```
(*$C Startup with login time bookkeeping, A-DATA 170824 *)

Program autostart;

(* This program is intended for autostarting.
 * It gives a welcome message together with the actual time.
 * It also administrates the bookkeeping of login date and time on the root
 * volume. Sector 3 on the disk is used for this purpose. See the dirtype
 * declaration (in the main program).
 * The system date is also changed, if necessary.
 *
 * Installs unit #10 in the system
 * Redirects the printer port for virtual printing to a PC.
 * Can also load multitasking support.
 *
 * IMPORTANT! Code type MUST be M_9900, because of direct VDP RAM access.
 *
 * A-DATA 170824  *)

   uses commandio,      (* redirection *)
        screenops,      (* date setting *)
        realtime,       (* clock reading *)
        support;        (* Screen color *)

   const
      rddata  = -30720; (* VDPRD *)
      rdstat  = -30718; (* VDPST *)
      wrtdata = -29696; (* VDPWD *)
      wrtaddr = -29694; (* VDPWA *)
      wrtenab =  16384; (* hex 4000, setting VDP address to write *)
      pabtbl  =  10716;  (* hex 29dc *)
      maxlen  =     43;
      valider = 'Has logtime '; (* Indicates valid login time on disk *)
      sysdate =  13840;  (* Address of system date in memory *)

   type
      byte = 0..255;
      window = record
         case boolean of
           true: (int: integer);
           false:(ptr:^integer);
         end;

      convdate = record
         case boolean of
           true :(int:integer);
           false:(dat:sc_date_rec);
         end;

      chartype=string[40];

      dirtype=record
        fill1:packed array[0..255] of char;    (* sector 0 *)
        fill2:packed array[0..255] of char;    (* sector 1 *)
        fill3:packed array[0..255] of char;    (* sector 2 *)
        logvalid:packed array[0..11] of char;  (* sector 3 *)
        logtime :string;
        fill4:packed array[0..161] of char;
        fill5:packed array[0..5] of char;      (* sector 4, Pascal directory *)
        dvid:string[7];
        fill6:packed array [0..5] of char;
```

```
        dlastboot:sc_date_rec;
        fill7:packed array[0..489] of char;    (* sector 4 and 5 *)
      end;

      texttab=array[1..12] of string[3];


   var
      cpuaddr: window;
      curlen,period,vdpaddr: integer;
      savevdpaddr: integer;
      ch: char;
      unitno: integer;
      i:      integer;
      pabname:string[43];

      info:sc_info_type;
      currdate,newdate:sc_date_rec;
      twindate:convdate;
      dateout:chartype;
      directory:dirtype;
      monthtab:texttab;         (* For text display of month *)
      printer:text;
      actdate:tdate;
      datestr :string;
      rtc_ok :boolean;


procedure swapbyte(var x:integer);

(* This procedure takes a word and reverses the order of the bytes *)

   type
      byteword = record
         case boolean of
         true: (addr:integer);
         false:(bytes: packed array[1..2] of byte);
       end;

   var
      word:  byteword;
      tbyte: byte;

   begin
      with word do
         begin
         addr := x;
         tbyte := bytes[1];
         bytes[1] := bytes[2];
         bytes[2] := tbyte;
         x := addr;
         end;  (* with statement *)
      end;  (* procedure *)

  procedure wrtvdpaddr (vdpaddr : integer);

    (* This procedure initializes the VDP ram chip to read/write from the *
     * address passed in the parameter vdpaddr.  *)

      begin
      cpuaddr.int := wrtaddr;
```

```
      swapbyte( vdpaddr );
      cpuaddr.ptr^ := vdpaddr;
      swapbyte( vdpaddr );
      cpuaddr.ptr^ := vdpaddr;
      end;  (* procedure *)

   function rdvdp (var vdpaddr : integer) : integer;

   (* This function reads a byte of data from the VDP ram address specified *
    * in the parameter vdpaddr.  *)

      begin
      wrtvdpaddr( vdpaddr );
      cpuaddr.int := rddata;
      rdvdp := cpuaddr.ptr^ div 256;  (* Right justify byte in word *)
      vdpaddr := vdpaddr + 1;
      end;  (* procedure *)

   procedure wrtvdp( var vdpaddr : integer;
                         data    : integer);

   (* This procedure writes the byte of data passed in the parameter *
    * data to the VDP ram address specified in vdpaddr.  *)

      var
         temp : integer;

      begin
        temp := vdpaddr + wrtenab;  (* Write enable the address *)
        wrtvdpaddr(temp);
        cpuaddr.int := wrtdata;
        cpuaddr.ptr^ := data * 256; (* Left justify byte in word and write *)
        vdpaddr := vdpaddr + 1;
      end;  (* procedure *)


procedure change_color;
 (* Changes screen color to black on white *)

type
  chrptr = ^char;

var
  temp: integer;
  chpt: chrptr;

begin
  temp := 10269;
  moveleft(temp,chpt,2);
  chpt^ := chr(31);
  temp := 11453;
  moveleft(temp,chpt,2);
  chpt^ := chr(31);
  set_scr_color(1,15);
end; (* change color *)


procedure poke(addr,value: integer);
  (* Stores value at addr in CPU RAM *)

var memaddr: window;
```

```
begin
  memaddr.int := addr;
  memaddr.ptr^ := value;
end; (* poke *)

procedure datetostring(thedate:sc_date_rec; var dateout:chartype);

(* Datetostring converts the internal date format to the display format.
   The array monthtab is not used, because the month is displayed lower-case
   but may be entered as lower- or upper-case. *)

var yearchar,monthchar,datechar:chartype;

begin
  with thedate do
  begin
    str(year,yearchar);
    str(day,datechar);
    case month of
       1:monthchar := 'Jan';
       2:monthchar := 'Feb';
       3:monthchar := 'Mar';
       4:monthchar := 'Apr';
       5:monthchar := 'May';
       6:monthchar := 'Jun';
       7:monthchar := 'Jul';
       8:monthchar := 'Aug';
       9:monthchar := 'Sep';
      10:monthchar := 'Oct';
      11:monthchar := 'Nov';
      12:monthchar := 'Dec';
    end;

    dateout := concat(datechar,'-',monthchar,'-',yearchar);
  end;
end;


procedure unit_10;
 (* Installes unit #10 in the system *)

const
  pointaddr = 11316;    (* Entry in unit type pointer table *)
  disk_kind = 11376;    (* Pointer to code for blocked units *)
  descraddr = 14094;    (* Address of unit descriptor *)

type
  dvidtype = string[7];
  dirtype = packed record
    fill1,
    dlastblk,
    dfirstblk :integer;
    dvid :dvidtype;
    deovblk :integer;
  end; (* dirtype *)

  unittype = packed record
    volume :dvidtype;
    is_blocked :boolean;
    blocks :integer;
```

```
  end; (* unittype *)

  dual = record
    case boolean of
      true  :(int :integer);
      false :(ptr :^unittype);
    end; (* dual *)

var
  window :dual;
  dir :dirtype;

begin (* unit 10 *)
  (* Insert blocked unit pointer in pointer table *)
  poke(pointaddr,disk_kind);
  (* Load something to begin with *)
  window.int := descraddr;
  unitclear(10);
  unitread(10,dir,sizeof(dir),2,0);   (* Fetch actual values *)
  with window.ptr^ do
  begin
    if (ord(dir.dvid[0])<=7) and (ioresult=0) then
    begin
      volume := dir.dvid;
      blocks := dir.deovblk;
    end
    else
    begin
      volume := '';
      blocks := 0;
    end;
  end; (* with *)
end; (* unit 10 *)


begin (* main program *)

  (* Write the welcome text *)
  change_color;
  page(output);
  gotoxy(0,2);
  writeln('Welcome to the UCSD p-system IV.0 at');
  (* Use actual time *)
  engtext(datestr);
  (* writeln(pos('Jan 01,',datestr));
  writeln(pos('00:0',datestr));
  writeln(datestr[18]); *)
  rtc_ok := not ((pos('Jan 01,',datestr)=1) and
   (pos('00:0',datestr)=14) and (datestr[18] in ['0','1']));
  if rtc_ok then
    writeln(datestr)
  else
    writeln('RTC battery low. Clock restarted.');
  writeln;

  (* Read and change login and system date in directory *)
  unitread(4,directory,sizeof(directory),0);
  with directory do
  begin
    (* Write last login time *)
    write('Last ',dvid,' login ');
```

```
  (* Use best date available *)
  if logvalid=valider then
  begin
    writeln(logtime);
  end
  else
  begin
    datetostring(dlastboot,dateout);
    writeln(dateout);
    logvalid := valider;
  end;

  if rtc_ok then
    logtime := datestr;   (* Set new login time *)

  (* Set the new system date, if a better one is available *)
  if rtc_ok then
  begin
    readdate(actdate);
    with dlastboot do
    begin
      if month>actdate.month then
        year := (year+1) mod 100;
      month := actdate.month;
      day := actdate.date;
    end; (* with dlastboot *)
    sc_use_info(sc_get,info);
    info.sc_date := dlastboot;
    sc_use_info(sc_give,info);
    twindate.dat := dlastboot;
    poke(sysdate,twindate.int);
  end; (* if rtc_ok *)
end; (* with directory *)

(* Write information back to directory *)
unitwrite(4,directory,sizeof(directory),0);

unit_10;      (* Install unit #10 *)

(* Start modification of the RS232 port *)
gotoxy(0,10);
writeln('Modifying the PRINTER:-port');
writeln;
writeln('RS232/2:');
writeln('  Baudrate : 9600');
writeln('  Data bits:    8');
writeln('  Parity   : none');
unitno := 6;

 (* Find the address of the PAB entry in VDP ram.  Read the current *
  * device definition and display it to the user.  If no device     *
  * currently defined, exit the program.                            *)

 cpuaddr.int := pabtbl + unitno * 2;
 vdpaddr := cpuaddr.ptr^ + 17;
 savevdpaddr := vdpaddr;
 curlen := rdvdp(vdpaddr);
 if curlen > 0 then
    begin
    pabname[0] := chr(curlen);
    for i := 1 to curlen do
```

```
        pabname[i] := chr(rdvdp(vdpaddr));

      pabname := 'RS232/2.BA=9600.DA=8.PA=N';

  (* Automatic setting of the type above *)

      period := 0;
        for i := 1 to length(pabname) do
        begin
          if (period = 0) and (pabname[i] = '.') then
            period := i - 1;
          if pabname[i] in ['a'..'z'] then
            pabname[i] := chr(ord(pabname[i])-32);
        end;  (* for loop *)

      if period = 0 then
        period := length(pabname);
      vdpaddr := savevdpaddr-17;
      for i := 1 to 2 do
        wrtvdp(vdpaddr,0);
      wrtvdp(vdpaddr, period);
      for i := 1 to 3 do
        wrtvdp(vdpaddr, 0);
      vdpaddr := savevdpaddr;
      for i := 0 to length(pabname) do
        wrtvdp(vdpaddr, ord(pabname[i]));
      unitclear(unitno);
      end;

      (* Enable multitasking if required *)
      gotoxy(0,18);
      write('Enable multitasking? ');
      read(ch);
      writeln;

      if ch in ['Y','y'] then
      begin
        chain('*multitask');
        writeln('Loading multitasking support');
      end;

  (* Sets default prefix to drive #5 *)

      if not redirect('P=#5') then
        exception(true);
end.
```