OCTOBER 1985
VOL.6, NO.10

PEEK (65)

The Unofficial OSI Users Journal

P.O. Box 347
Owings Mills, Md. 21117
(301) 363-3268

INSIDE

Column One

Maybe it is the temptations of summer and the great outdoors, but whatever it is, the summer doldrums are now over. "Thank Goodness", say the dealers and software houses, but it has always been that way in the summer. It's back to school, back to the grind, and things are beginning to happen again! A quick look at the index will show 12 authors and 12 subjects in this issue, covering an obviously wide range of topics. And again there are things cropping up on the manufacturing front.

At Isotron, the recent price reductions have made a big volume improvement even though the 700 series was not included in the cuts. Shortly after you read this, OSI is expecting to release an upgraded version of the 700 operating system to accommodate virtual memory - which is only possible with the 68010 and later CPUs. Aside from the obvious speed, it will also significantly increase disk accessibility as only a part of the program that is needed is loaded to memory. That, in turn means more memory free for other users/uses. Dealers, keep your eyes open for soon to be announced compilers for BASIC, FORTRAN and, by year end, an OS-U compiler. Within a month or two, look for COBOL RM and PHILON. On top of that, there are still a number of items that fall into the "Shhh!" category that are being held until COMDEX for release.

Meanwhile, the folks at DBI are putting the final touches on their vastly improved and expanded operating system, OS65-E. Alpha testing is going very well and beta is expected by year end. We are reserving space to tell you more about 65-E next month. The second announcement from DBI is that, by about the time you read this, networking should be officially out. It is done, unlike OSI, through the SCSI bus and can accommodate up to 75 users. Either 5 of the large or small machines can be hooked together (5 x 7 users or 5 x 15 users) and will be compatible with both OS-U and 65-E. The purpose is to maintain compatibility and continue support for the existing base of OS-U users.

After all the good news, here is some bad. You are letting us down. By this time last year, we were flooded with entries for our free listing of software. So much so that we again had to spill over into a second issue. Yes, we have received some great things, but... need I say more? It's FREE! See the form in the September issue!

You say your machine sometimes does weird things, Bunkie? Well, we are glad to have the efforts of two TOSIE true-blue's (John Horeman and Paul Chidley) who have attacked and resolved a number of the damned nuisances that may have left you high and dry. Salt these tips away. They may save you a trip to the doctor later.

What's that about "C" on OSI? Kent Behrends lets the tip of the iceberg show. Gee, Kent, I sure hope that there is more coming!

Who says the SB II is dead? Olof Swembel, Herbert Grassel, and John Whitehead gang up on CEGMON, ASM DABUG and other hardware that help these little rascals grow into quite respectable members of the family.

At the other end of the spectrum, Roger Clegg is tickling your thinking powers with yet more sort routines for OS-U. Put them all together and you will have quite an arsenal.

Ol' Wazzat Jankowski's at it again. Read, drool at what he has done and then just sit down and start typing in code.

Back in the OS-U department, Danny Schwartz comes to the rescue of the new breed of OS-U users - that's the type who run "U" on video machines and Bob Ankeney gives us the rundown on the Portland Board. Or is it the 517 board?

That leaves us with our old friend Earl Morris ever branching out in new directions. How about photographs on disk? Why not? He has slow-scan TV images on disk!

Now that doesn't sound like the summer doldrums, does it?

Eddie

TIME FOR "C"

By: Kent Anthony Behrends
17309 Mapes Ave.
Cerritos, CA 90701

I've been working and using my OSI since early '78. From then my C2-8P has grown from 32K video system to a 56K serial/video system running 65D and various CP/M editors, compilers and word processors. I now develop programs for VAX's, mostly in "C" and FORTH-83. Here is a set of routines and a program which I have used extensively in writing and modifying CP/M public domain programs.

TIME.C came out of a need for some long term data collection routines to record the time and date along with their data. An example might be to record when the wind changes direction or velocity, or when someone logged on thru BBS (bulletin board system).

The CA-20 clock is not directly accessible by the programmer. To access the clock, you must work through a pia (programmable interface adapter). You first configure the pia for reading (routine Init_read) or writing (routine Init_write). Once the pia is set up, read the current time/date (routine Read_clock) or write a new time/date (routine Set_clock). With all the support routines available, the main program is just a user interface that parses out the user requests and displays or sets the CA-20 clock. Other programs can just link to these support routines to access the clock in a uniform manner.

Look through this program. It is documented fairly well, and can be used as an example for other OSI control programs. If you don't know "C", you can pick up some by looking at these functions and comparing them to the OSI BASIC routines which do the same. Have fun!

Copyright • 1985 PEEK (65) Inc. All Rights Reserved.
published monthly
Editor - Eddie Gieske
Technical Editor - Brian Harston
Circulation & Advertising Mgr.-- Karin Q. Gieske
Production Dept. - A. Fusselbaugh, Ginny Mays

Subscription Rates	Air	Surface
US		$19
Canada & Mexico (1st class)		$26
So. & Cen. America	$38	$30
Europe	$38	$30
Other Foreign	$43	$30

All subscriptions are for 1 year and are payable in advance in US Dollars.
For back issues, subscriptions, change of address or other information, write to:
PEEK (65)
P.O. Box 347
Owings Mills, MD 21117 (301) 363-3268
Mention of products by trade name in editorial material or advertisements contained herein in no way constitutes endorsements of the product or products by this magazine or the publisher.

```
/***********************************************************************
*   time.c
*
*   This set of routines will read and write the osi ca-20 clock
*
*   Copyright (c) 1985 by kent anthony behrends
*   This software may be used for non-commericial purposes only
*   No commericial use of TIME may be make without author's express
*   written permission.
*   -----
*   created: kent anthony behrends       1 feb 1985 ver 01-000
*      This program will read and write the ca-20 clock board on the osi
*      computer
*   -----
*   Mod history in reverse order (most recent first)
*   -----
*   modified: kent anthony behrends     24 jul 1985 ver 01-500
*      This mod (1)allows time.c to be linked with set.c if SET is
*      TRUE and (2)makes time.c a set of easyly callable routine for
*      other C programs to use the ca-20 clock. They first call
*      clk_data_init then may use any the the functions
*   -----
*   modified: kent anthony behrends     10 jun 1985 ver 01-250
*      This mod (1)replaces all peeks and pokes to pointer offsets for
*      a little speed and (2)nicer looks
*   -----
*   use:
*       A0) TIME [-d][-h][-s curr-time]
*
*   where:
*           --- displays day month date hh:mm:ss.tt
*       -d  --- displays just day/date
*       -h  --- displays just hour/minute/second
*       -s  --- sets the clock to curr-time
*
*           curr_time is all or part of the format: fri 1 feb 12:01
*
*   examples:
*       A0) time (cr)             --- displays time/date
*       A0) time -h           --- displays the time
*       A0) time -s fri 1 feb 12:01 --- sets full time/date
*       A0) time -h -s 12:10         --- sets then display hour
*
*----------------------------------------------------------------------*/

#include (bdscio.h)      /* bds header */
#include setdefs.h       /* set.c header */

#define SET FALSE        /* YES if using these routine within SET.c */
                         /* NO if using standalone */
                         /* ----- */
                         /* Set.c allows this to be used as one of the
                            functions within set.  This function is
                            used as follows:
                             A0) SET TIME (cr)
                             A0) SET TIME cur-time (cr)
                            The first just displays the time. The
                            second sets all or pard of the current
                            time (SAT 27 JUL 14:30)
                            If argc == 0 then a single line help
                            line is displayed */
#if (SET)
#include setdefs.h                                  /* get set.c header */
    #define  MAIN time(argc,argv)                   /* main routine name */
    #define  EXITOK return ERR_TIME:ERR_NONE     /* error exit ok */
    #define  EXITER return ERR_TIME:ERR_UNKNOWN /* error exit error */
#else
    #define  MAIN main(argc,argv)
    #define  EXITOK exit()
    #define  EXITER exit()
#endif
/*
        define the pia addresses
*/
#define MONTH   7        /* pia address for month */
#define DATE    6        /* pia address for date */
#define DAY     5        /* pia address for day */
#define HOUR    4        /* pia address for hour */
#define MINUTE  3        /* pia address for minute */
#define SECOND  2        /* pia address for second */
/*
        days and months strings
*/
#define DAYS    "SunMonTueWedThuFriSat"
#define MONTHS  "JanFebMarAprMayJunJulAugSepOctNovDec"

#define CLOCK   51076       /* base clock address */
char    *pia_base;          /* pointer to the base of the clock pia */

char    lables[36];         /* days and months holder */

char    set_time;           /* if TRUE set the time */
char    show_all;           /* if TRUE show full date/time */
char    show_date;          /* if TRUE show the day/date */
char    show_hour;          /* if TRUE show the hour:min.sec */
char    show_time;          /* if TRUE show the time */

char    *buff_ptr;          /* pointer to time_buff */
#define BUFF_LEN 20
char    time_buff[BUFF_LEN];    /* buffer for day date mmm hh.mm:ss */

char    *time_ptr;              /* pointer to set time string on command line*/

int read_data[11];              /* data for read init */
int write_data[11];             /* data for write init */
```

Continued

```
/*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
* Init_write
*       init the ca-20 clock board for writing
*------------------------------------------------------------------------*/
init_write()
{
    int i;
    for (i=0; i<12; i+=2) *(pia_base+write_data[i])=write_data[i+1];
}
/*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
* Init_read
*       init the ca-20 clock board for reading
*------------------------------------------------------------------------*/
init_read()
{
    int i;
    for (i=0; i<12; i+=2) *(pia_base+read_data[i])=read_data[i+1],
}
/*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
* Within(sub_string,string)
*    --- returns index of substring +1 / returns 0 is not found
*       SUB_STRING  pointer to a string to find
*       STRING      pointer to a string
*------------------------------------------------------------------------*/
within(sub_string,string)
char *sub_string,*string;
{
    int i,j,match;
        for (match=FALSE,i=0, i<strlen(string)-strlen(sub_string)+1 && !match, i++)
        for (match=TRUE,j=0; j<strlen(sub_string) && match; j++)
            if (toupper(string[i+j])!=toupper(sub_string[j])) match=FALSE;
    return match ? ++i : 0;
}
/*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
* set2_clock(section,data)
*    --- lowest form of set time. This does the actual
*       pia addressing.
*       SECTION     pia address
*       DATA        data for pia addressed at SECTION
*------------------------------------------------------------------------*/
set2_clock(section,data)
int section,data;
{
    char c;
    if (data>10) data = ((data/10)*16)+(data-(data/10*10));
    *(pia_base)=section;              /* addrerss the pia */
    *(pia_base+2)=data;               /* place new value to pia */
    while (!(*(pia_base+3)&128));     /* wait for pia read */
    c = *(pia_base+2);                /* reset crb-7 */
    *(pia_base)=2;                    /* close the clock */
    *(pia_base+2)=0;                  /* no interrupts */
    while (!(*(pia_base+3)&128));     /* wait for pia read */
    c = *(pia_base+2);                /* reset crb-7 */
}
/*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
* set_clock(i,ptr)
*    --- set the ca-20 clock to specified value all or part of the time
*       can be set
*       I       number of arguments to parse
*       PTR     a pointer to a pointer of characters
*
*           A) TIME -s mon 22 jan 14:50
*           A) TIME -s 14:50
*           A) TIME -s jan 22
*------------------------------------------------------------------------*/
set_clock(i,ptr)
int i;
char **ptr,
{
    int j;          /* reserve a variable for status return */
    while (--i)     /* still more arguments? */
    {
        *++ptr;     /* bump pointer to next character */
                    /* scan the string looking for the end */
                    /* or a ':' */

        for (j=0; ptr[0][j] != '\0' && ptr[0][j] != ':'; j++);

        if (ptr[0][j] == ':')   /* if its a ':' then we set hour/min */
        {
            set2_clock(HOUR,atoi(*ptr));
            set2_clock(MINUTE,atoi(*ptr+j+1));
        }
        /*
            if this argument did not have a ':' in it then it
            can be the date,day, or month. If it's a number then
            we set the date to that number. If it can be found in
            the string DAYS the we set the day. If it can be found
            in the string MONTHS then we set the month
        */
        else if (j = atoi(*ptr)) set2_clock(DATE,j);
        else if (j = within(*ptr,DAYS)) set2_clock(DAY,(j/3)+1);
        else if (j = within(*ptr,MONTHS)) set2_clock(MONTH,(j/3)+1);
    }
}
/*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
* read_clock(section)
*    --- read a section of the clock and place in the
*       buffer pointed to by *buff_ptr
*
*       section can be: MONTH DAY DATE HOUR MINUTE SECOND
*------------------------------------------------------------------------*/

read_clock(section)
int section;
{
    int i;
    char c;
```

Continued

QUESTIONS AND ANSWERS

By: John Horemans
Courtesy of TOSIE
Toronto Ohio Scientific Idea Exchange
P. O. Box 29
Streetsville, Ont.
Canada L5M 2B7

I have just installed an EPROM monitor chip (C1E, C1S, ROM-TERM) and it doesn't work. What did I do wrong?

There are several possibilities. Have you jumpered pin 21 to +5 volts? There is a pad below the monitor ROM to do this. If you had a SYN600 monitor, do NOT invert the chip select to pin 18. The SYN600 has a low chip enable, just as the 2716 you are installing. If you had a 7287 ROM, you will need to invert the chip select. Both chip select low and chip select high are found on a pair of pads just to the left of the 74LS04 just above and to the left of the monitor ROM. You should now be up and running, at least at 1 MHz. To get 2 MHz with a 450 ns 2716 chip, you will need to leave the output enabled all the time. Do this by putting pin 20 to 0 volts. The outputs will now be enabled at all times, the chip can now be accessed in 120 ns, at a penalty of extra power consumption, as it will no longer power down between accesses. Maximum standby current is rated at 25 ma, while the active current is a maximum of 100 ma.

My 610 board (or other memory expansion) is not reliable. The failure is random, and the memory chips don't seem to be the problem. What do I do?

You can try to change the resistors on the DD line (data direction - it changes the direction the 8T28 buffers send data). OSI uses a 220 Ohm pullup resistor. This requires a lot of power to overcome, many of the driver chips cannot do this quickly enough. Try changing the two resistors. I found that 2K for R74 and 1K for R9 works well. These resistors are between the keyboard and the 8T28 buffer chips, and aren't too hard to change. Once this was fixed, the 610 board memory has been reliable. On the 505 CPU board for the bigger machines, OSI uses a 4.7K pullup resistor, so you may need to go even higher.

My computer locks up when I run certain programs. The main symptom is a flickering

```
    *(pia_base)=section;         /* set pia/clock address */
    *(pia_base+1)=54;            /* force clock read */
    c=*(pia_base+2),             /* read clock data from pia */
    *(pia_base+1)=62;            /* close clock port */
    i=(((c&240)/16)*10)+(c&15); /* BCD -> int */
    return i;
}
/*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
*   clk_init_data()
*       This routine inits all internal data for time.c. This makes
*       it easy for other programs to use these routines
*-------------------------------------------------------------------------*/
clk_init_data()
{
    time_ptr = NULL;          /* init the time_pointer to NULL */
    pia_base = CLOCK;         /* init the base pia pointer to pia */
    set_time = show_all = show_date = show_hour = FALSE,

    initw(write_data,"1,0,0,31,1,4,3,34,2,255,3,38");
    initw(read_data,"1,58,0,31,1,62,3,58,2,0,3,62");
}
/*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
*   Main-routine
*-------------------------------------------------------------------------*/
MAIN                                /* main routine name depends on SET */
int argc;                           /* # of arguments +1 */
char **argv;                        /* pointer to a pointer of arguments */
{
    int i,j;
    char c,*ptr;

    clk_init_data();                /* init all data used here */
                                    /* this is for programs that use */
                                    /* time.c as a library */
    if (argc == 1) show_all = TRUE;

#if (SET)
    /*
        If routines to be used in SET then no command line parsing
        is needed. Command is assumed to be -s or set. if argc==0
        then the user has asked for help and sample command line
        is displayed. If nothing was typed after the SET TIME then
        the current date time is displayed.
    */
    if (argc == 0)                  /* set asked for call line example */
    {
        puts("     A0) Set time [24 [jul [10:00:00]]]\n");
        EXITOK;
    }
    set_time = (argc > 1);          /* enough params for set time */
    i = argc;                       /* i for set time later */
    ptr = &argv[0];                 /* pointer to string before time string */
#else
    /*
        If program is to be run standalone then we must parse out
        the command line for -d,-h, and -s.
    */
    while (--argc && !set_time)     /* parse out the line */
    {
        ++argv,                     /* bump pointer */
        if (**argv == '-')          /* flag? */
        {
            switch(c = argv[0][1])
            {
                case 'D':           /* date request */
                    show_date = TRUE;
                    break;

                case 'H':           /* hh:mm request */
                    show_hour = TRUE;
                    break;

                case 'S':           /* set request */
                    set_time = TRUE;
                    if ((i=argc) == 1) goto usage;
                    ptr = argv;
                    break;

                default: goto usage;
            }
        }
    }
    if (!(show_all || show_date || show_hour || set_time))
    {
usage:      puts("Usage: A) TIME [-d][-h][-s cur-time]\n");
        puts("Where:\n");
        puts("      -D: show the date\n");
        puts("      -H: show the hour:min:sec\n");
        puts("      -S: set the time to cur-time\n\n");
        exit();
    }
#endif
/*****
    Do what was asked
*****/

    if (set_time)
    {
        init_write();                   /* init ca20 for write */
        set_clock(i,ptr);
    }
    if (!(show_all || show_date || show_hour)) EXITOK;
    init_read();
    setmem(time_buff,BUFF_LEN,32);      /* time filled w/spaces */
    buff_ptr = &time_buff;              /* init the buffer pointer */
    if (show_date || show_all)
    {
```

Continued on page 6

screen and a dead keyboard. What is wrong?

You have met the garbage collection bug. Once you dimension a string array like A$(15), and you then hit a garbage collection you get these symptoms. A garbage collection process cleans up the memory of unused leftover strings. Whenever you concatenate 2 strings, a new one is formed, with the 2 old ones left in memory, now useless. Eventually memory fills, and the garbage collector is invoked. There are software fixes. Most don't work well, and are a nuisance to implement. It is far easier and better to get the fixed ROM. Note to disk users: Only HEXDOS users need worry about getting a fixed ROM. OSI's DOS includes the BASIC on disk, which doesn't have this same problem.

How do I install the fixed BASIC 3 EPROM?

Note that your fixed BASIC 3 is in EPROM, while OSI's are in a 2316 ROM. You will need to invert the chip select to pin 18. Cut this line on the bottom of the Superboard/C1P, and jumper it to the other ROMs. The idea is to cut it off the BASIC 3 chip, but still have it continue to the other BASIC chips. Now bring in the chip select low, by getting it before the LS04 inverts it. On the rev D machines, U16 is the inverter, on the rev B U18 is the inverter. Rev. B schematics indicate a jumper pad to do this. Rev. D needs a wire under the board. Take the chip select before it is inverted and bring it to pin 18. Pin 20, output enable, needs to be low. Cut it from the other chips, be sure to jumper over to the other BASIC chips again, and connect pin 20, either to pin 18 of the same chip, or connect it to 0 volts. The discussion about power consumption, (see the previous question) applies here. Now we need to cut and put pin 21 to +5 v. Remember again not to isolate the remaining BASIC chips by jumpering the signals through to them. This is a little bit of a chore, but well worth having.

Can the BASIC be put into 2732 or a 2764 chip?

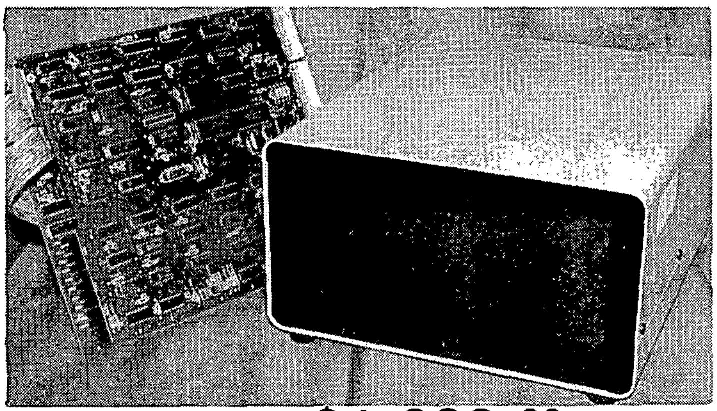
Certainly! You will need to change the decoding arrangements slightly, but there is

Continued on page 6

SUPER HARD DISK Subsystem!

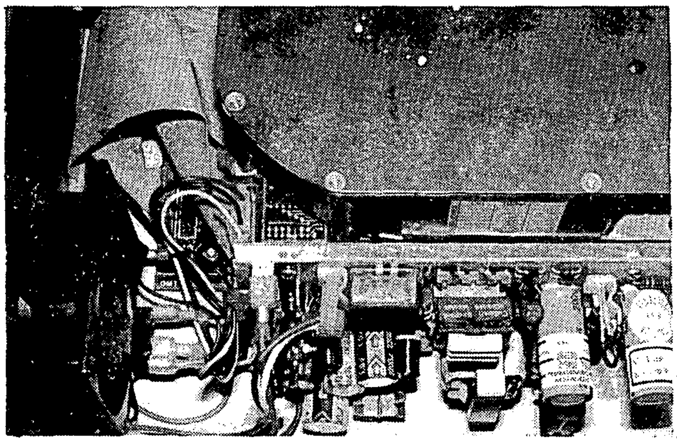
TURNS ANY FLOPPY BASED COMPUTER INTO HARD DISK BASED, INSTANTLY.

- PLUGS INTO ANY OSI TYPE BUS
- ONE RIBBON CABLE CONNECTS TO DRIVE
- COMPLETELY SELF CONTAINED
- 32 BIT ERROR DETECTION AND CORRECTION
- HAS REAL TIME CLOCK
  *CALENDAR W/BATTERY ON SCSI ADAPTER BOARD
- CAN BOOT DIRECTLY FROM OSI 505/510 CPUs OR DENVER BOARDS W/SCSI PROM
- IDEAL BACK-UP FOR ALL OSI HARD DISK COMPUTERS



FROM **$1,999.00**

The SPACE-COM SUPER SUBSYSTEM Uses 5¼" Industry Standard Hard Disk drives interfaced to the OSI bus by the DS-1 SCSI Host Adapter Board at the computer end and the state of the art OMTI 5000 series Intelligent Disk/Tape Controllers at the disk end. The Denver DS-1 Board not only provides the Bus Translation, but gives Real Time of Day, Day/Week, AM/PM, and Day/Mo. With on board battery, Date and Time are maintained w/o power.



The chassis is beautifully engineered with lighted on/off switch, standard a/c cord, and insulated spade terminals for easy service. A Corcom Emi Filter is incorporated in the a/c jack, and power is provided by an extremely efficient switching power supply. The case is also available in dual, side by side configuration and looks like an IBM PC box. It incorporates a larger power supply and can support 2 Winchester drives, or 1 drive and tape, or 2 5" floppies in place of one of the above.

Drives can be accessed from any single or multi-user OSI system by running an overlay program on that partition, or can be booted directly by replacing current ROM/PROM with our SCI 500 PROM, available for $49.00 extra.

Single 20 M/B drive (15.7 formatted) single case	$1,999.00
Single 26 M/B drive (21 formatted) single case	$2,199.00
Dual 20 M/B drives (31.4 formatted) dual case	$2,999.00
Dual 26 M/B drives (42 formatted) dual case	$3,299.00
Super Fast 85 M/B drive (70 formatted) single case	$3,999.00
Dual 85 M/B drives (140 formatted) dual case	$6,699.00

SPACE-COM International
14661A Myford Road, Tustin, CA 92680 (714) 731-6502

```
        strcpy(ptr = &lables,DAYS);     /* init lables to days of week */
        for (ptr += 3*(read_clock(DAY)-1),j=0; j(3; j++) *buff_ptr++ = *ptr++;
        sprintf(buff_ptr," %02d  ",read_clock(DATE));
        buff_ptr +=4;
        strcpy(ptr = &lables,MONTHS);
        for (ptr += 3*(read_clock(MONTH)-1),j=0; j(3; j++) *buff_ptr++ = *ptr++;
    }
    if (show_hour || show_all)
    {
        sprintf(buff_ptr," %02d:%02d:%02d",
            read_clock(HOUR),read_clock(MINUTE),read_clock(SECOND));
        buff_ptr +=9;
    }
    buff_ptr = NULL;
    printf("The time is %s",time_buff);

#if (SET)
    EXITOK;
#endif
}
```

no real problem. 2732's will fit right into the sockets, and only need to have A11 brought to pin 18, and the chip enable transferred to pin 20. You need to take the A11 signal out of the decoding. You use the signal to select the left or right set of BASIC sockets. The 28 pin 2764's take a bit more ingenuity. They are pin compatible, but you will have 4 pins sticking out over the edge of the socket. Not too tidy, but possible. Consider putting 2K RAMS in the remaining sockets if you do any of these.

I know 2114 chips take a fair amount of power. How much exactly?

A regular 2114 is rated for an input load current of 95 ma maximum, with 80 ma typical. The low power version 2114L or 21L14 is rated at 65 ma maximum. So 8K x 8 bits of regular 2114's take 16 x 95 ma, or 1.5 amps. The low power types will take about 1 amp for the 8K. Moral: check your power supply when adding this type of memory.

I have a 48 pin board hooked to my Superboard. Everything seems OK, but it won't work. It runs fine on my friend's 48 pin system.

Check the buffer chips. The Superboard uses 8T28 non-inverting chips, while the 48 pin bus uses 8T26's, the inverting cousin. The pinout is the same, so you can switch. If you are using the 610 board, you will have to use 8T28's on all the boards. If not, you may well be able to use either, as long as both sides are the same. You may also want to check the DD (data direction) diodes and pullup resistors, check some of the other questions for more information.

What is the best memory checking progam you know about?

Without question, it's Mike Putnam's single disk compare. It is meant to check one disk against another, but really shows its colors if you do a compare, leaving the same disk in the drive! Use a disk loaded with information, so your memory will be fully loaded. The program then goes back to compare memory to disk. In case of problems, it announces the disk track and sector, as well as the memory area giving problems. Note that the kind of nasty memory problems that I have had to deal with have often escaped detection by conventional memory tests, but readily show up with the disk compare. One more thing, Compare loads itself quite low into memory, and tests most of the resident language area, as well as the DOS memory area. It, as well as the closely related single disk copier, loads 23 tracks into my 48K machine, meaning that all but a few of the lower memory chips are tested. Of course, this all assumes your drive is working properly!

I have heard that germanium diodes are better. Is this true?

Germanium diodes have a lower forward voltage drop, and may help some circuits switch faster. I use them on the 48 pin boards to switch the Data Direction line. A 527 memory board I use was unreliable at 2 MHz until I removed the silicon diode and switched to germanium.

THE SEB-1 PARALLEL PORT HEXDOS 4.0 AND OS65D 3.3

By: Herbert H. Grassel
12838 Flack Street
Wheaton, MD 20906

The following program additions and modifications were written for a Superboard II computer with 610 and SEB-1 expansion boards, two mini-floppy disk drives and OSI and C1E+ ROMS. HEXDOS 4.0 is dated November 1982, OS65D 3.3 is dated July 1983.

When I originally purchased the Grafix SEB-1 expansion board, I figured that even if I never did much with the graphics, I could always use the extra memory located at \$8000 - \$97FF (32768 - 38911) and the Parallel Port, a MCS-6522 VIA located at \$9800 (38912). As it turned out, I use the 6K of HRG memory to store machine language routines, and the VIA to drive an EPSON MX-80 printer.

MISPRINTS AND MISSING ROUTINES

Getting the parallel port to work was a lot easier once I found the misprints in my copy of the SEB-1 manual. On page six, the decimal values given in the Memory Map Table are off by 1 after address 38915 and the pin-outs for CB1 and CB2 at J-3 are reversed on the schematic. The latter took a little while to find, but breaking the OSI information barrier, as always , proved to be the big challenge. It wasn't until I received my copy of the OS-65D 3.2 DIS-ASSEMBLY MANUAL from PEEK(65) that I found the real problem. Simply stated, the parallel print routine is MISSING from my version of 65D 3.3. The device 4 output vector still points to where the routine should be. All of the documentation that came with the DOS indicates there should be a print routine there -- but there isn't. In its place is part of the Keyboard Entry Routine. Imagine that!!! If anyone out there has any information about this particular version of DOS, please publish it.

HARDWARE HOOK-UP

The MX-80 provides several lines for different handshake schemes. I used the $\overline{\text{STROBE}}$ pulse on pin 1 to read data into the printer and the $\overline{\text{ACKNLG}}$ pulse on pin 10 to indicate that the data has been received.

The MCS6522 in turn provides

two peripheral control lines which can be programmed to act as interrupt inputs or as handshake outputs. For write handshaking, the processor generates the DATA READY signal (through the MCS6522) and the peripheral device must respond with the DATA TAKEN signal. The CB2 pin acts as a DATA READY output in the DC level or pulse mode and the CB1 pin accepts the DATA TAKEN signal, setting the interrupt flag and clearing the DATA READY output.

The Interconnect Table shown here lists the pin outs and respective signals for J-3 on the SEB-1 board and the Centronics Parallel input on the MX-8Ø.

PROGRAMMING THE MCS6522 VIA

Four Control Registers in the MCS6522 must be programmed before the Peripheral B Port will function as a parallel output port: Peripheral Control Register (PCR) at $98Ø3; Data Direction Register B (DDRB) at $98Ø8; Interrupt Enable Register (IER) at $98ØB; Auxiliary Control Register (ACR) at $98ØE. The Control Register Table lists their respective addresses and the values that program them to perform the described function.

UP AND RUNNING WITH HEXDOS 4.Ø

The HEXDOS parallel print routine, which begins at $ØA24 (2596), expects an output port and a handshake at $D9ØØ (55552). The SEB-1 parallel port is located at $98ØØ and the IFR handshake is at $98Ø7. Besides making these address changes, I moved the Auto Directory from Track Ø and in its place left a program that initializes the MCS6522, clears the parallel port, resets the printer if it's on line and runs the Directory, now stored on Track 2.

The sequence went like this. After initializing a new disk with FORMAT, re-boot the original disk and answer <RETURN> to "SELECTION ?". Insert the newly formatted disk and do a SAVE "DIR", then type NEW. Do a <BREAK>, get into the Monitor mode and make these changes:

```
$ØA27  $ØØ -->  $Ø7
$ØA28  $D9 -->  $98
$ØA3Ø  $D9 -->  $98
```

Do another <BREAK> and a Warm start. Now type in the following BASIC program:

INTERCONNECT TABLE

6522 SIGNAL	SEB-1 J-3	DIRECTION	MX-8Ø PORT	MX-8Ø SIGNAL
CB2	2	---------->	1	$\overline{\text{STROBE}}$
PBØ	1Ø	---------->	2	DATA 1
PB1	9	---------->	3	DATA 2
PB2	8	---------->	4	DATA 3
PB3	7	---------->	5	DATA 4
PB4	6	---------->	6	DATA 5
PB5	5	---------->	7	DATA 6
PB6	4	---------->	8	DATA 7
PB7	3	---------->	9	DATA 8
CB1	1	<----------	1Ø	$\overline{\text{ACKNLG}}$
CHASSIS-GND		<--------->	17	CHASSIS-GND
SIGNAL-GND		<----------	19-3Ø	SIGNAL-GND

CONTROL REGISTER TABLE

REGISTER	ADDRESS	DATA	FUNCTION
PCR	$98Ø3 (38915)	$8Ø (128)	Handshake output mode--Sets CB2 low on a write ORB operation. Resets CB2 high with an active transition of the CB1 input signal.
DDRB	$98Ø8 (3892Ø)	$FF (255)	Sets all 8 pins in ORB to data output mode.
IER	$98ØB (38923)	$9Ø (144)	Programs Interrupt Flag Register (IFR)--Sets bit 4 of IFR high on an active transition of the signal on the CB1 pin. Clears bit 4 on a read or write ORB.
ACR	$98ØE (38926)	$ØØ (ØØ)	Disables the Shift Register.
FCR	$98ØC (38924)	$FF (255)	Enables the HRG mode on the SEB-1 board. (Not required for parallel port operation.)

```
 1 POKE38915,128:POKE3892Ø,
   255:POKE38923,144:POKE
   38926,ØØ:POKE38924,255
 2 POKE38912,Ø:IFPEEK(38919)
   AND16THENPRINT#1,CHR$(27)
   ;CHR$(64);
 3 RUN"DIR"
```
and execute a SAVE#Ø,768.

If your disk drive won't write all of the initialization routine onto Track Ø, you can add line 2 to the Directory.

MODIFYING OS-65D 3.3

Under OS-65D 3.3, I put the MCS6522 initialization routine in the space on Track Ø from $2217 (8721) to $2234 (8756), most of which is already dedicated to setting up a parallel printer port. Then I put the parallel print routine in the space allocated for the UART I/O, $25ØD (9485) to $2526 (951Ø) and adjusted the vector

DISK DRIVE RECONDITIONING

WINCHESTER DRIVES

FLAT RATE CLEAN ROOM SERVICE.
(parts & labor included)

Shugart	SA4008	23meg	$550.00
Shugart	SA1004	10meg	$390.00
Seagate	ST412	10meg	$295.00

FLOPPY DRIVE FLAT RATES

8" Single Sided Shugart	$190.00
8" Double Sided Shugart	$250.00
8" Single Sided Siemens D&E Series	$150.00
8" Double Sided Siemens P Series	$170.00

Write or call for detailed brochure
90 Day warranty on Floppy & Large Winch.
1 Yr. Warranty on 5" & 8" Winchesters.

Phone: (417) 485-2501

FESSENDEN COMPUTERS
116 N. 3RD STREET
OZARK, MO 65721

at $2317/18 (8983/84) to point to the new location.

After booting up 65D 3.3 from Disk 5, copy Tracks Ø-39 onto a new disk. Boot the new disk, get into BASIC and type EXIT to get the A* prompt, then CA Ø2ØØ=Ø6,4 (CA Ø2ØØ=13,4 for 65D 3.2), and GO Ø2ØØ. This should bring up the Track Ø Read/Write Utility. Enter R42ØØ to read Track Ø into memory, E to return to the DOS, then EM to load the Extended Monitor.

Starting at 4217 (@4217) type in the following program to initialize the MCS6522.

```
4217 8E ØØ 98 STX PTRPIA
421A 8E ØE 98 STX PTRPIA+14
421D CA       DEX
421E 8E Ø8 98 STX PTRPIA+8
4221 A9 8Ø    LDA #8Ø
4223 8D Ø3 98 STA PTRPIA+3
4226 A9 9Ø    LDA #9Ø
4228 8D ØB 98 STA PTRPIA+11
422B 8E ØC 98 STX PTRPIA+12
422E EA       NOP
422F EA       NOP
423Ø 8E Ø1 DF STX KPORT+1
4233 A9 Ø4    LDA #4
```

Go to 45ØD (@45ØD) and type in the parallel print routine.

```
45ØD 6Ø       RTS
45ØE 48       PHA
45ØF A9 ØØ    LDA #ØØ
4511 2C Ø7 98 BIT PTRPIA+7
4514 1Ø FB    BPL FB
4516 68       PLA
4517 EA       NOP
4518 8D ØØ 98 STA PTRPIA
451B EA       NOP
451C EA       NOP
451D EA       NOP
451E 6Ø       RTS
```

Fill the space from 451F to 4526 with EA's. (This is a hybrid of the HEXDOS and 65D 3.2 routines.)

Change the Output Distribution Vector at $4317/18 (@4317) to point to the New Print Routine.

```
$4317  $9E -->  $OD
$4318  $24 -->  $25
```

Type EXIT, then CA Ø2ØØ=Ø6,4 and GO Ø2ØØ to bring up the Track Ø Read/Write Utility again and enter W42ØØ/22ØØ,8 to write the modified Track Ø on the new disk. Answer E to return to the DOS and type BASIC to load the interpreter. Enter RUN "BEXEC*" and open the system. All that remains is to add a line to BEXEC* to initialize the printer;

```
2 POKE38912,Ø:IFPEEK(38919)
  AND16THENPRINT#4,CHR$(27)
  ;CHR$(64);
```

Execute a DISK!"PUT BEXEC*" and you're done.

COMMENTS AND NOTES

With either operating system, and barring any typing mistakes, you should now have a new bootable Disk that Auto-Runs a directory or menu and initializes the parallel port and printer.

Both operating systems have problems with the MCS6522 during normal program execution. That's because the 6522's registers look like memory to the CPU. Answering 32768 to the MEMORY SIZE query when booting HEXDOS or setting the top of memory pointer on Track Ø at $2217 (8823) to $97 (151) when running 65D eliminates any conflicts.

★

OS65D V3.3 BUG!

By: Paul Chidley
Courtesy of TOSIE
Toronto Ohio Scientific Idea
  Exchange
P. O. Box 29
Streetsville, Ont.
Canada L5M 2B7

The old OSI Challenger series has over the years proven to be a very reliable machine, ... when it works. But when it's not feeling well, it is flaky as flaky can be. The following bug has plagued my system for years masquerading as a hardware problem when all along it was the software. So if you have ever had this problem, then this article should be worth many times the price of your subscription, and if you haven't had this problem you had better read this article anyhow. First the symptom, booting the system with V3.2 works fine but trying to boot V3.3 is flaky, i.e., sometimes it boots and sometimes it just hangs. However, once V3.3 has booted then everything works fine with no further problem. This problem occurred on C4P with SA455 disk drives. However, I have also seen the problem on the MPI drives and although I have never seen it on an eight inch system, the software bug is there so it could happen. Especially if you are trying to run your C8/C3 at 4 MHz.

To see where the problem is, we need to know what V3.3 does while booting that is so different from V3.2. The answer is found just after $22ØØ, there is a JSR to $2E79 the I/O scratch buffer. But why jump to a scratch buffer, the first time you call a file the code would get wiped out?! The answer is that all of this code is executed before we get that far and never needed again, so the scratch buffer is a good place to put a patch. The disassembled listings of what you will find in the patch are included here. Let's get back to what happens when you boot, we will use the 5" listing for this example. When you hit "D" the monitor ROM loads track Ø to $22ØØ then jumps to $22ØØ. The code on track Ø loads track 1 then comes the JSR to the patch. The patch then loads track 6 sector 2 to $ØØØØ, then loads track 13 sector 1 to $3274, then goes into a neat piece of code that determines what clock rate your CPU is running at and adjusts the TENMS delay subroutine accordingly, it then does an RTS and continues the usual boot up. Well, did you see it? If not, read that again and think about what is happening. The first thing the disk does is load tracks Ø and track 1, it then has to step all the way out to track 6, then track 13 and THEN it adjusts the subroutine that helps determine how fast the disk tries to step. The problem is this, most C4s are set to run at 2MHz, however, the default value on track Ø is set for 1MHz. Therefore, when you boot, the TENMS delay is only half as long as it should be until the patch sets up for 2 MHz. This short delay can cause the step pulses to come faster than the drive can step so you end up getting an Error #5. Only you don't get this message on the screen because the I/O pointers haven't been set yet, instead it just looks like the machine hung.

The solution can be easy or hard. The hard but possible better fix is to rewrite the patch so that the delay subroutine is adjusted first and then the other tracks are loaded. While this is not really that hard, I decided to take an easier way out. $267B was set to $31 on track Ø as a default to 1 MHz, so all I did was change this to $62 to make my default 2MHz. However, the location at $2EB1 also needed to be changed from $31 to $62, because this value is the default used if the routine can not find your 54ØB video board. Let me see if I can explain that one, if you look at the schematics for the 54Ø board, you can find a line in the timing chain called R7. This line is tied to bit 7 and

THE DATA SYSTEM

- Stored Report Formats
- Stored Jobs, Formats, Calcs.
- Multiple Condition Reports
- Multiple File Reports
- Calc. Rules Massage Data
- Up to 100 Fields Per Record
- User Designed Entry/Edit Screens
- Powerful Editor
- Merges - Append, Overlay, Match
- Posting - Batch Input
- Nested Sorts - 6 Deep
- Abundant Utilities

HARDWARE REQUIREMENTS: 48K OSI, Hard Disk, serial system, OS-65U 1.42 or Later; Space required: 1.3 megabytes for programs and data.

PRICE: $650.00 (User Manual $35.00, credited towards TDS purchase). Michigan residents add 4% sales tax. 30 day free trial, if not satisfied, full refund upon return.

TIME & TASK PLANNER

30 DAY FREE TRIAL — IF NOT SATISFIED, FULL REFUND UPON RETURN

- "Daily Appointment Schedule"
- "Future Planning List" - sorted
- "To Do List" - by rank or date
- Work Sheets for all Aspects
- Year & Month Printed Calendar
- Transfers to Daily Schedule

A SIMPLE BUT POWERFUL TOOL FOR SUCCESS

HARDWARE: 48K OSI, 8" floppy or hard disk, serial terminal system, OS-65U v. 1.3 or later.

PRICE: $300.00 (User Manual, $25.00, credited toward TTP purchase). Michigan residents add 4% sales tax.

FINANCIAL PLANNER

- Loan/Annuity Analysis
- Annuity 'Due' Analysis
- Present/Future Value Analysis
- Sinking Fund Analysis
- Amortization Schedules
- Interest Conversions

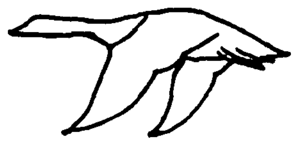
HARDWARE REQUIREMENTS: 48K OSI, 8" floppy or hard disk, serial terminal system, OS-65U v. 1.2 or later.

PRICE: $300.00 (User Manual, $25.00, credited toward Planner purchase). Michigan residents add 4% sales tax.

DEALERS: Your Inquiries Most Welcome

GANDER SOFTWARE, Ltd.

3223 Bross Road
"The Ponds"
Hastings, MI 49058
(616) 945-2821


"It Flies"

FROM THE FOLKS WHO BROUGHT YOU:
All This
THERE IS MORE COMING SOON:
Program Generator for TDS
Proposal Planner
Time and Billing A/R

can be examined by reading from location $DE00, the other bits at that location aren't used for anything. Since the frequency of R7 is uneffected by the CPU's clock frequency, it can be used as a known time base. This patch times how many loops it can do between this line shifting and, therefore, determine the CPU's real clock speed. The patch is also smart enough not to hang if the line never changes, this is what happens on my system since I don't use a 540 board, it then picks the value at $2EB1 as the default. In other words, if you don't have a 540B video board, then you must be running at 1 MHz, right? Wrong, but that's what you get for not running a stock machine. The eight inch version is different in the way that it determines the CPU speed, it uses the interrupt register in the ACIA to determine the transmitting baud rate of the ACIA. The reason that this bug is less of a problem on eight inch is that the default is already set to 2 MHz so most systems would probably never know the difference.

I know that I may not have explained everything in great detail, but I don't want to have all the the fun. Get out your disassembler and get to work, if you figure out the details on your own, you'll learn more from it. If, however, you have any questions just ask.

Note: The copy of version 3.2 that came with V3.3 as tutorial disk #2 also has a similar patch with the same problem. The simple fix is again to change the default speeds to match your system. Older versions of 3.2 do not have any such patch. My copy of WP6502 V1.3 is on a 3.2 disk, and it did have the patch, fixing it solved many of the problems I was having with it. Just make sure that track 11,1 is only one page long.

Not all releases of V3.3 are the same (or so I've been told), so if your version is not exactly the same as the examples you now know why (some of the very early versions may have been slightly different). Listed below is the TENMS delay subroutine I spoke of to get you started.

```
2678 A2 0C  TENMS LDX #$0C
267A A0 31        LDY #$31   ;LOOP COUNT FOR DELAY
267C 20 00 27     JSR DELAY  ;DO 1 MS DELAY
267F CA           DEX
2680 D0 F8        BNE TENMS+2;($267A)NOT DONE,LOOP
2682 60           RTS
```

★

THE PORTLAND BOARD

By: Bob Ankeney
5221 S.W. Corbett
Portland, OR 97201

The Portland Board is the latest in the line of boards for the OS-65U based 200-series computers from OSI. This board, manufactured by OSI as the 517 board, is the result of a collaboration between Generic Computer Products (principally involved in the hardware) and Brown/Collinson Associates (principally involved in the software).

This board is designed to drastically increase performance in systems by providing a high-speed processor for each Portland board user. In standard systems, a 2 MegaHertz processor is shared among all users.

Portland Boards consist of 2 users, each with a 4 MegaHertz processor, 64K static RAM, 4K ROM (addressable in 1K partitions), and RS232 serial interface. The board plugs directly into the OSI bus, and works with the standard memory partitions. Thus, a system can consist of both standard timeshare users and Portland Board users. The Portland Board users will see a marked improvement in response and run times, and timeshare users will also notice improvement due to the load reduction.

HOW IT WORKS

Each timeshare user has a 48K memory partition containing their particular application program. These partitions each receive a slice of time from a processor that is shared with other timeshare users. Each Portland Board user has its own processor dedicated to running its application program.

At times, all users will have need to access the OSI bus (usually for hard disk transfers or printer output). A Portland Board user has no direct access to the OSI bus, so when such transfers are needed, that user places its memory on the OSI bus and a request is made to the timeshare processor. The Portland Board user now looks much like a timeshare user, and is

Supercharge Your OSI Computer!

Isotron OSI'S new PORTLAND BOARD is designed for use in your C3 and 200 series computer. Each 2 user board contains two 4 MHz 6502 microprocessors with memory, and can be added to all level 3 timeshare systems. You keep all of your existing timeshare users and they still work as before. The PORTLAND BOARD is fully compatible with networking, and requires no software changes to your 65U programs. Now each terminal can operate twice as fast as the entire computer operated before! Requires 1.44 or later. $1600.00

Order your PORTLAND BOARD direct from the experts in Portland.

Fial Computer 5221 S.W. Corbett Ave., Portland, OR 97201 Telephone (503) 227-7083 granted a slice of time in which the transfer occurs. Once the transfer is complete, the user takes its memory back and continues processing independently.

PERFORMANCE

Due to the faster processor, each Portland Board user generally runs at least twice as fast as a timeshare user in compute-bound applications. As more timeshare users execute, the performance ratio increases further. For Portland Board users requiring a great deal of OSI bus access (e.g. hard-disk bound applications), performance can deteriorate to about that of a normal timeshare user. This is a common bottleneck in all multi-user systems.

THE HARDWARE

Memory consists of a 48K memory partition that can appear either on the OSI bus or the local Portland Board bus. The 4K block $DØØØ-DFFF contains a special copy of the normal $DØØØ portion of the V1.45 operating system. ROM appears at $FCØØ-$FFDF and $FFFØ-$FFFF. One of four 1K partitions of the ROM can be selected. Optionally, ROM can be disabled with RAM appearing in its place. RAM also appears from $EØØØ-$FBFF. Currently the memory from $CØØØ-$CFFF is not used and is available to the user. The memory from $EØØØ-$F7FF is reserved for future expansion. All I/O appears at $FFEØ-$FFEF. The RS232 serial interface is a 6551 chip that supports programmable baud rates from 5Ø-192ØØ baud. BREAK generation and other features are also supported. A 4Ø-pin expansion bus is provided for support of future products such as local printers.

THE SOFTWARE

Each Portland Board comes with a diskette to upgrade from OS-65U version 1.44 to version 1.45. Portions of the operating system (such as the scheduler) have been rewritten to improve performance. Most applications software should run without modification. Software that patches the operating system may need to be modified, as some locations have changed. Memory locations documented by OSI have remained unchanged (except for some networking locations which are listed in the documentation). Four new FLAG statements have been added. The first two can be used to force program execution by the timeshare processor. The other two determine whether memory above $CØØØ refers to the Portland Board local memory or to memory on the OSI bus. A program that changes the number of time slices for any user is also included. Thus certain users can be assigned higher priorities.

THE FIRMWARE

On power-up, a memory test is performed on each user. Errors are reported on the terminal at 96ØØ baud. Then an automatic baud rate detection routine is entered. If the user presses the RETURN key, the baud rate is detected and printed. Otherwise, the default baud rate (as determined in the program SLAVE*) is selected. An undocumented feature is a ROM monitor that can be entered by typing a Control-\ combination after auto-bauding. The commands are much the same as the standard 65A monitor, but in a far more readable format.

INSTALLATION

The Portland Board plugs directly into the OSI bus. The board size is a bit larger than most boards (8" x 11" instead of the usual 8" x 1Ø"). Cables must be connected for each RS232 serial port. A number of cabling options are available, and should be discussed with your dealer when ordering. Information is provided in the documentation for building your own cables.

ORDERING

The Portland Board is available from your local OSI dealer. The list price for a 2-user board is $16ØØ (excluding cables). For more information, contact Fial Computer (5Ø3) 227-7Ø83, your local dealer, or ISOTRON, Inc.

★

VIDEO DRIVER PATCHES FOR OS-65U

By: Danny Schwartz
75 E. 19Ø St.
Bronx, NY 1Ø468

This article will interest only a small number of users, but those who can use it will probably find it very useful. What it is, is a patch to the current version of OS-65U which will allow the Editor and Extended Input modes to function correctly with 54Ø video board-based systems. That doesn't mean you will get cursor-addressing or other special terminal functions. What it does mean is that you will be able to edit your BASIC programs using the Editor functions and to use Extended Input to get inputs of specified lengths and types. One control code is actually added to the list of those recognized by the video driver: control-L. or CHR$(12). will now be recognized as a non-destructive forward space. The CRT Ø file must be set to use this code for both the incoming-forward space and outgoing-forward-space function.

```
RUN"CHANGE","PASS

OS-65U Disk Change Utility
--------------------------
MODE: HEX(H), DEC(D) ? H
DEVice ? A
ADDRESS OFFSET ? CØØ
ADDRESS ? 3BB2
ØØØØ3BB2   9D ? 2C
ØØØØ3BB3   C7 ? .
ADDRESS ? 3BCC
ØØØØ3BCC   84 ? CØ
ØØØØ3BCD   AD ? 2Ø
ØØØØ3BCE   9C ? 36
ØØØØ3BCF ? 3F ? 3C
ØØØØ3BDØ   EA ? AD
ØØØØ3BD1   EA ? B6
ØØØØ3BD2   FØ ? 38
ØØØØ3BD3   Ø3 ? 29
ØØØØ3BD4   2Ø ? 7F
ØØØØ3BD5 6 36 ? EA
ØØØØ3BD6 < 3C ? EA
ØØØØ3BD7   AD ? EA
ØØØØ3BD8   B6 ? C9
ØØØØ3BD9 8 38 ? ØC (forward-
    space code - must match
    "CRT Ø" file)
ØØØØ3BDA ) 29 ? FØ
ØØØØ3BDB   7F ? 13
ØØØØ3BDC   C9 ? .
```

Listing continued on page 12

computer repair

Board level service on:
- OSI / Isotron
- TeleVideo
- IBM pc/xt

Floppy drive alignment:
- Siemens
- Shugart
- Teac

Terminal repair:
- TeleVideo
- Micro-Term

(1 week turnaround)

Sei
Sokol Electronics Inc.
474 N. Potomac St.
Hagerstown, Md. 21740
(301) 791-2562

```
ADDRESS ? 3BE7
ØØØØ3BE7   16 ? ØB
ØØØØ3BE8   C9 ? /
ØØØØ3BE9   2Ø ? /
ØØØØ3BEA Ø 3Ø ? /
ØØØØ3BEB   13 ? ØE
ØØØØ3BEC   9D ? /
ØØØØ3BED   EØ ? ØØ
ØØØØ3BEE   D2 ? D6
ØØØØ3BEF   E8 ? /
ØØØØ3BFØ   EØ ? FØ
ØØØØ3BF1   9B ? 15
ØØØØ3BF2   FØ ? 24
ØØØØ3BF3   13 ? CA
ØØØØ3BF4   A9 ? .
ADDRESS ? 3C11
ØØØØ3C11   D3 ? D7
ØØØØ3C12   E8 ? .
ADDRESS ? 3C1F
ØØØØ3C1F   AØ ? CØ
ØØØØ3C2Ø   DØ ? .
ADDRESS ? 3C28
ØØØØ3C28   D2 ? D7
ØØØØ3C29   9Ø ? .
ADDRESS ? 3C31
ØØØØ3C31   84 ? CØ
ØØØØ3C32 L 4C ? .
ADDRESS ? 3C3A
ØØØØ3C3A   EØ ? ØØ
ØØØØ3C3B   D2 ? D6
ØØØØ3C3C H 48 ? .
ADDRESS ? 3C41
ØØØØ3C41   EØ ? ØØ
ØØØØ3C42   D2 ? D6
ØØØØ3C43 h 68 ? .
ADDRESS ? 3C64
ØØØØ3C64   FF ? ØØ
ØØØØ3C65   8D ? /
ØØØØ3C66   ØØ ? 9E
ØØØØ3C67   DF ? 3A
ØØØØ3C68   AD ? /
ØØØØ3C69   ØØ ? /
ØØØØ3C6A   DF ? /
ØØØØ3C6B J 4A ? 8A
ØØØØ3C6C   DØ ? X
OK
```

★

BEGINNER'S CORNER

By: L. Z. Jankowski
Otaio Rd 1, Timaru
New Zealand

SEQUENTIAL FILE SORT

OS65D 3.3 users are supplied with a file sort utility on one of the tutorial disks. The program's name is GSOSRT and the sorting algorithm used is a good one, an insertion sort. Unfortunately, the speed of the sort is limited because the shell algorithm has not been incorporated. As it stands GSOSRT is excellent if there are only a few out of order elements in the list about to be sorted. If the elements of the list are badly out of order then the sort takes much longer to complete. The shell algorithm sort can be up to 10 times faster than the ordinary sort.

SEQUENTIAL FILE SORT

```
10 REM Sequential File Sort. (c) LZ Jankowski 1985
20 REM All Rights Reserved 1985. Version June 13th '85
30 :
40 POKE2972,13:POKE2976,13:POKE2888,0:POKE8722,0:POKE2073,173
50 CLEAR:PRINT!(21)!(25):Y=0:I=0:B=1:Z=0:C=0:Q=0:K=0:R=0:P=10:H=0:T=0
60 L=0:N$="":F$="------":F1$="------":E$="No disk errors":ER$=E$
70 Y$=CHR$(135):FORC=1TO4:U1$=U1$+Y$:NEXT:U2$=U1$+Y$+Y$:U$=U1$+U2$
80 Y$=CHR$(148):FORC=1TO21:U3$=U3$+Y$:NEXT:L$=CHR$(8):L1$=L$+L$+L$
90 A$="Ascending/Descending Order (a/d) ? a":R$="Ready ? ":L3$=L$+L1$
100 L2$=L3$+L3$:C$=CHR$(27)+CHR$(28):G=13026:T2=32:CU=128
110 :
120 S$(1)="1> Load a file":S$(2)="2> Save file":S$(3)="3> View file"
130 S$(4)="4> Sort file":S$(5)="5> Run Seq. File Ed.":S$(6)="6> End"
140 S$(7)="7> Reset":SR$="SEQFED":X=200:N=X:ZZ=0:POKE49154,0:GOTO180
150 :
160 GOSUB5370
170 REM ---------------- MAIN MEMU ---------------------------
180 PRINTC$
190 T=22:Y=1:PRINT&(T,Y)"---------------------":POKEG,T2
200 PRINT&(T,Y+1)"!    SEQUENTIAL    !"&(T,Y+2)"!    "U$"    !"
210 PRINT&(T,Y+3)"!  DATA FILE SORT  !"&(T,Y+4)"---------------------"
220 T=7:Y=18:PRINT&(T,8)U3$&(T+29,8)U3$&(T,Y)U3$&(T+29,Y)U3$:Y=9
230 PRINT&(T+8,Y)"MENU"&(T+36,Y)"Status"&(T+8,Y+1)U1$&(T+36,Y+1)U2$
240 H=10:FORC=1TO7:PRINT&(T+1,H+C)S$(C):NEXT:T=37
250 PRINT&(T,H+1)F$&(T,H+2)F1$&(T-1,H+4)L"records. "P"fields"
260 PRINT&(T-1,H+3)N-L"records free"&(T,H+6)ER$&(7,H+4+C)"Choice ";
270 POKEG,63:GOSUB5400:POKEG,CU:ER$=E$:IFY=0ORY>7THEN290
280 TRAP7000:GOSUB1060:ONYGOSUB320,530,650,790,1030,5420,1040
290 GOTO180
300 :
310 REM ------------- LOAD A FILE OF DATA -------------------
320 Y=13:IFLTHEN500
330 PRINT&(Y,10)"LOAD. File name"SPC(5)F$L2$;:INPUTY$:IFY$="x"THEN500
340 IFY$<>""THENF$=Y$
350 PRINT&(Y,12)"Max. # of Records   "N;L1$L1$;:INPUTY$
360 N=VAL(Y$):IFN<1THENN=X:IFY$="x"THEN500
370 PRINT&(Y-8,14)"If MDMS Key File, type # of key field   0"L1$;
380 INPUTY$:K=VAL(Y$):IFY$="x"ORK<0THEN500
390 IFKTHENF$=F$+CHR$(K):P=2:Y=5:GOTO440
400 PRINT&(Y,16)"Fields per record   "P;L3$L$;:INPUTY$:IFY$="x"THEN500
410 K=VAL(Y$):IFK>0THENP=K
420 PRINT&(Y-8,18)"When sorting ignore first:   0  fields "L2$L3$;
430 INPUTY$:Y=VAL(Y$):IFY$="x"ORY<0THEN500
440 IG=Y:IFFL=0THEN:DIMG$(IG),D$(N,P),N%(N):FL=-1
450 PRINTC$&(16,10)"* LOADing file from disk *"&(0,0):C=0
460 GOSUB5370:DISK OPEN,6,F$:TRAP8000
470 IFIGTHEN:FORC=1TOIG:INPUT#6,G$(C):NEXTC
480 L=L+1:FORK=LTON:FORC=1TOP:INPUT#6,D$(K,C):NEXTC,K
490 DISK CLOSE,6:K=K-1:GOSUB5370:GOSUB9000
500 RETURN
510 :
520 REM ------------ SAVE A FILE of DATA -------------------
530 IFL=0THEN620
540 PRINT"SAVE. File name   "F1$L2$;:INPUTY$:IFY$="x"THEN620
550 IFY$<>""THENF1$=Y$
560 PRINT&(10,12)A$L$;:GOSUB5400:IFY$="x"THEN620
570 Y=1:I=L:Q=1:IFY$="d"THENY=L:I=1:Q=-1
580 PRINTC$&(16,10)"* SAVEing file to disk *"&(0,0):GOSUB5370
590 DISK OPEN,6,F1$:IFIGTHEN:FORC=1TOIG:PRINT#6,G$(C):NEXTC
600 FORK=YTOISTEPQ:FORC=1TOP:PRINT#6,D$(N%(K),C):NEXTC,K
610 DISK CLOSE,6:GOSUB5370
620 RETURN
630 :
640 REM -------------- VIEW FIELDS -------------------------
650 R=1:F=2:IFR>LTHEN760
660 PRINT&(5,10)A$L$;:GOSUB5400:IFY$="x"THEN760
670 Z=R:I=L:Q=F:T=1:IFY$="d"THENZ=L:I=R:Q=-F:T=-1
680 PRINT&(0,2)!(22,62,20)!(24)
690 IFIGTHEN:FORC=1TOIG:PRINTG$(C):NEXTC:PRINT:PRINTR$;:GOSUB5400
700 FORK=ZTOISTEPQ:PRINT&(0,0)!(24):POKEG,T2
710 FORY=1TOP:PRINTKTAB(10)D$(N%(K),Y):NEXTY:IFK=ITHEN750
720 FORY=1TOP:PRINTK+TTAB(10)D$(N%(K+T),Y):NEXTY
730 POKEG,CU:GOSUB5400:IFY$="x"THENK=I:NEXTK:GOTO760
740 H=10*Y*SGN(Q):IFYTHENK=K+H-Q:IFK>LORK<1THENK=K-H+Q
750 NEXTK:POKEG,CU:PRINT:PRINTR$;:GOSUB5400
760 PRINT!(21):RETURN
770 :
780 REM ------------------- SORT -----------------------------
790 IFL=0THEN1000
800 PRINT"SORT. From record #   1"L1$;:INPUTY$:IFY$="x"THEN1000
810 Y=VAL(Y$):H=1:IFY>0THENH=Y
820 PRINT&(22,11)"To record   #  "L;L3$L$;:INPUTY$:IFY$="x"THEN1000
830 Y=VAL(Y$):T=L:IFY>HTHENT=Y
840 PRINT&(22,12)!(15)"On field    #  "B;L3$;:INPUTY$:IFY$="x"THEN1000
850 Y=VAL(Y$):IFY<0ORY>PTHEN840
860 IFYTHENB=Y
870 PRINT&(5,14)"Is the File partially Sorted on Field #"B"? y"L$;
880 GOSUB5400:IFY$="x"THEN1000
890 PRINTC$&(16,10)"* SORTING *":IFY$="n"ANDT-H+1>4THEN940
900 FORQ=H+1TOT:PRINT"Sorting ";:Y=Q:I=N%(Y)
910 IFD$(N%(Y-1),B)<=D$(I,B)THEN930
920 N%(Y)=N%(Y-1):Y=Y-1:IFY>HTHEN910
930 N%(Y)=I:NEXTQ:GOTO1000
940 I=(2^INT(LOG(T-H+1)/LOG(2)))-1
950 I=INT(I/2):PRINT"Still Sorting":IFI<1THEN1000
960 FORQ=HTOI+H-1:R=Q+I:FORC=RTOTSTEPI:Y=C:Z=N%(Y)
970 IFD$(N%(Y-I),B)<=D$(Z,B)THEN990
980 N%(Y)=N%(Y-I):Y=Y-I:IFY>I+H-1THEN970
990 N%(Y)=Z:NEXTC,Q:GOTO950
1000 RETURN
1010 :
1020 REM ------------- SUBROUTINES --------------------------
1030 PRINTC$:GOSUB5370:RUNSR$:RETURN
```

Continued on next page

ADVANTAGES

The program presented here has several advantages. When a file is loaded, the arrays in which it is stored can be declared exactly (line 44Ø), thereby maximizing the availability of free RAM. The more RAM available for the program to work in, the less chance there is of a time-consuming garbage collect being called.

The file can be saved and viewed either in ascending or descending order. See lines 57Ø & 6ØØ for save, and lines 67Ø & 7ØØ for view. When viewing records it is possible to move through them in steps of ten. For example, pressing "6" will advance viewing by 6Ø records - see line 74Ø.

SORTING

Sorting can be selective. For example, sort from record 41 to 53 inclusive. Two sorting algorithms are used. The first (lines 9ØØ-93Ø) is for those files whose records are nearly sorted and the second (lines 94Ø-1ØØØ) is for those files whose records are in random order. During either sort the records are not swapped, only the pointers to those records. The pointers are placed in array N% in line 9Ø2Ø. The advantage of sorting a pointer array is a speed increase of about 15 times (for 8 field records).

MDMS

MDMS key files can be loaded (line 37Ø), sorted and saved back to disk. But not MDMS master files; they are random files and not catered for.

DATA FIELDS

Many programs make the first few fields of a file one-off data fields which must not be sorted. In the program this is taken care of by placing such fields in array G$ - see line 47Ø. These data fields are always saved back to disk in front of the sorted file - see line 59Ø.

THE PROGRAM

In line 4Ø are the usual POKEs to allow data separators to be accepted by INPUT, for <RETURN> to return a null or zero after INPUT, and for CTRL-C to be disabled. To disable CTRL=C change the "173" on the end of line 4Ø to a "96". In line 5Ø "PRINT!(21) !(25)" homes the cursor and sets screen colors to b/w. In line 8Ø is "L$=CHR$(8)", a most valuable code that, when printed, moves the cursor one space to the left!

The program is menu driven and the status of each menu option is shown on the right hand side of the screen. A default input is offered on every INPUT. The maximum number of records allowed to be read off disk is controlled by the value of "X" - see the middle of line 14Ø. If stop/start of the disk drive is not required then remove the POKE in line 14Ø and place a REM at the beginning of line 537Ø.

But what's this in line 44Ø: "IF FL=Ø THEN:" ? This is correct; if FL=Ø then BASIC continues along the line and DIMensions the arrays.

POINTERS

A list of house numbers for a particular road, written down on a slip of paper, is an example of a pointer system. There is a unique correspondence between every house and its number, (one-to-one correspondence). The system works well because the houses are not numbered at random! Because the numbering is ordered, a house with a missing number can be easily identified by referring to the numbers of the houses on either side of it.

Imagine a list (one dimension array) named D$ that contains a random choice of letters of the alphabet. Another list, named P%, records the position of each letter in list D$. If D$(1) contains a "t" and P%(1) contains a "1" then at the moment P%(1) points to the letter "t" - "t" is produced by D$(P%(1)). Changing the contents of P%(1) from "1" to say "5" will change what P%(1) is pointing to. If now D$(5)="a" then D$(P%(1)) will produce the "a". (Remember, P%(1)=5).

Anyway, a program is worth a thousand words so run Listing 2 to see what I mean.

```
1040 PRINT"Reset <-- Sure ? n"L$;:GOSUB5400:IFY$="y"THENRUN
1050 RETURN
1060 PRINTC$SPC(11)"To exit, type an 'x'. Press <RETURN>.":Y$="-"
1070 FORC=1TO64:PRINTY$;:NEXTC:PRINT:PRINT&(16,10);:RETURN
5370 ZZ=NOT(ZZOR254):POKE49154,ZZ*-1:FORW9=1TO1200:NEXTW9:PRINT
5380 RETURN
5400 DISK!"GO 2336":Y$=CHR$(PEEK(9059)OR32):Y=VAL(Y$):RETURN
5420 PRINT!(28)&(16,12)"To RESTART type:-  GOTO 160"
5430 POKE13026,128:PRINT&(16,14)"Bye !":POKE2073,173:GOSUB5370:END
7000 PRINT&(16,10)!(15):ER$="DISK ERROR":GOSUB5370:GOTO190
8000 GOSUB5370:GOSUB9000:ER$="ERR #D. Don't worry!":GOTO180
9000 PRINTC$&(16,10)"* Editing. Please wait! *"
9010 FORQ=KTO1STEP-1:FORC=1TOP:IFD$(Q,C)<>N$THENL=Q:C=P:Q=1
9020 NEXTC,Q:FORC=1TOL:N%(C)=C:NEXTC:RETURN
```

POINTER DEMO

```
10 REM LISTING 2. POINTER DEMO (c) LZJ 6/1985
20 :
30 E=5: DIM D$(E), P%(E)
40 PRINT !(28)"Type" E "alphabetical characters.": PRINT
50 FOR C=1 TO E: PRINT C"is ";: GOSUB 5400: PRINT Y$
60 D$(C)=Y$: NEXT
70 FOR C=1 TO E: P%(C)=C: NEXT
80 PRINT !(28);: X=1: Y=19: GOSUB 500: PRINT: PRINT
90 PRINT TAB(14)">>> Sorting Pointer Array <<<": PRINT: PRINT
100 FOR Q=2 TO E: Y=Q: I=P%(Y)
110 IF D$(P%(Y-1))<=D$(I) THEN 130
120 P%(Y)=P%(Y-1): Y=Y-1: IF Y>1 THEN 110
130 P%(Y)=I: NEXT
140 X=13: Y=19: GOSUB 500
400 END
490 :
500 PRINT "Data array." TAB(Y-2) "Pointer array.": PRINT
510 FOR C=1 TO E: PRINT "D$(" C ")=" D$(C) TAB(Y-3) P%(C);
520 PRINT TAB(Y)"is in P%(" C ").....D$( P%(" C ") )=";
530 PRINT D$(P%(C)): NEXT C: RETURN
5390 :
5400 DISK !"GO 2336": Y$=CHR$(PEEK(9059) OR 32): RETURN
```

★

DABUG EDITING AND CTRL B FOR THE ASSEMBLER

PART II

By: John Whitehead
(with help from David Dodds)
17 Frudal Crescent
Knoxfield 3180
Australia

Below is the source to enable the tape version of the Assembler/Editor to be used with a Superboard series 2 in 24x24 or 48x12 with on screen editing.

To use the source below, load the Assembler, change the source file workspace start at $12C9 and $12CA to start at $1440 as page 26 of the manual. Type in the source. Assemble it using A and if there are no errors, save it on tape. Then use A3 to put the code into memory. Press the BREAK key, M 1300 G, then answer INIZ? with N. Now try out all the DABUG control keys.

The line feed key is now TAB and E takes you to the monitor. This can be changed in line 630 to go to EXMON in EPROM.

```
 10    * = $1300
 20         JMP DABUG
 30     * = $1315
 40         .BYT $0A ;line feed
 50     * = $1311
 60         JSR INPUT
 70     * = $1333
 80         JMP OUTPUT
 90     * = $1391
100  INPUT JSR PUSH
110         LDA #00
120         STA $0E
130         JSR $FFEB
140         PHA
150         LDA $0E
160         BEQ NOBACK
170 SHIFTO  DEC $3F
180         INC $0E
190         BMI SHIFTO
200 NOBACK  PLA
210         TAX
220         JMP PULL
230 OUTPUT  JSR PUSH
240         JSR $FFEE
250         JMP PULL
260 PUSH    PHA
270         TXA
280         PHA
290         LDX #$E0
300 M1      LDA $00,X
310         STA SPACE-$E0,X
320         LDA SPACE+$20-$E0,X
330         STA $00,X
340         INX
350         CPX #$00
360         BNE M1
370         PLA
380         TAX
390         PLA
400         RTS
410 PULL    PHA
420         TXA
430         PHA
440         LDX #$E0
450 M2      LDA $00,X
460         STA SPACE+$20-$E0,X
470         LDA SPACE-$E0,X
480         STA $00,X
490         INX
500         CPX #$00
510         BNE M2
520         PLA
530         TAX
540         PLA
550         RTS
560 DABUG   JSR $FAA1 ;switch to
                              DABUG
570         JSR PULL
580         JMP $1160
590 INOFF   LDA #$BA ;DABUG input
                              off
600         STA $0218
610         LDA #$FF
620         STA $0219
630         JMP $FE00
640 SPACE
650 * = $0670
660         JMP INOFF
670 .END
```

WAZZAT CORNER!

By: L. Z. Jankowski
Otaio Rd 1, Timaru
New Zealand

One of the great things about OS65D 3.3 is the line editing. And by that I don't mean its mechanics! The line editing commands that are available are particularly useful when typing in a line of BASIC, or when answering the INPUT prompt. The ability to retreat back up the typed line to make insertions and deletions is extremely useful - particularly when only one character is involved!

A good program will offer a default input on an INPUT prompt. (See the Seq. File Editor listing published last month, and in this issue). For example, the words "SAVE. File" appear on the screen with the cursor on the "F" in "File". Two choices are available. Either press <RETURN> and the default file-name "File" is accepted, or some other file name could be typed. Subsequent program code can discover which choice was made by checking for a null input thus, "IF Y$="" THEN ...".

This idea works well for a short input: there is not much to retype if a change has to be made. For long inputs, line editing is very desirable. A good example of this need is in this month's listing of the "EDIT" section of the Seq. File Editor. "EDIT" displays on the screen each sequential file entry, and pressing <RETURN> leaves an entry unchanged. If a change is required then the whole line has to be retyped, even if the change is one character at the start of the entry!

What is wanted is for the entry being edited on screen to be simultaneously placed in BASIC's input buffer. Input could then be made either immediately on the <RETURN> key being pressed, or after editing. The editing would be a breeze with CTRL-B, F, H, I and P.

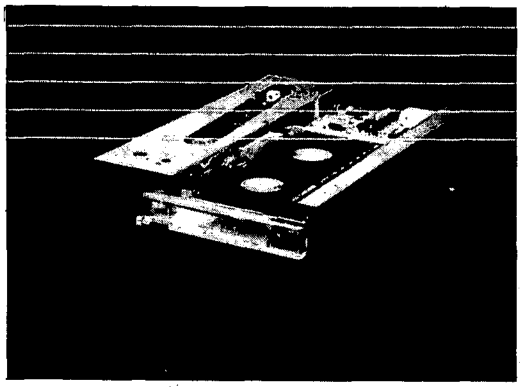
How this works is shown by the "INPUT with EDIT" program - see listing. The string to be input and edited, S$(1), is copied to the indirect file. This file can be set to begin anywhere in RAM with the "MEM" command. The preferred location is the directory buffer which begins at $2E79. The directory buffer is a block of 256 bytes and free for program use. Input to BASIC is made from this indirect file. The string is copied from the indirect file to the screen and the BASIC buffer, just as if it had been typed in from the keyboard! The cursor is placed at the beginning of the input line by the CHR$(6) written after string S$(1).

The main problem in making the program work is with the "]" characters that DOS insists on printing. The second "]" is printed to screen but not written to the buffer. This is taken care of in line 70. The first "]" is always printed to screen and BASIC buffer

```
10 PRINT!(28)&(0,10);                 :REM INPUT with EDIT by LZJ
20 :
30 S$(1)="This sentence has been copied to the BASIC buffer."
40 N$=CHR$(0)                         :REM Null !
50 F$=CHR$(6)                         :REM CTRL-F
60 :
70 POKE 9571,0                        :REM Null instead of ] !
80 C=9574: Y=234                      :REM C=$2566, Y=$EA
90 POKE C,Y: POKE C+1,Y: POKE C+2,Y :REM Stop print of 2nd ]
100 :
110 DISK!"MEM D100,D100"              :REM or $2E79
120 DISK!"IO ,10"                     :REM Open for Output to DV 5
130 PRINTS$(1) S$(1) F$ N$;               :REM Write to DV 5
140 DISK!"IO 10,02": PRINT            :REM Open for Input from DV 5
150 INPUT"INPUT "; S$(1)              :REM Input from DV 5
160 DISK!"IO 02,02"                   :REM Restore normal I/O
170 :
180 PRINT&(0,15)"S$(1) IS: " S$(1)        :REM Print edited string
190 :
200 POKE C,32: POKE C+1,70: POKE C+2,35 :REM Restore, line 90
210 POKE 9571,93                          :REM Restore ]
```

DBi, inc.

p.o. box 21146 ● denver, co 80221
phone (303) 428-0222



Wangtek sets the industry's standard for excellence in 1/4-inch streamer technology because its tape drives are all created with an uncompromising dedication to the highest possible quality in design, engineering and manufacturing. These factors combine to give the Wangtek 5000E tape drive a level of performance and reliability that is unexcelled in today's marketplace.

The Wangtek 5000E is uniquely suited to meet the backup demands of today's smaller size, higher capacity Winchester-based computer systems—it packs up to 60 MBytes of data storage in a compact, half-high form factor only 1.625 inches tall. For added user convenience, the drive accepts and automatically adjusts gains for either standard 45 MByte tape cartridges (450-foot cartridge) or high-capacity 60 MByte cartridges (600-foot cartridge).

WHAT'S NEW AT D.B.I. ???

What's the answer? The DMA 360 removable 5¼" Winchester. It's exactly the same size as a 5¼" half-height floppy drive—but that's where the similarity stops.

The DMA 360 gives you hard-disk reliability. Floppies don't.

The DMA 360 protects your data in a totally sealed cartridge. Floppies don't.

The DMA 360 packs 13 megabytes (10 formatted) on a single ANSI-standard cartridge. It takes up to 30 floppy disks to achieve an equal capacity.

The DMA 360 even has a lower cost-per-megabyte than a floppy. But it gives you so much more.

Like an average access time of 98 milliseconds. A transfer rate of 625 kilobytes per second. And an error rate on par with the most reliable conventional Winchester disk drives.

DMA Systems half-height removable 5¼" Winchester.

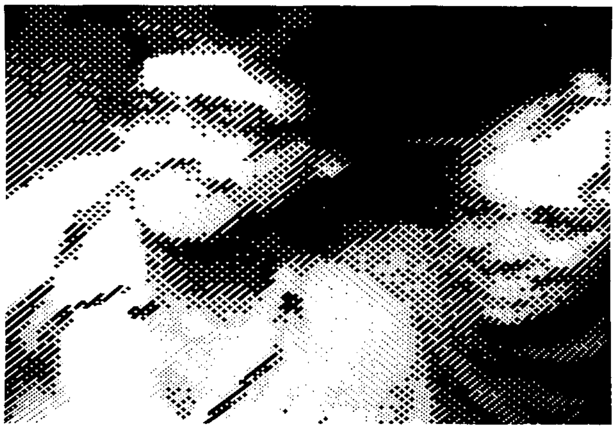
FOR PRICING AND DELIVERY CONTACT YOUR NEAREST D.B.I. DEALER!!!

*WANGTEK 5000E is a registered trademark of WANGTEK CORPORATION
*DMA 360 is a registered trademark of DMA SYSTEMS

(!), and then checked for its existence by DOS. Its appearance tells DOS to stop copying from the indirect file. DOS loads this end-of-file character into the accumulator at address 9570 and 9571. The solution is to make the end-of-file character a non-printing one, and the null character will do.

It works a treat! Why does DOS force the printing of an end-of-file character? Dumb!

★



SLOW SCAN TELEVISION PICTURES ON OSI

By: Earl Morris
3200 Washington St.
Midland, MI 68640

I have had a long standing interest in high resolution graphics for my OSI computer. The ultimate test of such graphics is to display a picture or photograph. The hardware I have displays 256 by 256 bits on the monitor screen. Compared to some commercial systems with 1024 by 1024 bits, this is only medium resolution.

Before a photo can be displayed, some sort of camera must be used to capture the image and store it into memory. Unfortunately a high resolution digital camera is out of my price range. Byte magazine in September 1983 described a less expensive camera based on a light sensitive memory chip. There are, however, photos flying around through the air free for the taking in the form of amateur radio slow scan television.

A normal TV picture is composed of 60 frames per second with 262.5 scan lines per frame. The horizontal scan rate is thus 15750 lines per second. Assuming a resolution of 400 dots per line, the bit rate is above 5 megacycles. This amount of information cannot be sent over an audio channel. The amateur radio operators have developed a means of slowing down the scan rate to allow photographs to be sent over radio, phone lines, tape recorders or any other normal audio channel. Eight seconds are required to send a complete frame. This becomes more like a series of slides than a normal TV picture. The video information is converted to audio tones of varying pitch according to the brightness level of each part of the picture. Black is 1500 cps, white is 2300 cps and the sync pulse is 1200 cps. The various shades of gray are represented by tones between 1500 and 2300 cps.

I recorded a number of pictures from the output of a friend's ham radio onto cassette tapes. These gave a test signal to play with at very little cost. My circuit converts the audio tones to a DC voltage (V/F conversion). This voltage is then fed to the computer's A/D input. By sampling this input with the proper timing loops, the photo is read into the computer. The major problem then becomes how to display an analog photo on a digital display.

The commercial slow scan receivers will display the proper gray levels directly. My computer graphics display either turns a pixel on or off - no intermediate gray levels are allowed. If the input video level varies from 0 (black) to 100 (white) a trigger at 50 can be set to switch on a pixel. This will give a very high contrast image which is really useless. A section of picture with gray level 49 will be jet black, while a very slightly brighter gray level of 51 will be full white. Shading can be introduced by a process called "dithering." A small amount of random signal or a sine wave is added to the video signal. Thus a gray level of 49 has a good chance of being over the trigger level of 50 when the "noise" is added. A gray level of 25 has a very slim chance of making it over the trigger level. In the other direction, a gray level of 51 can easily fall below the trigger when the dithering signal is added. However, a gray level of 80 will very seldom fall below the 50 trigger. Thus, dithering gives a digital picture shading by smearing out the trigger level.

A second way of simulating gray levels is to group the pixels together. Thus, if a group of four dots or pixels on the screen is considered one big dot, then the intensity of this large dot can be varied by turning on 0,1,2,3 or 4 smaller dots. From a long distance away, the eye perceives the number of dots turned on as a shading or gray level. Of course, grouping the dot by fours lowers the resolution of the picture.

The enclosed photo shows the best results I can get using a 256 x 256 graphics display by dithering and grouping. As with most computer generated photos, the further away you are, the more the individual dots blend to give shading and detail. Grouping the dots by four gives jet black, stark white and three grays. Looking at the photo close up, the various dot patterns making up the shading can be seen. This dot density is just not enough to yield a high quality picture.

The next step would be to either add, via hardware, levels of gray to the 256 x 256 board, or go to higher resolution. Doubling the resolution in each direction would yield 512 by 512. This gives four dots in place of each present dot, but requires 32K of RAM to display a black & white image. Is anyone interested in helping design such a board, or interested in purchasing a board if we ever get one made?

★

MORE SORTED ALGORITHMS

By: Roger Clegg
Data Product Maintenance Corp.
9460 Telstar, El Monte, CA 91731

The only algorithms I've been interested in are sorts, and I have long since chosen the best for my SORTER program (see last month) but here is a collection of commented sorting algorithms called SORTS.

```
1 GOTO 200:"*****************   S O R T S   ****************************
2
3   Collection of sorting algorithms.
4   The comments regard element 1 as the bottom and N as the top.
5
10 REM   INSERTION
11
12 FOR J=1 TO N-1           :REM J is highest number in sorted section
13   K=A(J+1)               :REM Value of next unsorted number
14   FOR I=J TO 1 STEP -1   :REM Compare downward through sorted section
15     IF A(I)>K THEN A(I+1)=A(I): NEXT I :REM Move up if larger
16   A(I+1)=K               :REM Insert in proper place
17 NEXT J
18 RETURN
19
20 REM   SHELL
21
22 G=N                      :REM Initialize gap between compared numbers
23 G=INT(G/2)               :REM Halve gap on each outer loop
24   K=0                    :REM Initialize flag for no swaps made
25     FOR I=1 TO N-G       :REM I is lower of two numbers compared
26       IF A(I)>A(I+G) THEN NULL A(I);A(I+G): K=1 :REM Reset flag
27     NEXT I
28   IF K=1 THEN 24         :REM If any swap was made, repeat gap level
29 IF G>1 THEN 23           :REM Next gap level.  G=1 is the bubble sort
30 RETURN
31
40 REM   SHELL-METZNER
41
42 G=N                      :REM Initialize gap between compared numbers
43 G=INT(G/2)               :REM Halve gap on each outer loop
44   FOR J=1 TO N-G         :REM J is highest number in sorted section
45     K=A(J+G)             :REM Value of next unsorted number
46     FOR I=J TO 1 STEP -G :REM Compare downward through sorted section
47       IF A(I)>K THEN A(I+G)=A(I): NEXT I :REM Move up if larger
48     A(I+G)=K             :REM Insert in proper place
49   NEXT J
50 IF G>1 THEN 43           :REM Next gap.  G=1 is the insertion sort.
51 RETURN
52
60 REM   SHELL-DPM
61
62 G=N
63 G=2*INT(G/7)+1           :REM This modification is 20% faster
64   FOR J=1 TO N-G
65     K=A(J+G)
66     FOR I=J TO 1 STEP -G
67       IF A(I)>K THEN A(I+G)=A(I): NEXT I
68     A(I+G)=K
69   NEXT J
70 IF G>1 THEN 63
71 RETURN
72
80 REM   HEAPSORT
81
82 REM  An array forms a heap when every parent element P is greater
83 REM  (or less in other algorithms) than its children 2*P and 2*P+1.
84
85 S=N                      :REM S = size of heap
86 FOR I=INT(N/2) TO 1 STEP -1 :REM Do siftup on lower half to form heap
87   K=A(I)                 :REM Value to be sifted up
88   GOSUB 100
89 NEXT I
90 I=1
91 FOR S=N-1 TO 1 STEP -1   :REM In second stage, move bottom (now
92   K=A(S+1)               :REM largest) element to top, decrement
93   A(S+1)=A(1)            :REM size of heap, and sift up the old top
94   GOSUB 100              :REM value K, starting at the bottom.
95 NEXT S                   :REM Repeat until heap is gone.
96 RETURN
97
```

Continued on next page

```
98 REM Subroutine sifts up value K, starting at element I
99
100 P=I                      :REM Initialize parent (to 1 in 2nd stage)
101   C=P+P                  :REM First of possible two children
102   IF C>S THEN 105        :REM If past end of heap then exit
103   IF C<S THEN IF A(C+1)>A(C) THEN C=C+1   :REM Set C = larger child
104   IF A(C)>K THEN A(P)=A(C): P=C: GOTO 101 :REM Move down if larger
105 A(P)=K                   :REM Insert in proper place for heap
106 RETURN
107
108
120 REM   RECURSIVE QUICKSORT  (Meaning that it calls itself)
121
122 REM   Before the first call, set L=1: U=N: S=0: DIM L(20),U(20).
123 REM   On the 6502, recursion sometimes causes stack overflow.
124
125 IF L>=U THEN RETURN     :REM If lower bound >= upper, no sort needed
126 NULL A(U);A((L+U)/2)    :REM Choose middle as pivot, move to top
127 K=A(U)                  :REM Value of pivot element
128 I=L-1                   :REM Initialize I, top index of section <= K
129 FOR J=L TO U            :REM Compare each number to pivot value
130   IF A(J)<=K THEN I=I+1: NULL A(I);A(J) :REM Move down if < or =
131 NEXT J                  :REM Final J moves the pivot to I in middle
132                         :REM The pivot is now in its final position
133 S=S+1: L(S)=I+1: U(S)=U :REM Save upper half on stack
134 U=I-1                   :REM Reset upper bound to top of lower half
135 GOSUB 120               :REM Sort lower half
136 L=L(S): U=U(S): S=S-1   :REM Retrieve upper half from stack
137 GOSUB 120               :REM Sort upper half
138 RETURN
139
140 REM   NON-RECURSIVE 1-INDEX QUICKSORT (Fastest Basic sort known)
141
142 L=1: U=N: S=0: DIM L(20),U(20)
143 NULL A(U);A((L+U)/2)    :REM Choose middle as pivot in case the
144 K=A(U)                  :REM array is partially sorted.  If it is
145 I=L-1                   :REM random, the first swap in all these
146 FOR J=L TO U            :REM Quicksorts can be omitted.
147   IF A(J)<=K THEN I=I+1: NULL A(I);A(J)
148 NEXT J
149 IF I+1<U THEN S=S+1: L(S)=I+1: U(S)=U     :REM Save if necessary
150 IF L<I-1 THEN U=I-1: GOTO 143             :REM Sort if necessary
151 IF S THEN L=L(S): U=U(S): S=S-1: GOTO 143 :REM Check stack for more
152 RETURN
153
160 REM   NON-RECURSIVE 2-INDEX QUICKSORT
161
162 L=1: U=N: S=0: DIM L(20),U(20)
163 NULL A(L);A((L+U)/2)    :REM Choose middle as pivot, move to bottom
164 I=L: J=U                :REM Initialize two approaching indexes
165 K=0                     :REM K=0 when pivot is at I, K=1 when at J
166   IF A(I)>A(J) THEN NULL A(I);A(J): K=1-K :REM Change K after swap
167   I=I+K: J=J+K-1        :REM Move one index towards the other
168   IF I<J THEN 166       :REM Repeat until indexes meet at pivot
169 IF I+1<U THEN S=S+1: L(S)=I+1: U(S)=U
170 IF L<I-1 THEN U=I-1: GOTO 163
171 IF S THEN L=L(S): U=U(S): S=S-1: GOTO 163
172 RETURN
173
180 REM   NON-RECURSIVE QUICKSORT WITH ONE INDEX AND GAP
181
182 L=1: U=N: S=0: DIM L(20),U(20)
183 NULL A(L);A((L+U)/2)
184 I=L                     :REM Same as above except that the upper
185 K=0                     :REM index is expressed as (I+G) in order
186 FOR G=U-L TO 1 STEP -1  :REM to replace the GOTO by a faster loop.
187   IF A(I)>A(I+G) THEN NULL A(I);A(I+G): K=1-K
188   I=I+K
189 NEXT G
190 IF I+1<U THEN S=S+1: L(S)=I+1: U(S)=U
191 IF L<I-1 THEN U=I-1: GOTO 183
192 IF S THEN L=L(S): U=U(S): S=S-1: GOTO 183
193 RETURN
194
200 REM   DEMONSTRATION
201
202 I=0: J=0: K=0: G=0: T=0: L=0: U=0: S=0: N=1000
203 DIM A(N): FOR I=1 TO N: A(I)=N*RND(1): NEXT: PRINT CHR$(7)
204 GOSUB 60: REM or whichever sort desired
205 PRINT CHR$(7): FOR I=1 TO N: PRINT A(I): NEXT
206 END
```

★

A CEGMON LINKER FOR 65D

By: Olof Swembel
Lundagardsv. 35
S-163 53 Spanga
Sweden

The only version of a "wedding" between 65D V3.0-2 and the CEGMON/C1E Monitor that I've seen published in the USA is the one by Dave Pompea in the Aardvark N.L., Nov. '81. I've used it to convert the C4P DOS to run on a C1P and for screen editing under OSIO's V5.14. It slows down displays to unadjusted V3.3 speed, but doesn't take up any extra RAM space and is very reliable.

In the OSI/UK User Group N.L. from Feb. '81, there is an article describing another way to go that creates faster display speeds than V3.2 and retains the original page 2 code used by Cegmon. IT EVEN PERMITS A WARM START AFTER A BREAK!! In the April/May'81 issue there is a linker for the SUPERBOARD II that needs a different linker.

Justin Johnson wrote the original "CEGMON-OS65D Linker". Since then it has been distributed by the OSI/UK Programme Library to all subscribers as a free utility for personal use.

LISTING 1

```
1 FOR I=1 TO 30: PRINT : NEXT
2 GOSUB 1320: REM "'CEG)65' 1983+850816
3 IF PEEK(548)=208 THEN PRINT "CEG.ACTIVE!": RUN "BEXEC*
4 PRINT : PRINT TAB( 10)"J.G.JOHNSON CEGMON ) OS65D LINKER FOR
                                                         BASIC"
5 PRINT : PRINT TAB( 10)"Restores Warm-start & all CEGMON func
                                                         tions!"
6 PRINT : PRINT TAB( 24)"For 64X30": PRINT
7 REM "Partition off top page of memory.
8 PRINT : L=PEEK(8960): POKE 133,(L-1): POKE 8960,(L-1)
9 REM "Transfer page-2 of 65D to top of memory.
10 FOR J=512 TO 767
20 A=PEEK(J): POKE J+(L-2)*256,A
25 NEXT
27 REM "Memory is now free from $0235 to $02FF!
29 :
30 REM "Change vectors from Page-2 to top page.
40 READ N: IF N=999 THEN 1000
60 POKE N+2,L
70 GOTO 40
100 DATA 2037: REM "$07F5
110 DATA 2041: REM "$07F9
120 DATA 8908: REM "$22CC
500 DATA 999
600 :
900 REM "Put data into page-2 for Cegmon---
1000 PRINT : PRINT : PRINT
1010 DATA 0,32,10,0,0,0,0,8,43,11,35,15
1020 DATA 87,11,165,9,165,8,0,10,100,10,10
1022 :
1025 REM "Check $FBB2-FBCE for YOUR values for following:
1026 REM " Dec.=64434-64462
1027 :
1030 DATA 0,70,251,155,255,148,251,112,254,123,254
1040 DATA 63,64,208,128,215,189,64,208,157
1050 DATA 64,208,202,96,0,32,64,208,71,37
1080 DATA -1
1110 J=511
```

Listing continued on next page

ATTENTION C-3 AND 200 SERIES USERS

**NOW IS THE TIME TO REPLACE THAT POWER CONSUMING MEMORY OR ADD ANOTHER PARTITION
ISOTRON IS OFFERING THE CM-20 WITH 2 MHz OF STATIC MEMORY AT THE SPECIAL LOW PRICE OF $275, WHILE SUPPLIES LAST.**

For more information on the CM-20 Special Offer or for other expansion ideas, contact your local ISOTRON dealer or call us directly.

ISOTRON

140 Sherman St. Fairfield, Ct. 06430 • (203)255-7443

It will only work with BASIC and from the Kernel "A*" prompt. It moves page 2 BASIC code to the highest page of memory, replacing it with Cegmon data up to $0234. This means that you can use $0235 to $02FF for all your old page 2 utilities without locking up! You can BREAK to the monitor to do machine code work and still get back into BASIC. I've been using it regularly for many years and haven't had any trouble with it.

The only thing you have to avoid is any routine that loads to $0200 and up, like the ZERO COPIER! A re-boot will be necessary in these cases. Keep at least one disk unlinked, if you decide to make pre-linked disks copies! A similar link can be made for the ASM, but I haven't made any attempt at that yet.

The display is set to 64 x 30 in lines 1040 to 1060. You should check the code in your version of Cegmon at $FBB2-FBCE inclusive, (Dec. 64434-64462). Copy this down to line 1030 after the "8", since this is machine dependent, unless you want to change your display! This is determined in line 1040, starting with "63, 64, 208, 128, 215". 63 is the display width-1. The 64, 208 are the LO/HI bytes for top of screen and 128, 215 the bottom.

REMs explain the rest of the changes. You don't need to make any changes to your disks other than perhaps putting it into your BEXEC* or calling it from there.

Listing No. 1 is the original code with a few extra additions that I use daily. (Spaces should be eliminated to get code, all 72 characters, into one line.)

Listing No. 2 contains a re-write that Steve Hanlan of Beaver Systems/UK did for the Superboard Series 2 and III with BEXEC* functions added for booting directly to the linker. Check for the SB 2/III pointers for the DATA screen POKEs at $FBD0-FBEA, Dec. 64464-64490, in your CEGMON monitor with lines 150-170.

The screen display is set for 48 x 12. To make a larger display, change to:

160 DATA 47,10,208,202,213,

This is the largest size and gives 48 x 16. If this does not work, try lowering the 213 to 212 or back to 211 again. You should see at least 2 more lines at the top of your screen. If it works, change the rest of the 138's on lines 160 and 170 to 10. Then all pointers will be the same for the Editor and the cursor when linked. The rest of the POKEs can be used from Listing No. 1 from line 1220 and on, if you wish.

CEGMON for the SB 2/III is different from the original version of CEGMON that I have, so I can't verify its function. Good Luck!

(NOTE: Backspacing when working from the A* prompt won't remove any mistakes! Just hit RETURN and try again. This can be corrected in the DOS at $2CB2. Try it for your own pleasure. See page 41 in the

```
1120 READ N: IF N=-1 THEN 1190
1130 J=J+1: POKE J,N
1140 GOTO 1120
1150 :
1190 REM "Poke to link backspace
1200 POKE 1419,255: POKE 2820,255: POKE 11430,255
1205 REM "Poke to change screen vector
1210 POKE 8979,53: POKE 8980,248
1212 REM "Poke to link editor
1213 POKE 9522,189: POKE 9523,250
1216 REM "Remove keyboard masking in BASIC and DOS.
1217 POKE 1382,1: POKE 1386,255: POKE 9061,255
1218 REM "To activate Ind.File, poke 1382,32
1219 :
1220 PRINT CHR$(26): PRINT : PRINT TAB( 25)"CEGMON LINKED
1230 PRINT : PRINT TAB( 26)"J.G.J. 1981": PRINT : PRINT
1232 :
1235 REM "Added by O.Sw.:
1240 REM "Disable ASM & EMON when CEGMON is linked
1245 POKE 11824,1: POKE 11825,1: POKE 11844,1: POKE 11845,1
1247 REM "Turn off Editor Cursor/Ensure W.St.
1250 POKE 516,0: POKE 0,76
1260 REM "Change Ctrl/X to Ctrl/I for INDIRECT FILE
1270 POKE 9594,9: REM "Formerly 24
1275 :
1280 PRINT TAB( 22)"REBOOT for ASM & EM!": PRINT : PRINT
1290 PRINT TAB( 24)"INDFILE changed": PRINT
1300 PRINT TAB( 23)"to CTRL/I at 9594": PRINT : PRINT
1310 PRINT "System open.": NEW
1320 IF PEEK(8999)=58 THEN "Not for V3.3 or V5.14!!": RUN"BEXEC*
1330 RETURN
Ok
```

LISTING 2

```
1 FOR I=1 TO 30: PRINT : NEXT
2 REM "'CII>65' 850816
4 PRINT "J.G.JOHNSON CEGMON > OS65D LINKER FOR BASIC"
6 PRINT : PRINT "For SUPERBOARD Series 2 and III": PRINT
7 REM "ORIG. CODE FROM LINE 10 ----------------
10 X=PEEK(10950): POKE 8993,X: POKE 8994,X
15 POKE 741,76: POKE 750,78
20 POKE 2073,173: POKE 2893,55: POKE 2894,8
25 L=PEEK(8960): POKE 133,(L-1): REM reserve top page of memory
30 POKE 8960,(L-1): FOR J=512 TO 767: A=PEEK(J)
35 POKE J+(L-2)*256,A: NEXT : REM swap look-up to top page
40 READ N: IF N=999 THEN J=511: GOTO 60
50 POKE N+2,L: GOTO 40: REM insert links into BASIC
60 READ N: IF N=-1 THEN 80
70 J=J+1: POKE J,N: GOTO 60: REM swap CEGMON pointers into page2
80 POKE 1419,255: POKE 11430,255
85 POKE 8979,54: POKE 8980,248: REM series 2 screen handler
90 POKE 9522,209: POKE 9523,250: REM series 2 input
95 POKE 1382,5: PRINT CHR$(26);
100 PRINT "CEGMON linked .... System open": NEW
110 DATA 2037,2041,8908,999
120 DATA 0,32,10,0,1,0,0,8,43,11,35
140 DATA 10,10,0: REM SYNMOM-type pointers
150 DATA 180,250,155,255,57,255,112,254,123,254: REM vectors
160 DATA 47,138,208,138,211: REM screen table values
170 DATA 189,138,208,157,138,208,202,96
180 DATA 0,64,138,208,-1
Ok
```

"Bible" (OS-65D V3.2 Disassembly Man.) sold dirt cheap by PEEK. EA's are used to fill up the code before you reach $2CB9 since half of the code isn't needed with CEGMON!)

LETTERS

ED:

POKE Y, PEEK(X)

In response to the question of Tom McGourin, the construction POKE Y,PEEK(X) does work but not in the manner intended. The problem is PEEK and POKE both use the same locations to store the calculated address. For example, the value of Y in POKE Y is first calculated and stored. Then the value of X in PEEK(X) is calculated and stored in the same location. Thus the value of Y is lost. The expression is reduced to POKE X,PEEK(X) which does function, but accomplishes nothing.

The following short program demonstrates it is possible to POKE a PEEK. Each video address is PEEKed, the value of 1 added, and POKEd back to the screen. You will see all of the screen display has been moved up one ASCII character. For example, a blank (ASCII=32) is now "!" (ASCII=33).

Program

```
10 FOR X=53376 TO 55040

20 POKE 0,PEEK (X) +1

30 NEXT X
```

The zero after POKE is meaningless, any value could be put here since it gets overwritten. POKE Y,PEEK (X) is useful only if you want to look at a location, change the value and put it back into the SAME location. I'm not sure how useful a tool this is. Other (non-OSI) versions of BASIC work properly.

POKE X, PEEK(Y)

As noted, the construction POKE X, PEEK(Y) does not function correctly. If you are using Disk BASIC, the POKE function can be patched to allow POKEing a PEEK. Following is Assembly code for a new routine to be located somewhere in free RAM. The first two subroutines get the first variable which is the POKE location. The result in locations $19 and $1A is then saved before picking up the second variable. This avoids having the PEEK location overwriting the POKE location.

```
                    ;       PATCH TO 65D3.2 TO ALLOW
                    ;      POKE X, PEEK(Y)
                    ;       TO WORK CORRECTLY
                    ;      INITIALIZE WITH  DISK!"GO 8000"
                    ;
8000                        *=$8000
8000 A94C                  LDA #$4C     STORE JMP IN POKE ROUTINE
8002 8D9316                STA $1693
8005 A910                  LDA #START   LO BYTE
8007 8D9416                STA $1694
800A A980                  LDA #START/256 HIGH BYTE
800C 8D9516                STA $1695
800F 60                    RTS
                    ;
                    ;     START NEW POKE ROUTINE
8010 20B90C START          JSR $0CB9   GET FIRST VARIABLE X
8013 207216                JSR $1672   CHECK IN RANGE
8016 A519                  LDA $19     SAVE RESULT LO BYTE
8018 8D2580                STA POKE+1
801B A51A                  LDA $1A     SAVE RESULT HI BYTE
801D 8D2680                STA POKE+2
8020 206C16                JSR $166C   GO BACK TO NORMAL POKE
                    ;                  TO PICK UP SECOND VARIABLE (Y)
8023 8A                    TXA        GET VALUE TO A
8024 8DFFFF POKE           STA $FFFF  SELF MODIFYING CODE- STORE A
8027 60                    RTS        BACK TO BASIC
```

```
10 REM  SAMPLE BASIC PROGRAM TO  POKE X,PEEK(Y)
20 REM  AND MOVE VIDEO SCREEN
30 :
40 DISK!"GO 8000": REM INITIALIZE PATCH
50 :
60 FOR X=53312 TO 55104
70 POKE X, PEEK(X+32)
80 NEXT
```

The location of the new routine must be patched into BASIC, however, you cannot use POKEs to modify the POKE routine! The command DISK!"GO 8000" is used to set up the jump to the new routine.

A sample BASIC program is included which switches the left and right hand sides of the video screen. Line 70 of the BASIC program now runs as expected.

Earl Morris
Midland, MI 48640

* * * * *

ED:

OS-65D 3.3 has a screen printer called by "!80". This sends the video screen to an EPSON printer. The trouble I'm having is that it prints the screen all on one 24 character line with a lot of "/////" marks where the blanks should be. Have you ever heard of this problem?

Herbert H. Grassel
Wheaton, MD 20906

* * * * *

ED:

I am looking for someone to convert a standard CP/M format disk (that's the IBM 3740 synchronous format) to the Lifeboat/OSI CP/M format (that's an OSI asynchronous format). Several companies which used to advertise this service in PEEK say that they no longer have this service available.

Alternatively, it would be nice to know if there is a source for CPMUG (or other CP/M group) library software on Lifeboat/OSI CP/M format disks.

If anyone knows where either service is currently available, it would be appreciated if they would write PEEK(65) with the details.

Ray Hackney
Dallas, TX 75253

* * * * *

OSI/ISOTRON

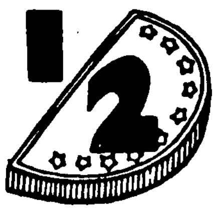
MICRO COMPUTER SYSTEM SERVICE

- - - - - - - - - - - - - - - - - - - -

*C2 AND C3 SERIES
*200 AND 300 SERIES
*FLOPPY DISK DRIVES
*HARD DISK DRIVES
CD 7/23/36/74
*TERMINALS, PRINTERS, MODEMS
*BOARD SWAPS
*CUSTOM CONFIGURATIONS
*CUSTOM CABLES
*SERVICE CONTRACTS

PHONE (616) 451-3778

COMPUTERLAB, INC.
307 MICHIGAN ST. N.E.
GRAND RAPIDS, MI. 49503



PRICE INVENTORY SALE

OUR STOCK ROOM IS OVERFLOWING!

FILL YOUR LIBRARY WITH MISSING MANUALS FOR LESS THAN 1/2 PRICE

All starred items are at 1/2 the marked price
Foreign orders by VISA/MASTER/CHOICE only, plus postage.
Orders can not be sent to P.O. Box addresses.

GOODIES for OSI Users!

PEEK (65)
The Unofficial OSI Users Journal

P.O. Box 347 • Owings Mills, Md. 21117 • (301) 363-3268

( )	**C1P Sams Photo-Facts Manual.** Complete schematics, scope waveforms and board photos. All you need to be a C1P or SII Wizard, just	$7.95 $ ______	★
( )	**C4P Sams Photo-Facts Manual.** Includes pinouts, photos, schematics for the 502, 505, 527, 540 and 542 boards. A bargain at	$15.00 $ ______	★
( )	**C2/C3 Sams Photo-Facts Manual.** The facts you need to repair the larger OSI computers. Fat with useful information, but just	$30.00 $ ______	★
( )	**OSI's Small Systems Journals.** The complete set, July 1977 through April 1978, bound and reproduced by PEEK (65). Full set only	$15.00 $ ______	★
( )	**Terminal Extensions Package** - lets you program like the mini-users do, with direct cursor positioning, mnemonics and a number formatting function much more powerful than a mere "print using." Requires 65U.	$50.00 $ ______	
( )	**RESEQ** - BASIC program resequencer plus much more. Global changes, tables of bad references, GOSUBs & GOTOs, variables by line number, resequences parts of programs or entire programs, handles line 50000 trap. Best debug tool I've seen. MACHINE LANGUAGE - VERY FAST! Requires 65U. Manual & samples only, $5.00 Everything for	$50.00 $ ______	
( )	**Sanders Machine Language Sort/Merge** for OS-65U. Complete disk sort and merge, documentation shows you how to call from any BASIC program on any disk and return it or any other BASIC program on any disk, floppy or hard. Most versatile disk sort yet. Will run under LEVEL I, II, or III. It should cost more but Sanders says, "...sell it for just..."	$89.00 $ ______	
( )	**KYUTIL** - The ultimate OS-DMS keyfile utility package. This implementation of Sander's SORT/MERGE creates, loads and sorts multiple-field, conditionally loaded keyfiles. KYUTIL will load and sort a keyfile of over 15000 ZIP codes in under three hours. Never sort another Master File.	$100.00 $ ______	
( )	**Assembler Editor & Extended Monitor Reference Manual** (C1P, C4P & C8P)	$6.95 $ ______	★
( )	**65V Primer.** Introduces machine language programming.	$4.95 $ ______	★
( )	**C1P, C1P MF, C4P, C4P DF, C4P MF, C8P DF Introductory Manuals** ($5.95 each, please specify)	$5.95 $ ______	★
( )	**Basic Reference Manual** — (ROM, 65D and 65U)	$5.95 $ ______	★
( )	**C1P, C4P, C8P Users Manuals** — ($7.95 each, please specify)	$7.95 $ ______	★
( )	**How to program Microcomputers.** The C-3 Series	$7.95 $ ______	★
( )	**Professional Computers Set Up & Operations Manual** — C2-OEM/C2-D/C3-OEM/C3-D/C3-A/C3-B/C3-C/C3-C'	$8.95 $ ______	★

( ) Cash enclosed ( ) Master Charge ( ) VISA

Account No. ______ Expiration Date ______

Signature ______

Name ______

Street ______

City ______ State ______ Zip ______

TOTAL	$______
MD Residents add 5% Tax	$______
C.O.D. orders add $1.90	$______
Postage & Handling	$ 3.70
TOTAL DUE	$______

POSTAGE MAY VARY FOR OVERSEAS

ED:

I noticed the warning in a recent issue to users of WP-6502 in 65U Multi-user environments. I have indeed had crashes due to the trash that this word-processing software leaves in the operating system. Fortunately, I never lost any data or ruined any files; now I only run it under Level 1.

Question: Could WP6502 be run in a partition under Level 3 by itself with no other software and not affect the other users or compromise system integrity?

Henry St. Clair
Roanoke, VA 24019

Henry:

I feel badly if I have unduly alarmed anyone who read the note about "garbage" systems.

Only the WP6502 user is involved when he subsequently runs an incompatible program, unless the user happens to be the Base or "0" user and he then "re-boots" another user.

It works like this. Running LEVEL III or re-booting a user copies the "System" in the base partition to the respective users. If the base has run WP6502 and garbaged the base system, re-booting users (all or specific) will proliferate the garbage. The only solution is to re-boot the machine from the H/D/M. Any other user may simply be re-booted provided the base system is "clean."

Eddie

* * * * *

ED:

Does anyone have a good circuit for connecting the video output from the computer directly to the video amplifier of a television, thus eliminating the need for the RF modulator? This is for a transformerless power supply television and would require the use of an optocoupler for isolation purposes. I would be glad to hear from anyone that is using such a circuit.

Robert L. Dingle
Dayton, OH 45429

HUMOR!

Real Programmers don't write in BASIC. Actually, no programmers write in BASIC .... after age twelve.

AD$

*** * * * * GIVE AWAY * * * * ***
Multi-Strike Printer Ribbons

What do you currently pay for a multi-strike ribbon cartridge? About $4.00 each in lots of 6?

We have found a solution that may cause you never to use a fabric ribbon again. 1) Did you know that most all multi-strike ribbon cartridges use the same ribbon bobbin? It is just pressed on a different size hub and put in your cartridge type. 2) We have found a source of recently outdated (yes, many are dated) Diablo Hi-Type I cartridges. We took the oldest one we could find, put it in our NEC cartridge and printed this ad. Now, honestly, do you see any difference? We can't either. So we are offering those of you who use Hi-Type I, or are willing to pry open whatever cartridge you are using and replace the bobbin, a deal you can't refuse.

Buy one box of 6 cartridges for $8.00 and we will give you a second box FREE. That's 66.66 cents a piece or 83% off. At that rate, how can you lose? Add $3.00 for postage and handling. Make check or money order (in U.S. funds, drawn on a U.S. bank) payable to PEEK(65). P. O. Box 347, Owings Mills, MD 21117. Order NOW, supply limited!

* * * * *

FOR SALE: 65U Telecommunications Software. Basic/Assembler package configurable to different protocols. Control characters allow "toggle" between Terminal/Computer Modes. Send/Receive files. I'm on-line with Western Union EZ-Link. Call Henry St. Clair (703) 992-1800 (Home 992-5666).

* * * * *

MAGNETIC TAPE: Graham magnetics 2400 ft reels 6250 BPI used once. 300 reels available at $10 each including shipping. Send payment with order to Associated Information Systems, 825 Osceola Dr., Rockledge, FL 32955.

* * * * *

FOR SALE: OSI C3-C Computer System - Including 104K RAM; 36 MByte Hard Disk; 2 8" Floppy Disk Drives; OS-65U Version 1.42; PDS-1A Power Sequencer Keyswitch; CA-10 Board with 4 Terminal Ports; 2 TVI 912 Terminals; Qume Sprint Series Daisy Wheel Printer with tractors; 300 Baud CAT Modem; 300 Baud Ventel Modem; Serial Switch Box for printers; 40 8" Floppy Disks; Terminal Cables and Connectors. Asking only $4,000.00 for the entire system. Everything is in good working condition. I will consider selling components. Please call (216) 757-8730 anytime between 9:00 AM & 5:00 PM Eastern Time. Ask for Jim.

* * * * *

OSI TECHNICIAN PROGRAMMER WANTED. Must have experience in general hardware and programming DMS Multi-user business applications. Needs general knowledge of accounting and related issues. Send resume, references, and a letter explaining the details of your experience. State your desired starting pay and expected pay goals for three years from now. No smokers or drinkers. Will pay reasonable moving expense. Fessenden Computers, 116 North 3rd St., Ozark, MO 65721.

* * * * *

FOR SALE: C3-OEM. Best Offer. Accounting software - General Ledger, Accounts Receivable, Accounts Payable, Payroll, under 65U. Mac N. Claxton, CPA. 8121 Georgia Avenue, Silver Spring, MD 20910. (301) 495-0035.

* * * * *

Send for free catalog, Aurora Software, 37 South Mitchell, Arlington Heights, IL 60005. Phone (312) 259-4071.

AT LAST!

SINGLE DISK FLOPPY COPIER FOR OS-U MACHINES WITH ONLY 1 FLOPPY DRIVE

SDCOPY

This is the utility missing since OSI started making single floppy drive machines. Now with Leo Jankowski's SDCOPY, you can make back-up floppies without involving your hard disk.

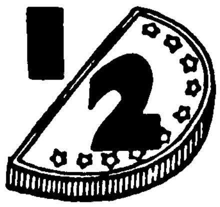
USA- Including P & H $25.50
Foreign- Plus Actual Postage

PEEK(65)
P. O. Box 347
Owings Mills, MD 21117
(301) 363-3268

PEEK [65]
The Unofficial OSI Users Journal

P.O. Box 347
Owings Mills, Md. 21117

DELIVER TO:



PRICE INVENTORY SALE

OUR SHELVES ARE BULGING!

HERE'S YOUR CHANCE TO COMPLETE YOUR LIBRARY AT LESS THAN 1/2 PRICE

Get a one year volume set (12 back issues) for $15.00 and we will pay UPS.
Get one back issue of the OSIO newsletter free with order.
Foreign orders by VISA/MASTER/CHOICE only, plus postage.
Orders can not be sent to P.O. Boxes.

NAME............................STREET.........................

CITY............................STATE ....

ZIP CODE........................COUNTRY........................

Please send me the following volume(s). I enclose:

Vol 2, 1981 ( )

JAN #1	FEB #2	MAR #3	APR # 4	MAY # 5	JUN # 6
JUL #7	AUG #8	SEP #9	OCT #10	NOV #11	DEC #12

Vol 3, 1982 ( )

JAN #1	FEB #2	MAR #3	APR # 4	MAY # 5	JUN # 6
JUL #7	AUG #8	SEP #9	OCT #10	NOV #11	DEC #12

Vol 4, 1983 ( )

JAN #1	FEB #2	MAR #3	APR # 4	MAY # 5	JUN # 6
JUL #7	AUG #8	SEP #9	OCT #10	NOV #11	DEC #12

Vol 5, 1984 ( )

JAN #1	FEB #2	MAR #3	APR # 4	MAY # 5	JUN # 6
JUL #7	AUG #8	SEP #9	OCT #10	NOV #11	DEC #12