December 1977 $1.50

ohio scientific's

SMALL SYSTEMS JOURNAL VOLUME I NO.6

features page

This is your last issue!
Renew *now* for 1978
See inside back cover.

The magazine for 6502 computer enthusiasts!

74 MEGABYTE DISK HARDWARE REVIEW

As you may recall, we had an announcement of the 74-Megabyte disk drive in "Memory Technologies for Small Computers " on page 11, August 1977. As stated then, the 74-Megabyte Disk Drive CD-74 is a rack mount non-removable cartridge Winchester technology disk drive which is now being offered as a mass storage peripheral for Ohio Scientific floppy disk-based computers. Specifications are:

74 million bytes storage (unformated)
18,560 bytes per track
12 tracks per cylinder
339 cylinders
10ms single track seek
75ms maximum access
7.3 Megabits/sec. data transfer rate
7" x 17 3/4" x 23 1/2" rack mount
110VAC 5amps running
30amps starting
Drive, cable, interface for OSI Challenger and OS-74 operating system software
$6,000 F.O.B. Aurora, Ohio

The CD-74 is called a Winchester technology disk drive because it utilizes a non-removable disk storage medium and new head technology. Conventional large disk drives utilize a linear motor to position the heads on the magnetic surfaces and a head retractor to quickly pull the heads away from the disk in the case of a power failure. On all large disk drives, including the Winchester technology disk drives, the heads float a few thousandths of an inch off the surface of the disk by a natural film of air generated by the high rotational speed of the disk drive. Large disks rotate from 1500 to 3000 rpm (the speed of the CD-74 is 2940 rpms). The linear drive motor of conventional large disks is very expensive and these large drives are prone to crashing due to the fact that if the head ever touches the surface of the disk, both the head and the disk may become damaged. Therefore head retractors are employed to remove the head from the disk in the event of power failures, brownouts, or anything else that may cause the head to slow down or stop and land on the disk surface.

There is another whole set of problems involved in removable cartridges containing magnetic platters. The removal of these platters subjects the disk surfaces to potential contamination by allowing dust in the air to get onto the disk surface. Dust particles may then collide with the head. Furthermore, different disk platters may have slightly different tolerances which creates positioning problems. Special circuitry must be utilized to accommodate the variations in platters. Finally, there is considerable risk involved when a warm disk pack is placed in a cold disk drive, or vice versa. The information is packed so tightly on the disk that even expansions caused by temperature variations can cause data to be lost.

The Winchester technology disk drive eliminates all these problems by having a sealed chamber which does not allow contamination. It has non-changeable (i.e., non-removable) cartridges, eliminating problems of temperature and interchangeability of disk packs. Consequently, the CD-74 can reliably achieve a very high bit density on the disk with relatively inexpensive hardware. The Winchester technology drives eliminate the power cycling, or power failure head crashes by deliberately landing the heads on the disk as the disk spins down, or loses power. This is accomplished by having specially designed heads and a specially defined landing area on the disk. When the disk is not rotating, the heads are at rest on the disk surface. As the disk platters come up to speed, the heads lift off by their natural air cushion. A permanent magnetic actuator insures that the heads will retract to their normal rest position whenever there is a power failure or the disk spin loses rotational speed. However, since the disk is in a totally sealed environment, and both the disk and the heads are designed for landing, occasional contact between the heads and the disk surfaces does not cause damage.

Thus the Winchester technology disk drive incorporates several new features and was designed to eliminate most, if not all the major problems that have traditionally plagued big disk drives, and which have driven up their cost. A typical 74 million-byte disk drive sells for $30,000 to $50,000, not including a maintenance contract which may run to several thousand dollars annually.

Let us examine the hardware of the CD-74. First consider Diagram 1. We will proceed from the disk drive to the computer. The CD-74 disk has four aluminum disk platters about twelve inches in diameter that rotate at about 3000 rpms. There are thirteen heads mounted on four arms connected to a rotary positioner which selects the individual tracks on the disk (Diagram 1). Twelve of these heads operate on the six data surfaces, and the thirteenth head is a read-only head which operates on the Servo surface. The heads are positioned by an extremely fast magnetic rotary positioner that must operate over a range of less than 45 degrees. The positioner has continuous motion, that is, it is not a stepping motor. However, the disk is set up for 339 discrete positions of the heads. Each position is called a cylinder, and corresponds to twelve data tracks and one Servo track. The user specifies the data tracks by selecting one of the twelve heads. Each of the 339 cylinders on the CD-74 has twelve tracks. This indicates, for instance, that twelve individual tracks of data are properly positioned under the heads at any given time, so that up to 18,560 bytes x 12=222,720 bytes are accessible without the heads being repositioned. The use of two heads for each surface limits the travel of the rotary positioner and enhances the positioning from cylinder to cylinder.

One of the major circuit boards in the disk drive, the internal mechanical control logic board, takes commands from the computer specifying the cylinder address and combines that with the information obtained by the Servo head to actuate the rotary positioner to select the proper cylinder. A read/write circuitry board takes the TTL-level signals in from the disk controller and uses this to write data onto the disk and reads data from the disk, presenting it back to the controller as TTL-level signals. This board also operates the Servo head and provides information from the Servo head to the rest of the system. The Servo head provides a Servo clock signal, a 14.66MHz signal derived directly from data on the Servo track. This assures the controller that operations will always be perfectly in sync with the disk, even if the disk is rotating a little off speed (since the signal is derived directly from the rotating disk). If the disk rotation varied by five percent, for example, the Servo clock would be off its median frequency likewise by five percent, and all operations would automatically be faster or slower to accommodate for the speed variation. Another signal provided by the Servo head is an index pulse. This index pulse is an absolute marker in the rotation of the disk and occurs once each rotation. It is used to arbitrarily specify the beginning of a track on the disk, and is simply a reference point. The third type of information it provides is information that verifies which cylinder the positioner is on. When the disk drive gets a command to move to a particular cylinder, the drive attempts to do so, and can verify by the Servo head that it is properly on cylinder. If not, it goes through a search routine to reseek that cylinder.

The internal logic also controls a bank of analog switches which selects one of twelve data heads. These, of course, specify the individual track of twelve possible tracks on each cylinder. The disk drive itself also includes power supplies and a complete fault monitoring system which warns the computer of minor faults and automatically shuts down in the case of major faults, preventing the loss of data in a fault situation.

The disk controller consists of three Ohio Scientific boards: a 525 dual-port RAM Board, a 590 Disk Controller Board, and a 592 Interface Board. The system also requires a small backplane board with two or more slots to connect the high-speed data channel or memory transfer bus to the disk controller and the dual-port RAM memory. The Model 592 Interface Board simply provides interfacing between the TTL-level signals on the internal boards in the disk drive and converts these to transmission line signals so that the computer can be some distance from the disk. It uses open collector drivers for low-speed

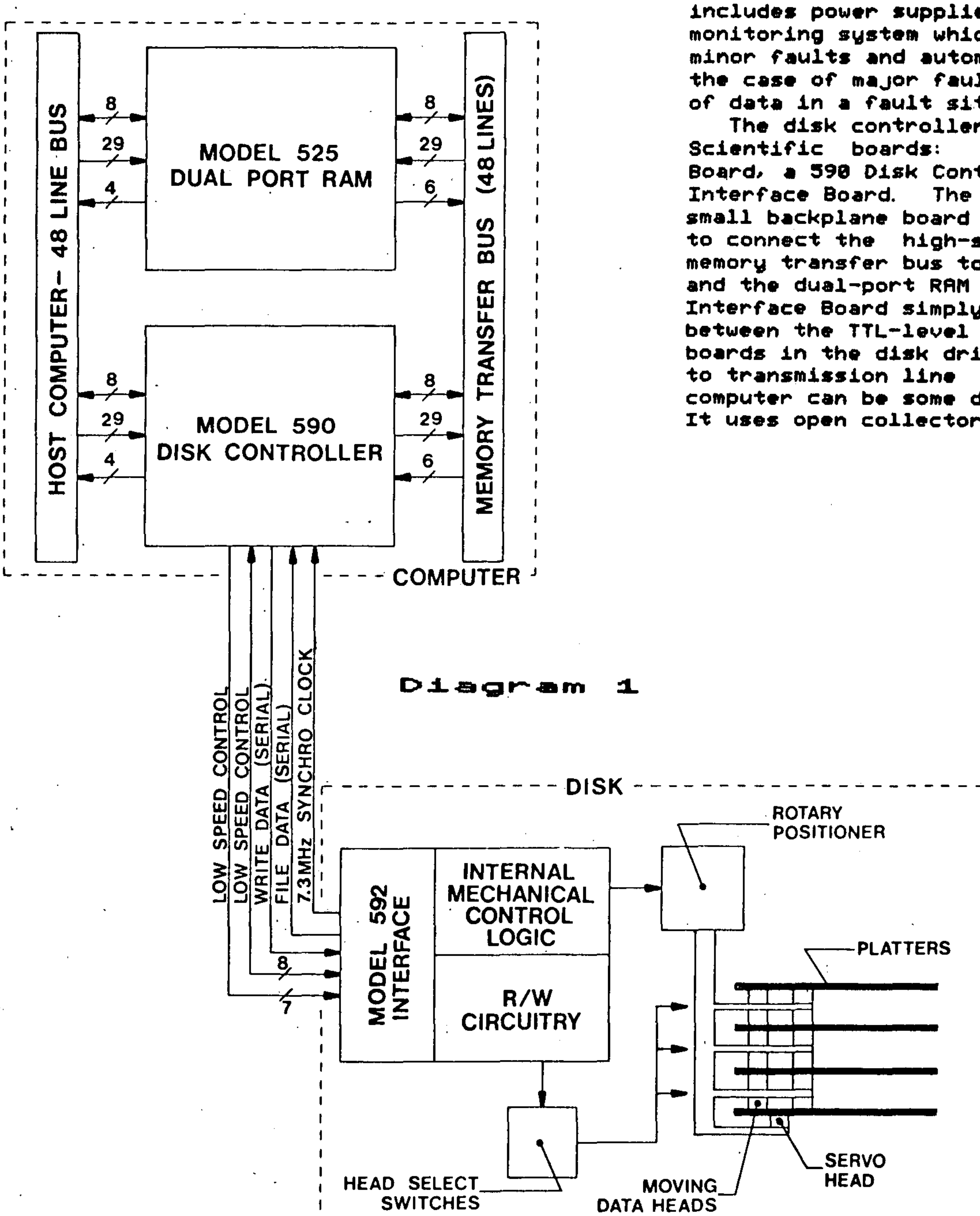


Diagram 1

signals and differential drivers and receivers for high-speed signals. This assures high noise immunity on the transmission lines between the 590 Disk Controller and the disk. The 592B Boards also have circuitry to select one of four 74-Megabyte disk drives through a two-bit binary selection code. The 590 Disk Controller Board is the "brains" of the disk interface. It connects to the computer by the standard 48-line bus connection and to the high speed data channel or memory transfer bus through a second 48-line bus at the end of the PC board. It then connects to the 592 Interface in the disk drive via two ribbon cables and six twisted pair wire sets. The 590 Board performs all disk control functions and all data transfers between the disk drive and the 525 dual port Memory Board.

Let us now consider the block diagram of the Model 590 Disk Controller (Diagram 2). This diagram shows the host computer's 48-line bus, 4K dual port RAM plugged into both 48-line busses, and a 48-line memory transfer bus. At the right it shows the connections to the 592 Disk Controller. The 590 Board contains conventional data buffers and register decode circuitry on the computer bus. This circuitry selects several internal read/write registers on the 590 and 592 Boards and on the internal circuit boards in the disk drives themselves. These signals are routed directly to the registers on the 590 Board, through buffers out to the transmission lines to the disk drive. Eight-bit bi-directional data is fed to all internal registers on the 590 Board through a buffer and on to transmission lines to the disk drive. The Model 590 controls the state of the 525 dual port RAM Board by a single signal line which specifies whether the 525 Memory Board will listen to the computer's bus or the memory transfer bus. When the 525 is commanded to listen to the computer's bus, it acts as a normal memory board. That is, the computer can read and write into the 525 Memory Board, as if it were any memory board in the system. When this line is switched, the dual port memory board is disconnected from the computer's bus and becomes active on the memory transfer bus. Both of these busses have identical interface and timing specifications so that the 525 dual port Memory Board can be used to tie two computers together, for instance. The 590 Board then provides a memory read/write circuit equivalent to a DMA circuit, but not the same, in the very important sense that the memory transfers to and from the 525 Board via the 590 Board do not in any way interfere with the computer's normal operation because they can act concurrently or simultaneously with computer memory operations, whereas a DMA operation cannot. So the 590 Board has data buffers, address counters and circuitry which allow it to simulate a standard 48-line bus of the computer for connection to the memory transfer bus. The data buffers on this memory transfer bus are fed to serial/parallel and parallel/serial converters which provide data to the disk drive in sequential-serial form and receive data from the disk drive in sequential-serial form. The data rate to or from the disk is 7.33 million bits/sec., or 916,000 bytes/sec. This independent data channel was designed into the computer system because the data rate transfers are too fast for the micro to handle under program control, and the blocks of data being transfered are far too large to allow the microprocessor to time-out during the transfers. In other words, it would be totally unacceptable for the microprocessor to have to be paralyzed during disk transfers. In this scheme, with a separate data channel, the microprocessor can function completely normally during disk transfers on other tasks and is completely free to service interrupts. This is very important in a multi-user distributed processor system where this micro would be the host or main computer.

The very important part of the 590 disk control is two large registers: Start-of-Transfer and End-of-Transfer. These registers are loaded by the computer with the track byte address of the start of transfer and the track byte address of the end of transfer. Recall that each track can store up to 18,560 bytes. These two registers then specify the first byte that will be transfered to and from the disk up to the last byte to be transfered to and from the disk. These registers can be set for anything between 0 and 18,560 bytes, so that the hardware is capable of transfering from one byte to the entire track, or any continuous portion in between. These transfer control registers feed write-control logic and read-control logic, which in turn provide commands to the disk drive, the address circuitry for the dual port RAM board, and parallel/serial and serial/parallel converters. Altogether these perform disk-read and disk-write operaions.

Let us now examine how the entire system performs.

Consider first a write operation. The software must specify a cylinder address between 0 and 339 and head address between 0 and 11 which names the individual track on the disk where the write operation will occur. There are 4,068 possible tracks. Then the software must load the Start-of-Transfer and End-of-Transfer Registers with the starting address on the track of the write, and the ending address on the track of the write. There are 18,560 bytes on the track, and theoretically any continuous block from one byte to the end of the track can be operated on. Usually a pre-specified sector format is used which specifies essential boundaries and lengths. In OS-74 there are five sectors per track, with a total of 3584 bytes per sector, a sector being the smallest transfer possible under this particular software package. In this case, therefore, the software will specify an individual sector and thus load the Start-of-Transfer with the beginning of that sector and the End-of-Transfer with end of that sector.

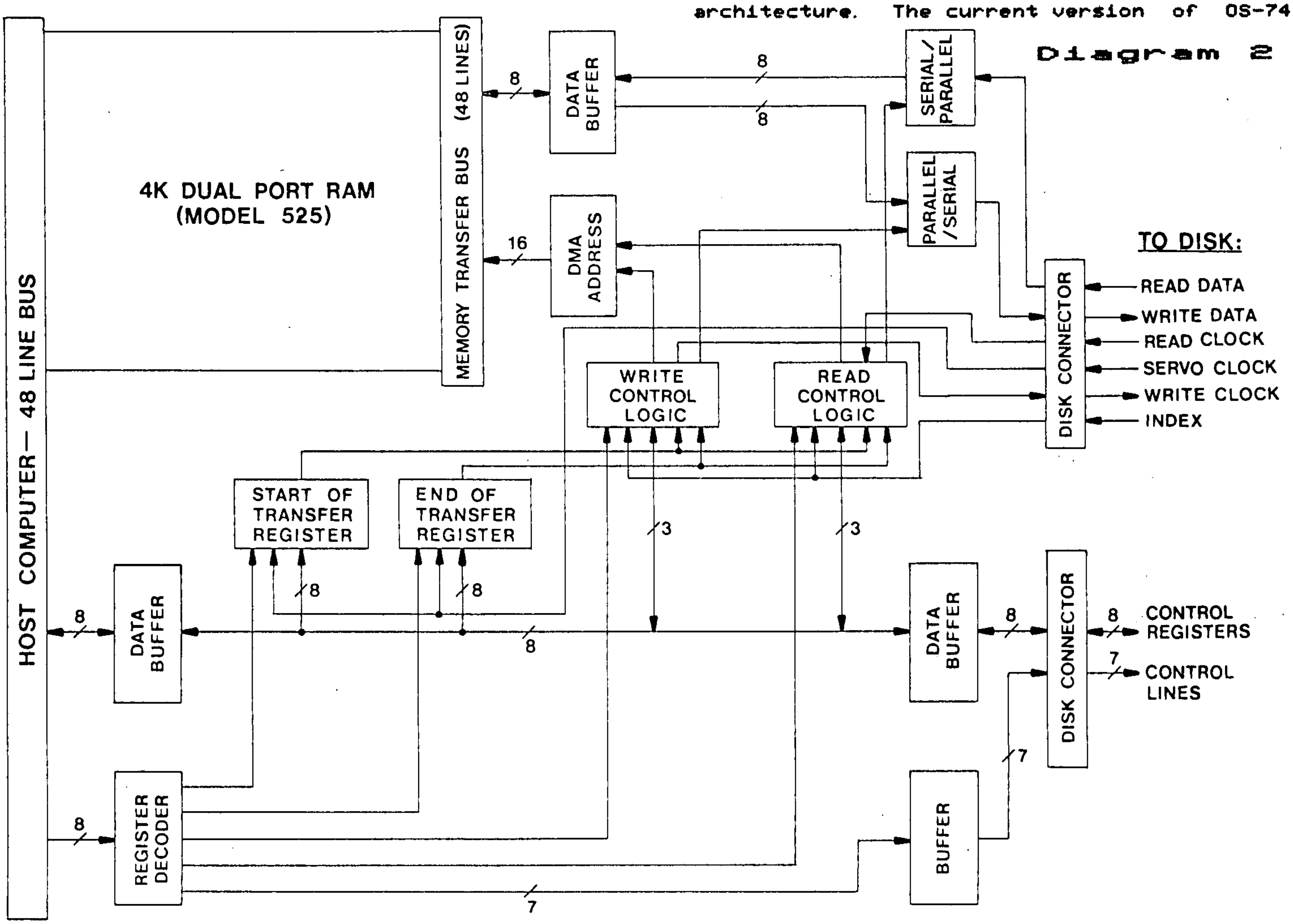
The software must be sure that the data to be written has been properly placed into the 525 Memory Board. This memory board is located at address E000 up and requires only 4K of RAM since the maximum transfer under this particular software package is under 4000 bytes. With this package, therefore the sector to be written into the disk is written into location E000 up on the 525 dual port Board. Once all this is set up, the software simply commands the disk controller to write via the disk control register. From there on, the disk write operation is performed by the controller. When the controller is done, it sets a status bit which can be polled at any time by the micro indicating "not busy." All operations except for actually writing into the 525 Memory Board initially are simply register loads with control data. Thus the disk drive can be operated virtually without any software at all. It is, for instance, possible to operate the disk drive totally in read and write modes by simply using the Extended Monitor to move blocks of data into the 525 buffer, and the Extended Monitor's memory modify commands to load the registers with proper information.

Read operations are conducted in a nearly identical fashion. The user specifies cylinder address, head address, Start-of-Transfer, and End-of-Transfer. He then commands the controller to read, and the controller goes out and reads that block of data into the 525 dual port Memory Board starting at E000 up. The disk controller then reports when it is done, and the computer can simply go to location E000 up for the data that was on the disk. This operation can be easily done with the Extended Monitor as well as with a software package.

This article is not intended to cover the software for the CD-74, but it has been necessary to discuss it to some extent in conjunction with the hardwre discussion. We will have another article in the journal outlining the details of OS-74 which is the operating system for use with the CD-74 disk drive. Some of the features of the operating system as mentioned above are: 3584-byte sectors of 14 pages; five sectors per track on 74-Megabyte; one sector per track on floppy disk; one sector per record on nine-track reel-to-reel mag tape, which can be used as a larger backup storage device to the 74-Megabyte disk drive. The drivers of OS-74 are capable of supporting an individual disk file of up to 74 million bytes with continuous addressing of entries within that disk file. The operating system is a full name file system for programs and data, and fully supports floppy disk drives, and optionally magnetic tape as backup for off-line storage for the 74-Megabyte disk.

The 590-592 controller system has been designed for expansion and even higher levels of performance in the future. The system is capable of handling up to four disk drives for a total of about 300 Megabytes on line. The use of the 525 dual port Memory Board and high-speed data channel or memory transfer bus allows a lot of flexibility in functional architecture. The current version of OS-74

Diagram 2



requires only a 4K buffer, but the hardware is capable of handling much larger memory buffers for larger transfers if desired. Furthermore, buffers can be paged, so that several dual port memory buffers can be connected to the memory transfer bus and paged by an external control on the CPU board, allowing individual users in a multiple user system to have their own unique buffer if desired. However, this is not at all necessary for normal operation in a multiple user environment.

A high-performance accessory currently in the design stage is a real-time string search board. This board will plug into the computer and utilize some signals that are on the transmission line between the 590 and 592. It will be able to perform real-time string searches on the data coming off the disk drive. The user will be able to pre-load the board with a string of up to 64 bytes and go through a disk search operation, looking for the target string on the disk. This board will be able to perform the search at the 7.3 Megabit/sec. data rate of the disk drive, and will report the disk address of the beginning of the string when it is found. This board has great potential in any large data file situation, because it will greatly reduce the requirements for advanced organization of disk files for quick access. For instance, it will allow simple access of sequential files to perform as well as multi-level index files in certain situations, thus relieving the task of the programmer while providing high performance computing for business applications. This accessory board will be simply a plug-in option to existing 74-Megabyte disk drives.

Employment opportunities

Ohio Scientific is greatly expanding its facilities by the addition of a new building in Aurora, Ohio, about eleven miles from our main offices in Hiram. This building has 7200 square feet of offices and 46,000 square feet of production space. This will bring our total space up to over 70,000 square feet. The new building is in an excellent location in the Cleveland area. It is about four minutes from an interchange on the Ohio Turnpike (Interstate 80) and easily reached from anywhere in the Cleveland and Akron areas via I-271, I-480, I-76, and I-77. To effectively utilize this space we need to hire many new people. There are positions open for technical personnel and virtually all areas of computer development, manufacturing and administration.

We need several people in each of the following areas. In all cases below, a good working knowledge of Ohio Scientific equipment will be an invaluable asset in applying for these jobs.

COMPUTER SALES

This job involves dealing with prospective purchasers and dealers via telephone, mail, and occasional visits to computer dealers. Applicants should have a good knowledge of computer systems and it will be extremely desirable to have some hands-on experience or ownership of an OSI system. Applicants must also have several years experience in sales to either consumers or the professional marketplace.

PROGRAMMERS

Ohio Scientific needs several programmers at all levels, ranging from multi-user and distributed processing time share systems with big disk drives down to the development of an extensive library of applications programs for the home in BASIC. Extensive programming experience in BASIC is a must. Knowledge of assembler and machine code programming is desirable, and any hands-on experience with a 6502 microprocessor is a definite plus in applying for a position in programming.

CUSTOMER SERVICE

These positions include working with customers and dealers after delivery of computer systems. They involve diagnosing possible problems and instruction in the use of the systems. Applicants should have experience in work with computer hardware (either in building kits or troubleshooting equipment), some knowledge of software and computer operations, and sales or other direct contact with the public. Again, experience with Ohio Scientific computer systems would be a most valuable asset in applying for these jobs.

SYSTEM TECHNICIANS

Ohio Scientific needs many people in this area. Responsibilities are in both the troubleshooting of circuit boards and the final check-out and testing of boards and systems. Work can range anywhere from our Challenger IIP to 300-Megabyte distributed processing systems. Several years experience as an electronics technician or in field experience as a repairer of electronics equipment is desirable along with an interest in computing, including hands-on experience in microcomputers. Knowledge of software and working experience with Ohio Scientific computers will be valuable assets.

All applicants should send their resumes to the attention of the Personnel Director c/o Ohio Scientific's Small Systems Journal, Box 36, Hiram OH 44234. Resumes should include salary requirements and a possible time frame for relocating to northeastern Ohio. Applications should be submitted soon since most positions are open immediately, as of 1 January 1978.

Article sponsorship program

Ohio Scientific has not had many articles in the major magazines in the past. This is because we have not actively sponsored people in writing articles about our equipment as most other manufacturers do. We are now interested in seeing articles in major magazines which make use of Ohio Scientific equipment and software, and are introducing this program to assist independent users in submitting articles for publication in the major small computer magazines. We are doing this by providing technical assistance and photographs to readers who demonstrate their interest in submitting an article to a major publication. We will also offer a credit of up to $50 per printed page towards Ohio Scientific's hardware or software for articles which actually do get published in these magazines.

An article which qualifies for this program must utilize Ohio Scientific computer equipment, as opposed to that of some other brand. An article may focus on Ohio Scientific equipment specifically or on a project, application or software which makes use of Ohio Scientific products. The articles should make adequate mention of Ohio Scientific products, and be generally positive on Ohio Scientific. However it certainly need not be an advertisement per se. Articles may be on your system, on software you have written, any hardware project you have worked on, or applications situations, such as a business package that you have developed. Or they can be independent discussions of the features of Ohio Scientific products. To qualify, you must first submit a rough manuscript to us (so that we know you are serious). Then Don Muchow of our advertising department will provide you with any additional information that you may need, and can possibly provide you with 8"x10" glossy pictures if appropriate. We can also recommend your article to magazine editors.

The first article submitted should have a very high acceptance potential because most magazine editors have been asking us for articles, and we have not had the time to supply them. If your article is actually accepted and printed, you will be issued a credit upon its actual publication, provided that you originally submitted a manuscript to us, and that the article incorporates Ohio Scientific products. The credit you will receive may be applied towards any hardware or software manufactured by Ohio Scientific for a period of up to six months after the article appears.

There are two credit plans, as shown below:

	Magazine	Aug 1978 or earlier	Sept--Dec 1978
I.	BYTE Kilobaud Interface Age	$50/per printed page	$25/per printed page
II.	Creative Computing ROM Personal Computing Computer	$25/per printed page	$25/per printed page

Plan II will be applied to an article appearing in any other national technical or semi-technical magazine with a monthly circulation exceeding 20,000. Note that all these magazines pay for the articles they print, so that you could realize as much as $100 per page total on articles for publication.

Please keep in mind that we must receive a rough manuscript from you on your article before we can assist you with additional technical information and pictures. This is necessary because many people start to write articles and never get around to finishing them. Therefore we must have evidence of your sincerity before we can invest time and effort in your project. Also keep in mind that the editorial copy is finalized two to three months before a magazine is published, so now is not too early to get started on an article, particularly if you wish to meet the deadline for the August issues of BYTE, Kilobaud, or Interface Age.

1K Corner

The general trend of our products and our customers' interests seems to be on BASIC, as expected, so that starting in the January-February 1978 issue, 1K Corner will be eliminated. It will be replaced by a new department, "Quickies." A Quickie is a small BASIC program which is short enough to be keyed in by hand and has some educational or practical value. We will try to have at least one Quickie in each issue of the journal next year, along with explanations of clever procedures that may be included.

The first program is a prime number generator. It generates all prime numbers between 1 and the integer limit of your computer. Although this program is quick to enter, it is certainly not quick running. It will take quite a while to list all the prime numbers from 1 to 10,000, for instance. The variable A is the number of the prime and the variable Y is the prime number itself, such that A=2 when Y=3, A=3 when Y=5, A=4 when Y=7, etc. The program operates by checking each number for divisibility by numbers up to its square root. If the number is found not to be evenly divisible by a number less than or equal to its square root, it is prime. This program could be easily reduced to three or four lines by stripping out the labels and placing multiple statements on a line.

```
10 PRINT "PRIME NUMBER"
11 PRINT "GENERATOR"
13 Y=2
15 A=1
17 GOTO 80
18 X=1
20 X=X+1
50 Z=INT(Y/X)
60 IF INT(Z*X)=Y GOTO 85
70 IF X*X>Y GOTO 80
75 GOTO 20
80 PRINT A,Y
82 A=A+1
85 Y=Y+1
90 GOTO 18
100 END
```

Shoot the Gluck

"Shoot the Gluck" is an Ohio Scientific assembly program written for a 12K Challenger with video at $D000 and keyboard input DF01. It is a variation of a game composed by Gary Miller of Desert Data Computer Store of Tucson AZ. A cannon is depicted moving from side to side along the bottom of the screen while a gluck flies near the top of the screen. The depression of any key causes the cannon to fire a shot at the gluck. The original game froze bird and cannon as the shot traveled upward.

A few modifications have been made so that now everything stays in motion. Multiple shots are permitted. These improvements can be traced to the use of event-paced multi-tasking. That is, each task uses the processor for as long as it wishes, then relinquishes control to the next task in the queue. This is in opposition to time-paced multi-tasking where a task uses the processor until a clock runs out and signals an interrupt. Lines 70-790 implement multi-tasking of up to twelve tasks (controlled by the values of lines 330 and 440--change these to get a different number of tasks). A section of $10 bytes of the stack is reserved for each task so that each task has a unique section of the stack. The S registers of each task are saved at QUEUE in lines 770-780. Be sure to enlarge the queue and the endpoints at lines 780,110 & 440 when you add more tasks.

Calling QSTART at line 70 ($2600) starts the whole show, clearing the queue, allocating a place for the master task in the queue, and executing the address at START, which starts the ball rolling. To spawn a new task, call SPAWN at line 440. The next two bytes are the starting address for the new task. See lines 2080-2130 ($27A3 to $27B1) for an example. To relinquish control of the processor, call NEXT at line 200. To terminate, a task calls DONE at line 180.

Now to the game. It starts at $2600, which clears the queue and calls the main task, STARTL at line 1960 ($278C). It clears the screen, then initiates three tasks, TIME, GLUCK, and CANNON. GLUCK moves the gluck around the top of the screen ($26AA to $26D6). CANNON moves the cannon along the bottom of the screen. The top of it is CANROW and CANPOS, while its left and right extremes are at $26F7 and $2703. TIME makes certain that each character moves at a reasonable speed by stopping time whenever it is given control of the processor. The delay constant is at $2694. The closer it is to 0, the slower the gluck and cannon will move.

Meanwhile the main routine monitors the keyboard at $27B5. If a key is depressed, it spawns a SHOT task whose main task is to find the top of the cannon and move an asterisk straight up, checking to see whether it hits the gluck. The position of the asterisk is kept on the stack, so that more than one task can use the code reserved for SHOT ($2734 to $2789). This type of code used is called re-entrant, since more than one task can use it, and the data it uses is separate from the code. If the asterisk misses, it terminates itself ($2747 to $275C), or pushes the new position of the asterisk onto the stack after first putting a new asterisk on the screen ($277C to $278B). If it hits the gluck, it puts out a message, waits for a key to be pressed, and restarts the show ($2766 to $277B).

Gerald Owens
S. U. P. O Box 9038
Tucson AZ 85720

```
.A

  10 0000              ;
  20 0000              ; QUEUE MANAGER
  30 0000              ;
  40 0000              BYTE = $FE
  50 2600              *=$2600
  60 2600              ;
  70 2600 A900         QSTART LDA #0
  80 2602 A200          LDX #0
  90 2604 9D8326       QLOOP STA QPOINT,X
 100 2607 E8            INX
 110 2608 E00D          CPX #$D
 120 260A D0F8          BNE QLOOP
 130 260C A2C0          LDX #$C0
 140 260E 9A            TXS
 150 260F 8E8426        STX QUEUE
 160 2612 6CD726        JMP (START)
 170 2615              ;
 180 2615 A200         DONE LDX #0
 190 2617 9A            TXS
 200 2618 BA           NEXT TSX
 210 2619 8A            TXA
 220 261A AE8326        LDX QPOINT
 230 261D 9D8426        STA QUEUE,X
 240 2620 CA           NEXL1 DEX
 250 2621 300B          BMI NEXL2
 260 2623 BD8426        LDA QUEUE,X
 270 2626 F0F8          BEQ NEXL1
 280 2628 8E8326        STX QPOINT
 290 262B AA            TAX
 300 262C 9A            TXS
 310 262D 60            RTS
 320 262E              BEX
 330 262E A20C         NEXL2  LDX #$C
 340 2630 CA           NEXL3  DEX
 350 2631 300B          BMI QEMPTY
 360 2633 BD8426        LDA QUEUE,X
 370 2636 F0F8          BEQ NEXL3
 380 2638 8E8326        STX QPOINT
 390 263B AA            TAX
 400 263C 9A            TXS
 410 263D 60            RTS
 420 263E 4C00FE       QEMPTY JMP $FE00
 430 2641              ;
 440 2641 A20C         SPAWN LDX #$C
 450 2643 A90E          LDA #$E
 460 2645 CA           SPL1 DEX
 470 2646 300B          BMI QFULL
 480 2648 BC8426        LDY QUEUE,X
 490 264B F009          BEQ SPFND
 500 264D 18            CLC
 510 264E 6910         ADC #$10
 520 2650 4C4526       JMP SPL1
 530 2653 4C00FE       QFULL JMP $FE00
 540 2656 9D8426       SPFND STA QUEUE,X
 550 2659 A8            TAY
 560 265A 18            CLC
 570 265B 68            PLA
 580 265C 6901          ADC #1
 590 265E 85FE          STA BYTE
 600 2660 68            PLA
 610 2661 6900          ADC #0
 620 2663 85FF          STA BYTE+1
```

```
630 2665 A200            LDX #0
640 2667 A1FE            LDA (BYTE,X)
644 2669 18              CLC
645 266A 69FF            ADC #$FF
650 266C 990101          STA $101,Y
660 266F E6FE            INC BYTE
670 2671 D002            BNE SPL2
680 2673 E6FF            INC BYTE+1
690 2675 A1FE           SPL2 LDA (BYTE,X)
694 2677 69FF            ADC #$FF
700 2679 990201          STA $102,Y
710 267C A5FF            LDA BYTE+1
720 267E 48              PHA
730 267F A5FE            LDA BYTE
740 2681 48              PHA
750 2682 60              RTS
760 2683 00              QPOINT .BYTE 0
770 2684                QUEUE=*
780 2693                *=*+$F
790 2693                ;
800 2693 A0F8           TIME LDY #$F8
810 2695 A200            LDX #0
820 2697 2E00FE         TIMEL ROL $FE00
822 269A 2E00FE          ROL $FE00
828 269D E8              INX
830 269E D0F7            BNE TIMEL
835 26A0 C8              INY
837 26A1 D0F4            BNE TIMEL
840 26A3 201826          JSR NEXT
850 26A6 4C9326          JMP TIME
860 26A9                 GLROW =$D000
870 26A9 00             GLPOS .BYTE 0
880 26AA A980           GLUCK LDA #$80
890 26AC 8DA926          STA GLPOS
900 26AF AEA926         GLUCKL LDX GLPOS
910 26B2 A920            LDA #$20
920 26B4 9D00D0          STA GLROW,X
930 26B7 E8              INX
940 26B8 E0A0            CPX #$A0
950 26BA D002            BNE GLUCK1
960 26BC A280            LDX #$80
970 26BE 8EA926         GLUCK1 STX GLPOS
980 26C1 A93C            LDA #'<
990 26C3 9D00D0          STA GLROW,X
1000 26C6 201826         JSR NEXT
1010 26C9 AEA926         LDX GLPOS
1020 26CC A93E          LDA #'>
1030 26CE 9D00D0         STA GLROW,X
1035 26D1 201826         JSR NEXT
1040 26D4 4CAF26         JMP GLUCKL
1050 26D7 8C27           START .WORD STARTL
1140 26D9                ;
1150 26D9               CANROW =$D300
1160 26D9 00            CANPOS .BYTE 0
1170 26DA FF            CANDIR .BYTE $FF
1180 26DB A926          CANNON LDA #$26    06
1190 26DD 8DD926         STA CANPOS
1200 26E0 A901           LDA #1
1210 26E2 8DDA26         STA CANDIR
1220 26E5 A920          CANL LDA #$20
1230 26E7 AED926         LDX CANPOS
1240 26EA 9D00D3  0006   STA CANROW,X
1250 26ED 9D20D3  4006   STA CANROW+$20,X
1260 26F0 9D40D3  8006   STA CANROW+$40,X
1265 26F3 9D60D3  C0D6   STA CANROW+$60,X
1270 26F6 E026           CPX #$26    06
1280 26F8 D008           BNE CANL2
1290 26FA A901           LDA #1
1300 26FC 8DDA26         STA CANDIR
1310 26FF 4C0B27         JMP CANL3
1320 2702 E03B          CANL2 CPX #$3B    1B
1330 2704 D005           BNE CANL3
1340 2706 A9FF           LDA #$FF
1350 2708 8DDA26         STA CANDIR
1360 270B 18            CANL3 CLC
1370 270C 8A             TXA
1380 270D 6DDA26         ADC CANDIR
1390 2710 8DD926         STA CANPOS
1400 2713 AA             TAX
1410 2714 A948           LDA #'H
1420 2716 9D00D3  00D6   STA CANROW,X
1430 2719 9D20D3  40D6   STA CANROW+$20,X
1440 271C 9D40D3  80D6   STA CANROW+$40,X
1445 271F 9D60D3  C0D6   STA CANROW+$60,X
1450 2722 201826         JSR NEXT
1460 2725 201826         JSR NEXT
1470 2728 201826         JSR NEXT
1480 272B 201826         JSR NEXT
1490 272E 201826         JSR NEXT
1510 2731 4CE526         JMP CANL
1520 2734               ;
1530 2734 A9D3    D6    SHOT LDA #$D3
1540 2736 48             PHA
1550 2737 ADD926         LDA CANPOS
1560 273A 48             PHA
1570 273B 68            SHOTL PLA
1580 273C 85FE           STA BYTE
1590 273E 68             PLA
1600 273F 85FF           STA BYTE+1
1610 2741 A000           LDY #0
1620 2743 A920           LDA #$20
1630 2745 91FE           STA (BYTE),Y
1640 2747 38             SEC
1650 2748 A5FE           LDA BYTE
1660 274A E920    40     SBC #$20
1670 274C 85FE           STA BYTE
1680 274E A5FF           LDA BYTE+1
1690 2750 E900           SBC #0
1700 2752 85FF           STA BYTE+1
1710 2754 C9CF           CMP #$CF
1720 2756 D003           BNE SHOTL1
1730 2758 4C1526         JMP DONE
1740 275B C9D0          SHOTL1 CMP #$D0
1750 275D D01D           BNE SHOTL2
1760 275F A5FE           LDA BYTE
1770 2761 CDA926         CMP GLPOS
1780 2764 D016           BNE SHOTL2
1790 2766 B97127        SHOTLM LDA MES,Y
1795 2769 F00B           BEQ SHOTD
1800 276B 91FE           STA (BYTE),Y
1810 276D C8             INY
1820 276E 4C6627         JMP SHOTLM
1830 2771 41            MES .BYTE 'AWK!',0
1830 2772 57
1830 2773 4B
1830 2774 21
1830 2775 00
1840 2776 20EDFE        SHOTD JSR $FEED
1850 2779 4C0026         JMP QSTART
1860 277C A92A          SHOTL2 LDA #'*
1870 277E 91FE           STA (BYTE),Y
1880 2780 A5FF           LDA BYTE+1
1890 2782 48             PHA
1900 2783 A5FE           LDA BYTE
1910 2785 48             PHA
1920 2786 201826         JSR NEXT
1940 2789 4C3B27         JMP SHOTL
1950 278C               ;
1960 278C A9D0   ---->  STARTL LDA #$D0
1970 278E 85FF           STA BYTE+1
1980 2790 A000           LDY #$0
1990 2792 84FE           STY BYTE
2000 2794 A920          STL1 LDA #$20
2010 2796 91FE          STL2 STA (BYTE),Y
2020 2798 C8             INY
2030 2799 D0FB           BNE STL2
2040 279B E6FF           INC BYTE+1
2050 279D A5FF           LDA BYTE+1
2060 279F C9D4           CMP #$D4    D7
2070 27A1 D0F1           BNE STL1
```

```
2080 27A3 204126        JSR SPAWN
2090 27A6 9326          .WORD TIME
2100 27A8 204126        JSR SPAWN
2110 27AB AA26          .WORD GLUCK
2120 27AD 204126        JSR SPAWN
2130 27B0 DB26          .WORD CANNON
2140 27B2 201826   STL3 JSR NEXT
2150 27B5 AD01DF        LDA $DF01
2160 27B8 30F8          BMI STL3
2170 27BA 204126        JSR SPAWN
2180 27BD 3427          .WORD SHOT
2190 27BF 201826        JSR NEXT
2200 27C2 201826        JSR NEXT
2202 27C5 201826        JSR NEXT
2204 27C8 201826        JSR NEXT
2210 27CB 4CB227        JMP STL3
2220 27CE 00       ENDQ .BYTE 0
2230 2F00               *=$2F00
2240 2F00 0026          .WORD QSTART,ENDQ
2240 2F02 CE27
2250 2F04               .END
```

Announcing Ohio Scientific's

FLOPPY DISK USERS GROUP

Ohio Scientific is forming a users group for the non-profit redistribution of user contributed software on diskettes. To qualify for a one-year membership in the users group, you must present a BASIC program, generally over 1K characters, or a machine code program, generally over 256 bytes long on diskette to Ohio Scientific's Small Systems Journal for approval. Programs should be original, and may be of special interest or general interest. All documentation for the program should be provided in the Assembler's Editor form on diskette. This can be OS-65D's Editor, or the Word Processing Editor. This will allow us to easily standardize the documentation and also preclude the necessity of duplicating documentation on paper. Virtually all types of programs and useful subroutines will be considered.

Subroutines will be considered if accompanied by a demonstration mainline program which is sufficient to verify proper operation of the subroutine as documented. Machine code programs should be provided in source code whenever possible. Patches or modifications to existing OSI programs and operating systems should be accompanied by installation instructions.

Send your entries on a floppy disk to Ohio Scientific's Small Systems Journal, Box 36, Hiram OH 44234. Be sure to reinforce the diskette's package with cardboard to preclude any bending or damage during shipment. If your entry is not accepted for the users group, your diskette will be returned to you with suggestions on how your entry might be modified to qualify for the users group. If your entry is accepted, your diskette will be returned with a complete users group software library and acknowledgment of your membership including a users group membership number and order forms for additional users group diskettes if desired. When submitting your program please specify the diskette library number which you desire to be returned on your diskette if your program is accepted. Users group disks will be specified by the numbers UG-1, UG-2, etc. Each diskette will have at least 50,000 bytes of actual programs plus instructions. The diskettes will be put together from programs as they are submitted and accepted with the only organization being that the diskettes will contain programs from compatible operating systems. At the moment all systems in use are compatible with each other, but in the future additional operating systems which are not interchangeable will possibly be in existence.

To initiate the program, Ohio Scientific has put together a collection of programs we have accumulated over a period of time. We have specified this as Users Group No. 1 (UG-1). We will also immediately start putting together UG-2, which will consist of the first user contributions.

Every time one of your programs is accepted you will get your choice of diskette copies on the diskette that you provide the original program on. Members can order an unlimited number of additional copies of existing users group library diskettes for ten dollars each, which should just barely cover the cost of duplicating and handling. Our objective is to have at least ten users group disks filled by the end of 1978. Please get your programs in now to get the users group off to a good start. The first entries will, of course, more likely be accepted because they will be unique by definition. However as time goes on, programs which are similar in function to other programs already in the users group may not always be accepted. You can submit as many programs as you want on individual diskettes and can defer the return of the diskettes by specifying UG-2, UG-3, UG-4, etc. as they become available if you already have a large number of programs you would like to submit to the library.

Users Group available:

Users Group No. 1 (UG-1)--Programs authored by Ohio Scientific

6502 MACHINE CODE PROGRAMS

- Cycle Time Test
- 6502 Diagnostics
- Memory Checksum Utility
- Track Zero Writer
- Multiplication Subroutine
- Mini-CRT Routine (440)
- High-Speed Paper Tape Reader via PIA
- Hex Calculator
- Paper Tape to Line Printer Output
- String Search Routine
- Point, Incremental, Vector Plot for 440 Graphics
- Circle Plot for 440 Graphics
- and more.

Users Group No. 2 (UG-2)--User-contributed programs through February 1978 or sooner, as the diskette fills up. Contents of this diskette will be your contributions.

Terminal/Cassette DOS Input Routine

Although OS-65D provides a simple means to save BASIC and Assembler source code on cassette (via the 430 Board's output flag), there is no simple way to reload that source code through the operating system. The following cassette/terminal input routine eliminates this problem. Additionally it provides a convenient way to transfer program source code for users wishing to upgrade their cassette-based system to disk.

USING THE PROGRAM

After the code has been entered, it is operated by setting the input flag to 10 (hex) [POKE 8707,16]. Note that the break bit 7 is set for the 440 keyboard. For use in a serial system, set the flag to 90 (hex) [POKE 8707,144]. The routine recognizes a serial- or video-based system by examining PROM location FE02 (hex) and responds to terminal inputs accordingly. To input from cassette, the user types a control-Z whereby control returns to the system terminal.

In the program below, the code for starting and ending cassette input is stored in registers "CHRSET" and "CHRCLR", respectively. Any characters may be used and need not be identical to those of this program.

SAVING THE PROGRAM IN OPERATING SYSTEM

Note: Although the following procedure is not difficult, it should only be tried by persons thoroughly familiar with the operating system. Otherwise major portions of the system may be irrevocably damaged.

After assembling (or typing) code into memory, return to the operating system and follow these steps:

```
C 3300=04,1
C 3D00=05,1
```

This loads the affected portion of the DOS into upper memory. Now enter the Extended Monitor and move the routine into this memory, as follows:

```
RE
M 3C46=2146,21CB
```

Now set up the dispatch table:

```
3DB3 CE 4C <LF>
3DB4 22 21
```

Exit to DOS and save:

```
D
S04,1=3300/B
S05,1=3D00/3
```

The routine is now part of the disk operating system and is available every time the system is booted in.

ASSEM

```
0000          ;TERMINAL / CASSETTE DOS INPUT ROUTINE
0000          ;
0000           PROMID=$FE02
0000           INEXIT=$230B
0000           UART=$FB00
0000           ACIA=$FC00
0000           KEYS=$DF01
0000          ;
2146          *=$2146
2146          ;
2146 2CC821   ENT     BIT UFLG    CHK IF 430 CASSET INPUT
2149 303D             BMI FRMCST  YES
214B 2C02FE           BIT PROMID  NO,CHK 65A OR 65V
214E 7022             BVS TYPA   IS 65A
2150 A908     TYPV    LDA #8     IS 65V,GET CHR FRM 440 BRD PORT
2152 F00C             BEQ TV3
2154 F8               SED
2155 38               SEC
2156 A2C6     TV1     LDX #$C6
2158 CA       TV2     DEX
2159 D0FD             BNE TV2
215B E901             SBC #1
215D B0F7             BCS TV1
215F D8               CLD
2160 AD01DF   TV3     LDA KEYS    GET CHR
2163 30EB             BMI TYPV   NO CHR,TRY AGAIN
2165 2C01DF   TV4     BIT KEYS
2168 10FB             BPL TV4
216A          ;
216A CDC921   TSTCHR CMP CHRSET TST FOR CONTROL CHR
216D F00E             BEQ TYPCST SWITCH TO CASSET IN
216F 4C0B23   TCX     JMP INEXIT
2172          ;
2172 AD00FC   TYPA    LDA ACIA    GET SERIAL INPUT
2175 4A               LSR A
2176 90FA             BCC TYPA
2178 AD01FC           LDA ACIA+1
```

```
380 217B B0ED           BCS TSTCHR
390 217D          ;
400 217D AD06FB  TYPCST LDA UART+6 INTL UART
410 2180 A9FF           LDA #$FF
420 2182 8D05FB         STA UART+5
430 2185 8DC821         STA UFLG   SET CASSET FLAG
440 2188          ;
450 2188 2C02FE  FRMCST BIT PROMID CHK PROM TYPE
460 218B 700C           BVS FC1
470 218D AD01DF         LDA KEYS
480 2190 301E           BMI UCHR   NO KEY PRESSED
490 2192 2C01DF  FC0    BIT KEYS
500 2195 10FB           BPL FC0
510 2197 3009           BMI CSTOFF
520 2199 AD00FC  FC1    LDA ACIA
530 219C 4A             LSR A
540 219D 9011           BCC UCHR   NO KEY PRESSED
550 219F AD01FC         LDA ACIA+1
560 21A2 CDCA21  CSTOFF CMP CHRCLR  TST FOR CONTROL CHR
570 21A5 D009           BNE UCHR
580 21A7 A900           LDA #0
590 21A9 8DC821         STA UFLG   CLEAR FLAG
600 21AC A90D           LDA #$D   OUTPUT <CR>
610 21AE 10BF           BPL TCX
620 21B0          ;
630 21B0 AD05FB  UCHR   LDA UART+5 GET CHAR FROM UART
640 21B3 4A             LSR A
650 21B4 90D2           BCC FRMCST NO CHR, TRY AGAIN
660 21B6 AD03FB         LDA UART+3
670 21B9 297F           AND #$7F
680 21BB 8D07FB         STA UART+7
690 21BE C90D           CMP #$D    SKIP ALL CONTROL CHR EXCEPT <CR>
700 21C0 F0AD           BEQ TCX
710 21C2 C920           CMP #'
720 21C4 10A9           BPL TCX
730 21C6 30C0           BMI FRMCST
740 21C8          ;
750 21C8          ;TEST REGISTERS AND FLAG
760 21C8 00      UFLG   .BYTE 0
770 21C9 1A      CHRSET .BYTE $1A  ASCII SUB - CONTROL Z
780 21CA 1A      CHRCLR .BYTE $1A  ASCII SUB - CONTROL Z
790 21CB          ;
800 22B3          *=$22B3
810 22B3 4621           .WORD ENT SET I/O DIST
820 22B5                .END
```

Have you ordered these new diskette software packages yet?

Ohio Scientific's Word Processor WP-1 & WP-1A (featured in November 1977 issue). This is a complete word processor package capable of character and line editing as well as formatted output. It is suitable for any text manipulating with the computer. It is designed specifically for non-technical users such as secretaries in a normal office environment. The WP-1 Word Processor software package (two diskettes and manual) sells for $79. WP-1A--as above, with fully integrated 6502 Assembler costs $99.

OS-65D Version 2.0 with Nine-Digit BASIC. This is our new Nine-Digit BASIC, which is virtually compatible with our old Six-Digit BASIC. It gives you nine digits of precision and is highly debugged. To order, specify OS-65D Version 2.0 with Nine-Digit BASIC. The package is two diskettes and manuals for $49

New Release--Disk-Test Disk. This diskette provides a quick functional check of your computer system and disk drive. The system boots in automatically and runs a basic diagnostic test which checks the memory and CPU in an actual functional test in BASIC. It then exits that program and runs a disk diagnostic test where it continually reads and writes from two tracks in drive A. The software package keeps track of any non-fatal errors that could accumulate over a period of time. It provides an excellent functional and reliability test on the disk portion of the system. The disk-test disk is available for $10.

Bank Accounts

For those of you who have had enough of games with spaceships, airplanes and bombs, this month we are presenting two practical, down-to-earth programs that allow you to figure out your bank accounts, checking and savings, respectively. In either program, be prepared to give the computer data whenever it asks for it. In the savings account program, all data is in numeric. In the checking account program, you give the name of the person(s) to whom the check is made out. These programs do not offer a permanent record, but with certain modifications using various forms of data storage techniques, it can be made to do so. For your own convenience, it is suggested that you run this program on SAVE (by typing SAVE before RUN). This slows down the baud rate, enabling you to read the output at a reasonable speed.

CHECKBOOK ACCOUNT

Lines	Function
4	Dimension Statement
5-54	Introductory statements
54	Initial amount input
65-66	Input number of checks
71-93	Check listing instructions
91-125	Check input routine
128-155	Check input listing
158-195	Mistake check
196-400	Route for correcting mistakes

```
1 REM***ROBERT L.
2 REM***COPPEDGE
4 DIM A$(20),Y(20),Q(20)
5 PRINT"THIS PROGRAM FIGURES"
6 PRINT"OUT YOUR CHECKING"
7 PRINT"ACCOUNT.  TO DO"
8 PRINT"THIS, A NUMBER OF"
9 PRINT"THINGS ARE NEEDED, SUCH"
10 PRINT"AS THE NUMBER OF CHECKS"
11 PRINT"TO BE USED, THE TOTAL"
12 PRINT"AMMOUNT TO START WITH,"
13 PRINT"AND THE NAME OF THE "
14 PRINT"RECIPIENT OF THE CHECK,"
15 PRINT"AS WELL AS THE AMOUNT."
16 PRINT:PRINT:PRINT
34 PRINT"IF YOU HAVE MADE A"
35 PRINT"MISTAKE, AND YOU WOULD"
36 PRINT"LIKE TO CORRECT IT,"
37 PRINT"MERELY TYPE IN THE WORD"
38 PRINT"'HELP' WHEN I ASK"
39 PRINT"YOU FOR A CHECK'S NAME."
45 HELP=-6
50 PRINT"ENTER IN HOW MUCH YOU HAD"
53 PRINT"IN YOUR ACCOUNT TO START"
54 PRINT"WITH";:INPUT T
60 IF T=-6 THEN 160
65 PRINT"HOW MANY CHECKS DO YOU"
66 PRINT"WANT TO BE TOTALLED";:INPUT X
67 IF X=0 THEN 160
70 IF X=-6 THEN GOTO 160
71 PRINT:PRINT:PRINT
75 PRINT"ENTER THE NAME(S) OF THE"
76 PRINT"PERSON OR PERSONS TO WHOM"
77 PRINT"THE CHECKS ARE MADE.  IT'S"
78 PRINT"A GOOD IDEA TO LIST THEM"
80 PRINT"IN THE ORDER THAT THEY"
82 PRINT"WERE MADE."
90 PRINT"IF IT IS A DEPOSIT, THEN"
91 PRINT"TYPE IN 'XDEPOSIT',AND"
92 PRINT"THEN THE AMMOUNT OF"
93 PRINT"DEPOSIT."
94 IF X=0 THEN 160
95 FOR Z=1 TO X
100 PRINT"CHECK NUMBER ";:PRINT Z
101 PRINT"MADE OUT TO:"
105 INPUT A$(Z)
110 IF A$(Z)="HELP" THEN 160
115 PRINT"AMMOUNT";:INPUT Y(Z)
120 IF Y(Z)=-6 THEN 160
124 IF A$(Z)="XDEPOSIT" THEN Y(Z)=-Y(Z)
125 NEXT Z
128 L=T:PRINT:PRINT:PRINT
129 PRINT"ORIGINAL AMMOUNT:  ";:PRINTT
130 PRINT:PRINT"#--MADE TO"
131 PRINTTAB(9)"AMMOUNT--BALANCE"
133 PRINT:PRINT
135 FOR Z=1 TO X
140 PRINTZ;:PRINTA$(Z)
146 LET Q(Z)=ABS(Y(Z))
150 L=L-Y(Z)
153 PRINTTAB(10)Q(Z);:PRINTTAB(20)L:PRINT
155 NEXT Z
156 FOR Z=1 TO 5000
157 NEXT Z
158 PRINT:PRINT:PRINT
160 PRINT:PRINT:PRINT:PRINT:PRINT
161 PRINT"IF THERE ARE ANY MISTAKES"
162 PRINT"YOU WOULD LIKE TO CHANGE"
165 PRINT"TYPE IN ONE OF THE"
166 PRINT"FOLLOWING NUMBERS:"
170 PRINT"A '1' IF THE TOTAL IS"
171 PRINT"WRONG."
175 PRINT"A '2' IF ONE OF THE"
176 PRINT"CHECKS OR DEPOSITS ARE"
177 PRINT"WRONG."
180 PRINT"A '3' IF EVERYTHING'S OK."
195 INPUT R
196 IF R<1 THEN 165
197 IF R>3 THEN 165
199 PRINT:PRINT:PRINT
200 ON R GOTO 220,230,205,215
205 PRINT"OK.":GOTO 400
215 PRINT"TRY AGAIN":GOTO 175
220 PRINT"ENTER IN NEW TOTAL";:INPUT T:GOTO 128
230 PRINT"DO YOU HAVE MORE CHECKS"
231 PRINT"OR DEPOSITS TO ADD"
232 PRINT"(YES OR NO)";:INPUT N$
234 PRINT:PRINT:PRINT
235 IF N$="NO" THEN 250
240 PRINT"HOW MANY MORE";:INPUT W
245 X=X+W :GOTO 310
250 PRINT"DO YOU WANT TO EITHER"
251 PRINT"CHANGE OR ERASE ONE OF"
252 PRINT"THE CHECKS OR DEPOSITS";
255 INPUT N$:IF N$="YES" THEN 270
256 PRINT:PRINT:PRINT
260 GOTO 160
270 PRINT:PRINT:PRINT"WHICH CHECK # DO YOU WANT"
271 PRINT"TO CHANGE";:INPUT M
274 PRINT:PRINT:PRINT
275 PRINT"DO YOU WANT TO CHANGE OR"
276 PRINT"ERASE IT (1=CHANGE, "
277 PRINT"2=ERASE)";:INPUT K
278 PRINT:PRINT:PRINT
280 ON K GOTO 300,290,275
290 A$(M)="DELETED"
291 Y(M)=0
295 GOTO 128
300 PRINT"NEW CHECK NAME"
301 INPUT A$(M)
302 PRINT"NEW AMMOUNT";:INPUT Y(M)
```

```
303 PRINT:PRINT:PRINT
305 PRINT"ANY MORE CHECKS TO BE"
306 PRINT"REDONE(YES OR NO)";:INPUT B$
307 IF B$="NO" THEN 130
308 IF B$="YES" THEN 273
309 GOTO 130
310 FOR Z=X-(W-1) TO X
315 PRINT"CHECK #":PRINTZ
320 PRINT"MADE OUT TO:"
325 INPUT A$(Z)
326 IF A$(Z)="HELP" THEN 160
330 PRINT"AMMOUNT";:INPUT Y(Z)
335 IF Y(Z)=-6 THEN 160
340 NEXT Z
345 GOTO 128
400 END
OK
```

SAVINGS ACCOUNT

Lines	Function
8	Dimension Statement
10-22	Instructions
24	Interest rate input
26-32	No. of times/year compounded
34-36	Initial balance input
37-38	Initial balance date
39-51	Check above for mistakes
70-86	Withdraw/deposit instructions
101-130	Mistake check
131-134	Input today's date
136-309	Interest & new balance computation
1000	Data

```
1 REM***ROBERT L.
2 REM***COPPEDGE
8 DIM K(25),P(12),A(25),B(25),C(25),D(25)
10 PRINT:PRINT:PRINT"THIS PROGRAM IS AN AID"
12 PRINT"IN FIGURING OUT YOUR"
14 PRINT"SAVINGS ACCOUNT. TO DO"
16 PRINT"THIS SEVERAL THINGS ARE"
18 PRINT"NEEDED, SUCH AS:"
20 PRINT"(PLEASE TYPE IN WHEN"
22 PRINT"'?' COMES UP)"
24 PRINT:PRINT:PRINT"INTEREST RATE(IN %)";:INPUT X
26 PRINT:PRINT"HOW MANY TIMES IS"
28 PRINT"INTEREST COMPOUNDED"
30 PRINT"ANNUALLY(IF DAILY,"
32 PRINT"THEN TYPE IN 365)";:INPUTY
34 PRINT:PRINT:PRINT"ALL RIGHT, WHAT IS YOUR"
36 PRINT"BALANCE";:INPUT Z:PRINT:PRINT
37 PRINT"AND THAT IS AS OF(MONTH,"
38 PRINT"DAY,YR)";:INPUT U,V,W
39 PRINT:PRINT:PRINT"OKAY, YOUR #COMPUNDS/YR"
40 PRINT"IS";Y:PRINT:PRINT
42 PRINT"YOUR INTEREST RATE IS:";X
43 PRINT:PRINT:PRINT"AND YOUR INITIAL BALANCE"
46 PRINT"IS:";:PRINTZ
47 PRINT:PRINT:PRINT"AS OF ";U;"-";V;"-";W
48 PRINT:PRINT:PRINT"DO YOU WISH TO CHANGE"
50 PRINT"ANY OF THESE";:INPUT A$
51 IF A$="Y" OR A$="YES" THEN 24
53 PRINT:PRINT:PRINT:PRINT:FOR Q=1 TO 12:READP(Q):NEXT
70 PRINT"NOW, THEN: LIST YOUR"
72 PRINT"CHANGES IN YOUR ACCOUNT"
73 PRINT"IN THE ORDER THAT THEY":PRINT"WERE MADE!":PRINT:PRINT
74 PRINT"IF IT WAS A WITHDRAWAL,"
76 PRINT"THEN PUT A MINUS SIGN(-)"
78 PRINT"IN FRONT OF THE AMMOUNT"
80 PRINT"AND THEN I'LL ASK YOU"
82 PRINT"MADE. TYPE IN A 0(ZERO)"
84 PRINT"WHEN YOU ARE FINISHED."
86 PRINT:PRINT:PRINT
88 FOR F=1 TO 25
```

```
90 PRINT"ENTER AMMOUNT";:INPUT A(F)
92 IF A(F)=0 THEN 101
94 PRINT"ENTER DATE MADE";:INPUT B(F),C(F),D(F)
96 PRINT:PRINT:PRINT:NEXT F
101 PRINT:PRINT:PRINT"WOULD YOU LIKE TO"
104 PRINT"CHANGE ANYTHING, TYPE"
106 PRINT"IN A:":PRINT:PRINT
108 PRINT"1)MEANS YOU WANT TO"
110 PRINT"REWRITE ENTIRE DEPOSIT/"
112 PRINT"WITHDRAWAL SERIES"
114 PRINT:PRINT"2) MEANS ONLY A SPECIFIC":PRINT"ONE"
116 PRINT:PRINT"3)EVERYTHING'S OK."
118 PRINT:PRINT:PRINT"HOW IS IT";:INPUTE:ON E GOTO 88,120,130
120 PRINT"WHICH NUMBER DID YOU"
122 PRINT"WANT TO CHANGE";:INPUT J
124 PRINT"YOUR NEW AMMOUNT";:INPUT A(J)
126 PRINT"NEW DATE";:INPUT B(J),C(J),D(J)
128 PRINT"ANY OTHERS";:INPUT A$
129 IF A$="Y" OR A$="YES" THEN 120
130 PRINT:PRINT"OKAY."
131 PRINT:PRINT:PRINT
132 PRINT"OH, YES. PLEASE ENTER"
134 PRINT"TODAYS DATE.";:INPUT G,H,I
136 PRINT:PRINT:PRINT
140 A(F)=Z:B(F)=U:C(F)=V:D(F)=W
145 FOR J=1 TO F
150 IF I=D(J) THEN 200
151 IF I=D(J)+1 THEN 180
155 T(J)=(I-W-1)*365
156 GOTO 180
180 FOR L=B(J) TO 12:T(J)=T(J)+P(L):NEXT L
182 FOR L=1 TO G-1:T(J)=T(J)+P(L):NEXT L
183 IF G=1 THEN T(J)=T(J)-31
184 T(J)=T(J)+H-C(J)
186 GOTO 220
200 FOR L=B(J) TO G-1:T(J)=T(J)+P(L)
202 NEXT L
204 IF B(J)=G THEN T(J)=0
206 T(J)=T(J)+H-D(J)
220 NEXT J
300 FOR L=1 TO F
302 T(L)=T(L)/365
304 A(L)=A(L)*(1+X/Y/100)↑(Y*T(L))
306 M=M+A(L)
308 NEXT L
309 M=INT(M*100)/100
310 PRINT"YOUR BALANCE NOW TOTALS"
320 PRINT"$";M;" AS OF"
330 PRINTG;"-";H;"-";I
900 END
1000 DATA 31,28,31,30,31,30,31,31,30,31,30,31
OK
```

If you haven't filled out the questionnaire on page 15 of the October 1977 issue, please do so. We are very sincerely interested in your opinions and are trying to utilize the results of this questionnaire in planning. We have had quite a number of responses so far, but do not yet have nearly one hundred percent return on the questionnaire. It would be in your best interest as an owner of Ohio Scientific equipment to take this formal opportunity to voice your opinions on how we might improve our company and serve you better in the future.

Ohio Scientific's Small Systems Journal is issued bimonthly by Ohio Scientific, Inc., P.O. Box 36, Hiram OH 44234. The subscription rate is six dollars per six issues. Individual copies are $1.50. Published in Twinsburg OH by the Twinsburg Bulletin.

Vol. 1, No. 6 — December 1977
Editor-in-Chief — Gary Deckant
Production Manager — Don Muchow
Contributing Editors — Mike Cheiky, Bob Coppedge, Eric Davis, Rick Whitesel

***

Ohio Scientific 11679 Hayden Hiram, OH 44234
SMALL SYSTEMS JOURNAL

HIRAM OHIO

First Class Mail
U.S. POSTAGE
PAID
PERMIT 12

OHIO SCIENTIFIC
11679 HAYDEN STREET HIRAM, OHIO 44234