PROGRAMMING FOR THE TEKTRONIX 4115B ON THE VAX:

A TUTORIAL

Carla Wenzlaff
For Kent Stevens
December 12. 1984

INTRODUCTION

This tutorial is to explain how to use the Tektronix 4115B with the VAX as host. Section I explains the method of logging in, while section II describes the library facilities available. Section III describes the basic program format to use and Section IV tells how to compile a C program so that it includes the library routines. The Appendix contains the library routines and some examples of C programs written especially for the Tektronix 4115B.

Special features of the Tektronix 4115B include 256 possible colors, 8 possible graphic surfaces, and a resolution of 1280 by 1024 pixels. There are 4 possible color modes: HLS, RGB, CMY, and machine RGB. HLS is the default color mode. In RGB, the red, green, and blue values go from 0 to 100, while in machine RGB they go from 0 to 255. CMY is cyan, magenta, and yellow color mode.

I. GETTING STARTED

First, turn on the Tektronix 4115. This is done by pressing the "power on" switch located in the upper right-hand corner of the pedestal. The computer then goes through its power-up sequence and red lights flash on the keyboard. Once this process is finished, the cursor appears in the upper left-hand corner of the screen. In order to communicate textual information through the VAX properly, the computer must be in "text" mode. To set this, press the <SET UP> key which is located in the upper right-hand side of the keyboard. The cursor has now changed to an asterisk. Now type in "code ansi" and hit <RETURN>. You are now ready to log in to the VAX. Just press <SET UP> again to get out of "set up" mode, and press the <RETURN> key once more. Log in to the VAX by your normal procedure.

If you forget to set the computer to "code ansi" before logging in, you can still do it afterwards. Simply press the <SET UP> key and procede according to the directions above. "Set up" mode is local, and always takes precedence over any other mode.

You may now use your favorite editor to write a program for the Tektronix 4115B.

II. AVAILABLE LIBRARY ROUTINES

Many library routines written in C are available for use within graphic programs. These routines facilitate the writing of programs because all that is needed is the command and its parameters; the sending of escape sequences and translating of values is done through the library routine. The header file is in Appendix A; this lists all the routines which can currently be included within a program. The functions have been separated into 5 files, each of which hold related routines: device.c, gin.c, line.c, pixel.c, and seg.c. These libraries can be found in Appendix B. The correct parameters for each function can be found in the Tektronix <u>Series Command Reference Guide</u>.

Device.c contains the C routines which set specific modes and alter device settings on the 4115. In common with all of these functions is that each must send an escape sequence to the terminal for each command. The actual command is composed of 2 letters and each letter is converted to its "ansi" integer equivalent. The function "cmdout" then sends the terminal the escape sequence along with the commands in integer form. All commands must go through this routine. Similarly, any integer argument to a command must be translated into a form that the terminal understands, and sent to the 4115 via the "intout" routine. Two other frequently used routines included in device.c are "setans" which sets the terminal to "ansi" mode for text processing, and "settek" which returns the terminal to graphics mode.

The gin.c file contains the various gin functions. These routines enable the drawing and locating of points through the use of the thumbwheels.

Line.c is the library of functions which enable line drawing. This includes routines such as "moveto" which moves the graphics beam to (x,y) without drawing, and "drawto" which draws a line to (x,y). The line color can be set through use of the "lincol" function which needs the index of the color as its argument.

The library file pixel.c contains the functions to enable pixel operations. The "begpo" function should be called previous to any other routine included in this file; this is the "begin-pixel-operations" function. Commands to set a viewport, fill a rectangle with color, and send an array of colors to the terminal are included within this library.

The final library is seg.c, which contains various routines involving segments. Segments can be defined within a program so that each segment is treated as a unit and can be modified separately. To begin a segment, the function "sopen" is called, and to end a segment, call "sclose."

Many more operations can be performed on the Tektronix 4115B than are contained within these 5 libraries. Additional functions could make these libraries more complete, and not all of the current routines have been thoroughly tested. Whenever an addition or modification is made to any of the library files, 2 other alterations must be made. First, if a new routine was added, the header file must be ammended with the name of the new function in the appropriate location. Second, the files must be recompiled through the "make graphlib" command.

III. PROGRAMMING FOR THE TEKTRONIX 4115B

The C programming language is used to write programs for the Tektronix 4115B with the VAX as host. The standard conventions of C must be maintained; if input-output facilities are required then the standard C library must be included.

To incorporate the library functions described above, 1 of the first program statements must be "#include "tek4115.h"." This incorporates the header file in your program and assures access to the library routines.

Within the program, a statement which must precede any graphics commands is "settek()." This will set the terminal to graphics mode. Then, at the end of the program, it is a good idea to return the terminal to text mode with the "setans()" command.

The 3 commands described above are the only necessary ones you need include in your program. As was stated above, each routine requires its own specific parameters; for details see the Tektronix Series Command Reference Guide.

Some sample C programs which were written for the Tektronix 4115B can be seen in Appendix C. "Cline.c" is an example of an interactive program; it allows the user to input beginning and ending coordinates and then lines are drawn from the first point entered to the second (x,y) value. The program continues drawing lines until CTL-D is entered.

The program "rgb.c" uses the pixel operations to fill rectangles with color. Three sets of grids are drawn on the screen to display the RGB color scheme. The first row of grids explores the blue color value; blue is kept at a constant value in each grid, while red and green values are incremented. The blue value begins at 0 and increases by 25, until it is 100 in the final grid. The next 2 rows of grids explore the green and red color values in the same manner.

IV. COMPILING YOUR C PROGRAM

To compile your C program for the Tektronix 4115B, the compiler must be made aware of the library routines if they are included in your program. To do this, type "cc <filename>.c tek4115.a -lm", where <filename> is the name of your program. You need the "-lm" because some of the routines use the standard C math library. The executable file will then be in "a.out" as usual.

```
                /* TEK4115.H */


/* This is a header file for the library functions          */
/* for the Tektronix 4115B.  The file name at the           */
/* beginning of each group is where that set is located. */


        /*** DEVICE.C ***/

clear ();         /* clear-dialog-scroll                    */
cmap ();          /* set-surface-color-map                  */
cmdout ();        /* send escape sequence to terminal       */
cmode ();         /* set-color-mode                         */
defmac ();        /* define-macro                           */
drwmrk ();        /* draw-marker                            */
flush ();         /* flush-output-buffer                    */
intout ();        /* send terminal translated integer value */
mrkmod ();        /* enter-marker-mode                      */
mrktype ();       /* select-marker-type                     */
page ();          /* clear-graphics area                    */
sdef ();          /* set-surface-definitiions               */
setans ();        /* set mode to ansi for screen editing    */
settek ();        /* set mode to tek for graphics display   */
xy_coord ();      /* translate (x,y) coordinates            */


        /*** GIN.C ***/

gin();            /* enable-gin                             */
ginqry();         /* query-gin-position                     */
gint();           /* disable-gin                            */
ginpl();          /* get-gin-pick-report                    */
ginin();          /* read current gin location              */
gpckap();         /* set-gin-pick-aperture                  */
gincur();         /* gin-cursor                             */
intrpt();         /* interpret-report-form-terminal         */
reom();           /* report-frequency                       */


        /*** LINE.C ***/

begpan ();        /* begin-panel-fill-operations            */
drawto ();        /* draw vector from current beam pos to (x,y) */
endpan ();        /* end-panel-operations                   */
filpat ();        /* fill-pattern                           */
lincol ();        /* set-line-color                         */
linvup ();        /* set-window for line drawings           */
moveto ();        /* move-graphics-beam to (x,y)            */
```

```
        /*** PIXEL.C ***/

begpo();        /* begin-pixel-operations                     */
pixvup();       /* set-pixel-viewport                         */
recfil();       /* rectangle-fill                             */
raswrt();       /* pixel-raster-write (6 bits)                */
raswtb();       /* pixel-raster-write (8 bits)                */
rasrun();       /* raster-runlength                           */
spbpos();       /* set-pixel-beam-position                    */


        /*** SEG.C ***/

pickid();       /* pick-id                                    */
pivot();        /* set-pivot-point                            */
renew();        /* renew-view                                 */
skill();        /* delete-segment                             */
sopen();        /* begin-segment                              */
spost();        /* segment-visible                            */
segpos();       /* segment-position                           */
sclose();       /* end-segment                                */
```

```
                        /*  this is DEVICE.C */

#include    <sys/ioctl.h>
#include    <stdio.h>
#include    "tek4115.h"

        /********************************************************/
        /****     clear dialog area                         ****/
        /********************************************************/

        clear ()

        { int command [2];
          command [0] = 76;
          command [1] = 90;
          cmdout (command);
        }

        /********************************************************/
        /****     set surface color map                     ****/
        /********************************************************/

        cmap (surface,index,index1,index2,index3)
               int surface,
                  index,
                  index1,                 /*** indices may be ***/
                  index2,                 /*** hls, rgb or    ***/
                  index3;                 /*** machine rgb    ***/
        { int command [2];

          command [0] = 84;
          command [1] = 71;
          cmdout (command);
          intout (surface);
          intout (4);
          intout (index);
          intout (index1);
          intout (index2);
          intout (index3);
        }

       /********************************************************/
       /****   set surface definitions                     ****/
       /********************************************************/

       sdef (spec)
              int *spec;       /* points to array "spec"  */
       { int command [2],
             i, n;

       command [0] = 82;
       command [1] = 68;
       cmdout (command);
       intout (*spec);
n = *spec;                    /* value indicates number of     */
                              /* writing surfaces which follow */

for (i = 1; i <= n; i++)
        intout (*(spec + i));
}


 /************************************************/
 /****   send escape sequence to terminal     ****/
 /************************************************/

 cmdout (command)
         int command [];

 { printf ("%c%c",27,command[0]);
   if (command [0] > 31)
       printf ("%c",command[1]);
 }

 /******************************************************/
 /****   set color mode                            ****/
 /******************************************************/

 cmode (color_type,overlay,gray_mode)
        int color_type,
            overlay,
            gray_mode;

 { int command [2];

   command [0] = 84;
   command [1] = 77;
   cmdout (command);
   intout (color_type);            /**** hls, rgb, machine rgb ****/
   intout (overlay);               /**** opaque, sub, add      ****/
   intout (gray_mode);             /**** b/w or color          ****/
 }

/******************************************************/
/****  define macro                              ****/
/******************************************************/

defmac (number,length,string)
        int number,
            length;
        char string;
{ int command [2],
      i;
  command [0] = 75;
  command [1] = 68;
  cmdout (command);
  intout (number);
  intout (length);
  for (i = 0; i < length; ++i)
       intout (string++);
}
```

```
/*****************************************************/
/****   draw marker                              ****/
/*****************************************************/

drwmrk (x,y)
        int x,y;
{ int command [2];
  command [0] = 76;
  command [1] = 72;
  cmdout (command);
  xy_coord (x,y);
}

/*****************************************************/
/****   flush output buffer                      ****/
/*****************************************************/

flush ()

{ int i,n;
  long int ln;

  ioctl (0,FIONREAD,&ln);
  n = ln + stdin -> _cnt;
  for (i = 0; i < n; ++i)
      getchar ();
}
/*****************************************************/
/****   send translated integer value to terminal ****/
/*****************************************************/

intout (value)
        int value;

{ int abs_val,
      length = 1,
      string [10];

  abs_val = abs (value);
  if (value >= 0)
      string [0] = (abs_val & 15) | 48;
  else
      string [0] = (abs_val & 15) | 32;
  abs_val = abs_val >> 4;
  while (abs_val > 0)
        { string [length++] = (abs_val & 63) | 64;
          abs_val = abs_val >> 6;
        }
  while (length-- > 0)
        putchar (string [length]);
}

/*****************************************************/
/****     enter marker mode                      ****/
/*****************************************************/

mrkmod()

{ printf ("\034");
}
```

```
/*******************************************************/
/****     select marker type                     ****/
/*******************************************************/

mrktyp(type)
        int type;
{ int command [2];
  command [0] = command [1] = 77;
  cmdout (command);
  intout(type);
}


/*******************************************************/
/****     clear graphics area                    ****/
/*******************************************************/

page()

{ int command [2];
  command [0] = 12;
  cmdout (command);
}

/*******************************************************/
/****    set mode to ansi for screen editing     ****/
/*******************************************************/

setans ()

{ int command [2];
  command [0] = 37;
  command [1] = 33;
  cmdout (command);
  intout (1);
}

/*******************************************************/
/****   set mode to tek for graphics display     ****/
/*******************************************************/

settek()

{ int command [2];
  command [0] = 37;
  command [1] = 33;
  cmdout (command);
  intout (0);
}
```

```
/****************************************************/
/****  translate x-y coordinates into form      ****/
/****     recognized by 41xx                    ****/
/****************************************************/

xy_coord (x,y)
          int x,
              y;

  /*** previous hiy extra loy hix lox ***/

{ static int prev_xy [5] = {0,0,0,0,0};
  int coord [5],
      hiy,extra,loy,hix,lox,
      send_extra,
      i,
      length;

  length = 0;
  hiy   = ((y >> 7) & 31) | 32;
  extra = ((y & 3) << 2)  | (x & 3) | 96;
  loy   = ((y >> 2) & 31) | 96;
  hix   = ((x >> 7) & 31) | 32;
  lox   = ((x >> 2) & 31) | 64;

  if (hiy != prev_xy [0])
     { coord [length++] = hiy;
       prev_xy [0] = hiy;
     }
  send_extra = 0;
  if (extra != prev_xy [1])
     { coord [length++] = extra;
       prev_xy [1] = extra;
       ++send_extra;
     }
  if (loy != prev_xy [2] || send_extra || hix != prev_xy [3])
     { coord [length++] = loy;
       prev_xy [2] = loy;
     }
  if (hix != prev_xy [3])
     { coord [length++] = hix;
       prev_xy [3] = hix;
     }
  coord [length++] = lox;              /*** always send lox ***/
  prev_xy [4] = lox;

  for (i = 0; i < length; ++i)
       putchar (coord [i]);
}
```

```
                     /* this is GIN.C */

#include    "tek4115.h"

        /*********************************************************/
        /****    enable GIN                                ******/
        /*********************************************************/

        gin (device,function,events)
              int device,function,events;

         { int command [2],
              devfun;

           command[0] = 73;
           command[1] = 69;
           cmdout(command);

           devfun = (device << 3) | function;
           intout (devfun);
           intout (events);
          }

        /*********************************************************/
        /****    query GIN  position                       ******/
        /*********************************************************/

        ginqry (device,function)
              int device,function;

         { int command [2];
           int devfun;

          command[0] = 73;
          command[1] = 80;
          cmdout(command);

          devfun = 8 * device + function;
          intout (devfun);
          }

        /*********************************************************/
        /****    disable GIN                               ******/
        /*********************************************************/

        gint (device,function)
              int device,function;

         { int command [2];
           int devfun;

           command[0] = 73;
           command[1] = 68;
           cmdout(command);
```

```
     devfun = 8 * device + function;
     intout (devfun);
   }

/*********************************************************/
/****     get GIN pick raport                        ****/
/*********************************************************/

  ginp1 (keyi,x,y,segno,idno)
        int idno,segno,x,y,keyi;
  {
   char str[12];

   scanf("%12s",str);
   x = ((str[4] - 32) << 7) + ((str[5] - 32) << 2);
   y = ((str[1] - 32) << 7) + ((str[3] - 32) << 2) + ((str[2] >> 1)
        % 4);
   keyi = str[0];

   segno = intrpt(str[6]);
   idno  = intrpt(str[9]);
  }

/*********************************************************/
/****     READ CURRENT GIN location                  ****/
/*********************************************************/

  ginin (key,x,y)
        int *key,*x,*y;

  { char s [7];

    reom (0);
    scanf("%7c",s);

    *x = (s[4] & 31) << 7 | (s[5] & 31) << 2 | (s [2] & 2);
    *y = (s[1] & 31) << 7 | (s[3] & 31) << 2 | ((s [2] & 2) >> 2);
    *key = s [0];
    drwmrk (*x,*y);
  }


/*********************************************************/
/****     set gin pick aperture                      ****/
/*********************************************************/
 gpckap (win)
        int win;

 {int command[2];

  command[0] = 73;
  command[1] = 65;
  cmdout (command);
  intout (win);
 }
```

```
/**********************************************************/
/****    gin cursor                                   ****/
/**********************************************************/
gincur (dev,fun,curno)
       int dev,fun,curno;

{int command[2];
 int devfun;

 command[0] = 73;
 command[1] = 67;
 cmdout(command);
 devfun = 8 * dev + fun;
 intout (devfun);
 intout (curno);
}

/*****************************************************/
/****    interpret report from terminal          ****/
/*****************************************************/

intrpt (str)
        char *str;
{ return (((*str++ - 32) << 10) + ((*str++ - 32) << 4) +
       (*str & 15));;}

/***************************************************/
/****   report frequency (REOM)               ****/
/***************************************************/

reom (freq)
        int freq;

{ int command [2];
  command [0] = 73;
  command [1] = 77;
  cmdout (command);
  intout (freq);
}
```

```
                    /* this is file LINE.C */

#include "tek4115.h"


        /****************************************************/
        /****    begin panel fill operation            ****/
        /****************************************************/

        begpan (x,y,linestyle)
                int x,y,
                    linestyle;

        { int command [2];
          command [0] = 76;
          command [1] = 80;
          cmdout (command);
          xy_coord (x,y);
          intout  (linestyle);
        }


         /**********************************************************/
         /****     draw vector from current beam pos to x,y   ****/
         /**********************************************************/

         drawto (x,y)
                int x,
                    y;
         { int command [2];
           command [0] = 76;
           command [1] = 71;
           cmdout (command);
           xy_coord (x,y);
         }


         /*********************************************************/
         /****     end panel operations                     ****/
         /*********************************************************/

         endpan ()

         { int command [2];
           command [0] = 76;
           command [1] = 69;
           cmdout(command);
         }
```

```
/*****************************************************/
/****   fill pattern                           *****/
/*****************************************************/

filpat (pattrn)
        int pattrn;

{ int command [2];
  command [0] = 77;
  command [1] = 80;
  cmdout (command);
  intout (pattrn);
}

 /******************************************************/
 /****    set line color                         ****/
 /******************************************************/

 lincol (index)
         int index;

 { int command [2];
   command [0] = 77;
   command [1] = 76;
   cmdout (command);
   intout (index);
 }

 /******************************************************/
 /****     set window for line drawings          ****/
 /******************************************************/

 linvup (lox,loy,hix,hiy)
        int lox,
            loy,
            hix,
            hiy;
 { int command [2];
   command [0] = 82;
   command [1] = 87;
   cmdout (command);
   xy_coord (lox,loy);
   xy_coord (hix,hiy);
 }

 /******************************************************/
 /****    move graphics beam to x,y              ****/
 /******************************************************/

 moveto (x,y)
         int x,
             y;
 { int command [2];
   command [0] = 76;
   command [1] = 70;
   cmdout (command);
   xy_coord (x,y);
 }
```

```
                /* this is file PIXEL.C */

#include "tek4115.h"
#include <math.h>

        /*****************************************************/
        /****    begin pixel operations                  ****/
        /*****************************************************/

        begpo (surface,alu_mode,bits_per_pixel)
               int surface,
                   alu_mode,
                   bits_per_pixel;

        { int command [2];

          command [0] = 92;
          command [1] = 95;
          cmdout (command);
          intout (surface);
          intout (alu_mode);
          intout (bits_per_pixel);
        }

        /******************************************************/
        /****    set pixel viewport                       ****/
        /******************************************************/

        pixvup (lox,loy,hix,hiy)
                int lox,
                    loy,
                    hix,
                    hiy;
        { int command [2];

          command [0] = 82;
          command [1] = 83;
          cmdout   (command);
          xy_coord (lox,loy);
          xy_coord (hix,hiy);
        }

        /***************************************************/
        /****   pixel rectangel fill                   ****/
        /***************************************************/

        recfil (lox,loy,hix,hiy,index)
                int lox,loy,hix,hiy,index;

        { int command [2];

          command [0] = command [1] = 82;
          cmdout (command);
          xy_coord (lox,loy);
          xy_coord (hix,hiy);
          intout (index);
        }
```

```
/***********************************************************/
/****   pixel raster write                            ****/
/****      write array of color indices               ****/
/***********************************************************/

raswrt (pixels,color_code)
        int pixels;
        int color_code [];

{ int i,
      length,
      command [2],
      out_code [650];

  command [0] = 82;
  command [1] = 80;
  cmdout (command);
  intout (pixels);

  /*** encode 6 bits per pixel ***/

  for (i = 0; i < pixels; ++i)
       out_code [i] = color_code [i] + 32;
  intout (pixels);
  length = pixels;
  while (length-- > 0)
        putchar (out_code [length]);
}

/***********************************************************/
/****   pixel raster write (8 bits/pixel)             ****/
/****      write array of color indices               ****/
/***********************************************************/

raswtb (inar)
        int inar[3];

{ int i,
      command [2],
      out_code [4];

  command [0] = 82;
  command [1] = 90;
  cmdout (command);
  intout (3);
  intout (4);

  /*** encode 8 bits per pixel ***/
  out_code[0] = ((inar[0] & 252) >> 2) + 32;
  out_code[1] = (((inar[0] & 3)  << 4) | (inar[1] & 240) >> 4) + 32;
  out_code[2] = (((inar[1] & 15) << 2) | (inar[2] & 192) >> 6) + 32;
  out_code[3] = (inar[2] &  63) + 32;
  for (i = 0; i < 4; ++i)
      putchar (out_code [i]);
}
```

```
/**************************************************/
/****    raster runlength                     ****/
/**************************************************/

rasrun (pixels,color_index,bits_per_pixel)
        int pixels,
            color_index,
            bits_per_pixel;

{ int command [2],
      runcode;
  double pow ();

  runcode = pixels * pow ((double) 2,(double) bits_per_pixel)
            + color_index;
  command [0] = 82;
  command [1] = 76;
  cmdout (command);
  intout (1);
  intout (runcode);
}


/**************************************************/
/****     set pixel beam position              ****/
/**************************************************/

spbpos (x,y)
        int x,
            y;
{ int command [2];

  command [0] = 82;
  command [1] = 72;
  cmdout (command);
  xy_coord (x,y);
}
```

```
                            /* this is file SEG.C */

#include "tek4115.h"

         /*********************************************************/
         /****  Pick Id                                        *****/
         /*********************************************************/

          pickid (idno)
                 int idno;

          { int command[2];

            command[0] = 77;
            command[1] = 73;
            cmdout(command);

            intout (idno);
          }

          /********************************************************/
          /****     set pivot point                           ****/
          /********************************************************/

            pivot (x,y)
                   int x,y;
          { int command [2];
            command [0] = 83;
            command [1] = 80;
            cmdout (command);
            xy_coord (x,y);
          }

          /********************************************************/
          /****     renew view                                ****/
          /********************************************************/

          renew (view)
                 int view;

          { int command [2];
            command [0] = 75;
            command [1] = 78;

            cmdout(command);
            intout (view);
          }
```

```
/************************************************************/
/****     Delete Segment                                ****/
/************************************************************/

skill (segno)
       int segno;

{int command[2];

 command[0] = 83;
 command[1] = 75;
 cmdout(command);
 intout (segno);
}

/************************************************************/
/****     Begin Segment                                 ****/
/************************************************************/
sopen (segno)
       int segno;

{int command[2];

 command[0] = 83;
 command[1] = 69;
 cmdout(command);
 intout (segno);
}

/*************************************************************/
/***** Segment Visible                                      */
/*************************************************************/
/*  inputs: segno - the segment no
            vis   - 0 for invisible
                    1 for visivble       */

spost (segno,vis)
       int segno,vis;

{int command[2];

 command[0] = 83;
 command[1] = 86;
 cmdout(command);

 intout (segno);
 intout (vis);
 }
```

```
/************************************************************/
/****  Segment Position                               *****/
/************************************************************/

 segpos (segno,x,y)
        int segno,x,y;

  {int command[2];

  command[0] = 83;
  command[1] = 88;
  cmdout(command);

  intout (segno);

  xy_coord(x,y);
  }

  /************************************************************/
  /****  End Segment                                    *****/
  /************************************************************/

  sclose ()

 { int command[2];

   command[0] = 83;
   command[1] = 67;
   cmdout(command);
 }
```

```
                        /* CLINE.C */

#include "tek4115.h"
#include <stdio.h>

main(argc, argv)        /* A program to draw lines on the screen        */
                        /* given inputs: start (x,y) and end (x, y).   */
                        /* The program can be called with an index      */
                        /* indicating a color value.  Once the program*/
                        /* begins, the user can input coordinate        */
                        /* values for minx, miny, maxx, maxy.  To       */
                        /* exit the program, a CTRL-D must be entered.*/
int argc;
char *argv[];
{
int minx, miny,         /* minimum (x,y) coordinates */
    maxx, maxy,         /* maximum (x,y) coordinates */
    index,              /* line color index          */
    one = 1;            /* default line color is white */

        settek();
        if (argc != 2)              /* no color index entered */
                lincol(&one);
        else  {                     /* set line color */
                index = atoi(argv[1]);
                lincol(&index);

        }
        while (scanf("%d %d %d %d", &minx, &miny, &maxx, &maxy) != EOF)
        {                           /* continue getting coordinates */

                moveto(&minx, &miny);
                drawto(&maxx, &maxy);
        }
        setansi();
}               /* end main */

atoi(s)         /* converts character string to integer */
char s[];
{
int i, n, sign;
        for (i = 0; s[i] == ' ' || s[i] == '\n' || s[i] == 't'; i++)
                ;       /* skip white space */
        sign = 1;
        if (s[i] == '+' || s[i] == '-') /* sign */
                sign = (s[i++] == '+') ? 1 : -1;
        for (n = 0; s[i] >= '0' && s[i] <= '9'; i++)
                n = 10 * n + s[i] - '0';
        return(sign * n);
}       /* end atoi */
```

```
        /* RGB.C */

    /* This program explores the color system RGB.  Three */
    /* rows of grids are drawn across the screen.  Each   */
    /* row describes constant planes of one color value,  */
    /* while the other two color values are incremeted.   */

#include "tek4115.h"

#define CTYPE 1          /* color type = RGB                             */
#define OVRLY 1          /* color overlay mode = opaque                  */
#define GRY   1          /* set for color                                */

#define NSURF 1          /* surface for which color is defined           */
#define SURF  0          /* current surface                              */
#define ALU   0          /* no change in ALU mode                        */
#define BITS  0          /* current number of bits-per-pixel             */
#define BITPL 8          /* number of bit planes defined for surface     */
#define STEP  6          /* increment for colors                         */
#define XSTEP 50         /* increment of x-coordinate                    */
#define YSTEP 40         /* increment of y-coordinate                    */


main()
{
int x1 = 0,
    y1,
    x2 = 50,
    y2,
    x1start,
    x2start,
    index = 1,           /* color index                        */
    r,                   /* red color value                    */
    g,                   /* green color value                  */
    b,                   /* blue color value                   */
    spec[2],             /* array for set-surface definition */
    i, j;                /* index counters                     */


        settek();
        page();          /* clear the screen         */

        cmode(CTYPE, OVRLY, GRY);       /* set-color-mode */

        spec[0] = 1;            /* define array for set-surface-definition */
        spec[1] = BITPL;


        begpo(SURF, ALU, BITS);                 /* begin pixel operations */

        for (b = 0; b <= 100; b += 25)  {       /* for each rectangle, keep */
                x1start = x1 + 10;              /* blue constant while in-  */
                x2start = x2 + 10;              /* crementing red and green */
                y1 = 10;
                y2 = 50;
```

```
		r = 0;
		g = 0;

		for (i = 1; i <= 4; i++)  {
			x1 = x1start;
			x2 = x2start;

			for (j = 1; j <= 4; j++)  {
				cmap(NSURF, index, r, g, b);
						/* set-surface-color-map */
				recfil(x1, y1, x2, y2, index++);
						/* rectangle fill */
				r += STEP;	/* increment red */
				g += STEP;	/* increment green */
				x1 += XSTEP;
				x2 += XSTEP;
			}
			y1 += YSTEP;
			y2 += YSTEP;


		}

	}

	x1 = 0; x2 = 50;
	for (g = 0; g <= 100; g += 25)  {	/* keep green constant */
		x1start = x1 + 10;		/* while incrementing  */
		x2start = x2 + 10;		/* red and blue        */
		y1 = 230;
		y2 = 270;

		r = 0;
		b = 0;

		for (i = 1; i <= 4; i++)  {
			x1 = x1start;
			x2 = x2start;
			for (j = 1; j <= 4; j++)  {
				cmap(NSURF, index, r, g, b);
				recfil(x1, y1, x2, y2, index++);
				r += STEP;	/* increment red  */
				b += STEP;	/* increment blue */
				x1 += XSTEP;
				x2 += XSTEP;
			}
			y1 += YSTEP;
			y2 += YSTEP;
		}
	}

	x1 = 0; x2 = 50;
	for (r = 0; r <= 100; r += 25)  {	/* keep red constant */
		x1start = x1 + 10;		/* while incrementing */
		x2start = x2 + 10;		/* blue and green     */
		y1 = 450;
		y2 = 490;
		g = 0;
		b = 0;
```

```
                for (i = 1; i <= 4; i++)  {
                        x1 = x1start;
                        x2 = x2start;

                        for (j = 1; j <= 4; j++)   {
                                cmap(NSURF, index, r, g, b);
                                recfil(x1, y1, x2, y2, index++);
                                g += STEP;       /* increment green */
                                b += STEP;       /* increment blue  */
                                x1 += XSTEP;
                                x2 += XSTEP;
                        }
                        y1 += YSTEP;
                        y2 += YSTEP;
                }
        }


        getchar();                      /* wait for carriage return */
        sdef(spec);                     /* set-surface-definition   */
                                        /* redefine color map and erase screen */

        setans();                       /* return to ansi mode */

}       /* end main */
```