September 1977 $1.50

ohio scientific's SMALL SYSTEMS JOURNAL VOLUME 1 NO. 2

Volume 1 no 3.

features

page

departments

The magazine for 6502 computer enthusiasts!

Ohio Scientific advances the state-of-the-art of small computers.

From our inexpensive 8K BASIC in ROM Challenger IIP to our powerful triple processor Challenger III, Ohio Scientific offers a full range of products that are technologically superior to anything available on the market today.

Challenger IIP

Challenger IIP from Ohio Scientific is our unique personal computer with BASIC in ROM and 4K RAM for programs in BASIC.

Complete with audio cassette interface and a full computer keyboard, Challenger IIP can be connected to a home TV via an RF converter and it's ready to go.

Challenger IIP comes fully assembled and tested for only **$598.00.**

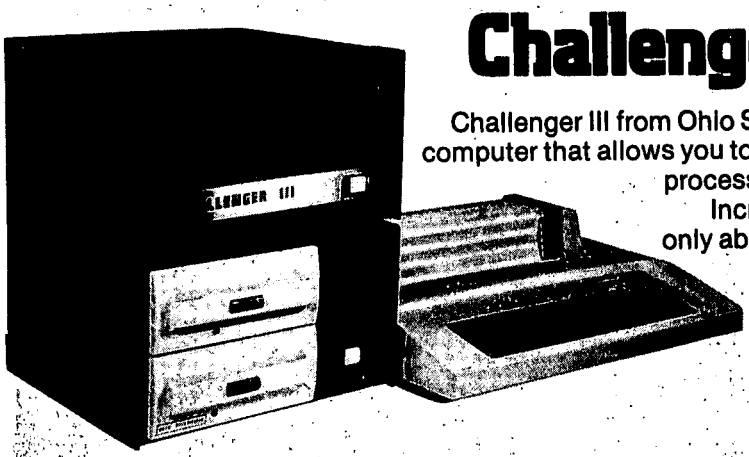
Challenger II

Challenger II from Ohio Scientific is a disk based computer capable of storing up to 500,000 bytes of information on an Ohio Scientific dual drive floppy disk.

Challenger II comes with 16K of RAM (the disk BASIC is automatically loaded into the computer so there is no need for ROM's) and our powerful Disk Operating System (DOS) which allows the computer to perform big computer functions like random access, sequential and index sequential files in BASIC, and I/O distributors which support multiple terminals and industry standard line printers.

And best of all a 16K Challenger II with serial interface, single drive floppy disk, (250,000 bytes) BASIC and DOS costs only **$1,964.00** fully assembled.

Challenger III



Challenger III from Ohio Scientific is the revolutionary, new triple processor computer that allows you to run programs written for the 6502A, 6800 and Z-80 processors.

Incredible as this is, a disk based Challenger III costs only about 10% more than conventional single processor microcomputers. A 32K Challenger III with a serial interface and a dual drive floppy disk assembled and tested costs **$3,481.00.**

OHIO SCIENTIFIC

11679 Hayden • Hiram, Ohio 44234

To order direct call 1-216-569-3241

For more information send for our Free, short form catalog, or send $1 for our 64 pg. Small Computing Buyers Guide.

Introduction

By now you will have undoubtedly noticed the change in format of the September issue of Ohio Scientific's Small Systems Journal. This month we are introducing our first edition of the Journal printed almost entirely on an OKI-DATA Model O-22 Line Printer. This demonstrates the upper and lower-case capability of the line printer as well as our soon-to-be-released word-processor package.

Our coverage in this issue includes a continuation of our article on memory devices for microcomputers. Our 560Z Expander Board is introduced as well, in an article which explains in detail its use in conjunction with several microprocessors. Several of our readers have been kind enough to submit their own programs and articles, whether at our request or at their own initiative. We have devoted appropriate space to these contributions. Our regular features have met with such favorable response that we are continuing them.

For those of you who are wondering what has been the reason for the delay in publishing this journal, you have only to glance through the full-line catalog for Fall 1977, which hopefully, you have already ordered. If not, there is an order form on page 18. In order to meet our immediate priority of rushing our catalog to you, we were forced to postponwe our September Journal. We will soon be back on schedule and are grateful that you were able to bear with us in the meantime.

Thank you again for your support and suggestions. We depend on you to let us know what you are looking for in terms of products and coverage. Continue to let us hear from you.

Ohio Scientific Small Systems Journal
Box 36
Hiram OH 44234

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

Vol. 1. No. 2 August 1977
Editor-in-Chief Gary Deckant
Production Manager Rob Spademan
Contributing Editors Mike Cheiky
Eric Davis
Marcel Meier

GET THE MOST OUT OF BASIC

PART TWO

FILES IN BASIC

Both 8K BASIC under OS-65D and the new ROM version of 8K BASIC are capable of files based on mass storage devices. 8K BASIC in ROM is capable of simple cassette-based sequential files. The BASIC for disk is capable of sequential, random access, indexed sequential, and other advanced file structures.

Let's first discuss why files based on mass storage devices are desirable. The first obvious reason is that we would like to store more data in the machine than our RAM memory will allow. Secondly, we would like to store information on a permanent basis so that if we enter it into the machine once it is always there. Thirdly, we would like to provide reference material or a library of data or information for programs to act on. Files are a necessity for applications such as small business programming. There are also applications in personal and home computing and educational computing which require the use of data files.

The simplest file from an organization and performance point of view is the sequential file. In this type of file, entries of variable length are simply made sequentially. To access any entry, the user must sequentially read through the file until he finds it. Sequential files are fine for applications such as mailing lists where one will normally want to output the file in a sequential manner. However, they have extremely long seek times when the user is looking for a particular piece of data. Manually operated audio cassette systems are capable only of sequential files because the recorder must be advanced, or read, at normal speed to get to the desired piece of data. For this reason, cassette-based files are very slow and not really practical for any business applications on the computer.

If the data in a file can be formated in some manner and put in some order such that the position of the data can be predicted by some mathematical equation, then a random access file can be used. Random access files are extremely fast because you can directly read the data that you desire. However, in many cases it is not feasible to provide a logical, calculable organization for your data file such as in business applications. The net result is that random access files are fast, but, do not lend themselves to many applications while sequential files are easy to use, but, very slow.

The index sequential file is a two-level file system merging the features of random access and sequential files. Index sequential files are most commonly used in applications such as business computing. An index sequential file is actually made up of two separate files: one file with the indexes, and one with the actual entries. An index sequential file works on the same principle as a standard library. The user has a catalog of available documents and the documents themselves. The catalog is the index, and the documents are the actual data file.

The index file is typically a short sequential file which has a key word entry followed by the index which points to the larger data base. This index is then used to randomly access the large data base. To clarify this, consider the example of a student report card record. Each student would have a group of entries (e.g., name, social security no., date of birth, etc.), followed by the classes he is enrolled in and his grades for those classes. The record for each student may contain thousands of bytes. Students may be placed in the file in any order as they are admitted to and withdrawn from the school. As a simple file, it would take an extremely long time to locate any student in the file because one would have to search through a large data base to find a student's name. If the file were set up as a random access file, the entire file would have to be reorganized every time there was a new enrollment or withdrawal of students. This operation may take several hours if the data base is particularly large.

The solution is an index sequential file. The index file simply contains the student's name and the index which points to the location of his actual file. The large data base contains the files. To access an individual student's records, the student's name is typed in, the index file is brought into memory, searched for the index, and the index is used to to pull in the student's file.

There are more advanced file systems which utilize the index sequential concept. For instance, it is possible to devise files which are both random access and sequentially addressable. This can be accomplished on an OSI computer system by preloading the file with a field of null characters. Then a random access file with variable length entries can be accessed sequentially if desired. There are techniques where files are inverted, or the data in files are placed in different orders to allow easy sorting and merging and accessing. However, the user would do well to master the concepts of the simple sequential, random access and index sequential file systems.

GUIDELINES FOR FILES WITH OSI BASIC

As mentioned above, simple sequential files are possible in ROM BASIC. Here are two routines that can be incorporated into any program to write an array out to a cassette and to read an array in from a cassette. These examples demonstrate fairly well the capability of the sequential file system in ROM BASIC. The reserved word SAVE switches the output from video display or serial port only to also include the audio cassette so that anything that goes out onto the terminal will also go out to the audio cassette. The reserved word LOAD switches input from being solely from the keyboard of the terminal to being from either keyboard or cassette. The first time the keyboard is actuated by the user after the LOAD command is executed, the input will revert solely

back to the keyboard.

In summary, the reserved word SAVE turns the cassette output on; the reserved word LOAD turns it off and turns cassette input on; and any intervention at the keyboard by the user shuts off cassette input. Data input and output to cassette is by simple PRINT and INPUT statements. It is important that the user remember to follow each data output by a carriage return so it can be accepted by an INPUT statement later. It is a good idea to prompt the user in cassette operation of these routines, as in lines 500, 550, and 560 below.

SIMPLE CASSETTE SEQUENTIAL FILES WITH ROM BASIC

Routine for storing an array

```
 500 PRINT "TURN CASSETTE RECORDER ON
RECORD."
 510 SAVE
 520 FOR X=1 TO I
 530 PRINT T(I)
 540 NEXT
 550 LOAD:PRINT"TURN RECORDER OFF."
 560 INPUT"TYPE ANY KEY TO CONT.";A$
```

ROUTINE FOR RECALLING THE ARRAY

```
 1000 PRINT"TURN CASSETTE RECORDER ON."
 1010 LOAD
 1020 FOR X=1 TO I
 1030 INPUT T(I)
 1040 NEXT
 1050 PRINT"TURN RECORDER OFF."
 1060 INPUT "TYPE ANY KEY TO CONT.";A$
```

OS-65D BASED FILES

OS-65D allows full user data files in 8K BASIC and machine language programs. The file system is totally open-ended, allowing sequential, random access, index sequential, and complex file structures, such as indirect command files, and multiple files of different length to be open at any given time. Because of OS-65D's generality in having variable sector lengths and variable numbers of files open at a time and total control of file operations, it is both extremely powerful and very difficult to implement. The difficulty in implementation is mainly due to the fact that there are virtually no error messages or error traps for operator errors because there are practically no illegal operations. In other words, since it is possible to have variable sector lengths and any number of files open simultaneously, it is impossible for the operating system to determine whether the user is attempting a legal or illegal operation since virtually any operation would be legal under some circumstances. For this reason, it is advised that the beginner with OS-65D confine himself to modifying existing file-based programs until he gets a good understanding of what is going on and refines his own personal programming procedures. Otherwise, he will be constantly plagued with system crashes and possibly the loss of important data on diskettes.

From here on, we will confine our discussion to data files in BASIC in OS-65D. A simplified picture of what is required of a data file in BASIC in OS-65D is that the BASIC program performs conventional INPUT and OUTPUT statements to a memory buffer. This buffer is periodically transferred to and from disk. It is possible to have any number of these memory buffers existing with different memory files present in them at any one time, limited only by the total amount of memory your system has. This discussion will limit itself to one- and two-buffer systems, however.

CREATING A FILE

The procedure to create a simple file is, first, to create a memory-resident file buffer. This is accomplished simply by allocating space for it at the top of memory when BASIC asks "MEMORY SIZE?" Secondly, you must set the I/O index pointer to the beginning of the file. Thirdly, you must, by use of the I/O distributor, output the desired data to the file. Finally, when you are ready, or that particular buffer is full, you must initiate a transfer from the buffer to the disk itself.

READING A FILE

First you must create a memory-resident file buffer, again, by simply allocating space for it when the machine asks "MEMORY SIZE?" upon power-up configuration. Secondly, you must transfer data desired from the disk to memory. Third, you must set the input/index pointer (I/O pointer) to the beginning of the file, or wherever in the file you would like to gain data from. And finally, you must by use of the I/O distributor input to an INPUT statement from a file.

To implement a simple sequential file is extremely straightforward, because with sequential files, the indexes on input and output simply move sequentially. In OS-65D, the disk indexes, or file buffer indexes, automatically increment after each character is outputed. To implement a random access file, you must have a way of calculating the input and output indexes by some formula. An example of this is discussed at the end of this article.

To implement an index sequential file, two files are needed, an index file and a data file. It is possible to have the index file and data file both utilize the same buffer by swapping in the index when desired, and then the data. However, with OS-65D, it is fully possible to allocate memory space for both the index file and part or all of the data file concurrently. Thus extremely fast access to any block of data is possible. The index file is typically organized as a simple sequential file, and the data file is organized as a simple random access file. The index obtained from the index file is used to directly access data in the data file. Of course, other, more complex, file systems are possible under OS-65D. Some examples of this are indirect command files which allow the disk file to act as the executive of the computer system. Also, because of the multiple simultaneously open file capability of OS-65D, merges and sorts of multiple files are both straightforward and extremely fast.

OS-65D Version 2.0 diskettes contain

three examples of data file-based programs in BASIC. A simple telephone directory program is located on track 18. It is documented as Example 5 in OS-65D Version 2.0 manuals. This telephone program simply allows a user to enter a person's name and telephone number and create a single track or 3,000-byte long file containing these names on disks. He can then at any later time go back to the program and call up a phone number based on a person's name. This program is a good example of a simple, single-track sequential file and, of course, can be readily changed for multiple entries such as for mailing lists, where three or four entries would be combined to form a record, such as a person's name, street address, city, and state. It has two limitations, however. It does not have a delete function, and is limited in size to be only one sector, that is, 256 to 3000 words. It cannot exceed one track in length. A refined version of the same telephone program is located on track 44. This is an extension of the simple telephone program which, again, can be easily modified and converted for any application requiring a sequential file. This program has a delete function and is set up for maximum file length of four tracks or 15,000 bytes. It can easily be changed to utilize up to an entire diskette so that a very large mailing list, for instance, could be stored. To conserve space, the program is not listed here. Instead, we ask that you list the program on track 44 in conjunction with this discussion. Make sure that you have a standard Version 2.0 diskette. The extended phone program is very similar to the original phone example except for the delete and multiple track capability. A delete is accomplished by replacing each entry, that is, name and phone number, with the character S, so that if you search for S, you will find all the deleted entries. The process of deleting an entry is to delete the old entry and place the new entry on the end. The program is set up to start at disk file on track 72 and expand downward to track 76. The program can be expanded to any length maximum by simply specifying the variable TT starting at some track lower than 72. It accomplishes its multiple track nature by constantly checking for end of sector, which in this case, equals one track in length. When it approaches the end of a track, it transfers that track out, resets all pointers, increments the track counter, and starts to fill up the buffer for the next track. The program can be directly converted to place the data files on disk drive B by simply changing the variable DN from input and output to drive A to input and output to drive B. With this method, it will be possible to utilize an entire disk as a file for this program.

Using an entire diskette as a file system, and assuming approximately 60 bytes per mailing label, the user should be able to place up to 3800 entries on a single diskette.

The disk-based random access file demonstration program is located on track 42 of OS-65D Version 2.0 diskettes, and the data file, including data, is located on track 43. Please list the program on track 42 in conjunction with this discussion. An operational printout from the file is listed on p. 7. The demonstration program utilizes a 3K byte data file which stores a functional description of the 7400 series TTL components. The user inputs a 7400 series TTL part, of which he wants a description, and from the part number, the index or description's address is calculated so that the program can virtually instantaneously obtain the desired data. The data file provided contains descriptions of 7400 series devices from 7400 to 7450. To run the program, be sure that you respond to MEMORY SIZE? with the number 16000 in order to allocate space for the disk buffer and that you use it with a serial terminal or modify the I/O to return control to a video system, if video is used. A sample printout is given on p. 7. This simple sample of a random access file can be expanded to any random access application.

By utilizing a sequential file to obtain indices, such as the simple phone program listed in the OS-65D manual, and using a random access file such as the 7400 series reference file, one can construct a general purpose index sequential file system which will access data as fast as any standard small business computer on the market today and significantly faster than most.

OS-65D places a lot of responsibility on the programmer's shoulders. However, if a program is properly developed by using OS-65D, it will achieve an extremely high level of performance with a minimal amount of memory overhead. We believe this to be the most important aspect of small computing. That is, in applications situations such as small business computing, the program must be developed once, then used many times. It is not efficient, or particular intelligent, to compromise the computer's performance on a day-to-day basis for the sake of making the task of a programmer a little easier. This unfortunately has been the case on the few other small computer systems which offer true disk file capability.

Ohio Scientific is in the process of developing resale arrangements for several applications packages written in OS-65D. The software packages currently under development include a word processor (now being utilized in the production of the Journal), a complete mailing list program (which was used to generate your mailing label), a complete business package , including accounts receivable, accounts payable, general ledger, inventory, and income tax; and a data-based management system for use with user programs. We would be very interested to hear any recommendations you might have for software packages to use under OS-65D. Please send your suggestions to the Small Systems Journal. We kindly ask that you <u>not</u> call Ohio Scientific to ask about the availability of these applications packages. You will be made aware of what is available through the Journal and by direct mailings.

```
RUN
OSI DISK BASED 7400 SERIES REFERENCE FILE
THE AVAILABLE COMMANDS ARE AS FOLLOWS -
NEW
ADD
END
SEARCH



IF FILE INFORMATION IS DESIRED TYPE 'SEARCH'
IN RESPONSE
TO THE PROMPTING 'COMMAND' PRINTOUT



INPUT COMMAND MODE
? SEARCH
INPUT 7400 SERIES PART NUMBER - E.G.  7400,740
4,ETC.
? 7420



? 7420 DUAL 4-INPUT NAND GATES
? END



INPUT 7400 SERIES PART NUMBER - E.G.  7400,740
4,ETC.
? 7450



? 7450 DUAL 2-WIDE 2-INPUT AND-OR-INVERT GATE
S
? END



INPUT 7400 SERIES PART NUMBER - E.G.  7400,740
4,ETC.
? 7460



NUMBER OUT OF RANGE
THE AVAILABLE COMMANDS ARE AS FOLLOWS -
NEW
ADD
END
SEARCH



IF FILE INFORMATION IS DESIRED TYPE 'SEARCH'
IN RESPONSE
TO THE PROMPTING 'COMMAND' PRINTOUT
```

RESEQUENCING

USING THE PEEK & POKE FUNCTIONS

In response to our request in our August 1977 Small Systems Journal (p. 8), one of our readers, a Mr. L. Barker of Chicago IL has sent us the following resequencing program using the PEEK and POKE functions of BASIC:

```
   1000 X=12671:Y=10
   1005 Y2=INT(Y/256):Y1=Y-Y2*256
   1010 POKE X+2,Y1:POKE X+3,Y2
   1020   Y=Y+10:X=PEEK(X)+256*PEEK(X+1):GOTO
1005
```

This program renumbers all program lines in steps of 10. When line 1005 is changed, the program ends and an error message is produced. If the program is very long and has a lot of GOTO and THEN messages, a second program can be added to PEEK from 12675 up, looking for a 160 or 136. If a 0 is encountered, add 5 to the PEEK number and continue. A table will have to be created and the old line numbers read into it. A blank should be left after the number in the original program, in the event that the GOTO line number has been changed from 99 to 100, for example. The finished program is 25 lines in length, and the running time is not fast. However, this program not only renumbers the lines, but changes all the GOTO and jumps to the new line numbers.

```
LIST

 3 REM   A RESEQUENCE DEMONSTRATION
 5 FOR X=1 TO 10
 7 PRINT "THIS IS A RESEQUENCE DEMO."
 11 NEXT X
 1000 X=12671:Y=10
 1005 Y2=INT(Y/256):Y1=Y-Y2*256
 1010 POKE X+2,Y1:POKE X+3,Y2
 1020 Y=Y+10:X=PEEK(X)+256*PEEK(X+1):GOTO1005
5
OK
RUN
THIS IS A RESEQUENCE DEMO.
THIS IS A RESEQUENCE DEMO.
THIS IS A RESEQUENCE DEMO.
THIS IS A RESEQUENCE DEMO.
THIS IS A RESEQUENCE DEMO.
THIS IS A RESEQUENCE DEMO.
THIS IS A RESEQUENCE DEMO.
THIS IS A RESEQUENCE DEMO.
THIS IS A RESEQUENCE DEMO.
THIS IS A RESEQUENCE DEMO.

?US ERROR IN  1020
OK
LIST

 10 REM   A RESEQUENCE DEMONSTRATION
 20 FOR X=1 TO 10
 30 PRINT "THIS IS A RESEQUENCE DEMO."
 40 NEXT X
 50 X=12671:Y=10
 60 Y2=INT(Y/256):Y1=Y-Y2*256
 1010 POKE X+2,Y1:POKE X+3,Y2
 1020 Y=Y+10:X=PEEK(X)+256*PEEK(X+1):GOTO 1005
```

MEMORY TECHNOLOGIES FOR SMALL COMPUTERS

PART TWO

EPROMS, PROMS, & ROMS

A very important part of any small computer system is its permanent non-volatile storage, that is, programs in the machine which are not forgotten or lost when the power is turned off. Any modern small computer system must have some amount of EPROM, PROM, or ROM memory to allow communications with the user when a machine is first turned on and to function as a bootstrap to load in other programs. However with recent technological advances and cost reductions, it is now practical to utilize ROMs where large fixed programs are frequently used. We will discuss this in detail later.

The term ROM stands for Read Only Memory. PROM is an acronym for Programmable Read Only Memory. EPROM stands for Erasable Programmable Read Only Memory.

EPROMS

EPROMS are available in two forms: as electrically alterable types, such as those made by Nitron ; or as UV (ultra-violet) erasable EPROMs manufactured by several firms. Electrically alterable EPROMs are very slow and expensive and require multiple power supplies, a fact which has prevented them from gaining in popularity for microcomputer systems. We are not aware of any microcomputer currently using electrically alterable EPROMs. The UV erasable EPROM, on the other hand, is very popular, and almost every microcomputer system on the market has some quantity of EPROMs.

EPROMs are similar to ROMs except that they have both read and write circuitry. The EPROM functions by trapping electrons or electron charges in buried gates on the integrated circuit. These gates are the gates of field-effect transistors and have no connection under normal circumstances. Once a charge is injected into them, they can maintain that charge for a very long period of time, up to several years. And of course, the operation of the corresponding field-effect transistors, whether they be on or off, is dependent upon the absence or presence of charge in these gates. This charge can be drawn off or discharged by subjecting the chip to strong ultra-violet radiation. The ultra-violet radiation has photons which have sufficient energy to overcome the potential well of the material and drain the charge off.

At any rate, an EPROM is programmed on a special PROM programmer circuit, typically, which applies pulsed high voltage to the memory cells of interest, injecting charge into these gates. It can then be read out repeatedly by using conventional TTL-level signals. The EPROM can be erased by subjecting the chip to a quartz window on top of the chip to short wave ultra-violet radiation, such as from sunlight, or from a germicidal fluorescent tube, that is, a fluorescent light which has a quartz envelope. Conventional EPROMs will not generally be erased by normal fluorescent room lighting or by subjecttion to sunlight through a glass window since in either case, the short wave ultra-violet radiation is filtered out by normal glass. A UV erasable EPROM can easily be recognized by the clear or translucent quartz window on top of the chip. We will discuss several types of EPROMs later on.

PROMS

PROMs are Programmable Read Only Memories. These are typically fusable link devices which utilize some metallization layer on the chip, such as nichrome, which can be blasted away by electric charges. That is, the device comes with links, or connections at all possible memory cells. By applying a large electrical pulse at a particular memory cell, these links can be blown away. This can only be done once, and of course, if a mistake is made, the part then becomes useless. Fusable link ROMs are usually built on a bipolar technology, that is, they utilize a technology similar to TTL. For this reason, they are typically very fast and have rather high power dissipations. Because of the high speed of the part and the technology involved, fusable link ROMs generally cost as much as UV erasable PROMs. However, they offer little benefit to microcomputers which cannot make use of their very high access speed. By reason of high cost and single-shot-type use, fusable link PROMs have not gained much acceptance in microcomputer applications, and they are not being used in any Ohio Scientific computers.

ROMs are devices which cannot be programmed by the user. A ROM is a device in which the code to be stored is designed into the part by the manufacturer. ROMs are much easier to manufacture than EPROMs, PROMs, or RAMs. The reason for this is that they have no need for storage as RAMs do, or any fancy programmable, erasable element as EPROMs do, or fusable link element as PROMs do, and have absolutely no need for any input drivers, as RAMs, EPROMs, and PROMs do. There are simply output drivers and an array of diodes, or lack of diode, on chips. For this reason, when large production quantities are anticipated, a ROM is a very economical device.

There two types of ROMs: mask ROMs and silicon gate ROMs. A mask ROM is a device which has diodes in place at every possible bit position. By applying the final level of metallization to the chip, one connects bits or leaves them open. In silicon gate ROMs, the actual silicon diodes are placed on the wafer of left off, depending on the bit pattern of the ROM. The simplest, and

therefore, least expensive to produce in quantity, is the silicon gate ROM. However, because of the high level of customization of the chip, the silicon gate ROM represents the largest investment in tooling of any type of ROM, PROM or EPROM to the company using them. We will discuss several types of ROMs in detail below.

The following discussion concerns itself with several EPROMs and ROMs which can be utilized in Ohio Scientific-based computers. Both engineering and end user data are provided here, so that even if you have no desire to utilize these parts in a non-standard configuration, studying them will give you greater insight into your OSI computer.

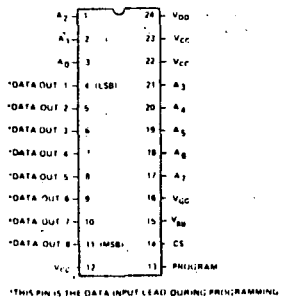
1702A

The 1702A is a UV erasable EPROM containing 2048 bits organized as 256 eight-bit words. The 1702A is one of the earliest UV erasable devices and for a long time has been the most popular permanent storage component in microcomputer systems. Its popularity is dropping now because of higher-technology parts which are now available. The 1702A utilizes two power supply voltages: +5 and -9. It is extremely difficult to program and requires special programmers which are available commercially for $2,000 up. There are a couple of low-cost hobbyist quality programmers available as accessories to hobby computer systems in the $200-and-up price range. Many hobbyist clubs have access to 1702A programmers.

The 1702A is not a particularly fast device. The standard A part is a one us part which will require wait states in any OSI-built 6502 system. Ohio Scientific offers an enhanced 1702A part, the 3702A-2, which is a 500ns access time part, allowing operation of the 6502 or 6800 system at 1.0MHz without wait states. Ohio Scientific utilizes 1702A-type parts on the 400, 500, and 510 CPU boards as monitor PROMs. We also offer an EPROM board which can be configured with up to sixteen 1702As for up to 4K of user permanent storage programs as an OSI 455 EPROM Board. EPROMs available from Ohio Scientific include a 65A, 65V, 68A, and 68V PROM Monitors, which provide the rudimentary load, dump, examine, and bootstrap capabilities for serial and video-based 6502 and 6800 systems. We also provide a floppy disk bootstrap PROM which can be utilized with either a 65A or 65V, and BASIC support EPROMs, used in conjunction with the 8K BASIC ROMs, which will be discussed later on in detail.

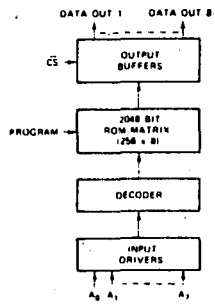
The prospective PROM user should only consider the 1702A part if he has access to 1702As at a discount price and a 1702A programmer. It is definitely not worthwhile to invest in a 1702A programmer at this time.

PIN CONFIGURATION



*THIS PIN IS THE DATA INPUT LEAD DURING PROGRAMMING

BLOCK DIAGRAM



NOTE: In the read mode a logic 1 at the address inputs and data outputs is a high and logic 0 is a low.

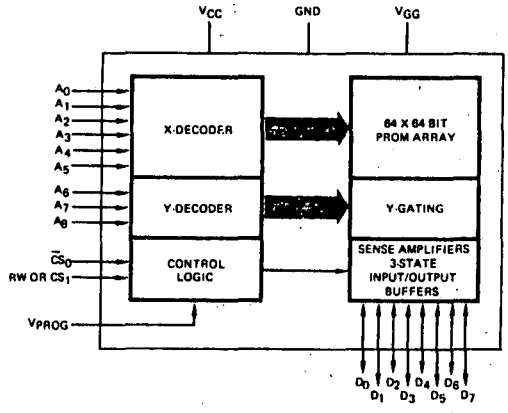
U.S. Patent No. 3660819

PIN NAMES

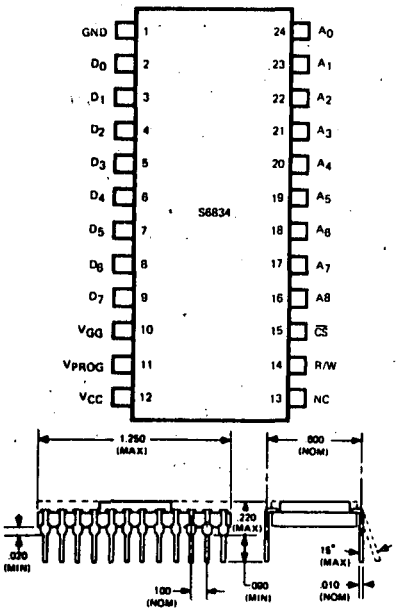
A_0–A_7	Address Inputs
$\overline{CS}$	Chip Select Input
D_{OUT1}–D_{OUT8}	Data Outputs

6834

This UV erasable PROM contains 4096 bits arranged as 512 eight-bit words. The device is manufactured solely by American Microsystems, Inc., commonly called AMI. The part features several improvements over the popular 1702A. It is twice as large as the 1702A, and is very easy to program, but does use +5 and -9 power supplies, although at a very low power dissipation. The 6834 is currently one of the best buys, bit-wise, in a UV erasable EPROM. Check our occasional users group specials in the Small Systems Journal for a current sale on 6834s. Ohio Scientific offers an 8K x 8 EPROM Board utilizing up to sixteen 6834 EPROMs. This board features its own on-board programmer which simply requires an external -50V power supply for operation. The 6834 programming program comes with the board, and is present on all OS-65D diskettes. The user must simply place the program he wishes to transfer into the 6834 PROM in memory, place the PROM in the programmer slot, execute the program, and verify proper programming. The 6834 and our low-cost Model 450 8K x 8 EPROM Board should be seriously considered by anyone who requires some permanent storage.



BLOCK DIAGRAM

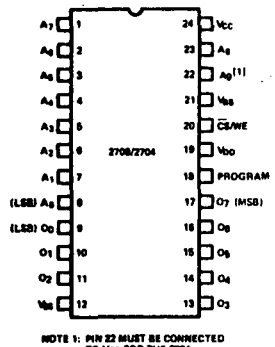


PIN/PACKAGE CONFIGURATION

2704 & 2708

The 2708 is an 8192-bit UV erasable PROM which organizes 1K x 8 of memory. The 2704 is a dropout 2708 part which has only 4K of usable bits and organizes 512 eight-bit words. The 2708 and 2704 are pinned out very similarly to a large family of other EPROMs, PROMs, and ROMs. There are a few disadvantages to the 2704 and 2708. It is true that they are relatively easy to program, but generally do require a dedicated programmer. They also require +12, +5, and-5 supply voltages. The 2704 and 2708 can be utilized on a standard OSI 500 CPU Board in the same sockets that are normally used for 8K BASIC in ROM so that up to 2K of 2704s or 4K of 2708s can be placed on a 500 CPU Board. This is accomplished by reprogramming the two 1610 IC locations which program the four sockets for any of several different parts. There is also a converter on the board which will convert the -9 power supply to -5 for use with these components. We do not recommend the 2704 or 2708 unless the user has access to these parts at a discount price, and has a programmer for them available.

PIN CONFIGURATION

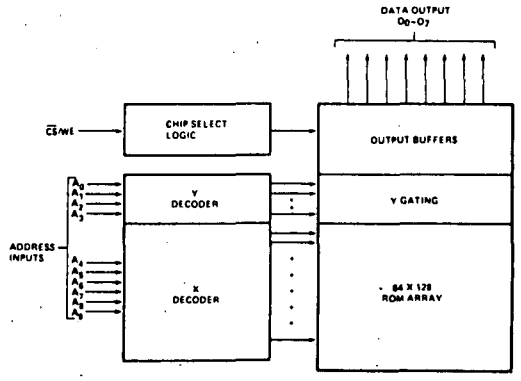


NOTE 1: PIN 22 MUST BE CONNECTED TO V_{SS} FOR THE 2704.

PIN NAMES

A_0-A_9	ADDRESS INPUTS
O_1-O_8	DATA OUTPUTS/INPUTS
$\overline{CS}$/$\overline{WE}$	CHIP SELECT/WRITE ENABLE INPUT

BLOCK DIAGRAM



PIN CONNECTION DURING READ OR PROGRAM

	PIN NUMBER							
MODE	DATA I/O 9-11, 13-17	ADDRESS INPUTS 1-8, 22, 23	V_{SS} 12	PROGRAM 18	V_{DD} 19	$\overline{CS}$/$\overline{WE}$ 20	V_{BB} 21	V_{CC} 24
READ	D_{OUT}	A_{IN}	GND	GND	+12	V_{IL}	-5	+5
DESELECT	HIGH IMPEDANCE	DON'T CARE	GND	GND	+12	V_{IH}	-5	+5
PROGRAM	D_{IN}	A_{IN}	GND	PULSED -6V	+12	V_{PROG}	-5	+5

2716

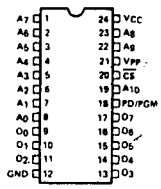
The 2716 is a 16K bit UV erasable PROM, which organizes 2K eight-bit words. The Intel 2716 currently represents the state-of-the-art in UV erasable PROMs. There are two manufacturers offering 2716s: Intel and Texas Instruments. The TI 2716 is a far cry from the Intel part. For instance, the Intel part uses a single +5 power supply and has automatic stand-by operation when it is not being accessed. It has an extremely simple programming procedure. The TI part, on the other hand, has a programming procedure and power supplies just like the 2708, that is, it is rather difficult to program, and has a +12, +5, and -5 power supply. We would strongly advise anyone interested in a 2K part to use the Intel 2716.

The 2716 can be easily programmed by providing latched address and data to the part, applying 25V to its programming pin, and pulsing a programming line for 50ms. Move on then to the next address and data, and pulse for 50ms. It is very easy to set up your own programmer for the part by simply hooking a few PIA ports to a breadboard, and driving the entire programmer with a simple program in BASIC with PEEK and POKE instructions.

The one and only shortcoming to the 2716 at the moment is that it is very expensive. The single-piece list price is in the area of $100. However, as other manufacturers produce the Intel-type 2716, the price will certainly come down and the part will see a great deal of popularity. Four 2716s can be used directly in place of our 8K BASIC ROMs on the OSI 500 CPU Board. As a matter of fact, the first 500 prototypes used 2716s in place of mask ROMs. These 2716s were programmed at OSI via two PIA ports and a very short program in OSI 8K BASIC. Another big feature in the Intel 2716 is that it operates off of a single +5 power supply. It has a low-power dissipation of only 500mw when accessed. It automatically reverts to about 130mw power dissipation when it is not

accessed. The standard part is sufficiently fast to operate on a 1MHz 6502 or 6800 system without any wait states.

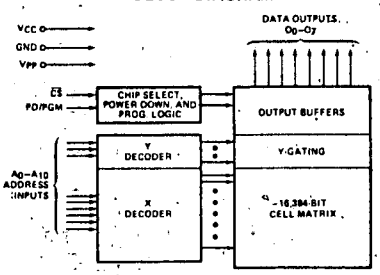
PIN CONFIGURATION



MODE SELECTION

MODE \ PINS	PD/PGM (18)	$\overline{CS}$ (20)	V_{PP} (21)	V_{CC} (24)	OUTPUTS (9-11, 13-17)
Read	V_{IL}	V_{IL}	+5	+5	D_{OUT}
Deselect	Don't Care	V_{IH}	+5	+5	High Z
Power Down	V_{IH}	Don't Care	+5	+5	High Z
Program	Pulsed V_{IL} to V_{IH}	V_{IH}	+25	+5	D_{IN}
Program Verify	V_{IL}	V_{IL}	+25	+5	D_{OUT}
Program Inhibit	V_{IL}	V_{IH}	+25	+5	High Z

BLOCK DIAGRAM



PIN NAMES

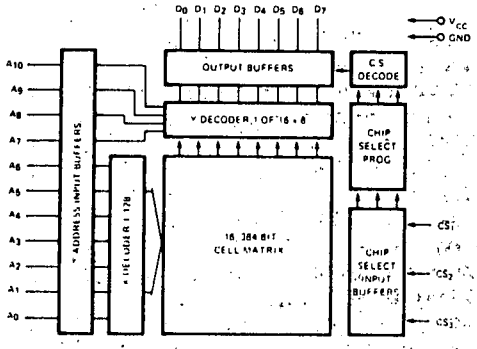
A_0–A_{10}	ADDRESSES
PD/PGM	POWER DOWN/PROGRAM
$\overline{CS}$	CHIP SELECT
O_0–O_7	OUTPUTS

ROMS

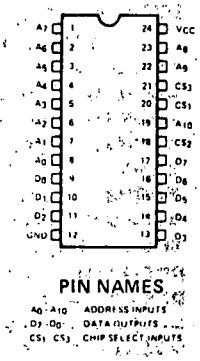
2316 & 2616

The 2316-type ROM has 16,384 bits organized as 2K words of eight bits each. This part goes by several different part numbers from the various manufacturers, but it is all basically the same. The part is available as a mask ROM and as a silicon gate ROM. The former is being offered in moderate quantity, and the latter for very large production quantities. The part operates with +5 voltage supply and is sufficiently fast for operation on a 1MHz 6502 or 6800 system. The part has the same pinout as the 2716, and is, of course, fully compatible with our Model 500 CPU Board. Ohio Scientific offers four Signetics 2616 2K x 8 ROMs, which make up 8K BASIC in ROM. These are silicon gate parts, and are addressed for A000 up, A800 up, B000 up, and B800 up. They contain our standard 8K BASIC interpreter and several I/O routines, including serial interface, serial I/O handler, video CRT routine, and audio cassette I/O. All subroutine calls, as well as all critical parameters are placed outside of the ROMs via hooks to another PROM, EPROM, or ROM at FF00 up. This EPROM is commonly referred to as the BASIC support EPROM, and is available at this time in two forms: one is for a serial interface-based 6502 system, and the other for a video interface-based 6502 system. By changing the locations in this EPROM, several functions of 8K BASIC can be changed. For instance, the start of user workspace and the start of the scratchpad are both specified in the EPROM. Of course, all the I/O calls and routines are patched out in the EPROM, as well as the starting and reentry points to BASIC. This allows the 8K BASIC ROMs to be totally general purpose so that they can be offered with several configurations. Another extremely important use of these BASIC ROMs which has not been explored yet is that of using BASIC as a programming language for industrial development; that is, it is possible with this ROM set and a proper support EPROM to write programs in BASIC, and store the actual programs in EPROM. Then with a modified support PROM, the BASIC scratchpad area is in RAM, but the BASIC program is in EPROM, and restart is always directly in the program. It is thus possible with these support ROMs to develop programs with dedicated applications in BASIC, blast them into PROM, and build a dedicated controller which always restarts directly in the BASIC program. The use of hooks out of the ROMs on all-important parameters insures the user against the obsolescence of the ROMs. They can therefore be used in several computer configurations. Ohio Scientific is also developing other 2K x 8 ROM programs such as advanced character generator with graphics and schematic symbols, and more advanced monitored ROMs, since at Ohio Scientific's current production level, silicon gate ROMs are less expensive than EPROMs or PROMs.

BLOCK DIAGRAM



PIN CONFIGURATION



PIN NAMES

A_0–A_{10}	ADDRESS INPUTS
D_7–D_0	DATA OUTPUTS
CS_1 CS_3	CHIP SELECT INPUTS

2633 & 2665

The 2316 family does not stop at a 16K bit ROM. Already prototypes of 32K bit and 64K bit ROMs are coming out. Examples of this are the Signetics 2633 32K bit organized as 4K x 8, and the 2665 65K bit organized as 8K words of eight bits. These ROMs have the same pinout as the 2716/2316, so that they can be used directly on OSI 500 CPU Boards. As a matter of fact, one 8K x 8 ROM could be substituted directly in place of four 2K x 8 ROMs. Ohio Scientific will switch over to a single 8K x 8 ROM for BASIC as soon as 8K x 8 ROMs come into production, but this will not be until mid-1978. The only advantage that the 8K part has over four 2K parts, will be slightly lower packaging costs and slightly lower PC board use. These large ROMs have dropped dramatically in price over the last year, primarily because of the high demand placed on them by video game manufacturers. The widespread use of home computers and video game computers will make large-scale silicon gate ROMs a household item in the form of ROM cartridges which contain games. This makes much more sense to the manufacturer, because it virtually eliminates the one common plague of our industry, that is, software bootlegging by hobbyists. By placing large programs in ROM cartridges, the user will be offered instant loading of a virtually indestructible program, and the computer manufacturer will be offered security in his software investment.

Contributed Program

```
.0 PRINT: PRINT
:0 PRINT "PROG. 0002       7/24/77"
:5 PRINT: PRINT"BY GARY A. SMITH CINTI OH"
:0 PRINT:PRINT
·0 PRINT"THIS PROGRAM DISPLAYS"
:0 PRINT"THE MEMORY ADDRESS AND"
·0 PRINT"DATA CONTAINED IN HEX"
'0 PRINT
:0 PRINT"USER INPUTS THE STARTING"
:0 PRINT"AND STOPPING ADDRESS IN"
.00 PRINT"DECIMAL"
.10 PRINT: PRINT
20 A=0: B=0: C=0: CC=0: F=0
30 INPUT "START ADDRESS";A
40 PRINT
50 INPUT "STOP ADDRESS";G
60 PRINT
70 B=A-1
80 C=C+1
90 CC=CC+1
00 F=0
10 D=C+B
20 GOTO 400
30 DATA 4096,256,16,1
40 H$="0123456789ABCDEF"
50 RESTORE
60 N=VAL(N$)
70 X$=""
80 J=4
90 READ P
00 FOR I=1 TO 16
10 IF N-I*P<0 THEN 350
20 NEXT I
30 PRINT">>>>INPUT ERROR ↑"
40 GOTO 110
50 X$=X$+MID$(H$,I,1)
50 N=N-(I-1)*P
70 J=J-1
30 IF J>0 THEN 290
90 GOTO 420
00 N$=STR$(D)
L0 GOTO 230
20 F=F+1
30 IF F=1 THEN PRINT CC; TAB(8);X$;" ";
10 IF F=2 THEN PRINT RIGHT$(X$,2)
50 IF F=2 AND G=CC+A THEN 490
50 IF F=2 THEN 180
70 N$=STR$(PEEK(D))
30 GOTO 230
90 PRINT
00 PRINT"END. . . . . "
.0 PRINT
20 GOTO 110
50 END
```

```
OK
RUN

PROG.  0002       7/24/77

BY GARY A.  SMITH CINTI OH

THIS PROGRAM DISPLAYS
THE MEMORY ADDRESS AND
DATA CONTAINED IN HEX

USER INPUTS THE STARTING
AND STOPPING ADDRESS IN
DECIMAL


START ADDRESS? 512

STOP ADDRESS? 535

 1       0200 F3
 2       0201 07
 3       0202 0F
 4       0203 07
 5       0204 02
 6       0205 0C
 7       0206 C5
 8       0207 08
 9       0208 F0
 10      0209 0A
 11      020A CB
 12      020B 0E
 13      020C 1C
 14      020D 0B
 15      020E 72
 16      020F 09
 17      0210 72
 18      0211 08
 19      0212 4A
 20      0213 08
 21      0214 F5
 22      0215 08
 23      0216 D3

END. . . . . .
```

L45,0

OSI 6502 CYCLE TIME TEST

Purpose:

To measure the cycle time of a system using a watch with second hand or a stopwatch.

Memory Allocations:

The program starting address is 0200, and the program resides from 0200 to 0223, using page zero locations 00-02.

Relocation Information:

Location 0221 contains the only position dependent instruction of the program. This should be adjusted if the program is being run at memory locations other than those specified above.

Use:

Start the program and your watch at exactly the same time. Push reset at the end of the timing period (any length from 0 to 500 seconds). Then examine the contents of locations 00, 01, and 02.

The contents of each of these locations contain a number from 0 to 99 in Binary Coded Decimal (BCD) format. Using the least four significant bits (one BCD digit) of location 00 as one's place, the four most significant bits of location 00 as ten's place, the four least significant bits of location 01 as hundred's place, the four most significant bits of location 01 as thousand's place, the four least significant bits of location 02 as ten thousand's place, and the four most significant bits of location 02 as hundred thousand's place, obtain a decimal number.

To obtain the machine cycle time, substitute in the following formula (results obtained will be in seconds):

$$N/(1000*T)$$

where N is the number of seconds the program is run and T is the number found in locations 00 to 02.

For example, a program was run for 40 seconds. The contents of location 00 were 47, the contents of 01 were 08, and the contents of 02 were 07. The resulting number was 070847 or 70847. In this case, N=40 and T=70847. Substituting into the formula, we have,

Cycle Time= 40/(1000*70847)=.56us.

The dump for this program is shown below, following which is an A and A2 assembly.

```
ASSE

 10 0000          ; CYCLE TIME TEST
 20 0000          ;   CT=N/(1000*T)
 30 0000          ;      WHERE:
 40 0000          ;        CT=CYCLE TIME
 50 0000          ;        N=SECONDS RUN
 60 0000          ;        T=3 BYTE BCD NUMBER IN C1,C2,C3 (C1 IS LSB)
 70 0000          ;
 80 0000          C1=0
 90 0000          C2=1
100 0000          C3=2
110 0000          ;
120 0200                *=$200
130 0200 A900     START LDA #0
140 0202 8500           STA C1
150 0204 8501           STA C2
160 0206 8502           STA C3
170 0208 F8             SED
180 0209 A500     LOOP  LDA C1
190 020B 6901           ADC #1
200 020D 8500           STA C1
210 020F A501           LDA C2
220 0211 6900           ADC #0
230 0213 8501           STA C2
240 0215 A502           LDA C3
250 0217 6900           ADC #0
260 0219 8502           STA C3
270 021B A2C2           LDX #$C2
280 021D CA       LOOP1 DEX
290 021E D0FD           BNE LOOP1
300 0220 EA             NOP
310 0221 4C0902         JMP LOOP  INFINITE LOOP
320 0224          ;               TERMINATED BY RESET.
330 0224                .END


A2

;180200A900850085018502F8A500690185 00A501690085 01A502690786
;0C0218008502A2C2CAD0FDEA4C090205E9
```

HARDWARE

COMPUTER LAB ON A BOARD

INTRODUCTION TO THE 560Z

The 560Z is a totally new and revolutionary computer product. We are inclined to call it a CPU Expander, or a computer lab on a board. The 560Z has many applications including the following: execution of standard PDP-8, Z-80, and 8080 programs; investigation of microcoding and multiprocessing; investigation of Z-80 and/or 6100 operation and architecture; a building block for large, experimental multiple processor arrays. Each of these applications lends itself to further discussion.

The 560Z Board utilizes an Intersil 6100 microprocessor which is capable of executing the standard PDP-8E CPU instruction set. It is important that we state that it executes only the CPU's instruction set, that is, it does not execute exactly I/O instructions that may be found on various PDP-8 configurations. This is one of the main reasons that PDP-8-compatible microcomputers have not been available, even though the Intersil 6100 chip has been around for a long time. This is because the PDP-8 uses what is known as microcoded IOT instructions; that is, each peripheral interface must have the intelligence approaching that of a conventional CPU. For instance, I/O devices can force skips, rotates, and other operations in the accumulator as well as providing data and other instructions to the processor during I/O operations. The 560Z overcomes these difficulties by having the executive 6502 processor provide the intelligence on all I/O instructions. It does this by actually microcoding the PDP-8 front panel, interrupt, and IOT instructions. We will discuss the concept of microcoding a little later, but what this actually boils down to is that the 560Z, in conjunction with a 6502 microprocessor, can totally emulate the functions of a PDP-8E computer, allowing it to run standard PDP-8 code without any modification to that code at all.

The PDP-8 currently has the largest library of public domain software available in the world. Ohio Scientific does not provide any standard PDP-8 software. We suggest that the user become a member of DECUS, the Digital Equipment's Users Group. There are thousands of programs available to members of DECUS at simply the cost of duplication, to run on PDP-8 I and E-based computers. Software provided by Ohio Scientific in this package should allow the user to run any standard PDP-8 4k Teletype-based program without any modification. By simply adding entries to the IOT microcode table in the software, the user should be able to emulate, or simulate, any standard PDP-8 configuration including those using disk operations and extended memory.

The 560Z also utilizes a Z-80 microprocessor, by means of which it is, of course, possible to run standard Z-80 programs. The 560Z is configured to allow the user to microcode interrupt and I/O instructions on the Z-80 via the 6502, so that he can simulate or emulate I/O ports found on other computer systems with his standard OSI computer system.

The Z-80 on the 560Z Board is capable of running 8080 code. It does handle interrupts slightly differently from the 8080; however, since the 560Z allows microcoding of I/O and interrupt operations, it is possible to configure the 560Z system to exactly emulate standard 8080-based computers to allow the operation of 8080 programs without any modification to those programs.

The field of microcoding, or microprogramming, is totally new to microcomputer users. Microcoding has really nothing to do with microprocessors, per se. It has to do with coding, programming, or specifying the instruction set of a computer. That is, during the initial design phases of any computer, and in some large computer systems, it is possible or necessary to specify or dynamically specify the binary codes which perform certain operations. The question is, what does the machine specify, and what exactly does it do when it performs an ADD operation? The 560Z system allows an elementary form of microcoding to be used with the 6100 and Z-80 microprocessors. This is possible because the 6502 system can have complete executive control over the computer system. It can read any signal line on either of the processors and can force certain conditions. The use of microcoding is necessary for I/O instructions and interrupts on both the Z-80 and 6100 to perform the desired emulations, but can be utilized in other areas. To clarify this application of microcoding, let us consider a couple of specific examples.

Take first the case of a Z-80 processor executing a program and encountering an INPUT statement. On the 560Z, normally configured, when the Z-80 sees an INPUT or OUTPUT statement, a signal line on the processor goes low which stops its clock so that the Z-80 is stopped dead with the INPUT instruction on its bus and the INPUT address port on the low eight bits of its address bus. The 6502 system processor then observes this condition, goes to a table in its code, which specifies what action to take (based on this input port), obtains the proper information, and shoves it into the Z-80. It then single-steps the Z-80 out of the INPUT instruction, in which case, the machine takes off at full speed again. In this way, the 6502 system, by a process of microcoding the Z-80 instruction, has simulated or emulated some standard 8080 or Z-80 I/O port by utilizing the resources that it has. For instance, an 8080 program may specify a Teletype port at input port 20, and the user may have only an OSI video board. This can

easily be accommodated by microcoding the input.

In another example, a 6100 is performing an IOT instruction to Teletype. This is more sophisticated because the Teletype port must specify conditions within the accumulator of the 6100. But the principle is the same. The 6100 sees the instruction and its processor, and its clock automatically stops, leaving the critical data and addresses on its buses. The 6502 then reads these buses, takes appropriate action, and single-steps the 6100 chip through the IOT instruction, providing it with information, and reading information from it as necessary. When the 6100 has been clocked all the way through the instruction, the machine takes off at full speed. Although the microcoded I/O instructions themselves take much longer than normal instructions do, their occurrence in programs is usually infrequent so that the actual program execution takes only a fraction of a percent longer than it would take with conventional I/O.

All signal lines and the address and data lines of the Z-80 and 6100 are available to be examined under program control by the 6502. The Z-80 and 6100 clocks are also single-cyclable by the 6502, and the machines are fully static. This allows a very elaborate and detailed study of the Z-80 and 6100 on a cycle-by-cycle basis with complete printout of all signal lines. This enables the student or engineer to completely understand the operation of these processors including many subtleties which are not documented in any manuals for either processor.

The 560Z system can be utilized as a true multiprocessor. That is, the 6502 executive can set up a task on the 6100 or Z-80 and then detach itself from it so that the 6100 or Z-80 is running completely by itself, independent of the 6502, which can then go on to another task. Thus, in the 560Z system, two processors can be running separately and independently. Multiple 560Zs can be placed on one 6502 system, and 560Zs can be placed on the system bus side of other 560Zs, so there is really no limit to the number of processors or amount of memory present on 560Z-based computers; allowing large, multiprocessor arrays.

It is apparent from the above discussion that the 560Z is a powerful research tool. Several 560Zs can be placed on an individual computer bus and 560Zs can be daisy-chained. It is also possible to cross-couple units with or without processors. That is, two 560Z Boards can be populated to have portholes only with no processors present, and be utilized between two computer systems as high-speed portholes from one system to the other. Other exotic multiple processor arrays are of course possible by using the 560Z Board as a building block with the standard OSI system bus.

The 560Z output bus drivers are fully tristatable by an external device. This is done specifically so that other processors can be present on the system side, as well as on the executive (MOS) bus side. This is of particular interest to 510 Board users, because the 510 software switch has a fourth position which can be utilized to control the tristate outputs of the 560Z. It is therefore possible to have a 510 CPU Board on the executive (MOS) bus side of a 560Z and to have a 510 present on the system side. This means that it is a very straightforward operation to utilize a 560Z Board as a porthole between two 510 systems, or as an intelligent porthole with its own processor

The 560Z is finally available from stock! Consult the current price list for details.

MEMORY TEST

A MEMORY TEST PROGRAM FOR VIDEO AND SERIAL-BASED COMPUTERS

The following program is listed here and on track 29 for serial-based computers and track 40 for video-based computers on all OS-65D Version 2.0 diskettes. The Memory Test Program consists of three parts: a test part, which checks dynamically for pattern senstivity and bit failures; a static memory test load routine; and a static memory test compare routine.

The dynamic portion of the Memory Test Program utilizes a romping bit test successively through memory and, on video systems, it prints out the pass that it is on. When the dynamic memory test makes 256 passes through memory, it will place an X on the video screen or on the serial system. If a block of memory makes it through this test twice, that is, places two Xs, there is a 99.5% probability that it has no dynamic problems. If an error occurs, the location of the error will be printed out, along with the data that was supposed to be in that memory location, and the data that was really in that memory location.

There are several possible failure modes that can be shown by this. This test can find address line shorts, data line shorts, address-to-data line shorts, pattern sensitivities, and dead memory chips. If your memory board has once worked, such as an OSI-manufactured memory board, then the possibility of shorts occurring later on the board is very unlikely. Thus, if you are checking a system that has started to break down, it has most likely developed a problem in a memory chip. This can be recognized with the dynamic memory test by having one bit be wrong at all times within a block of memory. For instance, the test shows a 60 turned into a 62 at location 6023. This would indicate that something is wrong with data bit 1 in that block of memory. If the memory board were a 4K, and it were found that the same failure occurred anywhere in the block from 6000 to 7000, you may be

virtually assured of a dead memory chip. However, memory chips do not always fail completely. That is, they may have one or a few bad locations, while the rest are still fine.

Another dynamic memory failure is pattern sensitivity. Most cases of pattern sensitivity cannot be checked for with the Memory Test because, in order to thoroughly test the memory chip for pattern sensitivity, you will need a number of patterns equal to the factorial of the size of the chip. For instance, if you have a 1K memory chip, you need 1024! patterns to test the chip, which is of course impossible, because that number is larger than the total number of atoms in the universe! Fortunately, memory chip design is carefully done to minimize the problem of pattern sensitivity. It is therefore extremely unlikely that it will occur in any of your memory boards. Out of thousands of boards, we have seen only about a half-dozen cases of memory failure based on pattern sensitivity.

If you are testing a memory board out for the first time, the situation could be much trickier. Whenever possible, test a new memory board with memory chips which are known to be good; that is, take the memory chips out of a working board, and put them in your new board, and put your new chips in your working board. In this way, you will be able to check whether you have failures in memory chips, solder bridges or open foils. Address line shorts can be somewhat difficult to interpret from the Memory Test Program, and address-to-data line shorts can be extremely difficult for a layman to find by way of a Memory Test Program.

Once you get a memory board to pass a dynamic memory test, you should then check it for static operation. That is, a very common failure mode of memory chips is to forget or change data over a period of several seconds to a few minutes. There is a memory load command which allows you to load a block of memory with a particular bit pattern. There is a memory compare routine which allows you to compare a block of memory against the bit pattern. The procedure for this test should be to load memory with zeros, then wait two to five minutes and compare that block of memory against zeros. Then load the memory with FF, wait two to five minutes, and compare it with FF. Then load the memory with AA, and compare again. Finallly load the memory with 55. Any errors will be reported as with the dynamic test. If the memory board passes the dynamic test, but fails this test, it is most probable that the memory chip specified directly by the error message is at fault.

PROGRAM OPERATION

The following instructions are for program operation. Both the serial and video programs start at location 0200 and extend upward for approximately two pages. The lower limit of testing on either program is location 0400 upward. By attempting to test locations below this point, you would bomb the Memory Test program. When the Memory Test Program is entered, a ? appears on the terminal. It then expects a T for (dynamic) Test, an L for Load, or a C for Compare. Your first test should be the dynamic test. Therefore, type a T, then a 0400, the starting address of the test, followed by a four-digit ending address of the test. Thus, if you had a 4K machine, you would type T04001000.

On video-based computers, you will then see a two-digit counter increment, until it passes through FF, in which case it will place an X on the screen and increment again. The presence of an X indicates that it has successfully passed the test once. On serial systems, you will not see anything until an X is printed out or a failure occurs. The Memory Test will take approximately two minutes to test 4K of memory.

The only way to exit a successfully running dynamic memory test is by resetting the computer. If a failure occurs, the test program reverts to the command mode so that additional commands can be typed in. The syntax for the L Command is simply L,SSSS,EEEE,nn, where nn is the two-digit byte pattern you wish to load, SSSS is the starting address of the memory you wish to load, and EEEE is its ending address. Once the load is complete, an asterisk (*) is put out on the screen. This should occur nearly instantaneously. The syntax for a compare command is C SSSS,EEEE,nn. If the compare is successful, a * is printed on the screen. If not, the first location to fail is printed out with its contents. In all tests, the last location specified is not tested.

See Page 17

BUGS & FIXES

8K BASIC users should note that while executing a WAIT statement, 8K BASIC does not recognize Control-C. BASIC was designed this way to provide the user with maximum polling speed. If a mistake is made during a WAIT statement, the user must reset the machine to regain control of it.

A bug has been discovered in the Assembler that causes erroneous object code from an A3 assembly. The following corrections should be made to your Assembler to rectify this error:

Location	Contents	Change to
1067	1C	1E
106B	1D	1F
1071	1C	1E
107D	1C	1E
123C	1C	1E

Serial Memory Test

```
:U0200,039C
      0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
0200  4C 76 03 AD 00 FC 4A 90 FA AD 01 FC 29 7F 60 D8
0210  A9 3F 20 57 03 20 03 02 C9 43 F0 08 C9 4C F0 04
0220  C9 54 D0 F1 C6 FE 20 57 03 A9 3A 20 57 03 A2 00
0230  20 03 02 C9 0D F0 C9 20 1F FE 30 F4 20 38 03 E8
0240  E0 04 30 EC D0 05 A9 2C 20 57 03 E0 08 30 E1 D0
0250  5D AD A0 03 29 3F C9 14 D0 4F A2 02 A9 B7 20 38
0260  03 CA D0 F8 A9 E4 85 FE 20 25 03 4C 85 02 A5 FF
0270  48 A9 D0 85 FF 85 FE 8A 20 6E 03 8A 20 72 03 68
0280  85 FF 68 85 FE 20 2E 03 91 FA 20 14 03 D0 F6 20
0290  25 03 20 2E 03 D1 FA D0 52 20 14 03 D0 F4 E8 D0
02A0  C7 A9 58 20 57 03 4C 68 02 A9 3D 20 57 03 E0 0A
02B0  30 9B AD A0 03 29 3F C9 03 F0 20 20 25 03 A5 F7
02C0  91 FA 20 14 03 D0 F7 A9 20 20 57 03 4C 85 03 EA
02D0  EA 20 03 02 C9 0D D0 F9 4C 00 02 20 25 03 A5 F7
02E0  D1 FA D0 07 20 14 03 D0 F5 F0 DC 85 F9 B1 FA 85
02F0  F8 A2 04 A9 CF 85 FE A9 20 20 57 03 B5 F7 20 6E
0300  03 B5 F7 20 72 03 CA F0 08 E0 02 F0 EA 10 ED 30
0310  E6 4C 8C 03 E6 FA D0 02 E6 FB A5 FB C5 FD D0 04
0320  A5 FA C5 FC 60 A5 F8 85 FA A5 F9 85 FB 60 A5 FA
0330  45 FB 85 F7 8A 45 F7 60 48 A0 04 0A 0A 0A 0A 2A
0340  26 F7 26 FC 26 FD 26 F8 26 F9 88 D0 F2 68 09 30
0350  C9 3A 30 03 18 69 07 48 91 FE E6 FE D0 0C A9 A0
0360  85 FE EA EA EA EA EA EA EA 68 4C 0B FE 4A 4A
0370  4A 4A 29 0F 10 D8 A9 03 85 FF A9 A0 85 FE A0 00
0380  EA EA 4C 8C 03 A9 2A 20 57 03 D0 EA A9 0D 20 0B
0390  FE A9 0A 20 0B FE A2 FF 9A 4C 0F 02 00 00 00 00
```

Video Memory Test

```
:U0200,0376
      0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
0200  20 65 03 A8 A9 20 20 57 03 D0 F9 A9 C4 85 FE D8
0210  A9 3F 20 57 03 20 ED FE C9 43 F0 08 C9 4C F0 04
0220  C9 54 D0 F1 C6 FE 20 57 03 A9 3A 20 57 03 A2 00
0230  20 ED FE C9 0D F0 C9 20 93 FE 30 F4 20 38 03 E8
0240  E0 04 30 EC D0 05 A9 2C 20 57 03 E0 08 30 E1 D0
0250  5D AD C4 D0 29 3F C9 14 D0 4F A2 02 A9 B7 20 38
0260  03 CA D0 F8 A9 E4 85 FE 20 25 03 A5 FE 48 A5 FF
0270  48 A9 D0 85 FF 85 FE 8A 20 6E 03 8A 20 72 03 68
0280  85 FF 68 85 FE 20 2E 03 91 FA 20 14 03 D0 F6 20
0290  25 03 20 2E 03 D1 FA D0 52 20 14 03 D0 F4 E8 D0
02A0  C7 A9 58 20 57 03 4C 68 02 A9 3D 20 57 03 E0 0A
02B0  30 9B AD C4 D0 29 3F C9 03 F0 20 20 25 03 A5 F7
02C0  91 FA 20 14 03 D0 F7 A9 20 20 57 03 A9 2A 20 57
02D0  03 20 ED FE C9 0D D0 F9 4C 00 02 20 25 03 A5 F7
02E0  D1 FA D0 07 20 14 03 D0 F5 F0 DC 85 F9 B1 FA 85
02F0  F8 A2 04 A9 CF 85 FE A9 20 20 57 03 B5 F7 20 6E
0300  03 B5 F7 20 72 03 CA F0 C8 E0 02 F0 EA 10 ED 30
0310  E6 00 00 00 E6 FA D0 02 E6 FB A5 FB C5 FD D0 04
0320  A5 FA C5 FC 60 A5 F8 85 FA A5 F9 85 FB 60 A5 FA
0330  45 FB 85 F7 8A 45 F7 60 48 A0 04 0A 0A 0A 0A 2A
0340  26 F7 26 FC 26 FD 26 F8 26 F9 88 D0 F2 68 09 30
0350  C9 3A 30 03 18 69 07 91 FE E6 FE D0 02 E6 FF A9
0360  D4 C5 FF D0 08 A9 D0 85 FF A9 00 85 FE 60 4A 4A
0370  4A 4A 29 0F 10 D8 F5 BD E7 F3 67 B7 F3 F5 F3 BF
```

1K CORNER

CLOSE THE WINDOW

Close the Window is a dice game designed to be played on OSI 65V computers.

At the start of the game, the player is presented with nine "windows," the digits from 1 through 9. After each roll, the player may close up to three windows whose sum is equal to the value of the roll. For example, at game start with a roll of 10, the player may close windows 1 and 9, or 6, 3, and 1, or 1, 2, and 7, etc. If after any given roll, the player is unable to close any windows, he loses the game. If the player closes all the windows, he wins.

Control keys:

Carriage return: after roll, causes roll of dice.

Carriage return: after "close" and N digits (where 0<N<=3), closes legal windows. If not legal, choice is cleared, but roll remains.

/ (slash): clears close digit errors.

^ (shift-N): new game.

The operational portion of the program resides from 0200 (hex) to 02FD (hex), and execution starts at 0200 (hex). Zero page locations F0 (hex) through FA (hex) should be loaded as follows for ASCII words output:

F0-52	F3-4C	F6-4C	F9-45
F1-4F	F4-2C	F7-4F	FA-3F
F2-4C	F5-43	F8-53	

Optionally, the short loading routine (0300 to 0328) may be used with the program initialized at 0300.

Close the window

```
*RE
U0200,0328
      0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
0200  F8 A2 01 8A 95 E0 E8 E0 0A D0 F8 A9 D0 85 FF A9
0210  C6 85 FE A0 FF A9 20 91 FE 88 D0 FB A2 01 B5 E0
0220  09 30 C9 30 D0 02 A9 20 91 FE C8 C8 E8 E0 0A D0
0230  ED A0 83 A2 00 B5 F0 91 FE C8 E8 E0 05 D0 F6 20
0240  DF 02 C9 5E F0 BA C9 0D D0 F5 18 A5 EA 65 EB 85
0250  EC 4A 4A 4A 4A 09 30 C9 30 D0 02 A9 20 91 FE A5
0260  EC C8 09 30 91 FE A0 E0 A2 00 B5 F5 91 FE E8 C8
0270  F0 08 E0 06 30 F4 A9 20 10 F2 A2 00 A0 E7 20 DF
0280  02 C9 5E F0 BF C9 2F F0 DD C9 0D F0 13 C9 31 30
0290  ED C9 3A 10 E9 E8 E0 04 F0 CC 91 FE C8 C8 30 DE
02A0  18 A0 E7 A9 00 85 ED A9 0F 31 FE 65 ED C8 C8 C0
02B0  ED D0 F2 C5 EC D0 AF A9 FF 85 EE A0 E7 A9 0F 31
02C0  FE F0 0B AA D5 E0 D0 9E A5 EE D0 02 95 E0 C8 C8
02D0  C0 ED D0 E9 A9 00 C5 EE D0 DF A0 00 4C 0B 02 A9
02E0  06 C6 EA F0 11 C6 EB F0 11 AD 01 DF 30 F1 48 AD
02F0  01 DF 10 FB 68 60 85 EA 10 EF 85 EB 10 EB EA EA
0300  A9 52 85 F0 A9 4F 85 F1 85 F7 A9 4C 85 F2 85 F3
0310  85 F6 A9 2C 85 F4 A9 43 85 F5 A9 53 85 F8 A9 45
0320  85 F9 A9 3F 85 FA 4C 00 02 FF FF FF FF FF FF FF
```

ODDS & ENDS

If you type too many characters on a line, the terminal will cease displaying the characters typed, but instead sounds a bell (Control-G). If the terminal does not have a bell, such as on systems utilizing the 440 Board, you see a character G on the screen.

USERS GROUP SPECIAL

Buy a 450 8K PROM board at the regular price of $35

and get one 6834 (512x8) EPROM FREE

Order additional 6834 EPROMs for $15

(The 450 features its own on Board programmer!)

Offer good only through November 31

JOURNAL SUBSCRIBERS

If you have not purchased a fully assembled Challenger system or paid $6 for a regular subscription this is the last complimentary issue you will receive. To continue enjoying the Journal, complete the form below and return it to OHIO SCIENTIFIC.

Please enter my subscription to the Journal.
I enclose $6.

Name_____

Address_____

City_____ State_____ Zip_____

OHIO SCIENTIFIC 11681 Hayden Hiram, OH 44234

SYSTEMS ORDER FORM

For Complete Integrated Systems Only

1. Name_____ Date of Order:_____

Address_____ City_____ State_____ Zip_____

Phone_____ Shipping Address(if different)_____

City_____ State_____ Zip_____

2. Payment by: Personal Check____ Money Order____ Certified Check____

Bank Americard/Visa____ Master Charge____

Name as it appears on the card_____ Number as it appears on card_____

Expiration date of card_____ Signature_____ Amount being charged_____

	PRODUCT NAME/NUMBER	PRICE
3. mainframe		
4. options		
5. memory		
6. mass storage		
7. accessories		

Terms:
* Checks must clear BEFORE shipment is made. (Shipping charges $4.00)
* 20% deposits required on all C.O.D. orders.
* All orders shipped via insured UPS unless otherwise requested.
* Purchase orders accepted from large, well rated companies and schools only.
* Prices, specs., and terms subject to change without notice.

Merchandise total_____

TAX_____

Shipping Charges_____

Amount enclosed or charged_____

Balance due COD_____

Return to:

OHIO SCIENTIFIC
11681 Hayden Hiram, OH 44234 216-569-3241

PRICE LIST

The following is Ohio Scientific's current price list. Each item in this list appears with a page number in section D of this catalog where you will find a detailed specifications table for that particular item. All prices and specifications are subject to change without notice.

400/500 Bare Boards with Manuals and Kits

Model 500 CPU Board & Manual (D-7)	$39.00
Model 504V Serial CPU Kit including 500 Board Manual & all parts for 1K serial interface computer including 65A PROM Monitor (D-7)	$149.00
Model 504V Video CPU Kit containing 500 Board & all parts for complete 1K video computer used in conjunction with 440 Video Board. Kit includes 65V PROM Monitor. (D-7)	$134.00
Model 420C Memory Board & Manual (D-1)	$35.00
Model 422 Memory Board & parts including 4K x 8 low-power 350ns memories for 2MHz operation (D-1)	$99.00
Model 427 Memory Board & parts including 4K x 8 of 450ns medium power 2102 memories suitable for use with ROM BASIC (D-1)	$79.00
Model 430B I/O Board (D-2)	$35.00
Model 440B Video Graphics Board (D-3)	$35.00
Model 446 Video Graphics Board & parts-complete kit for an alphabetics-only video display, with some parts for expansion to graphics included (D-3)	$129.00
Model 450B PROM Board (D-4)	$35.00
Model 455 PROM Board (D-4)	$35.00
Model 560Z Board with Z-80 & 6100 CPU chips (D-11)	$125.00
Model 475 Kit including Model 470 Controller Board in kit form, fully-assembled and tested GSI Model 110 drive & interconnecting cable, and OS-65D software, including 8K BASIC & a 65F Floppy Disk Bootstrap PROM (D-5)	$749.00
Model 480 Backplane Board with complete set of male & female connectors (D-6)	$39.00
Model 495 Prototyping Board (D-6)	$29.00
Model 498 Card Edge Extender Board complete with connectors (D-6)	$29.00

Monitor PROMs

65A for serial interface 6502 systems for use on 400, 500, & 510 CPU Boards (D-12)	$29.00
65V for video interface 6502 systems for use on 400, 500, & 510 CPU Boards (D-12)	$29.00
68A Mod 2 for serial interface 6800s for use on 400 & 510 CPU Boards (D-12)	$29.00
68V Mod 2 for video interface 6800s for use on 400 & 510 CPU Boards (D-12)	$29.00
65F Floppy Disk Bootstrap PROMs for 6502-based 400, 500, & 510 CPU Boards (D-12)	$29.00

Note: All necessary PROMs are provided on kits, fully-assembled boards, and fully-assembled products. These PROMs should only be necessary for bare board purchasers, and those who would like to convert from serial to video or vice versa. The one exception to this is that the Floppy Disk Bootstrap PROMs must be purchased in conjunction with fully-assembled disk drives, when a disk is purchased as an expansion item for an existing system.

Parts

2513 Character Generator	$12.00
8T26 Buffers	$3.00
6850 ACIA	$15.00
6520 PIA	$10.00
S1883 UART	$10.00
K-1 Connector Kit including four sets of male & female molex connectors	$3.00
1408L8 D/A Converter	$10.00

Note: Parts are provided by Ohio Scientific as a service to our bare board purchasers.

Challengers, all fully assembled

C-SIT Challenger* System with Microterm Terminal and Monitor	$2,599.00
C-S1 Challenger* System without Terminal	$2,099.00
C-S2 Challenger* Video System	$2,499.00

*Delivered as Partially Populated Challenger II.

Disk Systems

C-D1 Single Drive Floppy Disk (D-5, D-13)	$990.00
C-D2 Dual Drive Floppy Disk (D-5, D-13)	$1,590.00
C-D74 Hard Disk (D-14)	$6,000.00

Challenger II Products with 8K BASIC in ROM, unless specified

C2-0 500 Board Fully Populated (D-7)	$298.00
C2-1 500-1 with 4K RAM & Small Cabinet (D-7, D-15)	$429.00
C2-8S 500-8 or Challenger II with 4K RAM (D-7, D-17)	$629.00
A1 Add Memory Management & Parallel Port (D-19)	$50.00
A2 2MHz Operation (D-19)	$50.00
A3 Configure for Disk Use (must have 16K RAM minimum) add Disk Bootstrap and delete 8K BASIC ROMs (D-19)	-$30.00
C2-8V Challenger IIV with 4K RAM (D-7, D-18)	$750.00
A4 Add Serial Port (D-19)	$50.00
A1 Add Memory Management & Parallel Port (D-19)	$50.00
A3 Configure for Disk Use (16K minimum). Add Disk Bootstrap and Delete 8K BASIC ROMs (subtract from above) (D-19)	-$30.00
A5 Add 128 x 128 Graphics (D-19)	$150.00
C2-4P Challenger IIP--includes 4-slot backplane, 8K BASIC in ROM, 4K RAM, Special 32 x 64 Display, Audio Cassette Interface, and Keyboard in Special Enclosure (D-7, D-16)	$598.00

Challenger IIV replaces Challenger 65V and has 440 Video Board.

Challenger IIP is a four-slot computer similar in appearance to the Sol-20 which is designed for direct competition with the Commodore PET

Challenger Accessories

CM-1 4K 1MHz Challenger Memory for use with ROM BASIC (D-1, D-21)	$125.00
CM-2 4K 2MHz Low-Power Memory (D-1, D-21)	$149.00
CM-3 16K 1.5MHz Ultra-Low-Power Memory (D-9, D-21)	$596.00
CA-6 Challenger Audio Cassette (D-2, D-22)	$99.00
CA-7 Fully Populated 430B Board (D-2, D-22)	$399.00
CA-8 560Z Subsystem (D-25)	$1,190.00

The CM-1 Memory Boards are for use with ROM BASIC machines, since BASIC in ROM can run only at 1MHz. CM-2 and CM-3 Memories should be used in large systems for low-power dissipation.

Challenger III Products

C3-8 Challenger III--6502A, 6800, Z-80, 16K RAM Crystal Control, Serial Port, and Floppy Disk Bootstrap (D-8, D-20)	$1,295.00
A-100 1-Megabyte Memory Management, Software Processor Switch, Swappable RAM, and Additional Parallel Port (D-21)	$150.00
A-101 Video Board and Video PROM (D-3, D-21)	$189.00
C3-0 510 CPU Board (for System Upgrades) (D-8)	$359.00
A-100 applies (D-21)	$150.00

Factory Direct Trade-ins are available for Challenger owners.

The Challenger III comes fully equipped for disk, but the price does not include the disk drive(s) C-D1 or C-D2. Challenger IIs with the A-100 option will take 15 to 30 days longer to deliver.

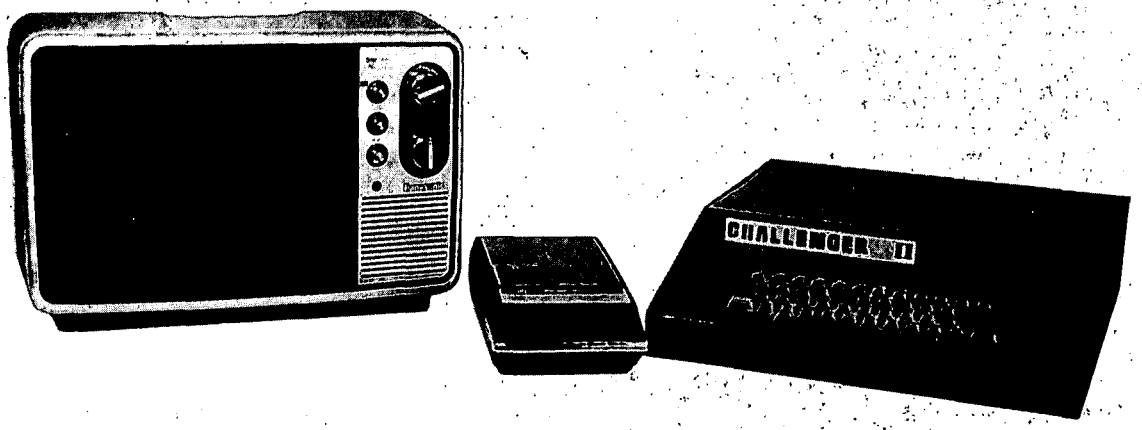
System Accessories

AC-1 Keyboard, Enclosure & Cable (D-22)	$149.00
AC-2 Cassette Recorder & Cables (D-22)	$59.00
AC-3 Video Monitor (D-23)	$159.00
AC-4 OKI-Data Model 110 with Cable & Interface (D-23)	$1,900.00
AC-5 OKI-Data Model 22 with Cable & Interface (D-23)	$2,900.00
AC-6 Micro-Term ACT-1 (D-24)	$550.00
AC-7 Hazeltine 1500 (D-24)	$1,095.00

Software: Paper Tape, Cassette, or Diskette

S-1 Introductory Software Package 12S-1 (paper tape only) (D-26)	$20.00
S-2 Assembler/Editor (D-27)	$35.00
S-3 8K BASIC (D-28, D-29)	$50.00
S-4 Extended Monitor (D-30)	$15.00
S-5 Life (D-26)	$10.00
S-6 Graphics Editor (no Paper Tape) (D-26)	$8.00
S-7 Tiny BASIC (D-31)	$10.00
OS-65D Diskette with copies of all software purchased by customer (D-32)	$15.00

Meet Challenger IIP from Ohio Scientific.



Unlike any other personal computer available today

Complete with BASIC in ROM and 4K RAM, Challenger IIP is the ideal computer for programs in BASIC.

BASIC is there the instant you turn the computer on with a full 32 x 64 character video display. Challenger IIP also comes with an Audio Cassette Interface for program storage. The user simply connects a Video Monitor or a TV via an RF Converter (not supplied) and the machine is ready to use.

Challenger IIP is ideal for both the home user who is new to computing or the experienced user who wants expansion capabilities. Challenger IIP comes with a four slot backplane and is expandable via the full Ohio Scientific product line, which includes 15 system boards offered in over 40 different versions.

Ohio Scientific has always maintained upward expandability from old models to new models, which is nice to know considering the rate at which technology is constantly improving. For example, Ohio Scientific's original 400 series products can be plugged right into the new Challenger IIP. And Ohio Scientific has 2 years of experience in building personal computers, so we're not new to this business unlike some of our competitors.

Complete with a full computer keyboard Challenger IIP comes fully assembled for $598 from Ohio Scientific.

Check the chart below and compare Challenger IIP with other BASIC in ROM computers. Unlike other personal computers, Challenger IIP has a much greater capacity for expansion and the capability to perform big computer functions with all of its big computer features.

	Ohio Scientific Challenger IIP	Other BASIC in ROM Computers
Processor	6502A	6502 or Z-80
Clock	1 or 2 MHz	slower
Display (Lines/Characters)	32/64	25/40 or 16/64
Keyboard	Full Computer (Capacitive Contact)	4 Function Calculator Type or Full Computer (Mechanical Contact)
Display Characters	256	128 or 64
Lower Case	Yes	No
Plotting	Yes	Yes
Audio Cassette Interface	Yes	Yes
BASIC	8K By Microsoft	some have only 4K BASIC
String Functions PEEK, POKE, User Machine Language Accessible	Yes	Not Always
Optional Assembler/Editor	Yes	Not Always
Disk Option Available Now	Yes	No
In Case Memory Expansion Ability	36K	Less
Expansion Boards Available Now	15	None

HOW TO ORDER

If you are ordering boards, kits, or assembled products and software to expand an existing system, please use the enclosed components order form. If you are ordering a computer system, please place all items you want installed on that computer or for use with that computer on the systems order form. Please place kits and bare boards and other accessories on a separate order form.

There is a definite advantage to buying accessories with the computer. That is, all components will come configured to work together. For instance, if you buy a serial terminal in conjunction with a computer from Ohio Scientific, the entire system will come ready to operate. Possibly the best way to buy a computer system from OSI is to purchase one of our standard computer packages which are described immediately preceding the price list and in our applications sections. Simply place all the items listed under the package on the systems order form. The other way to order a computer system from OSI is to specify your own custom configuration.

1. Choose the computer mainframe that you will be using. Please use the numbers appearing in the price list and specifications sheets. Your choices of computers are C2-O, C2-1, C2-8S, C2-8V, C2-4P, C3-8, or our Super Kit special.

2. Specify any options which you want on the CPU board that is in your computer. That is, if your computer has a 500 CPU board, you can specify the A1, A2, A3, or A4 options. If it is a C3-8, you can specify the A-100 and A-101 options.

3. Specify any additional memory. Your options here are the CM-1, CM-2, or CM-3. Ordering the CM-3 16K memory board can be a little tricky. First of all it requires a +12V power supply which is only available on eight-slot computers. Secondly, the Challenge II eight-slot computers can be configured so that all memories are on 16K boards. The advantage to this is a lower power consumption. To order C2-8V or C2-8S with all memory on one or more 16K memory boards, simply specify on the order form that all memory is to be on 16K memory boards, then subtract $149 from the retail price of the computer mainframe to delete the 4K of on-board memory, and add the full price of the CM-3 16K memory boards which you desire.

4. Select your mass storage device, such as a Challenger audio cassette, fully-populated 430 board, or single or dual drive floppy disk. Remember to specify the A-3 option on Challenger II machines when they are to be used in conjunction with floppy disks.

5. You may desire to order system accessories with the computer, such as a keyboard, cassette recorder, video monitor, line printer, or terminal. The advantage of ordering these from OSI is that you are assured of instant compatibility and that you get all the cables that you will require on a system.

6. You may wish to order additional software with your computer. Computer systems have 8K BASIC in ROM or come with BASIC in a disk operating system as standard equipment. The Assembler/Editor and Extended Monitor and other programs are optional extras.

Direct Orders:

Terms of Purchase: Payment by Bank Americard, Master Charge or certified check is preferred since we will not have to delay shipment of your order until a check clears.

Bank Americard and Master Charge orders are charged against your account on the day of shipment so that your money is not tied up on long delivery orders. C.O.D. orders are only accepted with a 20% prepaid downpayment.

Purchase Orders are accepted only from educational institutions and nationally recognized companies. Partial shipments will not be made against purchase orders, so, separate long delivery items from stock items.

Use accompanying order forms when possible. Place fully assembled product on a separate order from bare boards and kits. If at all possible, use the Challenger order form for system purchases.

Delivery:

Boards and kits: Stock to 30 Days typically
Assembled Products: (except as noted) 60 Days
Special Order Products: 90 Days

Unfilled orders can be cancelled without penalty if delivery is delayed beyond 120 days. Orders cancelled before this time are subject to a 10% restocking charge. Each order change after receipt of initial order will be subject to an additional $4.00 shipping and handling charge.

As you can see from this catalog, there are a lot of factors involved in customizing your own configuration, but the procedure is fairly straightforward.

Terms:
*Checks must clear BEFORE shipment is made. (Shipping charges $4.00)
* 20% deposits required on all C.O.D. orders.
* All orders shipped via insured UPS unless otherwise requested.
* Purchase orders accepted from large, well rated companies and schools only.
* Prices, specs., and terms subject to change without notice.

ORDER FORM

For Components, Accessories and Software

1. Name _____

Address _____ City _____

State _____ Zip _____ Phone _____

Shipping Address(if different) _____

City _____ State _____ Zip _____

2. Payment by: Date of Order: _____

Money Order ____ Personal Check ____ Certified Check ____

Master Charge ____ BankAmericard/Visa ____

Name as it appears on card _____

Exact number as it appears on card _____

Expiration date of card _____

Signature _____ Amount being charged _____

3. I would like to order the following:

Quantity	Product Number	Name of Product	Price	Total Price

Return to:

Ohio Scientific
216-569-3241
11681 Hayden Hiram, OH 44234

Merchandise total _____
TAX _____
Shipping Charges _____
Amount enclosed or charged _____
Balance due COD _____

Ohio Scientific 11679 Hayden Hiram, OH 44234
SMALL SYSTEMS JOURNAL