SALIENT FEATURES OF PERSONAL COMPUTER (PC-AT)

The Personal Computer (HCL's BUSYBEE AT PLUS) used in this study is a compact and versatile system supporting both single user and multi-user environments. It is ideally suited for higher-end personal computing and lower-end multi-user data processing.

It is based on the powerful 16-bit INTEL 80286 processor. With a CPU clock speed of 10MHz, it is 77% faster than the industry standard IBM PC/AT. The system supports the INTEL 80287 coprocessor for number crunching applications. It is capable of addressing up to 8 M bytes of main memory starting with a minimum of 512 K bytes.

The system supports mini floppy drives and disk drives for auxiliary storage. Mini floppy drives of 360Kb/1.2Mb and disk drives of 20, 40 or 80M Bytes capacity can be attached for large volumes of data storage. Indeed, for such data processing environments, it is capable of supporting up to 11 terminals including the console. The system can support Dot Matrix Printers as well as Letter Quality Printers.

The system offers a wide range of add-on options - a Mouse as a fast pointing device, a Digitizer for capturing graphics data, an X-Y Plotter for printing colour graphics, and a Cartridge tape drive for back-ups.

The system supports two operating systems: (i) The MSDOS 3.0/3.1/3.2/4.01 etc for single user
environment making it IBM PC compatible. Hence, all PC software - the vast variety of popular application software packages and all language compilers - can be run on this machine. (ii) The Xenix V/286 for multi-user environment, derived from unix system, it offers the total power of Unix, supporting all the software that comes with it. (HCL Busy Bee PC-AT users Manual).
I/O CARD DETAILS

The I/O card developed by Electrosystems Associates Pvt. Ltd. which is compatible for PC/XT/AT is used in the present study. It has two programmable peripheral Interface devices (INTEL 8255A) and one Programmable Interval Timer (INTEL 8253). The card can be plugged into any one of the free extension slots of the system. The details of circuit diagram and pin diagram are presented in Annexure-II.

The PPI used with I/O card is an INTEL 8255A. It is a general purpose Programmable Peripheral Interface device. It is compatible with any microprocessor. It can be programmed to transfer data under various conditions, from simple I/O to interrupt I/O. It is a silicon gate MOS chip available in a 40 pin, dual in-line package. The block diagram of the 8255A is shown in Fig.2.2. The 8255 has 24 I/O lines which can be grouped primarily into two 8-bit parallel ports- A&B, with remaining 8-bits as port-C. The eight bits of Port-C can be used as individual bits or can be grouped into two 4-bit ports. The functions of these ports are defined by writing a control word in the control register as mentioned in Fig.2.3.

The 8255A has three major modes of operation. In mode 0, each group is programmable as either input or output. Basically, mode 0 gives three 8-bit I/O ports A,B and C. Mode 1 has two 8-bit ports that can be defined as either a handshake input or output port. Each
of the two ports has associated with a 4 bit control port. In mode 2, an 8-bit bi-directional data bus is provided with an associated 5-bit control port. The additional control line is borrowed from one of the other groups. It has bit set/reset capability that gives the system a high degree of flexibility. The 8255A can be interfaced with any microprocessor for parallel transfer of data under program control.

In the present study, the device is used in simple I/O mode to interface the A/D converter (Data Acquisition System) and the stepper motor (Micrometer Drive) with an appropriate software.

The card is mapped to different addresses by setting the DIP switches SW1 - SW4 suitably. Table 2.1 gives the I/O address mapping for different positions. (Eletrosystems Associates Manual). A In the present study, the following ports of 8255A of I/O card are selected by means of setting switch positions as mentioned in Table 2.1.
#include "stdio.h"
#include "bwindow.h"
#include "boptions.h"

void tnpn(void);
void tnpn(void);
void tujt(void);
void tibf(void);
void tib2(void);
void tib3(void);
void tib4(void);
void tib5(void);
void tdraw(void);
void fn_fet(void);
void fp_fet(void);
void fvg0(void);
void fvg1(void);
void fvg2(void);
void fvg3(void);
void fvg4(void);
void fvg5(void);
void fdraw(void);
void sscr(void);
void striac(void);
void sfor(void);
void srev(void);
void svk(void);
void svec(void);
void sdraw(void);
void dfor(void);
void drev(void);
void dvk(void);
void dvk(void);
void ddraw(void);
void idraw(void);
void quit(void);
void setdefault();

int tibmax, ttype, ftype, fvgmax,
   stype, sbias, svmax, dbias, dvmax,
    ibias, svmax;
```c
/* menu tables */
char *selcsO = {
"NPN",
"PNP",
"UJT",
"IBax. 10uA",
"IBax. 100mA",
"IBax. 1mA",
"IBax. 10mA",
"IBax. 100mA",
"Draw char.",
NULL
};
char *selcsP = {
"M-FET",
"P-FET",
"VGax. 0V",
"VGax. 1V",
"VGax. 2V",
"VGax. 3V",
"VGax. 4V",
"VGax. 5V",
"Draw char.",
NULL
};
char **selcsC = {
"SCR",
"Triac",
"Forward",
"Reverse",
"Vmax. 10V",
"Vmax. 20V",
"Draw char",
NULL
};
char *selcsF = {
"Forward",
"Reverse",
"Vmax. 10V",
"Vmax. 20V",
"Draw char",
NULL
};
char *selcsR = {
"Forward",
"Reverse",
"Vmax. 10V",
"Vmax. 20V",
"Draw char",
NULL
};
char *selcsV = {
"Quit",
NULL
};
```
static void (*tfuncsG)() = tnpn, tpnp, tujt, tib1, tib2, tib3, tib4, tib5, tdraw;
static void (*ffuncsG)() = fn_fet, fp_fet, fvg0, fvg1, fvg2, fvg3, fvg4, fvg5, fdraw;
static void (*sfuncsG)() = sscr, striac, sfor, srev, svf1, svf2, sdraw;
static void (*ifuncsG)() = ifor, irev, ivf1, ivf2, idraw;
static void (*qfuncsG)() = quit;
static void (*dfuncsG)() = tdfor, drev, dvf1, dvf2, ddraw;
static void (*qfuncG)() = (quit);
static MENU tnmG = {
  " Trans/UJT ", tselcs, tfuncs),
  " FET ", fselcs, ffuncbs),
  " SCR/Triac ", sselcs, sfuncbs),
  " Diac " , dselcs, dfuncbs),
  " Diodes ", iselcs, ifuncbs),
  " Quit ", qselcs, qfuncbs),
  (NULL, NULL, NULL)
};
void execO
{
  setdefaultO;
  menu_select("DEVICE CHARACTERISTICS EXECUTIVE", tnm);
}
void setdefaultO
{
  tibmax = MICROD10;
  ttype = PNP ;
  ftype = N_FET ;
  fvgmax = 0 ;
 stype = SCR ;
  sbias = FORWARD ;
  svmax = 10 ;
  dbias = FORWARD ;
  dvmax = 10 ;
  ibias = FORWARD ;
  ivmax = 10 ;
}
void tnpn(void)
{
  ttype = NPN ;
}
void tpnp(void)
{
  ttype = PNP ;
}
void tujt(void)
{
  ttype = UJT ;
}

void tib1(void)
{
  tibmax = MICROD10;
}
void tib2(void)
{
  tibmax = MICROD100;
}
void tib3(void)
{
    tibmax = MILLI11;
}
void tib4(void)
{
    tibmax = MILLI10;
}
void tib5(void)
{
    tibmax = MILLI100;
}
void tdraw(void)
{
    if(ttype == NPN)
        npndraw();
    else
        if(ttype == PNP)
            pnpdraw();
        else
            ujtdraw();
}
void fn_fet(void)
{
    ftype = N_FET;
}
void fp_fet(void)
{
    ftype = P_FET;
}
void fvg0(void)
{
    fvgmax = 0;
}
void fvg1(void)
{
    fvgmax = 1;
}
void fvg2(void)
{
    fvgmax = 2;
}
void fvg3(void)
{
    fvgmax = 3;
}
void fvg4(void)
{
    fvgmax = 4;
}
void fvg5(void)
{
    fvgmax = 5;
}
void fdraw(void)
{
    if(ftype == N_FET)
        nfeidraw();
    else
        if(ftype == P_FET)
            pfetdraw();
}
void sscr(void)
{
    slype = SCR;
}
void striac(void)
{
    slype = TRIAC;
}
void sfor(void)
{
    sbias = FORWARD;
}
void srev(void)
{
    sbias = REVERSE;
}
void sv1(void)
{
    svmax = 10;
}
void sv2(void)
{
    svmax = 20;
}
void sdraw(void)
{
    if(sbias == FORWARD)
        srcfoward();
    else
        if(sbias == REVERSE)
            srcreverse();
}
void dfor(void)
{
    dbias = FORWARD;
}
void drev(void)
{
    dbias = REVERSE;
}
void dv1(void)
{
    dymax = 10;
}
void dv2(void)
{
    dymax = 20;
}
void ddraw(void)
{
    if (dbias == FORWARD)
        diacforward();
    else
        if (dbias == REVERSE)
            diacreverse();
}
void ifor(void)
{
    ibias = FORWARD;
}
void irev(void)
{
    ibias = REVERSE;
}
void iv1(void)
{
    ivmax = 10;
}
void iv2(void)
{
    ivmax = 20;
}
void idraw(void)
{
    if (ibias == FORWARD)
        diodeforward();
    else
        diodereverse();
}
void quit(void)
{
    exit(1);
}
`include "stdio.h"
`include "conio.h"
`include "stdlib.h"
`include "bkeys.h"
`include "window.h"

extern int VSG;

WINDOW *open_aenu(char *anae, MENU *an, int hsel); int gethaenu(MENU *an, WINDOW *hmenu, int hsel); int getvan(MENU *an, WINDOW *hmenu, int hsel, int vsel); int haccent(MENU *an, WINDOW *hmenu, int hsel, int vsel); void dimension(char *slc), int *ht, int *wd); void light(MENU *an, WINDOW *hmenu, int hsel, int d);

/* -- display and process a menu -- */
void menu_select(char *name, MENU *an) {
    WINDOW *hmenu;
    int sx, sy;
    int hsel = 1, vsel;
    curr_cursor(sx, sy);
    cursor(0, 26);
    hmenu = open_aenu(name, an, hsel);
    while(hsel = gethaenu(an, hmenu, hsel)) {
        vsel = 1;
        while(vsel = getvan(an, hmenu, &hsel, vsel)) {
            delete_window(hmenu);
            (*an+hsel-1)->func[vsel-1](hsel, vsel);
            hmenu = open_aenu(name, an, hsel);
        }
        delete_window(hmenu);
        cursor(sx, sy);
    }
/* -- open a horizontal menu -- */
static WINDOW *open_aenu(char *anae, MENU *an, int hsel) {
    int i = 0;
    WINDOW *hmenu;
    set_help("menu ", 30, 10);
    hmenu = establish_window(0, 0, 3, 80);
    set_title(hmenu, an);
    set_colors(hmenu, ALL, BLUE, AGWA, BRIGHT);
    set_colors(hmenu, ACCENT, WHITE, BLACK, DIM);
    display_window(hmenu);
    while((an+i)->anae) {
        wprintf(hmenu, ", 10.10s ", (an+i++)->anae);
        light(an, hmenu, hsel, 1);
        cursor(0, 26);
        return(hmenu);
    }
}
/* - get a horizontal selection - - - - */
static int getsubmenu(MENU *an, WINDOW *haenu, int hsel)
{
    int sel;
    light(an, haenu, hsel, 1);
    set_help("help ", 50, 50);
    while(TRUE) {
        switch(sel = get_char()) {
            case FWD :
                case BS :
                    hsel = haccent(an, haenu, hsel, sel);
                    break;
            case ESC :
                    return 0;
            case "\r" :
                    return hsel;
            default :
                    putchar(BELL);
                    break;
        }
    }
}

/* - pop down a vertical menu - - - - */
static int getvmenu(MENU *an, WINDOW *haenu, int *hsel, int vsel)
{
    WINDOW *vmenu;
    int ht = 10, wd = 20;
    char ***ap;
    set_help("vhelp ", 50, 50);
    while(TRUE) {
        dimension((an+*hsel-1)->aselcs, &ht, &wd);
        vmenu = establish_window(2+(*hsel - 1) + 12, 2, ht, wd);
        set_colors(vmenu, ALL, BLUE, AQUA, BRIGHT);
        set_colors(vmenu, ACCENT, WHITE, BLACK, DIM);
        set_border(vmenu, ACCENT);
        display_window(vmenu);
        ap = (an+*hsel - 1)->aselcs;
        while(*ap! = NULL) {
            wprintf(vmenu, "\n3s", *ap++);
            vsel = get_selection(vmenu, vsel, ");
            delete_window(vmenu);
            if(vsel == FWD || vsel == BS) {
                *hsel = haccent(an, haenu, *hsel, vsel);
                vsel = 1;
            } else
                return vsel;
        }
    }
}

/* - - manage the horizontal menu selection accent - - */
static int haccent(MENU *an, WINDOW *haenu, int hsel, int sel)
{
    switch(sel) {
        case FWD :
            light(an, haenu, hsel, 0);
            if((an+hsel)->name)
                hsel++;
            else
                hsel = 1;
            light(an, haenu, hsel, 1);
            break;
        case BS :
            light(an, haenu, hsel, 0);
if (hsel == 1)
    while ((m + hsel)->name)
        hsel++;
else
    --hsel;
light(m, hmenu, hsel, 1);
break;
default :
    break;
}
return hsel;

/*/ - - - compute menu's height and width - - - */
static void dimension(char *sl, int *ht, int *wd)
{
    unsigned strlen(char *sl)
    *ht = *wd = 0;
    while (sl[*ht])
    {
        if (strlen(sl[*ht]) > *wd)
            *wd = strlen(sl[*ht]);
        /* *wd = max(*wd, strlen(sl[*ht])); */
        (*ht)++;
    }
    *ht += 2;
    *wd += 2;
}

/* - - accent a horizontal selection menu - - - */
static void light(MENU *mn, WINDOW *hmenu, int hsel, int d)
{
    if (d)
        reverse_video(hmenu);
    wcursor(hmenu, (hsel - 1) * 12 + 2, 0);
    wprintf(hmenu, (m + hsel - 1)->name);
    normal_video(hmenu);
    cursor(0, 26);