Part 1
A Small Operating System: OS65D The Kernel

T. R. Berger

You switch on your computer, insert a disk, press the RESET button, then press "D". After a second or two of whirring and clicking, a menu flashes on the screen asking what you wish to do. At the time of turn-on, your computer knew nothing. Now BASIC LANGUAGE is in control. How did this happen? How does the machine get data and programs to and from the disk? How does the computer know if you are using a video monitor with a polled keyboard or an expensive serial monitor terminal? How does the computer decide to send its messages to the printer, the monitor, or to memory? How does the computer find its way among BASIC, the Assembler, and the Extended Monitor? The glib answer to all these questions is that the disk operating system makes all decisions and performs all control operations.

It is the task of the operating system to:

1. Start the computer on RESET (BREAK);
2. Manage and control all external input and output devices including keyboards, monitors, printers, and so on;
3. Manage the functioning of the disk (the single most important function of an operating system);
4. Manage loading and execution of system software in the software segment of memory, including BASIC, Assemblers, etc.; and
5. In general, keep tidy control over all transfers between these various functions.

The diagram in FIGURE 1 illustrates this mediating function of an operating system.

I hope, in several articles, to describe some of the general features of a small operating system by describing in some detail how the Ohio Scientific OS65D disk operating system functions. OS65D is a minimal function, small sized operating system. Therefore, mere mortals can comprehend its structure. I hope to convey not only a general understanding of this system, but also to provide you with some nuts and bolts to use in your own programming efforts. This includes memory maps of all subroutines.

The OS65D operating system is divided into several parts.

1. Cold Start ROM

This program takes about 256 bytes (one page) of ROM and accomplishes an absolute minimum of functions. Its major role is to load Track 0 of the disk into memory and start it running.

2. The Preparation Program

This program is in memory only long enough to do its job. All the various bits and pieces hanging on your microprocessor wake up either turned off or in some random state. Some of these need preparation before they will function properly. For example, a serial input/output port (such as used by a terminal) operates through an ACIA (Asynchronous Communication Interface Adapter....who thought up that mouthful?) which must be prepared for proper functioning. This program carries out these preparations.

3. The Operating System Kernel

This program is the 'BOSS'. It contains all the commands by which you may direct the operating system. It directs the functioning of the remaining parts of the operating system.

4. Disk Routines

These are the programs which make the disk the magic storage medium which it is. These routines start and stop the clicking and whirring you hear when the disk operates.

5. The Input and Output Routines

Input may come from a keyboard, a serial terminal, or any of many sources. Similarly, output may go to a serial terminal, a video monitor, a printer, or some other device. Each such device needs its own input or output program. Further, there must be one supervisory program which can choose from as many or as few of these input/output devices as are desired at any one time. These programs constitute the Input and Output Routines.

6. Utility Programs

Certain programs are needed only occasionally. These include disk copying programs, and Track 0 modification programs. These utility routines are only loaded into memory when needed. They hide in sectors placed after the major system software on various tracks of the disk.

In this article we will explore the 'BOSS', i.e. the Operating System Kernel. The obvious part of the kernel is the set of commands. Not so obvious, but very useful, are the line input routine, the line buffer reader, and various Hex to ASCII and ASCII to Hex conversion and other routines. Let's go through the various commands availalbe in OS65D and see how they function.

OSI

SOFTWARE FOR OSI

OSI

A JOURNAL FOR OSI USERS!!

The Aardvark Journal is a bimonthly tutorial for OSI users. It features programs customized for OSI and has run articles like these:

1) Using String Variables.

2) High Speed Basic On An OSI.

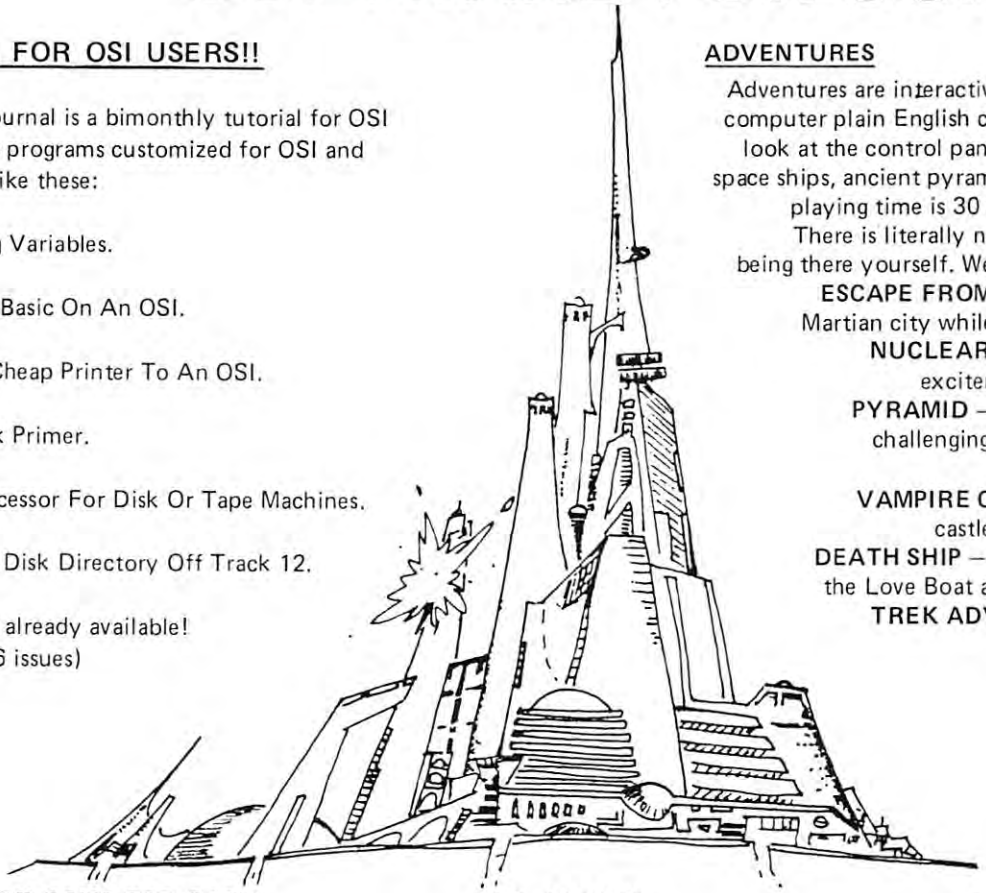
3) Hooking a Cheap Printer To An OSI.

4) An OSI Disk Primer.

5) A Word Processor For Disk Or Tape Machines.

6) Moving The Disk Directory Off Track 12.

Four back issues already available!
$9.00 per year (6 issues)



ADVENTURES

Adventures are interactive fantasies where you give the computer plain English commands (i.e. take the sword, look at the control panel.) as you explore alien cities, space ships, ancient pyramids and sunken subs. Average playing time is 30 to 40 hours in several sessions. There is literally nothing else like them – except being there yourself. We have six adventures available.

ESCAPE FROM MARS – Explore an ancient Martian city while you prepare for your escape.

NUCLEAR SUBMARINE – Fast moving excitement at the bottom of the sea.

PYRAMID – Our most advanced and most challenging adventure. Takes place in our own special ancient pyramid.

VAMPIRE CASTLE – A day in old Drac's castle. But it's getting dark outside.

DEATH SHIP – It's a cruise ship – but it ain't the Love Boat and survival is far from certain.

TREK ADVENTURE – Takes place on a familiar starship. Almost as good as being there.

$14.95 each

NEW SUPPORT ROMS FOR BASIC IN ROM MACHINES

C1S – for the C1P only, this ROM adds full screen edit functions (insert, delete, change characters in a basic line.), Software selectable scroll windows, two instant screen clears (scroll window only and full screen.), software choice of OSI or standard keyboard format, Bell support, 600 Baud cassette support, and a few other features. It plugs in in place of the OSI ROM. NOTE: this ROM also supports video conversions for 24, 32, 48, or 64 characters per line. All that and it sells for a mesly $39.95.

C1E/C2E for C1/C2/C4/C8 Basic in ROM machines.
This ROM adds full screen editing, software selectable scroll windows, keyboard correction (software selectable), and contains both an extended machine code monitor and a fix for the string handling bug in OSI Basic!! It has breakpoint utilities, machine code load and save, block memory move and hex dump utilities. A must for the machine code programmer replaces OSI support ROM. Specify system! $59.95

STRING BUG FIX (replaces basic ROM chip number 3)
All this chip does is to replace the third basic ROM and correct the errors that were put into the ROM mask. $19.95

DATA SHEETS

OS65D LISTING
Commented with source code, 83 pages. $24.95

THE (REAL) FIRST BOOK OF OSI
65 packed pages on how OSI basic works. Our best selling data sheet. $15.95

OSI BASIC IN ROM
Ed Carlson's book of how to program in basic. Now available from Aardvark. $8.95

P.C. BOARDS

MEMORY BOARDS!! – for the C1P. – and they contain parallel ports!
Aardvarks new memory board supports 8K of 2114's and has provision for a PIA to give a parallel ports! It sells as a bare board for $29.95. When assembled, the board plugs into the expansion connector on the 600 board. Available now!

REAL SOUND FOR THE C1P – and it's cheap! This bare board uses the TI sound chip to give real arcade type sound. The board goes together in a couple of hours with about $20.00 in parts. Bare board, plans, and sample program – $15.95

ARCADE AND VIDEO GAMES

ALIEN INVADERS with machine code moves – for fast action. This is our best invaders yet. The disk version is so fast that we had to add selectable speeds to make it playable.
Tape - $10.95 – Disk - $12.95

TIME TREK (8K) – real time Startrek action. See your torpedoes move across the screen! Real graphics – no more scrolling displays. $9.95

STARFIGHTER – a real time space war where you face cruisers, battleships and fighters using a variety of weapons. Your screen contains working instrumentation and a real time display of the alien ships. $6.95 in black and white - $7.95 in color and sound.

SEAWOLFE – this one looks like it just stepped out of the arcades. It features multiple torpedoes, several target ships, floating mines and real time time-to-go and score displays. – $6.95 in black and white $7.95 in color and sound.

SCREEN EDITORS

These programs all allow the editing of basic lines. All assume that you are using the standard OSI video display and polled keyboard.

C1P CURSOR CONTROL – A program that uses no RAM normally available to the system. (We hid it in unused space on page 2). It provides real backspace, insert, delete and replace functions and an optional instant screen clear.
$11.95

C2/4 CURSOR. This one uses 366 BYTES of RAM to provide a full screen editor. Edit and change lines on any part of the screen. (Basic in ROM systems only.)

FOR DISK SYSTEMS – (65D, polled keyboard and standard video only.)

SUPERDISK. Contains a basic text editor with functions similar to the above programs and also contains a renumberer, variable table maker, search and new BEXEC* programs. The BEXEC* provides a directory, create, delete, and change utilities on one track and is worth having by itself. – $24.95 on 5" disk - $26.95 on 8".

DISK UTILITIES

SUPER COPY – Single Disk Copier
This copy program makes multiple copies, copies track zero, and copies all the tracks that your memory can hold at one time – up to 12 tracks at a pass. It's almost as fast as dual disk copying. – $15.95

DISK CATALOGER
This utility reads the directory of your disks and makes up an alphabetic list off all your programs and what disks they are on. $14.95

MACHINE CODE RENUMBERER
(C2/4-MF only)
Renumbers all or part of a program at machine code speeds. – $15.95

VISA

This is only a partial listing of what we have to offer. We now offer over 100 programs, data sheets, ROMS, and boards for OSI systems. Our $1.00 catalog lists it all and contains free program listings and programming hints to boot.

master charge

OSI

Aardvark Technical Services • 1690 Bolton • Walled Lake, MI 48088
(313) 669-3110 or (313) 624-6316

OSI

OS65D Kernel Command Descriptions

The kernel has 18 user commands. These may be divided into four categories as follows: (1) Commands which move data or programs from the disk to memory; (2) Commands which reverse this process and move data or programs from memory to the disk; (3) Commands used for disk diagnostics and preparation; and (4) Other commands. With this division in mind, let's discuss the function of each command by category.

Transfers from the disk to memory

The 6 commands (BA, AS, EM, XQ, LO, CA) in this category can be subdivided into four which load and run: BA, AS, EM, XQ, and two which just load: LO, CA. The ones which load and run have their vital statistics listed in Table 1 where the track numbers are for 8'' diskettes. After the tracks have been loaded to memory, program control is transferred to the location listed under 'jump' in the table. The commands BA (BASIC), AS (Assembler), and EM (Extended Monitor) are self-explanatory. They load and run languages and systems supplied with your computer. A general command much like BA, AS, and EM is XQ (EXECUTE). If you develop machine language systems to run at $317E then XQ NAME or XQ TRACK will load and run these sytems where NAME is the name of your file and TRACK is the first track number of the file. These commands load integral numbers of tracks, and thus will not load sectors from within tracks. They offer great ease of operation but practically no versatility.

To add versatility, we need two more general commands (LO, CA). We need a command LO (LOAD) to accomplish what the previous commands do for whole tracks without adding the 'run' feature at the end. This one additional command is not enough. Each track stores 3K bytes of data. It is rather inefficient to store a 200 byte program on one full track. Therefore, the operating system allows us to divide each track into sectors. We are still limited by the fact that a sector must be an integral multiple of pages (1/4 K or 256 bytes) up to 3K. However, it is less wasteful to store 200 bytes in a 256 byte sector than to store it in a 3K track. The CA (CALL) command allows this sector type of operation.

For full tracks and for sectors we have two load commands listed in Table 1. First, LO NAME or LO TRACK loads a file named NAME or a file beginning at track number TRACK to memory. Second, CA MEMORY = TRACK, SECTOR calls sector number SECTOR on track number TRACK to memory, starting the load at memory address MEMORY. Note that LO specifies no starting address. Further, 'LO NAME' specifies no track number for the disk. When a file is named, a track number is found in the disk directory which resides in Track 8. The load vector (memory start address) is usually $3179 for the LO command. Since BASIC disk buffers are kept between $317E and the start of your program, this means that any BASIC program with a buffer will use disk space to preserve buffer space. Disk space is wasted, but the operating system remains very simple. Sectors could also be named in a directory with load vectors written into the first few bytes, but that would enlarge the memory requirements of the operating system and add to its complexity. The authors of OS65D chose to forgo enhancements. Thus the CA command requires all of the load data except the length of the sector, which is stored among the first few bytes of the sector.

The six commands just described (BA, AS, EM, XQ, LO, CA) provide a small, yet very powerful set for obtaining files from the disk. For simplicity and compactness of the system, the user is asked to suffer a little inconvenience in loading sectors. Further, since most data and program files will reside in named files, some disk inefficiency is accepted as the price of a compact operating system. In particular, no matter how long or how short a BASIC program is, it will always be stored on an integral number of tracks. A 1K program will use a 3K Track (or more if there are buffers). To change this would require more elaborate programming of sectors and directories. Under such a more elaborate system the disk would appear to be much larger. On the other hand, because more elaborate programming is necessary, the disk would run more slowly. However, compared to cassette tape, even these more elaborate programs would seem jet propelled.

Transfers from memory to the disk

There are no commands for saving memory which might be analogous with a 'load and run' command. Thus the operating system need only have commands which perform functions opposite to LO and CA. These are PU (PUT) and SA (SAVE) and are also given in Table 1. In analogy with LO, PU NAME or PU TRACK will put memory onto an integral number of disk tracks. If the file is named NAME, the directory will specify the starting track and how many tracks are available. The transfer always starts at memory location $3179 and will save T tracks (about T X 3K of memory) where T is given in $317D.

Similarly, SA TRACK, SECTOR = MEMORY/PAGE will SAVE memory beginning at memory address MEMORY and continuing for PAGE number of pages on track number TRACK in sector number SECTOR. The number of pages in a sector is saved on the disk, but is usually not stored in memory. Therefore, when saving memory, the length of the segment to be saved must be given in the command. The symbols ',', ' = ', and '/' used in the SA command serve only to separate addresses and numbers and to complicate your life. They all can be changed easily to ',' or spaces. The disadvantage of making such a change is that the order of the numbers is vital. Presumably ',', ' = ', followed by '/' help you keep the numbers in the

right order. If you don't, you get an error message rather than a disastrous SAVE which might overwrite some of your more beautiful programming efforts.

Commands used for disk diagnostics

Being mechanical devices, disks are not perfect. Occasionally you need to manipulate the disk or examine the entire contents of a given track. Further, you need to copy old and initialize new disks. There are commands for doing these things in the operating system kernel. We may divide these commands into 3 sets: (a) Reading from the disk, (b) Writing on the disk, and (c) Manipulating the disk.

These commands are listed in Table 1. First come the diagnostic read commands EX (EXAMINE) and DI (DIRECTORY). The command EX MEMORY = TRACK reads everything for examination from track number TRACK to memory beginning at address MEMORY. If you are encountering disk trouble or suspect a bad diskette, this is a very useful command. If you have inadvertently erased or overwritten part of a disk, this command may help salvage some of the remaining programs. If you are just trying to learn how your disk stores memory, this is a helpful command.

On the other hand, if you just wish to learn the status of a particular track (i.e. how many and how long are its sectors) then using EX can prove to be very tedious.

The command DI TRACK will print out a sector number and length directory (in pages) for track number TRACK. The disk directory tells us that OS65D occupies Tracks 0 - 8, but does not give us information as to how many sectors reside in, say Track 8. On the other hand, DI 08 tells us there are 4 sectors of length 1 page each on Track 8. Unfortunately, OS65D does not allow us to name individual sectors within a track. We can, however, name the track in which these sectors reside by using the BASIC 'CREATE' program.

There is one diagnostic command IN (INITIALIZE) for writing on the disk. It allows us to initialize a whole disk by IN or an individual track number TRACK by the command IN TRACK. When a track is initialized, the beginning of the track is found and track identification data are placed on the disk. Then the rest of the track is completely erased. No sector identification marks are placed on the disk so the track is not useable by LO or PU as it stands. The BASIC CREATE program will fix this problem.

Finally, there are three diagnostic disk manipulation commands (HO, SE, D9). The disk changes tracks by stepping the read head outward toward Track 0 or inward toward Track 76 one track at a time. The head moves when the stepper motor spins a fixed fraction of a revolution. This process is not perfect and occasionally the head will be misplaced on the disk. There is only one track (Track 0) where there is a sensor to detect whether or not the head is correctly positioned over the track. All other tracks are found by counting steps inward from Track 0, or counting up or down from the present track number. The present track number is saved in memory ($265D). Usually when the head is misplaced, it is only very slightly off the circular data stream on the disk. You may have noted this phenomenon in another context with music filled cassette tapes. A friend loans you his great sounding 'BOOMBAH' cassette which he made live. It sounds great on his HIFI but lousy on yours. The reason is that his recorder put the music track onto the tape in a position differing slightly from the place where your recorder is trying to find it.

If the disk head is slightly out of position on the disk, the same thing occurs, i.e. a lousy read. Your disk will detect this and step the head down one track then back to try again. Even though this process occurs very quickly, it is imperfect at best. Memory tells where the head is supposed to be. But in many jumps back and forth between tracks, 'supposed to be' an 'really is' could differ. If after a few tries at repositioning the head, the disk still fails to find the track, it quits and sends an error message. The solution to this problem is to start all over again. Move the head to Track 0 where it can mechanically sense its position then start up again. The HO (HOME) command does this by homing the read head to Track 0.

OSI SOFTWARE

VIDEOTREK NOT your ordinary STARTREK game, VIDEOTREK is a non-stop action chase around the galaxy in pursuit of invading Klingon cruisers. Stars, planets and Black Holes all must be avoided--and watch out for the Doomsday Machine! Four challenging levels of play with bonus time for high score. If you like STARTREK or TIMETREK, this is your game! $9.95

REBEL GUNNER The Rebel Alliance is in danger! It's up to you and your X-Wing fighter to catch the TIE fighters in your sector and destroy as many as you can. The TIE fighters dodge so quickly that your targeting computer is on automatic fire control while your hands are full just lining up your sights on the targets! The graphics in this one will make you forget You're sitting in front of your computer. Three levels. $9.95

RED BARON Battle the notorious Red Baron! Your squadron consists of three Spads and it's a race against time as you try to down as many enemy planes as you can before your fuel runs out. Three levels of difficulty. Wait'll you see what happens when you have to bail out after running out of gas or getting shot down by the Baron! $7.95

TANK'MAZE As Tank Commander, your mission is to blow up all the abandoned gun emplacements on the battlefield. Your task is complicated by the other objects in the area however. Mines, trees, houses and civilians must be avoided as you race to complete your mission in time. Each maze is different and each contains over 200 obstacles. $7.95

BARRIER TANK An enemy tank is placing barriers on the battlefield. You must destroy the enemy tanks to keep the area open. As the game goes on, the open spaces rapidly disappear, making the job more difficult. Two levels of difficulty. In Level 2 they're laying explosive mines! $7.95

NEW NEW NEW

aaaRRGGG! A fast paced, frantic chase around the screen, trying to catch some very elusive targets. It's a race against the clock as you try to roll up the highest possible score. Five levels of play and bonus time for high score! You'll understand the name when you try Level 5! $7.95

EARTH vs. FLYING SAUCERS The Earth is in a panic! Saucers are coming to open a new Peopleburger franchise! Can you stop them? Two levels of difficulty. This one is simpler to play than the others, with only one button to push, but it's still quite a good challenge. $6.50

All prices are postpaid-no "hidden" handling charges. All run in 8K on any OSI C1, C2 or C4 tape based computer. All are recorded twice on each tape and are covered by a Limited replacement warranty, return for replacement.

BOB RETELLE
2005 WHITTAKER RD.,YPSILANTI, MI. 48197

If you have run a BASIC program which requires a disk read midway and have been thrown out of your program with the Error #5 then you know how annoying this can be. The cure is to find the step in the BASIC program where the disk read occurs. Just preceding this step, insert a step with DISK!"HO". This instruction assures you that if the track requested in the next step can be found, it will be found without error. A more elaborate operating system would incorporate such a step in its track seeking logic (i.e. if the head fails to find the track after several tries, it would go to Track 0 and start over).

If you own more than one disk drive, (lucky you!) you may select any one by the command SE (SELECT) via SE DRIVE where DRIVE is A,B,C, or D (OS65D can control up to 4 drives). When you select a drive it is automatically homed and thus starts out aligned at Track 0.

Older versions of OS65D did not properly find the disk index hole at the beginning of a track. Newer versions do not have this problem, and go further to incorporate an error if the beginning of a track cannot be found quickly (i.e. within one revolution of the disk). Since older disks may take several revolutions before data synchronization takes place, OS65D will refuse to read these disks. Command D9 (DELETE 9) is supposed to eliminate this condition by short circuiting the new error. Even though the D9 subroutine is included in my version of OS65D, it is not connected. If I enter command D9, my command table sends the computer to the 'syntax error' subroutine instead of the D9 subroutine. This can be corrected by putting the D9 subroutine address (minus one) into the command table in place of the 'syntax error' address. I own no old OS65D disks, so I have not changed anything.

At this point, it might be worth alluding to diagnostic features of OS65D not in the kernel. Ohio Scientific was mortally afraid you might damage the vital kernel information on Track 0. Thus the kernel mightily protects Track 0 against your invasions.

If you happen to load a program into memory, to save it back onto the disk, and in the middle of the save, to change your mind and quickly to remove the diskette from the drive, then you will certainly cause an erasure somewhere on the diskette. This procedure (which you should avoid) places a very strong, rapidly varying magnetic field at an undetermined place on the diskette. Rapidly varying magnetic fields erase diskettes. If the undetermined place is in Track 0, part of Track 0 is lost. Therefore, every single part of the diskette must be changeable by the computer user, including Track 0. OS65D has a Track 0 read/write utility to accomplish this.

Most people have only one disk drive. In order to copy a disk one moves programs from an old diskette to memory and from there onto the new diskette. It's tedious, but it works. It would be very helpful to one-drive owners if you wrote a machine language program to simplify this process as much as possible.

There are others (with spare money) who have two or more disk drives. Two drives have the advantage that it is easy to copy from an old diskette in one drive to a new diskette in another drive, if you have a program. OS65D also contains a disk copying utility.

The copier and Track 0 utility programs are available in Sector 2 of Track 1 on the disk. In order to further protect you from the ways of error, and to save memory, these programs are not normally in memory. They are not part of the kernel. Thus we will discuss them in another article of this series. However, these programs are available for diagnostic purposes. They can be loaded by CA 0200 = 01,2. They can be run by GO 0200. I advise you to know what this program does and how it works before you try it. (Either wait for me or read your manual carefully.)

Other commands

There are 4 additional commands (RE, GO, IO, ME) in the kernel not associated with the disk:

The first of these is the restart command RE (RESTART). If you have just entered a BASIC program from the keyboard and wish to know how many tracks it will occupy on the disk, you type EXIT. This puts you in the command mode of the operating system kernel. If you typed BA (BASIC) to return to BASIC, a minor disaster would occur. BASIC would be loaded from the disk and the source file initialized. In simple terms, your program would be gone. (It really is salvageable, but that is a complicated process.) To avoid this problem we have a restart command. To restart BASIC, the command is RE B.

When BASIC is in memory, the Assembler and Extended Monitor are not. If you try to restart the Extended Monitor with RE E when BASIC is loaded, you receive a syntax error message. Using the RE command you may restart BASIC (RE B), the Assembler (RE A), the Extended Monitor (RE E), or the ROM Monitor (RE M) if they are in memory.

At this point it is worth discussing a rather subtle matter. Anytime you are someplace else in memory and able to GO at an arbitrary address, then you may restart OS65D by starting at $2A51. However, if you have used the keyboard without using the keyboard I/O routine in OS65D, you will have crashed BASIC or the Assembler, whichever is in memory. The reason is that the keyboard polling routine was written for ROM BASIC machines and as such uses storage locations $0213-$0216. Unfortunately, these locations are vital to BASIC and the Assembler. Thus, the I/O routine in OS65D swaps these locations out before going to the keyboard polling routine in ROM. After completing the keyboard poll, these locations are swapped back in again.

When you use RE M, these locations are swapped out since the ROM monitor uses the ROM keyboard polling routine. To swap these locations back in again you do not type $2A51G from the ROM monitor. Instead, you use a routine in the I/O section of OS65D which first swaps the keyboard back again and then goes to $2A51. So from the ROM Monitor, you restart OS65D by $2547G.

Through its various programs, the computer transfers control from one program to another. For example, RE B causes the computer to leave the kernel at the address $2C0D and enter BASIC at its WARM START location $20C4. If you have written your own machine programs, you may start them from the ROM monitor, the Extended Monitor, or the Operating System Kernel. To start a program from the kernel at address $4C00, the command is GO 4C00.

The final two OS65D kernel commands (IO, ME) control input to and output from the computer in a very simple way. One byte of memory consists of eight bits; each bit is either a 0 or a 1. One byte of memory is allocated as an input flag ($2321) and one as an output flag ($2322). Each of the eight bits represents an input (or output) device. If a particular device bit is 1, then that device is connected; if it is 0, that device is disconnected. We may imagine the bits arranged in a row as follows:

7	6	5	4	3	2	1	0
a	b	c	d	e	f	g	h

The bit itself is denoted by a letter in a box, and the number above is its position. The positions 0-7 stand for devices. These are given in Table 2. You may not recognize some of the devices because they are not part of your computer. However, if you so choose, you may buy these devices from OSI.

If bit 1 is 1 (g = 1) and all other bits are 0 in the input flag ($2322) and then input is taken from device 1, the keyboard. If bits 1 and 3 are 1 (e = 1 and g = 1) and all others are 0 in the output flag ($2321) then output is sent to the video monitor and the parallel printer. We may change the bits in the IO (INPUT/OUTPUT) flags ($2321 and $2322) via the INPUT/OUTPUT command IO INPUT,OUTPUT where INPUT and OUTPUT are the hexadecimal versions of the bits in the boxes. (IO, OUTPUT changes just the output flag and IO INPUT just the input flag.)

There is one intriguing device (bit 4 for both input and output) called MEMORY. How can memory be an input or output device? (Actually, memory is a storage device, just as cassette tape or a disk is. Thus we can put stuff into it and take it back out. As long as we do not erase memory, it will remain there. Usually material is put into and taken out of memory under program control. There may be circumstances where we do not want memory under program control. For example, suppose you have a long BASIC program that works on a large amount of text stored as strings (such as a justification program for a text editor). Assume the final text is to be sent out via a MODEM to a distant printer. Your justifier will chomp away producing and sending a string every now and then wasting a great deal of telephone time. A better approach would be to temporarily justify into memory, then send the resulting text. A computer has no idea where it gets its input or sends its output except via a subroutine. It does not care if it sends to the video monitor, the disk, a telephone, memory, or the moon.

The memory input/output capability is also used by the Indirect File. This program resides in the Input/Output section of OS65D and will be discussed in another article. One of the many uses of the Indirect File is to append many short BASIC programs end to end to make one long one.

To make use of memory as an input/output device via the command ME (MEMORY) we must know which part of memory to address. ME INPUT,OUTPUT sets the start address of the input to INPUT and the start address of the output to OUTPUT.

Hopefully, these descriptions of the OS65D commands, in conjunction with your OS65D USER's GUIDE will help you to make better use of the commands in your operating system. BASIC can execute any operating system command via DISK! "any OS65D command string". For example, if we have the following program lines then a program allows the user to select disk drive A or B.

```
100 INPUT "WHICH DRIVE (A/B)";A$
110 IF A$<>"A" AND A$<>"B" THEN 100
120 DISK!"SE" + A$
```

The Extended Monitor and Assembler can also send operating system commands via any OS65D command string.

TABLE 1

COMMAND	NAME	TRACKS	ADDRESS	JUMP
AS	ASSEMBLER	5 - 6	0200-1700	1300
BA	BASIC	2 - 4	0200-22FF	20E4
EM	EXTENDED MONITOR	7	1700-1FFF	1700
XQ	EXECUTE	USER	3179-	317E
CA	CALL	USER	USER	-
LO	LOAD	USER	3179-	-
PU	PUT	USER	3179-	-
SA	SAVE	USER	3179-	-

		FUNCTION
DI	DIRECTORY	SECTOR DIRECTORY
D9	DISK #9	DISABLE ERROR #9
EX	EXAMINE	EXAMINES A FULL TRACK
HO	HOME	HOME THE DISK TO TRACK 0
IN	INITIALIZE	INITIALIZE A DISK
SE	SELECT	SELECT A DRIVE
GO	GOTO	EXECUTE A MACHINE PROGRAM
IO	INPUT/OUTPUT	SET INOUT/OUTPUT FLAGS
ME	MEMORY	SET MEMORY IO VECTORS
RE A	RESTART ASSEMBLER	
RE B	RESTART BASIC	
RE E	RESTART EXTENDED MONITOR	
RE M	RESTART ROM MONITOR	

TABLE 2

INPUT/OUTPUT

BIT NUMBER	INPUT FLAG
0	SERIAL INPUT (ACIA)
1	POLLED KEYBOARD
2	CASSETTE INPUT ON 430 BOARD
3	NULL (0) INPUT
4	MEMORY INPUT
5	DISK BUFFER #1 INPUT
6	DISK BUFFER #2 INPUT
7	SERIAL INPUTS FROM 550 BOARD

BIT NUMBER	OUTPUT FLAG
0	SERIAL OUTPUT (ACIA)
2	VIDEO MONITOR
3	LINE PRINTER
4	MEMORY OUTPUT
5	DISK BUFFER #1 OUTPUT
6	DISK BUFFER #2 OUTPUT
7	SERIAL OUTPUTS FROM 550 BOARD ©

Next time: Subroutine descriptions...

COMPUTE!
Is Looking For
Good Articles For
Your Gazette

Send Program Listings, Articles, Hints, Odds and Ends, etc. to
The Editor
COMPUTE!
P.O. Box 5406
Greensboro, NC 27403 USA

OSI C1P Fast Screen Clears Revisited

Charles L. Stanford

Since writing the article on Screen Clear Routines for the OSI C1P for Compute II, Issue 1, I've been particularly sensitive to variations on machine language programming methods which could be used to improve the use of the computer. Several publications have been of considerable help, especially Compute and Compute II, Micro, the Aardvark and Progressive Computing Catalogs, and of course Edward Carlson's fine book on OSI BASIC. Mr. Carlson recently published an article which has led, indirectly, to a way of tapping into the Monitor and BASIC routines which input from the keyboard and write to the screen, ACIA, etc. Certainly, these techniques are well known to the more advanced C1P owners. Unfortunately, these people, with few exceptions, aren't writing for publication. So most information is being passed (slowly) by word of mouth or by club newsletters.

There are at least four points at which you can "break into" routines which are actively treating inputs or outputs. These are the Subroutines at $00BC and $0207, and the Jump vectors at $0218 and $021A. I'm sure there are more there for the finding. For this article, the Input vector at $0218 will be used.

Normally, this location holds a Jump Indirect to the routines starting at $FFBA in the monitor ROM which input a character from the keyboard or cassette. But it's no trick to poke a new address into this location, then do a little modifying of the routine. In this case, as shown in the listings, we are changing the vector from $FFBA to $00D8. This is near the end of zero page, which is not used by BASIC. Note, however, that it is used by the Monitor, so a Break to the Monitor followed by a Warm Start will require that the vector be reset and that the program be reentered.

The program is short and simple in operation. Essentially, it Goes sub to FFBA, which inputs a character. Next, the character is tested, and if it is a $7F, the RUBOUT key code, one of the more efficient machine language screen clear routines is effected. If it is any other character, this is skipped, and the program goes on about its business.

Note also that line 2010 in Listing II also POKEs the vector into location $0B, the USR vector. Thus, you will have both a single key screen clear by pressing the rubout and a programmable one by calling X = USR(X).

LIST 1

```
00D8 20 BA FF   JSR $FFBA     GET A CHARACTER
00DB C9 7F      CMP #$7F      IS IT A RUBOUT?
00DD D0 15      BNE $00F4     IF NO SKIP TO END
00DF 48         PHP           SAVE THE CHAR
00E0 A0 00      LDY #$00
00E2 A9 20      LDA #$20$     BLANK CHAR
00E4 99 00 D3   STA-Y         STORE BLANK AT 256
00E7 99 00 D2   STA-Y         LOCATIONS IN FOUR
00EA 99 00 D1   STA-Y         PAGES OF VIDEO RAM
00ED 99 00 D0   STA-Y
00F0 C8         INY           NEXT ADDRESS
00F1 D0 F1      BNE $00E4     PAGE DONE?
00F3 68         PLA           RETRIEVE CHAR
00F4 60         RTS           EXIT SUBROUTINE
```

List 2

```
47000 REM-ONE KEY SCREEN CLEAR
47010 POKE 11, 223:POKE 12, 0:POKE 536, 216:POKE
    537, 0
47020 FOR M = 216 TO 244:READ D:POKE M, D:NEXT
47030 DATA 32, 186, 255, 201, 127, 208, 21, 72, 160, 0
47040 DATA 169, 32, 153, 0, 211, 153, 0, 210, 153, 0, 209
47050 DATA 153, 0, 208, 200, 208, 241, 104, 96 ©
```