Listing of Header File <digital.h>

```
#include<stdlib.h>
#include<iostream.h>
#include<graphics.h>

int near5(int x)
{
  int ret=(x/10)*10;
  int n=x%10;
  switch(n)
  { case 0: case 1: case 2: break;
    case 3 : case 4 : case 5 : case 6 : case 7 : ret+=5; break;
    case 8 : case 9 : ret+=10; break;
  }
  return(ret);
}

typedef struct Point
{
  int x,y;
} Point;

//The base class for all digital gates
class Gate
{
  public:
  int type;
  int n;
  int value;
  int flag;
  int count;
  int ox;
  int oy;
  int h;
  int color;
  int startx;
  int endx;
  int starty;
  int endy;
  Gate *input[20];
  Gate(int);
  //To be used by derived classes
  void setSize(int s) {    n=s;  }
  void setInput(Gate *g)  {    input[count++]=g;  }
  void setPosition(int a,int b)  {   ox=a; oy=b;  }
  void reset()  { flag=0; }
  void resetConn() { count=0; }
  //To be implemented by derived classes
  virtual int getOutput(){return(0);};
  virtual void draw(int c=1){};
  virtual Point getO() { Point p;return(p);};
  virtual int isInput(Point p) { return(0);}
  virtual int isInside(int x,int y) { return(0);}
};

Gate::Gate(int s)
```

```
{
  n=s;
  count=0;
  flag=0;
  h=(n-1)*10+10;
  ox=oy=0;
}
//AND gate
class And : public Gate
{
  public:
  And(int s) : Gate(s) {color=BLUE;type=0;}
  int getOutput();
  void draw(int);
  Point getO();
  int isInput(Point p);
  int isInside(int a,int b)
  {
    if((ox-h/2<=a)&&(oy-h/2<=b)&&(a<=ox+5)&&(b<=oy+h/2)) return(1);
    return(0);
  }
};

int And::getOutput()

{
  int ret=1;
  if (flag==1) return(value);
  flag++;
  for(int i=0;i<count;i++)
  {
    ret = ret && input[i]->getOutput();
  }
  value=ret;
  return(value);
}
void And::draw(int c=1)
{
  if (c) setcolor(BLUE); else setcolor(BLACK);
  line(ox,oy-h/2,ox+5,oy-h/2);
  line(ox,oy+h/2,ox+5,oy+h/2);
  line(ox+5,oy-h/2,ox+5,oy+h/2);
  for(int i=0;i<n;i++)
  line(ox+5,oy-h/2+5+(i*10),ox+10,oy-h/2+5+(i*10));//inputs
  line(ox-h/2,oy,ox-h/2-5,oy);//output
  arc(ox,oy,90,270,h/2);
}
Point And::getO()
{
  Point p;
  p.x=ox-h/2-5;
  p.y=oy;
  return(p);
}

int And::isInput(Point p)
{
```

```
  for(int i=0;i<n;i++)
  {
    if ((p.x==(ox+10))&&(p.y==(oy-h/2+5+(i*10))))
    { return(1);}
  }
  return(0);
}
//OR gate
class Or : public Gate

{
  public:
  Or(int s) : Gate(s) {color=RED;type=2;}
  int getOutput();
  void draw(int);
  Point getO();
  int isInput(Point p);
  int isInside(int a,int b)
  {
    if ((ox-h<=a)&&(oy-h/2<=b)&&(a<=ox+5)&&(b<=oy+h/2)) return(1);
    return(0);

  }
};
int Or ::getOutput()
{
  int ret=0;
  if (flag==1) return(value);
  flag++;
  for(int i=0;i<count;i++)
  {
    ret = ret || input[i]->getOutput();
  }
  value=ret;
  return(value);
}
void Or::draw(int c=1)
{
  if (c) setcolor(RED); else setcolor(0);
  line(ox,oy-h/2,ox+5,oy-h/2);
  line(ox,oy+h/2,ox+5,oy+h/2);
  for(int i=0;i<n;i++)
  line(ox+5,oy-h/2+5+(i*10),ox+10,oy-h/2+5+(i*10));//inputs
  line(ox-(int)(.86*h),oy,near5(ox-(int)(.86*h))-5,oy);//output
  arc(ox+h-h/7+5,oy,180-30,180+30,h);
  arc(ox,oy+h/2,90,150,h);
  arc(ox,oy-h/2,210,270,h);
}

Point Or::getO()
{
  Point p;
  p.x=near5(ox-(int)(0.86*h))-5;
  p.y=oy;
  return(p);
}
```

```
int Or::isInput(Point p)
{
  for(int i=0;i<n;i++)
  {
    if ((p.x==(ox+10))&&(p.y==(oy-h/2+5+(i*10))))
    { return(1);}
  }
  return(0);
}

//NOT gate
class Not : public Gate
{
  public:
  Not() : Gate(1) {color=YELLOW;type=4;}
  int getOutput();
  void draw(int);
  Point getO();
  int isInput(Point p);
  int isInside(int a,int b)
  {
    if ((ox-2<=a)&&(oy-4<=b)&&(a<=ox+10)&&(b<=oy+4)) return(1);
    return(0);
  }
};

int Not::getOutput()
{
  int ret=1;
  if (flag==1) return(value);
  flag++;
  for(int i=0;i<count;i++)
  {
    ret = ret && input[i]->getOutput();
  }
  value=!ret;
  return(value);
}

void Not ::draw(int c=1)
{
  if (c) setcolor(YELLOW); else setcolor(0);
  circle(ox,oy,2);
  line(ox+2,oy,ox+10,oy-4);
  line(ox+2,oy,ox+10,oy+4);
  line(ox+10,oy-4,ox+10,oy+4);
  line(ox+10,oy,ox+15,oy);//input
  line(ox-2,oy,ox-10,oy);//output
}
Point Not::getO()
{
  Point p;
  p.x=ox-10;
  p.y=oy;
  return(p);
}
int Not::isInput(Point p)
```

```
{
  if ((p.x==(ox+15))&&(p.y==oy))  return(1);
  return(0);
}

// LED
class Led : public Gate
{
  public:
  Led() : Gate(1) {color=BROWN;type=5;}
  int getOutput();
  void draw(int);
  Point getO();
  int isInput(Point p);
  int isInside(int a,int b)
  {
    if ((ox-15<=a)&&(oy-5<=b)&&(a<=ox-5)&&(b<=oy+5)) return(1);
    return(0);
  }
};

int Led::getOutput()
{
  int ret=0;
  if (flag==1) return(value);
  flag++;
  for(int i=0;i<count;i++)
  {
    ret = ret || input[i]->getOutput();
  }
  value=ret;
  draw(2);
  return(value);
}

void Led::draw(int c=1)
{
  if (c) setcolor(GREEN); else setcolor(0);
  line(ox,oy,ox-5,oy);//input (ox,oy)
  circle(ox-10,oy,5);
  if (c==2)
  {
    if (value==1) { setcolor(YELLOW); setfillstyle(1,YELLOW);}
    else { setcolor(BLACK); setfillstyle(1,BLACK);}
    pieslice(ox-10,oy,0,360,4);
  }
}

Point Led::getO()
{
  Point p;
  p.x=-1;
  p.y=-1;
  return(p);
}

int Led::isInput(Point p)
```

```
{
  if ((p.x==ox)&&(p.y==oy)) return(1);
  return(0);
}

// SWITCH
class Switch : public Gate
{
  public:

  Switch() : Gate(0) {color=BROWN;type=6;value=0;}
  int getOutput();
  void draw(int);
  Point getO();
  int isInput(Point p);
  int isInside(int a,int b)
  {
    if ((ox-7<=a)&&(oy-7<=b)&&(a<=ox+7)&&(b<=oy+7)) return(1);
    return(0);
  }
  void toggle() { value=1-value;}
};

int Switch::getOutput()
{
  return(value);
}

void Switch::draw(int c=1)
{
  if (c) setcolor(BROWN); else setcolor(0);
  circle(ox,oy,5);
  rectangle(ox-7,oy-7,ox+7,oy+7);
  line(ox-7,oy,ox-15,oy);
  if (c==2)
  {
    if (value==1) { setcolor(YELLOW); setfillstyle(1,YELLOW);}
    else { setcolor(BLACK); setfillstyle(1,BLACK);}
    pieslice(ox,oy,0,360,4);
  }
}

Point Switch::getO()
{
  Point p;
  p.x=ox-15;
  p.y=oy;
  return(p);
}
int Switch::isInput(Point p)
{ p.x=p.y; return(0); }

// NAND gate
class Nand : public Gate
{
  public:
  Nand(int s) : Gate(s) {color=MAGENTA;type=1;}
```

```
  int getOutput();
  void draw(int);
  Point getO();
  int isInput(Point p);
  int isInside(int a,int b)
  {
    if((ox-h/2<=a)&&(oy-h/2<=b)&&(a<=ox+5)&&(b<=oy+h/2)) return(1);
    return(0);
  }
};

int Nand::getOutput()
{
  int ret=1;
  if (flag==1) return(value);
  flag++;
  for(int i=0;i<count;i++)
  {
    ret = ret && input[i]->getOutput();
  }
  value=!ret;
  return(value);
}

void Nand::draw(int c=1)
{
  if (c) setcolor(MAGENTA); else setcolor(0);
  line(ox,oy-h/2,ox+5,oy-h/2);
  line(ox,oy+h/2,ox+5,oy+h/2);
  line(ox+5,oy-h/2,ox+5,oy+h/2);
  for(int i=0;i<n;i++)
  line(ox+5,oy-h/2+5+(i*10),ox+10,oy-h/2+5+(i*10));//inputs
  circle(ox-h/2-2,oy,2);
  line(ox-h/2-5,oy,ox-h/2-10,oy);//output
  arc(ox,oy,90,270,h/2);
}

Point Nand::getO()
{
  Point p;
  p.x=ox-h/2-10;
  p.y=oy;
  return(p);
}

int Nand::isInput(Point p)
{
  for(int i=0;i<n;i++)
  {
    if ((p.x==(ox+10))&&(p.y==(oy-h/2+5+(i*10))))
    { return(1);}
  }
  return(0);
}

// NOR gate
class Nor : public Gate
```

```
{
  public:
  Nor(int s) : Gate(s) {color=LIGHTBLUE;type=3;}
  int getOutput();
  void draw(int);

  Point getO();
  int isInput(Point p);
  int isInside(int a,int b)
  {
    if ((ox-h<=a)&&(oy-h/2<=b)&&(a<=ox+5)&&(b<=oy+h/2)) return(1);
    return(0);
  }
};

int Nor::getOutput()
{
  int ret=0;
  if (flag==1) return(value);
  flag++;
  for(int i=0;i<count;i++)
  {
    ret = ret || input[i]->getOutput();
  }
  value=!ret;
  return(value);
}

void Nor::draw(int c=1)
{
  if (c) setcolor(LIGHTBLUE); else setcolor(0);
  line(ox,oy-h/2,ox+5,oy-h/2);
  line(ox,oy+h/2,ox+5,oy+h/2);
  for(int i=0;i<n;i++)
  line(ox+5,oy-h/2+5+(i*10),ox+10,oy-h/2+5+(i*10));//inputs
  circle(ox-(int)(.86*h)-2,oy,2);
  line(ox-(int)(.86*h)-5,oy,ox-near5((int)(.86*h))-10,oy);//output
  arc(ox+h-h/7+5,oy,180-30,180+30,h);
  arc(ox,oy+h/2,90,150,h);
  arc(ox,oy-h/2,210,270,h);

}

Point Nor::getO()
{
  Point p;
  p.x=near5(ox-(int)(.86*h))-10;
  p.y=oy;
  return(p);
}

int Nor::isInput(Point p)
{
  for(int i=0;i<n;i++)
  {
    if ((p.x==(ox+10))&&(p.y==(oy-h/2+5+(i*10))))
    { return(1);}
```

```
  }
  return(0);
}

// CONNECTOR
class Connector : public Gate
{
  public:
  Connector() : Gate(1)  { startx=starty=endx=endy=0;color=LIGHTRED;type=7; }
  int getOutput();
  void draw(int);
  Point getO();
  int isInput(Point p);
  int isInside(int x,int y);
  void setIn(int x,int y) { startx=x; starty=y; }
  void setOut(int x,int y) {  endx=x;   endy=y; }
};

int Connector::getOutput()
{
  int ret=1;
  if (flag==1) return(value);
  flag++;
  for(int i=0;i<count;i++)
  {
    ret = ret && input[i]->getOutput();
  }
  value=ret;
  return(value);
}

void Connector::draw(int c)
{
  if (c) setcolor(LIGHTRED); else setcolor(0);
  line(startx,starty,endx,endy);
}

Point Connector::getO()
{
  Point p;
  p.x=endx;
  p.y=endy;
  return(p);
}
int Connector::isInput(Point p)
{
  if ((p.x==startx)&&(p.y==starty)) return(1);
  return(0);
}
int Connector::isInside(int x,int y)
{
  int left_x=startx;
  int top_y=starty;
  int right_x=endx;
  int down_y=endy;

  if (left_x>right_x)
```

```
  {
    int temp=left_x;
    left_x=right_x;
    right_x=temp;
  }//swap
  if (top_y>down_y)
  {
    int temp=top_y;
    top_y=down_y;
    down_y=temp;
  }//swap

  if (startx==endx) { left_x-=5; right_x+=5; }
  if (starty==endy) { top_y-=5; down_y+=5; }

  if ( (x>=left_x) && (x<=right_x) && (y>=top_y) && (y<=down_y)) return(1);
  return(0);
}
```

Listing of <mouse.h>

```
#include<dos.h>
union REGS i,o;
void showmouse(void)
{   i.x.ax=1;
    int86(0x33,&i,&o);
}

void hidemouse(void)
{   i.x.ax=2;
    int86(0x33,&i,&o);
}

int mousex(void)
{   i.x.ax=3;
    int86(0x33,&i,&o);
    return(o.x.cx);
}

int mousey(void)
{   i.x.ax=3;
    int86(0x33,&i,&o);
    return(o.x.dx);
}

int mousebtn(void)
{  i.x.ax=3;
   int86(0x33,&i,&o);
   return( o.x.bx );
}

int m_status(int *x,int *y)
{   i.x.ax=3;
    int86(0x33,&i,&o);
    *x=o.x.cx;
    *y=o.x.dx;
```

```
    return(o.x.bx);
}
void swap(int *a,int *b)
{  int temp;
   temp=*a;
   *a=*b;
   *b=temp;
}

int inside(int x,int y,int left_x,int top_y,int right_x,int down_y)
{
    if (left_x>right_x)  swap(&left_x,&right_x);
    if (top_y>down_y)    swap(&top_y,&down_y);

    if ( (x>=left_x) && (x<=right_x) && (y>=top_y) && (y<=down_y)) return(1);
    return(0);
}
```

Listing of <Dia.h>

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
#include<graphics.h>
#include<conio.h>
#include<dos.h>

typedef struct file
{
  char name[10];
} file;

void *screen_ptr;
file getfile();
int getInt();

void getScreen(int x,int y,int w,int h)
{
   int size = imagesize(x,y,x+w,y+h);
   screen_ptr = malloc(size);
   getimage(x,y,x+w,y+h,screen_ptr);
}

void restoreScreen(int x,int y)
{
   putimage(x,y,screen_ptr,0);
   free(screen_ptr);
}

file getfile()
{
  char array[10];
  int count=0;
  int h=30,w=225;
  int x=(getmaxx()-w)/2;
  int y=(getmaxy()-h)/2;
```

```
  file f;
  getScreen(x,y,w,h);
  setcolor(BLUE);
  rectangle(x,y,x+w,y+h);
  setfillstyle(1,0);
  bar(x+1,y+1,x+w-1,y+h-1);
  setcolor(RED);
  outtextxy(x+10,y+10,"Enter File Name:-");
  setcolor(BROWN);
  int curx=x+15+textwidth("Enter File Name:-");
  int cury=y+10;
  char cstr[2];
  cstr[1]='\0';
  line(curx+1,cury-1,curx+1,cury+7);
  while(1)
  {  char c=getch();
     if (c==13) break;
     if (c==27) { array[0]='\0';break;}
     if (c==32) continue;
     if (c==8)
     { if (count>0)
       {
       setcolor(0);
       line(curx+1,cury-1,curx+1,cury+7);
       cstr[0]=array[count-1];
       curx-=textwidth(cstr);
       outtextxy(curx,cury,cstr);
       count--;
       setcolor(BROWN);
       line(curx+1,cury-1,curx+1,cury+7);
       }
     }
     else if (count<8)
     {
       setcolor(0);
       line(curx+1,cury-1,curx+1,cury+7);
       setcolor(BROWN);
       cstr[0]=c;
       outtextxy(curx,cury,cstr);
       curx+=textwidth(cstr);
       line(curx+1,cury-1,curx+1,cury+7);
       array[count]=c;
       count++;
     }
  }
  array[count]='\0';
  restoreScreen(x,y);
  strcpy(f.name,array);
  return(f);
}


int getInt()
{
  char array[10];
  int count=0;
  int h=30,w=250;
```

```
  int x=(getmaxx()-w)/2;
  int y=(getmaxy()-h)/2;
  getScreen(x,y,w,h);
  setcolor(BLUE);
  rectangle(x,y,x+w,y+h);
  setfillstyle(1,0);
  bar(x+1,y+1,x+w-1,y+h-1);
  setcolor(RED);
  outtextxy(x+10,y+10,"Enter Number of Inputs:-");
  setcolor(BROWN);
  int curx=x+15+textwidth("Enter Number of Inputs:-");
  int cury=y+10;
  char cstr[2];
  cstr[1]='\0';
  line(curx+1,cury-1,curx+1,cury+7);
  while(1)
  {  char c=getch();
     if (c==13) break;
     if (c==27) { array[0]='\0';break;}
     if (c==32) continue;
     if ((c<'0')||(c>'9')) continue;
     if (c==8)
     { if (count>0)
       {
       setcolor(0);
       line(curx+1,cury-1,curx+1,cury+7);
       cstr[0]=array[count-1];
       curx-=textwidth(cstr);
       outtextxy(curx,cury,cstr);
       count--;
       setcolor(BROWN);
       line(curx+1,cury-1,curx+1,cury+7);
       }
     }
     else if (count<4)
     {
       setcolor(0);
       line(curx+1,cury-1,curx+1,cury+7);
       setcolor(BROWN);
       cstr[0]=c;
       outtextxy(curx,cury,cstr);
       curx+=textwidth(cstr);
       line(curx+1,cury-1,curx+1,cury+7);
       array[count]=c;
       count++;
     }
  }
  array[count]='\0';
  restoreScreen(x,y);
  return(atoi(array));
}

void dia(char str[])
{
  int h=textheight(str)+20;
  int w=textwidth(str)+20;
  int x=(getmaxx()-w)/2;
```

```
  int y=(getmaxy()-h)/2;
  getScreen(x,y,w,h);
  setcolor(BLUE);
  rectangle(x,y,x+w,y+h);
  setfillstyle(1,0);
  bar(x+1,y+1,x+w-1,y+h-1);
  setcolor(RED);
  outtextxy(x+10,y+10,str);
  getch();
  restoreScreen(x,y);
}
```

Listing of "main.cpp"

```
#include<stdio.h>
#include<graphics.h>
#include<conio.h>
#include<math.h>
#include<string.h>
#include"mouse.h"
#include"digital.h"
#include"dia.h"

#define CKT_X (30)
#define CKT_Y (30)

int run_flag=0;
int open_flag=0;

class Button
{
  int x;
  int y;
  int h;
  int w;
  int color;
  char *text;
  int (*f)();
  public:
  Button(int a,int b,int c,int d,int col)
  { x=a; y=b; w=c; h=d; color=col;}
  void setText(char str[]) { text=str; }
  void setFunction(int (*func)()) { f=func; }
  void show(int val)
  { hidemouse();
    setcolor(color);
    rectangle(x,y,x+w,y+h);
    setfillstyle(1,0);
    setcolor(0);
    bar(x+1,y+1,x+w-1,y+h-1);
    if (val==1) setcolor(RED); else setcolor(color);
    int px=x+(w-textwidth(text))/2;
    int py=y+(h-textheight(text))/2;
    outtextxy(px,py,text);
    showmouse();
  }
```

```
  int isClicked(int a,int b)
  {
    int btn=mousebtn();
    if ((x<=a)&&(y<=b)&&(x+w>=a)&&(y+h>=b))
    {  show(btn);
       if (btn==1) { if ((*f)()) show(0);}
       return(1);
    }
    return(0);
  }
};

file filename;

void connector();
void rewire();
void cktdelete(int);

int fOpen();
int fSave();
int fRun()
{ run_flag=1;
  setcolor(GREEN);
  setfillstyle(1,GREEN);
  circle(60,getmaxy()-10,5);
  floodfill(60,getmaxy()-10,GREEN);
  return(0);
}
int fStop()
{ run_flag=0;
  setcolor(RED);
  setfillstyle(1,RED);
  circle(60,getmaxy()-10,5);
  floodfill(60,getmaxy()-10,RED);
  rewire();
  return(0);
}
int fClear();
void fMove(int);

Gate *cktStore[200];
int cktcount=0;
int currGate=-1;

Led *ledStore[100];
int ledcount=0,ledindex[200];

Switch *switchStore[100];
int switchcount=0,switchindex[200];


void main(int argc,char *argv[])
{
  int gd=DETECT,gm,ox=120,oy,i,j;
  Gate *m_gate[8];
  initgraph(&gd,&gm,"C:\\DIGISIM");
```

```
Button clear(ox,getmaxy()-20,60,20,BLUE);
clear.setText("Clear");
clear.setFunction(fClear);
clear.show(0);
ox+=60;

Button open(ox,getmaxy()-20,60,20,BLUE);
open.setText("Open");
open.setFunction(fOpen);
open.show(0);
ox+=60;

Button save(ox,getmaxy()-20,60,20,BLUE);
save.setText("Save");
save.setFunction(fSave);
save.show(0);
ox+=60;

Button run(ox,getmaxy()-20,60,20,BLUE);
run.setText("Run");
run.setFunction(fRun);
run.show(0);
ox+=60;

Button stop(ox,getmaxy()-20,60,20,BLUE);
stop.setText("Stop");
stop.setFunction(fStop);
stop.show(0);

ox=getmaxx()-25;
oy=25;

setcolor(DARKGRAY);
rectangle(getmaxx()-60,0,getmaxx(),getmaxy());
setcolor(YELLOW);
outtextxy(getmaxx()-50,20,"GATES");
setcolor(RED);
rectangle(0,0,getmaxx()-70,getmaxy()-30);
for(i=0;i<getmaxx()-70;i+=10)
  for(j=0;j<getmaxy()-30;j+=10) putpixel(i,j,RED);
setcolor(DARKGRAY);

//And Gate
oy+=50;
m_gate[0]=new And(2);
m_gate[0]->setPosition(ox,oy);
m_gate[0]->draw();
//Nand
oy+=50;
m_gate[1]=new Nand(2);
m_gate[1]->setPosition(ox,oy);
m_gate[1]->draw();
//Or Gate
oy+=50;
m_gate[2]=new Or(2);
m_gate[2]->setPosition(ox,oy);
m_gate[2]->draw();
```

```
//Nor
oy+=50;
m_gate[3]=new Nor(2);
m_gate[3]->setPosition(ox,oy);
m_gate[3]->draw();
//Not Gate
oy+=50; ox-=5;
m_gate[4]=new Not();
m_gate[4]->setPosition(ox,oy);
m_gate[4]->draw();
//Led
oy+=50; ox+=15;
m_gate[5]=new Led();
m_gate[5]->setPosition(ox,oy);
m_gate[5]->draw();
//Switch
oy+=50; ox-=10;
m_gate[6]=new Switch();
m_gate[6]->setPosition(ox,oy);
m_gate[6]->draw();
//Connector
oy+=50; ox-=5;
Connector *c=new Connector;
m_gate[7]=c;
c->setIn(ox-15,oy-15);
c->setOut(ox+15,oy+15);
m_gate[7]->draw();
setcolor(BROWN);
outtextxy(0,getmaxy()-14,"Status");
fStop();
open_flag=1;
for(i=1;i<argc;i++)
{
  strcpy(filename.name,argv[i]);
  fOpen();
}
filename.name[0]='\0';
open_flag=0;
setviewport(0,0,getmaxx()-70,getmaxy()-30,0);

int prev=1,cur=0;
showmouse();
while(1)
{
  char s[10];
  int in=0;
  cur=mousebtn();
  if (cur!=prev)
  {
    int mx=mousex();
    int my=mousey();
    clear.isClicked(mx,my);
    open.isClicked(mx,my);
    save.isClicked(mx,my);
    run.isClicked(mx,my);
    stop.isClicked(mx,my);
    for(i=0;i<=7;i++)
```

```
{ setcolor(0);
rectangle(getmaxx()-59,i*50+51,getmaxx()-1,i*50+99);
if ((m_gate[i]->isInside(mx,my))&&(run_flag==0))
{ //gate creator routine
  if (cur)
  {int num;
   hidemouse();   setcolor(YELLOW);
   rectangle(getmaxx()-59,i*50+51,getmaxx()-1,i*50+99);
   showmouse();
   if ((i>=0)&&(i<=3))
   {   num=getInt(); if (num==0) continue;   }
   switch(i)
   {
     case 0: cktStore[cktcount++]=new And(num); break;
     case 1: cktStore[cktcount++]=new Nand(num); break;
     case 2: cktStore[cktcount++]=new Or(num); break;
     case 3: cktStore[cktcount++]=new Nor(num); break;
     case 4: cktStore[cktcount++]=new Not; break;
     case 5:
            Led *l=new Led;
            ledindex[cktcount]=ledcount;
            cktStore[cktcount++]=l;
            ledStore[ledcount++]=l;
            break;
     case 6:
            Switch *s=new Switch;
            switchindex[cktcount]=switchcount;
            cktStore[cktcount++]=s;
            switchStore[switchcount++]=s;
            break;
     default: break;
   }
   if (i==7) { connector(); rewire(); }
   else
   { cktStore[cktcount-1]->setPosition(CKT_X,CKT_Y);
     cktStore[cktcount-1]->draw();
     break;
   }
  }//if mouse == 1
}//if any gate is clicked
}//for all gates in the menu
if (cur)
{ currGate=-1;
if (run_flag==0)
{ for(int i=cktcount-1;i>=0;i--)
  {
    if (cktStore[i]->isInside(mx,my))
    { currGate=i;
      fMove(currGate);
      rewire();
      break;
    }
  }//for
}
if (run_flag==1)
{ hidemouse();
```

```
        for(i=0;i<switchcount;i++)

        {
          if (switchStore[i]->isInside(mx,my))  switchStore[i]->toggle();
          switchStore[i]->draw(2);
        }
        showmouse();
      }
      }//if (cur)
    }//if clicked
    prev=cur;
    if (kbhit()) in=getch();
    if (in==27) break;
    if ((in==83)&&(!run_flag)) { cktdelete(currGate); rewire(); }
    if (run_flag==1)
    { //hidemouse();
      for(int i=0;i<cktcount;i++)
         { cktStore[i]->reset();}
      for(i=0;i<ledcount;i++)
         { ledStore[i]->getOutput();}
      //showmouse();
    }

  }//while true

  for(i=0;i<7;i++) delete(m_gate[i]);
  for(i=0;i<cktcount;i++) delete(cktStore[i]);
  closegraph();
}


void fMove(int i)
{
  int x,prevx,y,prevy,dsx,dsy,dex,dey;
  x=prevx=near5(mousex());
  y=prevy=near5(mousey());
  int dx=cktStore[i]->ox-x;
  int dy=cktStore[i]->oy-y;
  if (cktStore[i]->type==7)
  {
    dsx=cktStore[i]->startx-x;
    dsy=cktStore[i]->starty-y;
    dex=cktStore[i]->endx-x;
    dey=cktStore[i]->endy-y;
  }
  setviewport(0,0,getmaxx()-70,getmaxy()-30,1);
  setfillstyle(1,0);
  hidemouse();
  while(mousebtn())
  {
    x=near5(mousex());
    y=near5(mousey());
    if ((x!=prevx)||(y!=prevy))
    {
      cktStore[i]->draw(0);
      cktStore[i]->setPosition(x+dx,y+dy);
      if (cktStore[i]->type==7)
```

```
      {
      cktStore[i]->startx=x+dsx;
      cktStore[i]->starty=y+dsy;
      cktStore[i]->endx=x+dex;
      cktStore[i]->endy=y+dey;
      }
      setcolor(RED);
      rectangle(0,0,getmaxx()-70,getmaxy()-30);
      for(int k=0;k<getmaxx()-70;k+=10)
      for(int j=0;j<getmaxy()-30;j+=10) putpixel(k,j,RED);
      for(j=0;j<cktcount;j++) cktStore[j]->draw();
    }
    prevx=x; prevy=y;
  }

  setviewport(0,0,getmaxx()-70,getmaxy()-30,0);
  showmouse();
}

void rewire()
{
  Gate *in, *out;
  hidemouse();
  setcolor(RED);
  setfillstyle(1,0);
  bar(1,1,getmaxx()-71,getmaxy()-31);
  rectangle(0,0,getmaxx()-70,getmaxy()-30);
  for(int k=0;k<getmaxx()-70;k+=10)
  for(int j=0;j<getmaxy()-30;j+=10) putpixel(k,j,RED);
  for(j=0;j<cktcount;j++) cktStore[j]->draw();
  setcolor(RED);
  setfillstyle(1,RED);
  for(int i=0;i<cktcount;i++)
  {
    in=cktStore[i];
    in->resetConn();
    for(int j=0;j<cktcount;j++)
    {
      if (i==j) continue;
      out=cktStore[j];
      Point p=out->getO();
      if (in->isInput(out->getO()))
      {
      in->setInput(out);
      bar(p.x-1,p.y-1,p.x+1,p.y+1);
      }
    }// for j
  }// for i
  showmouse();
}//rewire

void connector()
{
  int btn,prev;
  int count=0;
  int x[2],y[2];
  btn=prev=mousebtn();
```

```
  while(1)
  {
    btn=mousebtn();
    if (btn!=prev)
    {
      if (count==0)
      { hidemouse();
      setcolor(0);
      rectangle(getmaxx()-59,7*50+51,getmaxx()-1,7*50+99);
      showmouse();
      }
      else
      { if ((mousex()>getmaxx()-70)||(mousey()>getmaxy()-30)) return;
      x[count-1]=near5(mousex());
      y[count-1]=near5(mousey());
      }
      count++;
    }
    if (count==3) break;
    prev=btn;
  }
  for(int i=0;i<2;i++)
  { Gate *g;
    Point p,q;
    int j=1-i;
    for(int k=0;k<cktcount;k++)
    {
      g=cktStore[k];
      p.x=x[i];
      p.y=y[i];
      if (g->isInput(p))
      {
       Connector *c=new Connector();
       c->setIn(x[j],y[j]);
       c->setOut(x[i],y[i]);
       cktStore[cktcount++]=c;
       hidemouse(); c->draw(1); showmouse();
       return;
      }
      q=g->getO();
      if ((p.x==q.x)&&(p.y==q.y))
      {
       Connector *c=new Connector();
       c->setIn(x[i],y[i]);
       c->setOut(x[j],y[j]);
       cktStore[cktcount++]=c;
       hidemouse(); c->draw(1); showmouse();
       return;
      }
    }
  }
}

void cktdelete(int n)
{
  if (n==-1) return;
  if (cktStore[n]->type==5)
```

```
  {
    int nindex=ledindex[n];
    for(int i=nindex;i<ledcount-1;i++)
    ledStore[i]=ledStore[i+1];
    ledcount--;
  }
  if (cktStore[n]->type==6)
  {
    int nindex=switchindex[n];
    for(int i=nindex;i<switchcount-1;i++)
    switchStore[i]=switchStore[i+1];
    switchcount--;
  }

  delete(cktStore[n]);
  for(int i=n;i<cktcount-1;i++)
  cktStore[i]=cktStore[i+1];
  cktcount--;
  currGate=-1;
}

int fClear()
{
  for(int i=0;i<cktcount;i++)
  delete(cktStore[i]);
  cktcount=0;
  ledcount=0;
  switchcount=0;
  filename.name[0]='\0';
  rewire();
  return(0);
}

int fOpen()
{ int type,ox,oy,num,n;
  if (!open_flag) filename=getfile();
  if (strlen(filename.name)==0) { return(1); }

  FILE *fp=fopen(filename.name,"r");
  if (!fp)  { dia("File could not be Opened"); return(1); }
  hidemouse();
  while(!feof(fp))
  {
    fscanf(fp,"%d %d %d %d\n",&type,&ox,&oy,&num);
    switch(type)
     {
       case 0: cktStore[cktcount++]=new And(num); break;
       case 1: cktStore[cktcount++]=new Nand(num); break;
       case 2: cktStore[cktcount++]=new Or(num); break;
       case 3: cktStore[cktcount++]=new Nor(num); break;
       case 4: cktStore[cktcount++]=new Not; break;
       case 5:
            Led *l=new Led;
            ledindex[cktcount]=ledcount;
            cktStore[cktcount++]=l;
            ledStore[ledcount++]=l;
            break;
```

```
        case 6:
             Switch *s=new Switch;
             switchindex[cktcount]=switchcount;
             cktStore[cktcount++]=s;
             switchStore[switchcount++]=s;
             break;
        case 7:
             int sx,sy,ex,ey;
             fscanf(fp,"%d %d %d %d\n",&sx,&sy,&ex,&ey);
             Connector *c=new Connector();
             c->setIn(sx,sy);
             c->setOut(ex,ey);
             cktStore[cktcount++]=c;
             break;
        default: break;
       }
       cktStore[cktcount-1]->setPosition(ox,oy);
       cktStore[cktcount-1]->draw();
  }//while
  rewire();
  showmouse();
  fclose(fp);
  filename.name[0]='\0';
  return(1);
}

int fSave()
{ int flag=0;
  if (cktcount==0) { dia("No circuit to Save"); return(1);}
  if (strlen(filename.name)==0) { filename=getfile(); flag=1; }
  if (strlen(filename.name)==0) { return(flag); }

  FILE *fp=fopen(filename.name,"w");
  if (!fp)  { dia("File could not be Saved"); return(flag); }
  for(int i=0;i<cktcount;i++)
  {
    Gate *g;
    g=cktStore[i];
    fprintf(fp,"%d %d %d %d\n",g->type,g->ox,g->oy,g->n);
    if (g->type==7)
    fprintf(fp,"%d %d %d %d\n",g->startx,g->starty,g->endx,g->endy);
  }
  fclose(fp);
  char msg[100];
  sprintf(msg,"Circuit saved in File \"%s\".",filename.name);
  dia(msg);
  return(1);
}
```