筑波大学体育科学系紀要 Bull. Health & Sports Sciences, Univ. of Tsukuba 10：269—298, 1987.

バイオメカニクスの分析に適した汎用グラフィックプログラムChartの作成

宮地 力

Implementation of Chart: A Graphics Program Suitable For Biomechanical Analysis

Chikara MIYAJI

This study explains a program Chart. It generates device independent output, easy to use, and produce graphical output which is suitable for biomechanical analysis. Usually controle command of graphic device different from each other, so it is difficult to write a general purpose graphics program. This study shows one method to write device independent graphic progmam.

The results are summarized as follows,

1) To obtain device independency, program divided in two parts, main program and device driver. Chart generates Pseudo Graphic Code for drivers. Each driver interprets that codes and produces real device controle sequences.

2) Chart inputs are data and commands. Commands are mixed with data, this makes data handling very flexible. There are 21 Chart commands.

3) Minimum number of commands are chosen for Pseudo Graphic Code to keep device portability. 7 commands are enough for Chart output.

4) Chart is written in C language, so it works on most Unix machine, and some micro-computer which support C language.

5) Chart can be used asa filter command and also accepts complex command file for special graphic format. Especially @ex command, Chart works flexibly with other existent program.

1．はじめに

現在コンピュータソフトウェアとして多くのグラフィック用プログラムが市販されている。しかしその多くはビジネス用であったり，目的が限られており，すべての要求を満たすものは少ない。

バイオメカニクスの分野においても加速度，速度，変位などの時系列データのグラフィック表示が頻繁に利用されている。しかし，現在までの市販のプログラムには，汎用で他のプログラムとも結合しやすい時系列データのグラフィックプログラムは少ない。現在は，各研究者が独自にプログラムのなかにグラフィックのルーチンを埋め込み，出力を得ていることが多い。

しかし，この方法では，1つ1つの出力装置に対し別個にプログラムを作成することやグラフの書式の細部のためにプログラムの殆どを費やしていることが多い。これには，グラフィック出力装置の命令が統一化されていないこと，プログラム言語によってはプログラムの再利用が難しいことなどが考えられる。しかしバイオメカニクスの分野で取り扱うデータ量も膨大なものになってきており，データをプログラムを作らずにグラフィック化できるソフトウェアの必要性は増してきている。

本研究では，時系列分析に適し，出力装置を選ばずに統一的にグラフィックを取り扱えるプログ

ラムChartを作成した。プログラム作成に際しては，以下の方針を立てた。

1．出力装置を選ばずに同じ描図が得られる。
2．グラフィックの書式指定は省略可能であり，複雑な命令を全く知らなくても利用できる。
3．他のプログラムとの融合がとりやすく，パイプライン的利用が出来る。
4．時系列のデータの表示に適したフォーマットが得られる。

2．プログラムの装置独立性

グラフィックのプログラムの作成に関しての最も大きな問題の一つは，グラフィック出力装置が多種多様に渡り，その命令体系が装置毎に異なり，ほとんど統一がとれていない点である。そこで，おおくのプログラムは，XYプロッター用のもの，CRTディスプレー用のもの等々と，別個に作成されているのが一般である。またひとつのプログラムでいくつかの出力装置を設定できるとしても，その数は限られている。一つの大きな欠点は，新しい装置を使うときには，再度，ソースプログラムを書き直しコンパイルする必要がある点である。ソースの無いプログラムであれば，その装置は使えない事態にもなる。

このような各装置固有の命令体系の差異を消すために，実際の出力装置とプログラムのあいだにドライバーとよばれるプログラムを使う方法が，よく用いられている。プログラムは仮想命令を出力し，ドライバーがその仮想命令を解釈して実際の装置に対する命令を出力する。この方法では，新しい装置に対してもドライバーを作るだけで済むと言う利点がある。たとえば，汎用の文書フォーマッターTEXは，dviという中間コードを出力し，各装置のドライバーが実際の出力をする。

Chartにおいても，前述の方法を採った。Fig. 1に示すようにChartは仮想グラフィックコードを出力し，各出力装置に対する仮想グラフィックコードを解釈するドライバーが実際の出力をする。この方法によって，Chartは，おおくの出力装置にたいして全く同じグラフィックの結果を出すことができた。出力装置のドライバーは，7種類の異なるグラフィックCRTディスプレーと，2種類の異なるXYプロッター用のものがある。仮想グラフィックコードの仕様は必要最小限のものに限っているので，あたらしい出力装置にたいしてのドライバーも容易に作成できた。

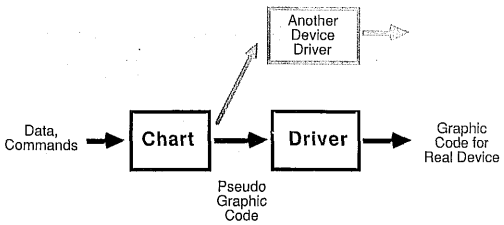

Fig. 1　Diagram for Data Flow.

3．Chartの言語仕様

時系列データのグラフィックプログラムでの入力は，基本的に，データと描画の書式指令の二つがある。その各々を別々に扱う方法もあるが，Chartでは，データの中に描画の書式指令も埋め込むと言う方法を採った。このやり方は，データの扱いが柔軟になる利点がある。これは，Unixの文書フォーマッターRoffなどに用いられている方法である。

Chartの言語仕様をBNF記法により表したものをFig. 2に示す。Chartの入力は行で区切られ，コマンド行と呼ばれるChartに対する命令の行と，データ行と呼ばれるデータの並びと，コメント行がある。

#で始まる行がコメント行であり，その1行はChartに無視される。

データ行は，浮動小数点表示の数字の並びである。1行に1つの数字のデータの場合，時系列データのYとし，Xは時間として0から順にインクリメントした値が対応する。2つのデータの場合，そのままXとY座標の値となる。

@で始まる行がコマンド行であり，その後に続くアルファベット2文字で1つのコマンドを示す。コマンドにはいくつかのアーギュメントがつく場合もある。

コマンドは大別すると，描画フレーム設定コマンド群，タイトル指定コマンド群，入出力・実行コマンド群，描画コントロールコマンド群とその他のコマンド群に分けられる。

コマンドの仕様はFig. 3にまとめた。コマンドは全部で21種類ある。しかし，これはプログラム作成の当初よりあったものではない。これは，少ないコマンドから始めそれを使用者の要望などか

```
Program         ::=   Statement '\n'
                  |   Statement '\n' Program
                  ;

Statement       ::=   Comment-line
                  |   Data-line
                  |   Command-line
                  ;

Comment-line    ::=   '#' Strings
                  ;

Data-line       ::=   Floating-point-number
                  |   Floating-point-number  Floating-point-number
                  ;

Command-line    ::=   '@' Cmd Option
                  ;

Cmd             ::=   'ti'   | 'xt'  | 'yt'  | 'xl'  | 'yl'  | 'xh'
                  |   'yh'   | 'sh'  | 'ex'  | 'go'  | 'll'  | 'ss'
                  |   'po'   | 'fr'  | 'er'  | 'cs'  | 'lo'  | 'hi'
                  |   'xw'   | 'yw'  | 'so'
                  ;

Option          ::=   Strings | Integer-number
                  |   Integer-number  Integer-number
                  |   '*'
                  |   File-name
                  ;
```

Fig. 2　Chart input syntax specification.

GROUP	COMMAND NAME AND OPTION	ACTION
Title Setting	@ti *str*	Set main title above frame.
	@xt *str*	Set title of X axsis.
	@yt *str*	Set title of Y axsis.
	@xl *str*	Set left side title of X axsis.
	@xh *str*	Set right side title of X axsis.
	@yl *str*	Set lower side title of Y sxsis.
	@yh *str*	Set upper side title of Y axsis.
File and Execution	@sh *str*	Execute str as shell command.
	@ex *str*	Execute str and take that output as input of Chart.
	@so *filename*	Read from file as input.
	@so *	Read from standard input.
Drawing Controle	@go	Start drawing from new setting.
	@ll	Draw only data line.
	@ss	Start Drawing on previous setting.
	@fr	Draw only frame.
Frame	@lo *x y*	Set lower left corner of frame.
	@hi *x y*	Set upper right corner of frame.
	@xw *x1 x2*	Set X data range as frame size.
	@yw *y1 y2*	Set Y data range as frame size.
Misc.	@po *str*	Pause Chart execution, and print out str to terminal.
	@er	Erase screen.
	@cs *m n*	Set frame tick (x=m, y=n. Default=10)

Fig.3　Chart commands and its action.

ら少しずつ増やしていったものである。

4．仮想グラフィックコードの仕様

Chartは装置独立性を持たせるため仮想グラフィックコードを出力する。この仮想グラフィックコードをドライバーが実際の出力装置への命令に解釈していく。出力装置は数多くあるため，仮想グラフィックコードはすべてのグラフィック出力装置に共通する命令をとり，特殊な機能は省かざるを得ない。例えば，XYプロッターで何種類もの大きさの字体を指定できてもCRTディスプレーでは一般には1種類の字体しか使えない。そこで仮想グラフィックコードはChartを描くための必要最小限の命令に限定した。

Fig. 4にその命令を示した。命令は全部で7つある。仮想グラフィックコードもすべて英数字で構成されているので文字としてみることが出来る。これは，Chartプログラム，各出力装置のドライバーの作成，デバックを容易にした。また，この仮想グラフィックコードは他のプログラムにも利用されている。

5．Chartの引数と実行時の展開

Chartは実行に際し以下のようなオプション指定が出来る。

Chart[[-n string] ...] [file....]

引数指定がなければ標準入力から入力をとってくる。Chartは標準出力に仮想グラフィックコードを出力する。そこで，Chartの出力はパイプラインを用いてドライバーに渡され，ドライバーの出力はリダイレクトを用いて実際の出力装置に出される。典型的なコマンドラインは以下のようになる。

```
Chart -1 'Subj. 1' file. ct | wx4636 >/dev/tty 9
        command option   Driver   Real Device
```

[[-n string] ...] は実行時展開の指令で，nは0－9の数字である。この指定の後につづくコマンドファイルのなかにある%nを指定文字列に展開する機能を持つ。このオプションによって，Chartコマンドファイルを書き換えることなくファイルの内容を変更できる。この指定によりコ

COMMAND	ACTION
M*x y*	Move to *(x y)*.
D*x1 y2*	Draw line from *(x1 y1) to (x2 y2)*...
x2 y2	
...	
(Empty line)	
P*x y*	Point at *(x y)*.
L*x1 y1 x2 y2*	Line from *(x1 y1) to (x2 y2)*.
E	Erase screen.
S*x1 y1 x2 y2*	Set coordinate.
T*str*	Print *str* from current location.

All numbers are represented as integer.

Fig. 4　Specification of Pseudo Graphic Code

マンドファイルをテンプレートとして使うことが出来る。

[file...] はいくつかのChartコマンドファイルであり順番にChartに読み込まれる。

6．Chartの移植性

Chart，ドライバー共にC言語を用いて作成した。Chartは定義ファイルをふくめて800行程度，各ドライバーは300行程度になる。どちらも標準的な関数を利用しており，パイプラインを呼び出す@exコマンド等のほかはオペレーティングシステムに依存する関数も少ない。ソースプログラムを，付録のA－Cに示した。

Chartは，はじめソード社のM685上で作られた。しかし他の機械でもUNIXが動くものであれば稼働する。また，OS-9上でもデータ配列の上限を小さくするだけで使用できた。

7．簡単な使用法

Chartは全くの初心者にも簡単に使えるように設計されている。例えばFig. 5に示すデータがあるとき，このデータ例をグラフ化するためには，以下のコマンドで良い。

Chart infile | driver>/dev/device

このコマンドで，Fig. 6と同じグラフが希望の出力装置にえられる。Chartは描図の指定が全く無くても適切なデフォルト値を採ることで図を描く。フレームの位置は画面のほぼ中央に，上限と

```
$ cat file1            ...Listing Command on Shell.
6.5
10.1
12.3
4.3
2.8                    ...This is the contents of file1.
0.8
4.2
3.1
4.7
8.1
9.0

$ chart file1 | tek >/dev/tty3        ...Simple command line for Chart on Shell.
```

Fig. 5　An example data

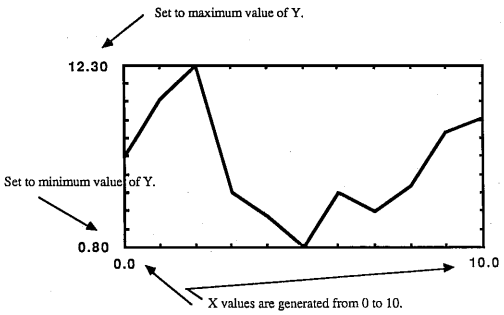

Fig. 6　An example graphic output

下限はデータの最大値最小値に設定される。

また，Chartをたんなるフィルターコマンドとしても利用することが出来る。例えば，あるプログラムprogがあり，標準出力にデータを出力する。これをグラフ化するには，その出力をChartに送るだけで良い。

prog | chart | driver>/dev/device

このように，データ処理プログラムとグラフィック処理を分離することが出来る。そのため，プログラム作成にあたっても必要なデータ処理に集中することができる。

8．Chartコマンドファイル

Chartは，コマンドを利用して，複雑な指定をすることも出来る。このようなコマンド例のファイルをコマンドファイルと呼ぶ。Fig. 7は，このようなコマンドファイルの1例である。描図に関する指示，データファイルの所在，コメントによる細かい説明を1つのコマンドファイルにまとめることでデータの管理も容易になる。Fig. 8にFig. 7の出力結果と，関連するコマンドを示した。また，@exコマンドを使うことで，Chartに他のプラムの出力を取り込むことが出来る。あるデータに数

値フィルターをほどこし，それを微分したものをグラフィク化することも@exコマンドでコマンドファイル中で出来る。

@ex filter＜data ｜ differential

Chartはフィルターとして使うことができ，コマンドファイルとしても使える。そして，既存のプログラムとも柔軟に組み合わせることが出来る利点がある。

9．ま と め

本研究では，出力装置を選ばずに統一的にグラフィックを取り扱え，使いやすく，時系列分析に適したプログラムChartを作成した。グラフィック出力装置は多種多様に渡り，その命令体系が装置毎に異なり，ほとんど統一がとれていない。このような各装置固有の命令体系の差異を消すために，ドライバーとよばれるプログラムを使う方法を用いた。結果を以下にまとめた。

1）プログラムの装置独立性を得るために，プログラムは，本体とドライバーの2つの部分に分けた。Chartは仮想グラフィックコードを出力し，ドライバーがその命令を解釈して実際の装置に対する命令を出力する。

2）データと描画の書式指令の二つを，Chartでは，データの中に埋め込むと言う方法を採った。Chartには，21のコマンドがある。

3）仮想グラフィックコードは装置独立性を得るためChartを描くための必要最小限の7命令に限定した。

4）Chartも，ドライバー共にC言語を用いて作成した。UNIXが動く機械であれば稼働する。また，マイクロコンピュータでもC言語があれば稼働する。

5）Chartはフィルターとして使うことができ，複雑な書式指定のグラフィックもコマンドファイルとしても使える。また@exコマンドによって，既存のプログラムとも柔軟に組み合わせることが出来る利点がある。

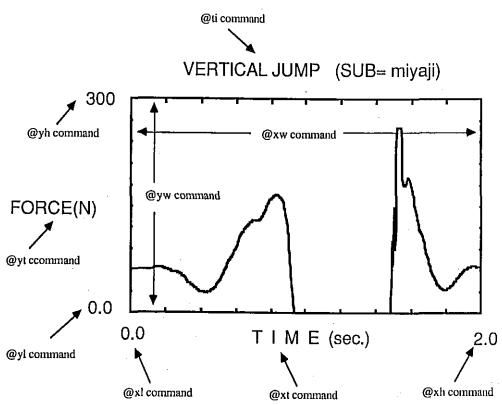

Fig. 7　Complex example for command file

```
$ cat file.ct                              ...Listing command for Shell
#
#       This chart command file            ...This is a comment line.
#       generated vertical jump Force Curve
#
@ti VERTICAL JUMP (SUB= %1)     ...Set title. %1 is replaced to some strings when
#                                  Chart start.
#
#       Set titles.
@xt T I M E (sec.)              ...Set title for X-axsis and Y-axsis.
@yt FORCE(N)
#
#       Set axsis values.
@yl 0.0
@yh 300
@xl 0.0
@xh 2.0
#
#       Get calibrated data using awk command ...Using @ex command.
#
@ex awk '{print 4.23 * $1 - 100.0 }' < %2
#
#       Start drawing
@go
#
#       end of chart
```

Fig. 8　Output of Fig. 7

References

1）Bourne, S. R.: The Unix System, Addison-Wesley, Reading, Mass, 1982

2）Kerhighan, B. W. and Plauger, P. J.: Software Tools in Pascal, Addison-Wesley, Reading, Mass., 227-264, 1981

3）Kernighan, B. W. and Pike, R.: The Unix Programming Environment, Printice-Hall., Englewood Cliffs, N. J., 1984

4）Kernighan, B. W. and Ritchie, D. M.: The C Programming Language, Prentice-Hall, Englewood Cliffs, N. J., 179-219, 1978

5）Knuth, D. E.: TEX and METAFONT, Digital Press, 1979

6）Ossana, J. F.: Nroff/Troff User's Manual, Colt Rinehart and Winston, New York, 196-229, 1983

Appendix A.
Header File Chart.h

```
#define MAXSIZE 4096
#define MAXSTR   256
#define AHX 1000
#define AHY  600
#define YES 1
#define NO  0
#define DEFAULT  2
#define COMMAND  '@'
#define COMMENT  '#'
#define ARGUMENT '%'
/* define command name */
#define UNKNOWN  -1
#define SO  0
#define LO  1
#define HI  2
#define XW  3
#define YW  4
#define GO  5
#define XT  6
#define YT  7
#define TI  8
#define EX  9
#define XL  10
#define XH  11
#define YL  12
#define YH  13
#define LL  14
#define SS  15
#define FR  16
#define PO  17
#define ER  18
#define CS  19
#define SH  20
#define LT  21
#define LC  23
#define DT  22
```

Appendix B.
Source File Chart.c

```
/* chart
 *
 */
#define DEFLT  0
#define DEFLC  0
#include <stdio.h>
#include <ctype.h>
#include "chart.h"
#define DOUBLE_MAX    1.0e38
#define DOUBLE_MIN    -1.0e38
#define skipsp(x)   {while(*(x)==' '||*(x)=='\t'||*(x)==',')++(x);}
#define skipbl(x)   {while(*(x)==' '||*(x)=='\t')++(x);}
#define skipcom(x)    {while(*(x)!=' '&&*(x)!='\t'&&*(x)!='\n')++(x);}
#define max(x,y)    (((x)>(y))?(x):(y))
#define min(x,y)    (((x)<(y))?(x):(y))


/* global definition */
double  xdata[MAXSIZE],
  ydata[MAXSIZE],
  xdata_max = DOUBLE_MIN,
  xdata_min = DOUBLE_MAX,
  ydata_max = DOUBLE_MIN,
  ydata_min = DOUBLE_MAX;
double  lowPx,lowPy,       /* plot area */
  highPx,highPy,
  low_x,low_y,
  high_x,high_y;
char  title[MAXSTR],
  x_title[MAXSTR],
  y_title[MAXSTR],
  xl_title[MAXSTR],
  xh_title[MAXSTR],
  yl_title[MAXSTR],
  yh_title[MAXSTR],
  *r[10];     /* strings registers */
int nx = 10, ny = 10;
int lt_num = 0, dt_num = 0;      /* line type and dot type */
int lc_num = 0;      /* line color type */
int num;
int x_set = NO, y_set = NO;      /* flag for width setting */
int xl_set = DEFAULT, xh_set = DEFAULT;       /* flag for x axsis value setting */
int yl_set = DEFAULT, yh_set = DEFAULT;       /* flag for y axsis value setting */
int dt_set = NO;     /* flag for lt and dt */
int lt_set = NO, lc_set = NO;

main(ac,av)
int ac; char *av[];
{
  FILE *fp;
  int i;

  init();
  if(ac == 1)
    process(stdin);
  else
    for(i = 1; i < ac; ++i){
      if(av[i][0] == '-')     /* option */
        switch(av[i][1]){
          case '0':
          case '1':
          case '2':
          case '3':
          case '4':
```

```
            case '5':
            case '6':
            case '7':
            case '8':
            case '9':
              r[av[i][1]-'0'] = av[i+1];
              i++;
              break;
            default:
              comerr("unknoen option:%c\n",av[i][1]);
              exit(1);
          }
        else {
          if((fp = fopen(av[i],"r")) == NULL)
            comerr("can't open %s\n",av[i]);
          process(fp);
          fclose(fp);
        }
    }
  if(num != 0)     /* flush out data */
    go();
}

process(f)
FILE *f;
{
  char inbuf[MAXSTR], buf[MAXSTR];

  while(fgets(buf,MAXSTR-1,f) != NULL){
    expand(buf,inbuf);
    if(inbuf[0] == COMMAND)
      command(inbuf);
    else if(inbuf[0] != COMMENT)
      text(inbuf);
  }
}

init()
{
  lowPx = AHX/6;
  lowPy = AHY/6;
  highPx = (AHX*5)/6;
  highPy = (AHY*5)/6;
  title[0] = '\0';
  x_title[0] = '\0';
  y_title[0] = '\0';
  r[0] = NULL;
  r[1] = NULL;
  r[2] = NULL;
  r[3] = NULL;
  r[4] = NULL;
  r[5] = NULL;
  r[6] = NULL;
  r[7] = NULL;
  r[8] = NULL;
  r[9] = NULL;
  space(0,0,AHX,AHY);
}

text(buf)
char *buf;
{
  double x, y;
  int d;
```

```
  if(num > MAXSIZE){
    fprintf(stderr,"Chart:no data space\n");
    return;
  }
  if((d = getdata(buf,&x,&y)) < 0){
    fprintf(stderr,"Chart:data format error\n");
    return;
  }
  if(d == 0)       /* empty line */
    return;
  if(d == 1){   /* one data in a line */
    y = x;
    x = (double)num;
  }
  setdata(&xdata_max,&xdata_min,&xdata[num],x);
  setdata(&ydata_max,&ydata_min,&ydata[num],y);
  ++num;
}

getdata(buf,x,y)
char *buf;double *x,*y;
{
  double atof();

  skipbl(buf);
  if(*buf == '.'||*buf == '+'||*buf == '-'||isdigit(*buf))
    *x = atof(buf);
  else if(*buf == '\n'||*buf == '\0')
    return(0);   /* empty line */
  else
    return(-1);  /* no convertion */
  while(*buf != ' '&&*buf != '\t'&&*buf != ','&&*buf != '\n')
    ++buf;      /* eat up first data */
  skipsp(buf);
  if(*buf == '.'||*buf == '+' || *buf == '-' ||isdigit(*buf)){
    *y = atof(buf);
    return(2);   /* convert two data */
  }
  else
    return(1);   /* convert one data */
}


setdata(maxval,minval,d,x)
double *maxval,*minval,*d, x;
{
  if(*maxval < x)
    *maxval =x;
  if(*minval > x)
    *minval = x;
  *d = x;
}

command(buf)
char *buf;
{
  double x,y;
  int ct;
  char s[MAXSTR];

  if((ct = comtype(buf)) == UNKNOWN)
    return;
  skipcom(buf);
  switch(ct){
```

```
case  SO:
  source(buf);
  break;
case  LO:
  if(getdata(buf,&x,&y) == 2){
    lowPx = max(0,x);
    lowPy = max(0,y);
  }
  break;
case  HI:
  if(getdata(buf,&x,&y) == 2){
    highPx = min(AHX,x);
    highPy = min(AHY,y);
  }
  break;
case  XW:
  if(getdata(buf,&x,&y) == 2){
    low_x = x;
    high_x = y;
    x_set = YES;
  }
  else
    x_set = NO;
  break;
case  YW:
  if(getdata(buf,&x,&y) == 2){
    low_y = x;
    high_y = y;
    y_set = YES;
  }
  else
    y_set = NO;
  break;
case  GO:
  go();
  break;
case  EX:
  expipe(buf);
  break;
case  SH:
  excom(buf);
  break;
case  XT:
  getstr(buf,x_title);
  break;
case  XL:
  getstr(buf,xl_title);
  if(xl_title[0] == '\0')
    xl_set = NO;
  else
    xl_set = YES;
  break;
case  XH:
  getstr(buf,xh_title);
  if(xh_title[0] == '\0')
    xh_set = NO;
  else
    xh_set = YES;
  break;
case  YL:
  getstr(buf,yl_title);
  if(yl_title[0] == '\0')
    yl_set = NO;
  else
    yl_set = YES;
```

```
    break;
  case  YH:
    getstr(buf,yh_title);
    if(yh_title[0] == '\0')
      yh_set = NO;
    else
      yh_set = YES;
    break;
  case  CS:
    if(getdata(buf,&x,&y) == 2){
      nx = x;
      ny = y;
    }
    break;
  case  LL:
    wrdata();
    break;
  case  SS:
    sameset();
    break;
  case  FR:
    wrframe();
    break;
  case  YT:
    getstr(buf,y_title);
    break;
  case  TI:
    getstr(buf,title);
    break;
  case  LT:
    if(getstr(buf,s) != 0)
      lt_num = atoi(s);
    else
      lt_num = 0;
    lt_set = NO;
    break;
  case  DT:
    if(getstr(buf,s) != 0)
      dt_num = atoi(s);
    else
      dt_num = 0;
    dt_set = YES;
  case  LC:
    if(getstr(buf,s) != 0)
      lc_num = atoi(s);
    else
      lc_num = 0;
    lc_set = YES;
    break;
  case  PO:
    getstr(buf,s);
    pause(s);
    break;
  case  ER:
    erase();
    break;
  default:
    return;
  }
}

expand(s1,s2)
char *s1,*s2;
{
```

```
char *p;

while(*s1 != '\0' && *s1 != ARGUMENT)
   *s2++ = *s1++;
if(*s1 == '\0'){
   *s2 = '\0';
   return;
}
s1++;
if(isdigit(*s1)){
   p = r[*s1-'0'];
   while(*p != '\0')
        *s2++ = *p++;
     s1++;
  }
  else  /* ARGUMENT, but not expand it */
     *s2++ = ARGUMENT;
  expand(s1,s2);
}

comtype(buf)
char buf[];
{
  if(buf[1] == 's' && buf[2] == 'o')
     return(SO);
  else if(buf[1] == 'l' && buf[2] == 'o')
     return(LO);
  else if(buf[1] == 'h' && buf[2] == 'i')
     return(HI);
  else if(buf[1] == 'x' && buf[2] == 'w')
     return(XW);
  else if(buf[1] == 'y' && buf[2] == 'w')
     return(YW);
  else if(buf[1] == 'g' && buf[2] == 'o')
     return(GO);
  else if(buf[1] == 'e' && buf[2] == 'x')
     return(EX);
  else if(buf[1] == 's' && buf[2] == 'h')
     return(SH);
  else if(buf[1] == 'x' && buf[2] == 't')
     return(XT);
  else if(buf[1] == 'y' && buf[2] == 't')
     return(YT);
  else if(buf[1] == 't' && buf[2] == 'i')
     return(TI);
  else if(buf[1] == 'x' && buf[2] == 'l')
     return(XL);
  else if(buf[1] == 'x' && buf[2] == 'h')
     return(XH);
  else if(buf[1] == 'y' && buf[2] == 'l')
     return(YL);
  else if(buf[1] == 'y' && buf[2] == 'h')
     return(YH);
  else if(buf[1] == 'c' && buf[2] == 's')
     return(CS);
  else if(buf[1] == 'l' && buf[2] == 'l')
     return(LL);
  else if(buf[1] == 's' && buf[2] == 's')
     return(SS);
  else if(buf[1] == 'f' && buf[2] == 'r')
     return(FR);
  else if(buf[1] == 'l' && buf[2] == 't')
     return(LT);
  else if(buf[1] == 'l' && buf[2] == 'c')
```

```
    return(LC);
  else if(buf[1] == 'd' && buf[2] == 't')
    return(DT);
  else if(buf[1] == 'p' && buf[2] == 'o')
    return(PO);
  else if(buf[1] == 'e' && buf[2] == 'r')
    return(ER);
  else
    return(UNKNOWN);
}

source(buf)
char *buf;
{
  FILE *f;
  char s[MAXSTR];

  skipbl(buf);
  if(*buf == '*'&& iswhite(*(buf+1)))       /* read from stdin */
    process(stdin);
  else if(getstr(buf,s) > 0)
      if((f = fopen(s,"r")) != NULL)
        process(f);
}

getstr(s1,s2)    /* get string from s1 to s2 */
char *s1,*s2;
{
  int n = 0;

  skipbl(s1);
  while(*s1 != '\n' && *s1 != '\0'){
    ++n;
    *s2++ = *s1++;
  }
  *s2 = '\0';
  return(n);
}

go()
{
  if(x_set == NO){
    low_x = xdata_min;
    high_x = xdata_max;
  }
  if(y_set == NO){
    low_y = ydata_min;
    high_y = ydata_max;
  }
  wrframe();
  wrdata();
}

sameset()
{
  wrframe();
  wrdata();
}


expipe(s)
char *s;
{
  FILE *fp;
```

```
    skipbl(s);
    if((fp = popen(s,"r")) != NULL){
      process(fp);
      pclose(fp);
    }
}

excom(s)
char *s;
{
    skipbl(s);
    spawnl(stdin,stderr,stderr,"csh","-c", s, NULL);
}

wrframe()
{
    int x1,y1,x2,y2;
    int i;
    double dx,dy;

    linetype(DEFLT);
    linecolor(DEFLC);

    dx = (highPx-lowPx)/(double)nx;
    dy = (highPy-lowPy)/(double)ny;
    /* frame with ruler */
    y1 = lowPy;
    for(i = 0; i < nx; ++i){
      x1 = i * dx + lowPx;
      x2 = x1 + dx;
      y2 = y1 + (highPx-lowPy)/100.0;
      frsub(x1,y1,x2,y1,x2,y2);
    }
    x1 = highPx;
    for(i = 0; i < ny; ++i){
      y1 = i * dy + lowPy;
      y2 = y1 + dy;
      x2 = x1 - (highPy - lowPy)/100.0;
      frsub(x1,y1,x1,y2,x2,y2);
    }
    y1 = highPy;
    for(i = 0; i < nx; ++i){
      x1 = highPx - i * dx;
      x2 = x1 - dx;
      y2 = y1 - (highPx-lowPx)/100.0;
      frsub(x1,y1,x2,y1,x2,y2);
    }
    x1 = lowPx;
    for(i = 0; i < ny; ++i){
      y1 = highPy - i * dy;
      y2 = y1 - dy;
      x2 = x1 + (highPy-lowPy)/100.0;
      frsub(x1,y1,x1,y2,x2,y2);
    }
    wrxval();
    wryval();
    wrtitle();
    linetype(lt_num);
    linecolor(lc_num);

}

frsub(x1,y1,x2,y2,x3,y3)
int x1,y1,x2,y2,x3,y3;
```

```
{
  draw(x1,y1);
  cont(x2,y2);
  cont(x3,y3);
  stop();
}

wrtitle()
{
  int x,y;

  if(title[0] != '\0'){
    x = (highPx+lowPx)/2.0 - strlen(title)*AHX/160.0;
    y = highPy + AHY * 1.0 / 25.0;
    move(x,y);
    label(title);
  }
}
wrxval()
{
  int x,y;

  if(xl_set == DEFAULT)
    sprintf(xl_title,"%6.1f",low_x);
  if(xh_set == DEFAULT)
    sprintf(xh_title,"%6.1f",high_x);

  x = lowPx - strlen(xl_title)*AHX/160.0;
  y = lowPy - AHY * 1.5 / 25.0;
  if(xl_set != NO){
    move(x,y);
    label(xl_title);
  }
  if(x_title[0] != '\0'){
    x = (highPx+lowPx)/2.0-strlen(x_title)*AHX/160.0;
    move(x,y);
    label(x_title);
  }
  x = highPx - strlen(xh_title)*AHX/160.0;
  if(xh_set != NO){
    move(x,y);
    label(xh_title);
  }
}


wryval()
{
  int x,y;

  if(yl_set == DEFAULT)
    sprintf(yl_title,"%6.2f",low_y);
  if(yh_set == DEFAULT)
    sprintf(yh_title,"%6.2f",high_y);

  y = lowPy - AHY / 50.0;
  x = lowPx - (strlen(yl_title)+2)*AHX/80.0;
  if(yl_set != NO){
    move(x,y);
    label(yl_title);
  }
  if(y_title[0] != '\0'){
    y = (lowPy+highPy)/2.0-AHY/50.0;
    x = lowPx - (strlen(y_title)+2)*AHX/80.0;
```

```
    move(x,y);
    label(y_title);
  }
  y = highPy - AHY / 50.0;
  x = lowPx - (strlen(yh_title)+2)*AHX/80.0;
  if(yh_set != NO){
    move(x,y);
    label(yh_title);
  }
}


wrdata()
{
  int px1, px2, py1, py2;
  double dx,dy;
  int i;

  dx = (highPx-lowPx)/(high_x-low_x);
  dy = (highPy-lowPy)/(high_y-low_y);
  px1 = (xdata[0]-low_x)*dx+lowPx;
  py1 = (ydata[0]-low_y)*dy+lowPy;
  if(dt_set == YES){
    dottype(dt_num);
    linecolor(lc_num);
    point(px1,py1);
    for(i = 1; i < num; ++i){
      px2 = (xdata[i]-low_x)*dx+lowPx;
      py2 = (ydata[i]-low_y)*dy+lowPy;
      point(px2,py2);
    }
  }
  else{
    linetype(lt_num);
    linecolor(lc_num);
    draw(px1,py1);
    for(i = 1; i < num; ++i){
      px2 = (xdata[i]-low_x)*dx+lowPx;
      py2 = (ydata[i]-low_y)*dy+lowPy;
      cont(px2,py2);
    }
    stop();
  }
  num = 0;    /* clear buffer */
  xdata_max = ydata_max = DOUBLE_MIN;
  xdata_min = ydata_min = DOUBLE_MAX;
  flushcom();
}

space(x1,y1,x2,y2)
int x1,y1,x2,y2;
{
  printf("S%d %d %d %d\n",x1,y1,x2,y2);
}

line(x1,y1,x2,y2)
int x1,y1,x2,y2;
{
  printf("L%d %d %d %d\n",x1,y1,x2,y2);
}

move(x,y)
int x,y;
{
```

```
  printf("M%d %d\n",x,y);
}

point(x,y)
int x,y;
{
  printf("P%d %d\n",x,y);
}

linetype(n)
int n;
{
  printf("J%d\n",n);
}

linecolor(n)
int n;
{
  printf("A%d\n",n);
}

dottype(n)
int n;
{
  printf("K%d\n",n);
}

label(s)
char *s;
{
  printf("T%s\n",s);
}

draw(x,y)
int x,y;
{
  printf("D%d %d\n",x,y);
}

cont(x,y)
int x,y;
{
  printf("%d %d\n",x,y);
}

stop()
{
  printf("\n");
}

pause(p)
char *p;
{
  FILE *f;
  char s[MAXSTR];

  flushcom();   /* flush out buffer to chang the paper */
  fprintf(stderr,"%s\n",p);
  f = fopen(ttyname(2),"r");
  fgets(s,MAXSTR,f);
  fclose(f);
}

flushcom()
```

```
{
  printf("!\n");
  fflush(stdout);
}

erase()
{
  printf("E\n");
}
```

Appendix C.
Driver Source File
for Watanabe WX4636
Driver.c

```
#include <stdio.h>
#define BUFSIZ   256

static int cx= 0,cy = 0, dtype = 0;
static float a = 1.0, b = 1.0, c = 0.0, d = 0.0;

main()
{
  init();
  process(stdin);
  end();
}

process(f)
FILE *f;
{
  char s[BUFSIZ];

  while(fgets(s,BUFSIZ-1,f) != NULL){
    switch(toupper(s[0])){
      case 'M':
        movecom(s);
        break;
      case 'D':
        drawcom(s,f);
        break;
      case 'P':
        pointcom(s);
        break;
      case 'L':
        linecom(s);
        break;
      case 'J':     /* line type set */
        ltcom(s);
        break;
      case 'K':
        dtcom(s);   /* dott type set */
        break;
      case 'E':
        erase();
        break;
      case 'I':
        include(s);
        break;
      case 'S':
        spacecom(s);
        break;
      case 'T':
        labelcom(s);
        break;
      case '!':
        flushcom();
        break;
      default:
        break;
    }
  }
}

movecom(s)
char *s;
{
  int x1,x2;

  sscanf(s+1,"%d%d",&x1,&x2);
```

```
  move(x1,x2);
}
ltcom(s)
char *s;
{
  int n;

  sscanf(s+1,"%d",&n);
  linetype(n);
}

dtcom(s)
char *s;
{
  int n;

  sscanf(s+1,"%d",&n);
  dottype(n);
}

drawcom(s,f)
char *s;
FILE *f;
{
  int x1,x2;

  sscanf(s+1,"%d%d",&x1,&x2);
  draw(x1,x2);
  for(;;){
    if(fgets(s,BUFSIZ-1,f) == NULL)
      break;
    if(s[0] == '\n')      /* empty line */
      break;
    sscanf(s,"%d%d",&x1,&x2);
    cont(x1,x2);
  }
  stop();      /* end of draw command */
}

pointcom(s)
char *s;
{
  int x1,x2;

  sscanf(s+1,"%d%d",&x1,&x2);
  point(x1,x2);
}


linecom(s)
char *s;
{
  int x1,x2,x3,x4;

  sscanf(s+1,"%d%d%d%d",&x1,&x2,&x3,&x4);
  line(x1,x2,x3,x4);
}

spacecom(s)
char *s;
{
  int x1,x2,x3,x4;

  sscanf(s+1,"%d%d%d%d",&x1,&x2,&x3,&x4);
```

```
  space(x1,x2,x3,x4);
}

include(s)
char *s;
{
  FILE *f;
  char *s1;

  s1 = s;
  while(*s1 != '\n' && *s1 != '\0')
    s1++;
  *s1 = '\0';
  if((f = fopen(s+1,"r")) != NULL){
    process(f);
    fclose(f);
  }
}

labelcom(s)
char *s;
{
  char *s1;

  s1 = s;
  while(*s1 != '\n' && *s1 != '\0')
    s1++;
  *s1 = '\0';
  label(s+1);
}


init()
{
}

end()
{
}

erase()
{
}

#define ASC 35   /* alpha scale */
#define ASP 50   /* alpha space */
label(s)
char *s;
{

  printf("M%d,%d\n",cx,cy+25);
        printf("S%d,Q%d,R%d,P%s%c",ASC,ASP,0,s,0x03);
}

move(x,y)
int x,y;
{
        cx=a*(x-c);
        cy=b*(y-d);
        printf("M%d,%d\n",cx,cy);
}

linetype(n)
int n;
{
```

```
  printf("L%d\n",n);
}

dottype(n)
int n;
{
  dtype = n;
}

draw(x,y)
int x,y;
{
  move(x,y);
        cx=a*(x-c);
        cy=b*(y-d);
        printf("D%d,%d",cx,cy);
}

cont(x,y)
int x,y;
{
        cx=a*(x-c);
        cy=b*(y-d);
        printf(",%d,%d",cx,cy);
}

space(x1,y1,x2,y2)
int x1,y1,x2,y2;
{
        a=(float)3800/(x2-x1);
        b=(float)2500/(y2-y1);
  c=x1;
  d=y1;
}

point(x,y)
int x,y;
{
  move(x,y);
  printf("N%d\n",dtype);
}

stop()
{
        printf("%c",0x03);
}

line(x1,y1,x2,y2)
int x1,y1,x2,y2;
{
  draw(x1,y1);
  cont(x2,y2);
  stop();
}

flushcom()
{
  fflush(stdout);
}
```