CHAPTER : V
AN
INTERACTIVE DRAWING
PROGRAM

CHAPTER-V AN INTERACTIVE DRAWING PROGRAM

5.1 COMPUTER AIDED DRAWING AND COMPUTER AIDED DESIGN

Computer aided drawing (picture construction) forms an important part of most computer aided design systems. The systems having both drawing and designing enable designers to work in a two-dimensional or picture domain.[1] With such integrated drawing and designing systems, an electrical engineer can work with circuit diagrams and an architect with elevations, or other projections of buildings, rather than just numbers. A computer aided design software accepts the pictures as input and duduces certain information from it. It gives the output in required form either graphically or numerically.

5.2 APPLICATION DESCRIPTION

The interactive drawing program developed in this chapter provides the user a cross-hair[3] which can freely move on the screen, under the control of direction keys of keyboard. A line drawing can be built up on the screen using the command to draw a line segment whose length and direction are controlled from the keyboard. The interactive program also provides a command to draw rubber band rectangle, which displays a rectangle with one comer steered by the cursor and the diagonally opposite comer fixed. Commands are provided to draw a inductance (L), resistor (r),

capacitance (c) and a source of voltage. You can develop a circuit diagram from these components. There is a facility to give labels to these components, so that their values can be displayed. Moreover by the combination of rubber band rectangle or rubber band lines and text commands, one can draw block diagrams. The displays for column and row position are provided. As the cross-hair moves allover the screen, the displays show the current row and column position. A provision for changing the speed of the cross-hair is also kept. Commands for fine adjustments are also provided.

5.3 USER INTERFACE DESIGN

The user interface is the key to a smooth interactive application. A good user interface enhances the joy of using the package while a bad ones test the patience of the most patient users. The most common technique for invoking commands with a graphical input device is the menu. Menu is nothing but the representation of commands interms of several small pictures known as icons. The icons in the menu can be fixed, or they can appear and disappear from the screen to reflect changes in the set of applicable commands as the application proceeds.

There are some lessons that developers of the emerging young iconic system can learn from the oldest living one.[2] Graphics should not be too overwhelming to be processed

effectively (Iconic system is useful for limited number of icons). The iconic system should be in an orderly fashion, so that there will be less difficulty to remember them. A iconic system having bad pictures confuse the user. Keeping these points in mind, all the icons are not displayed on the screen. The user interface of the program implemented here consists of options for cross-hair, horizontal inductor, vertical inductor, horizontal capacitor, vertical capacitor, horizontal resistor and vertical resistor. The options for taking the hardcopy & labelling are also displayed.

5.4 CREATING LINE DRAWINGS

The program implemented here (refer to source code listings on page Number 147) draw rubber band style lines. The program allows us to drag lines on the screen using the arrow keys. A rubber band line is a line with one endpoint steered by the cursor and the other end fixed. This allow us to draw and move lines without erasing other objects. This type of drawing technique is employed by Computer Aided Design Programs.

Exclusive-or (XOR) logic is used for this purpose. The PC's graphics hardware allows pixels to be XORed; thus a pixel that intersects with another can be erased without removing the original pixel. Illustration is given on page number 143)

5.5 RUBBER BAND BOX

A rubber band box displays a rectangle with one corner steered by the cursor and the diagonally opposite corner fixed. A rubber band box changes size and shape as the user moves the cursor. With the text command i.e.'P', you can also introduce textual information within the box. The procedure of drawing rubber box/is illustrated on page number 144).

5.6 ELECTRONIC CIRCUIT DRAWING

With the program implemented here, one can draw circuit with components resistor, inductor and capacitor. A picking and dragging technique is implemented. This technique consists of picking any icon from the menu and dragging it to the desired position. The object is fixed at that place.

Here a slightly different approach is implemented. Instead of picking and dragging the full object, the cross-hair is moved to the desired position and a letter key corresponding to the particular object is pressed. The illustration of drawing a circuit diagram is given on page number 145).

5.7 DESCRIPTION OF SOFTWARE

The program listing is selfexplanatory as there are comments spread allover the listing functions are declared for vertical, resistor, inductor, capacitor and horizontal resistor, inductor, capacitor. Other functions are also declared for cross-hair generation, its location display, for down menu and its removal when necessary.

The 'main routine' part of the program mainly declare the variable types. The end point of line segment, the coordinates of the block are also defined. Flages for rubber band box and rubber band line are set to zero.

The 'graph init' module of the program initializes the graphics system. It calls the cursor creating function, sets the cursor speed and sets the text attributes. It also initializes the pointer for the characters.

After the 'graph init' part, the main loop starts. It has several subparts. First part is the initialisation of the loop at some initial location. It then waits for the key which corresponds to proper drawing operation. The 'band' and 'rubber box' subpart checks the flags and performs the appropriate X-OR plotting. The subpart 'rubber band box' draws the box when key 'B' is pressed. One corner is fixed while other corner will be decided by the direction keys. By pressing the key 'ESC' one can exit to DOS. The subpart 'step change' changes the speed of corsshair cursor. Implementation of rubber band line is done in the subpart 'Lines'. The 'text' module puts text on the screen wherever necessary. With key 'F5', the screen gets cleared. The functions in 'cursor movement' subpart are prototype in the file 'CUR-MOVE.H' which facilitates the cursor movement. (Refer to program listings on page number).

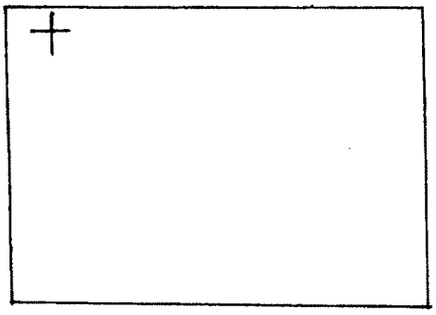
Fro the adjustment slow movement cursor facility is also kept. By pressing control key and direction key simultaneously, the components can be adjusted at the desired location in an accurate manner. The 'cursor position' subpart of the program displays the row and column positions of cursor.

The cross-hair cursor is actually drawn using line commands (relative). The 'create-cursor' part of the listing defines a function m-x_hair, which draws the cross-hair cursor. The 'down manu' module displays the available operations and the corresponding leys. The function x_hair locates the cross-hair cursor.

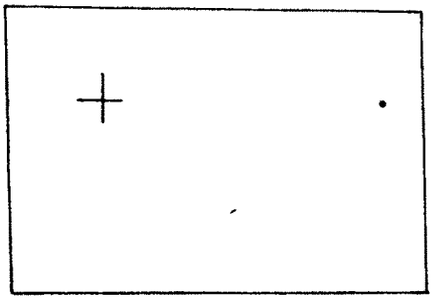
Description of Cursor Movement Header File

The CUR-MOVE.H is a file included in the main program. This file allows the left, right, up and down motion of cross hair cursor from current position. Facility for variation of speed of cursor is also kept.

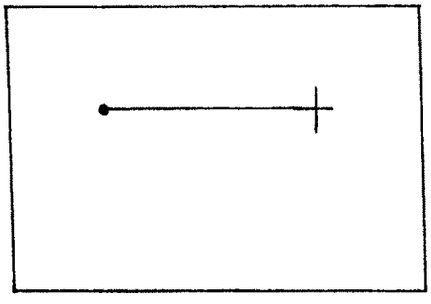
Both the main program LCD.C and header file CUR-MOVE.H have included the file defn.h (refer to page number 163). This header file define all the constants used in LCD.C and CUR-MOVE.H.



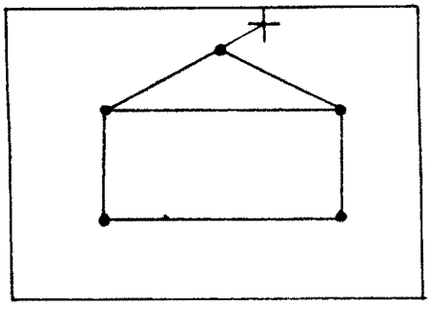
Start of program a cross-hair is provided.



Move cross-hair to desired point with direction keys and press L (This is first end point).

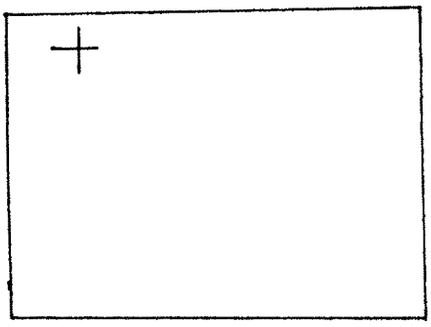


with direction keys, move cross-hair to new point and press 'L'.A line is generated.

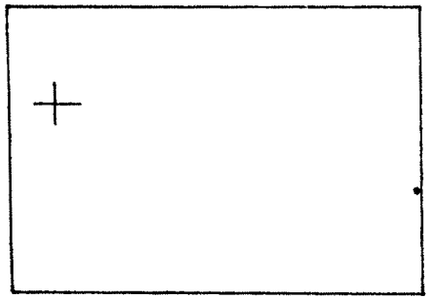


Thus any sequence of line can be drawn

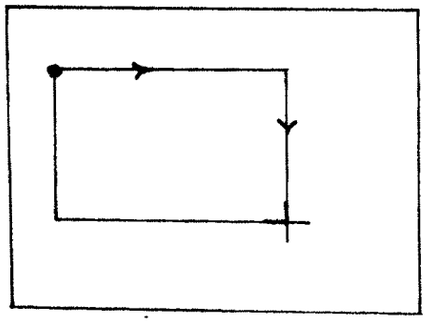
Fig.7 : Illustration of Rubber Band Line Drawing.



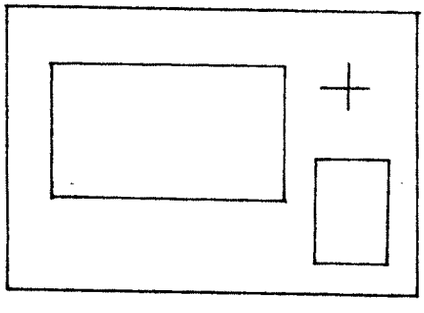
Cross-hair cursor is provided at the Start of the program



Move cross-hair at the desired point and press key 'B'.This will be one end point of the box.

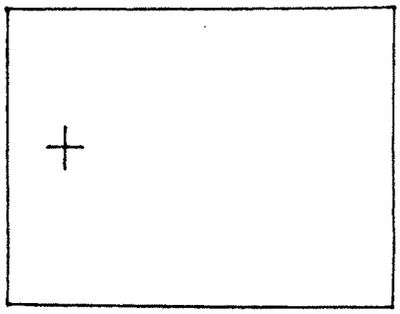


With direction keys move cross-hair at the diagonallyopposite end of tne box opposite end of the box (indicate direction key used)

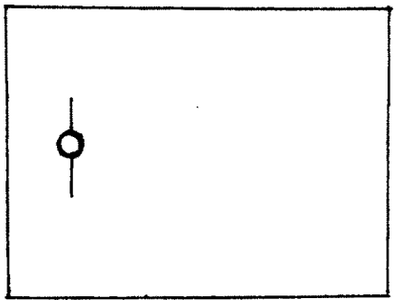


In this way you can generate several rubber band boxes. For introducing text go to proper location and press 'P' followed by text to be introduced

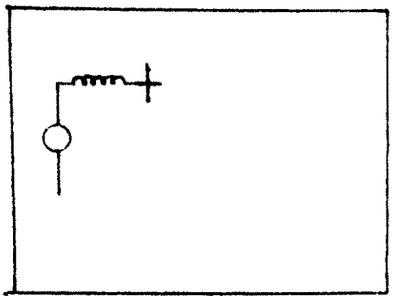
Fig.8 : Illustration of Rubber Band Box Drawing



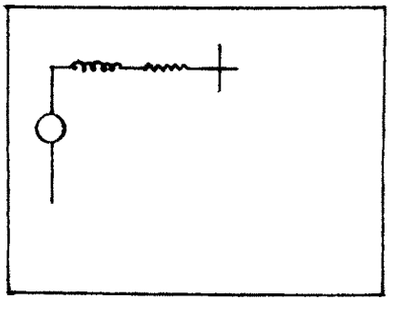
Moving the cross-hair at the deisred location with the help of direction keys



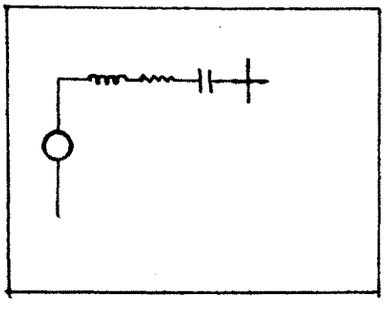
Press'S' a source appears at the location of cross-hair.Now press'ENTER and then 'n'.Source settles down and you can move cross-hair to another point By pressing ' , you can put text as 'a.c' or 'd.c'



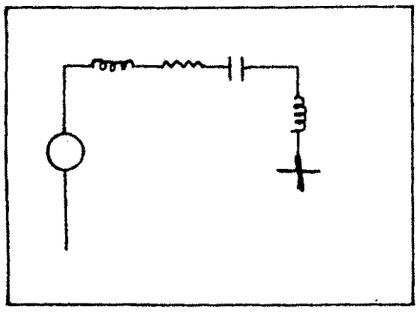
Press 'H',horizontal inductor appears, press 'CR' press 'n' and it settles down



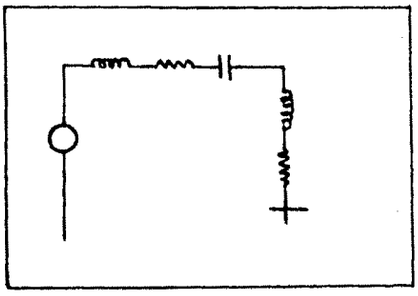
Press 'R' a horizontal resistor appears , and press 'enter' and 'n' to fix it and move cross hair to next location



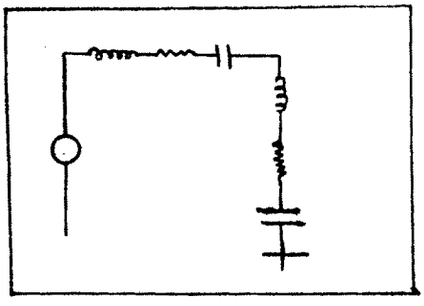
Press 'C' a horizontal capacitor appears press 'enter'and 'n' to fix and move cross-hair to next location.



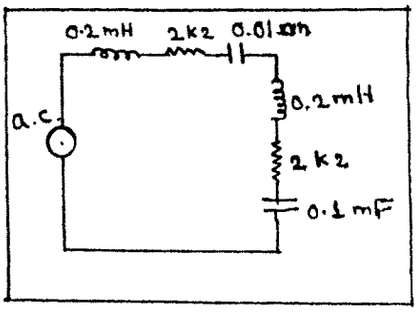
Press 'h' then 'enter' and 'n' a vertical inductor is drawn, move cross hair to next location.



Press 'r' followed by 'enter' and 'n' a vertical resistor is drawn, move cross-hair to next location.



Press 'C' followed by 'enter' and 'n' a vertical capacitor is drawn.



Complete the path from source by 'L' (line) command and put text using ' '

Fig.10 : Illustration of Electronic Circuit Drawing

```
/*  LCD.C           16-11-93  to   12-3-94*/
/* This includes initiaisation of the graphics */
 /*    mode.Initialisation of background color,     */
 /*     foreground color etc.the main switch - case */
 /*     loop etc.                                    */
 /*  THIS PROGRAM CAN BE ENJOYED MORE ON COLOUR
 /*  MONITOR                                         */
 /*  THIS  PROGRAM ILLUSTRATES RUBBER  BAND  LINE  ,*/
 /*  RUBBER BAND BOX, PICKING & DRAGGING TECHNIQUE. */
 /* ALSO ILLUSTRATES ICONIC APPROACH                */
  /*  PROGRAM  BEGINS HERE     */

   #include<graphics.h>
   #include<stdio.h >
   #include<conio.h >
   #include<stdlib.h>
   #include <math.h>
   #include <dos.h>
   #include "defn.h"
   #include  "cur_move.h"

         /* Cursor Movement  */
   #include "gprnt.h"


   union REGS r;
   static int x,y;   /* locations of the cursor*/
   static int colr = 7;/*foreground  color*/
   static int bk   = 4;/*   background  color*/
   static int box = 0;  /* flag -rubber box present
                              or absent */
   static int trace;  /*  trace  flag*/
   static int step;  /*speed of cursor*/

   void *move;
   void *hb;  /* For   Cursor xhair       */
   void *rh; /*   For  RES  HOR           */
   void *rv;
   void *lh;
   void *lv;   /* INDUCTOR /*
   void *cv;
   void *ch;    /* CAPACITOR /*
   void *bac;

   static char **menubegn;  /*   allocation  for
                                 menus     */
   static int  menu_no;


   /** F U N C T I O N S  I N  T H I S  F I L E**/

   void m_xhair(void);   /* creates  the  crosshair
                            cursor */
   void vertrest(void);
```

```
void makecap(void);
void induct(void);
 void vinduct(void);
void  xhair(int,int,void  *);  /*  locates   the
                              crosshair cursor   */

void menu_d(void);  /* display down menu */
void down_menu(void);
void remove_dn_menu(void);/* removes down menu*/
void batt(void);
void akshar(void);    /* put  text */
void rubber_box(int,int,int,int,int);
void   title(int,int,char   *);/*displays    the
                        title*/


/******* M A I N    R O U T I N E*********/

main()
{
 int drive,mode;
  char nmx[4],nmy[4];/*itoa for cur position   */
  int menu = 0;      /*menu page*/
  int radius;     /*  circles*/
  int cx = 300;
  int cy = 200;  /*  center of the circle */
  int  lx  =  20,ly = 20;/*   line  segment  end
                              points   */
  int mnno;      /*  menu item no                */
  int ans;     /*  decides when exiting       */
  int oeam; /*  color change                 */

  int xasp;  /*  Aspect ratio                 */
  int yasp;

  int dwn_scale=0 ;/*down scale*/
  int *px,*py;    /*  to push CP during moves */

  static int pcolor, style;/*in color section*/

  char *fname;  /*file name              */

  int  box = 0;  /* flag -rubber box present
                       or absent */

  int  x1  =  110; /* diagonal  corners  of  the
                          block*/
  int y1 = 110;
  int x2 = 120;
  int y2 = 120;
```

```
int  x3 = 100; /* CP when  calling  transforma
                  tions    */
int y3 = 100;

int  line_draw  = 0; /* flag - line  is  being
                        drawn          */

union inkey          /* to split the asci &
                            scan codes     */
æ  char ch[2A; /* returned by the bioskey()*/
   int i;
ä  c;


/* INITIALISATION  OF  GRAPHICS  MODE  */
 drive = DETECT;
 mode = 1;
 initgraph(&drive,&mode,"");


 /*    CREATE  A  CURSOR                    */
 m_xhair();
 makecap();
 cleardevice();
 induct();
 vinduct();
 vertrest();
 batt();

 menu_d();

/*    SET  CURSOR  SPEED                   */
step = 8;

/* SET TEXT  ATTRIBUTES                    */
setcolor(HOT);
settextstyle(DEFAULT_FONT,HORIZ_DIR,1);
outtextxy(5,8,"TRACE");
outtextxy(60,8,"OFF");
outtextxy(300,8,"X:");
outtextxy(400,8,"Y:");

rectangle(5,20,570,VGACHN+30);
trace = 0;


/* initialize  the pointer to char  */
 px = (int*)calloc(1,2);
 py = (int*)calloc(1,2);

 fname = (char *)calloc(10,1); /*  file name */
```

```
/********          MAIN LOOP**********************/

/*   INITIALISATION   of the loop  */

 x = 16;  y = 16;
 lx = x;   ly = y;
 move = hb;
 moveto(x,y);

/*  puts the cursor at x,y  */
 xhair(x,y,move);      /*   crosshair cursor   */
                        /* loop starts here    */
 setbkcolor(3);
 setcolor(1);

 for(;;)æ
   while(!bioskey(1));     /*  waits till a  key
                          is pressed */
   c.i  =  bioskey(0);    /*  gets the asci  &
                           scan code   */
   xhair(x,y,move);
                          /*  remove the cursor*/


 /*_______BAND_____________________________*/

  if(line_draw) /* if line is being drawn, the */
      æ         /*line drawn at the end of the*/
  outp(0x3ce,3);  /*loop with  XOR is removed */
  outp(0x3cf,0x18);/*by drawing over it  again */
  setcolor(0);
  moveto(lx,ly);    /*  using black    */
  lineto(x,y);
  setcolor(colr);

      å


  /*_____RUBBER  BOX_____________________*/

  if(box)          /*if a block is being marked*/
     æ         /*  the rectangle drawn at the  */
      rubber_box(x1,y1,x,y,0);
           /*  end of loop is removed.     */
     å


   /*  if key pressed is special key */
   /*  then the asci code is zero    */
   /*   for other keys the asic code  *
   /*  is returned as first byte     */
```

```
            if(c.ch[0A)  æ

              switch(c.ch[0A)  æ

              /*      CASE `t'            */

    /*  TRACE   SETS  TRACE  FLAG  17-11-93   >>>>>*/
/* moves CP to current position sets/resets TRACE flag
  now  when cursor is moved line is drawn from cp to new
    position                                          */

 case 'T':
 case 't':  if(line_draw00box) break;
                          /*if line or box is */
                          /*being made break  */
            trace = !trace;

            if(trace)
            æ
            setcolor(HOTBK);
            outtextxy(60,8,"OFF");
            setcolor(HOT);
            outtextxy(60,8," ON");
            å
            else
            æ
            setcolor(HOTBK);
            outtextxy(60,8," ON");
            setcolor(HOT);
            outtextxy(60,8,"OFF");
            å

            setcolor(colr);

            moveto(x,y);
            break;


       /* -----RUBBER BAND BOX------------ */
      /* BOX  flag is inverted. Then it is checked. if
         it  is one  the x1,y1 is assigned to the CP.  Now
         the blocks at the starting & end of the
         loop will draw the XORed rectangles taking  x1,y1
         & CP as the corners
         When  ENTER  is pressed again, x2,y2 will  be  as
         signed  the  CP. This will be the  marked  block.
         This block will saved in the current active
         buffer block in the RAM.*/

 case 'b':
 case 'B':   if(line_draw) break;
                        /*if line        is */
                       /*being made break  */
```

```
box = !box;

 if(box)
 æ
 x1 = x;
 y1 = y;
 å
 else
 æ
 x2 = x;
 y2 = y;
 å
 break;



/* ___ RECANGLE _______r _______4-3-94____*/
          •
    case 'A':
    case 'a':
    if(line_draw) break; /*if line        is */
    if(!box)      break   /*being made break  */
                  /* if box is not started */
     rectangle(x1,y1,x,y);


     box = !box;

     break;

 /*_ENTER_____________________________________*/


     case 13: if(boxøline_draw) break;
       if(move == hb)     break;
          putimage(x-10,y-5,move,COPY_PUT);
        break;

/*_______ CAP,RES ,IND  ____________________*/

      case 'r': move = rv; break;
      case 'R': move = rh; break;
      case 'c': move = cv; break;
      case 'C': move = ch; break;

      case 'n': move = hb;break;
      case 'h': move = lv;break;
      case  'H':  y = y - 5;
             /* to adjust position */
     move = lh;
     break;
```

```
        case 's': move = bac; break;


      /*________ PRINT ___ p _____________________ */
        case 'p' :
              ghardpict(2,20,600,290);
         break;


     /* ____EXIT TO DOS ____  Esc _________________*/

         case 27 : if(line_draw) æ
                   line_draw  = !line_draw;
                   break;
                   ä
          if(box)  æ
                   box  = !box;
                   break;
              .    ä
          remove_dn_menu();
          setcolor(HOT);
          outtextxy(10,YDW+5,"Really want to
                                    Exit  (Y/N) ?");
          setcolor(colr);

          ans = getch();
          remove_dn_menu();
          if( (ans == 'y') øø (ans == 'Y') )
                              æ
                              closegraph();
                              exit(0);
                              ä
          else
                  æ
                  setcolor(HOTBK);
          outtextxy(10,YDW+5,"Really want to
                                    Exit  (Y/N) ?");
                      down_menu();
                      break;
                  ä


     /* _____STEP CHANGE ____6-2-1994_____________*/

case '+': if(line_drawøøbox) break;
                           /*   if line or box is */
                           /*being made break  */
           step += 8 ;
           break;

case '-':  if(line_drawøøbox) break;
                            /*if line or box is */
                            /*being made break  */
```

```
            if(step <= 8) break;

              step -= 8 ;

              break;
          /*_____LINES_____20-11-93_____________*/

        /* draws line from CP to previos lx,ly, & assigns
        lx,ly values of CP                          */
        /*  modified  on   17-1-1994  */
        /* if line_draw is active   */

        case 'L':
        case  'l':  if(box)  break;
                /*if box is being made break  */

              if(line_draw)  lineto(lx,ly);
              if(!line_draw)
                 æ
                      lx = x;
                      ly = y;
                 å

              line_draw = !line_draw;
              moveto(x,y);
              break;

      /*   ___ TEXT  ____ x_________ 9-3-94 _________ */

    case 'X':
    case 'x': if(line_drawøøbox)  break;

              akshar();
              break;

                    å

                 å

              elseæ
                switch(c.chÆ1Å)  æ

        /* clear Screen _______ F 5 ------------*/

          case 63: if(line_drawøøbox) break;
          setcolor(WHITE);
          setfillstyle(EMPTY_FILL,0);
          floodfill(50,50,WHITE);
          setcolor(colr);
          break;
```

```
/*  CURSOR MOVEMENT_________                    _____________*/
    case 77: left();
    break;

    case 75: right();
    break;

    case 72: up();
    break;

    case 80: down();
    break;


      /*_____________________20_11_93_____________*/

       /*  left - up       Home    */
    case 71: left_up();
    break;    .

    /*    right -  up         PgUp */
    case 73: right_up();
    break;

   /*  right - down             PgDn */
     case 81: right_down();
     break;

       /* lift - down                  End */
     case 79: left_down();
    break;


  /* SLOW MOVEMENT OF CURSOR __  Ctrl Arrows ___*/


   case 0x73 : if(trace)linerel(1,0);
                                /* left */
        x--;
        if(x==15) x++;
        moveto(x,y);
        break;


        case  0x74 :  if(trace)linerel(1,0);
                                /* right */
        x++;
        if(x==600) x--;
        moveto(x,y);
        break;
```

```
    case  (119):  if(trace)linerel(1,0);
                  /* slow move up */
     y--;
     if(y==YTL) y++;
     moveto(x,y);
     break;

    case  (117):  if(trace)linerel(1,0);
                  /* slow move down */
     y++;
     if(y==330) y--;
     moveto(x,y);
     break;


      a
     a


    if(x<16) x=16;
    if(y<16) y=16;

/*_____BAND_____________________________*/

   if(line_draw)
    æ
        outp(0x3ce,3);
        outp(0x3cf,0x18);
        setcolor(1);
        lineto(lx,ly);
        moveto(x,y);
    a

  /*_____RUBBER  BOX_____________________*/

       if(box)
   æ
        rubber_box(x1,y1,x,y,1);
   a

      /*  HIGH LIGHT  */
       xhair(x,y,move);

   /*  DISPLAY  CURSOR  POSITION    */
        setcolor(HOTBK);
                 /* erase previous nos  */
        outtextxy(350,8,nmx);
        setcolor(HOTBK);
        outtextxy(450,8,nmy);

        setcolor(HOT);
           /*  now display new coordinates */
```

```
            itoa(x,nmx,10);
            outtextxy(350,8,nmx);

            itoa(y,nmy,10);
            outtextxy(450,8,nmy);

            moveto(x,y);
            setcolor(colr);
              /* change back  to original color */

               ä
            ä


/*  _  CROSS HAIR CURSOR __17-11-93_______*/
/*  CREATE  CURSOR  FUNCTION ___*/
/*  ALSO HORIZONTAL  RESISTOR  ___ */

 void m_xhair(void)
     æ
          int i;

          setcolor(1);
          line(0,10,30,10);
          line(15,0,15,20);
          hb = (char *)malloc(30*20);
          getimage(0,0,30,20,hb);
          putimage(0,0,hb,XOR_PUT);

          line(0,15,5,15);
          moveto(5,15);
          linerel(3,-5);
          linerel(5,11);
          linerel(5,-11);
          linerel(5,11);
          linerel(5,-11);
          linerel(5,11);
          linerel(3,-6);
          linerel(5,0);

          i = imagesize(0,10,40,22);
          rh = (void *)malloc(i);
          getimage(0,10,40,22,rh);
          putimage(0,10,rh,XOR_PUT);
           ä


     /*_________ RESIST VERTI _________________*/

          void vertrest(void)
             æ
                int i;
```

```
                line(8,0,8,4);
                moveto(8,4);
                linerel(8,2);
                linerel(-16,4);
                linerel(16,4);
                linerel(-16,4);
                linerel(8,2);
                linerel(0,4);

                i = imagesize(0,0,16,24);
                rv = (void *)malloc(i);
                getimage(0,0,16,24,rv);
                putimage(0,0,rv,XOR_PUT);
                ä

/*__ C VERICAL & HORIZONTAL_____________________ */
          void makecap(void)
               æ
                   int i;
                   line(10,2,10,5);
                        /* vertical  */
                   line(0,5,20,5);
                   line(0,12,20,12);
                   line(10,12,10,15);

                   i = imagesize(0,2,20,15);
                   cv = (void *)malloc(i);
                   getimage(0,2,20,15,cv);
                   putimage(0,2,cv,XOR_PUT);

                    line(3,5,6,5);
                        /* HORIZONTAL */
                    line(6,0,6,12);
                    line(16,0,16,12);
                    line(16,5,19,5);

                    i = imagesize(3,0,19,12);
                    ch = (void *)malloc(i);
                    getimage(3,0,19,12,ch);
                    putimage(3,0,ch,XOR_PUT);

                     ä


      /*_puts the cursor at said position_   */

       /*  DRAWN USING PUT IMAGE Fn.*/

  void xhair(int x,int y,void *pic)
   æ
   if(pic==hb) putimage(x - 15,y - 10,pic,XOR_PUT);
```

```
        else
       putimage(x - 10,y - 5,pic,XOR_PUT);  å

         /* _____ DOWN MENU ___  25-3-94_______ */
/*displayes the menu related to hot keys at the bottom/

     void down_menu(void)
        æ

         setcolor(HOTBK);
         rectangle(0,YDW,640,350);
         floodfill(2,YDW+2,HOTBK);

         setcolor(RED);
                  /* mark hot keys   */
         outtextxy(1*8,YDW+10,"F1");
         outtextxy(10*8,YDW+10,"F2");
         outtextxy(19*8,YDW+10,"F3");
         outtextxy(28*8,YDW+10,"CntrF3");
         outtextxy(44*8,YDW+10,"F10");
         outtextxy(54*8,YDW+10,"Cntr k");



        setcolor(HOT);      /* write menus       */

        outtextxy(3*8,YDW+10,"-Help");
        outtextxy(12*8,YDW+10,"-Save");
        outtextxy(21*8,YDW+10,"-Load");
        outtextxy(34*8,YDW+10,"-Overlap");
        outtextxy(47*8,YDW+10,"-Menu");
        outtextxy(60*8,YDW+10,"-Transformations");

        setcolor(colr);  /* restore the previous color*/

          å


         /*    REMOVES DOWN MENU */


        void remove_dn_menu(void)
            æ
              setcolor(0);      /* mark hot keys   */
              outtextxy(1*8,YDW+10,"F1");
              outtextxy(10*8,YDW+10,"F2");
              outtextxy(19*8,YDW+10,"F3");
              outtextxy(28*8,YDW+10,"CntrF3");
              outtextxy(44*8,YDW+10,"F10");
              outtextxy(54*8,YDW+10,"Cntr k");
              setcolor(0);      /* write menus    */
```

```
  outtextxy(3*8,YDW+10,"-Help");
  outtextxy(12*8,YDW+10,"-Save");
  outtextxy(21*8,YDW+10,"-Load");
  outtextxy(34*8,YDW+10,"-Overlap");
  outtextxy(47*8,YDW+10,"-Menu");
  outtextxy(60*8,YDW+10,"-Transformations");
  setcolor(colr);

  }


void rubber_box(x1,y1,x,y,f)
  int x1,y1,x,y,f;
  {
    if(f) setcolor(HOT);
     else setcolor(HOTBK);

    line(x1,y1,x1,y);
    line(x1,y,x,y);
    line(x,y,x,y1);
    line(x,y1,x1,y1);

    setcolor(colr);
     moveto(x,y);
  }



/*___VERTI      INDUCTOR ____________________________ */

    void vinduct(void)
    {
     int i;
     line(10,0,10,3);
     arc(10,10,210,90,8);
     for(i = 18; i < 30; i+=8)
             /* 18,26,34,42  */
     arc(10,i,210,150,8);

     arc(10,34,270,150,8);
     line(10,41,10,44);

      i = imagesize(0,0,20,44);
      lv = (void *)malloc(i);
      getimage(0,0,20,44,lv);
      putimage(0,0,lv,XOR_PUT);
       }
```

```
/*___ HORI        INDUCTOR _____________________________ */

          void induct(void)
            {
             int i;
             line(0,10,2,10);
             arc(10,10,-50,180,8);
             for(i= 20; i < 45; i+=10)
                   /* 18,26,34,42,50  */
             arc(i,10,-50,230,8);

             arc(50,10,0,230,8);
             line(58,10,62,10);

             i = imagesize(0,0,62,20);
             lh = (void *)malloc(i);
             getimage(0,0,62,20,lh);
             putimage(0,0,lh,XOR_PUT);

            }


          void batt(void)
            {
                int i;
                circle(20,20,15);
                i = imagesize(5,8,35,32);
                if(i == 0) exit(0);
                bac = malloc(i);
                getimage(5,8,35,32,bac);
                putimage(5,8,bac,XOR_PUT);
            }


          void menu_d(void)
            {
             putimage(XM,YM,hb,XOR_PUT);
             outtextxy(XM + 1,YM + 30,"n");

            putimage(XM + DF*1,YM,rv,XOR_PUT);
            outtextxy(XM + DF*1+5,YM + 30,"r");

            putimage(XM + DF*2,YM,rh,XOR_PUT);
            outtextxy(XM + DF*2+5,YM + 30,"R");

            putimage(XM + DF*3,YM,cv,XOR_PUT);
            outtextxy(XM + DF*3+5,YM + 30,"c");

            putimage(XM + DF*4,YM,ch,XOR_PUT);
            outtextxy(XM + DF*4+5,YM + 30,"C");
```

```
putimage(XM + DF*5,YM,lv,XOR_PUT);
outtextxy(XM + DF*5-10,YM + 30,"h");

putimage(XM + DF*6-14,YM,lh,XOR_PUT);
outtextxy(XM + DF*6+15,YM + 30,"H");

 putimage(XM + DF*7,YM,bac,XOR_PUT);
 outtextxy(XM + DF*7+5,YM + 30,"s");

putimage(XM + DF*7,YM,lh,XOR_PUT);
outtextxy(XM + DF*8,YM + 10,"PRINT");
outtextxy(XM + DF*8+5,YM + 30,"p");

 outtextxy(XM + DF*9,YM + 10,"TEXT");
 outtextxy(XM + DF*9+5,YM + 30,"p");

 å


void akshar(void)
    æ
   int i = -1;
    char vachÆ10A;

     doæ
       i++ ;
        vachÆiA = getch();
       å
while(! ( (vachÆiA == 27)ø(vachÆiA == 13))) ;
      i--;
  vachÆiA = '0';
  outtext(vach);
          å
```

```
/*  DEFN.H                  /*

/* LISTING OF DEFN.H        /*
/* USED IN LCD.C            /*
/* COMMENTS ARE PUT TO DEFINE THE PARAMETERS /*

#define XMLL    535   /* menu window locations  */
#define XMRL    640
#define YMTL    15
#define YMBL    350
#define  BLANK  "ÆÆÆÆÆÆÆÆÆÆÆÆÆÆ" /*used to remove  a
                                    string  */

#define XMENU   540 */* Menu location*/
#define YMENU   80                      /*  40  */
#define HOT      8  /* Color of menu items  */
#define HOTBK   15  /*  Menu background color  */
#define X_PROB  100  /*  Problem & Help location */
#define Y_PROB  100
#define P_WID   300   /* Width of problem window */
#define P_H     60   /* Height of problem window */
#define HELP_H  120   /* Height of help window*/
#define XZ      50   /*  position where zoomed*/
#define YZ      30    /*  image is to be drawn          */

#define XLL     0     /*  x left limit                   */
#define XRL     530 /*  x right limit                  */
#define YTL     15     /*  y top limit                     */
#define YBL     330    /*  x bottom limit               */

#define YDW     330   /*   down window position         */

#define PI      (22/7)
#define rad(x)  (x) * 0.0175
#define XB      8
#define YB      6
#define CLMNT   300
#define VGACHN  300
```

```
/* PROGRAM LISTING OF GPRNT.H */
/* A HEADER FILE DEVELOPED FOR TAKING PRINTOUT
   OF GRAPHIC IMAGES ON THE SCREEN */
/* FINAL VERSION OF PROGRAM COMPLETED
   ON :09-07-1994                    */

/*;;;;; PROGRAM STARTS HERE ;;;;;;;;*/

#include <dos.h>
#include <stdlib.h>
#include <stdio.h>

 /* FUNCTIONS  ARE  HERE  */
void ghardpict(int ,int ,int ,int );
unsigned char put_out(unsigned char );
unsigned char status(void);
unsigned char getdownbyte(int ,int );


/* _____ PRINT SCREEN _____*/

 void ghardpict(int x1,int y1,int x2,int y2)
  æ

    int x,y;
    int i;
    double xmax,ymax,xmin,ymin;
    double yw,asymax,asymin;
    int xasp,yasp;
    unsigned char pb; /* print byte*/
       union prnno æ
         unsigned int ydiff;
         unsigned char splitÆ2A;
                     å pr;

    xmax = max(x1,x2);  /* get top-left  bottom_right
                           corners */
    xmin = min(x1,x2);
    ymax = max(y1,y2);
    ymin = min(y1,y2);

    getaspectratio(&xasp,&yasp);


    asymax = ymax * yasp;    /* lengths in real
                                co_ordinates */

    asymin = ymin * yasp;

    pr.ydiff = (int)(  (asymax - asymin)/xasp  );
```

```
/*  PRINTER   initializations     */

 put_out(27);
 put_out(64);      /*  initialize    */

 put_out(27);
 put_out(65);
 put_out(8);  /*  line feed for 4 lines   */


for(x = xmin; x <= xmax; x += 8 )
   æ
    /* takes printer in geraphics mode     */

  put_out(27);
  put_out(42);     /*  ' ESC * '        */
  put_out(5);      /*  mode ' 5 ' plotter      */
  put_out(pr.splitÆ0A);  /*  no of  lines  */
  put_out(pr.splitÆ1A);


       /* yw is inc by xasp */

  for(yw = asymax; yw >= asymin  ; yw -= xasp)
   æ

       y = (int) (yw / yasp);

       pb  =  getdownbyte(x,y); /*  eight  pixels
                                  compressed */
        /* on a single byte          */
       put_out(pb);  /* out to the printer*/

   á

       put_out(13);   /*   CR & LINE FEED   */
       put_out(10);

    á


     á
/*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*/

/*___OUT TO PRINT!_____________________________________*/

unsigned char put_out(character)
unsigned char character;
æ
  union REGS r;

  while(!status());
```

```
                        /*if status is null,condn is
                          true & it will   be in the locp
                           till the status is any thing
                          other  than null. thus it is  a
                          wait till busy  */

        r.h.ah = 0;
        r.h.al = character;
        r.x.dx = 0;          /* select first printer */

        int86(0x17,&r,&r);
        return(r.h.ah);        /* returns status */
        æ

      /*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*/

    /*___GETS THE STATUS._____________________________________*/

       unsigned char status(void)
        æ
           union REGS r;

           r.h.ah = 2; /*  check the printer status  */
           r.x.dx = 0; /* SELECT THE FIRST PRINTER  */
           int86(0x17,&r,&r);
           return(r.h.ah & 0x80);
        æ
   /*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*/

/* ___  SUPPORTS DOWN LOAD_____________________________________*/
    /*  input: a co-ordinat of pixel;reads 8  consecutive
        pixels in a row creats one byte from them returrs
        this value                 */

        unsigned char getdownbyte(int x,int y)
        æ
          int color,i;
          unsigned char dbyte;
           unsigne char maskÆ8Å = æ
              0x80,0x40,0x20,0x10,0x08,0x04,0x02,0x01æ;


          /*   initialize  */

                dbyte = 0x00;
                              /* fill in the masks   */
                for(i = 0; i < 8 ; i++)
                æ
                 color = getpixel(x+i,y);
                  if(color)
                      æ
                      dbyte = dbyte ø maskÆiÅ;
                       æ
```

```
ä

    return dbyte;

ä
/*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*/


 /*_________ TEXT PRINT_____________________________*/

void cenprint(char *cench)
  æ
   int lc,nb,n;
   lc = strlen(cench);
   nb = 40 - lc/2;
     for(n = 0; n < nb; n++)
       fputc(' ',stdprn);
       fputs(cench,stdprn);
  ä
```

```
/* CUR_MOVE.H  print out   on      30-08-93*/
/* Header  file  for cursor movement */
   #include "defn.h"

/*FUNCTIONS  USED  FOR  CURSOR MOVEMENT */

 extern int step;  /*speed of cursor */
 extern int x,y;   /*co-ordinates of the cursor*/
 extern int trace; /*   trace  on / off*/


 /* L I S T O F  F U N C T I O N S*/

 void left(void);
 void right(void);
 void up(void);
 void down(void);
 void left_up(void);
 void right_up(void);
 void left_down(void);
 void right_down(void);
 void slowmove_left(void);


/*   SLOW   LEFT   MOTION    */
  void slowmove_left(void)
  {
    if(trace)linerel(1,0);
    x++;
    if(x==540) x--;
    moveto(x,y);
  }


    void left(void)
     {
       if(trace) linerel(step,0);
       x += step;      /*   left   */
       if(x>540) x -= step;
       moveto(x,y);
     }


     void right(void)
       {
        if(trace)linerel(-step,0);
        x -= step;    /*   right    */
        if(x<=10) x += step;
        moveto(x,y);
        }
```

```
void up(void)
  æ
    if(trace)linerel(0,-step);
    y -= step;/*     up     */
    if(y<=10) y += step;
    moveto(x,y);
  ä

void down(void)
  æ
    if(trace)linerel(0,step);
    y += step;/*    down    */
    if(y>=VGACHN + 30) y -= step;
    moveto(x,y);
  ä

  /*_____.____________________02_09_93_____________*/

      /*  left - up    */
  void left_up(void)
  æ

    if(trace)linerel(-step,-step); */(Home)*/
     x -= step;/*RIGHT_UP */
    if(x<=10)
  æ
    x += step;
    moveto(x,y);
    return;
  ä

    y -= step;/* up     */
    if(y<=10)
  æ
    y += step;
    x += step;
    moveto(x,y);
    return;
  ä
    moveto(x,y);
  ä

/*    right -  up  */
    void right_up(void)
     æ
      if(trace) linerel(step,-step);*/(PgUp)*/
       x += step; /*     right - UP   */
      if(x>=540)
        æ
         x -= step;
         moveto(x,y);
         return;
         ä
```

```
    y -= step; /*    up    */
    if(y<=10)
    æ
      y += step;
      x -= step;
      moveto(x,y);
      return;
     ä
       moveto(x,y);
     ä

/*  right - down  */
     void right_down(void)
        æ
         if(trace)linerel(step,step);
                        /*(PgDn)  */
         x += step;
                 /*   right    */
         if(x>=540)
                    /* edge */
          æ
            x -= step;
            moveto(x,y);
                  /* restore */
            return;
           ä

            y += step;
                /*   down   */
            if(y>=VGACHN + 30)
           æ
             y -= step;
             x -= step;
             moveto(x,y);
             return;
           ä
             moveto(x,y);
           ä


 /* left - down  */
     void left_down(void)
           æ
           if(trace)linerel(-step,step);
                 /*   (End)  */

            x -= step;
                 /*  left     */
             if(x<=10)
              æ
                x += step;
                moveto(x,y);
                return;
               ä
```

```
                y += step;
        /*    down    */
                if(y>=VGACHN + 30)
                 æ
                  y -= step;
                  x += step;
                  moveto(x,y);
                  return;
                 ä
                  moveto(x,y);
                 ä
```

```
/*&&&&&&&&&&&&&&& END OF CURMOV.H  &&&&&&&&*/
```

REFERENCES:

1) Mc Gregor J. & Watt A., The art of Graphics for the IBM PC, Addison - Wesley Publishers Ltd., - 1986

2) Reddy B.R. and Premkumar R., Visual Programming (or How to use your right brain Right !)

C & C June 1993 PP 97-100

3) Sproull R.F., Sutherland W.R. and Ullner M.K., Device independent Graphics with examples from IBM PC.,
Mcgraw Hill Book Inc.. 1989

****