h))
#define GrSetPSOnlyMode() (gPSOnlyMode = True)
#define GrResetPSOnlyMode() (gPSOnlyMode = False)

/*
 * プライベート定数の定義
 */
/* 目盛り定数定義 */
#define GR_TICK_NUM_TICKS               5       /* Major目盛りの数 */
#define GR_TICK_MAX_TICKS               20      /* Major目盛りの最大数 */
#define GR_TICK_MAX_TICKS_MINOR         5       /* Minor目盛りの最大数 */
#define GR_TICK_MAX_ALL_TICKS           ((GR_TICK_MAX_TICKS)*(GR_TICK_MAX_TICKS_MINOR)) /*
 目盛りの最大数 */
#define GR_TICK_LENGTH                  0.01    /* Major目盛りの相対長さ */
```

```
#define GR_TICK_MINOR_LENGTH            0.005   /* Minor目盛りの相対長さ */

#define GR_POS_DEFAULT_X                0.01    /* ウィンドウの左上の相対位置 */
#define GR_POS_DEFAULT_Y                0.01    /* ウィンドウの左上の相対位置 */
#define GR_POS_DEFAULT_WIDTH            0.98    /* ウィンドウの左上の相対幅 */
#define GR_POS_DEFAULT_HEIGHT           0.98    /* ウィンドウの左上の相対高さ */
#define GR_POS_LABEL_HEIGHT             19      /* ラベル領域の高さ(pixel) */
#define GR_POS_XTICKS_HEIGHT            20      /* x目盛りの高さ(pixel) */
#define GR_POS_YTICKS_WIDTH             28      /* y目盛りの幅(pixel) */
#define GR_POS_XTICKS_MARGIN            0.01    /* x目盛りの相対マージン */
#define GR_POS_YTICKS_MARGIN            4 //8       /* x目盛りのマージン */
#define GR_POS_TITLE_HEIGHT             5       /* タイトル領域の高さ */
#define GR_POS_GRAPH_AREA_LEFT_MARGIN   (GR_POS_LABEL_HEIGHT) /*  */
#define GR_POS_GRAPH_AREA_RIGHT_MARGIN  5       /*  */
#define GR_POS_GRAPH_AREA_TOP_MARGIN    8       /*(GR_POS_LABEL_HEIGHT)*/
#define GR_POS_GRAPH_AREA_BOTTOM_MARGIN (GR_POS_LABEL_HEIGHT) /*  */
#define GR_POS_PLOT_AREA_LEFT_MARGIN    (GR_POS_YTICKS_WIDTH) /*  */
#define GR_POS_PLOT_AREA_RIGHT_MARGIN   5       /*  */
#define GR_POS_PLOT_AREA_TOP_MARGIN     10      /*  */
#define GR_POS_PLOT_AREA_BOTTOM_MARGIN  (GR_POS_XTICKS_HEIGHT) /*  */
#define GR_POS_METER_TICK_MARKS         0.90    /*  */
#define GR_POS_METER_WIDTH_MARGIN       0.10    /*  */
#define GR_POS_METER_HEIGHT_MARGIN      0.05    /*  */
#define GR_POS_METER_MAJOR_TICKS1       0.80    /*  */
#define GR_POS_METER_MAJOR_TICKS2       0.70    /*  */
#define GR_POS_METER_MINOR_TICKS1       0.78    /*  */
#define GR_POS_METER_MINOR_TICKS2       0.72    /*  */
#define GR_POS_METER_STING1             0.65    /*  */
#define GR_POS_METER_STING2             0.00    /*  */
#define GR_COLOR_LOW                    (-0.25) /*  */
#define GR_COLOR_HIGH                   (0.15)  /*  */

#define Free(a)                 ((a) ? ((void)delete[] a,a=NULL): NULL)

#ifndef exp10
#define exp10(a)  pow(10, (a))
#endif

/*
 * 関数の宣言
 */

int  grFreeAxisAttributes(int);
int  grGetPointByAngle(int, int, float, float, int *, int *);
int  GrPlot(Graph, Number *, int, Number *, int, Number *, char *);
int  GrSet(Graph gr, ...);
int  GrPrint(Graph, FILE *);
int  grCalcAxisAttributes(int);
int  grCalcAxisOrigin(int);
int  grMakeNormalTicks(float *, float **, int *n, float **, int *);
int  grMakeTickMarks(float *, int, char ***, int *, char **, int);
int  grPSSetFont(FILE *, char *, char *, char *, int);

void GrInit();
void hsv_to_rgb(float, float, float, float *, float *, float *);
void rgb_to_hsv(float, float, float, float *, float *, float *);

Graph GrOpen(DATA *, int);

#endif /* end of __graph_h__ */
```

```
// HeadDlg.cpp : インプリメンテーション ファイル
//

#include "stdafx.h"
#include "PLOT.h"
#include "HeadDlg.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CHeadDlg ダイアログ


CHeadDlg::CHeadDlg(CWnd* pParent /*=NULL*/)
        : CDialog(CHeadDlg::IDD, pParent)
{
        //{{AFX_DATA_INIT(CHeadDlg)
                // メモ - ClassWizard はこの位置にマッピング用のマクロを追加または削除しま
す。
        //}}AFX_DATA_INIT
}


void CHeadDlg::DoDataExchange(CDataExchange* pDX)
{
        CDialog::DoDataExchange(pDX);
        //{{AFX_DATA_MAP(CHeadDlg)
        DDX_Control(pDX, IDC_EDIT2, m_BodyName);
        DDX_Control(pDX, IDC_EDIT1, m_HeadName);
        //}}AFX_DATA_MAP
}


BEGIN_MESSAGE_MAP(CHeadDlg, CDialog)
        //{{AFX_MSG_MAP(CHeadDlg)
        ON_BN_CLICKED(IDC_BUTTON1, OnRefHead)
        ON_BN_CLICKED(IDC_BUTTON2, OnRefBody)
        //}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CHeadDlg メッセージ ハンドラ

void CHeadDlg::OnRefHead()
{
        CString      csPath;
        CFileDialog  f(TRUE, NULL, NULL, OFN_HIDEREADONLY|OFN_OVERWRITEPROMPT,
                                "すべてのファイル(*.*)|*.*||");

        f.m_ofn.lpstrTitle = _T("ヘッダーファイルを開く");
        if (f.DoModal() == IDOK) {
                csPath = f.GetPathName();
                m_HeadName.SetWindowText(csPath);
        }
}

void CHeadDlg::OnRefBody()
{
        CString      csPath;
        CFileDialog  f(TRUE, NULL, NULL, OFN_HIDEREADONLY|OFN_OVERWRITEPROMPT,
                                "すべてのファイル(*.*)|*.*||");

        f.m_ofn.lpstrTitle = _T("ボディファイルを開く");
        if (f.DoModal() == IDOK) {
                csPath = f.GetPathName();
                m_BodyName.SetWindowText(csPath);
        }
}
```

```
void CHeadDlg::OnOK()
{
        m_HeadName.GetWindowText(m_csHead);
        m_BodyName.GetWindowText(m_csBody);

        // 不要な文字（空白、タブ）を取り除く
        m_csHead.TrimLeft();
        m_csHead.TrimRight();
        m_csBody.TrimLeft();
        m_csBody.TrimRight();

        CDialog::OnOK();
}

void CHeadDlg::OnCancel()
{
        // TODO: この位置に特別な後処理を追加してください。

        CDialog::OnCancel();
}

BOOL CHeadDlg::OnInitDialog()
{
        CDialog::OnInitDialog();

        m_HeadName.SetWindowText("");
        m_BodyName.SetWindowText("");

        m_HeadName.SetFocus();

        return FALSE;  // コントロールにフォーカスを設定しないとき、戻り値は TRUE となりま
す
                       // 例外: OCX プロパティ ページの戻り値は FALSE となります
}

void CHeadDlg::GetFname(char *cHead, char *cBody)
{
        // ヘッダーファイル名の取得
        strcpy(cHead, (char*)LPCTSTR(m_csHead));

        // ボディファイル名の取得
        strcpy(cBody, (char*)LPCTSTR(m_csBody));
}
```

```
#if !defined(AFX_HEADDLG_H__606F0DA0_6F39_11D3_ABB7_00C04F8862AB__INCLUDED_)
#define AFX_HEADDLG_H__606F0DA0_6F39_11D3_ABB7_00C04F8862AB__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000
// HeadDlg.h : ヘッダー ファイル
//

/////////////////////////////////////////////////////////////////////////////
// CHeadDlg ダイアログ

class CHeadDlg : public CDialog
{
// コンストラクション
public:
        CHeadDlg(CWnd* pParent = NULL);   // 標準のコンストラクタ

// ダイアログ データ
        //{{AFX_DATA(CHeadDlg)
        enum { IDD = IDD_DIALOG1 };
        CEdit   m_BodyName;
        CEdit   m_HeadName;
        //}}AFX_DATA


// オーバーライド
        // ClassWizard は仮想関数のオーバーライドを生成します。
        //{{AFX_VIRTUAL(CHeadDlg)
        protected:
        virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV サポート
        //}}AFX_VIRTUAL

// インプリメンテーション
protected:

private:
        CString  m_csHead;    // ヘッダーファイル名
        CString  m_csBody;    // ボディファイル名

public:
        void GetFname(char*, char*);

        // 生成されたメッセージ マップ関数
        //{{AFX_MSG(CHeadDlg)
        afx_msg void OnRefHead();
        afx_msg void OnRefBody();
        virtual void OnOK();
        virtual void OnCancel();
        virtual BOOL OnInitDialog();
        //}}AFX_MSG
        DECLARE_MESSAGE_MAP()
};

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ は前行の直前に追加の宣言を挿入します。

#endif // !defined(AFX_HEADDLG_H__606F0DA0_6F39_11D3_ABB7_00C04F8862AB__INCLUDED_)
```

```
// Line.cpp
//
#include "stdafx.h"
#include "Line.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif
//
//////////////////////////////////////////////////////////////////////////
// CDashLine

CLine::CLine(CDC& dc): m_DC(dc)
{
}

CLine::~CLine()
{
}

void CLine::DrawLine(int x1, int y1, int x2, int y2)
{
        m_DC.MoveTo(x1, y1);
        m_DC.LineTo(x2, y2);
}

void CLine::DrawRectangle(int x, int y, int w, int h)
{
        m_DC.MoveTo(x, y);
        m_DC.LineTo(x, y+h);
        m_DC.LineTo(x+w, y+h);
        m_DC.LineTo(x+w, y);
        m_DC.LineTo(x, y);
}

void CLine::DrawSegments(RECT rec[], int num)
{
        for (int i=0; i<num; i++) {
                m_DC.MoveTo(rec[i].left, rec[i].top);
                m_DC.LineTo(rec[i].right, rec[i].bottom);
        }
}

void CLine::DrawLines(POINT pos[], int num)
{
        if (num < 2) return;

        m_DC.MoveTo(pos[0].x, pos[0].y);

        for (int i=0; i<num; i++) {
                m_DC.LineTo(pos[i].x, pos[i].y);
        }
}
```

```
// Line.h :
////////////////////////////////////////////////////////////////////////////
// CLine

class CLine
{
public:
        CLine(CDC&);
        ~CLine();

protected:
        CDC& m_DC;

public:
        void DrawLine(int, int, int, int);
        void DrawRectangle(int, int, int, int);
        void DrawSegments(RECT*, int);
        void DrawLines(POINT*, int);
};
```

```
// MainFrm.cpp : CMainFrame クラスの動作の定義を行います。
//

#include "stdafx.h"
#include "PLOT.h"

#include "MainFrm.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CMainFrame

IMPLEMENT_DYNCREATE(CMainFrame, CFrameWnd)

BEGIN_MESSAGE_MAP(CMainFrame, CFrameWnd)
        //{{AFX_MSG_MAP(CMainFrame)
        ON_WM_CREATE()
        ON_WM_SIZE()
        //}}AFX_MSG_MAP
END_MESSAGE_MAP()

static UINT indicators[] =
{
        ID_SEPARATOR,           // ステータス ライン インジケータ
        ID_INDICATOR_KANA,
        ID_INDICATOR_CAPS,
        ID_INDICATOR_NUM,
        ID_INDICATOR_SCRL,
};

/////////////////////////////////////////////////////////////////////////////
// CMainFrame クラスの構築/消滅

CMainFrame::CMainFrame()
{
        // TODO: この位置にメンバの初期化処理コードを追加してください。

}

CMainFrame::~CMainFrame()
{
}

int CMainFrame::OnCreate(LPCREATESTRUCT lpCreateStruct)
{
        if (CFrameWnd::OnCreate(lpCreateStruct) == -1)
                return -1;

        if (!m_wndToolBar.CreateEx(this, TBSTYLE_FLAT, WS_CHILD | WS_VISIBLE | CBRS_TOP
                | CBRS_GRIPPER | CBRS_TOOLTIPS | CBRS_FLYBY | CBRS_SIZE_DYNAMIC) ||
                !m_wndToolBar.LoadToolBar(IDR_MAINFRAME))
        {
                TRACE0("Failed to create toolbar\n");
                return -1;      // 作成に失敗
        }

        if (!m_wndStatusBar.Create(this) ||
                !m_wndStatusBar.SetIndicators(indicators,
                  sizeof(indicators)/sizeof(UINT)))
        {
                TRACE0("Failed to create status bar\n");
                return -1;      // 作成に失敗
        }

        // TODO: ツール バーをドッキング可能にしない場合は以下の３行を削除
        //       してください。
        m_wndToolBar.EnableDocking(CBRS_ALIGN_ANY);
        EnableDocking(CBRS_ALIGN_ANY);
        DockControlBar(&m_wndToolBar);

        return 0;
}

BOOL CMainFrame::PreCreateWindow(CREATESTRUCT& cs)
{
        int  x, y, cx, cy;

        ((CPLOTApp *)AfxGetApp())->GetGeometry(&x, &y, &cx, &cy);

        cs.dwExStyle |= WS_EX_ACCEPTFILES;

        cs.cx = cx;
        cs.cy = cy;
        cs.x = x;
        cs.y = y;

        if (!CFrameWnd::PreCreateWindow(cs))
                return FALSE;

        return TRUE;
}

/////////////////////////////////////////////////////////////////////////////
// CMainFrame クラスの診断

#ifdef _DEBUG
void CMainFrame::AssertValid() const
{
        CFrameWnd::AssertValid();
}

void CMainFrame::Dump(CDumpContext& dc) const
{
        CFrameWnd::Dump(dc);
}

#endif //_DEBUG

/////////////////////////////////////////////////////////////////////////////
// CMainFrame メッセージ ハンドラ

void CMainFrame::OnSize(UINT nType, int cx, int cy)
{
        CFrameWnd::OnSize(nType, cx, cy);

        RECT  rc;

        GetWindowRect(&rc);

        ((CPLOTApp*)AfxGetApp())->SetCxy(rc.right-rc.left, rc.bottom-rc.top);
}
```

```
// MainFrm.h : CMainFrame クラスの宣言およびインターフェイスの定義をします。
//
/////////////////////////////////////////////////////////////////////////////

#if !defined(AFX_MAINFRM_H__1B91EA89_65AE_11D3_ABB3_00C04F8862AB__INCLUDED_)
#define AFX_MAINFRM_H__1B91EA89_65AE_11D3_ABB3_00C04F8862AB__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

class CMainFrame : public CFrameWnd
{

protected: // シリアライズ機能のみから作成します。
        CMainFrame();
        DECLARE_DYNCREATE(CMainFrame)

// アトリビュート
public:

// オペレーション
public:

// オーバーライド
        // ClassWizard は仮想関数のオーバーライドを生成します。
        //{{AFX_VIRTUAL(CMainFrame)
        public:
        virtual BOOL PreCreateWindow(CREATESTRUCT& cs);
        //}}AFX_VIRTUAL

// インプリメンテーション
public:
        virtual ~CMainFrame();
#ifdef _DEBUG
        virtual void AssertValid() const;
        virtual void Dump(CDumpContext& dc) const;
#endif

protected:  // コントロール バー用メンバ
        CStatusBar  m_wndStatusBar;
        CToolBar    m_wndToolBar;

// 生成されたメッセージ マップ関数
protected:
        //{{AFX_MSG(CMainFrame)
        afx_msg int OnCreate(LPCREATESTRUCT lpCreateStruct);
        afx_msg void OnSize(UINT nType, int cx, int cy);
        //}}AFX_MSG
        DECLARE_MESSAGE_MAP()
};

/////////////////////////////////////////////////////////////////////////////

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ は前行の直前に追加の宣言を挿入します。

#endif // !defined(AFX_MAINFRM_H__1B91EA89_65AE_11D3_ABB3_00C04F8862AB__INCLUDED_)
```

```
// PLOT.cpp : アプリケーション用クラスの機能定義を行います。
//

#include "stdafx.h"
#include "PLOT.h"

#include "MainFrm.h"
#include "PLOTDoc.h"
#include "PLOTView.h"
#include "HeadDlg.h"
#include "PropDlg.h"
#include "SvpsDlg.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

int GrClose(int);
int GrChangeFont(int);
int DrawGraph(DATA *, BOOL);
int PrintPsGraph(char *, DATA *);

/////////////////////////////////////////////////////////////////////////////
// CPLOTApp

BEGIN_MESSAGE_MAP(CPLOTApp, CWinApp)
        //{{AFX_MSG_MAP(CPLOTApp)
        ON_COMMAND(ID_APP_ABOUT, OnAppAbout)
        ON_COMMAND(ID_PROP, OnProp)
        ON_COMMAND(ID_HELP_PLOT, OnHelpPlot)
        ON_UPDATE_COMMAND_UI(ID_HELP_PLOT, OnUpdateHelpPlot)
        ON_UPDATE_COMMAND_UI(ID_FILE_PRINT, OnUpdateFilePrint)
        ON_UPDATE_COMMAND_UI(ID_FILE_PRINT_PREVIEW, OnUpdateFilePrintPreview)
        ON_UPDATE_COMMAND_UI(ID_FILE_PRINT_SETUP, OnUpdateFilePrintSetup)
        ON_UPDATE_COMMAND_UI(ID_FILE_SAVE_PS, OnUpdateFileSavePs)
        ON_UPDATE_COMMAND_UI(ID_PROP, OnUpdateProp)
        //}}AFX_MSG_MAP
        // 標準のファイル基本ドキュメント コマンド
        ON_COMMAND(ID_FILE_SAVE_PS, OnFileSavePS)
        ON_COMMAND(ID_FILE_OPEN, OnFileOpen)
        ON_COMMAND(ID_FILE_OPEN_HEAD, OnFileOpenHead)
        // 標準の印刷セットアップ コマンド
        ON_COMMAND(ID_FILE_PRINT_SETUP, CWinApp::OnFilePrintSetup)
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CPLOTApp クラスの構築

CPLOTApp::CPLOTApp()
{
        OSVERSIONINFO  os;

        if (GetVersionEx(&os) == TRUE) {
                m_bWin98 = (os.dwPlatformId==VER_PLATFORM_WIN32_WINDOWS)? TRUE: FALSE;
        }
        else {
                m_bWin98 = TRUE;
        }

        try {
                // 必要な領域の確保
                m_Path = new char[PATH_LEN+1];          // カレントパス名
                m_IniFname = new char[FNAME_LEN+1];     // 初期化ファイル名
                m_DataFname = new char[FNAME_LEN+1];    // データファイル名
                m_BodyFname = new char[FNAME_LEN+1];    // ボディファイル名
                m_HeadFname = new char[FNAME_LEN+1];    // ヘッダーファイル名
                m_PsFname = new char[FNAME_LEN+1];      // ヘッダーファイル名
                m_EpsFname = new char[FNAME_LEN+1];     // ヘッダーファイル名
                m_HelpFname = new char[FNAME_LEN+1];    // ヘルプファイル名

                for (int i=0; i<4; i++) {
                        m_FontName[i] = new char[FNFACE_LEN+1];    // フォント名（タイトル
、ラベル、目盛り等）
                        m_FontSize[i] = new char[FNSIZE_LEN+1];    // フォントサイズ
                        m_FontStyle[i] = new char[FNSTYLE_LEN+1];  // フォントスタイル
                }
        }
        catch (CMemoryException* e) {
                char buf[100];
                sprintf(buf, "Cannot allocate memory.\n");
                PrtMessage(buf, 0);
                e->Delete();
                exit(1);
        }

        // 初期値の設定
        m_x  = DEF_X;       // 表示Ｘ座標
        m_y  = DEF_Y;       // 表示Ｙ座標
        m_cx = DEF_CX;      // 画面幅
        m_cy = DEF_CY;      // 画面高

        // 初期化、データ、ボディ、ヘッダー、ＰＳ、ＥＰＳファイル名の設定
        GetCurrentDirectory(PATH_LEN, m_Path);
        sprintf(m_IniFname, "%s\\PLOT.ini", m_Path);
        sprintf(m_DataFname, "%s\\%s", m_Path, DEFAULT_FILENAME);

        m_BodyFname[0] = m_HeadFname[0] = 0;
        m_PsFname[0] = m_EpsFname[0] = 0;

        m_Ori = DEFAULT_PAPER_ORIENTATION;
        m_Aspect = -1.0f;
        m_Dot = 1.0f;

        // 初期化ファイルの読み込み
        GetIniFile();

        if (m_Ori == PAPER_PORTRAIT) {
                if (m_cx==DEF_CX && m_cy==DEF_CY) {
                        m_cx = DEF_CY;
                        m_cy = DEF_CX;
                }
        }

        m_pData = NULL;
}

/////////////////////////////////////////////////////////////////////////////
// 唯一の CPLOTApp オブジェクト

CPLOTApp theApp;

/////////////////////////////////////////////////////////////////////////////
// CPLOTApp クラスの初期化

BOOL CPLOTApp::InitInstance()
{

        AfxEnableControlContainer();

        // 標準的な初期化処理
        // もしこれらの機能を使用せず、実行ファイルのサイズを小さく
        // したければ以下の特定の初期化ルーチンの中から不必要なもの
        // を削除してください。

#ifdef _AFXDLL
        Enable3dControls();             // 共有 DLL の中で MFC を使用する場合にはここを呼
び出してください。
#else
        Enable3dControlsStatic();       // MFC と静的にリンクしている場合にはここを呼び出
してください。
#endif

        // 設定が保存される下のレジストリ キーを変更します。
        // TODO: この文字列を、会社名または所属など適切なものに
```

```
        // 変更してください。
//      SetRegistryKey(_T("Local AppWizard-Generated Applications"));

        LoadStdProfileSettings();  // 標準の INI ファイルのオプションをロードします (MRU
を含む)

        // アプリケーション用のドキュメント テンプレートを登録します。ドキュメント テンプ
レート
        //  はドキュメント、フレーム ウィンドウとビューを結合するために機能します。

        CSingleDocTemplate* pDocTemplate;
        pDocTemplate = new CSingleDocTemplate(
                IDR_MAINFRAME,
                RUNTIME_CLASS(CPLOTDoc),
                RUNTIME_CLASS(CMainFrame),       // メイン SDI フレーム ウィンドウ
                RUNTIME_CLASS(CPLOTView));
        AddDocTemplate(pDocTemplate);

        // コマンドオプションの解析
        if (AnaArgs(m_lpCmdLine) == FALSE) return FALSE;

        // データファイル等を読み込む
        if (m_pData = ReadDataFile(m_DataFname, m_HeadFname, m_BodyFname)) {

                m_pData->aspectRatio = m_Aspect;
                m_pData->orientation = m_Ori;
                m_pData->dotratio = m_Dot;
                // 画面サイズの設定
                m_pData->mainWinWidth = m_cx;
                m_pData->mainWinHeight = m_cy;

                SetPlotWindowSize(m_pData);

                if (m_PsFname[0] || m_EpsFname[0]) {  // -ps,-epsオプション指定
                        m_pData->mode = PS_ONLY_MODE;
                        DrawGraph(m_pData, FALSE);

                        if (m_PsFname[0]) PrintPsGraph(m_PsFname, m_pData);
                        if (m_EpsFname[0]) {    // EPSファイル名
                                m_pData->mode = EPS_ONLY_MODE;
                                PrintPsGraph(m_EpsFname, m_pData);
                        }
                        return FALSE;
                }
                else {
                        CreateFont();           // フォントを作成
                        m_FrPen.CreatePen(PS_SOLID, 1, RGB(0,0,0));
                        DrawGraph(m_pData, FALSE);
                }
        }
        else {
//              PrtMessage("ACR PLOT ツールが起動できません。");
                return FALSE;
        }

        // DDE、file open など標準のシェル コマンドのコマンドラインを解析します。
        CCommandLineInfo cmdInfo;
        ParseCommandLine(cmdInfo);

        cmdInfo.m_nShellCommand = CCommandLineInfo::FileNew;
        // コマンドラインでディスパッチ コマンドを指定します。
        if (!ProcessShellCommand(cmdInfo))
                return FALSE;

        // メイン ウィンドウが初期化されたので、表示と更新を行います。
        m_pMainWnd->ShowWindow(SW_SHOW);
        m_pMainWnd->UpdateWindow();

#if 0
        // ヘルプファイルがない時は、メニューから使い方を削除
        if (m_HelpFname[0] == 0) {
                (m_pMainWnd->GetMenu())->DeleteMenu(ID_HELP_PLOT, MF_BYCOMMAND);
                m_pMainWnd->DrawMenuBar();
        }
#endif
        return TRUE;
}

// コマンドオプションの解析
BOOL CPLOTApp::AnaArgs(char *lpCmdLine)
{
        char  cmdbuf[101];
        char  *bp, *be;
        int   cmd_no=0;
        float f;
        void  PrtUsage();

        while (*lpCmdLine) {
                // 先頭の空白およびタブを除く
                while (*lpCmdLine==' ' || *lpCmdLine=='\t') lpCmdLine++;
                bp = cmdbuf;
                be = bp + 100;
                while (*lpCmdLine && bp<be && *lpCmdLine!=' ') *bp++ = *lpCmdLine++;
                *bp = 0;

                if (bp > cmdbuf) {
                        if (bp >= be) {
                                PrtUsage();
                                return FALSE;
                        }
                        else {   // オプションのチェック
                                if (cmd_no==0 && cmdbuf[0]=='-') {
                                        bp = cmdbuf + 1;
                                        if (!strcmp(bp, "data")) cmd_no = 1;
                                        else if (!strcmp(bp, "geometry")) cmd_no = 2;
                                        else if (!strcmp(bp, "head")) cmd_no = 3;
                                        else if (!strcmp(bp, "body")) cmd_no = 4;
                                        else if (!strcmp(bp, "ps")) cmd_no = 5;
                                        else if (!strcmp(bp, "eps")) cmd_no = 6;
                                        else if (!strcmp(bp, "landscape"))
                                                m_Ori = PAPER_LANDSCAPE;
                                        else if (!strcmp(bp, "portrait"))
                                                m_Ori = PAPER_PORTRAIT;
                                        else if (!strcmp(bp, "dotratio")) cmd_no = 7;
                                        else if (!strcmp(bp, "aspectratio")) cmd_no = 8;
                                        else {
                                                PrtUsage();
                                                return FALSE;
                                        }
                                }
                                else {
                                        switch (cmd_no) {
                                        case 1:         // データファイルの指定
                                                SetFileName(cmdbuf, m_DataFname);
                                                break;
                                        case 2:         // 画面表示サイズと位置の指定
                                                SetGeometry(cmdbuf);
                                                break;
                                        case 3:         // ヘッダーファイルの指定
                                                SetFileName(cmdbuf, m_HeadFname);
                                                break;
                                        case 4:         // ボディファイルの指定
                                                SetFileName(cmdbuf, m_BodyFname);
                                                break;
                                        case 5:         // PSファイルの指定
                                                SetFileName(cmdbuf, m_PsFname);
                                                break;
                                        case 6:         // EPSファイルの指定
                                                SetFileName(cmdbuf, m_EpsFname);
                                                break;
                                        case 7:         // DotRatio指定
                                                sscanf(cmdbuf, "%f", &f);
                                                SetDotR(f);
                                                break;
                                        case 8:         // AspectRatio指定
                                                sscanf(cmdbuf, "%f", &f);
```

```
                                        SetAspectR(f);
                                        break;
                                default:        // データファイルとして処理
                                        SetFileName(cmdbuf, m_DataFname);
                                        break;
                                }
                                cmd_no = 0;
                        }
                }
                else
                        return TRUE;
        }

        return TRUE;
}

// 画面の再描画
void CPLOTApp::ReDrawGraph(int cx, int cy)
{
#if 0
        m_cx = cx;
        m_cy = cy;
#endif

        // クライアント領域最大幅、高さの設定
        m_pData->mainWinWidth = cx;
        m_pData->mainWinHeight = cy;

        // グラフのサイズを計算、設定
        SetPlotWindowSize(m_pData);

        DrawGraph(m_pData, TRUE);
}

// 画面の再描画
void CPLOTApp::ReDrawGraph()
{
        // 縦横比、ドット比、表示の向き、画面最大幅、高さの設定
        m_pData->aspectRatio = m_Aspect;
        m_pData->orientation = m_Ori;
        m_pData->dotratio = m_Dot;
//      m_pData->mainWinWidth = m_cx;
//      m_pData->mainWinHeight = m_cy;

        // グラフのサイズを計算、設定
        SetPlotWindowSize(m_pData);

        DrawGraph(m_pData, FALSE);

        // 再描画
        m_pMainWnd->SetWindowPos(0, 0, 0, m_cx, m_cy,
                                                SWP_NOMOVE|SWP_NOZORDER);
        m_pMainWnd->Invalidate();
}

// 画面の再描画
void CPLOTApp::ReDrawGraph(float aspectR, float dotR, int nOri)
{
        // 縦横比、ドット比、表示の向きの設定
        m_pData->aspectRatio = m_Aspect = aspectR;
        m_pData->orientation = m_Ori = nOri;
        m_pData->dotratio = m_Dot = dotR;

        // グラフのサイズを計算、設定
        SetPlotWindowSize(m_pData);

        DrawGraph(m_pData, FALSE);

        // 再描画
        m_pMainWnd->Invalidate();
}
```

```
void CPLOTApp::SetFileName(char *cmd, char *fname)
{
        char  buf[101];
        char  *bp, *p;

        // ファイル名が" "でくくられていた時、削除
        if (cmd[0] == '"') {
                cmd[strlen(cmd)-1] = 0;
                p  = cmd;
                bp = cmd + 1;
                while (*bp) *p++ = *bp++;
                *p = 0;
        }

        if (cmd[0]=='\\' || cmd[1]==':') {    // 絶対パスで指定
                strcpy(fname, cmd);
        }
        else {
                // カレントディレクトリを取得
                GetCurrentDirectory(PATH_LEN, fname);
                p = fname + strlen(fname);

                while (*cmd) {
                        bp = buf;
                        while (*cmd && *cmd!='\\') *bp++ = *cmd++;
                        *bp = 0;
                        if (*cmd == 0) {
                                *p++ = '\\';
                                strcpy(p, buf);
                        }
                        else {
                                if (!strcmp(buf, "..")) {
                                        while (p > fname) {
                                                if (*p == '\\') {
                                                        *p = 0;
                                                        break;
                                                }
                                                p--;
                                        }
                                }
                                else if (buf[0] != '.') {
                                        *p++ = '\\';
                                        strcpy(p, buf);
                                        p += strlen(p);
                                }
                                cmd++;
                        }
                }
        }
}

void CPLOTApp::SetGeometry(char *cmd)
{
        char  buf[101];
        char  *bp;
        int   cx, cy;
        int   flag=0;
        int   plus=0;

        // デスクトップの幅、高さを取得
        cx = ::GetSystemMetrics(SM_CXSCREEN);
        cy = ::GetSystemMetrics(SM_CYSCREEN);

        while (*cmd) {
                bp = buf;
                while (*cmd && *cmd!='x' && *cmd!='+' && *cmd!='-')
                        *bp++ = *cmd++;
                *bp = 0;

                if (bp > buf) {    // 指定あり
                        switch (flag) {
                        case 0:    // 画面幅指定
                                m_cx = atoi(buf);
```

```
                        if (*cmd == 'x') flag = 1;
                        else {
                                flag = 2;
                                plus = (*cmd == '+')? 1: 0;
                        }
                        break;
                case 1:   // 画面高さ指定
                        m_cy = atoi(buf);
                        flag = 2;
                        plus = (*cmd == '+')? 1: 0;
                        break;
                case 2:   // 画面X位置指定
                        if (plus) m_x = atoi(buf);
                        else {
                                m_x = cx - m_cx - atoi(buf);
                                if (m_x < 0) m_x = 0;
                        }
                        flag = 3;
                        plus = (*cmd == '+')? 1: 0;
                        break;
                case 3:   // 画面Y位置指定
                        if (plus) m_y = atoi(buf);
                        else {
                                m_y = cy - m_cy - atoi(buf);
                                if (m_y < 0) m_y = 0;
                        }
                        flag = 4;
                        break;
                }
        }
        else {
                if (*cmd == 'x') flag = 1;
                else {
                        flag = 2;
                        plus = (*cmd == '+')? 1: 0;
                }
        }

        if (*cmd == 0) return;
        cmd++;
    }
}

void CPLOTApp::SetOri(int nOri)
{
    m_Ori = nOri;

    // 横方向で高さが大きい、縦方向で幅が大きい時に
    // 高さと幅の値を入れ替える。
    if ((m_Ori==PAPER_LANDSCAPE && m_cx<m_cy) ||
        (m_Ori==PAPER_PORTRAIT && m_cx>m_cy)) {
#if 1
        int  nWk = m_cx;
        m_cx = m_cy;
        m_cy = nWk;
#else
        if (m_Ori == PAPER_LANDSCAPE) {
            m_cx = m_Lcx;
            m_cy = m_Lcy;
        }
        else {
            m_cx = m_Lcy;
            m_cy = m_Lcx;
        }
#endif
    }
}

void CPLOTApp::GetFileName(char *dfile, char *bfile, char *hfile)
{
    strcpy(dfile, m_DataFname);    // データファイル名
    strcpy(bfile, m_BodyFname);    // ボディファイル名
    strcpy(hfile, m_HeadFname);    // ヘッダーファイル名
}

void CPLOTApp::GetPsFileName(char *psfile, char *epsfile)
{
    strcpy(psfile, m_PsFname);     // PSファイル名
    strcpy(epsfile, m_EpsFname);   // EPSファイル名
}

BOOL CPLOTApp::GetFontInfo(int id, char *name, char *size, char *style)
{
    if (id>=0 && id<4) {
        strcpy(name,  m_FontName[id]);      // 湊歌FACE名
        strcpy(size,  m_FontSize[id]);      // 湊歌参殉
        strcpy(style, m_FontStyle[id]);     // 湊歌柔荷
        return TRUE;
    }
    else
        return FALSE;
}

// フォント情報をメンバ変数に設定する
void CPLOTApp::SetFontInfo(int id, char *name, char *size, char *style)
{
#ifdef _WIN98_PLOT_
    static char *fn_style[] = {
        "標準", "太字", "斜体", "太字 斜体",
    };
#else /* WINNT */
    static char *fn_style[] = {
        "標準", "涙為爪", "伽惆, "涙為爪 伽惆,
    };
#endif

    if (id>=0 && id<4) {
        if (name) {
            strcpy(m_FontName[id],  name);
            strcpy(m_FontSize[id],  size);
            strcpy(m_FontStyle[id], style);
        }
        // 湊歌柔荷番号に変換
        m_nFontStyle[id] = 0;
        for (int i=0; i<4; i++) {
            if (!strcmp(fn_style[i], style)) {
                m_nFontStyle[id] = i;
                break;
            }
        }
    }
}

// フォントを作成しなおす
void CPLOTApp::ReMakeFont()
{
    // 現在使用のフォントを削除
    DeleteFont();

    // 新しいフォントを作成
    CreateFont();
}

// 新しいフォントを作成する
void CPLOTApp::CreateFont()
{
    HDC    hDC;
    int    size;                // フォントサイズ
    static struct {             // フォントスタイル
        int   nWeight;          // フォントの太さ
        BYTE  bItalic;          // イタリック文字か
    } style[4] = {
        {FW_NORMAL, 0}, {FW_BOLD, 0}, {FW_NORMAL, TRUE}, {FW_BOLD, TRUE},
    };

    // デスクトップウィンドウのDCを取得
```

```
        hDC = ::GetDC(::GetDesktopWindow());

        // タイトル、ラベル、目盛り、目盛り（上付）のフォントを作成
        for (int i=0; i<4; i++) {
                // サイズを高さに変更
                size = -MulDiv(atoi(m_FontSize[i]), ::GetDeviceCaps(hDC, LOGPIXELSY), 72);

                m_Font[i].CreateFont(size, 0, 0, 0, style[m_nFontStyle[i]].nWeight,
                                                        style[m_nFontStyle[i]].bItalic, 0,
 0,
                                                        DEFAULT_CHARSET, OUT_DEFAULT_PRECI
S,
                                                        CLIP_DEFAULT_PRECIS, DEFAULT_QUALI
TY,
                                                        DEFAULT_PITCH,
                                                        &m_FontName[i][0]);
        }

        // Ｙ軸用ラベルフォントの作成
        size = -MulDiv(atoi(m_FontSize[1]), ::GetDeviceCaps(hDC, LOGPIXELSY), 72);
        m_Font[4].CreateFont(size, 0, 900, 0, style[m_nFontStyle[1]].nWeight,
                                                        style[m_nFontStyle[1]].bItalic, 0,
 0,
                                                        DEFAULT_CHARSET, OUT_DEFAULT_PRECI
S,
                                                        CLIP_DEFAULT_PRECIS, DEFAULT_QUALI
TY,
                                                        DEFAULT_PITCH,
                                                        &m_FontName[1][0]);

}

// フォントオブジェクトの削除
void CPLOTApp::DeleteFont()
{
        for (int i=0; i<5; i++) {
                m_Font[i].DeleteObject();
        }
}

// 初期化ファイル（PLOT.INI）に情報を書き込み
void CPLOTApp::WriteIniFile()
{
        char  buf[10];

        // 縦横比の書き込み
        sprintf(buf, "%.3f", m_Aspect);
        WritePrivateProfileString("ACR_PLOT", "ASPECT", buf, m_IniFname);

        // 線幅比の書き込み
        sprintf(buf, "%.3f", m_Dot);
        WritePrivateProfileString("ACR_PLOT", "DOT", buf, m_IniFname);

        // 表示の向きの書き込み
        sprintf(buf, "%d", m_Ori+1);
        WritePrivateProfileString("ACR_PLOT", "ORIENTATION", buf, m_IniFname);

        // フォントフェース名、サイズ、スタイルの書き込み
#ifdef _WIN98_PLOT_
        for (int i=0; i<4; i++) {
                if (strcmp(m_FontStyle[i], "太字") == 0)
                        strcpy(m_FontStyle[i], "涙為爪");
                else if (strcmp(m_FontStyle[i], "ホ体") == 0)
                        strcpy(m_FontStyle[i], "伽惆");
                else if (strcmp(m_FontStyle[i], "太字 斜体") == 0)
                        strcpy(m_FontStyle[i], "涙為爪 伽惆");
        }
#endif

        WritePrivateProfileString("ACR_TITLE_FONT", "FACE", m_FontName[0], m_IniFname);
        WritePrivateProfileString("ACR_TITLE_FONT", "SIZE", m_FontSize[0], m_IniFname);
        WritePrivateProfileString("ACR_TITLE_FONT", "STYLE", m_FontStyle[0], m_IniFname);
        WritePrivateProfileString("ACR_LABEL_FONT", "FACE", m_FontName[1], m_IniFname);
        WritePrivateProfileString("ACR_LABEL_FONT", "SIZE", m_FontSize[1], m_IniFname);
        WritePrivateProfileString("ACR_LABEL_FONT", "STYLE", m_FontStyle[1], m_IniFname);
        WritePrivateProfileString("ACR_MEM_FONT", "FACE", m_FontName[2], m_IniFname);
        WritePrivateProfileString("ACR_MEM_FONT", "SIZE", m_FontSize[2], m_IniFname);
        WritePrivateProfileString("ACR_MEM_FONT", "STYLE", m_FontStyle[2], m_IniFname);
        WritePrivateProfileString("ACR_MEM2_FONT", "FACE", m_FontName[3], m_IniFname);
        WritePrivateProfileString("ACR_MEM2_FONT", "SIZE", m_FontSize[3], m_IniFname);
        WritePrivateProfileString("ACR_MEM2_FONT", "STYLE", m_FontStyle[3], m_IniFname);

}

// 初期化ファイル（PLOT.INI）から情報を取得
void CPLOTApp::GetIniFile()
{
        char   buf[FNAME_LEN+1];

        // DATAファイル名の取得
        GetPrivateProfileString("ACR_PLOT", "DATA", "", buf, FNAME_LEN, m_IniFname);
        if (buf[0]) strcpy(m_DataFname, buf);

        // HEADファイル名の取得
        GetPrivateProfileString("ACR_PLOT", "HEAD", "", m_HeadFname, FNAME_LEN, m_IniFname
);

        // BODYファイル名の取得
        GetPrivateProfileString("ACR_PLOT", "BODY", "", m_BodyFname, FNAME_LEN, m_IniFname
);

        // PSファイル名の取得
        GetPrivateProfileString("ACR_PLOT", "PS", "", m_PsFname, FNAME_LEN, m_IniFname);

        // EPSファイル名の取得
        GetPrivateProfileString("ACR_PLOT", "EPS", "", m_EpsFname, FNAME_LEN, m_IniFname);


        // 縦横比の取得
        GetPrivateProfileString("ACR_PLOT", "ASPECT", "", buf, FNAME_LEN, m_IniFname);
        if (buf[0]) m_Aspect = (float)atof(buf);

        // 線幅比の取得
        GetPrivateProfileString("ACR_PLOT", "DOT", "", buf, FNAME_LEN, m_IniFname);
        if (buf[0]) m_Dot = (float)atof(buf);

        // 表示向きの取得
        GetPrivateProfileString("ACR_PLOT", "ORIENTATION", "", buf, FNAME_LEN, m_IniFname)
;
        if (buf[0]) m_Ori = atoi(buf) - 1;

        // 初期表示サイズと位置の取得
        GetPrivateProfileString("ACR_PLOT", "GEOMETRY", "", buf, FNAME_LEN, m_IniFname);
        if (buf[0]) SetGeometry(buf);

        // ヘルプファイルの取得
        GetPrivateProfileString("ACR_PLOT", "HELP", "", m_HelpFname, FNAME_LEN, m_IniFname
);

        // フォントフェース名、サイズ、スタイルの取得
        GetPrivateProfileString("ACR_TITLE_FONT", "FACE", "Arial", m_FontName[0], FNAME_LE
N, m_IniFname);
        GetPrivateProfileString("ACR_TITLE_FONT", "SIZE", "14", m_FontSize[0], FNAME_LEN,
m_IniFname);
        GetPrivateProfileString("ACR_TITLE_FONT", "STYLE", "涙為爪", m_FontStyle[0], FNAME
_LEN, m_IniFname);
        GetPrivateProfileString("ACR_LABEL_FONT", "FACE", "Arial", m_FontName[1], FNAME_LE
N, m_IniFname);
        GetPrivateProfileString("ACR_LABEL_FONT", "SIZE", "10", m_FontSize[1], FNAME_LEN,
m_IniFname);
        GetPrivateProfileString("ACR_LABEL_FONT", "STYLE", "標準", m_FontStyle[1], FNAME_L
EN, m_IniFname);
        GetPrivateProfileString("ACR_MEM_FONT", "FACE", "Arial", m_FontName[2], FNAME_LEN,
 m_IniFname);
        GetPrivateProfileString("ACR_MEM_FONT", "SIZE", "9", m_FontSize[2], FNAME_LEN, m_I
niFname);
```

```
        GetPrivateProfileString("ACR_MEM_FONT", "STYLE", "標準", m_FontStyle[2], FNAME_LEN
, m_IniFname);
        GetPrivateProfileString("ACR_MEM2_FONT", "FACE", "Arial", m_FontName[3], FNAME_LEN
, m_IniFname);
        GetPrivateProfileString("ACR_MEM2_FONT", "SIZE", "8", m_FontSize[3], FNAME_LEN, m_
IniFname);
        GetPrivateProfileString("ACR_MEM2_FONT", "STYLE", "標準", m_FontStyle[3], FNAME_LE
N, m_IniFname);

#ifdef _WIN98_PLOT_
        for (int i=0; i<4; i++) {
                if (strcmp(m_FontStyle[i], "涙為爪") == 0)
                        strcpy(m_FontStyle[i], "太字");
                else if (strcmp(m_FontStyle[i], "伽惆") == 0)
                        strcpy(m_FontStyle[i], "ホ体");
                else if (strcmp(m_FontStyle[i], "涙為爪 伽惆") == 0)
                        strcpy(m_FontStyle[i], "太字 斜体");
                SetFontInfo(i, NULL, NULL, m_FontStyle[i]);
        }
#else
        for (int i=0; i<4; i++) SetFontInfo(i, NULL, NULL, m_FontStyle[i]);
#endif
}

// メッセージボックスを表示
void CPLOTApp::PrtMessage(char *msg, int mode)
{
        UINT  nType;

        if (mode == 0)      nType = MB_OK | MB_ICONSTOP;
        else if (mode == 1) nType = MB_OK | MB_ICONEXCLAMATION;

        AfxMessageBox(msg, nType);
}

BOOL CPLOTApp::LoadFile()
{
        int    tmp_cx, tmp_cy;
        extern int gGraphNo[];

        tmp_cx = m_pData->mainWinWidth;
        tmp_cy = m_pData->mainWinHeight;

        // 前使用のデータ領域の開放
        if (m_pData) {
                for (int i=0; i<m_pData->nGraphs; i++) {
                        GrClose(gGraphNo[i]);
                }
                delete[] m_pData->table;
                if (m_pData->table_label) delete[] m_pData->table_label;
                delete m_pData;
        }

        // データファイル等を読み込む
        if ((m_pData = ReadDataFile(m_DataFname, m_HeadFname, m_BodyFname)) == NULL) {
                PrtMessage(_T("ReadDataFile Error"));
                return FALSE;
        }

        // 縦横比、表示の向き、ドット比の設定
        m_pData->aspectRatio = m_Aspect;
        m_pData->orientation = m_Ori;
        m_pData->dotratio = m_Dot;
        // 画面サイズの設定
        m_pData->mainWinWidth = tmp_cx;
        m_pData->mainWinHeight = tmp_cy;
        // グラフサイズ等の計算、設定
        SetPlotWindowSize(m_pData);
        // 画面表示モードで
        m_pData->mode = WIN_DRAW_MODE;
        DrawGraph(m_pData, FALSE);

        m_pMainWnd->Invalidate();
        return TRUE;
}

// 「開く」用のコールバック関数
void CPLOTApp::OnFileOpen()
{
        CFileDialog  f(TRUE, NULL, NULL, OFN_HIDEREADONLY|OFN_OVERWRITEPROMPT,
                                        "データファイル(*.APT)|*.APT|すべてのファイル(*.*)
|*.*||");

        f.m_ofn.lpstrTitle = _T("データファイルを開く");
        if (f.DoModal() == IDOK) {
                // データファイル名の取得
                CString csPath = f.GetPathName();
                strcpy(m_DataFname, (char*)LPCTSTR(csPath));
                m_HeadFname[0] = m_BodyFname[0] = 0;

                // ファイルを読み込み、設定する
                LoadFile();
        }
}

// ヘッダを指定してオープン
void CPLOTApp::OnFileOpenHead()
{
        CHeadDlg  dlg;

        if (dlg.DoModal() == IDOK) {
                // ヘッダー、ボディファイル名の取得
                dlg.GetFname(m_HeadFname, m_BodyFname);

                // ファイルを読み込み、設定する
                LoadFile();
        }
}

// PSに変換して保存
void CPLOTApp::OnFileSavePS()
{
        char      cFname[FNAME_LEN+1];
        char      cPsFname[FNAME_LEN+1];
        CSvpsDlg  dlg;

        if (dlg.DoModal() == IDOK) {
                // 保存ファイル名を取得
                dlg.GetFname(cFname, FNAME_LEN);

                // ファイル保存モードに変更
                m_pData->mode = PS_ONLY_MODE;
                DrawGraph(m_pData, FALSE);

                // PSファイルに変換して保存
                if (dlg.GetSelPs() & 0x01) {
                        sprintf(cPsFname, "%s.ps", cFname);
                        PrintPsGraph(cPsFname, m_pData);
                }

                // EPSファイルに変換して保存
                if (dlg.GetSelPs() & 0x02) {
                        sprintf(cFname, "%s.eps", cFname);
                        m_pData->mode = EPS_ONLY_MODE;
                        PrintPsGraph(cFname, m_pData);
                }

                // 画面表示モードに戻す
                m_pData->mode = WIN_DRAW_MODE;
                DrawGraph(m_pData, FALSE);

                m_pMainWnd->Invalidate();
        }
}
```

```
/////////////////////////////////////////////////////////////////////////////
// アプリケーションのバージョン情報で使われる CAboutDlg ダイアログ

class CAboutDlg : public CDialog
{
public:
        CAboutDlg();

// ダイアログ データ
        //{{AFX_DATA(CAboutDlg)
        enum { IDD = IDD_ABOUTBOX };
        //}}AFX_DATA

        // ClassWizard 仮想関数のオーバーライドを生成します。
        //{{AFX_VIRTUAL(CAboutDlg)
        protected:
        virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV のサポート
        //}}AFX_VIRTUAL

// インプリメンテーション
protected:
        //{{AFX_MSG(CAboutDlg)
                // メッセージ ハンドラはありません。
        //}}AFX_MSG
        DECLARE_MESSAGE_MAP()
};

CAboutDlg::CAboutDlg() : CDialog(CAboutDlg::IDD)
{
        //{{AFX_DATA_INIT(CAboutDlg)
        //}}AFX_DATA_INIT
}

void CAboutDlg::DoDataExchange(CDataExchange* pDX)
{
        CDialog::DoDataExchange(pDX);
        //{{AFX_DATA_MAP(CAboutDlg)
        //}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CAboutDlg, CDialog)
        //{{AFX_MSG_MAP(CAboutDlg)
                // メッセージ ハンドラはありません。
        //}}AFX_MSG_MAP
END_MESSAGE_MAP()

// ダイアログを実行するためのアプリケーション コマンド
void CPLOTApp::OnAppAbout()
{
        CAboutDlg aboutDlg;
        aboutDlg.DoModal();
}

/////////////////////////////////////////////////////////////////////////////
// CPLOTApp メッセージ ハンドラ

void CPLOTApp::OnProp()
{
        CPropDlg  dlg;

        dlg.DoModal();
}

// ヘルプ画面の表示
void CPLOTApp::OnHelpPlot()
{
        int hIns;

        hIns = (int)ShellExecute(NULL,_T("open"), m_HelpFname,
                        NULL, NULL, SW_SHOW);

        if (hIns < 33) {
                char  buf[101];
                if (hIns == ERROR_FILE_NOT_FOUND) {
                        sprintf(buf, "'%s' file not found.", m_HelpFname);
                }
                else {
                        sprintf(buf, "can't open '%s'.", m_HelpFname);
                }
                PrtMessage(buf);
        }
}

void CPLOTApp::OnUpdateHelpPlot(CCmdUI* pCmdUI)
{
        // ヘルプファイルが存在すれば選択可、そうでない時は選択不可
        pCmdUI->Enable(m_HelpFname[0]? TRUE: FALSE);
}

void CPLOTApp::OnUpdateFilePrint(CCmdUI* pCmdUI)
{
        // グラフ表示時は選択可、そうでない時は選択不可
        if (!m_pData || m_pData->mode==NODRAW_MODE) {
                pCmdUI->Enable(FALSE);
        }
}

void CPLOTApp::OnUpdateFilePrintPreview(CCmdUI* pCmdUI)
{
        // グラフ表示時は選択可、そうでない時は選択不可
        if (!m_pData || m_pData->mode==NODRAW_MODE) {
                pCmdUI->Enable(FALSE);
        }
}

void CPLOTApp::OnUpdateFilePrintSetup(CCmdUI* pCmdUI)
{
        // グラフ表示時は選択可、そうでない時は選択不可
        if (!m_pData || m_pData->mode==NODRAW_MODE) {
                pCmdUI->Enable(FALSE);
        }
}

void CPLOTApp::OnUpdateFileSavePs(CCmdUI* pCmdUI)
{
        // グラフ表示時は選択可、そうでない時は選択不可
        if (!m_pData || m_pData->mode==NODRAW_MODE) {
                pCmdUI->Enable(FALSE);
        }
}

void CPLOTApp::OnUpdateProp(CCmdUI* pCmdUI)
{
        // グラフ表示時は選択可、そうでない時は選択不可
        if (!m_pData || m_pData->mode==NODRAW_MODE) {
                pCmdUI->Enable(FALSE);
        }
}

CDocument* CPLOTApp::OpenDocumentFile(LPCTSTR lpszFileName)
{
        strcpy(m_DataFname, (char*)lpszFileName);
        m_HeadFname[0] = m_BodyFname[0] = 0;

        // ファイルを読み込み、設定する
        LoadFile();

//      return CWinApp::OpenDocumentFile(lpszFileName);
        return NULL;
}
```

```
// PLOT.h : PLOT アプリケーションのメイン ヘッダー ファイル
//

#if !defined(AFX_PLOT_H__1B91EA85_65AE_11D3_ABB3_00C04F8862AB__INCLUDED_)
#define AFX_PLOT_H__1B91EA85_65AE_11D3_ABB3_00C04F8862AB__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

#ifndef __AFXWIN_H__
	#error include 'stdafx.h' before including this file for PCH
#endif

#include "resource.h"		// メイン シンボル
#include "PLOTDEF.h"

#define  _WIN98_PLOT_

// デフォルト値の定義
#define  DEF_X			100		// 初期表示Ｘ位置
#define  DEF_Y			100		// 初期表示Ｙ位置
#define  DEF_CX		(290*3)		// 初期画面幅
#define  DEF_CY		(200*3)		// 初期画面高

/////////////////////////////////////////////////////////////////////////////
// CPLOTApp:
// このクラスの動作の定義に関しては PLOT.cpp ファイルを参照してください。
//

class CPLOTApp : public CWinApp
{
public:
	CPLOTApp();
	void  GetGeometry(int *x, int *y, int *cx, int *cy) {  // 画面の位置、サイズ取得
		*x=m_x; *y=m_y; *cx=m_cx; *cy=m_cy;
	}
	void  GetFileName(char *, char *, char *);   // データ、ヘッダー、ボディファイル名取得
	void  GetPsFileName(char *, char *);
	BOOL  GetFontInfo(int, char*, char*, char*);   // フォント名、サイズ、スタイル取得

	int   GetOri() { return m_Ori; }
	float GetAspectR() { return m_Aspect; }
	float GetDotR() { return m_Dot; }
	DATA *GetData() { return m_pData; }
	CFont *GetFont(int id) { return &m_Font[id]; }
	CPen  *GetFrPen() { return &m_FrPen; }

	void  ReDrawGraph();                              // グラフの再描画
	void  ReDrawGraph(int, int);
	void  ReDrawGraph(float, float, int);
	void  ReMakeFont();                               // フォントの作成
	void  SetFontInfo(int, char*, char*, char*);      // フォント情報の設定
	void  SetAspectR(float r) { m_Aspect = r; }
	void  SetDotR(float r) { m_Dot = r; }
	void  SetOri(int);
	void  SetCxy(int cx, int cy) { m_cx = cx; m_cy = cy; }
	void  WriteIniFile();                             // 初期化ファイルに書き込み

	static void PrtMessage(char *, int mode=0);

private:
	void  GetIniFile();
	BOOL  LoadFile();

// オーバーライド
	// ClassWizard は仮想関数のオーバーライドを生成します。
	//{{AFX_VIRTUAL(CPLOTApp)
	public:
	virtual BOOL InitInstance();
	virtual CDocument* OpenDocumentFile(LPCTSTR lpszFileName);
	//}}AFX_VIRTUAL
```

```
private:
	char  *m_Path;			// カレントパス名
	char  *m_IniFname;		// 初期化ファイル名
	char  *m_DataFname;		// データファイル名
	char  *m_BodyFname;		// ボディファイル名
	char  *m_HeadFname;		// ヘッダーファイル名
	char  *m_PsFname;		// ヘッダーファイル名
	char  *m_EpsFname;		// ヘッダーファイル名
	char  *m_HelpFname;		// ヘルプファイル名
	char  *m_FontName[4];		// フォント名（タイトル、ラベル、目盛り等）
	char  *m_FontSize[4];		// フォントサイズ
	char  *m_FontStyle[4];		// フォントスタイル
	int   m_cx, m_cy;		// 画面幅、高さ
	int   m_x, m_y;			// 画面左上位置
	int   m_Ori;			// 表示の向き
	int   m_nFontStyle[4];		// 漢数柔荷番号（0:標準 1:涙為爪 2:伽惆3:涙為爪伽惆
	float m_Aspect;
	float m_Dot;
	BOOL  m_bWin98;			// Win98(TRUE) or WinNT(FALSE)
	DATA  *m_pData;
	CFont m_Font[5];
	CPen  m_FrPen;				// 前景用ペン（黒）

	BOOL  AnaArgs(char*);			// コマンドオプションを解析
	void  CreateFont();
	void  DeleteFont();
	void  SetFileName(char*, char*);
	void  SetGeometry(char*);

// インプリメンテーション
	//{{AFX_MSG(CPLOTApp)
	afx_msg void OnAppAbout();
	afx_msg void OnFileSavePS();
	afx_msg void OnFileOpen();
	afx_msg void OnFileOpenHead();
	afx_msg void OnProp();
	afx_msg void OnHelpPlot();
	afx_msg void OnUpdateHelpPlot(CCmdUI* pCmdUI);
	afx_msg void OnUpdateFilePrint(CCmdUI* pCmdUI);
	afx_msg void OnUpdateFilePrintPreview(CCmdUI* pCmdUI);
	afx_msg void OnUpdateFilePrintSetup(CCmdUI* pCmdUI);
	afx_msg void OnUpdateFileSavePs(CCmdUI* pCmdUI);
	afx_msg void OnUpdateProp(CCmdUI* pCmdUI);
	//}}AFX_MSG
	DECLARE_MESSAGE_MAP()
};

/////////////////////////////////////////////////////////////////////////////

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ は前行の直前に追加の宣言を挿入します。

#endif // !defined(AFX_PLOT_H__1B91EA85_65AE_11D3_ABB3_00C04F8862AB__INCLUDED_)
```

```
/*
 * ACR PLOT TOOL Version 1.0
 *    (c) Copyright 1999
 *    ATR Adaptive Communications Research Laboratories
 *    All Rights Reserved
 */

/*
 * plot.h - プロット情報定義ファイル
 */
#ifndef __plotdef_h__
#define __plotdef_h__
#include <stdio.h>

#ifndef M_PI
#define M_PI                    (3.14)
#endif

#define  FNAME_LEN          300         // 最大ファイル名長（パス含む）
#define  PATH_LEN           250         // 最大パス名長
#define  FNFACE_LEN          50         // 最大フォント名長
#define  FNSIZE_LEN          10         // 最大フォントサイズ長
#define  FNSTYLE_LEN         30         // 最大フォントスタイル長

#define Number                  float           /* 数値演算の精度 */

/* 用紙方向の定数 */
#define PAPER_LANDSCAPE         0               /* 横置き */
#define PAPER_PORTRAIT          1               /* 縦置き */

/* モード */
#define NODRAW_MODE         (-1)                /* 非表示モード */
#define WIN_DRAW_MODE           0               /* Window描画モード */
#define PS_ONLY_MODE            1               /* PS出力モード */
#define EPS_ONLY_MODE               2                       /* EPS出力モード */

#define MAX_TITLE_LEN           256             /* タイトルの最大長さ */
#define MAX_HEADER_LEN          256             /* ヘッダの最大長さ */

#define MAX_GRAPH_COUNT             100                 /* グラフの最大数 */
#define MAX_COLUMN_COUNT        100             /* 列の最大数 */
#define STOP_CHAR               '@'             /* 読み込み停止文字 */
#define STOP_STRING             "@@@@@"         /* 読み込み停止文字列 */
#define COMMENT_CHAR            '!'             /* コメント文字 */
#define MAX_BUFFER_LEN          4096            /* 行バッファの最大長さ */
#define HEADER_TITLE            "タイトル："
#define ulong                   unsigned long   /* 正長整数 */

/* デフォルト値の定義 */
#define DEFAULT_FILENAME        "fort.99"
#define DEFAULT_MAIN_WINDOW_WIDTH  (290*3)      /* 初期ウィンドウの幅 */
#define DEFAULT_MAIN_WINDOW_HEIGHT (200*3)      /* 初期ウィンドウの高さ */
#define DEFAULT_TITLE_WINDOW_HEIGHT 0.06        /* タイトルウィンドウの大きさ(相対値) */
#define DEFAULT_PAPER_ORIENTATION (PAPER_LANDSCAPE) /* 用紙の向き */

/* グラフウィンドウ数 */
#define MAX_WINDOW_COUNT        24              /* 重ね合わせも含めて、表示されるグラフの
数 */
/* オプション指定 */
#define MAX_OPTION_LEN          1024            /* オプション指定の最大長さ */
#define LABEL_OPTION            "label"         /* ラベル列指定 */
#define LABEL_OPT_SHORT         "l"             /* ラベル列指定（短縮形）*/
#define DOT_SIZE_OPTION         "dotsize"       /* 線幅、点の大きさの指定 */
#define DOT_SIZE_OPT_SHORT      "ds"            /* 線幅、点の大きさの指定（短縮形） */
#define DOT_COLOR_OPTION        "dotcolor"      /* グラフの色の指定 */
#define DOT_COLOR_OPT_SHORT     "dc"            /* グラフの色の指定（短縮形） */
#define LABEL_SIZE_OPTION       "labelsize"     /* ラベル文字列の大きさの指定 */
#define LABEL_SIZE_OPT_SHORT    "ls"            /* ラベル文字列の大きさの指定（短縮形） */

#define DUPLICATE_OPTION        "duplicate"     /* 直前のグラフ上への描画指定 */
#define DUPLICATE_OPT_SHORT     "d"             /* 直前のグラフ上への描画指定（短縮形） */

#define LABEL_DIST_OPTION       "labeldistance" /* ラベル文字列描画位置指定 */
```

```
#define LABEL_DIST_OPT_SHORT    "ld"            /* ラベル文字列描画位置指定（短縮形） */
#define DUPLICATE_ON            1               /* 直前のグラフに描画 */
#define DUPLICATE_OFF           0               /* 直前のグラフに描画しない */
/* カラーテーブル */
#define MAX_COLOR_TABLE         10              /* カラーテーブルの数 */
#define MAX_COLOR_LEN           60              /* カラー定義の最大長さ */
#define MAX_LABELDATA_LEN       60              /* ラベルデータの文字列の大きさの最大 */

extern char gColorTable[][MAX_COLOR_LEN];

/* プロット情報の定義 */
typedef struct _DATA {
  char title[MAX_TITLE_LEN+1];          /* タイトル */
  int nWindows;                         /* ウィンドウの数 */
  int nGraphs;                          /* グラフの数 */
  int nRows;                            /* データ行数 */
  int nCols;                            /* データ列数 */
  int nValidRows;                       /* 有効列数 */
  int colX[MAX_GRAPH_COUNT];            /* X軸の列番号 */
  int colY[MAX_GRAPH_COUNT];            /* Y軸の列番号 */
  int colLabel[MAX_GRAPH_COUNT];        /* ラベル列の列番号 */
  Number rangeX[MAX_GRAPH_COUNT][2];    /* X軸の範囲 */
  Number rangeY[MAX_GRAPH_COUNT][2];    /* Y軸の範囲 */
  int lineType[MAX_GRAPH_COUNT];        /* グラフの線種 */
  char colHeader[MAX_COLUMN_COUNT][MAX_HEADER_LEN+1];  /* グラフのヘッダ */
  int winX[MAX_GRAPH_COUNT];            /* ウィンドウの左上のx座標 */
  int winY[MAX_GRAPH_COUNT];            /* ウィンドウの左上のy座標 */
  unsigned int winWidth[MAX_GRAPH_COUNT];  /* 各グラフウィンドウの幅 */
  unsigned int winHeight[MAX_GRAPH_COUNT]; /* 各グラフウィンドウの高 */
  Number *table;                        /* データの配列 */
  char *table_label;                    /* ラベルデータの配列 */
  int   dotlabel_fontsize[MAX_GRAPH_COUNT];          /* ラベルデータの
                                                        文字列の大きさ
                                                        ( label_size オプション用) */
  int   dotlabel_dist[MAX_GRAPH_COUNT]; /* ラベルデータ位置
                                                        ( label_distanceオプション用 ) */
  char  dot_color[MAX_GRAPH_COUNT][MAX_COLOR_LEN];       /* グラフの色の指定
                                                        ( dot_color オプション用 ) */
  int   duplicate[MAX_GRAPH_COUNT];     /* duplicateオプション用 */


  float dotratio;                       /* 点の大きさ
                                           ( コマンド引数 -dotratio用)  */
  unsigned int mainWinWidth;            /* メインウィンドウの幅 */
  unsigned int mainWinHeight;           /* メインウィンドウの高 */
  int orientation;                      /* 用紙／画面の方向 */
  int titleWinX;                        /* タイトルウィンドウのx座標 */
  int titleWinY;                        /* タイトルウィンドウのy座標 */
  unsigned int titleWinWidth;           /* タイトルウィンドウの幅 */
  unsigned int titleWinHeight;          /* タイトルウィンドウの高 */
  int mode;                             /* 描画/出力モード */
  float aspectRatio;                    /* 縦横比 */
  float dotSize[MAX_GRAPH_COUNT];       /* 点の大きさ */
} DATA;

DATA *ReadDataFile(char *, char *, char *);
int   ReadLine(FILE *, char *);
int       StringSeparate(char *, char *, int);
int   SetGraphOption(DATA *, int, char *);
int   CheckGraphOption(DATA *, int, char *);
int   StringSeparate(char *, char *, int);
int   SetPlotWindowSize(DATA *);
char *StringIndex(char *, char);
int   fpSetToData(FILE *, DATA *);
void  PsInit();

#endif  /* end of __plot_h__ */
```

```
// PLOTDoc.cpp : CPLOTDoc クラスの動作の定義を行います。
//

#include "stdafx.h"
#include "PLOT.h"

#include "PLOTDoc.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CPLOTDoc

IMPLEMENT_DYNCREATE(CPLOTDoc, CDocument)

BEGIN_MESSAGE_MAP(CPLOTDoc, CDocument)
	//{{AFX_MSG_MAP(CPLOTDoc)
		// メモ - ClassWizard はこの位置にマッピング用のマクロを追加または削除しま
す。
		//		この位置に生成されるコードを編集しないでください。
	//}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CPLOTDoc クラスの構築/消滅

CPLOTDoc::CPLOTDoc()
{
	// TODO: この位置に１度だけ呼ばれる構築用のコードを追加してください。

}

CPLOTDoc::~CPLOTDoc()
{
}

BOOL CPLOTDoc::OnNewDocument()
{
	if (!CDocument::OnNewDocument())
		return FALSE;

	// TODO: この位置に再初期化処理を追加してください。
	// (SDI ドキュメントはこのドキュメントを再利用します。)

	return TRUE;
}



/////////////////////////////////////////////////////////////////////////////
// CPLOTDoc シリアライゼーション

void CPLOTDoc::Serialize(CArchive& ar)
{
	if (ar.IsStoring())
	{
		// TODO: この位置に保存用のコードを追加してください。
	}
	else
	{
	}
}

/////////////////////////////////////////////////////////////////////////////
// CPLOTDoc クラスの診断

#ifdef _DEBUG
void CPLOTDoc::AssertValid() const
{
	CDocument::AssertValid();
}

void CPLOTDoc::Dump(CDumpContext& dc) const
{
	CDocument::Dump(dc);
}
#endif //_DEBUG

/////////////////////////////////////////////////////////////////////////////
// CPLOTDoc コマンド
```

```
// PLOTDoc.h : CPLOTDoc クラスの宣言およびインターフェイスの定義をします。
//
/////////////////////////////////////////////////////////////////////////////

#if !defined(AFX_PLOTDOC_H__1B91EA8B_65AE_11D3_ABB3_00C04F8862AB__INCLUDED_)
#define AFX_PLOTDOC_H__1B91EA8B_65AE_11D3_ABB3_00C04F8862AB__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000


class CPLOTDoc : public CDocument
{
protected: // シリアライズ機能のみから作成します。
        CPLOTDoc();
        DECLARE_DYNCREATE(CPLOTDoc)

// アトリビュート
public:

// オペレーション
public:

//オーバーライド
        // ClassWizard は仮想関数のオーバーライドを生成します。
        //{{AFX_VIRTUAL(CPLOTDoc)
        public:
        virtual BOOL OnNewDocument();
        virtual void Serialize(CArchive& ar);
        //}}AFX_VIRTUAL

// インプリメンテーション
public:
        virtual ~CPLOTDoc();
#ifdef _DEBUG
        virtual void AssertValid() const;
        virtual void Dump(CDumpContext& dc) const;
#endif

protected:

// 生成されたメッセージ マップ関数
protected:
        //{{AFX_MSG(CPLOTDoc)
                // メモ - ClassWizard はこの位置にメンバ関数を追加または削除します。
                //        この位置に生成されるコードを編集しないでください。
        //}}AFX_MSG
        DECLARE_MESSAGE_MAP()
};

/////////////////////////////////////////////////////////////////////////////

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ は前行の直前に追加の宣言を挿入します。

#endif // !defined(AFX_PLOTDOC_H__1B91EA8B_65AE_11D3_ABB3_00C04F8862AB__INCLUDED_)
```

```
// PLOTView.cpp : CPLOTView クラスの動作の定義を行います。
//

#include "stdafx.h"
#include "PLOT.h"

#include "PLOTDoc.h"
#include "PLOTView.h"
#include "DashLine.h"
#include "Line.h"
#include "graph.h"
#include <math.h>

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CPLOTView

IMPLEMENT_DYNCREATE(CPLOTView, CView)

BEGIN_MESSAGE_MAP(CPLOTView, CView)
        //{{AFX_MSG_MAP(CPLOTView)
        ON_WM_SIZE()
        ON_WM_DROPFILES()
        //}}AFX_MSG_MAP
        // 標準印刷コマンド
        ON_COMMAND(ID_FILE_PRINT, CView::OnFilePrint)
        ON_COMMAND(ID_FILE_PRINT_DIRECT, CView::OnFilePrint)
        ON_COMMAND(ID_FILE_PRINT_PREVIEW, CView::OnFilePrintPreview)
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CPLOTView クラスの構築/消滅

CPLOTView::CPLOTView()
{
}

CPLOTView::~CPLOTView()
{
}

BOOL CPLOTView::PreCreateWindow(CREATESTRUCT& cs)
{
        // TODO: この位置で CREATESTRUCT cs を修正して Window クラスまたはスタイルを
        //  修正してください。

        return CView::PreCreateWindow(cs);
}

/////////////////////////////////////////////////////////////////////////////
// CPLOTView クラスの描画

void CPLOTView::OnDraw(CDC* pDC)
{
        CPLOTDoc* pDoc = GetDocument();
        ASSERT_VALID(pDoc);

        DATA *data = ((CPLOTApp*)AfxGetApp())->GetData();
        if (data && data->mode==WIN_DRAW_MODE) {    // データが存在する時
                for (int i=0; i<data->nGraphs; i++) {
                        // グラフの描画
                        DrawGraph(pDC, gGraphNo[i]);
                        // ラベルの描画
                        DrawLabel(pDC, gGraphNo[i]);
                }
                // タイトル文字列の描画
                DrawTitle(pDC, gGraphNo[0], data);
        }
}

void CPLOTView::DrawPathOutline(const COLORREF& c, CDC& dc, int size)
{
        LOGBRUSH  lb;

        lb.lbStyle = BS_SOLID;          // 実線描画
        lb.lbColor = c;                 // 色の指定
        lb.lbHatch = 0;

        CPen cPen(PS_GEOMETRIC|PS_SOLID|PS_JOIN_MITER|PS_ENDCAP_FLAT, size, &lb);
        CPen *pOldPen = dc.SelectObject(&cPen);
        dc.StrokePath();
        dc.SelectObject(pOldPen);
}

int CPLOTView::GetLinePattern(int style, unsigned int type[])
{
        int  ret;

        switch (style) {
        case 1:     // 実線
                type[0] = 1000;
                ret = 1;
                break;
        case 2:     // 点線
                type[0] = type[1] = 3;
                ret = 2;
                break;
        case 3:     // 一点鎖線
                type[0] = 12;
                type[1] = type[2] = type[3] = 3;
                ret = 4;
                break;
        case 4:     // 長点線
                type[0] = 24;
                type[1] = 6;
                ret = 2;
                break;
        case 5:     // 長一点鎖線
                type[0] = 48;
                type[1] = type[2] = type[3] = 3;
                ret = 4;
                break;
        default:
                ret = 0;
        }

        return ret;
}

/////////////////////////////////////////////////////////////////////////////
// CPLOTView クラスの印刷

BOOL CPLOTView::OnPreparePrinting(CPrintInfo* pInfo)
{
        // デフォルトの印刷準備
        if (pInfo->m_bPreview == TRUE) {
                pInfo->m_nNumPreviewPages = 1;
                pInfo->SetMaxPage(1);
#if 0
                if (AfxGetApp()->GetPrinterDeviceDefaults(&pInfo->m_pPD->m_pd)) {
                        LPDEVMODE pDevMode = pInfo->m_pPD->GetDevMode();
                        if (pDevMode) {
                                pDevMode->dmOrientation = 2;  // 1:portrait  2:landscape
                                ::ResetDC(pInfo->m_pPD->m_pd.hDC, pDevMode);
                                ::GlobalUnlock(pInfo->m_pPD->m_pd.hDevMode);
                        }
                }
#endif
        }
        return DoPreparePrinting(pInfo);
}
```

```
void CPLOTView::OnBeginPrinting(CDC* /*pDC*/, CPrintInfo* /*pInfo*/)
{
        // TODO: 印刷前の特別な初期化処理を追加してください。
}

void CPLOTView::OnEndPrinting(CDC* /*pDC*/, CPrintInfo* /*pInfo*/)
{
        // TODO: 印刷後の後処理を追加してください。
}

/////////////////////////////////////////////////////////////////////////////
// CPLOTView クラスの診断

#ifdef _DEBUG
void CPLOTView::AssertValid() const
{
        CView::AssertValid();
}

void CPLOTView::Dump(CDumpContext& dc) const
{
        CView::Dump(dc);
}

CPLOTDoc* CPLOTView::GetDocument() // 非デバッグ バージョンはインラインです。
{
        ASSERT(m_pDocument->IsKindOf(RUNTIME_CLASS(CPLOTDoc)));
        return (CPLOTDoc*)m_pDocument;
}
#endif //_DEBUG

/////////////////////////////////////////////////////////////////////////////
// CPLOTView クラスのメッセージ ハンドラ

void CPLOTView::OnInitialUpdate()
{
        DATA  *pData;

        CView::OnInitialUpdate();

        pData = ((CPLOTApp *)AfxGetApp())->GetData();

}

void CPLOTView::OnPrint(CDC* pDC, CPrintInfo* pInfo)
{
        int  nOldMapMode;
        int  hr, vr;
        int  x, y, cx, cy;

        ((CPLOTApp *)AfxGetApp())->GetGeometry(&x, &y, &cx, &cy);
        hr = pDC->GetDeviceCaps(HORZRES);
        vr = pDC->GetDeviceCaps(VERTRES);
        nOldMapMode = pDC->SetMapMode(MM_ISOTROPIC);
        pDC->SetWindowExt(cx, cy);
        pDC->SetViewportExt(hr, vr);

        CView::OnPrint(pDC, pInfo);
}

int CALLBACK aaa(ENUMLOGFONT* lp, NEWTEXTMETRIC* lp2, int nType, int lParam)
{
        AfxMessageBox(lp->elfLogFont.lfFaceName);
        return TRUE;
}

void CPLOTView::DrawTitle(CDC* pDC, int gr, DATA* pData)
{
        int       x, y;
        GrAttr    *p;
        CFont     *pOldFont;
        CFont     *pFont;
        CPLOTApp *pApp;

        p = gGraph + gr;

        pApp = (CPLOTApp *)AfxGetApp();
        pFont = pApp->GetFont(0);

        pOldFont = pDC->SelectObject(pFont);
        GetCenterPos(pDC, pData->title, 0, 0, pData->titleWinWidth,
                                pData->titleWinHeight, &x, &y);
        pDC->TextOut(x, y, pData->title, strlen(pData->title));
        pDC->SelectObject(pOldFont);
}

int CPLOTView::DrawGraph(CDC* pDC, int gr)
{
        GrAttr    *p;
        int               i;
        POINT     *ps;
        CPen      *pOldPen;
        CFont     *pOldFont;        // 以前のフォント
        CFont     *pNewFont;        // 新しいフォント
        CLine      lin(*pDC);
        CPLOTApp *pApp;

        if (gr<0 || gr>=GR_MAX_GRAPH || !gGraph[gr].on)
                return -1;

        pApp = (CPLOTApp*)AfxGetApp();  // アプリケーションクラスへのポインタ取得

        p = gGraph + gr;

        /* チェック */
        if (p->width<=0 || p->height<=0 || p->datacount<0)
                return -2;

#if 0
        if (p->graph_type == GR_TYPE_METER) {
                return(grDrawMeter(gr));
        }
#endif

        /* 座標属性の計算 */
        if (p->hold == FALSE) grCalcAxisAttributes(gr);

        /* 描画領域の計算 */
        grCalcAxisOrigin(gr);

        /* duplicate がoff(0)の時だけ、グラフ描画領域をクリアする */
        if (p->duplicate == DUPLICATE_OFF) {
                ClearGraph(pDC, gr);
        }

        if (p->hold == FALSE) {
                /* 装飾の描画 */
                if (p->decoration_color) {
//                      grDrawDecoration(gr);
                }
                else{
                        /* duplicate がoff(0)の時だけ、ウィンドウをクリアする */
                        if (p->duplicate == DUPLICATE_OFF){
//                              GrClear(gr);
                        }
                }

                /* ボックス/座標軸の描画 */

                /* duplicateが off(0)の時だけ座標軸、メモリ文字列、
                   X/Y軸ラベルを描画する */
                if (p->duplicate != DUPLICATE_ON){
                        pOldPen = pDC->SelectObject(pApp->GetFrPen());
                        if (p->box) {
                                lin.DrawRectangle(p->Ox, p->Oy, p->Ow, p->Oh);
```

```
                        }
                        else {
                                /* x座標軸 */
                                lin.DrawLine(p->Ox-1, p->Oy-1+p->Oh+2,
                                                        p->Ox-1+p->Ow+2, p->Oy-1+p->Oh+2)
;
                                /* y座標軸 */
                                lin.DrawLine(p->Ox-1, p->Oy-1+p->Oh+2,
                                                p->Ox-1, p->Oy-1);
                                lin.DrawLine(p->Ox-1, p->Oy-1,
                                                (int)(p->Ox-1+p->Aw*GR_TICK_LENGTH/3)
,
                                                p->Oy-1);
                                lin.DrawLine(p->Ox-1+p->Ow+2, p->Oy-1+p->Oh+2,
                                                p->Ox-1+p->Ow+2,
                                                (int)((p->Oy-1+p->Oh+2)-p->Ah*GR_TICK
_LENGTH/3));
                        }
                        pDC->SelectObject(pOldPen);

                        // 背景モードを変更
                        int oldBkMode = pDC->SetBkMode(TRANSPARENT);

                        /* x目盛り文字列の描画 */
                        if (p->num_xtick_marks > 0) {
                                int             x1, y1, x, y;
                                float   rx, dx;

                                rx = p->xrange[1] - p->xrange[0];

                                if (rx == 0) dx = 0;
                                else         dx = p->Ow / rx;

                                // フォントの切り替え
                                pNewFont = pApp->GetFont(2);
                                pOldFont = pDC->SelectObject(pNewFont);
                                for (int i=0; i<p->num_xtick_marks; i++) {
                                        x1 = (int)(p->Ox + (p->xticks[i]-p->xrange[0])*dx)
;
                                        y1 = (int)(p->Oy + p->Oh + p->Ah*GR_POS_XTICKS_MAR
GIN);

                                        GetCenterPos(pDC, p->xtick_marks[i],
                                                                x1, y1, 0, GR_POS_XTICKS_
HEIGHT, &x, &y);
                                        pDC->TextOut(x, y, p->xtick_marks[i],
                                                                strlen(p->xtick_marks[i]))
;
                                }
                                // 以前のフォントに戻す
                                pDC->SelectObject(pOldFont);
                        }

                        /* xスケールの描画 */
                        if (p->xtick_scale) {
                                int  x, y;
                                char str[20], *ptr;

                                x = p->Rx + p->Rw;
                                y = (int)(p->Oy + p->Oh + p->Ah*GR_POS_XTICKS_MARGIN
                                  + GR_POS_XTICKS_HEIGHT);

                                strcpy(str, "x");
                                strcat(str, p->xtick_scale);
                                if ((ptr = strchr(str, '^')) != NULL)
                                  *ptr++ = '\0';

                                // フォントの切り替え
                                pNewFont = pApp->GetFont(2);
                                pOldFont = pDC->SelectObject(pNewFont);
                                GetCenterPos(pDC, str, x, y, -1, 0, &x, &y);
                                pDC->TextOut(x, y+2, str, strlen(str));
                                if (ptr) {
                                        // フォントの切り替え
                                        pNewFont = pApp->GetFont(3);
                                        pDC->SelectObject(pNewFont);
                                        CSize cSz = pDC->GetTextExtent(str, strlen(str));
                                        pDC->TextOut(x+cSz.cx+1, y-cSz.cy+10, ptr, strlen(
ptr));
                                }
                                pDC->SelectObject(pOldFont);
                        }

                        /* y目盛り文字列の描画 */
                        if (p->num_ytick_marks > 0) {
                                int             x1, y1, x, y;
                                float   ry, dy;

                                ry = p->yrange[1] - p->yrange[0];
                                if (ry == 0) dy = 0;
                                else         dy = p->Oh / ry;

                                // フォントの切り替え
                                pNewFont = pApp->GetFont(2);
                                pOldFont = pDC->SelectObject(pNewFont);
                                for (int i=0; i<p->num_ytick_marks; i++) {
                                        x1 = p->Ox - GR_POS_YTICKS_MARGIN;
                                        y1 = (int)(p->Oy + p->Oh - (p->yticks[i]-p->yrange
[0])*dy);
                                        GetCenterPos(pDC, p->ytick_marks[i], x1, y1, -1, 0
, &x, &y);
                                        pDC->TextOut(x, y, p->ytick_marks[i], strlen(p->yt
ick_marks[i]));
                                }
                                // 以前のフォントに戻す
                                pDC->SelectObject(pOldFont);
                        }

                        /* xラベル文字列の描画 */
                        if (p->xlabel && *p->xlabel) {
                                int             x, y;
                                unsigned int  w, h;

                                w = p->Ow;
                                h = (p->Ay + p->Ah) - (p->Ry + p->Rh);
                                // フォントの切り替え
                                pNewFont = pApp->GetFont(1);
                                pOldFont = pDC->SelectObject(pNewFont);
                                GetCenterPos(pDC, p->xlabel, p->Ox, p->Ry+p->Rh,
                                                        w, h, &x, &y);
                                pDC->TextOut(x, y, p->xlabel, strlen(p->xlabel));
                                // 以前のフォントに戻す
                                pDC->SelectObject(pOldFont);
                        }

                        /* yラベル文字列の描画 */
                        if (p->ylabel && *p->ylabel) {
                                int     x, y;
                                unsigned int    w, h;

                                w = p->Rx - p->Ax;
                                h = p->Oh;
                                // フォントの切り替え
                                pNewFont = pApp->GetFont(4);
                                pOldFont = pDC->SelectObject(pNewFont);
                                GetCenterPos(pDC, p->ylabel, 0, 0, h, w, &y, &x);
                                pDC->TextOut(p->Ax+w-x-20, p->Oy+p->Oh-y, p->ylabel, strle
n(p->ylabel));
                                // 以前のフォントに戻す
                                pDC->SelectObject(pOldFont);
                        }

                        /* yスケールの描画 */
                        if (p->ytick_scale) {
                                int  x, y;
                                char str[20], *ptr;
```

```
                                x = p->Ox;
                                y = p->Ry;
                                // フォントの切り替え
                                pNewFont = pApp->GetFont(2);
                                pOldFont = pDC->SelectObject(pNewFont);
                                GetCenterPos(pDC, str, x, y, -1, 0, &x, &y);

                                strcpy(str, "x");
                                strcat(str, p->ytick_scale);
                                if ((ptr = strchr(str, '^')) != NULL) *ptr++ = 0;
                                pDC->TextOut(x, y, str, strlen(str));
                                if (ptr) {
                                        // フォントの切り替え
                                        pNewFont = pApp->GetFont(3);
                                        pDC->SelectObject(pNewFont);
                                        CSize  cSz = pDC->GetTextExtent(str, strlen(str));

                                        pDC->TextOut(x+cSz.cx+1, y-cSz.cy+10, ptr, strlen(
ptr));
                                }
                                // 以前のフォントに戻す
                                pDC->SelectObject(pOldFont);
                        }
                        // 元の背景モードに戻す
                        pDC->SetBkMode(oldBkMode);
                }
        }

        /* duplicateがoff(0)の時だけ、目盛を描画する */
        if (p->duplicate != DUPLICATE_ON) {
                /* x目盛りの描画 */
                if (p->xticks) {
                        RECT    rec[GR_TICK_MAX_ALL_TICKS];
                        int             i, len;
                        float   rx, dx;

                        rx = p->xrange[1] - p->xrange[0];
                        if (rx == 0) dx = 0;
                        else dx = p->Ow/rx;

                        /* Major目盛りの描画 */
                        len = (int)(p->Ah*GR_TICK_LENGTH);
                        if (len < 2) len = 2;
                        for (i=0; i<p->num_xticks; i++) {
                                rec[i].left = rec[i].right = (long)(p->Ox + (p->xticks[i]-
p->xrange[0])*dx);
                                rec[i].top = p->Oy + p->Oh - 1;
                                rec[i].bottom = p->Oy + p->Oh - len - 1;
                        }
                        pOldPen = pDC->SelectObject(pApp->GetFrPen());
                        lin.DrawSegments(rec, p->num_xticks);

                        /* Minor目盛りの描画 */
                        len = (int)(p->Ah*GR_TICK_MINOR_LENGTH);
                        if (len < 1) len = 1;
                        if (p->num_xticks_minor > 0) {
                                for (i=0; i<p->num_xticks_minor; i++) {
                                        rec[i].left = rec[i].right = (int)(p->Ox +
                                          (p->xticks_minor[i]-p->xrange[0])*dx);
                                        rec[i].top = p->Oy + p->Oh - 1;
                                        rec[i].bottom = p->Oy + p->Oh - len - 1;
                                }
                                lin.DrawSegments(rec, p->num_xticks_minor);
                        }
                        pDC->SelectObject(pOldPen);
                }

                /* y目盛りの描画 */
                if (p->yticks) {
                        RECT    rec[GR_TICK_MAX_ALL_TICKS];
                        int             i, len;
                        float           ry, dy;
```

```
                        ry = p->yrange[1] - p->yrange[0];
                        if (ry == 0) dy = 0;
                        else dy = p->Oh/ry;

                        /* Major目盛りの描画 */
                        len = (int)(p->Aw*GR_TICK_LENGTH);
                        if (len < 2) len = 2;
                        for (i=0; i<p->num_yticks; i++) {
                                rec[i].top = rec[i].bottom = (long)(p->Oy + p->Oh
                                  - (p->yticks[i]-p->yrange[0])*dy);
                                rec[i].left = p->Ox + 1;
                                rec[i].right = p->Ox + len + 1;
                        }
                        pOldPen = pDC->SelectObject(pApp->GetFrPen());
                        lin.DrawSegments(rec, p->num_yticks);

                        /* Minor目盛りの描画 */
                        len = (int)(p->Aw*GR_TICK_MINOR_LENGTH);
                        if (len < 1) len = 1;
                        if (p->num_yticks_minor > 0) {
                                for (i=0; i<p->num_yticks_minor; i++) {
                                        rec[i].top = rec[i].bottom = (long)(p->Oy + p->Oh

                                          - (p->yticks_minor[i]-p->yrange[0])*dy);
                                        rec[i].left = p->Ox + 1;
                                        rec[i].right = p->Ox + len + 1;
                                }
                                lin.DrawSegments(rec, p->num_yticks_minor);
                        }
                        pDC->SelectObject(pOldPen);
                }
        }
        /* データの描画 */
        if (p->ydata) {
                Number  x, y;

                /* データの相対座標から絶対座標への変換 */
                ps = new POINT[p->datacount];
                for (i=0; i<p->datacount; i++) {
                        x = (p->Ox + (p->xdata[i]-p->xrange[0])
                                        /(p->xrange[1]-p->xrange[0])*p->Ow);
                        y = (p->Oy + p->Oh - (p->ydata[i]-p->yrange[0])
                                        /(p->yrange[1]-p->yrange[0])*p->Oh);
                        ps[i].x = (int)(x+0.5);
                        ps[i].y = (int)(y+0.5);
                }

                if (pDC->IsPrinting() == 0) {  // 印刷中でない時
                        /* プロット領域をクリップする */
                        CRgn cRgn;
                        cRgn.CreateRectRgn(p->Ox-1, p->Oy-1, p->Ox+p->Ow+1, p->Oy+p->Oh+1)
;
                        pDC->SelectClipRgn(&cRgn);
                }

                /*
                 * 線種による描きわけ
                 */
                /* 直線 */
                if (!strcmp(p->line_style, GR_LINE_STYLE_LINE)) {
                        CPen      pen;
                        LOGBRUSH  lb;

                        lb.lbStyle = BS_SOLID;          // 実線描画
                        lb.lbColor = p->line_col;       // 色の指定
                        lb.lbHatch = 0;
                        pen.CreatePen(PS_GEOMETRIC|PS_SOLID|PS_ENDCAP_FLAT|PS_JOIN_MITER,

                                                (int)p->dot_size, &lb);
                        pOldPen = pDC->SelectObject(&pen);
                        lin.DrawLines(ps, p->datacount);
                        pDC->SelectObject(pOldPen);
                }
```

```
		/* 点 */
		else if (!strcmp(p->line_style, GR_LINE_STYLE_DOT)) {
			float d = p->dot_size/2;
			float dot_size = p->dot_size;
			if (p->dot_size >= 1.5) {
				CPen	cPn(PS_GEOMETRIC|PS_SOLID, 1, p->line_col);
				CPen	*pOldPn;
				CBrush cBr(p->line_col);
				CBrush *pOldBr;
				// ペンとブラシの切り替え
				pOldPn = pDC->SelectObject(&cPn);
				pOldBr = pDC->SelectObject(&cBr);
				if (p->dot_size < 2.0) {
					dot_size = 2.0f;
					d = 1.0f;
				}
				for (int i=0; i<p->datacount; i++) {
					pDC->Ellipse((int)(ps[i].x-d), (int)(ps[i].y-d),
								(int)(ps[i].x-d+dot_size)
,
								(int)(ps[i].y-d+dot_size)
);
				}
				// 元のペンとブラシに戻す
				pDC->SelectObject(pOldPn);
				pDC->SelectObject(pOldBr);
			}
			else {
				// 背景色を取得
				int bkcol = pDC->GetBkColor();
				for (int i=0; i<p->datacount; i++) {
					pDC->FillSolidRect((int)ps[i].x, (int)ps[i].y, 1,
1, p->line_col);
				}
				// 背景色を元に戻す
				pDC->SetBkColor(bkcol);
			}
		}
		/* 点線 */
		/* 点線をオプション指定に沿って描画する */
		else {
			int		cnt;
			unsigned int type[4];

			/* 点線をオプション指定に沿って描画する */
			if(!strcmp(p->line_style, GR_LINE_STYLE_DASH)){
				// 通常の点線
				cnt = GetLinePattern(2, type);
			}
			else if(!strcmp(p->line_style, GR_LINE_STYLE_DOTTEDLINE)){
				// 通常の１点鎖線
				cnt = GetLinePattern(3, type);
			}
			else if(!strcmp(p->line_style, GR_LINE_STYLE_DASH_LONG)){
				// 長めの点線
				cnt = GetLinePattern(4, type);
			}
			else if(!strcmp(p->line_style, GR_LINE_STYLE_DOTTEDLINE_LONG)){
				// 長めの１点鎖線
				cnt = GetLinePattern(5, type);
			}
			CDashLine Dline(*pDC, type, cnt);
			pDC->BeginPath();
			Dline.Polyline(ps, p->datacount);
			pDC->EndPath();
			DrawPathOutline(p->line_col, *pDC, (int)p->dot_size);
		}

		pDC->SelectClipRgn(NULL);
		Free(ps);
	}

	/* 次のグラフが上書きに指定されているなら、それを呼び出す */
	if (p->duplicate == DUPLICATE_ON) {
		DrawGraph(pDC, gr+1);
	}

	return 0;
}

int CPLOTView::ClearGraph(CDC* pDC, int gr)
{
	GrAttr	*p;

	if (gr<0 || gr>=GR_MAX_GRAPH || !gGraph[gr].on)
		return -1;

	p = gGraph + gr;

	pDC->FillSolidRect(p->Ox, p->Oy, p->Ow, p->Oh, pDC->GetBkColor());
	return 0;
}

int CPLOTView::GetCenterPos(CDC* pDC, char *str, int x, int y,
				int w, int h, int *xx, int *yy)
{
	TEXTMETRIC	tm;
	CSize cSz = pDC->GetTextExtent(str, strlen(str));

	pDC->GetTextMetrics(&tm);

	/* 右寄せ */
	if (w < 0) *xx = x - cSz.cx;
	else	   *xx = x + (w - cSz.cx) / 2;
//	*yy = y + (h - cSz.cy) / 2 + tm.tmAscent;
	*yy = y + (h - cSz.cy) / 2;

	return 0;
}

/* 各点のラベル描画関数 */
int CPLOTView::DrawDotLabel(CDC* pDC, int gr, POINT *ps)
{
	GrAttr	*p;
	char	label[MAX_LABELDATA_LEN];
	char	baselabel[MAX_LABELDATA_LEN];
	int		dist, arg;
	int		xdist, ydist;
	int		wh0, wh1, ww0, ww1;
	int	x, y;
	int		arg_flg;
	int	oldBkMode;
	CFont	*pOldFont;

	p = gGraph + gr;
	dist = p->dotlabel_dist;
	if (dist < 0) dist = 0;

	/* 文字列を描く限界 */
	ww0 = p->Ax + 1;
	ww1 = p->Ax + p->Aw - 1;
	wh0 = p->Ay + p->Ah - 1;
	wh1 = p->Ay + 1;

	// 使用するフォントを変更
	pOldFont = pDC->SelectObject(&p->dotlabel_font);
	// 背景モードを変更
	oldBkMode = pDC->SetBkMode(TRANSPARENT);

	for (int i=0; i<p->datacount; i++) {
		memset(label, 0, sizeof(label));
		arg = 0;
		arg_flg = 0;
		/* ラベルがデータ列指定の時 */
		if (p->labeldata_fp) {
			sprintf(label, "%.2f", *(p->labeldata_fp+i));
```

```
                }
                else if(p->labeldata_cp) {
                        /* ラベルがラベル列指定の時 */
                        strcpy(baselabel, p->labeldata_cp+i*MAX_LABELDATA_LEN);
                        if (strcmp(baselabel, "@x" ) == 0){
                                /* 予約語@x指定の時  (x座標値をラベルとする) */
                                sprintf(label, "%.2f", p->xdata[i]);
                        }
                        else if (strcmp(baselabel, "@y") == 0) {
                                /* 予約語@y指定の時  (y座標値をラベルとする) */
                                sprintf(label, "%.2f", p->ydata[i]);
                        }
                        else if (strcmp(baselabel, "@0") == 0) {
                                /* 予約語@0指定の時、ラベルなし */
                                strcpy(label, "");
                        }
                        else if (baselabel[0] == '@') {
                                /* 予約語@000LABELの時 */
                                arg_flg = 1;
                                sscanf(baselabel, "@%d%s", &arg, label);
                                if (dist == 0) dist = 1;
                                arg = arg % 360;
                                TEXTMETRIC  tm;
                                pDC->GetTextMetrics(&tm);
                                CSize cSz = pDC->GetTextExtent(label, strlen(label));
                                GetLabelPosition(dist, arg, cSz.cx, cSz.cy-tm.tmHeight, &x
dist, &ydist);
                        }
                        else {
                                strcpy(label, baselabel);
                        }
                }
                if (strlen(label) > 0) {
                        if (arg_flg == 0) {
                                TEXTMETRIC  tm;
                                pDC->GetTextMetrics(&tm);
                                CSize cSz = pDC->GetTextExtent(label, strlen(label));
                                GetLabelPosition(dist, 90, cSz.cx, cSz.cy-tm.tmHeight, &xd
ist, &ydist);
                        }
                        x = ps[i].x + xdist;
                        y = ps[i].y + ydist;
                        pDC->TextOut(x, y, label, strlen(label));
                }
        }

        // 背景モードを元に戻す
        pDC->SetBkMode(oldBkMode);
        // フォントを元に戻す
        pDC->SelectObject(pOldFont);

        return 0;
}

int CPLOTView::DrawLabel(CDC* pDC, int gr)
{
        GrAttr  *p;
        int             i;
        POINT   *ps;
        float   x, y;

        p = gGraph + gr;

        /* データの相対座標から絶対座標への変換 */.
        ps = new POINT[p->datacount];
        for (i=0; i<p->datacount; i++) {
                x = (p->Ox + (p->xdata[i]-p->xrange[0])
                        /(p->xrange[1]-p->xrange[0])*p->Ow);
                y = (p->Oy + p->Oh - (p->ydata[i]-p->yrange[0])
                        /(p->yrange[1]-p->yrange[0])*p->Oh);
                ps[i].x = (int)(x+0.5);
                ps[i].y = (int)(y+0.5);
        }
```

```
        /* 各点にラベルを描く */
        if (p->labeldata_fp || p->labeldata_cp) {
                DrawDotLabel(pDC, gr, ps);
        }

        Free(ps);

        /* 次のグラフが上書きに指定されているなら、それを呼び出す */
        if (p->duplicate == DUPLICATE_ON){
                DrawLabel(pDC, gr+1);
        }

        return 0;
}

void CPLOTView::GetLabelPosition(int dist, int arg, int width, int height,
                int *xdist, int *ydist)
{
        if (arg == 0) {
                *xdist = dist;
                *ydist = 0;
        }
        else if (0<arg && arg<90) {
                *xdist = (int)(dist*cosh(arg/180.0*M_PI)+0.5);
                *ydist = -(int)(dist*sinh(arg/180.0*M_PI)+0.5);
        }
        else if (arg == 90) {
                *xdist = 0;
                *ydist = -dist;
        }
        else if (90<arg && arg<180) {
                *xdist = (int)(dist*cos(arg/180.0*M_PI)-0.5);
                *ydist = -(int)(dist*sin(arg/180.0*M_PI)+0.5);
        }
        else if (arg == 180) {
                *xdist = -dist;
                *ydist = 0;
        }
        else if (180<arg && arg<270) {
                *xdist = (int)(dist*cos(arg/180.0*M_PI)-0.5);
                *ydist = -(int)(dist*sin(arg/180.0*M_PI)-0.5);
        }
        else if (arg == 270){
                *xdist = 0;
                *ydist = +dist;
        }
        else if (270<arg && arg<360) {
                *xdist = (int)(dist*cos(arg/180.0*M_PI)+0.5);
                *ydist = -(int)(dist*sin(arg/180.0*M_PI)-0.5);
        }

        *xdist -= width/2;
        *ydist += height/2;
}


void CPLOTView::OnSize(UINT nType, int cx, int cy)
{
        CView::OnSize(nType, cx, cy);

        ((CPLOTApp*)AfxGetApp())->ReDrawGraph(cx, cy);
}

void CPLOTView::OnDropFiles(HDROP hDropInfo)
{

        CView::OnDropFiles(hDropInfo);
}
```

```
// PLOTView.h : CPLOTView クラスの宣言およびインターフェイスの定義をします。
//
/////////////////////////////////////////////////////////////////////////////

#if !defined(AFX_PLOTVIEW_H__1B91EA8D_65AE_11D3_ABB3_00C04F8862AB__INCLUDED_)
#define AFX_PLOTVIEW_H__1B91EA8D_65AE_11D3_ABB3_00C04F8862AB__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000


class CPLOTView : public CView
{
protected: // シリアライズ機能のみから作成します。
        CPLOTView();
        DECLARE_DYNCREATE(CPLOTView)

// アトリビュート
public:
        CPLOTDoc* GetDocument();

private:

// オペレーション
public:
        void DrawPathOutline(const COLORREF& c, CDC& dc, int);
        void DrawTitle(CDC*, int, DATA*);
        void GetLabelPosition(int, int, int, int, int*, int*);
        int  GetLinePattern(int, unsigned int*);
        int  DrawGraph(CDC*, int);
        int  DrawLabel(CDC*, int);
        int  DrawDotLabel(CDC*, int, POINT*);
        int  ClearGraph(CDC*, int);
        int  GetCenterPos(CDC*, char*, int, int, int, int, int*, int*);

// オーバーライド
        // ClassWizard は仮想関数のオーバーライドを生成します。
        //{{AFX_VIRTUAL(CPLOTView)
        public:
        virtual void OnDraw(CDC* pDC);  // このビューを描画する際にオーバーライドされます
。
        virtual BOOL PreCreateWindow(CREATESTRUCT& cs);
        virtual void OnInitialUpdate();
        protected:
        virtual BOOL OnPreparePrinting(CPrintInfo* pInfo);
        virtual void OnBeginPrinting(CDC* pDC, CPrintInfo* pInfo);
        virtual void OnEndPrinting(CDC* pDC, CPrintInfo* pInfo);
        virtual void OnPrint(CDC* pDC, CPrintInfo* pInfo);
        //}}AFX_VIRTUAL

// インプリメンテーション
public:
        virtual ~CPLOTView();
#ifdef _DEBUG
        virtual void AssertValid() const;
        virtual void Dump(CDumpContext& dc) const;
#endif

protected:

// 生成されたメッセージ マップ関数
protected:
        //{{AFX_MSG(CPLOTView)
        afx_msg void OnSize(UINT nType, int cx, int cy);
        afx_msg void OnDropFiles(HDROP hDropInfo);
        //}}AFX_MSG
        DECLARE_MESSAGE_MAP()
};

#ifndef _DEBUG  // PLOTView.cpp ファイルがデバッグ環境の時使用されます。
inline CPLOTDoc* CPLOTView::GetDocument()
   { return (CPLOTDoc*)m_pDocument; }
```

```
#endif

/////////////////////////////////////////////////////////////////////////////

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ は前行の直前に追加の宣言を挿入します。

#endif // !defined(AFX_PLOTVIEW_H__1B91EA8D_65AE_11D3_ABB3_00C04F8862AB__INCLUDED_)
```

```
/*
 * ACR PLOT TOOL Version 1.0
 *               Version 2.0
 *    (c) Copyright 1997,1998
 *    ATR Adaptive Communications Research Laboratories
 *    All Rights Reserved
 */

/*
 * print.c -
 */
#include "stdafx.h"
#include <stdio.h>
#include <math.h>
#include <string.h>

#define EXTRN

#include "plotdef.h"
#include "graph.h"
extern char *psGetRgbColor( char* );
int grPSSetFont(FILE*, char*, char*, char*, int);
int psDrawDotLabel(FILE *, Graph);
int psLoadFontSize(char *, char *, char *, char *, int *, int *, int *);

/*      8       16      24      32      40      48      56                    */
/*------+-------+-------+-------+-------+-------+-------+--------------------*/
/*****************************************************************************
 *        Name: GrPrint
 *    Function: グラフのPS出力
 *    Argument:
 *              Graph gr        I       グラフハンドル
 *              FILE out        I       出力ファイルハンドル
 *      Return:
 *              0  成功
 *             -1  失敗
 * Description:
 *             指定されたファイルへPS形式でグラフの出力を行なう。
 *
 *****************************************************************************/
int GrPrint(Graph gr, FILE *out)
{
        GrAttr  *p;
        int             i;

        if (gr<0 || gr>=GR_MAX_GRAPH || !gGraph[gr].on)
                return -1;

        p = gGraph + gr;

        /* チェック */
        if (p->width<=0 || p->height<=0 || p->datacount<0)
                return -2;

        if (p->graph_type == GR_TYPE_METER) {
                /* not implemented */
        }

        /* 座標属性の計算 */
        if (p->hold == FALSE) {
                grCalcAxisAttributes(gr);
        }

        /* 描画領域の計算 */
        grCalcAxisOrigin(gr);

        /* ボックス/座標軸の描画 */
        if (p->box) {
                fprintf(out, "  %.5g %.5g %.5g %.5g rectangle\n",
                                (float)p->Ox/p->Ww,
                                (float)-p->Oy/p->Wh,
                                (float)(p->Ox + p->Ow)/p->Ww,
                                (float)-(p->Oy + p->Oh)/p->Wh);
        }
        else {
                /* x座標軸 */
                fprintf(out, "  %.5g %.5g %.5g %.5g line\n",
                                (float)p->Ox/p->Ww, (float)-(p->Oy + p->Oh)/p->Wh,
                                (float)(p->Ox + p->Ow)/p->Ww, (float)-(p->Oy + p->Oh)/p->W
h);

                /* y座標軸 */
                fprintf(out, "  %.5g %.5g %.5g %.5g line\n",
                                (float)(p->Ox)/p->Ww, (float)-(p->Oy + p->Oh)/p->Wh,
                                (float)(p->Ox)/p->Ww, (float)-(p->Oy)/p->Wh);

                fprintf(out, "  %.5g %.5g %.5g %.5g line\n",
                                (float)p->Ox/p->Ww,
                                -(float)p->Oy/p->Wh,
                                (float)(p->Ox + p->Aw*GR_TICK_LENGTH/3)/p->Ww,
                                -(float)p->Oy/p->Wh);
                fprintf(out, "  %.5g %.5g %.5g %.5g line\n",
                                (float)(p->Ox + p->Ow)/p->Ww,
                                -(float)(p->Oy + p->Oh)/p->Wh,
                                (float)(p->Ox + p->Ow)/p->Ww,
                                -(float)((p->Oy + p->Oh)-p->Ah*GR_TICK_LENGTH/3)/p->Wh);
        }

        /* x目盛り文字列の描画 */
        if (p->num_xtick_marks > 0) {
                int             i, x1, y1;
                float   rx, dx;

                /* フォントの設定 */
                grPSSetFont(out,
                                        p->fontname,
                                        p->fontbold,
                                        p->fontitalic,
                                        p->fontsize);

                rx = p->xrange[1] - p->xrange[0];
                if (rx == 0) dx = 0;
                else dx = p->Ow/rx;
                for (i = 0;i < p->num_xtick_marks;i++) {
                        x1 = (int)(p->Ox + (p->xticks[i]-p->xrange[0])*dx);
                        y1 = (int)(p->Oy + p->Oh + p->Ah*GR_POS_XTICKS_MARGIN);
                        fprintf(out, "  (%s) %.5g %.5g %.5g %.5g centerText\n",
                                        p->xtick_marks[i],
                                        (float)x1/p->Ww,
                                        -(float)y1/p->Wh,
                                        (float)0,
                                        -(float)GR_POS_XTICKS_HEIGHT/p->Wh);
                }
        }

        /* xスケールの描画 */
        if (p->xtick_scale) {
                int  x, y;
                char str[20], *ptr;

                x = p->Rx + p->Rw;
                y = (int)(p->Oy + p->Oh + p->Ah*GR_POS_XTICKS_MARGIN
                  + GR_POS_XTICKS_HEIGHT);

                strcpy(str, "x");
                strcat(str, p->xtick_scale);
                if ((ptr = strchr(str, '^')) != NULL)
                  *ptr++ = '\0';

                grPSSetFont(out,
                                        p->tick_fontname,
                                        p->tick_fontbold,
                                        p->tick_fontitalic,
                                        p->tick_fontsize);
                fprintf(out, "  (%s) %.5g %.5g rightText\n",
                                str,
```

```
                                (float)x/p->Ww,
                                -(float)y/p->Wh);

                if (ptr) {
                        grPSSetFont(out,
                                                p->ticksub_fontname,
                                                p->ticksub_fontbold,
                                                p->ticksub_fontitalic,
                                                p->ticksub_fontsize);
                        fprintf(out, "  (%s) %.5g %.5g leftText\n",
                                        ptr,
                                        (float)x/p->Ww,
                                        -(float)(y-4)/p->Wh);
                }
        }

        /* y目盛り文字列の描画 */
        if (p->num_ytick_marks > 0) {
                int             i, x1, y1;
                float           ry, dy;

                /* フォントの設定 */
                grPSSetFont(out,
                                        p->fontname,
                                        p->fontbold,
                                        p->fontitalic,
                                        p->fontsize);

                ry = p->yrange[1] - p->yrange[0];
                if (ry == 0) dy = 0;
                else dy = p->Oh/ry;
                for (i = 0;i < p->num_ytick_marks;i++) {
                        x1 = p->Ox - GR_POS_YTICKS_MARGIN;
                        y1 = (int)(p->Oy + p->Oh - (p->yticks[i]-p->yrange[0])*dy);
                        fprintf(out, "  (%s) %.5g %.5g rightText\n",
                                        p->ytick_marks[i],
                                        (float)x1/p->Ww,
                                        -(float)y1/p->Wh);
                }
        }

        /* タイトル文字列の描画 */
        if (p->title && *p->title) {
//              int     x, y;
        }

        /* 重ね表示対応によりタイトル文字列やスケールは１番最初のグラフに合わせる
                (一番最初のグラフのタイトルとグラフ枠のみを表示) */
        if (p->duplicate != DUPLICATE_ON) {
                /* xラベル文字列の描画 */
                if (p->xlabel && *p->xlabel) {
                        unsigned int    w, h;

                        w = p->Ow;
                        h = (p->Ay + p->Ah) - (p->Ry + p->Rh);
                        grPSSetFont(out,
                                                p->label_fontname,
                                                p->label_fontbold,
                                                p->label_fontitalic,
                                                p->label_fontsize);
                        fprintf(out, "  (%s) %.5g %.5g %.5g %.5g centerText\n",
                                        p->xlabel,
                                        (float)p->Ox/p->Ww,
                                        -(float)(p->Ry + p->Rh)/p->Wh,
                                        (float)w/p->Ww,
                                        -(float)h/p->Wh);
                }

                /* yラベル文字列の描画 */
                if (p->ylabel && *p->ylabel) {
                        unsigned int    w, h;

                        w = p->Oh;
                        h = p->Rx - p->Ax;
                        grPSSetFont(out,
                                                p->label_fontname,
                                                p->label_fontbold,
                                                p->label_fontitalic,
                                                p->label_fontsize);
                        fprintf(out, "  (%s) %.5g %.5g %.5g %.5g centerTextRot90\n",
                                        p->ylabel,
                                        (float)p->Rx/p->Ww,
                                        -(float)(p->Oy + p->Oh)/p->Wh,
                                        (float)w/p->Wh,
                                        -(float)h/p->Ww);
                }

                /* yスケールの描画 */
                if (p->ytick_scale) {
                        int  x, y;
                        char str[20], *ptr;

                        x = p->Ox;
                        y = p->Ry;
                        strcpy(str, "x");
                        strcat(str, p->ytick_scale);
                        if ((ptr = strchr(str, '^')) != NULL)
                                *ptr++ = '\0';
                        grPSSetFont(out,
                                                p->tick_fontname,
                                                p->tick_fontbold,
                                                p->tick_fontitalic,
                                                p->tick_fontsize);
                        fprintf(out, "  (%s) %.5g %.5g rightText\n",
                                        str,
                                        (float)x/p->Ww,
                                        -(float)y/p->Wh);
                        if (ptr) {
                                grPSSetFont(out,
                                                        p->ticksub_fontname,
                                                        p->ticksub_fontbold,
                                                        p->ticksub_fontitalic,
                                                        p->ticksub_fontsize);
                                fprintf(out, "  (%s) %.5g %.5g leftText\n",
                                                ptr,
                                                (float)x/p->Ww,
                                                -(float)(y-4)/p->Wh);
                        }
                }

                /* x目盛りの描画 */
                if (p->xticks) {
                        int     i, len;
                        float   rx, dx;

                        rx = p->xrange[1] - p->xrange[0];
                        if (rx == 0) dx = 0;
                        else dx = p->Ow/rx;

                        /* Major目盛りの描画 */
                        len = (int)(p->Ah*GR_TICK_LENGTH);
                        if (len < 2) len = 2;
                        for (i = 0;i < p->num_xticks;i++) {
                                fprintf(out, "  %.5g %.5g %.5g %.5g line\n",
                                                (float)(p->Ox + (p->xticks[i]-p->xrange[0]
)*dx)/p->Ww,
                                                -(float)(p->Oy + p->Oh)/p->Wh,
                                                (float)(p->Ox + (p->xticks[i]-p->xrange[0]
)*dx)/p->Ww,
                                                -(float)(p->Oy + p->Oh - len)/p->Wh);
                        }

                        /* Minor目盛りの描画 */
                        len = (int)(p->Ah*GR_TICK_MINOR_LENGTH);
                        if (len < 1) len = 1;
                        if (p->num_xticks_minor > 0) {
```

```
                                for (i = 0;i < p->num_xticks_minor;i++) {
                                        fprintf(out, "  %.5g %.5g %.5g %.5g line\n",
                                                (float)(p->Ox + (p->xticks_minor[i
]-p->xrange[0])*dx)/p->Ww,
                                                -(float)(p->Oy + p->Oh)/p->Wh,
                                                (float)(p->Ox + (p->xticks_minor[i
]-p->xrange[0])*dx)/p->Ww,
                                                -(float)(p->Oy + p->Oh - len)/p->W
h);
                                }
                        }
                }

                /* y目盛りの描画 */
                if (p->yticks) {
                        int     i, len;
                        float   ry, dy;

                        ry = p->yrange[1] - p->yrange[0];
                        if (ry == 0) dy = 0;
                        else dy = p->Oh/ry;

                        /* Major目盛りの描画 */
                        len = (int)(p->Aw*GR_TICK_LENGTH);
                        if (len < 2) len = 2;
                        for (i = 0;i < p->num_yticks;i++) {
                                fprintf(out, "  %.5g %.5g %.5g %.5g line\n",
                                        (float)(p->Ox)/p->Ww,
                                        -(float)(p->Oy + p->Oh - (p->yticks[i]-p->
yrange[0])*dy)/p->Wh,
                                        (float)(p->Ox + len)/p->Ww,
                                        -(float)(p->Oy + p->Oh - (p->yticks[i]-p->
yrange[0])*dy)/p->Wh);
                        }

                        /* Minor目盛りの描画 */
                        len = (int)(p->Aw*GR_TICK_MINOR_LENGTH);
                        if (len < 1) len = 1;
                        if (p->num_yticks_minor > 0) {
                                for (i = 0;i < p->num_yticks_minor;i++) {
                                        fprintf(out, "  %.5g %.5g %.5g %.5g line\n",
                                                (float)(p->Ox)/p->Ww,
                                                -(float)(p->Oy + p->Oh
                                                        - (p->yticks_mino
r[i]-p->yrange[0])*dy)/p->Wh,
                                                (float)(p->Ox + len)/p->Ww,
                                                -(float)(p->Oy + p->Oh
                                                        - (p->yticks_mino
r[i]-p->yrange[0])*dy)/p->Wh);
                                }
                        }
                }
        }

        /* データの描画 */
        if (p->ydata) {
                Number          x, y, oldx, oldy;
#if 0
                /* データの相対座標から絶対座標への変換 */
                ps = (XPoint *)malloc(sizeof(XPoint)*(p->datacount));
#endif
                fprintf(out, "  %.5g %.5g %.5g %.5g setAxesScale\n",
                        (float)p->Ox/p->Ww, -(float)(p->Oy + p->Oh)/p->Wh,
                        (float)p->Ow/p->Ww, (float)p->Oh/p->Wh);

                /* 描画領域のクリッピングをおこなう (現在はクリッピングしない)  */
                /* fprintf(out, "  %d %d %d %d setRectClip\n", 0, 0, 1, 1 ); */

                /* 線分の色を指定する */
                fprintf(out, "  %s setrgbcolor\n", psGetRgbColor(p->dot_color));

                /* 線分のサイズを指定する */
                fprintf(out, "  %f setlinewidth\n", p->dot_size);
```

```
                /*
                 * 線種による描きわけ
                 */
                /* 直線 */
                if (!strcmp(p->line_style, GR_LINE_STYLE_LINE)) {
                        oldx = (p->xdata[0]-p->xrange[0]) / (p->xrange[1]-p->xrange[0]);
                        oldy = (p->ydata[0]-p->yrange[0]) / (p->yrange[1]-p->yrange[0]);
                        /* 線分の開始点を指定する */
                        fprintf(out, "  %.5g %.5g setStartLine\n", oldx, oldy);
                        for (i=1; i<p->datacount; i++) {
                                x = (p->xdata[i]-p->xrange[0]) / (p->xrange[1]-p->xrange[0
]);
                                y = (p->ydata[i]-p->yrange[0]) / (p->yrange[1]-p->yrange[0
]);
                                /* 線分のポイントを指定する */
                                fprintf(out, "  %.5g %.5g setLineTo\n", x, y);
                        }
                        /* 線分終了を指定する */
                        fprintf(out, "  setEndLine\n");
                }
                /* 点 */
                else if (!strcmp(p->line_style, GR_LINE_STYLE_DOT)) {
                        x = (p->xdata[0]-p->xrange[0]) / (p->xrange[1]-p->xrange[0]);
                        y = (p->ydata[0]-p->yrange[0]) / (p->yrange[1]-p->yrange[0]);
                        /* ドットサイズの半径を指定できるように対応 */
                        fprintf(out, "  %.6g %.6g %.6g dot1\n",
                                        x, y, (float)p->dot_size/2.0);
                        for (i=1; i<p->datacount; i++) {
                                x = (p->xdata[i]-p->xrange[0]) / (p->xrange[1]-p->xrange[0
]);
                                y = (p->ydata[i]-p->yrange[0]) / (p->yrange[1]-p->yrange[0
]);
                                /* ドットサイズの半径を指定できるように対応 */
                                fprintf(out, "  %.5g %.5g %.5g dot1\n",
                                                x, y, (float)p->dot_size/2.0);
                        }
                }
                else if (!strcmp(p->line_style,  GR_LINE_STYLE_DASH)) {
                        /* 線幅可変対応(線幅をdot_sizeの等倍に可変させる場合はコメントをと
る)*/
                        /* if(p->line_width != 0.0 ) */
                        /* fprintf(out, "[%f] 0 setdash\n",
                                p->dot_size*PS_DASH_PATTERN_DASH_LEN); */
                        /* else */
                        /* ラインタイプに点線を指定([2]は２単位分の点線を描く --  --  --..
..) */
                        fprintf(out, "[%f] 0 setdash\n",PS_DASH_PATTERN_DASH_LEN);

                        oldx = (p->xdata[0]-p->xrange[0]) / (p->xrange[1]-p->xrange[0]);
                        oldy = (p->ydata[0]-p->yrange[0]) / (p->yrange[1]-p->yrange[0]);
                        /* 線分の開始点を指定する */
                        fprintf(out, "  %.5g %.5g setStartLine\n", oldx, oldy);

                        for (i=1; i<p->datacount; i++) {
                                x = (p->xdata[i]-p->xrange[0]) / (p->xrange[1]-p->xrange[0
]);
                                y = (p->ydata[i]-p->yrange[0]) / (p->yrange[1]-p->yrange[0
]);
                                /* 線分のポイントを指定する */
                                fprintf(out, "  %.5g %.5g setLineTo\n", x, y);
                        }

                        /* 線分終了を指定する */
                        fprintf(out, "  setEndLine\n");
                }
                /* 点線 */
                else if (!strcmp(p->line_style, GR_LINE_STYLE_DOTTEDLINE)) {
                        /* 線幅可変対応(線幅をdot_sizeの等倍にする場合はコメントをとる) */

                        /* if(p->line_width != 0.0 ) */
                        /* fprintf(out, "[%f 2 2 2] 0 setdash\n",
                                p->dot_size*PS_DASH_PATTERN_DOTTEDLINE_LEN); */
                        /* else */
```

```
                /* ラインタイプに１点鎖線の指定を行う
                   ([12 2 2 2]は１２単位と２単位の鎖線を描く
                   ----------  --  ----------  -- .....) */
                fprintf(out, "[%f 2 2 2] 0 setdash\n",
                                PS_DASH_PATTERN_DOTTEDLINE_LEN );

                oldx = (p->xdata[0]-p->xrange[0]) / (p->xrange[1]-p->xrange[0]);
                oldy = (p->ydata[0]-p->yrange[0]) / (p->yrange[1]-p->yrange[0]);
                /* 線分の開始点を指定する */
                fprintf(out, "  %.5g %.5g setStartLine\n", oldx, oldy);

                for (i=1; i<p->datacount; i++) {
                        x = (p->xdata[i]-p->xrange[0]) / (p->xrange[1]-p->xrange[0
]);
                        y = (p->ydata[i]-p->yrange[0]) / (p->yrange[1]-p->yrange[0
]);
                        /* 線分のポイントを指定する */
                        fprintf(out, "  %.5g %.5g setLineTo\n", x, y);
                }

                /* 線分終了を指定する */
                fprintf(out, "  setEndLine\n");
        }
        else if (!strcmp(p->line_style, GR_LINE_STYLE_DASH_LONG)) {
                /* 線幅可変対応(線幅をdot_sizeの等倍にする場合はコメントをとる) */

                /* if(p->line_width != 0.0 ) */
                /* fprintf(out, "[%f 2] 0 setdash\n",
                        p->dot_size*PS_DASH_PATTERN_DASH_LONG_LEN); */
                /* else */
                /* ラインタイプに長点線の指定を行う([20 2]は２０単位の長点線を描く

                    ----------  ----------  .....) */
                fprintf(out, "[%f 2] 0 setdash\n",PS_DASH_PATTERN_DASH_LONG_LEN);

                oldx = (p->xdata[0]-p->xrange[0]) / (p->xrange[1]-p->xrange[0]);
                oldy = (p->ydata[0]-p->yrange[0]) / (p->yrange[1]-p->yrange[0]);
                /* 線分の開始点を指定する */
                fprintf(out, "  %.5g %.5g setStartLine\n", oldx, oldy);

                for (i=1; i<p->datacount; i++) {
                        x = (p->xdata[i]-p->xrange[0]) / (p->xrange[1]-p->xrange[0
]);
                        y = (p->ydata[i]-p->yrange[0]) / (p->yrange[1]-p->yrange[0
]);
                        /* 線分のポイントを指定する */
                        fprintf(out, "  %.5g %.5g setLineTo\n", x, y);
                }

                /* 線分終了を指定する */
                fprintf(out, "  setEndLine\n");
        }
        else if (!strcmp(p->line_style, GR_LINE_STYLE_DOTTEDLINE_LONG)) {
                /* 線幅可変対応(線幅をdot_sizeの等倍にする場合はコメントをとる) */

                /* if(p->line_width != 0.0 ) */
                /* fprintf(out, "[%f 2 2 2] 0 setdash\n",
                   p->dot_size*PS_DASH_PATTERN_DOTTEDLINE_LONG_LEN); */
                /* else */
                /* ラインタイプに長１点鎖線の指定を行う
                   ([48 2 2 2]は４８単位と２単位の鎖線を描く
                   ----------------  --  ----------------  -- .....) */
                fprintf(out, "[%f 2 2 2] 0 setdash\n",
                                PS_DASH_PATTERN_DOTTEDLINE_LONG_LEN);

                oldx = (p->xdata[0]-p->xrange[0]) / (p->xrange[1]-p->xrange[0]);
                oldy = (p->ydata[0]-p->yrange[0]) / (p->yrange[1]-p->yrange[0]);
                /* 線分の開始点を指定する */
                fprintf(out, "  %.5g %.5g setStartLine\n", oldx, oldy);

                for (i=1; i<p->datacount; i++) {
                        x = (p->xdata[i]-p->xrange[0]) / (p->xrange[1]-p->xrange[0
]);
```

```
                        y = (p->ydata[i]-p->yrange[0]) / (p->yrange[1]-p->yrange[0
]);
                        /* 線分のポイントを指定する */
                        fprintf(out, "  %.5g %.5g setLineTo\n", x, y);
                }

                /* 線分終了を指定する */
                fprintf(out, "  setEndLine\n");
        }

        /* 線分の幅を指定する */
        fprintf(out, "  %f setlinewidth\n",p->dot_size);

        /* 線分の色指定をデフォルトに戻す */
        fprintf(out, "  0 0 0 setrgbcolor\n");

        /* ラベル表示データを作成する */
        psDrawDotLabel(out, gr);

        /* クリップ領域を解除する (現在はクリッピングしない)  */
        /* fprintf(out, "  initclip\n"); */

        fprintf(out, "  stroke\n");
        fprintf(out, "  %.5g %.5g %.5g %.5g restoreAxesScale\n",
                        (float)p->Ox/p->Ww, -(float)(p->Oy + p->Oh)/p->Wh,
                        (float)p->Ow/p->Ww, (float)p->Oh/p->Wh);
#if 0
        Free(ps);
#endif
    }

    return 0;
}

int grPSSetFont(FILE *fp, char *fontName, char *fontBold,
                                char *fontItalic, int fontSize)
{
    char name[40];
    char bold[40];
    char italic[40];

    if (!fontName) return -1;

    strcpy(name, fontName);
    *name = (char)toupper(*name);
    if (fontBold && strcmp(fontBold, "medium")) {
            strcpy(bold, fontBold);
            bold[0] = (char)toupper(bold[0]);
            strcat(name, "-");
            strcat(name, bold);
    }
    else if (fontItalic && strcmp(fontItalic, "roman")) {
            strcpy(italic, fontItalic);
            italic[0] = (char)toupper(italic[0]);
            strcat(name, "-");
            strcat(name, italic);
    }
    fprintf(fp, "/%s %d setFontInfo\n", name, fontSize);
    return 0;
}

int PSDrawLines(Number xdata[], Number ydata[], int datacount,
                                Number xrange[], Number yrange[])
{
    int i;
    Number dx, dy;

    dx = xrange[1] - xrange[0];
    dy = yrange[1] - yrange[0];

    for (i=0; i<datacount; i++) {
            printf("  %lg %lg line\n",
                    (xdata[i]-xrange[0])/dx,
```

```
                        (ydata[i]-yrange[0])/dy);
        }
        return 0;
}

/* ＲＧＢカラー取得関数ＰＳ用 */
char *psGetRgbColor(char *aDotColor)
{
        int             i;
        int             Color;
        float           RColor;
        float           GColor;
        float           BColor;
        static char     Buff[64];

        strcpy(Buff, "0 0 0");

        if (aDotColor == NULL) return Buff;

        if (aDotColor[0] == '#') {
                /* Ｘの表示指定をＰＳの表示指定に変換する */
                Buff[0] = aDotColor[1];
                Buff[1] = aDotColor[2];
                Buff[2] = '\0';
                sscanf( Buff, "%x", &Color );
                RColor = (float)Color;
                RColor = (float)(RColor * ( 1.0 / 255.0 ));

                Buff[0] = aDotColor[3];
                Buff[1] = aDotColor[4];
                Buff[2] = '\0';
                sscanf( Buff, "%x", &Color );
                GColor = (float)Color;
                GColor = (float)(GColor * ( 1.0 / 255.0 ));

                Buff[0] = aDotColor[5];
                Buff[1] = aDotColor[6];
                Buff[2] = '\0';
                sscanf( Buff, "%x", &Color );
                BColor = (float)Color;
                BColor = (float)(BColor * ( 1.0 / 255.0 ));
        }
        else {
                /* ｐｌｏｔ０～９指定をＲＧＢに変換する */
                for (i=0; aDotColor[i]!='\0' && aDotColor[i]!=':'; i++ );

                if (aDotColor[i] != ':') return Buff;

                Buff[0] = aDotColor[i+1];
                Buff[1] = aDotColor[i+2];
                Buff[2] = '\0';
                sscanf( Buff, "%x", &Color );
                RColor = (float)Color;
                RColor = (float)(RColor * ( 1.0 / 255.0 ));

                Buff[0] = aDotColor[i+3];
                Buff[1] = aDotColor[i+4];
                Buff[2] = '\0';
                sscanf( Buff, "%x", &Color );
                GColor = (float)Color;
                GColor = (float)(GColor * ( 1.0 / 255.0 ));

                Buff[0] = aDotColor[i+5];
                Buff[1] = aDotColor[i+6];
                Buff[2] = '\0';
                sscanf(Buff, "%x", &Color);
                BColor = (float)Color;
                BColor = (float)(BColor * (1.0 / 255.0));
        }

        sprintf(Buff, "%f %f %f", RColor, GColor, BColor);

        return Buff;
}

/* 各点のラベル描画関数 */
int psDrawDotLabel(FILE *fp, Graph gr)
{
        GrAttr *p;
        int         i;
        char    label[MAX_LABELDATA_LEN];
        char    baselabel[MAX_LABELDATA_LEN];
        int         dist, arg;
        int         xdist, ydist;
        int         width, height,size;
        float   fx, fy;
        int         arg_flg;

        p = gGraph + gr;
        dist = p->dotlabel_dist;
        if (dist < 0) dist = 0;

        /* ポイント数分ラベルがあれば表示する */
        for (i=0; i<p->datacount; i++) {
                memset(label, 0, sizeof(label));
                arg = 0;
                arg_flg = 0;

                if (p->labeldata_fp) {
                        sprintf( label, "%.2f", *(p->labeldata_fp+i));
                }
                else if (p->labeldata_cp != NULL) {
                        strcpy(baselabel, p->labeldata_cp+i*MAX_LABELDATA_LEN);
                        if (strcmp(baselabel, "@x") == 0) {
                                /* Ｘ座標データを表示ラベルとして使用する */
                                sprintf(label, "%.2f", p->xdata[i]);
                        }
                        else if (strcmp( baselabel, "@y") == 0) {
                                /* Ｙ座標データを表示ラベルとして使用する */
                                sprintf(label, "%.2f", p->ydata[i] );
                        }
                        else if (strcmp( baselabel, "@0") == 0) {
                                /* ラベル無しを指定 */
                                strcpy(label, "" );
                        }
                        else if (baselabel[0] == '@') {
                                /* ラベルにユーザーラベルと表示角度を指定 */
                                arg_flg = 1;
                                sscanf(baselabel, "@%d%s", &arg, label);
                                if (dist == 0) dist = 1;
                                arg = arg%360;

                                /* 各角度でのオフセットを算出する */
                                if (arg == 0) {
                                        xdist = dist;
                                        ydist = 0;
                                }
                                else if (0<arg && arg<90) {
                                        xdist = (int)((float)dist*cos((float)arg/180.0*M_P
I));
                                        ydist = (int)((float)dist*sin((float)arg/180.0*M_P
I));
                                }
                                else if (arg == 90) {
                                        xdist = 0;
                                        ydist = dist;
                                }
                                else if (90<arg && arg<180) {
                                        xdist = (int)((float)dist*cos((float)arg/180.0*M_P
I));
                                        ydist = (int)((float)dist*sin((float)arg/180.0*M_P
I));
                                }
                                else if (arg == 180) {
                                        xdist = -dist;
                                        ydist = 0;
```

```
                        }
                        else if (180<arg && arg<270 ){
                                xdist = (int)((float)dist*cos((float)arg/180.0*M_P
I));
                                ydist = (int)((float)dist*sin((float)arg/180.0*M_P
I));
                        }
                        else if (arg == 270) {
                                xdist = 0;
                                ydist = -dist;
                        }
                        else if (270<arg && arg<360) {
                                xdist = (int)((float)dist*cos((float)arg/180.0*M_P
I));
                                ydist = (int)((float)dist*sin((float)arg/180.0*M_P
I));
                        }
                }
                else{
                        strcpy(label, baselabel);
                }
        }

        /* ラベル表示の指定があればラベルを表示 */
        if (strlen(label) > 0) {
                /* 角度指定がなされていない場合は上に表示 */
                if (arg_flg == 0) {
                        xdist = 0;
                        ydist = dist;
                }

                width = height = 0;
                size = p->dotlabel_fontsize;
                /* フォント情報をロード */
                psLoadFontSize(label, p->dotlabel_fontname,
                                        p->dotlabel_fontbold, p->dotlabel_fonti
talic,
                                        &size, &width, &height);

                /* X座標を算出する */
                fx = (p->xdata[i]-p->xrange[0]) / (p->xrange[1]-p->xrange[0]);
                /* Y座標を算出する */
                fy = (p->ydata[i]-p->yrange[0]) / (p->yrange[1]-p->yrange[0]);

                /* フォント情報を設定する */
                grPSSetFont(fp, p->dotlabel_fontname,
                                        p->dotlabel_fontbold,
                                        p->dotlabel_fontitalic,
                                        p->dotlabel_fontsize);

                /* 各角度に対したラベル表示位置を指定する */
                if (arg==0 && arg_flg==0) {
                        /* ラベルの中心指定 */
                        fprintf(fp,     "(%s) %.5g %.5g %.5g %.5g setText\n",
                                        label, fx, fy, (float)xdist, (float)ydist
);
                }
                else if (arg == 0) {
                        /* ラベル左詰めを指定 */
                        fprintf(fp,     "(%s) %.5g %.5g %.5g %.5g setText\n",
                                        label, fx, fy, (float)xdist, (float)ydist
);
                }
                else if (0<arg && arg<90) {
                        /* ラベル左詰めを指定 */
                        fprintf(fp,     "(%s) %.5g %.5g %.5g %.5g setText\n",
                                        label, fx, fy, (float)xdist, (float)ydist
);
                }
                else if (arg == 90) {
                        /* ラベルの中心指定 */
                        fprintf(fp,     "(%s) %.5g %.5g %.5g %.5g setText\n",
                                        label, fx, fy, (float)xdist, (float)ydist
);
                }
                else if (90<arg && arg<180) {
                        /* ラベル右詰めを指定 */
                        fprintf(fp,     "(%s) %.5g %.5g %.5g %.5g setText\n",
                                        label, fx, fy, (float)xdist, (float)ydist
);
                }
                else if (arg == 180) {
                        /* ラベル右詰めを指定 */
                        fprintf(fp,     "(%s) %.5g %.5g %.5g %.5g setText\n",
                                        label, fx, fy, (float)xdist, (float)ydist
);
                }
                else if (180<arg && arg<270) {
                        /* ラベル右詰めを指定 */
                        fprintf(fp,     "(%s) %.5g %.5g %.5g %.5g setText\n",
                                        label, fx, fy, (float)xdist, (float)ydist
);
                }
                else if (arg == 270) {
                        /* ラベルの中心指定 */
                        fprintf(fp,     "(%s) %.5g %.5g %.5g %.5g setText\n",
                                        label, fx, fy, (float)xdist, (float)ydist
);
                }
                else if (270<arg && arg<360) {
                        /* ラベル左詰めを指定 */
                        fprintf(fp,     "(%s) %.5g %.5g %.5g %.5g setText\n",
                                        label, fx, fy, (float)xdist, (float)ydist
);
                }
        }

        return 0;
}

int psLoadFontSize(char *label, char *name, char *bold, char *italic,
                                int *size, int *width, int *height)
{
        static char     *fns[] = GR_FONT_NAME_RANGE, **p;

        /* フォント名のチェック */
        if (!name)
                name = GR_DEFAULT_FONTNAME;
        else {
                p = fns;
                while (*p) {
                        if (!strcmp(name, *p))
                          goto next;
                        p++;
                }
                fprintf(stderr, "WARNING: Can't find font name \"%s\"\n", name);
                name = GR_DEFAULT_FONTNAME;
        }
  next:
        /* フォントタイプのチェック */
        if (!bold)
          bold = GR_DEFAULT_FONTBOLD;

        /* サイズのチェック */
        if      (*size <=  8) *size =  8;
        else if (*size <= 10) *size = 10;
        else if (*size <= 12) *size = 12;
        else if (*size <= 14) *size = 14;
        else if (*size <= 18) *size = 18;
        else if (*size <= 48) *size = 24;
        else                  *size = GR_DEFAULT_FONTSIZE;

        *width = (*size) * strlen(label);
        *height = (*size);
```

```
        return 0;
}
```

0

```
// PropDlg.cpp : インプリメンテーション ファイル
//

#include "stdafx.h"
#include "PLOT.h"
#include "PropDlg.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CPropDlg ダイアログ


CPropDlg::CPropDlg(CWnd* pParent /*=NULL*/)
        : CDialog(CPropDlg::IDD, pParent)
{
        //{{AFX_DATA_INIT(CPropDlg)
                // メモ - ClassWizard はこの位置にマッピング用のマクロを追加または削除しま
す。
        //}}AFX_DATA_INIT
}

void CPropDlg::DoDataExchange(CDataExchange* pDX)
{
        CDialog::DoDataExchange(pDX);
        //{{AFX_DATA_MAP(CPropDlg)
        DDX_Control(pDX, IDC_BUTTON3, m_TitleFont);
        DDX_Control(pDX, IDOK, m_Ok);
        DDX_Control(pDX, IDOK2, m_Apply);
        DDX_Control(pDX, ID_SAVE_INI, m_Save);
        DDX_Control(pDX, IDC_EDIT1, m_Aspect);
        DDX_Control(pDX, IDC_EDIT3, m_Dot);
        //}}AFX_DATA_MAP

        for (int i=0; i<4; i++) {
                DDX_Control(pDX, IDC_EDIT4+2*i, m_FontName[i]);
                DDX_Control(pDX, IDC_EDIT5+2*i, m_FontSize[i]);
                DDX_Control(pDX, IDC_EDIT12+i, m_FontStyle[i]);
        }
}


BEGIN_MESSAGE_MAP(CPropDlg, CDialog)
        //{{AFX_MSG_MAP(CPropDlg)
        ON_BN_CLICKED(IDC_RADIO1, OnTate)
        ON_BN_CLICKED(IDC_RADIO2, OnYoko)
        ON_EN_UPDATE(IDC_EDIT1, OnUpdateAspect)
        ON_EN_UPDATE(IDC_EDIT3, OnUpdateDot)
        ON_BN_CLICKED(IDOK2, OnApply)
        ON_BN_CLICKED(ID_SAVE_INI, OnSaveIni)
        ON_BN_CLICKED(IDC_BUTTON3, OnTitleFont)
        ON_BN_CLICKED(IDC_BUTTON4, OnLabelFont)
        ON_BN_CLICKED(IDC_BUTTON5, OnMemFont)
        ON_BN_CLICKED(IDC_BUTTON6, OnMemFont2)
        //}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CPropDlg メッセージ ハンドラ

void CPropDlg::OnOK()
{
        UpdateData(TRUE);

        if (m_UpdFlag) {      // 内容が更新されている時
                SetPropaty();
        }

        CDialog::OnOK();
}

void CPropDlg::OnCancel()
{
        CDialog::OnCancel();
}

void CPropDlg::OnTate()
{
        if (m_Ori != PAPER_PORTRAIT) {
                m_Ori = PAPER_PORTRAIT;
                m_UpdFlag = 1;

                // 適用ボタンを使用可に
                m_Apply.EnableWindow(TRUE);
                // 保存ボタンを使用可に
                m_Save.EnableWindow(TRUE);
        }
}

void CPropDlg::OnYoko()
{
        if (m_Ori != PAPER_LANDSCAPE) {
                m_Ori = PAPER_LANDSCAPE;
                m_UpdFlag = 1;

                // 適用ボタンを使用可に
                m_Apply.EnableWindow(TRUE);
                // 保存ボタンを使用可に
                m_Save.EnableWindow(TRUE);
        }
}

BOOL CPropDlg::OnInitDialog()
{
        CDialog::OnInitDialog();

        char      buf[100];
        char      name[FNFACE_LEN+1];
        char      size[FNSIZE_LEN+1];
        char      style[FNSTYLE_LEN+1];
        float     ratio;
        CPLOTApp *pApp;

        // アプリケーションクラスへのポインタ取得
        pApp = (CPLOTApp *)AfxGetApp();

        // 縦横比の取得
        ratio = pApp->GetAspectR();
        sprintf(buf, "%.3f", ratio);
        m_Aspect.SetWindowText(buf);

        // 線幅の取得
        ratio = pApp->GetDotR();
        sprintf(buf, "%.3f", ratio);
        m_Dot.SetWindowText(buf);

        // 表示の向きを取得
        m_Ori = pApp->GetOri();
        if (m_Ori == PAPER_LANDSCAPE)
                ((CButton*)(GetDlgItem(IDC_RADIO2)))->SetCheck(1);
        else
                ((CButton*)(GetDlgItem(IDC_RADIO1)))->SetCheck(1);

        // 使用中のフォント名、サイズを取得
        for (int i=0; i<4; i++) {
                pApp->GetFontInfo(i, name, size, style);
                m_FontName[i].SetWindowText(name);
                m_FontSize[i].SetWindowText(size);
                m_FontStyle[i].SetWindowText(style);
        }

        // 適用ボタンを使用不可に
```

```
	m_Apply.EnableWindow(FALSE);
	// 保存ボタンを使用不可に
	m_Save.EnableWindow(FALSE);

	// Okボタンにフォーカスを
	m_Ok.SetFocus();

	m_UpdFlag = 0;	// 更新フラグをリセット

	return FALSE;  // コントロールにフォーカスを設定しないとき、戻り値は TRUE となりま
す
			// 例外: OCX プロパティ ページの戻り値は FALSE となります
}

void CPropDlg::OnUpdateAspect()
{
	m_UpdFlag = 1;

	// 適用ボタンを使用可に
	m_Apply.EnableWindow(TRUE);
	// 保存ボタンを使用可に
	m_Save.EnableWindow(TRUE);
}

void CPropDlg::OnUpdateDot()
{
	m_UpdFlag = 1;

	// 適用ボタンを使用可に
	m_Apply.EnableWindow(TRUE);
	// 保存ボタンを使用可に
	m_Save.EnableWindow(TRUE);
}

// 変更プロパティの画面への反映
void CPropDlg::OnApply()
{
	UpdateData(TRUE);

	// 適用ボタンを使用不可に
	m_Apply.EnableWindow(FALSE);

	// プロパティを設定
	SetPropaty();

	// Okボタンにフォーカスを
	m_Ok.SetFocus();
}

// 初期化ファイルへの保存
void CPropDlg::OnSaveIni()
{
	UpdateData(TRUE);

	// 適用ボタンを使用不可に
	m_Apply.EnableWindow(FALSE);
	// 保存ボタンを使用不可に
	m_Save.EnableWindow(FALSE);

	// プロパティを設定
	SetPropaty();

	// 初期化ファイルにプロパティを保存する
	((CPLOTApp*)AfxGetApp())->WriteIniFile();
}

void CPropDlg::SetPropaty()
{
	char  buf[31];
	CPLOTApp *pApp = (CPLOTApp*)AfxGetApp();

	// 縦横比を取得し、設定
	m_Aspect.GetWindowText(buf, 30);
	pApp->SetAspectR((float)atof(buf));

	// 線幅比を取得し、設定
	m_Dot.GetWindowText(buf, 30);
	pApp->SetDotR((float)atof(buf));

	// 表示の向きを取得し、設定
	if (((CButton*)(GetDlgItem(IDC_RADIO1)))->GetCheck() == 1)
		pApp->SetOri(PAPER_PORTRAIT);
	else
		pApp->SetOri(PAPER_LANDSCAPE);

	// フォントを取得し、設定
	char  name[FNFACE_LEN+1];
	char  size[FNSIZE_LEN+1];
	char  style[FNSTYLE_LEN+1];
	for (int i=0; i<4; i++) {
		m_FontName[i].GetWindowText(name, FNFACE_LEN);
		m_FontSize[i].GetWindowText(size, FNSIZE_LEN);
		m_FontStyle[i].GetWindowText(style, FNSTYLE_LEN);
		pApp->SetFontInfo(i, name, size, style);
	}
	// フォントの再作成
	pApp->ReMakeFont();

	// グラフの再描画
	pApp->ReDrawGraph();

	m_UpdFlag = 0;
}

void CPropDlg::OnTitleFont()
{
	SelectFont(0);
	m_TitleFont.SetState(0);
	Invalidate();
}

void CPropDlg::OnLabelFont()
{
	SelectFont(1);
}

void CPropDlg::OnMemFont()
{
	SelectFont(2);
}

void CPropDlg::OnMemFont2()
{
	SelectFont(3);
}

void CPropDlg::SelectFont(int id)
{
	char	style[FNSTYLE_LEN+1];
	LOGFONT	lf;
	CFont	*pFont;

	pFont = ((CPLOTApp*)AfxGetApp())->GetFont(id);
	pFont->GetLogFont(&lf);

	CFontDialog  dlg(&lf, CF_SCREENFONTS|CF_TTONLY|CF_NOVERTFONTS|CF_USESTYLE);

	// フォントスタイルの設定
	m_FontStyle[id].GetWindowText(style, FNSTYLE_LEN);
	dlg.m_cf.lpszStyle = style;

	if (dlg.DoModal() == IDOK) {
		// フォントデータ (FACE名、参殉、柔荷) の取得
		char  buf[FNSIZE_LEN+1];
		CString  csFn = dlg.GetFaceName();
		CString  csSty = dlg.GetStyleName();
```

```
                int  size = dlg.GetSize();

                // ダイアログ画面への表示
                sprintf(buf, "%2d", size/10);
                m_FontName[id].SetWindowText(csFn);
                m_FontSize[id].SetWindowText(buf);
                m_FontStyle[id].SetWindowText(csSty);

                m_UpdFlag = 1;  // 更新フラグをオン

                // 適用ボタンを使用可に
                m_Apply.EnableWindow(TRUE);
                // 保存ボタンを使用可に
                m_Save.EnableWindow(TRUE);
        }

        // Okボタンにフォーカスを
        m_Ok.SetFocus();
}
```

```
#if !defined(AFX_PROPDLG_H__606F0DA2_6F39_11D3_ABB7_00C04F8862AB__INCLUDED_)
#define AFX_PROPDLG_H__606F0DA2_6F39_11D3_ABB7_00C04F8862AB__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000
// PropDlg.h : ヘッダー ファイル
//

/////////////////////////////////////////////////////////////////////////////
// CPropDlg ダイアログ

class CPropDlg : public CDialog
{
// コンストラクション
public:
        CPropDlg(CWnd* pParent = NULL);   // 標準のコンストラクタ

// ダイアログ データ
        //{{AFX_DATA(CPropDlg)
        enum { IDD = IDD_DIALOG2 };
        CButton m_TitleFont;
        CButton m_Ok;
        CButton m_Apply;
        CButton m_Save;
        CEdit   m_Aspect;
        CEdit   m_Dot;
        //}}AFX_DATA

        CEdit   m_FontSize[4];
        CEdit   m_FontName[4];
        CEdit   m_FontStyle[4];
        int  m_UpdFlag;     // 更新フラグ
        int  m_Ori;         // 表示の向き

// オーバーライド
        // ClassWizard は仮想関数のオーバーライドを生成します。
        //{{AFX_VIRTUAL(CPropDlg)
        protected:
        virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV サポート
        //}}AFX_VIRTUAL

// インプリメンテーション
private:
        void SetPropaty();
        void SelectFont(int);

protected:

        // 生成されたメッセージ マップ関数
        //{{AFX_MSG(CPropDlg)
        virtual void OnOK();
        virtual void OnCancel();
        afx_msg void OnTate();
        afx_msg void OnYoko();
        virtual BOOL OnInitDialog();
        afx_msg void OnUpdateAspect();
        afx_msg void OnUpdateDot();
        afx_msg void OnApply();
        afx_msg void OnSaveIni();
        afx_msg void OnTitleFont();
        afx_msg void OnLabelFont();
        afx_msg void OnMemFont();
        afx_msg void OnMemFont2();
        //}}AFX_MSG
        DECLARE_MESSAGE_MAP()
};

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ は前行の直前に追加の宣言を挿入します。

#endif // !defined(AFX_PROPDLG_H__606F0DA2_6F39_11D3_ABB7_00C04F8862AB__INCLUDED_)
```

```
/*
 * ACR PLOT TOOL Version 1.0
 *              Version 2.0
 *    (c) Copyright 1997,1998
 *    ATR Adaptive Communications Research Laboratories
 *    All Rights Reserved
 */

/*
 * prtps.cpp -
 */
#include "stdafx.h"
#include <stdio.h>
#include <stdarg.h>
#include <math.h>

#include "plotdef.h"
#include "graph.h"

int grPSDrawTitleCenter(FILE *, int, char *, float *, float *, float *, float *);

Graph PsOpen(DATA *data, int no)
{
        int             i;                      /* ループカウンタ */
        GrAttr  *p;                     /* カレントグラフ情報 */

        /* 空いているグラフ情報を捜す */
        for (i=0; i<GR_MAX_GRAPH; i++) {
                if (gGraph[i].on == FALSE)
            break;
        }
        if (i >= GR_MAX_GRAPH) return -1;

        p = gGraph+i;
        p->on                   = TRUE;
        p->x                    = (float)GR_POS_DEFAULT_X;
        p->y                    = (float)GR_POS_DEFAULT_Y;
        p->width                = (float)GR_POS_DEFAULT_WIDTH;
        p->height               = (float)GR_POS_DEFAULT_HEIGHT;
        strcpy(p->line_style, GR_DEFAULT_LINE_STYLE);
        p->line_width           = GR_DEFAULT_LINE_WIDTH;
        p->dot_size             = 1.0;
        p->xdata                = NULL;
        p->xdatatype            = GR_FLOAT;
        p->xrange[0]            = 0;
        p->xrange[1]            = -1;
        p->ydata                = NULL;
        p->ydatatype            = GR_FLOAT;
        p->datacount            = 0;
        p->yrange[0]            = 0;
        p->yrange[1]            = -1;
        /* ラベルデータへのポインタ (数値(f)、または文字(c)) */
        p->labeldata_fp         = NULL;
        p->labeldata_cp                 = NULL;
        /* ラベルデータ描画用フォント及びGC設定 */
        p->dotlabel_fontname            = strdup(GR_PS_FONTNAME);
        p->dotlabel_fontbold            = strdup(GR_DEFAULT_FONTBOLD);
        p->dotlabel_fontitalic          = strdup(GR_DEFAULT_FONTITALIC);
        p->dotlabel_fontsize            = GR_PS_FONTSIZE;
        p->dotlabel_color               = NULL;
        /* 線、点描画用GC設定 (初期化)  */
        p->dot_color            = NULL;
        /* label distance */
        p->dotlabel_dist                = 0;
        /* duplicate flag */
        p->duplicate            = DUPLICATE_OFF;
        p->box                  = FALSE;
        p->aspect_ratio         = -1;
        p->xaxis                = TRUE;
        p->yaxis                = TRUE;

        p->fontname             = strdup(GR_PS_FONTNAME);
        p->fontbold             = strdup(GR_DEFAULT_FONTBOLD);
        p->fontitalic           = strdup(GR_DEFAULT_FONTITALIC);
        p->fontsize             = GR_PS_FONTSIZE;
        p->xticks               = NULL;
        p->num_xticks           = 0;
        p->yticks               = NULL;
        p->num_yticks           = 0;
        p->xlabel               = NULL;
        p->ylabel               = NULL;
        p->xtick_marks          = NULL;
        p->num_xtick_marks      = 0;
        p->ytick_marks          = NULL;
        p->num_ytick_marks      = 0;
        p->xticks_minor         = NULL;
        p->num_xticks_minor     = 0;
        p->yticks_minor         = NULL;
        p->num_yticks_minor     = 0;
        p->xtick_scale          = NULL;
        p->ytick_scale          = NULL;
        p->decoration_color     = NULL;
        p->decoration_type      = 1;

        /* タイトルフォント */
        p->title_fontname       = strdup(GR_PS_TITLE_FONTNAME);
        p->title_fontbold       = strdup(GR_DEFAULT_TITLE_FONTBOLD);
        p->title_fontitalic     = strdup(GR_DEFAULT_TITLE_FONTITALIC);
        p->title_fontsize       = GR_PS_TITLE_FONTSIZE;
        p->title_color          = NULL;

        /* ラベルフォント */
        p->label_fontname       = strdup(GR_PS_LABEL_FONTNAME);
        p->label_fontbold       = strdup(GR_DEFAULT_LABEL_FONTBOLD);
        p->label_fontitalic     = strdup(GR_DEFAULT_LABEL_FONTITALIC);
        p->label_fontsize       = GR_PS_LABEL_FONTSIZE;
        p->label_color          = NULL;

        /* 目盛りフォント */
        p->tick_fontname        = strdup(GR_PS_TICK_FONTNAME);
        p->tick_fontbold        = strdup(GR_DEFAULT_TICK_FONTBOLD);
        p->tick_fontitalic      = strdup(GR_DEFAULT_TICK_FONTITALIC);
        p->tick_fontsize        = GR_PS_TICK_FONTSIZE;
        p->tick_color           = NULL;

        /* 目盛り(上付)フォント */
        p->ticksub_fontname     = strdup(GR_PS_TICKSUB_FONTNAME);
        p->ticksub_fontbold     = strdup(GR_DEFAULT_TICKSUB_FONTBOLD);
        p->ticksub_fontitalic   = strdup(GR_DEFAULT_TICKSUB_FONTITALIC);
        p->ticksub_fontsize     = GR_PS_TICKSUB_FONTSIZE;
        p->ticksub_color        = NULL;

        p->draw_mode            = GR_MODE_NORMAL;
        p->draw_size            = 0;
        p->draw_ptr             = 0;
        p->draw_buffer          = NULL;
        p->hold                 = FALSE;
        p->max_exp              = GR_DEFAULT_MAX_EXP;
        p->graph_type           = GR_DEFAULT_GRAPH_TYPE;

        return i;
}

int PrintPsGraph(char *filename, DATA *data)
{
        int i;
        FILE *fp;
        float x, y, width, height;
        float paperWidth, paperHeight;
        float topMargin, bottomMargin, leftMargin, rightMargin;
        float scaleFont;
        float Bwidth, Bheight;

        /* PS出力ファイルのオープン */
        if ((fp = fopen(filename, "w+")) == NULL) {
                fprintf(stderr, "Cannot open file '%s'.\n", filename);
```

```
        return -1;
    }

    /* 各ページパラメータの設定(A4,by inch) */
    paperWidth   = 8.5;
    paperHeight  = 11.7f;
    topMargin    = 0.4f;
    bottomMargin = 0.4f;
    leftMargin   = 0.7f;
    rightMargin  = 0.4f;
    scaleFont    = 0.75f;                    /* フォントの全体的なスケーリング */

    /* ヘッダ出力 */
    if (data->mode == PS_ONLY_MODE) {
        fprintf(fp, "%%!PS-Adobe-1.0\n");
    }
    else {
        fprintf(fp, "%%!PS-Adobe-3.0 EPSF-3.0\n");

        Bwidth = (float)(paperWidth * 72.0)/* + 1.0*/;
        Bheight = (float)(paperHeight * 72.0)/* + 1.0*/;

        fprintf(fp, "%%%%BoundingBox: 5 5 %d %d \n", (int)Bwidth, (int)Bheight);
    }
    fprintf(fp, "%%%% DocumentFonts: Helvetica\n");
    fprintf(fp, "%%%% Title: %s\n", data->title);
    fprintf(fp, "%%%% Creator: \n");
    fprintf(fp, "%%%% CreationDate: \n");
    fprintf(fp, "%%%% For: \n");
    fprintf(fp, "%%%% Page: 1\n");
    fprintf(fp, "/inch { 72 mul } def\n");
    fprintf(fp, "/TopMargin    %g inch def\n", topMargin);
    fprintf(fp, "/BottomMargin %g inch def\n", bottomMargin);
    fprintf(fp, "/LeftMargin   %g inch def\n", leftMargin);
    fprintf(fp, "/RightMargin  %g inch def\n", rightMargin);
    fprintf(fp, "/PaperWidth   %g inch def\n", paperWidth);
    fprintf(fp, "/PaperHeight  %g inch def\n", paperHeight);
    fprintf(fp, "/TopLeftPos { LeftMargin PaperHeight TopMargin sub } def\n");
    fprintf(fp, "\n");
    fprintf(fp, "%% define font information\n");
    fprintf(fp, "/TheFontName /Helvetica def\n");
    fprintf(fp, "/TheFontSize  12 def\n");
    fprintf(fp, "/TheLineWidth 1 def                %% linewidth scaling factor\n");
    fprintf(fp, "/TheScaleX    1 def                %% x-axis scaling factor\n");
    fprintf(fp, "/TheScaleY    1 def                %% y-axis scaling factor\n");
    fprintf(fp, "/TheScaleFont 0.9 def              %% font scaling factor\n", scaleFon
t);
    fprintf(fp, "\n");
    fprintf(fp, "%% define procedures\n");
    fprintf(fp, "/dot1 {\n");
    /* ドット出力時に半径を指定できるようにパラメータ(r)を追加した。 */
    fprintf(fp, "%% called as: x y r dot1\n");
    fprintf(fp, "/r exch def\n");
    fprintf(fp, "TheScaleY mul /y exch def \n");
    fprintf(fp, "TheScaleX mul /x exch def\n");
    /* 指定された点の大きさの1/4を設定 */
    /* ドット出力時に半径を指定できるようにパラメータ(r)を追加した。 */
    fprintf(fp, "x y r 0 360 arc fill\n");
    fprintf(fp, "} def\n");
    fprintf(fp, "\n");
    fprintf(fp, "/line { \n");
    fprintf(fp, "%% called as: x1 y1 x2 y2 line\n");
    fprintf(fp, "TheScaleY mul /y2 exch def\n");
    fprintf(fp, "TheScaleX mul /x2 exch def\n");
    fprintf(fp, "TheScaleY mul /y1 exch def\n");
    fprintf(fp, "TheScaleX mul /x1 exch def\n");
    fprintf(fp, "x1 y1 moveto\n");
    fprintf(fp, "x2 y2 lineto\n");
    fprintf(fp, "stroke\n");
    fprintf(fp, "} def\n");
    fprintf(fp, "\n");
    fprintf(fp, "/rectangle { \n");
    fprintf(fp, "%% called as: x1 y1 x2 y2 rectangle\n");
    fprintf(fp, "TheScaleY mul /y2 exch def\n");
    fprintf(fp, "TheScaleX mul /x2 exch def\n");
    fprintf(fp, "TheScaleY mul /y1 exch def\n");
    fprintf(fp, "TheScaleX mul /x1 exch def\n");
    fprintf(fp, "x1 y1 moveto\n");
    fprintf(fp, "x1 y2 lineto\n");
    fprintf(fp, "x2 y2 lineto\n");
    fprintf(fp, "x2 y1 lineto\n");
    fprintf(fp, "closepath\n");
    fprintf(fp, "stroke\n");
    fprintf(fp, "} def\n");
    fprintf(fp, "\n");

    /* 線分を出力する場合の開始指定マクロ */
    fprintf(fp, "/setStartLine { \n");
    fprintf(fp, "%% called as: x1 y1 setStartLine\n");
    fprintf(fp, "TheScaleY mul /y1 exch def\n");
    fprintf(fp, "TheScaleX mul /x1 exch def\n");
    /* 線分の接続タイプをsetlinejoinにて指定(0:miter,1:round,2:bevel) */
    fprintf(fp, "  0 setlinejoin\n");
    /* 線分の両端のキャップタイプをsetlinecapにて指定
       (0:butt,1:rount,2:projecting squre) */
    fprintf(fp, "  0 setlinecap\n");
    fprintf(fp, "x1 y1 moveto\n");
    fprintf(fp, "} def\n");
    fprintf(fp, "\n");

    /* 線分の出力ポイントを指定するマクロ */
    fprintf(fp, "/setLineTo { \n");
    fprintf(fp, "%% called as: x1 y1 setLineTo\n");
    fprintf(fp, "TheScaleY mul /y1 exch def\n");
    fprintf(fp, "TheScaleX mul /x1 exch def\n");
    fprintf(fp, "x1 y1 lineto\n");
    fprintf(fp, "} def\n");
    fprintf(fp, "\n");

    /* 線分の終了を指定するマクロ */
    fprintf(fp, "/setEndLine { \n");
    fprintf(fp, "%% called as: setEndLine\n");
    fprintf(fp, "stroke\n");
    fprintf(fp, "} def\n");
    fprintf(fp, "\n");

    /* テキスト出力オフセット指定対応版 */
    fprintf(fp, "/setText {\n");
    fprintf(fp, "%% called as: text x y xoffset yoffset setText\n");
    fprintf(fp, "/yoffset exch def\n");
    fprintf(fp, "/xoffset exch def\n");
    fprintf(fp, "TheScaleY mul /y exch def\n");
    fprintf(fp, "TheScaleX mul /x exch def\n");
    fprintf(fp, "dup\n");
    fprintf(fp, "stringwidth\n");
    fprintf(fp, "pop\n");
    fprintf(fp, "/sh TheFontSize TheScaleFont mul def\n");
    fprintf(fp, "/sw exch def\n");
    /* X座標にオフセットを加算する */
    fprintf(fp, "0 sw sub 2 div x add xoffset add \n");
    /* Y座標にオフセットを加算する */
    fprintf(fp, "0 sh sub 2 div y add yoffset add \n");
    fprintf(fp, "moveto\n");
    fprintf(fp, "show\n");
    fprintf(fp, "stroke\n");
    fprintf(fp, "} def\n");
    fprintf(fp, "\n");

    /* rightTextのオフセット指定対応版 */
    fprintf(fp, "/rightText2 {\n");
    /* オフセット指定対応により、パラメータを追加 */
    fprintf(fp, "%% called as: text x y xoffset yoffset rightText2\n");
    fprintf(fp, "/yoffset exch def\n");
    fprintf(fp, "/xoffset exch def\n");
    fprintf(fp, "TheScaleY mul /y exch def\n");
    fprintf(fp, "TheScaleX mul /x exch def\n");
```

```
        fprintf(fp, "dup\n");
        fprintf(fp, "stringwidth\n");
        fprintf(fp, "pop\n");
        fprintf(fp, "/sh TheFontSize TheScaleFont mul def\n");
        fprintf(fp, "/sw exch def\n");
        /* X座標にオフセットを加算する */
        fprintf(fp, "x sw sub xoffset add\n");
        /* Y座標にオフセットを加算する */
        fprintf(fp, "y sh 2 div sub yoffset add\n");
        fprintf(fp, "moveto\n");
        fprintf(fp, "show\n");
        fprintf(fp, "stroke\n");
        fprintf(fp, "} def\n");
        fprintf(fp, "\n");

        /* leftTextのオフセット指定対応版 */
        fprintf(fp, "/leftText2 {\n");
        fprintf(fp, "%% called as: text x y xoffset yoffset leftText2\n");
        fprintf(fp, "/yoffset exch def\n");
        fprintf(fp, "/xoffset exch def\n");
        fprintf(fp, "TheScaleY mul /y exch def\n");
        fprintf(fp, "TheScaleX mul /x exch def\n");
        fprintf(fp, "dup\n");
        fprintf(fp, "stringwidth\n");
        fprintf(fp, "pop\n");
        fprintf(fp, "/sh TheFontSize TheScaleFont mul def\n");
        fprintf(fp, "/sw exch def\n");
        /* X座標にオフセットを加算する */
        fprintf(fp, "x xoffset add\n");
        /* Y座標にオフセットを加算する */
        fprintf(fp, "y sh 2 div sub yoffset add\n");
        fprintf(fp, "moveto\n");
        fprintf(fp, "show\n");
        fprintf(fp, "stroke\n");
        fprintf(fp, "} def\n");
        fprintf(fp, "\n");
        fprintf(fp, "/centerText {\n");
        fprintf(fp, "%% called as: text x y w h centerText\n");
        fprintf(fp, "TheScaleY mul /height exch def\n");
        fprintf(fp, "TheScaleX mul /width exch def\n");
        fprintf(fp, "TheScaleY mul /y exch def\n");
        fprintf(fp, "TheScaleX mul /x exch def\n");
        fprintf(fp, "dup\n");
        fprintf(fp, "stringwidth\n");
        fprintf(fp, "pop\n");
        fprintf(fp, "/sh TheFontSize TheScaleFont mul def\n");
        fprintf(fp, "/sw exch def\n");
        fprintf(fp, "width sw sub 2 div x add\n");
        fprintf(fp, "height sh sub 2 div y add\n");
        fprintf(fp, "moveto\n");
        fprintf(fp, "show\n");
        fprintf(fp, "stroke\n");
        fprintf(fp, "} def\n");
        fprintf(fp, "\n");
        fprintf(fp, "/centerTextRot90 {\n");
        fprintf(fp, "%% called as: text x y w h centerTextRot90\n");
        fprintf(fp, "TheScaleX mul /height exch def\n");
        fprintf(fp, "TheScaleY mul /width exch def\n");
        fprintf(fp, "TheScaleY mul /y exch def\n");
        fprintf(fp, "TheScaleX mul /x exch def\n");
        fprintf(fp, "dup\n");
        fprintf(fp, "stringwidth\n");
        fprintf(fp, "pop\n");
        fprintf(fp, "/sh TheFontSize TheScaleFont mul def\n");
        fprintf(fp, "/sw exch def\n");
        fprintf(fp, "height sh sub 2 div x add\n");
        fprintf(fp, "width sw sub 2 div y add\n");
        fprintf(fp, "moveto\n");
        fprintf(fp, "90 rotate\n");
        fprintf(fp, "show\n");
        fprintf(fp, "-90 rotate\n");
        fprintf(fp, "stroke\n");
        fprintf(fp, "} def\n");
```

```
        fprintf(fp, "\n");
        fprintf(fp, "/rightText {\n");
        fprintf(fp, "%% called as: text x y rightText\n");
        fprintf(fp, "TheScaleY mul /y exch def\n");
        fprintf(fp, "TheScaleX mul /x exch def\n");
        fprintf(fp, "dup\n");
        fprintf(fp, "stringwidth\n");
        fprintf(fp, "pop\n");
        fprintf(fp, "/sh TheFontSize TheScaleFont mul def\n");
        fprintf(fp, "/sw exch def\n");
        fprintf(fp, "x sw sub\n");
        fprintf(fp, "y sh 2 div sub\n");
        fprintf(fp, "moveto\n");
        fprintf(fp, "show\n");
        fprintf(fp, "stroke\n");
        fprintf(fp, "} def\n");
        fprintf(fp, "\n");
        fprintf(fp, "/leftText {\n");
        fprintf(fp, "%% called as: text x y leftText\n");
        fprintf(fp, "TheScaleY mul /y exch def\n");
        fprintf(fp, "TheScaleX mul /x exch def\n");
        fprintf(fp, "dup\n");
        fprintf(fp, "stringwidth\n");
        fprintf(fp, "pop\n");
        fprintf(fp, "/sh TheFontSize TheScaleFont mul def\n");
        fprintf(fp, "/sw exch def\n");
        fprintf(fp, "x\n");
        fprintf(fp, "y sh 2 div sub\n");
        fprintf(fp, "moveto\n");
        fprintf(fp, "show\n");
        fprintf(fp, "stroke\n");
        fprintf(fp, "} def\n");
        fprintf(fp, "\n");
        fprintf(fp, "/setFontInfo {\n");
        fprintf(fp, "%% called as: font-name font-size setFontInfo\n");
        fprintf(fp, "/TheFontSize exch def\n");
        fprintf(fp, "/TheFontName exch def\n");
        fprintf(fp, "TheFontName findfont TheFontSize TheScaleFont mul scalefont setfont\n
");
        fprintf(fp, "} def\n");
        fprintf(fp, "\n");
        fprintf(fp, "/setAxesScale {\n");
        fprintf(fp, "%% called as: x y scale-x scale-y setAxesScale\n");
        fprintf(fp, "/scaley exch def\n");
        fprintf(fp, "/scalex exch def\n");
        fprintf(fp, "TheScaleY mul /y exch def\n");
        fprintf(fp, "TheScaleX mul /x exch def\n");
        fprintf(fp, "gsave\n");
        fprintf(fp, "x y translate\n");
        fprintf(fp, "/TheScaleX TheScaleX scalex mul def\n");
        fprintf(fp, "/TheScaleY TheScaleY scaley mul def\n");
        fprintf(fp, "} def\n");
        fprintf(fp, "\n");
        fprintf(fp, "/restoreAxesScale {\n");
        fprintf(fp, "%% called as: x y scale-x scale-y restoreAxesScale\n");
        fprintf(fp, "/scaley exch def\n");
        fprintf(fp, "/scalex exch def\n");
        fprintf(fp, "TheScaleY mul /y exch def\n");
        fprintf(fp, "TheScaleX mul /x exch def\n");
        fprintf(fp, "grestore\n");
        fprintf(fp, "/TheScaleX TheScaleX scalex div def\n");
        fprintf(fp, "/TheScaleY TheScaleY scaley div def\n");
        fprintf(fp, "} def\n");
        fprintf(fp, "\n");
        fprintf(fp, "%% set initial/default values\n");
        fprintf(fp, "/Helvetica 12 setFontInfo\n");
        fprintf(fp, "0.5 setlinewidth\n");
        fprintf(fp, "\n");
        fprintf(fp, "%% set translation to whole paper\n");

        /* 用紙全体の座標系の設定 */
        if (data->orientation == PAPER_LANDSCAPE) {
                fprintf(fp, "%% paper orientation is landscape.\n");
```

```
        fprintf(fp, "%.5g inch %.5g inch %.5g inch %.5g inch setAxesScale\n",
                        paperWidth - rightMargin,
                        bottomMargin,
                        paperHeight - topMargin - bottomMargin,
                        paperWidth - rightMargin - leftMargin);
        fprintf(fp, "90 rotate\n");
}
else {
        fprintf(fp, "%% paper orientation is portrait.\n");
        fprintf(fp, "%.5g inch %.5g inch %.5g inch %.5g inch setAxesScale\n",
                        leftMargin,
                        bottomMargin,
                        paperWidth - rightMargin - leftMargin,
                        paperHeight - topMargin - bottomMargin);
}
fprintf(fp, "\n");

for (i=0; i<data->nGraphs; i++) {
        /* 個々のグラフの相対位置、サイズを求める */
        x = (float)data->winX[i]/data->mainWinWidth;
        y = (float)(data->mainWinHeight-data->winY[i])/data->mainWinHeight;
        width = (float)data->winWidth[i]/data->mainWinWidth;
        height = (float)data->winHeight[i]/data->mainWinHeight;

        /* 個々のグラフ座標系の設定 */
        fprintf(fp, "%%%%\n");
        fprintf(fp, "%%%% Graph %d\n", i+1);
        fprintf(fp, "%%%%\n");
        fprintf(fp, "%.5g %.5g %.5g %.5g setAxesScale\n",
                        x, y, width, height);
        fprintf(fp, "\n");

        /* グラフの描画出力 */
        GrPrint(gGraphNo[i], fp);

        /* 用紙全体の座標系へ戻す */
        fprintf(fp, "\n");
        fprintf(fp, "%.5g %.5g %.5g %.5g restoreAxesScale\n",
                        x, y, width, height);
        fprintf(fp, "\n");
}

/* タイトルウィンドウの描画 */
fprintf(fp, "%%%%\n");
fprintf(fp, "%%%% Title\n");
fprintf(fp, "%%%%\n");
x = (float)data->titleWinX/data->mainWinWidth;
y = (float)(data->mainWinHeight-data->titleWinY)/data->mainWinHeight;
width = (float)data->titleWinWidth/data->mainWinWidth;
height = -(float)data->titleWinHeight/data->mainWinHeight;
grPSDrawTitleCenter(fp, gGraphNo[0], data->title,
                                        &x, &y, &width, &height);

/* 中心点の出力 */
fprintf(fp, "%%%%\n");
fprintf(fp, "%%%% Center mark\n");
fprintf(fp, "%%%%\n");
x = (float)data->titleWinX/data->mainWinWidth;
y = (float)(data->mainWinHeight-data->titleWinY)/data->mainWinHeight;
width = (float)data->titleWinWidth/data->mainWinWidth;
height = -(float)data->titleWinHeight/data->mainWinHeight;
grPSSetFont(fp, "Helvetica", NULL, NULL, 10);
if (data->orientation == PAPER_LANDSCAPE) {
        fprintf(fp, "(%s) %.5g %.5g %.5g %.5g centerText\n",
                        "+", 0.0, 1.0, 1.0, 0.08);
}
else {
        fprintf(fp, "(%s) %.5g %.5g %.5g %.5g centerText\n",
                        "+", 0.0, 0.0, -0.08, 1.0);
}

/* Prologの出力 */
fprintf(fp, "\n");
```

```
        fprintf(fp, "showpage\n");
        fclose(fp);

        return 0;
}

int grPSDrawTitleCenter(FILE *fp, int gr, char *title, float *x, float *y,
                                                float *width, float *height)
{
        GrAttr *p;

        p = gGraph + gr;
        grPSSetFont(fp, p->title_fontname, p->title_fontbold,
                                p->title_fontitalic, p->title_fontsize);
        fprintf(fp, "  (%s) %.5g %.5g %.5g %.5g centerText\n",
                        title, *x, *y, *width, *height);

        return 0;
}
```

```
/*
 * ACR PLOT TOOL Version 1.0
 *               Version 2.0
 *               Version 2.1
 *    (c) Copyright 1997,1998,1999
 *    ATR Adaptive Communications Research Laboratories
 *    All Rights Reserved
 */

/*
 * read.c - データファイル読み込み
 */

#include "stdafx.h"
#include "PLOT.h"
#include "graph.h"

static int  gLineNo;                                /* データファイルの行番号 */
static char gBuf[MAX_BUFFER_LEN+1];
static char gErrBuf[300];

char gColorTable[][MAX_COLOR_LEN] = {
  "black:000000",
  "red:FF0000",
  "blue:0000FF",
  "green:00FF00",
  "pink:FFB0C0",
  "violet:F080F0",
  "yellow:FFFF00",
  "orange:FFA000",
  "brown:A03030",
  "spring green:00FF80"
};

static char  *usage[11] = {
        "Usage: plot.exe [options...]\r\n",
        "  -data  <file>  data filename. default is 'fort.99'.\r\n",
        "  -head  <file>  header filename.\r\n",
        "  -body  <file>  body filename. setup with -header options.\r\n",
        "  -ps    <file>  output PS filename.\r\n",
        "  -eps   <file>  output EPS filename.\r\n",
        "  -geometry  <geometry>  size and position.\r\n",
        "  -landscape    landscape.\r\n",
        "  -portrait     portrait.\r\n",
        "  -aspectratio  aspect ratio.\r\n",
        "  -dotratio     dot size ratio. default is 1.",
};

void PrtUsage()
{
        char  *p = gErrBuf;
        for (int i=0; i<11; i++) {
                strcpy(p, usage[i]);
                p += strlen(p);
        }

        CPLOTApp::PrtMessage(gErrBuf, 0);
}

/*
 * データファイルの読み込み
   datafname: データファイル名
   headerfname: ヘッダーデータファイル名
   bodyfname: ボディデータファイル名
 */
DATA *ReadDataFile(char *datafname, char *headerfname, char *bodyfname)
{
        FILE  *fp;
        char  *p;
        DATA  *data;
        int    i, j;
        int        cnt_labeldata = 0;   /* ラベル列の数 */
        double val;
        char     val_lbl[MAX_LABELDATA_LEN];
        char     filename[BUFSIZ+1];

        // データおよびボディファイル名の指定なし
        // ボディファイル名のみ指定時はエラー
        if ((datafname[0]==0 && bodyfname[0]==0) ||
                (bodyfname[0] && headerfname[0]==0)) {
                PrtUsage();
                return NULL;
        }

        /* ヘッダファイル指定の時は、ヘッダファイルをオープンする */
        if (headerfname[0])     strcpy(filename, headerfname);
        else                    strcpy(filename, datafname);

        /* データの領域確保 */
        try {
                data = new DATA;
        }
        catch (CMemoryException* e) {
                sprintf(gErrBuf, "Cannot allocate memory.\n");
                CPLOTApp::PrtMessage(gErrBuf, 0);
                e->Delete();
                return NULL;
        }

        /* データファイルのオープン （ヘッダファイル指定の時はヘッダファイル
           をオープン:VERSION2 )*/
        if ((fp = fopen(filename, "r")) == NULL) {
                data->mode = NODRAW_MODE;
                data->nGraphs = 0;
                data->nRows = data->nCols = 0;
                data->table = new Number[1];
                return data;
        }

        data->mode = WIN_DRAW_MODE;

        /* タイトル文字列の読み込み */
        if (ReadLine(fp, gBuf) == 0) {
                sprintf(gErrBuf, "%s:%d: unexpected end of file\n",
                                filename, gLineNo);
                CPLOTApp::PrtMessage(gErrBuf, 0);
                fclose(fp);
                delete data;
                return NULL;
        }

        strcpy(data->title, gBuf);

        /* グラフの数、データの行数、データ列数の読み込み */
        if (ReadLine(fp, gBuf) == 0) {
                fclose(fp);
                delete data;
                return NULL;
        }

        sscanf(gBuf, "%d %d %d", &(data->nGraphs), &(data->nRows), &(data->nCols));

        /* グラフ数のチェック */
        if (data->nGraphs<=0 || data->nGraphs>MAX_GRAPH_COUNT) {
                sprintf(gErrBuf, "%s:%d: number of graphs is out of range (%d)\n",
                                filename, gLineNo, data->nGraphs);
                CPLOTApp::PrtMessage(gErrBuf, 0);
                fclose(fp);
                delete data;
                return NULL;
        }
        /* 列数のチェック */
        if (data->nCols <= 0) {
                sprintf(gErrBuf, "%s:%d: number of columns is less than 0 (%d)\n",
                                filename, gLineNo, data->nCols);
                CPLOTApp::PrtMessage(gErrBuf, 0);
```

```
                fclose(fp);
                delete data;
                return NULL;
        }
        /* 行数のチェック */
        if (data->nRows <= 0) {
                sprintf(gErrBuf, "%s:%d: number of rows is less than 0 (%d)\n",
                                filename, gLineNo, data->nRows);
                CPLOTApp::PrtMessage(gErrBuf, 0);
                fclose(fp);
                delete data;
                return NULL;
        }

        /* グラフのX/Y軸の列番号、線種の読み込み
           線種指定は以下の指定が可能。
           0    折線
           1    点
           2    点線
           3    1点鎖線
           4    長点線
           5    長1点鎖線

           オプション指定の読み込み
           オプション指定は省略可能。指定の方法は、<属性>=<値>とし、属性は複数設定で
           きる。同一属性が複数回指定された場合は、最後の指定が有効となる。属性の種類
           は以下の通り。
           ・label          ラベル列指定。データ列の列番号を指定する。ラベル列は必ずし
                                        も列ラベルで@label指定した列でなくてもよい。
           ・dotsize        線幅、点の大きさの指定。デフォルト値は1。
           ・dotcolor       グラフの色の指定。指定の方法は、カラー番号(ヘッダーで定義
                                        済みの10色程度の色番号)の他、"#RRGGBB"のようにR
GB値での指
                                        定も可能。
           ・labelsize      ラベル文字列の大きさの指定。
           ・duplicate      duplicate=1の場合、直前のグラフ上に描画する。
                                        複数行に渡って指定可能。
           ・labeldist      ラベル文字列を描画する位置の指定。
                                        どれだけデータ点と離すかを指定する。
       例) label=3 dotsize=2 dotcolor=1 duplicate=1
           */
        data->nWindows = 0;

        for (i=0;i<data->nGraphs; i++) {
                if (ReadLine(fp, gBuf) == 0) {
                        fclose(fp);
                        delete data;
                        return NULL;
                }
                sscanf(gBuf, "%d %d %d",
                        &(data->colX[i]), &(data->colY[i]), &(data->lineType[i]));

                /* チェック */
                if (data->colX[i]<1 || data->colX[i]>data->nCols) {
                        sprintf(gErrBuf, "%s:%d: number of x column no. is out of range (%
d)\n",
                                        filename, gLineNo, data->colX[i]);
                        CPLOTApp::PrtMessage(gErrBuf, 0);
                        fclose(fp);
                        delete data;
                        return NULL;
                }
                if (data->colY[i]<1 || data->colY[i]>data->nCols) {
                        sprintf(gErrBuf, "%s:%d: number of y column no. is out of range (%
d)\n",
                                        filename, gLineNo, data->colY[i]);
                        CPLOTApp::PrtMessage(gErrBuf, 0);
                        fclose(fp);
                        delete data;
                        return NULL;
                }

                /* グラフのX/Y軸の列番号、線種に続くオプション指定の数を求める */
```

```
                if (CheckGraphOption(data, i, gBuf) < 0) {
                        sprintf(gErrBuf, "%s:%d: graph option error\n",
                                        filename, gLineNo);
                        CPLOTApp::PrtMessage(gErrBuf, 0);
                        fclose(fp);
                        delete data;
                        return NULL;
                }

                /* ウィンドウ数のカウント */
                if (!data->duplicate[i]) data->nWindows++;
        }

        if (data->nWindows<1 || data->nWindows>MAX_WINDOW_COUNT) {
                sprintf(gErrBuf, "%s:%d: number of windows is out of range (%d)\n",
                                filename, gLineNo, data->nWindows);
                CPLOTApp::PrtMessage(gErrBuf, 0);
                fclose(fp);
                delete data;
                return NULL;
        }

        /* グラフのX/Y軸のラベルの取得 */
        for (i=0; i<data->nCols; i++) {
                if (ReadLine(fp, gBuf) == 0) {
                        sprintf(gErrBuf, "%s:%d: unexpected end of file\n",
                                        filename, gLineNo);
                        CPLOTApp::PrtMessage(gErrBuf, 0);
                        fclose(fp);
                        delete data;
                        return NULL;
                }
                strncpy(data->colHeader[i], gBuf, MAX_HEADER_LEN);
                /* ラベル列の数を数える */
                if (strcmp(data->colHeader[i], "@label") == 0) cnt_labeldata++;
        }

        /* 指定データ列がラベル列になってないかチェックする */
        for (i=0; i<data->nGraphs; i++) {
                if (strcmp(data->colHeader[data->colX[i]-1], "@label") == 0) {
                        sprintf(gErrBuf, "order X column[%d] is Label Data column.\n",
                                        data->colX[i]);
                        CPLOTApp::PrtMessage(gErrBuf, 0);
                        fclose(fp);
                        delete data;
                        return NULL;
                }
                if (strcmp(data->colHeader[data->colY[i]-1], "@label") == 0) {
                        sprintf(gErrBuf, "order Y column[%d] is Label Data column.\n",
                                        data->colY[i]);
                        CPLOTApp::PrtMessage(gErrBuf, 0);
                        fclose(fp);
                        delete data;
                        return NULL;
                }
        }

        /* データ領域の確保 */
        try {
                data->table = new Number[data->nCols*data->nRows];
        }
        catch (CMemoryException* e) {
                sprintf(gErrBuf, "Cannot allocate memory.(%dx%d)\n",
                                data->nCols, data->nRows);
                CPLOTApp::PrtMessage(gErrBuf, 0);
                fclose(fp);
                delete data;
                e->Delete();
                return NULL;
        }

        /* ラベルデータ領域の確保 */
        data->table_label = NULL;
```

```
        if (cnt_labeldata > 0) {
                try {
                        data->table_label = new char[cnt_labeldata*data->nRows*MAX_LABELDA
TA_LEN];
                }
                catch (CMemoryException* e) {
                        sprintf(gErrBuf, "Cannot allocate label data memory.(%dx%d)\n",
                                        cnt_labeldata, data->nRows);
                        CPLOTApp::PrtMessage(gErrBuf, 0);
                        fclose(fp);
                        delete data;
                        e->Delete();
                        return NULL;
                }

                memset(data->table_label, 0, sizeof(data->table_label));
        }
        /* ヘッダファイル指定の時はファイルを閉じ、データファイルをオープンする */
        if (headerfname[0]) {
                fclose(fp);
                if (bodyfname[0]) strcpy(filename, bodyfname);
                else              strcpy(filename, datafname);

                gLineNo = 0;
                /* データファイルのオープン */
                if ((fp = fopen(filename, "r")) == NULL) {
                        sprintf(gErrBuf, "Cannot open file '%s'.\n", filename);
                        CPLOTApp::PrtMessage(gErrBuf, 0);
                        if (cnt_labeldata > 0) delete[] data->table_label;
                        delete data;
                        return NULL;
                }

                /* データファイル指定時のみヘッダを読み飛ばす */
                if (bodyfname[0] == 0) {
                        /* データファイルにヘッダが含まれているときは、そこを読み飛ばす */

                        if (fpSetToData(fp, data) == -1) {
                                fclose(fp);
                                if (cnt_labeldata > 0) delete[] data->table_label;
                                delete data;
                                return NULL;
                        }
                }
        }

        /* データの読み込み */
        for (i=0; i<data->nRows; i++) {
                /* データを一行読み込む */
                if (ReadLine(fp, gBuf) == 0) {
                        sprintf(gErrBuf,
                                        "%s:%d: Warning: valid rows less than previous.(%d
<%d)\n",
                                        filename, gLineNo, i, data->nRows);
                        CPLOTApp::PrtMessage(gErrBuf, 1);
                        break;
                }
                if (strncmp(gBuf, STOP_STRING, strlen(STOP_STRING)) == 0) {
                        break;
                }

                p = gBuf;
                cnt_labeldata = 0;
                for (j=0; j<data->nCols; j++) {
                        if (*p == '\0') {
                                sprintf(gErrBuf, "%s:%d: valid columns less than previous.
(%d<%d)\n",
                                                filename, gLineNo, j, data->nCols);
                                CPLOTApp::PrtMessage(gErrBuf, 0);
                                fclose(fp);
                                if (cnt_labeldata > 0) delete[] data->table_label;
                                delete data;
                                return NULL;
                        }
                        if (strcmp(data->colHeader[j], "@label") != 0) {
                                sscanf(p, "%lf", &val);
                                /* 列優先でデータを格納する */
                                *(data->table+j*data->nRows+i) = (Number)val;
                        }
                        else {
                                sscanf(p, "%s", val_lbl);
                                strcpy(data->table_label+
                                        (cnt_labeldata*data->nRows+i)*MAX_LABELDATA_LEN
,
                                        val_lbl);
                                cnt_labeldata++;
                        }

                        /* 次の空白文字までよみ飛ばす */
                        while (*p && *p!=' ' && *p!='\t' && *p!='\n') p++;

                        /* 次の空白以外の文字までよみ飛ばす */
                        while (*p==' ' || *p=='\t' || *p=='\n') p++;

                }
        }

        data->nValidRows = i;

        fclose(fp);
        return data;
}

/*
 * ファイルから一行よみ出す
 *      空行をよみ飛ばすことに注意
 *      1カラム目が"!"で始まる行はコメント行と見なして読み飛ばす
 *                                                      (Version 2.0)
 *      関数仕様変更:char *ReadLine(FILE*)->int ReadLine(FILE*,char*)
 *              (Version 2.1)
 */
int ReadLine(FILE *fp, char *bp)
{
        char  *p;

        while (fgets(bp, MAX_BUFFER_LEN, fp)) {
                gLineNo++;
                p = bp;
                /* 先頭の空白、タブ、改行は読み飛ばす */
                while (*p==' ' || *p=='\t' || *p=='\n') p++;
                /* コメント行でない時 */
                if (*p && *p!='!') {
                        while (*p && *p!='\n') *bp++ = *p++;
                        *bp = 0;
                        return 1;
                }
        }

        return 0;
}

/*
        グラフ定義のオプション指定を設定する
*/
int SetGraphOption(DATA *data, int n, char *opt)
{
        char    zoku[MAX_OPTION_LEN];
        char    *c;
        int     i;
        int     num;
        float   wnum;

        if ((c = StringIndex(opt, '=')) != NULL) {
                i = strlen(opt)-strlen(c);
                strncpy(zoku, opt, i);
                zoku[i] = '\0';
```

```
		if (!strcmp(zoku, LABEL_OPTION) || !strcmp(zoku, LABEL_OPT_SHORT)) {
			num = atoi(c+1);
			if (num<0 || num>data->nCols) {
				sprintf(gErrBuf, "%s=%d is wrong\n", zoku, num);
				CPLOTApp::PrtMessage(gErrBuf, 0);
				data->colLabel[n] = -1;
			}
			else {
				data->colLabel[n] = num;
			}
		}
		else if (!strcmp(zoku, DOT_SIZE_OPTION) ||
				!strcmp(zoku, DOT_SIZE_OPT_SHORT)) {
			sscanf(c+1, "%f", &wnum);
			if (wnum >= 0) data->dotSize[n] = wnum;
			else {
				sprintf(gErrBuf, "%s=%f is wrong\n", zoku, wnum);
				CPLOTApp::PrtMessage(gErrBuf, 0);
			}
		}
		else if (!strcmp(zoku, DOT_COLOR_OPTION) ||
				!strcmp(zoku, DOT_COLOR_OPT_SHORT)) {
			if ((c+1)[0] == '#') {
				sprintf(data->dot_color[n], "%s", c+1);
			}
			else {
				num = atoi(c+1);
				if (num<0 || num>MAX_COLOR_TABLE-1) {
					sprintf(gErrBuf, "%s=%d is wrong\n", zoku, num);
					CPLOTApp::PrtMessage(gErrBuf, 0);
					sprintf(data->dot_color[n], "%s", gColorTable[0]);

				}
				else {
					sprintf(data->dot_color[n], "%s", gColorTable[num]
);
				}
			}
		}
		else if (!strcmp(zoku, LABEL_SIZE_OPTION) ||
				!strcmp(zoku, LABEL_SIZE_OPT_SHORT)) {
			num = atoi(c+1);
			if (num > 0) {
				data->dotlabel_fontsize[n] = num;
			}
			else {
				sprintf(gErrBuf, "%s=%d is wrong\n", zoku, num);
				CPLOTApp::PrtMessage(gErrBuf, 0);
			}
		}
		else if (!strcmp(zoku, DUPLICATE_OPTION) ||
				!strcmp(zoku, DUPLICATE_OPT_SHORT)) {
			num = atoi(c+1);
			if (num==1 && n>0) {
				data->duplicate[n] = 1;
			}
			else if (num != 0) {
				sprintf(gErrBuf, "%s=%d is wrong\n", zoku, num);
				CPLOTApp::PrtMessage(gErrBuf, 0);
			}
		}
		else if (!strcmp(zoku, LABEL_DIST_OPTION) ||
				!strcmp(zoku, LABEL_DIST_OPT_SHORT)) {
			num = atoi(c+1);
			if (num >= 0) {
				data->dotlabel_dist[n] = num;
			}
			else {
				sprintf(gErrBuf, "%s=%d is wrong\n", zoku, num);
				CPLOTApp::PrtMessage(gErrBuf, 0);
			}
		}
		else {
```

```
			sprintf(gErrBuf, "%s is invalid option\n", zoku);
			CPLOTApp::PrtMessage(gErrBuf, 0);
			return -1;
		}
		return 0;
	}
	else {
		sprintf(gErrBuf, "%s is invalid option format\n", opt);
		CPLOTApp::PrtMessage(gErrBuf, 0);
		return -1;
	}
}

/*
   グラフ定義のグラフのX/Y軸の列番号、線種に続くオプション指定を調べる
*/
int CheckGraphOption(DATA *data, int n, char *buf)
{
	int	cnt, char_cnt, total_cnt;
	int	cp;
	char	opt[MAX_OPTION_LEN];
	int	i, l;

	/* 初期値設定 */
	sprintf(data->dot_color[n], "%s", gColorTable[0]);
	data->duplicate[n]		= 0;
	data->dotSize[n]		= 1;
	data->colLabel[n]		= -1;
	data->dotlabel_fontsize[n] = GR_DEFAULT_LABEL_FONTSIZE;
	data->dotlabel_dist[n]	= 0;
	cnt = 0;

	cp = 0;
	char_cnt = 0;
	total_cnt = 0;

	l = strlen(buf);
	for (i=0; i<=l; i++) {

		/* 空白文字の場合 */
		if ((buf[i] == ' ') || (buf[i] == '\n') ||
			(buf[i] == '\t') || (buf[i] == '\0')) {

			/* 文字がある場合 */
			if (char_cnt > 0) {
				sscanf(&buf[cp], "%s", opt);
				if (total_cnt<3 && isdigit(*opt)) {
					/* Skip reading */
				}
				else {
					SetGraphOption(data, n, opt);
					cnt++;
				}
				total_cnt++;
				cp = i;
				char_cnt = 0;
			}

			/* 文字がない場合 */
			else {
				/* Skip */
			}
		}

		/* 空白文字でない場合 */
		else {
			char_cnt++;
		}
	}

	return cnt;
}
```

```
/*
        カラー定義文字列を別ける
        */
int StringSeparate(char *s, char *color, int cn)
{
        char  *c;

        if (c = StringIndex(s, ':')) {
                switch (cn) {
                  case 0:
                        strncpy(color, s, strlen(s)-strlen(c));
                        color[strlen(s)-strlen(c)] = '\0';
                        break;
                  default:
                        strcpy(color, c+1);
                        break;
                }
                return 0;
        }
        else {
                return -1;
        }
}

char *StringIndex(char *s, char c)
{
        while (*s && *s!=c) s++;

        return (*s)? s: NULL;
}

/* データファイルのヘッダを読み飛ばす */
int fpSetToData(FILE *fp, DATA *data)
{
        int i;
        int     graph, row, col;
        int rowcnt = 0;

        /* 2行目まですらす */
        for (i=0; i<2; i++) {
                if (ReadLine(fp, gBuf) == 0) return -1;
                rowcnt++;
        }

        /* グラフ数、データ行数、データ列数を求める */
        sscanf(gBuf, "%d %d %d", &graph, &row, &col);

        /* ヘッダファイルのデータ行数、列数と同じかチェック */
        if (row!=data->nRows || col!=data->nCols) {
                sprintf(gErrBuf, "file format is different. header file and data file\n");

                CPLOTApp::PrtMessage(gErrBuf, 0);
                return -1;
        }

        /* グラフ指定行を読み飛ばす */
        for (i=0; i<graph; i++) {
                if (ReadLine(fp, gBuf) == 0) {
                        return -1;
                }
                rowcnt++;
        }

        /* グラフのX/Y軸ラベル行を読み飛ばす */
        for (i=0; i<col; i++) {
                if (ReadLine(fp, gBuf) == 0) {
                        return -1;
                }
                rowcnt++;
        }

        return 1;
}
```

```
//{{NO_DEPENDENCIES}}
// Microsoft Developer Studio generated include file.
// Used by PLOT.rc
//
#define IDOK2                           3
#define ID_SAVE_INI                     4
#define IDD_ABOUTBOX                    100
#define IDR_MAINFRAME                   128
#define IDR_PLOTTYPE                    129
#define IDD_DIALOG1                     130
#define IDD_DIALOG2                     131
#define IDD_DIALOG3                     132
#define IDI_ICON1                       134
#define IDI_PLOT                        134
#define IDC_EDIT1                       1000
#define IDC_BUTTON1                     1001
#define IDC_EDIT3                       1001
#define IDC_EDIT2                       1002
#define IDC_RADIO1                      1002
#define IDC_BUTTON2                     1003
#define IDC_RADIO2                      1003
#define IDC_RADIO3                      1004
#define IDC_CHECK1                      1004
#define IDC_CHECK2                      1005
#define IDC_BUTTON3                     1006
#define IDC_BUTTON4                     1007
#define IDC_BUTTON5                     1008
#define IDC_BUTTON6                     1009
#define IDC_EDIT4                       1010
#define IDC_EDIT5                       1011
#define IDC_EDIT6                       1012
#define IDC_EDIT7                       1013
#define IDC_EDIT8                       1014
#define IDC_EDIT9                       1015
#define IDC_EDIT10                      1016
#define IDC_EDIT11                      1017
#define IDC_EDIT12                      1018
#define IDC_EDIT13                      1019
#define IDC_EDIT14                      1020
#define IDC_EDIT15                      1021
#define ID_FILE_OPEN_HEAD               32771
#define ID_FILE_SAVE_PS                 32772
#define ID_PROP                         32773
#define ID_HELP_PLOT                    32774

// Next default values for new objects
//
#ifdef APSTUDIO_INVOKED
#ifndef APSTUDIO_READONLY_SYMBOLS
#define _APS_3D_CONTROLS                     1
#define _APS_NEXT_RESOURCE_VALUE        135
#define _APS_NEXT_COMMAND_VALUE         32776
#define _APS_NEXT_CONTROL_VALUE         1008
#define _APS_NEXT_SYMED_VALUE           101
#endif
#endif
```

```
// stdafx.cpp : 標準インクルードファイルを含むソース ファイル
//              PLOT.pch : 生成されるプリコンパイル済ヘッダー
//              stdafx.obj : 生成されるプリコンパイル済タイプ情報

#include "stdafx.h"
```

```
// stdafx.h : 標準のシステム インクルード ファイル、
//            または参照回数が多く、かつあまり変更されない
//            プロジェクト専用のインクルード ファイルを記述します。
//

#if !defined(AFX_STDAFX_H__1B91EA87_65AE_11D3_ABB3_00C04F8862AB__INCLUDED_)
#define AFX_STDAFX_H__1B91EA87_65AE_11D3_ABB3_00C04F8862AB__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

#define VC_EXTRALEAN            // Windows ヘッダーから殆ど使用されないスタッフを除外しま
す。

#include <afxwin.h>         // MFC のコアおよび標準コンポーネント
#include <afxext.h>         // MFC の拡張部分
#include <afxdisp.h>        // MFC のオートメーション クラス
#include <afxdtctl.h>           // MFC の Internet Explorer 4 コモン コントロール サポート

#ifndef _AFX_NO_AFXCMN_SUPPORT
#include <afxcmn.h>                     // MFC の Windows コモン コントロール サポート
#endif // _AFX_NO_AFXCMN_SUPPORT


//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ は前行の直前に追加の宣言を挿入します。

#endif // !defined(AFX_STDAFX_H__1B91EA87_65AE_11D3_ABB3_00C04F8862AB__INCLUDED_)
```

```
// SvpsDlg.cpp : インプリメンテーション ファイル
//

#include "stdafx.h"
#include "PLOT.h"
#include "SvpsDlg.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CSvpsDlg ダイアログ


CSvpsDlg::CSvpsDlg(CWnd* pParent /*=NULL*/)
        : CDialog(CSvpsDlg::IDD, pParent)
{
        //{{AFX_DATA_INIT(CSvpsDlg)
                // メモ - ClassWizard はこの位置にマッピング用のマクロを追加または削除しま
す。
        //}}AFX_DATA_INIT
}


void CSvpsDlg::DoDataExchange(CDataExchange* pDX)
{
        CDialog::DoDataExchange(pDX);
        //{{AFX_DATA_MAP(CSvpsDlg)
        DDX_Control(pDX, IDOK, m_Ok);
        DDX_Control(pDX, IDC_EDIT1, m_Fname);
        //}}AFX_DATA_MAP
}


BEGIN_MESSAGE_MAP(CSvpsDlg, CDialog)
        //{{AFX_MSG_MAP(CSvpsDlg)
        ON_BN_CLICKED(IDC_BUTTON1, OnRefPS)
        ON_BN_CLICKED(IDC_CHECK1, OnSelPS)
        ON_BN_CLICKED(IDC_CHECK2, OnSelEPS)
        ON_EN_UPDATE(IDC_EDIT1, OnUpdateFname)
        //}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CSvpsDlg メッセージ ハンドラ

void CSvpsDlg::OnOK()
{
        int   num;

        num = m_Fname.GetLine(0, m_cFname, FNAME_LEN);
        if (num > 0) {
                for (int i=0; i<num; i++) {
                        if (m_cFname[i]!=' ' && m_cFname[i]!='\t' && m_cFname[i]!='\n') {
                                m_cFname[num] = 0;
                                CDialog::OnOK();
                                return;
                        }
                }
        }

        CPLOTApp::PrtMessage(_T("ファイル名を指定して下さい"));

        m_Fname.SetWindowText("");
        // ファイル名入力欄にフォーカスを
        m_Fname.SetFocus();

        // OKボタンを使用不可に
        m_Ok.EnableWindow(FALSE);
//      CDialog::OnOK();
}

void CSvpsDlg::OnCancel()
{
        CDialog::OnCancel();
}

void CSvpsDlg::OnRefPS()
{
        CFileDialog  f(FALSE, NULL, NULL, OFN_HIDEREADONLY,
                                                "PSファイル(*.ps)|*.ps|EPSファイル(*.eps)|*.eps|す
べてのファイル(*.*)|*.*||");

        f.m_ofn.lpstrTitle = _T("累縦集整滴");
        if (f.DoModal() == IDOK) {
                CString  csPath;
                csPath = f.GetPathName();
                m_Fname.SetWindowText(csPath);
        }
}

void CSvpsDlg::OnSelPS()
{
        if ((((CButton *)GetDlgItem(IDC_CHECK1))->GetState() & 0x0003) == 0)
                m_SelPS &= 0x02;
        else
                m_SelPS |= 0x01;
}

void CSvpsDlg::OnSelEPS()
{
        if ((((CButton *)GetDlgItem(IDC_CHECK2))->GetState() & 0x0003) == 0)
                m_SelPS &= 0x01;
        else
                m_SelPS |= 0x02;
}

void CSvpsDlg::OnUpdateFname()
{
        m_Ok.EnableWindow(TRUE);
}

BOOL CSvpsDlg::OnInitDialog()
{
        CDialog::OnInitDialog();

        m_SelPS = 0;    // チェックなし

        ((CButton *)GetDlgItem(IDC_CHECK1))->SetCheck(FALSE);
        ((CButton *)GetDlgItem(IDC_CHECK2))->SetCheck(FALSE);

        // OKボタンを使用不可に
        m_Ok.EnableWindow(FALSE);

        // ファイル名入力欄にフォーカスを
        m_Fname.SetFocus();

        return FALSE;  // コントロールにフォーカスを設定しないとき、戻り値は TRUE となりま
す
                       // 例外: OCX プロパティ ページの戻り値は FALSE となります
}

void CSvpsDlg::GetFname(char *pFname, int nMax)
{
        memset(pFname, 0, nMax);
        strncpy(pFname, m_cFname, nMax);
}
```

```
#if !defined(AFX_SVPSDLG_H__79A70BF1_7090_11D3_ABB8_00C04F8862AB__INCLUDED_)
#define AFX_SVPSDLG_H__79A70BF1_7090_11D3_ABB8_00C04F8862AB__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000
// SvpsDlg.h : ヘッダー ファイル
//

/////////////////////////////////////////////////////////////////////////////
// CSvpsDlg ダイアログ

class CSvpsDlg : public CDialog
{
// コンストラクション
public:
        CSvpsDlg(CWnd* pParent = NULL);   // 標準のコンストラクタ

// ダイアログ データ
        //{{AFX_DATA(CSvpsDlg)
        enum { IDD = IDD_DIALOG3 };
        CButton m_Ok;
        CEdit   m_Fname;
        //}}AFX_DATA


// オーバーライド
        // ClassWizard は仮想関数のオーバーライドを生成します。
        //{{AFX_VIRTUAL(CSvpsDlg)
        protected:
        virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV サポート
        //}}AFX_VIRTUAL

// インプリメンテーション
        private:
                char m_cFname[FNAME_LEN+1];
                int  m_SelPS;     // PSファイルに変換して保存

        public:
                int  GetSelPs() { return m_SelPS; }
                void GetFname(char *, int);

protected:

        // 生成されたメッセージ マップ関数
        //{{AFX_MSG(CSvpsDlg)
        virtual void OnOK();
        virtual void OnCancel();
        afx_msg void OnRefPS();
        afx_msg void OnSelPS();
        afx_msg void OnSelEPS();
        afx_msg void OnUpdateFname();
        virtual BOOL OnInitDialog();
        //}}AFX_MSG
        DECLARE_MESSAGE_MAP()
};

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ は前行の直前に追加の宣言を挿入します。

#endif // !defined(AFX_SVPSDLG_H__79A70BF1_7090_11D3_ABB8_00C04F8862AB__INCLUDED_)
```