LIMITED WARRANTY

Ohio Scientific, Inc., 1333 S. Chillicothe Road, Aurora, Ohio 44202 (the "Warrantor") hereby warrants to the original purchaser that its hardware equipment will be free from defects in materials and workmanship for a period of ninety (90) days from the date of receipt by purchaser, when operated and maintained in accordance with Ohio Scientific's recommendations. This warranty includes power supplies and floppy disk drives. It specifically excludes terminals, video monitors, audio cassettes and keyboards not manufactured by Ohio Scientific.

Ohio Scientific warrants its software against media that is defective, such that it is not readable by the computer system, for a period of ninety (90) days from the date of receipt by purchaser. The software is thoroughly tested and thought to be reasonably bug-free when released. Ohio Scientific maintains a full staff of software experts, and will endeavor to correct any serious bugs that may be discovered in the software after release in a reasonable amount of time. However, this is a statement of intent and not a warranty or guarantee in such event. (Software sold with annual site licenses offers additional support commitments. See their contracts for details.)

You must have purchased the product from a duly authorized Ohio Scientific dealer, whose name appears in Ohio Scientific's current dealer listings, to qualify for the 90 day warranty. Ohio Scientific makes no other express warranty than that made above. Any implied warranty, including, but not limited to, the implied warranty of MERCHANTABILITY or fitness for a particular purpose, shall not be extended beyond the ninety (90) day period.

Ohio Scientific's obligation under the above warranty is limited to the repair of the product, without charge, if it is defective and has not been misused, carelessly handled, or defaced by repairs made or attempted by others, and it is returned to Ohio Scientific for repair. Ohio Scientific shall not be liable for any other loss or damage resulting directly or indirectly from the defect in the product including, but not limited to, incidental or consequential damages for lost profits, lost sales, injury to person or property, or any other incidental or consequential loss.

In the event that you desire to obtain performance of any warranty obligation, please return the product, in its original or other adequate packaging, to Ohio Scientific, Inc., or by prior arrangement to the dealer from whom you purchased the unit.

Ohio Scientific reserves the ultimate authority to determine what constitutes in-warranty repair in circumstances where circuit modification, abuse, misuse, or shipping damage occurs. If it is determined that the product is not under warranty, it will be repaired using Ohio Scientific's standard rates for parts and labor. Ohio Scientific will use its best efforts to repair the product within three weeks after receipt thereof. However, Ohio Scientific shall not be responsible for delays beyond its control such as, but not limited to, those caused by shipping or long delivery of replacement components.

The warranty contained herein is the only warranty which any Ohio Scientific dealer is authorized to give in conjunction with the product. Ohio Scientific shall not be bound by any other warranty made by the dealer to the purchaser. The support of such warranty or maintenance contract is the sole responsibility of the dealer offering the warranty.

When requesting performance under the terms of this warranty, the original purchase date, or date of purchaser's receipt of the product, must be established by means of a bill of sale, invoice, or other acceptable documentation.

This warranty gives you specific legal rights, and you may also have other rights which vary from state to state.

If there are any questions about this warranty, or if a complaint has not been answered by the dealer to your satisfaction, please contact:

OHIO SCIENTIFIC
1333 SOUTH CHILLICOTHE ROAD
AURORA, OH 44202

For your records:

Model Number ____________________

Serial Number ____________________

Date Purchased ____________________

Dealer ____________________

C4P
OPERATORS MANUAL

TABLE OF CONTENTS

SECTION	PAGE
1. GENERAL INTRODUCTION	1
2. VIDEO DISPLAY CONNECTION	2
3. CONNECTING THE FLOPPY OR CASSETTE SYSTEM	3
A. Cassette System	4
B. Floppy Disk System	4
4. STARTING THE MACHINE	5
A. Cassette System	5
B. Floppy Disk System	6
5. RUNNING A CANNED PROGRAM	9
A. Cassette System	9
B. Disk Based System	10
6. BASIC PROGRAMMING	12
7. GRAPHICS	27
8. SOUND	30
A. Tone Generator	30
B. DAC Use (via monitor)	31
9. STORING FILES ON CASSETTE OR DISKS	33
A. To Load Cassette: Programs into RAM	33
B. Saving Programs on Cassette	34
C. Use of Cassettes as a Data Storage Medium	35
D. Reading Data From Cassette Tape	35
E. To Write to Disk	36
F. To Read from Disk	37
G. Operating System Organization	38
10. ADVANCED FEATURES	40
11. JOYSTICKS AND KEYPADS	42
A. Joysticks	42
B. Keypads	47

SECTION	PAGE
12. AC REMOTE CONTROL, SECURITY	50
A. Appliance Control	50
B. Home Security	52
13. PARALLEL I/O	54
A. External Switches	54
B. PIA Registers	55
14. CONNECTION OF 16 PIN BUS DEVICES	59
A. CA-15 Board, the Interface Board	59
B. CA-20 Board, the Expander Board	59
C. CA-21 Board, the Parallel I/O Expansion Board	60
D. CA-23 Board, the EPROM Programmer	60
E. CA-24 Board, the Experimenter Board	61
F. CA-25 Board, the Accessory Interface	61
G. CA-22 Board, Analog I/O	61
15. MODEM AND TERMINAL COMMUNICATIONS	63
16. PRINTER COMMUNICATIONS	65
17. ADVANCED TOPICS	67
A. Plot Basic	67
B. Files	67
C. Home Control and Real Time Operating Systems	68
D. Real Time Clock	68
a. Time of Day Clock	68
b. Count Down Timer	69
c. Real Time Monitor, RTMON	69
d. A Greenhouse Example	71

APPENDICES **PAGE**

A. TROUBLESHOOTING AND MACHINE ORGANIZATION........................ 75

B. DETAILED A-15 BOARD PIN CONNECTIONS................................ 77

C. MEMORY MAP AND MINI-FLOPPY DISK ORGANIZATION 78

D. DISK BASIC STATEMENTS AND ERROR LISTINGS........................... 81

E. POKE AND PEEK LIST.. 87

F. PIANO KEYBOARD 92

G. DISK UTILITY PROGRAMS.. 93

- a. Delete 93
- b. Rename... 93
- c. Change 93
- d. Copy 95
- e. Create... 97

H. HEX TO DECIMAL TUTOR, CONVERSION TABLES 99

I. ASCII CONVERSION CHART 108

J. CHARACTER GRAPHICS AND VIDEO SCREEN LAYOUT........................ 110

K. OS-65D USER'S GUIDE 118

L. MACHINE MONITOR, 65V.. 125

M. USR(X) FUNCTION 127

- a. Using the Assembler.. 134

N. EXECUTING A DISK RESIDENT MACHINE LANGUAGE PROGRAM.................. 136

O. INDIRECT FILES.. 139

P. BEXEC* 142

Q. I/O DISTRIBUTION 144

INDEX 149

SECTION 1

GENERAL INTRODUCTION

You are using a state-of-the-art Ohio Scientific computer system which brings cost effective processing to the popular computing field. The high instruction rate and expandable architecture of the OSI bus bring computing power within the reach of home and office while a wide range of software supports the various applications, such as recordkeeping, security systems, education, computation, and entertainment.

This manual is a general guide to your computer's features. It gives applications and examples to aid you in your programs and applications. We hope it will lead you to consider new ways to benefit from your computer's features. More detailed manuals from OSI cover the definitive use of option boards or operating system and software details. However, the material in this manual should be sufficient to show most of the features you will need in common applications.

To aid in quick reference, the features and functions referred to throughout the manual are contained in separate appendices and listed in the index.

The C4 is a self-contained computer and a highly reliable system. To prevent abuse and assure this level of performance, please follow these instructions:

1. Insure that the power outlet is part of a 3-wire grounded 11ØV AC system. If the existing system is a two wire system, then a securely attached wire from the computer's cabinet must be run to a clamp-on a cold water pipe. *Only then* may a two wire adapter be used on the computer's power cable.

   Failure to follow these precautions may present a shock hazard and cause computer damage from static discharges. Such damages are specifically not covered under the Warranty.

2. Connect the system together with the cables provided according to Figure 1. Press the cable connectors firmly for good contact. If a monitor provided by OSI is not used, then read the section on "Video Display Connection."

3. Put the floppy disks aside until needed. These disks should be stored upright in their sleeves in a clean, dry area. They should not be bent, folded or twisted.

   Do not use paper clips or other fasteners on the disks.

   Mark the disks with felt tip pens, only. Ball point pens and pencils will dent the disks.

   Do not touch the inner disk surface, as body oils and dirt will degrade performance.

   Keep disks away from magnetic fields (magnets, motors, computer power supplies, etc.). Do not leave disks on the cabinet tops, as the magnetic fields and temperature can be excessive.

   Disk temperature should be maintained between 1Ø°C—5Ø°C (5Ø°F—125°F). If the temperature is comfortable for a person, the disk will not suffer either! Storage in direct sunlight, adjacent to heating vents, or in a car trunk should be avoided.

   The disks must never be left in the disk drives when any part of the system is turned OFF or ON.

SECTION 2

VIDEO DISPLAY CONNECTION

There are three different methods of attaching a video display to the C4P computers. These are outlined as follows:

1. Preferred method—connect the supplied computer video cable to the high impedance (Hi-Z) input of a closed-circuit TV video monitor. Ohio Scientific offers a color television set, modified for video monitoring. Ohio Scientific also offers the Model AC-3P 12" black and white monitor. Both are ideal for this application. The units double as television receivers when the video cable is disconnected.
2. Connect the supplied computer video cable to an "RF modulator" which is, in turn, connected to a standard television's antenna terminals. RF Modulators are inexpensive and allow you to use almost any television with the computer. They are sold in kit form.
3. Have a standard AC transformer-operated television modified to accept direct video entry. This requires special safety precautions.

CLOSED-CIRCUIT VIDEO MONITOR CONNECTION

1. Refer to Figure 1. Attach the supplied video cable to the computer as shown.
2. Connect the other end of the cable to the high impedance input of the video monitor. The AC-3 monitor has a Hi-Z RCA-type phono jack input. On other monitors, a high impedance—low impedance selector switch is sometimes present, or there may be two or more inputs. Consult the manufacturer's instructions.
3. Observe the manufacturer's power recommendations. If the monitor has a 3-wire grounded plug, connect it to a properly grounded 3-wire AC outlet.
4. Turn on the computer and monitor.
5. Allow the monitor to warm-up. The screen should be filled with random graphics characters, alphabet, etc.
6. If necessary, adjust the VERTICAL and HORIZONTAL controls to obtain a stable picture.

RF MODULATOR/STANDARD TV CONNECTION

1. Refer to Figure 1. Review the manufacturer's instructions included with the RF modulator.
2. Connect the computer video cable to the computer as shown.
3. Connect the video cable to the RF Modulator.
4. Connect the modulator to the television's antenna terminals (consult modulator instructions).
5. Plug in the television and computer.
6. Turn on the computer, television, and modulator (consult modulator instructions).
7. At this point, the proper TV channel must be selected and the television's fine tuning adjusted as necessary (consult modulator instructions).
8. When the television warms up a screen filled with random graphics characters should be observed. If the picture is not stable, adjust the television's VERTICAL or HORIZONTAL controls as needed.

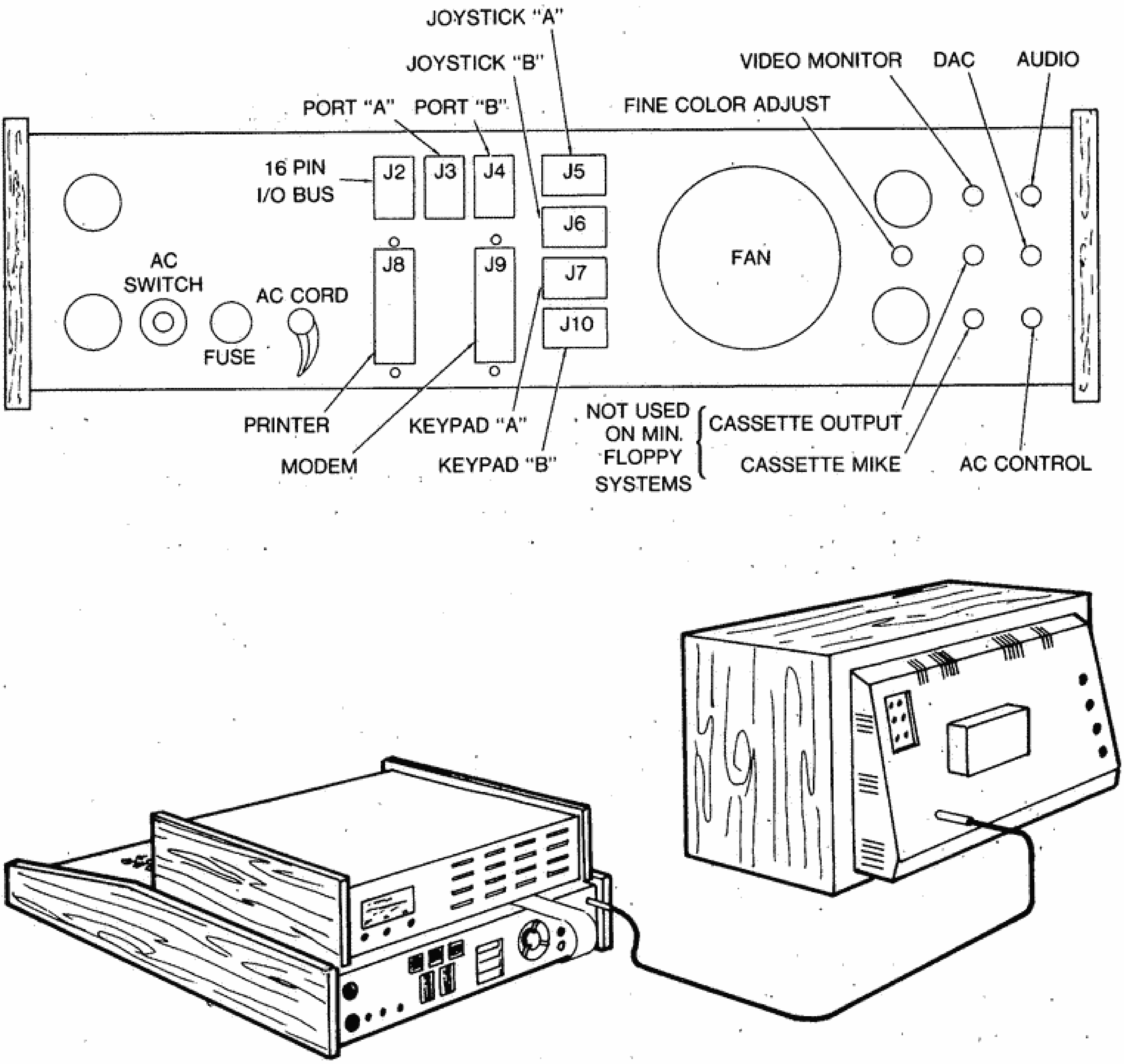


Fig. 1 C4P Back Panel and Video Interconnection

SECTION 3

CONNECTING THE FLOPPY OR CASSETTE SYSTEM

CASSETTE SYSTEM

The manual to this point has made no differentiation between a C4P (cassette) computer and a C4P MF (Mini-Floppy). Hereafter, all information pertinent to the cassette model will be marked with the same border as that on this page.

The cassette provides an economical bulk storage medium, though the data transfer rate is considerably lower than the disk's rate. The internal configuration of computer components is slightly different than the mini-floppy configuration. Externally, the computer and accessories should agree with Figure 2.

The cassette recorder should be a medium price audio tape recorder. If price is indicative of quality, then $35-$5Ø would be a price guide. Volume and tone controls should be set at mid-range. If 11ØV AC is not used for the recorder power, be sure to use fresh batteries. (Speed variations due to weak batteries can create errors.)

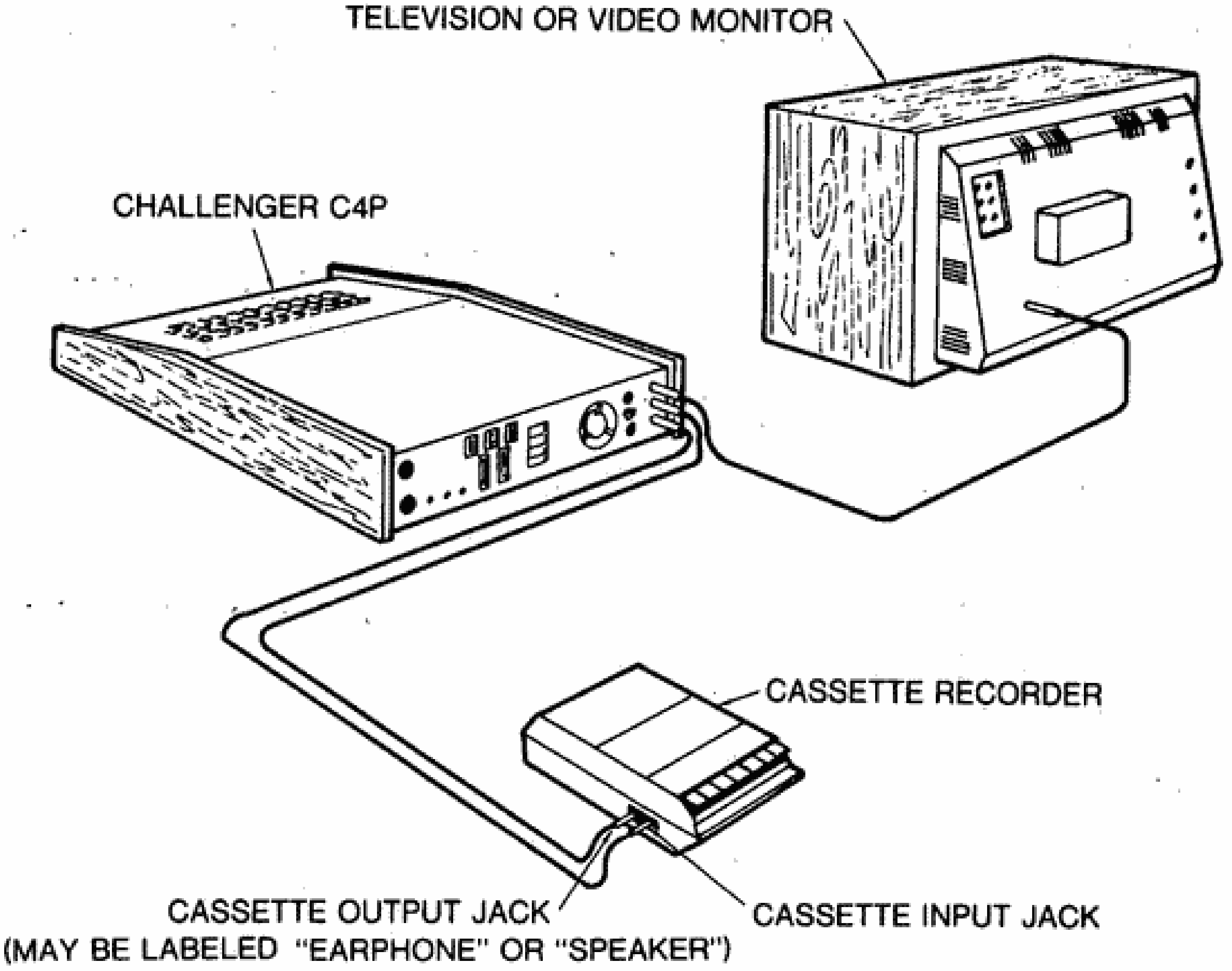


Fig. 2 Computer and Accessories

The cassette based systems are not permitted to use back panel connections J2 and J4 and J8 and J9.

FLOPPY DISK SYSTEM

1. The mini-floppy disk provides a large performance benefit for the relatively small investment above a C4P (cassette) system; the chief benefits of the C4P MF are file handling and high speed data transfer.
2. Floppy disk drive units will be connected at the factory. The removal of packing material, done earlier, is the only preparation step required. Externally the interconnection of computers and accessories should agree with Fig. 1.

SECTION 4

STARTING THE MACHINE

CASSETTE SYSTEMS: COLD START

The precautions and discussion given in the main part of this manual for the C4P MF (mini-floppy) system, still apply. As a reminder these are outlined.

1. Assemble the computer system according to Figure 2. The use of OSI supplied cables will assure reliable and firm connections between units.
2. Turn on the computer. The switch is on the back panel.
3. Turn on the monitor. (Only OSI modified monitors or RF modulators should be used. Damage produced by unauthorized monitors will void all warranty coverage.)
4. Turn on the cassette recorder power.
5. Press the "BREAK" key.
6. Rewind the cassette so that the tape "leader" is visible on the take-up spool. OSI software will be supplied on high quality tapes. Use of low quality tapes will cause erratic performance and excessive recorder wear.
7. Respond to the terminal screen message

   C/W/M?

   by pressing the "SHIFT LOCK" key down and then respond

   C <RETURN>

   If the "SHIFT LOCK" key is not depressed, the keyboard message will not be understood by the computer.
8. When the computer requests

   MEMORY SIZE?

   just press the "RETURN" key.
9. The computer will next ask

   TERMINAL WIDTH?

   Again, press the "RETURN" key.
10. The prompt

    OK

    should appear at the bottom of the screen. If it does not, repeat steps 1 thru 1Ø.

This prompt indicates the BASIC program is ready for operation. The cassette supported C-4P is a BASIC-in-ROM system, having a 6-digit BASIC stored in read only memory (ROM).

Section 5, Running a Canned Program, will introduce some OSI software and a demonstration program. Cassette system users skip over the disk oriented material in this section and proceed directly to Section 5.

FLOPPY DISK SYSTEMS

POWER UP

a. Check that the system is connected according to Figure 1 and the related instructions. Make sure that there is clearance for ventilating air in the back of the C4P system.

b. Plug in power cords.

c. Turn on power on the back of the keyboard console.

d. Turn on floppy disk power (switch is on rear of disk drive).

e. Turn on CRT and any other accessories.

f. Depress the SHIFT LOCK key. Now press the "BREAK" key on the keyboard.

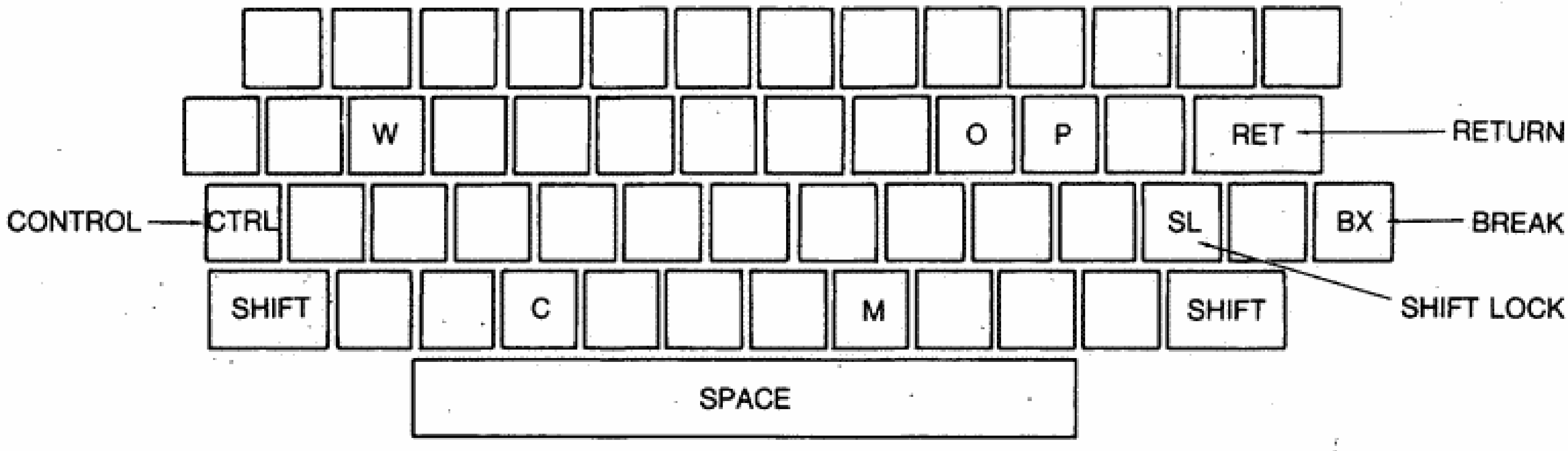


Fig. 3 Keyboard Layout

g. Remove the disk labeled "Customer Demo Disk" from its covering sleeve. Carefully insert the disk with right thumb on the label. Keep the disk label on the top side. Refer to Fig. 4.

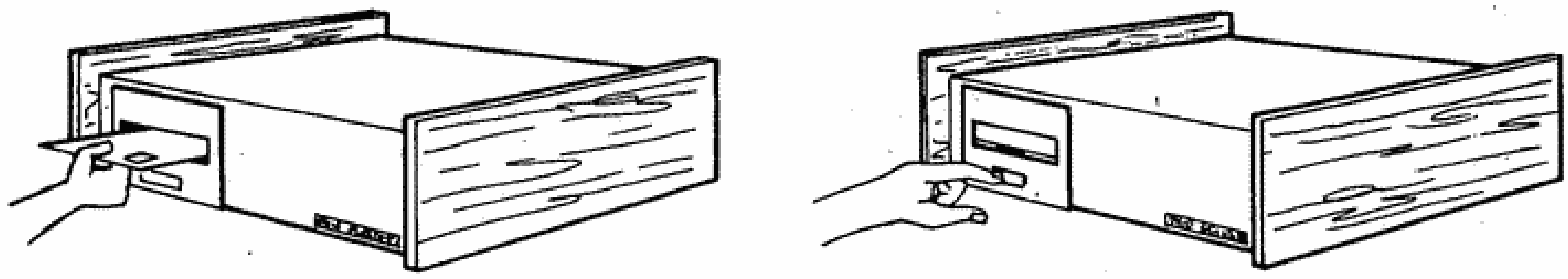
The disk should be inserted firmly until a click is heard or slight resistance is encountered. Close the door on the disk drive.

h. MAKE SURE THE "SHIFT LOCK" KEY IS DEPRESSED. When the computer responds "H/D/M?" on the CRT (television screen), type

D

The program will automatically be loaded into the computer from the disk.

This disk will repeat its program endlessly.



Inserting a Disk. To remove a Disk.

Fig. 4 Disk Placement

NOTATION

"Throughout the remainder of this manual, the following notation conventions shall be employed:
The shorthand notation:

<RETURN>

will be used instead of writing "Press the "RETURN" key." Do not type the brackets or the word RETURN letter-by-letter.

Blank spaces will be indicated by a blank in the typing, such as

1Ø GOTO 5 <RETURN>

rather than writing

1Ø <SPACE> GOTO <SPACE> 5 <RETURN>.

When the operator is to enter something from the keyboard, his responses will be underlined or in brackets (the messages produced by the C4P will not be underlined). In the following example,

FUNCTION?

UNLOCK <RETURN>

The C4P ask the question "FUNCTION?" and the operator's repsonse would be to type out "UNLOCK" (note that all of the letters are capitalized) and then a carriage return.

DISK PROGRAMS

The Customer Demo Disk contains a continuously sequenced animation, showing the power of the OSI C4P computer and its software. This manual, will show how to adapt some of these programs to individual purposes. Similar programs are available from OSI dealers. When finished, remove the disk from the drive and store the disk in its protective sleeve. To use another disk, press

<BREAK>

insert the new disk in the disk drive, then repeat Step g of the previous section.

The "Dealer Demo Disk" contains the programs

a. Graphics Demo, an image generator which shows the tools of animation and graphing.

b. Plane Banner, a simulated airplane made from the C4P's Character set. A wide variety of shapes is possible.

c. Random Square, an animated pattern generator to show the color range available.

d. Kaleidoscope, a continuously changing pattern to illustrate the variety of symbols available.

e. Space Wars, a game to pit your starship against the enemy empire.

f. Hectic, a ricochet simulation game. Both scientific problem simulation and games can use these techniques.

g. Tiger Tank, a combat game to show real-time player interaction.

h. Set Time, a clock function which does more than keep time. This program can be used to control other programs.

i. AC Demo, a home light and appliance control program. With the external lamp modules attached, the pictures on the CRT screen will be echoed by the device behavior. (Note that remote module switches must be properly set to use this program.)

These programs can be readily adapted to individual use. After becoming familiar with the C4P system, the operator will be able to list these programs and extract the examples for his special purposes. These well written examples provide programming lessons and power for sophisticated programs.

In order to access specific programs on the Customer Demo Disk (ONLY), the operator/user must be provided with a "menu" of programs from which to choose. To examine the directory of programs on this disk, press

<CONTROL> <SHIFT>

simultaneously. These keys are adjacent to each other on the left of the keyboard. These keys must be held down for several seconds, as the program checks them infrequently.

Upon being presented with the menu of programs, respond to the request for response by typing

PASS

to immediately bring up the BASIC program.

To run a simple program stored on the disk, enter

RUN "DIR" <RETURN>

The DIR program will ask

LIST ON LINE PRINTER INSTEAD OF DEVICE #2?

Answer

NO <RETURN>

at which time a listing of the directory appears on the screen. Each program stored on the disk is listed by name and the numbers of the disk tracks it occupies.

An alternative way to run the program DIR is by specifying the track on which it is resident. On the Dealer Demo Disk, DIR is resident on track 11. The alternate method to RUN the program DIR is to enter

RUN "11" <RETURN>

at which time the sequence displayed when RUN "DIR" was typed will repeat.

Once in memory, the program can be RUN yet again by typing

RUN <RETURN>

since it need not be loaded from disk again.

POWER DOWN

When ready to turn the system off:

a. Remove the disk from the disk drive by pushing the rectangular button below the disk door. Then remove the disk, placing it back in its sleeve.

b. Turn off peripheral devices, if any.

c. Turn off CRT. (Video monitor)

d. Turn off disk drive (for disk systems only).

e. Turn off computer power (back of keyboard console) last.

The hardest part of using the C4P MF computer has just been completed. From hereon, care of the computer and orderly handling of materials will pay for itself in reliability and enjoyment of the C4P system. Now go on to using the system in some applications!

SECTION 5

RUNNING A CANNED PROGRAM

CASSETTE SYSTEMS

In Section 4, the procedure to turn on the C4P Computer was covered. Now the power of the OSI Software available to support the computer will be shown by running a demonstration program.

1. Turn on the computer and bring up BASIC, as described in Section 4.
2. Place the demonstration cassette (Marked SCX-1Ø4, C-2-4P/C4P Sampler) in the tape recorder which is connected to the system as shown in Fig. 2
3. Turn on the recorder power.
4. Type

   LOAD

   but DO NOT press <RETURN> yet.
5. Press the PLAY switch of the recorder. When the tape begins to move past the leader, as indicated by brown tape moving off the left hand spool and winding onto the right, then press the computer's

   <RETURN>

   key. After a few symbols appear on the screen, then a listing of the program will appear on the screen. The first program on the cassette will take approximately 3 minutes to load. Programs requiring 4K of memory will load in approximately 3 minutes, those requiring 8K of memory will require 5 minutes.
6. When the program is loaded, the message

   ?S┘ ERROR

   OK

will appear on the screen. Now, stop the tape or the next program on the tape will be loaded into memory over the program which was loaded first.

7. Press

   <SPACE>

   then

   <RETURN>
8. The program listing may be examined by typing

   LIST <RETURN>
9. To execute the program, type

   RUN <RETURN>

at which time, the program will prompt the operator through the first program on the tape, a Math Tutorial. Other sample programs on the tape may be examined by repeating steps 4 to 9, after going through the startup procedure of Section 4. The Cold Start is necessary to return to the BASIC program control.

10. Rewind the cassette and return it to safe storage before powering down the tape recorder and computer.

Each program on cassette is separated by approximately 1Ø seconds of blank tape. If the tape is not rewound after loading a program, it will be positioned to load the next program.

The programs recorded on the demonstration cassette are:

Side I: "Basic Math" is an educational quiz program that gives addition, subtraction, multiplication, and division problems.

"Checking Account" will help balance the checkbook. Just give the computer the initial balance and check amounts and let the computer do the work.

"Trig Tutor" explains and diagrams three trig functions: sine, cosine, tangent. The computer then tests comprehension of these functions with a quiz.

"Star Wars" is an arcade-type computer game. The player moves the cross-hairs around the screen trying to draw a bead on the target ship.

Side II: "Counter" is a combination of educational game and cartoon for youngsters learning to count from one to ten.

"President's Quiz" asks 2Ø historical questions about various presidents.

By using the tape counter on the recorder, the tape can be positioned to return to any program. When the cassette is turned to Side II to load the program "Counter," a 2Ø-3Ø second delay occurs before listing begins on the screen.

These programs provide usefulness in application and serve as models of well written software. A listing of available OSI Software, continually expanded and updated, is available from OSI dealers.

DISK BASED SYSTEMS

Several disk based programs have already been reviewed in previous sections. The disk labeled OS-65D, described following the procedure of Section 4B, "Starting The Machine," presents a menu display on the screen. When the standard OS-65D development disk is loaded, the following text is displayed on the screen:

```
BASIC EXECUTIVE FOR
OS-65D V3. N
MO, DAY, YR RELEASE
FUNCTIONS AVAILABLE:
COLORS-TEST PATTERN TO ADJUST COLOR MONITORS
CHANGE-ALTER WORK-SPACE LIMITS
DIR-PRINTS DIRECTORY
UNLOCK-UNLOCKS SYSTEM FOR END USER MODIFICATIONS
FUNCTION?
```

This menu offers four program choices. COLORS, the first choice, presents a test pattern to adjust the color video monitor controls, if needed. If the second choice, CHANGE, is selected, the computer will automatically LOAD and RUN a program by the name of CHANGE. If the response DIR is entered, the computer will LOAD and RUN a program named DIR. If the response is UNLOCK, then the system is unlocked. This allows the user to assume control of the system with the capability of entering and listing new programs in the workspace. The response UNLOCK places the system in the BASIC immediate mode and displays the prompt OK.

For now, focus on the program DIR. This program prints a directory of the files present on the diskette. If the response to the query FUNCTION? is DIR, the computer will ask

```
LIST ON LINEPRINTER INSTEAD OF DEVICE #2?
```

Responding NO will cause the following output to appear on the screen:

```
              OS-65D VERSION 3.N
                -DIRECTORY-
FILE NAME                   TRACK RANGE
OS-65D3                           Ø-12
BEXEC*                           14-14
CHANGE                           15-16
CREATE                           17-19
DELETE                           2Ø-2Ø
DIR                              21-21
DIRSRT                           22-22
RANLST                           23-24
RENAME                           25-25
SECDIR                           26-26
SEQLST                           27-28
TRACE                            29-29
ZERO                             3Ø-31
ASAMPL                           32-32
COLORS                           33-33
C-ASM1                           37-37
C-ASM2                           38-38
COMPAR                           39-39
      46 ENTRIES FREE OUT OF 64
OK
```

Some of the files of this directory listing will be discussed in detail in Appendix G. The files listed contain utility programs written in BASIC. Note that two of these programs, CHANGE and DIR, were introduced on the previous page in the menu. In addition to listing the names of the programs on the diskette, the directory tells where they are located on the diskette. For example, the program DIR is located on track 21 and is one track long while CHANGE is a 2 track program starting on track 15. (Each diskette has 4Ø tracks, numbered Ø through 39.)

Any of the BASIC programs on this disk can be run by responding UNLOCK to the query FUNCTION? and then entering the command RUN "NAME" where NAME is the name of the program or the number of the first track where it is stored. For example, either of the commands RUN"DIR" or RUN"21" would run the program DIR.

Most of the applications diskettes do not offer the user the option of unlocking the system. On these diskettes programs are run by entering the appropriate response when the menu is displayed.

The use of mini-floppy diskettes for storing programs will be discussed in detail in section nine.

SECTION 6

BASIC PROGRAMMING

The applications programs provided on the customer demo disk have been used to demonstrate the power of the OSI C4P system. The next step is to write personal programs in a powerful but simple language. *BASIC* is such a language.

An excellent book by Dwyer and Critchfield, *BASIC and the Personal Computer*, is available from OSI dealers. However, the information in this manual will suffice to teach some simple programs. This section is not intended to cover all of BASIC. Instead, it is to show extensions and differences of OSI's BASIC that the user should know. A few simple examples are included to familiarize the new users with applications.

For Cassette Based systems:

1. Turn on the computer power and the video display console.
2. When the display has warmed up, press

   <BREAK>

3. In response to the query

   C/W/M?

   refer to steps 7-10 on page 5 for procedure

   BASIC, as indicated by the prompt

   OK

   is now ready to operate.

For Disk Based systems:

First, turn on the OSI C4P computer. Remember

1. Turn on the computer power first and the floppy disk's power second (power switches are located on the rear panel; see Figure 1).
2. Turn on the video display console.
3. Press <BREAK>.
4. Insert the minifloppy disk marked simply "OS-65D 3.N".
5. Verify that the shift lock key is down. Press D on the keyboard.
6. Respond to the question

   FUNCTION?

   by typing

   UNLOCK <RETURN>

   (As established under "2.Notation," Section 4, B, the operator's entries will be underlined for emphasis.)

Now clean out the work space (memory where the program is running) by responding to the BASIC prompter

OK

by typing

NEW <RETURN>

This will erase the old programs which occupied the available memory. Next type

LIST <RETURN>

to verify that no programs are present.

CALCULATOR MODE (IMMEDIATE MODE)

As an example of one of the easiest forms of BASIC math operations, type the line below

PRINT 5+3 <RETURN>

(Remember underlined quantities are entered by the operator.) The computer will return the answer

8

For brevity, the question mark, "?" can also be used in place of PRINT as

? 5+3 <RETURN>

The result is the same. This calculator-like function is called the immediate mode of operation. It can be used like a scientific calculator.

PROGRAM MODE

Now repeat this program with the input and the output controlled by the computer (program mode). Type

1Ø ? 5+3 <RETURN>

or

1Ø PRINT 5+3 <RETURN>

Because of starting the line with a number, the computer will await any further numbered lines before performing the required calculations. This is the first program or set of instructions (in BASIC)! When ready to have the calculations run, type

RUN <RETURN>

The C4P will now execute the one line program that was just entered. The answer is, as before,

8

The numbering of lines (also called "labeling" for "statements") may be used to perform many instructions consecutively. It is a good practice to number statements as 1Ø, 2Ø, 3Ø, . . . , leaving room for easy future addition of lines. Be careful to arrange the lines in the order in which they are to be performed. The clarity and the usefulness of the previous program will be improved by allowing input to the computer when the program is run.

To prompt the program user, quotation marks are placed around words to be printed on the videomonitor when the statement is performed. The name of the variable to be entered follows the prompting quote, separated by a semi-colon.

Intermediate variables, with convenient names (which *do not* include words reserved for use by BASIC, such as FOR and WAIT-see the appendix) should be chosen to keep the program statements simple. The final statement, END, in line 5Ø in this example, indicates to the computer that this is the end of the program.

Write out this example program. Type

1Ø INPUT "ENTER THE FIRST NUMBER";A <RETURN>

2Ø INPUT "ENTER THE SECOND NUMBER";B <RETURN>

3Ø SUM=A+B <RETURN>

4Ø PRINT "THE SUM IS";SUM <RETURN>

5Ø END <RETURN>

In case of a typing mistake, simply pressing

<RETURN>

and retyping the line will force the error to be thrown out. If a long line has been typed, this is inconvenient. Pressing the keys, SHIFT and O simultaneously, as

<SHIFT O>

will cause the last character typed to be removed. In disk based BASIC, the last character will simply disappear. In Cassette BASIC-IN-ROM, the <SHIFT O> will cause an underline symbol to be printed, rather than erase the deleted character. The statement

1Ø PRX _INT "HELP"

would appear as

1Ø PRINT"HELP"

The correction could be checked by doing a

LIST 1Ø

command, showing the symbol X has been truly deleted. Cassette BASIC-IN-ROM error message codes differ from those given for disk based BASIC. Lists of error codes for both versions of BASIC are given in Appendix D.

When ready to run the program that has just been entered, type

RUN <RETURN>

the message in between quotes in line 1Ø will appear as

ENTER THE FIRST NUMBER?

The BASIC program follows the message by a ? to indicate an operator entry is expected. Respond by typing a number, then a <RETURN>, such as

5 <RETURN>

The computer will inquire again

ENTER THE SECOND NUMBER?

Type the second number in the same manner, such as

3 <RETURN>

The computer will respond by printing

THE SUM IS 8

Now type

RUN <RETURN>

The computer will again RUN the program and ask for numbers.

The above examples illustrate that the *BASIC* language is algebraic in form, with simple input and output statements. By numbering the statements, the order of execution of program statements is arranged. Upon typing

RUN <RETURN>

the ordered sequence of statements is executed. Note that the words appearing between the quotation marks will be printed on the CRT screen as prompting statements.

Multiple calculations can be performed by using loop statements. For example, computation of the squares of the numbers from 1 to 6 inclusive could be done by the following program

```
1Ø REM SQUARES OF NUMBERS PROGRAM
2Ø FOR I=1 TO 6
3Ø SQ=I*I
4Ø PRINT "THE SQUARE OF";I;"IS=";SQ
5Ø NEXT I
6Ø END
RUN
```

Remarks are denoted by the word REM. Remarks are used for program clarity and are not executed by the BASIC program. The writing of <RETURN> at the end of each line has been discontinued to make the program look less cluttered. The operator must still enter <RETURN> when entering the program from the keyboard.

To illustrate another method of performing the same operation, type

```
3Ø SQ=IΛ2
```

(The up-arrow is entered by <SHIFT N>). This will replace the old Statement (3Ø SQ=I*I) and will also run but will yield slight variations in the answers. This is due to the algorithm (method of calculation) which OSI BASIC uses. The up-arrow, Λ, means "To the power of." It involves the use of algorithms instead of merely multiplying.

To do a computation until a desired value is found involves the use of the less than, greater than, or equal (<, >, =) signs. An example might be to find the smallest integer whose square exceeds 6ØØ.

```
 1Ø REM FIND THE INTEGER X SUCH THAT
 2Ø REM (X−1) Λ 2 IS <6ØØ AND
 3Ø REM (XΛ2) IS> 6ØØ
 4Ø X=1
 5Ø SQ=X*X
 6Ø IF SQ> 6ØØ THEN GOTO 9Ø
 7Ø X=X+1
 8Ø GOTO 5Ø
 9Ø PRINT "THE LOWEST INTEGER X WITH XΛ2> 6ØØ IS";X
1ØØ END
```

Statement 6Ø is a conditional statement. If it is satisfied, i.e., SQ> 6ØØ is true, then the next statement to be executed is number 9Ø. If SQ> 6ØØ is false, the next statement in order, number 7Ø, is executed. This branching between statements permits a program to be modified, depending on the result of a calculation. This branching technique makes high speed decisions possible, based on the data which is evaluated by the computer. When the conditional branch to statement 9Ø is made, the answer is then printed.

CHARACTER MANIPULATION

In addition to handling numbers, OSI BASIC language can also be used to manipulate characters. For example, to read in a string of characters, type

```
1Ø INPUT "YOUR CHARACTERS ARE";A$
```

The dollar sign after the variable name implies that this is a character string, rather than a number, per se.

Several character string operations are possible. It is possible to print out the characters by typing

```
2Ø PRINT A$
```

To run the program at this point, type *RUN*, then respond to

```
YOUR CHARACTERS ARE?
```

by typing

```
NOW <RETURN>
```

and see the result in the print out

```
NOW
```

If

```
NOW IS THE TIME <RETURN>
```

had been typed the character string

```
NOW IS THE TIME
```

would have been printed. This last string consists of 12 letters and the three blanks in between words. These strings can be operated upon with string operations.
One of the possible string operations is counting the string length

```
3Ø L=LEN(A$)
```

Therefore, the program

```
1Ø INPUT "WHAT ARE YOUR CHARACTERS";A$
2Ø PRINT A$; " WERE READ IN "
3Ø L=LEN(A$)
4Ø PRINT "THERE WERE" ;L; "CHARACTERS"
5Ø END
```

will read in the character string, echo the characters for verification, and print the character count. (BASIC expects 72 or less characters to be input at any time.) Entering "LONG" will echo "LONG" and report four characters.

Other useful string operations are picking out the leftmost I characters in a string. For example, the leftmost character in the string A$ is found via

```
1Ø L$=LEFT$(A$,1)
```

The two lefthand characters in the string A$ are

```
1Ø L$=LEFT$(A$,2)
```

Similarly, the rightmost two characters in the string A$ are

```
1Ø R$=RIGHT$(A$,2)
```

Likewise, the midrange J characters which start from the Ith one are

```
M$=MID$(A$,I,J)
```

Thus, the second, third and fourth characters of the string A$ are given by

```
M$=MID$(A$,2,3)
```

For example, the program

```
1Ø A$="FRIDAY"
2Ø PRINT MID$(A$,2,3)
```

will result in the output

RID

Now enough information has been presented to write a simple two person hangman type game. Let the first person type a three letter word. The computer will then erase the screen. The second person will try to guess the letters. If the player fails to guess in six tries, the first player wins.

```
 10 REM GUESSING GAME
 20 INPUT "PLAYER #1 ENTER A 3 LETTER WORD";A$
 30 FOR I=1 TO 32 : REM CLEAR
 40 PRINT          :REM THE
 50 NEXT I         :REM SCREEN
 60 COUNT=0        :REM COUNT IS CORRECT GUESS COUNTER
 70 TURN=0         :REM TURN COUNTS TOTAL GUESSES
 80 INPUT "YOUR ONE LETTER GUESS IS";B$
 90 IF LEFT$(A$,1)=B$ THEN PRINT LEFT$(A$,1)
100 IF LEFT$(A$,1)=B$ THEN COUNT=COUNT+1
120 IF RIGHT$(A$,1)=B$ THEN PRINT RIGHT$(A$,1)
130 IF RIGHT$(A$,1)=B$ THEN COUNT=COUNT+1
150 IF MID$(A$,2,1)=B$ THEN PRINT MID$(A$,2,1)
160 IF MID$(A$,2,1)=B$ THEN COUNT=COUNT+1
170 TURN=TURN+1
180 IF COUNT=3 THEN GOTO 300
190 IF TURN=6 THEN GOTO 600
200 GOTO 80
300 PRINT "YOU WIN, THE WORD WAS";A$
310 GOTO 700
600 PRINT "YOU LOST, THE WORD WAS";A$
700 END
```

Of course, if a player gets one letter correct, it is possible to cheat by re-entering that letter three times, but then, this was just to try out the ideas. A program does what it is told to do, not necessarily what is desired for it to do.

For complicated programs, a picture is usually drawn of the thought or decision process. This picture is called a flow chart. For the previous program, the flow chart in Fig. 5A & B applies:

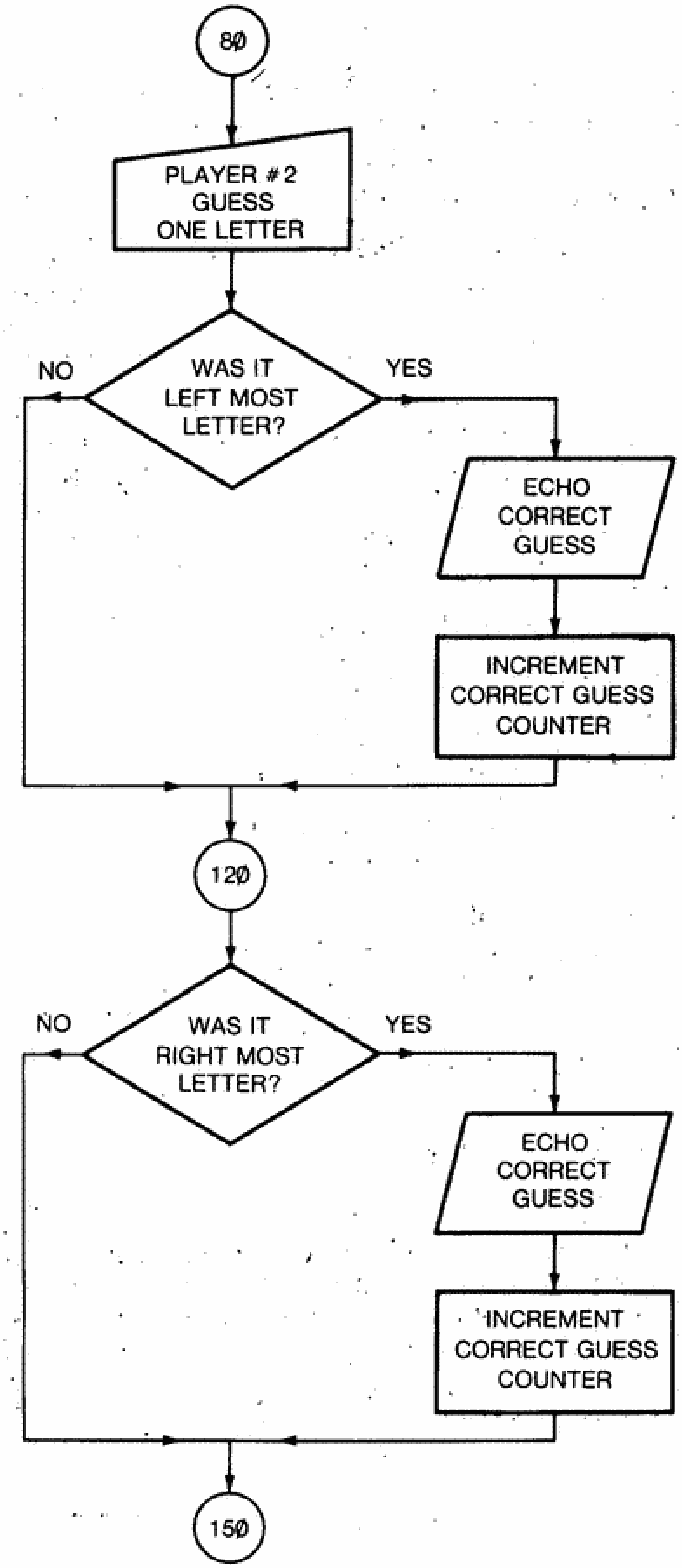


Fig. 5A Flow Chart (8∅ to 15∅)

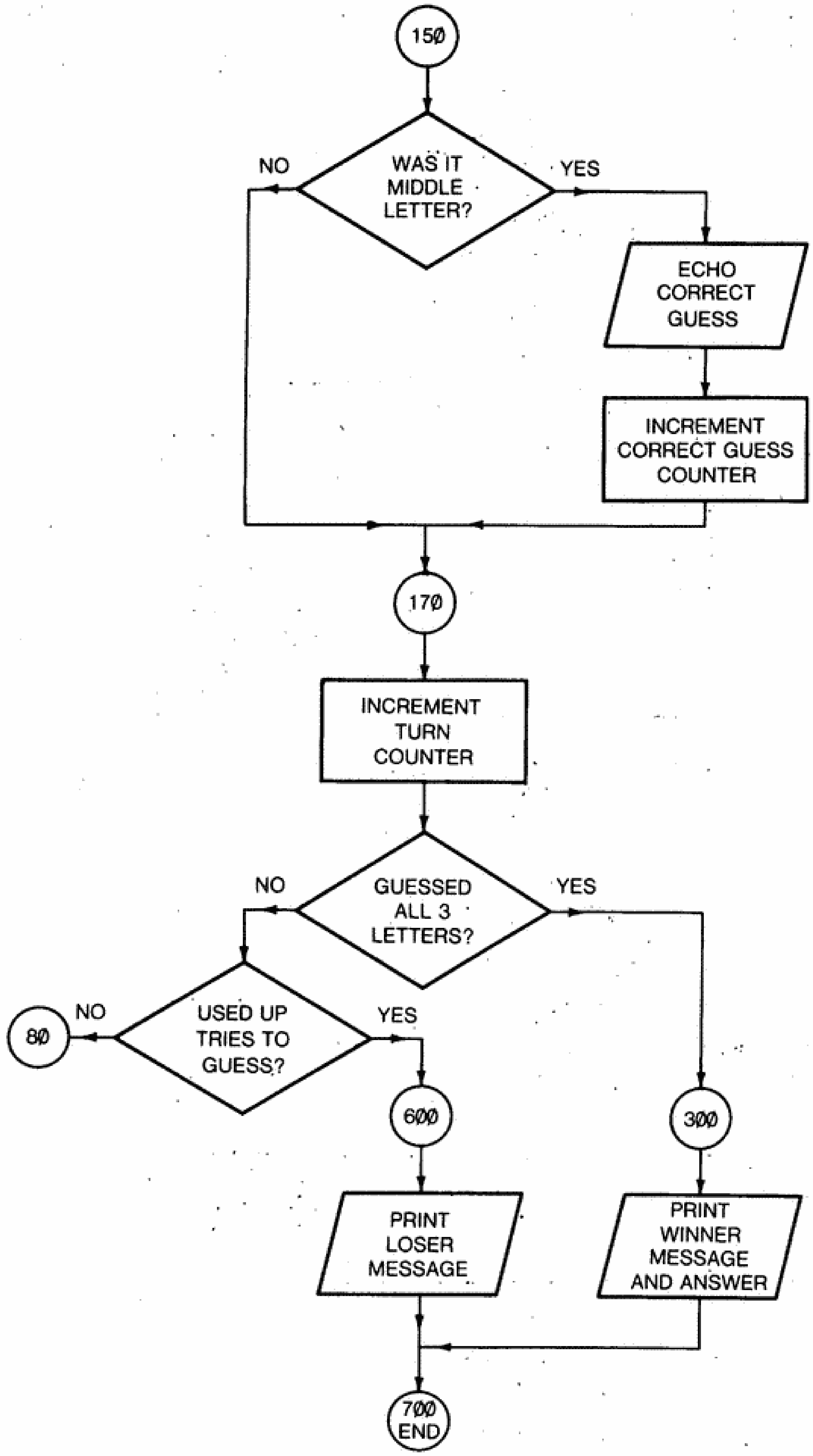
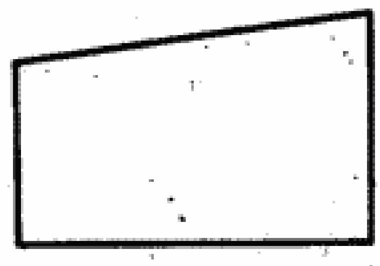
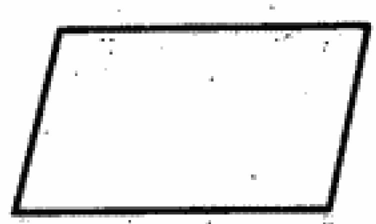


Fig. 5B Flow Chart (15Ø to 7ØØ)

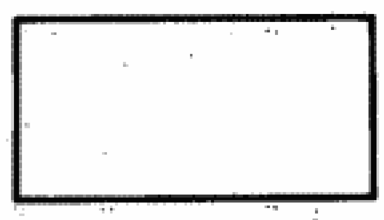
This picture was then directly written as a BASIC program, since the programming decisions had been made.
Statement numbers in circles, known as "connection points" are used to indicate program start, stop, and branching connections. Input operations are represented by a sideview drawing of a key board:



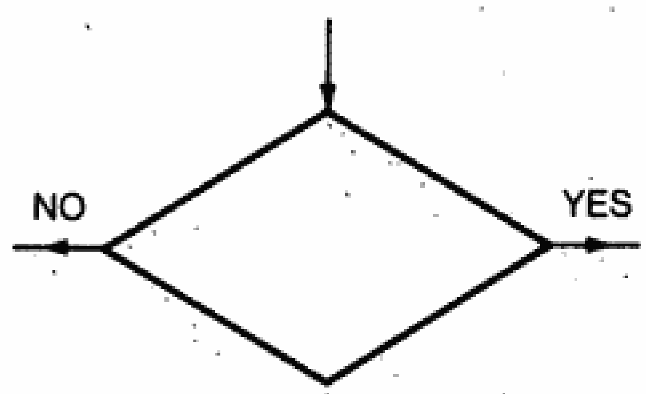
Printing on the video monitor is shown by a:



and calculations are shown by a:



Branching statements are shown by



where the two possible branching choices are indicated. These symbols are standard. However, a distinct set of shapes (from any available template) will encourage the use of flow charts. The path of calculations, from one operation to the next, is shown by arrows.

Simplification of this program is made possible by using the MID$ string operation as

```
 90 FOR CHAR=1 TO 3
100 IF MID$ (A$,CHAR,1)=B$ THEN PRINT B$
110 IF MID$ (A$,CHAR,1)=B$ THEN COUNT=COUNT+1
120 NEXT CHAR
130 REM—THE MID$ OPERATION CAN
140 REM—REPLACE THE LEFT$
150 REM—AND RIGHT$ OPERATIONS
160 REM—WITH RESULTING SIMPLICITY
```

The flow chart drawing for this new program segment (statements 90 to 160) can be shown as a loop in Fig. 6.

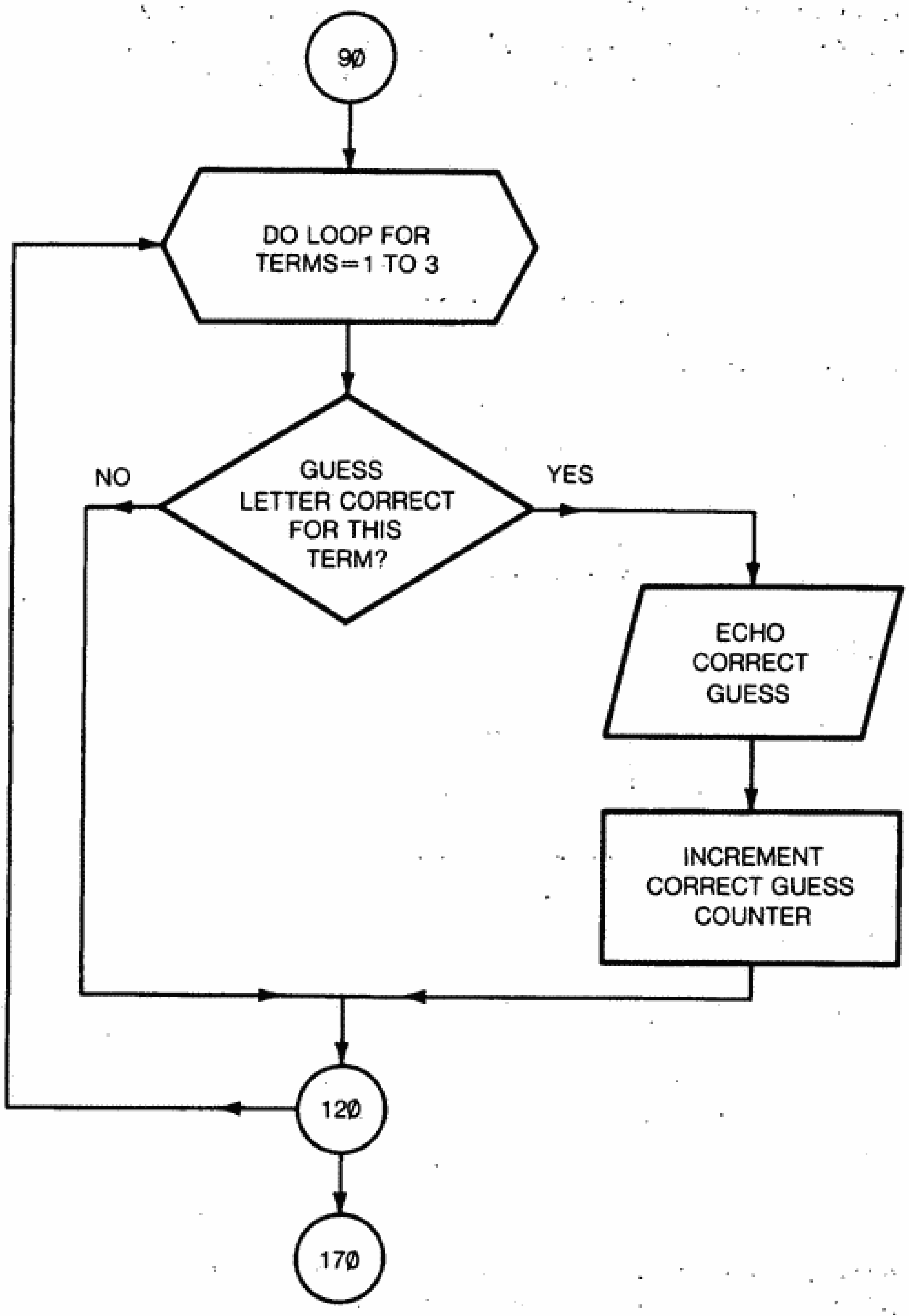


Fig. 6 Flow Chart (9∅ to 17∅)

Each term is considered in the same way, so the loop examines the first, second, and third letters of the answer in order.

If it were desired to rewrite this game program for different-length words, this last form would be easier to follow. In programming, sacrifice anything but clarity.

Now rewrite the program for words up to five letters in length. Output a blank for each letter as a prompt. As the player guesses a correct letter, fill in the blanks and show them (including repeated letters in the word). Most importantly, eliminate the chance to cheat by barring reuse of correctly guessed letters, while allowing the opportunity to repeat incorrectly guessed letters.

The former error was a logic error, discovered by playing (testing?) the game. The program writer could have written the program to generously forgive repeated wrong entries, but this would have made the example longer (and easier for the player)!

The subscripted variables, such as C$(1), C$(2), C$(3), ..., will be used to hold the value of the first, second, third, ... , correctly guessed letter(s). This will permit clearer printed messages to the player. By using the same variable name, each subscripted variable can be used by merely changing the subscript.

With this more complicated program, a flow chart is needed. Start with an overall flow chart (Fig.7), the individual boxes of which get expanded as follows: (Fig.8A, B, C)

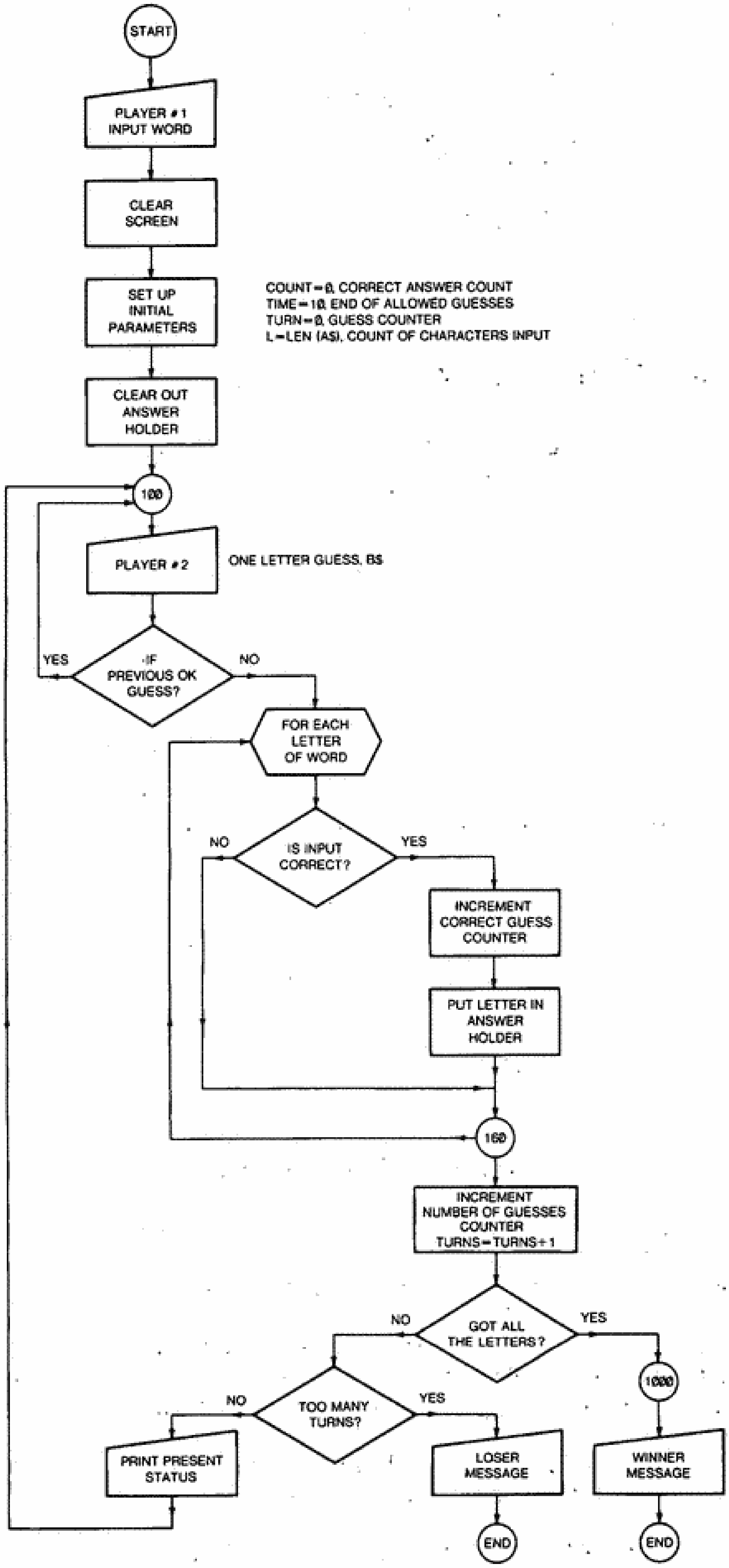
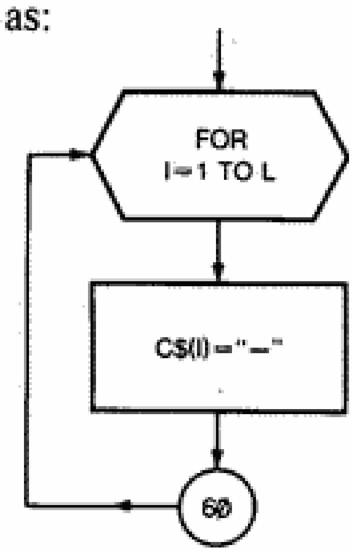


Fig. 7 Flow Chart (Overall)

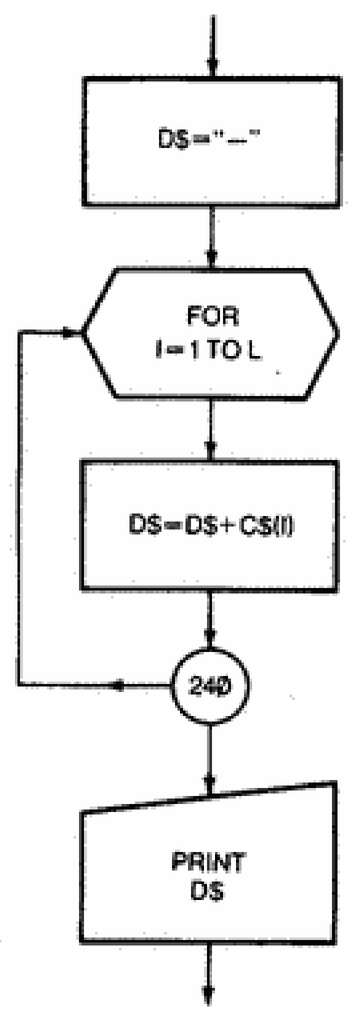
The "clear out answer holder" is expanded as:

Fig. 8A



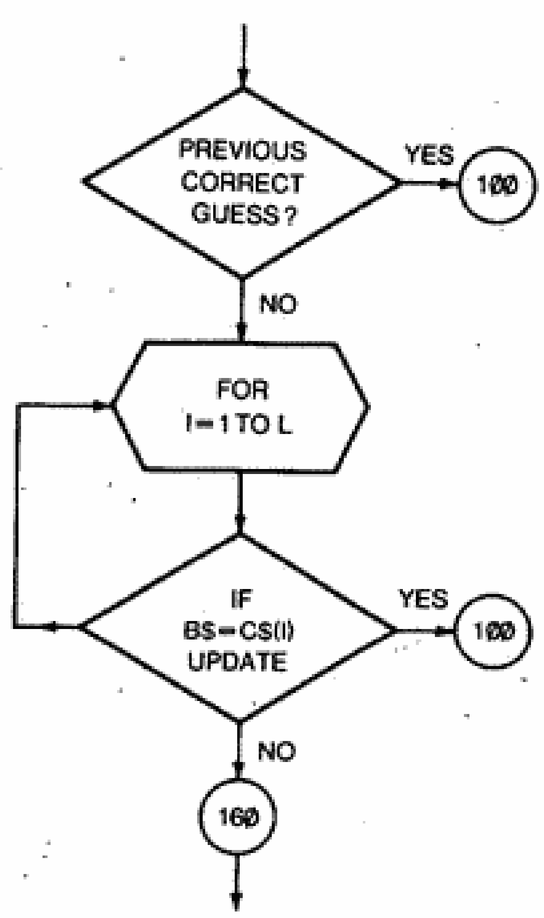
and "print present status" becomes

Fig. 8B



The "previous correct guess" test is:

Fig. 8C



Now convert these flow charts into a program. If a flow chart is well written, the program can be coded as fast as the programmer can type.

```
  10 REM PROGRAM:HANG AUTHOR: L. ROEMER JULY 1979
  20 INPUT "PLAYER #1";A$
  30 COUNT=0:TIMES=10:TURNS=0:L=LEN(A$)
  40 FOR I=1 TO L
  50 C$(I)="-"
  60 NEXT I
  70 FOR I=1 TO 32:PRINT:NEXT I
 100 INPUT "YOUR GUESS";B$
 110 FOR I=1 TO L:IF B$=C$(I)THEN GOTO 100
 120 NEXT I
 130 FOR I=1 TO L
 140 IF MID$(A$,I,1)=B$ THEN COUNT=COUNT+1:C$(I)=B$
 150 NEXT I
 160 TURNS=TURNS+1
 170 IF COUNT=L THEN GOSUB 1000
 180 IF TURNS=TIMES THEN GOSUB 2000
 200 D$=""
 210 FOR I=1 TO L
 220D$=D$+C$(I)
 230 NEXT I:PRINT D$
 240 GOTO 100
1000 PRINT"CHEERS"
1100 END
2000 PRINT"BUMMER"
2100 END
```

Note: In Microsoft BASIC, the conditional statement at 140 also imposed the condition on the statement following the colon ":". The colon serves as a separator between BASIC statements which are written on the same line. An equivalent program segment would have been.

```
140 IF MID$(A$,I,1)=B$ THEN COUNT=COUNT+1
145 IF MID$(A$,I,1)=B$ THEN C$(I)=B$
```

The program still could be improved. For example, the variable C$(I) has been used to store the correct guesses. In order to use more than a ten letter word, additional memory must be reserved for the variable C$(I). This must be done by dimensioning the variable C$(I), for example, for a maximum length of 20 letters in a word as

```
5 DIM C$(20)
```

If a subscripted variable is not dimensioned, BASIC will default to the assumption of 10 subscripts possible. Fortunately, the other variables do not have to be dimensioned, as they are either single characters or, in the case of A$, a single string of characters.

A character string is a set of characters stored under a single variable name.

To play this game, the computer to user dialog would be, typically,

PLAYER #1? GHOST

Then after the screen is cleared,

YOUR GUESS: G

G_ _ _ _

YOUR GUESS? B

G_ _ _ _

This dialog continues until either the winner message of

CHEERS

or losing message of

BUMMER

is printed.

Further improvements in the program could be made by providing a preselected vocabulary or having a stick figure drawn as player errors occur. The program works; the style will be up to the individual.

ASCII CODE

In using string operations, the distinction must be made between a character and its representation inside the computer. For example, to display the number 1, a value of 49 decimal (31 hexadecimal) is sent to the display terminal. This code, called ASCII (*A*merican *S*tandard *C*ode for *I*nformation *I*nterchange), is used for small computer systems. To find the ASCII representation of a character, such as the letter A, use the BASIC command ASC as follows:

```
10 A$="A"
20 X=ASC(A$)
30 REM THE ASCII REPRESENTATION
40 REM OF THE FIRST CHARACTER IN A$
50 PRINT "THE ASCII CODE FOR";A$;"IS";X
60 END
```

This process may be inverted to find whether 65 is really the code for the letter A by using the command CHR$

```
10 X=65
20 A$=CHR$(X)
30 PRINT "65 CONVERTS TO";A$
40 END
```

One application of the ASCII code conversion is in using POKE's. For example, if the command

LIST

is used to clear prior programs from user memory, the letter "L" will be found in location 741 decimal. To examine this, type

PRINT (PEEK(741))

which will return

76

76 is the ASCII code for the letter L (See appendix I for ASCII code list.) Any other symbol in location 741 will disable the command LIST. It would have been easier to have typed

```
PRINT (CHR$(PEEK(741))
```

Conversion to the expected symbol L would have been done directly.

Another example is found when changing the cursor symbol. The cursor symbol is found in Location 9680 decimal. The command

```
POKE 9680,42
```

will make the symbol * into the cursor symbol. However

```
POKE 9680,ASC("*")
```

could have been used to achieve the same result, avoiding looking up the ASCII code. This would be an easier statement to program and a clearer statement to read.

Finally, consider some interesting arithmetic. Since the alphabetic characters are ASCII coded sequentially, from 65 decimal for A to 90 for Z, the statement

```
PRINT(ASC("Z")-ASC("A"))
```

will answer

```
25
```

the difference in code of the 26th and 1st characters of the alphabet. Alphabetical sorting can be readily done using this observation.

For example, read in two letters, arbitrarily placing the first one in string variable FIR$, the second entry in SEC$. Now to test the variables' order of precedence, rearrange the variables into their natural order by the program:

```
10 REM PROGRAM SORT
20 INPUT "FIRST LETTER";FIR$
30 INPUT "SECOND LETTER";SEC$
40 REM EACH LETTER IS INPUT
50 IF FIR$> SEC$ THEN TEMP$=FIR$:FIR$=SEC$:SEC$=TEMP$
60 REM ALL STATEMENTS ON LINE 50 HAVE CONDITION APPLIED
70 REM REVERSE ORDER ONLY IF NEEDED
80 PRINT "LETTERS ARE";FIR$,SEC$
RUN
```

The variables will be rearranged into their normal ordering. A typical dialog is

```
FIRST LETTER? M
SECOND LETTER? C
LETTERS ARE C M
```

This sorting takes advantage of the coding without explicitly using the string commands.

SECTION 7

GRAPHICS

High quality graphics have been provided on the C4P system by dedicating memory to retain the image of the TV screen. The entire screen is normally divided into 64 columns by 32 rows. Other screen arrangements are possible, however. These choices are selected by a BASIC command

```
POKE 56832,N
```

where N is selected as

N	Characters Per Line	Sound On/Off	Color/ Black & White
Ø	32	Off	B & W
1	64	Off	B & W
2	32	On	B & W
3	64	On	B & W
4	32	Off	Color
5	64	Off	Color
6	32	On	Color
7	64	On	Color

To select a B & W screen (64 characters by 32 lines) with the sound off, the command would be

```
POKE 56832,1
```

The same command for color display (64 characters by 32 lines) but keeping the sound off is

```
POKE 56832,5
```

Each character to be displayed is an 8 by 8 array of dots (cell).

There are 256 selectable characters available for use. The 256 characters, selected from a larger possible set, provide versatile graphics without heavy demands for memory. See appendix J for a complete list.

The memory selected for storing the screen image is from 53248 to 55295 decimal. The color selected for each symbol is stored in another set of memory locations from 57344 to 59391. The locations for storing color values are 4Ø96 locations beyond the location for the corresponding symbol. (Since 16 colors are available, only 4 bit (half byte) storage is provided). Memory might be regarded as an image of the screen (See Fig. 9).

A work sheet is provided in the appendix to make an easier task of screen picture layout.

Display of any image is achieved by placing (in BASIC, using the "POKE" command) the character value and its color in the desired locations. For example, the following BASIC program will turn on the color in the 64 character display mode, leave the sound off, clear the screen, fill the color memory with Red using POKE's, place an "X" in a blue square and sit in a delay loop for a few seconds

```
1Ø POKE 56832,5
2Ø FOR I = 1 TO 32 : PRINT : NEXT
3Ø FOR J = 57344 TO 59391 : POKE J,2 : NEXT
4Ø POKE 543Ø2,188 : POKE 58348,8
5Ø FOR TO = 1 TO 5ØØØ : NEXT
```

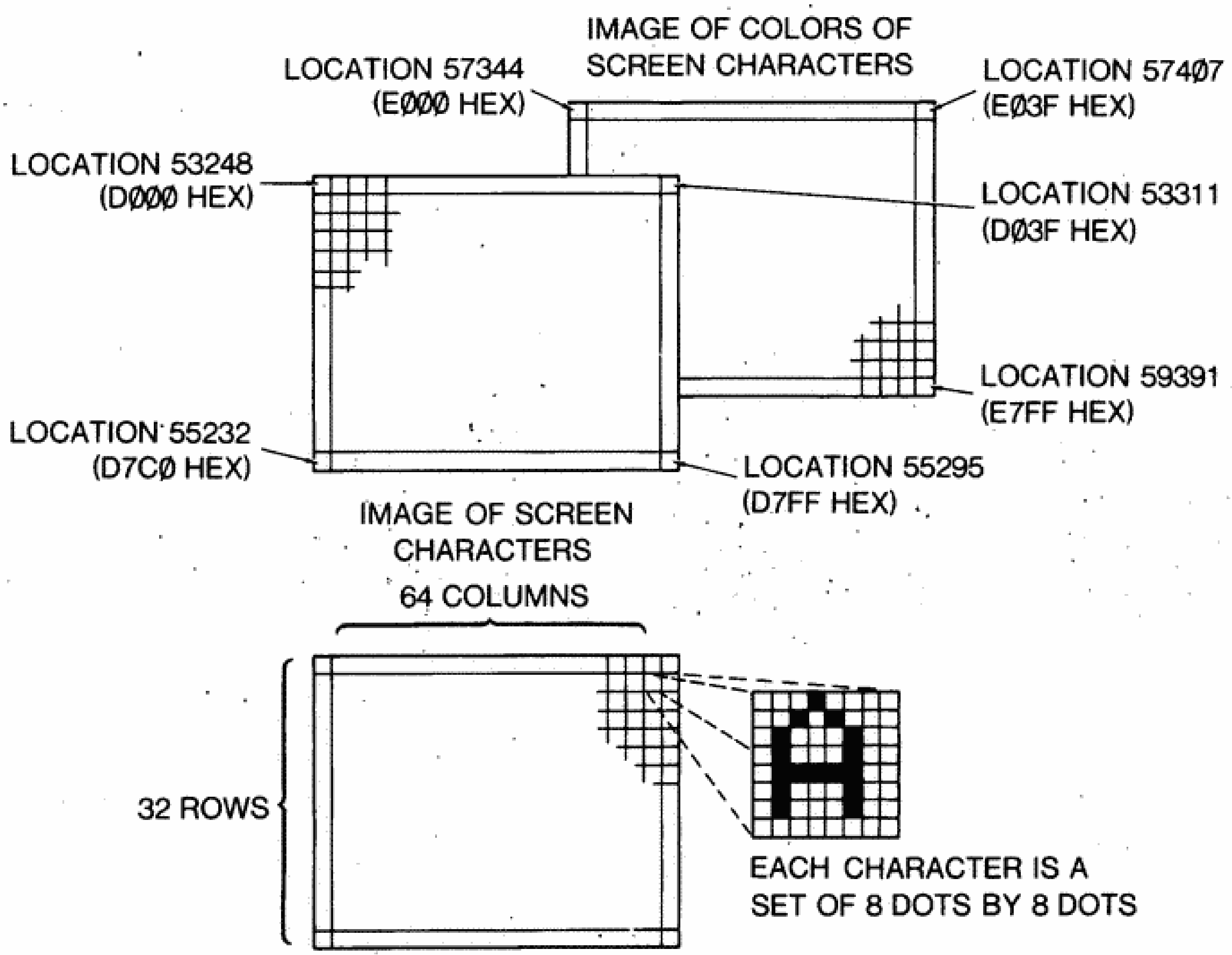


Fig. 9 Make-up of Video Screen

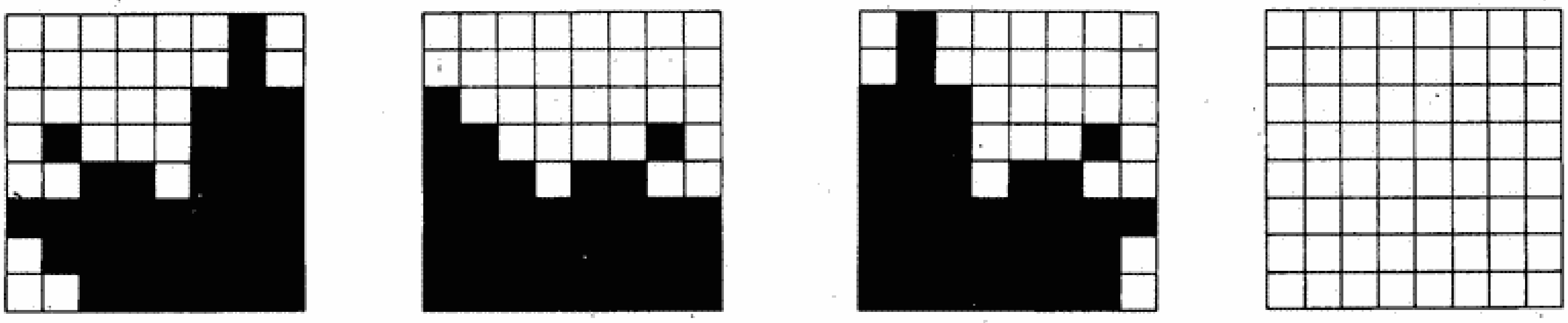
Color selections must be made from this list:

Decimal Value	Color
Ø	Yellow
1	Inverted Yellow
2	Red
3	Inverted Red
4	Green
5	Inverted Green
6	Olive Green
7	Inverted Olive Green
8	Blue
9	Inverted Blue
1Ø	Purple
11	Inverted Purple
12	Sky Blue
13	Inverted Sky Blue

14	Black
15	Inverted Black (no color)

An inverted color is a black background with the symbol in color. Each of the 32 by 64 cells can be colored. To improve viewing, only the center two-thirds of the screen is used for graphics. For any line, the left and right border's color is the same as the last cell on the line (rightmost). The right border wraps its color around to the left border. The cell immediately before the leftmost (addressable) cell has the same color as the leftmost cell.

To illustrate the color choices, the following is a program that places the symbol numbers 181, 182, 18Ø (the shape of a ship in that order) into adjacent locations.



181 182 18Ø 96

This ship will be displayed across four columns for 16 times. Each time the color shall be changed. The program is

```
10 POKE 56832,5  : REM SET UP COLOR ON, SOUND OFF
2Ø ST=53248      : REM START AT UPPER LEFT
3Ø C=ST+4Ø96     : REM COLOR AT 4Ø96 BEYOND SCREEN LOCATION
4Ø FOR RW=Ø
   TO 32         : REM ROW INCREMENT LOOP
5Ø FOR CM=Ø TO 63 STEP 4 : REM COLUMN INCREMENT LOOP
6Ø D=64+CM : REM COMPUTE SCREEN DISPLACEMENT
7Ø POKE ST+D+Ø,181 : REM SHIP USES 4 CELLS
8Ø POKE ST+D+1,182
9Ø POKE ST+D+2,18Ø
1ØØ POKE ST+D+3,96
11Ø FOR I=1 TO 3
12Ø POKE C+D+I,INT(CM/4) : REM SAME COLOR FOR WHOLE SHIP
13Ø NEXT I
14Ø NEXT CM
15Ø NEXT RW
16Ø GOTO 2Ø
```

Since the program is looped on itself, <CONTROL C> must be used to exit.

Examing the possible character fonts in the appendix shows a wide variety of useful images for program sources.

SECTION 8

SOUND

A standard feature of the C-4P system is the ability to generate tones and sound waveforms for music generation or for signaling (e.g. alarms, bells). Two methods are provided for sound generation. The simplest method is the Tone Generator, a device which puts out a continuous stream of square pulses at a programmably selectable frequency. The more versatile method, though more detailed in the requirements in its use, is the companding digital to analog converter. The Companding DAC is capable of generation of arbitrary waveforms, over the common voltage ranges used by audio amplifiers.

Look at the specific characteristics of the two methods.

TONE GENERATOR EFFECTS

For games or test signals, it is often desirable to have a tone generated at a specific frequency. This frequency can be heard when the audio output (See Figure 1) of the C-4P is connected to the audio input jack of the AC-3P video monitor (or other audio amplifier).

This facility is available when the sound is turned on by

```
POKE 56832,7
```

for color and sound or

```
POKE 56832,3
```

for black and white and sound.

The other sound options are listed in the "Video Graphics" section.

The tone generator's frequency is set by

```
Frequency out = 49152/I
```

where I is an integer between 1 and 255. The value of I is stored in 57Ø89 by

```
POKE 57Ø89,I
```

The registers at 56832 and 57Ø89 are write only locations, and cannot be PEEKed.

A familiarization test program which demonstrates the range of tones produced is

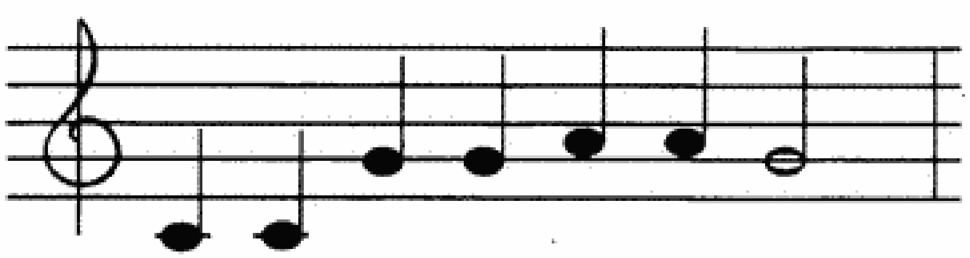
```
1Ø TUNES=57Ø89
2Ø CST=49152        :REM CONSTANT FOR FREQUENCY CALCULATION
3Ø FOR I=1 TO 255
4Ø POKE TUNES,I
5Ø F=INT(CST/I)     :REM F IS FREQUENCY IN HERTZ (CPS)
6Ø PRINT I;F
7Ø NEXT I
8Ø POKE TUNES,I     :REM BE SURE TO TURN TONE OFF!
9Ø END
```

To try this computer feature in a more interesting tune, the first seven notes of "Twinkle, Twinkle Little Star" have been found to be frequencies of 261.6, 261.6, 392.Ø, 392.Ø, 44Ø.Ø, 440.Ø, 392.Ø Hertz (cycles per second). The frequency of the different notes appears in many encyclopedias and handbooks as well as Appendix F. These data form a BASIC program as shown:

```
  5 REM TWINKLE TWINKLE TUNE
 1Ø TUNE=57Ø89
 2Ø FOR T=1 TO 7
 3Ø READ N,BEATS
 4Ø I=INT(49152/N)
 5Ø POKE TUNE,I
 6Ø FOR DELAY=1 TO 5ØØ*BEATS
 7Ø NEXT DELAY
 8Ø FOR D=1 TO 5Ø:POKE TUNE,I:NEXT D
 9Ø NEXT T
1ØØ DATA 261.6,1,261.6,1
11Ø DATA 392.Ø,1,392.Ø,1
12Ø DATA 44Ø.Ø,1,44Ø.Ø,1
13Ø DATA 392.Ø,2
```

The tone generator continues to put out a tone without requiring the computer to do additional calculations. This achieves efficient use of the computer for signaling at audio rates. A keyboard and note guide is provided in the appendix to help write tunes.

Twinkle Tune



DIGITAL TO ANALOG (D/A) CONVERTER

For general applications, the C-4P is equipped with a companding digital to analog converter (DAC). This DAC is coupled to the output through a capacitor. Therefore, only changing voltages can be observed. A constant voltage will be blocked by the capacitor. For example, a positively increasing signal from the DAC will appear at the output as a positive voltage. A decreasing signal from the DAC will appear as a negative voltage. The peak to peak voltage range is about 3 volts. (Brief maximum excursions of up to ±3 volts are possible at start up.)

Since the output of the DAC must change rapidly to pass through the capacitor coupling to the output, the program code which drives the DAC must be in machine code, rather than in BASIC.

A program to drive the DAC can be loaded under the machine monitor at boot up by responding to

H/D/M?

with

M <RETURN>

Press the "period" (" . ") to enter the address mode and type

Ø3ØØ

as an address, then press the "slash" (" / ") to alter the memory locations. Enter the two digit hex code at the adresses indicated

Address		
Ø3ØØ	E8 <RETURN>	Increment X
Ø3ØØ	8E <RETURN>	
Ø3Ø2	Ø1 <RETURN>	Store X at location $DFØ1
Ø3Ø3	DF <RETURN>	
Ø3Ø4	4C <RETURN>	
Ø3Ø5	ØØ <RETURN>	To return to start
Ø3Ø6	Ø3 <RETURN>	

Then type " . " again to return to the address mode. Type

Ø3ØØG

to run the program starting at location 3ØØ hexadecimal.

This program will produce a "saw-tooth" (roughly triangular) waveform at the DAC output. Music generation of pleasing quality, imitative of musical instruments can be played by this device (with additional programming).

Be cautioned that the DAC output should not be tied together with any other output of the computer (such as the tone generator). Further, only one audio output should be used at a time since the register assignment of the audio output devices is the same.

The sound output should be taken from the DAC (See Figure 1) output jack, of course.

SECTION 9

STORING FILES ON CASSETTES OR DISKS

The need to be able to store long programs for rapid reentry is rapidly evident to the new computer user. The chance for typing errors, compounded with the waste of time, encourages one to find an inexpensive medium which maintains program fidelity. Both Cassette and Disk offer such a medium. Cassette provides the most economical medium, since low cost tape recorders suffice for reproducing the signals stored on tape. Low speeds (several minutes for typical programs) and the lack of program selection under computer control are the drawbacks which balance the low cost. The disk provides the speed and computer controlled program selection, but at a somewhat higher cost than a tape recorder.

Following are examinations of the methods of LOADing a program into memory for execution, and the storage of a user written program on the storage medium. First, a look at the cassette as a storage medium.

TO LOAD CASSETTE: PROGRAMS INTO RAM (MEMORY)

Enter BASIC, as shown in the previous section.

1. Place the demonstration cassette in the recorder.
2. Rewind the cassette tape. When the tape stops rewinding return the selection switch(s) to STOP.
3. Type

   <u>NEW</u> <RETURN>

   This will clear memory in preparation for reading the cassette.
4. Type

   <u>LOAD</u>

   but not <RETURN>
5. Start the cassette in the PLAY mode, in order to play back the demonstration programs into the computer memory.
6. As soon as the tape leader has moved past the recorder head (is no longer visible on the wound up reel), press the

   <RETURN>
7. The computer will type

   ?S ⌋ ERROR

   OK

   Which may be ignored. The computer will then list the program being read. The program appears on the terminal screen and is simultaneously stored in memory. If there is a large unused tape region between the tape leader and the program, meaningless characters will be printed. They may be ignored, as they will not affect the program operation.
8. When the program is finished listing, there will be printed

   OK

   ?S ⌋ ERROR

   OK

9. Turn off the cassette recorder, then type

   <SPACE>

   then

   <RETURN>

   The program is now in memory and may be examined by typing

   LIST <RETURN>

10. When finished, store the cassette away from heat or magnets. Do not leave the cassettes on the computer case, as the temperature and proximity to the iron transformers can degrade the programs stored on tape.

SAVING PROGRAMS ON CASSETTE

First clear memory by typing

NEW <RETURN>

The computer responds

OK

Now write a short program

```
1Ø PRINT"NOW IS THE TIME"
2Ø PRINT"FOR ALL GOOD MEN"
3Ø END
```

to be stored on tape.

1. Rewind the tape.
2. Type

   SAVE <RETURN>

   The computer responds

   OK

3. Now type

   LIST

   but not <RETURN>!

4. Start the recorder in the record mode. This operation is obtained by pressing the RECORD and PLAY switches, simultaneously. (This two switch operation is meant to reduce inadvertent writing over programs not meant to be destroyed).
5. As soon as the leader passes the recording heads (disappears from sight on the windup reel), type

   <RETURN>

6. When the listing is complete, turn off the tape recorder and type

   LOAD <RETURN>

   <SPACE>

   <RETURN>

7. Now rewind the tape and check that the recording is satisfactory by following the instructions to **LOAD** the cassette.

USE OF CASSETTES AS A DATA STORAGE MEDIUM

Intermediate data within programs can be stored on cassette. This provides easy retrieval of data and intermediate calculations for future use.

As an example, this is how to print the numbers 1 to 15 on the cassette. After rewinding the tape, the sequence of operations would be

1. Write the program to create the desired data, such as

   ```
   10 FOR I=1 TO 15
   20 PRINT I
   30 NEXT I
   40 END
   ```

2. Type

   SAVE <RETURN>

3. Type

   NULL 8 <RETURN>

   This step, although optional, is recommended.

4. Type

   RUN

   but not <RETURN>

5. Start the recorder in the record mode (PLAY and RECORD switches depressed). As soon as the tape leader has passed the recording head, press

   <RETURN>

6. The data will be recorded on tape and listed on the terminal screen.
7. When the listing of data is complete, turn off the tape recorder and type

   LOAD <RETURN>

   <SPACE>

   <RETURN>

   to return to normal operation.

Note that this set of procedure steps was almost the same set used to SAVE a program.
This data can be input for another program in a similar manner.

READING DATA FROM CASSETTE TAPE

In a manner similar to LOADing programs from cassette, data can be read from cassette. The steps are

1. Rewind the cassette tape.
2. Type

   NEW <RETURN>

3. Enter the program which will use the data on tape. A typical program might be

```
10 INPUT A
20 PRINT "DATA IS=";A
30 IF A <15 THEN GOTO 10
40 END
```

   Now type

   RUN

   but not <RETURN>

4. Start the tape in the PLAY mode to play back the data. When the tape leader is beyond the recorder's head, then press

   <RETURN>

5. The requests for data will be shown on the terminal screen as typically

```
?1
DATA IS=1
?2
DATA IS=2
```

   etc.

6. Upon completion of the program (or the tape's being wound up on the reel), turn off the tape recorder. Then type

   <SPACE>

   and

   <RETURN>

   The computer will now be in the BASIC program.

These techniques should permit a flexible use of the cassette, both as a program and data storage medium. For extensive data handling, however, the drive control of a disk will give enhanced speed and control. Therefore, its use is encouraged.

The alternative to Cassette storage is use of disk. For comparison, examine the equivalent operations on a disk. Even cassette users should examine the power of a disk operating system. The large convenience of such a system may justify the modest additional cost.

TO WRITE TO DISK

The operating system (OS-65D V3.N) contains simple and powerful routines to handle disk input and output. These routines permit using low cost disk storage rather than using the more expensive random access memory (RAM).

A simple connection for storing BASIC programs is available.
First, create a file, say "SCRTCH" (see Appendix G, Section E); then a simple program such as

```
10 PRINT "NEW TEST"
20 END
```

can be stored on the file "SCRTCH" by typing

DISK!"PUT SCRTCH" <RETURN>

Now type

NEW <RETURN>

LIST <RETURN>

and see that nothing is printed, since the work space was cleaned by the NEW command.
If LIST yields a Syntax error, type POKE 741,76 <RETURN> to enable LIST.

TO READ FROM DISK

To load the program from disk into the BASIC work space, type

DISK!"LOAD SCRTCH" <RETURN>

Then the LIST command

LIST <RETURN>

will result in the listing of the previously stored program.
Another method to store and retrieve the program on SCRTCH is available. BASIC can be exited by typing

EXIT <RETURN>

Then respond to the DOS prompt:

A*

by typing

PUT SCRTCH <RETURN>

to store the program directly under control of DOS.
The copying of file "SCRTCH" into the work space is accomplished by typing

LOAD SCRTCH <RETURN>

To be able to specify the disk locations and memory locations, a more detailed set of commands are CALL and SAVE.
These commands are used after the operating system prompt (and generally apply only to machine code programs)

A*

as

CALL address = track, sector <RETURN>

and

SAVE track, sector = address/page <RETURN>

These commands transfer a specified track (1 to 39), and sector (1 to the maximum used on that track). A page is 256 bytes. Each sector is an integer multiple of pages, i.e., 1, 2, 3 pages of 256 bytes each. The address must always be a four digit hexadecimal value, track must be two decimal digits (so track 2 is written Ø2), and sector is one decimal digit. Pages must be one hexadecimal digit within the range 1 to 8. A given sector can be referenced only if all lower numbered sectors exist on the specified track.

The CALL and SAVE commands are particularly suited to storing and retrieving machine code programs. An example of this is shown in the use of disk copy routines given in the appendix. The CALL and SAVE also permit storing data on a track without the requirement of creating a named file.

Since all these routines can be invoked within a BASIC program, the ability is provided to run complete BASIC programs which use other BASIC and machine code programs, brought in as needed from disk. This allows the use of large programs, small parts of which are brought into memory as needed.

However, the frequent use of the routines, CALL and SAVE, under BASIC, is probable. The DISK! command can be used to gain access to the operating system commands while remaining in the BASIC program. For example, to SAVE a program on track 39 for 1 sector, where the program is resident at memory location 3279 hexadecimal, and it is less than one page (256 characters) long, the command is

DISK!"SAVE 39,1 = 3279/1" <RETURN>

Likewise, to recall this same program back into these same memory locations, write

DISK!"CALL 327 = 39,1" <RETURN>

Caution is urged, as it is possible to bring the disk program on top of a program in use. This will destroy the program which is overlaid. Each command that gives additional power or discretion carries the need for additional caution.

OPERATING SYSTEM ORGANIZATION

An operating system is a program, or set of programs, which supervises the running of individual programs. That's not a purist definition, but it will do.

The central part of the OSI disk operating system (DOS on Figure 1Ø) supervises the running of all programs. It can call for three subsidiary (or utility) programs: BASIC, ASSEMBLER language (ASM), and the EXTENDED MONITOR (EM).

BASIC is the program commonly in use. It is almost conversational in form. Since it is a high level language, it is very powerful and rapid for program writing.

ASSEMBLER is a shorthand way to write machine language programs. The details are covered in the *Ohio Scientific 6500 Assembler/Editor User's Manual* and MOS Technology's *Microcomputers.*

EXTENDED MONITOR provides the ability to inspect, alter, or fill memory locations. It can also move blocks of program from one memory region to another. Details are discussed in the *Ohio Scientific Extended Machine Language Monitor User's Manual.*

The inter-relation of these programs is shown in Figure 1Ø. The recommended way to go from one program to another is shown beside the direction arrows. These are the commands to be typed.

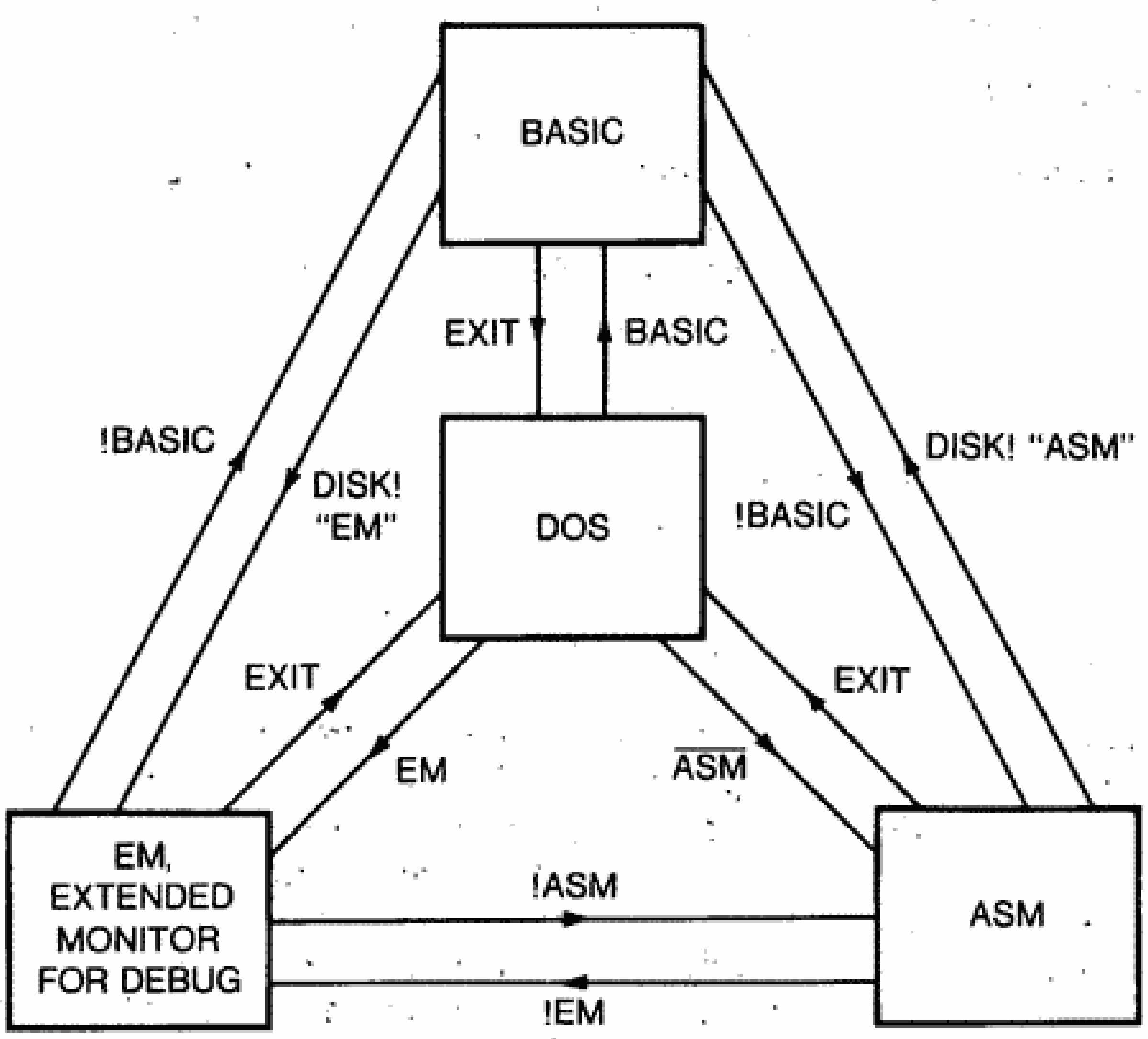


Fig. 1Ø Operating System Flow Chart

At boot up time, the operating system will deliver the BASIC program as a default. To illustrate, when in BASIC, as shown by the prompt

OK

type

DISK!"EM"

and see the EXTENDED MONITOR prompt

:

Upon typing

EXIT

EM will be left and the computer will be back in DOS as indicated by the * prompter. Return to BASIC can be effected by typing

BA

(Note: valid only if BASIC is still in memory) which is a return to the starting point.

Since different services are provided by BASIC, EM, and ASM, it is nice to be able to use these programs interchangeably.

SECTION 10

ADVANCED FEATURES

KEYBOARD

The keyboard provides a useful input device for games and home control. The easiest way to use the keyboard is to use the BASIC command INPUT, as

INPUT A

However, the INPUT command causes a "?" prompt to be printed. Also, scrolling (movement) of the video screen display occurs. These effects could detract from game and display use. A method to avoid these problems is available.

The keyboard consists of rows and columns of conductors. When a key is depressed, contact between the row conductor and the column conductor is made. To determine whether or not a key is depressed, certain values can be entered into the keyboard address by a POKE command and the results observed by a PEEK command. The values which need to be POKEd and PEEKed are shown in Fig. 11:

VALUES TO POKE (DECIMAL VALUE)	VALUES FOUND WHEN PEEKED (DECIMAL VALUE): 128 C7	64 C6	32 C5	16 C4	8 C3	4 C2	2 C1	1 CØ
128 R7	1	2	3	4	5	6	7	
64 R6	8	9	Ø	:	-	RUB OUT		
32 R5	.	L	O	LF	CR			
16 R4	W	E	R	T	Y	U	I	
8 R3	S	D	F	G	H	J	K	
4 R2	X	C	V	B	N	M	,	
2 R1	Q	A	Z	SPACE	/	;	P	
1 RØ	REPT	CTRL	ESC			L SHIFT	R SHIFT	SHIFT LOCK

Fig. 11 Keyboard (Values to be POKEd and PEEKed)

The keyboard appears at address 57Ø88. To test for depression of the key "V", for example, the sequence would be

```
1Ø R2=4:C5=32
2Ø POKE 57Ø88,R2
3Ø VT=PEEK(57Ø88) :REM VT IS TEST FOR V'S COLUMN
4Ø IF VT=C5 THEN A$="V"
```

to set A$ equal to the "V" key, if it were depressed.

The possibility of depressing several keys simultaneously requires the disabling of the <CONTROL C> feature to avoid problems in identifying which keys are used. This is done by a POKE 2Ø73,96 to disable the <CONTROL C> feature, prior to polling (examining) the keyboard.

The polled keyboard achieves its economy by reading the switch closures within a program. The keyboard appears to be memory, located at 57Ø88, as seen by a program.

The keyboard is a standard 53-key layout, with a few minor exceptions. These exceptions are:

1. The "here is" key on standard layouts is deleted. It has been replaced by a "rub out" key in this position.
2. The standard "rub out" key position is filled by a "shift lock" key. This key is locked in the depressed position in normal use.
3. The "left shift" and "right shift" keys are separately decoded to permit greater versatility.
4. The "break" key is brought directly to the computer reset circuits. Use of this key restarts the system operation.

Lower case letters and fonts can be obtained when the SHIFT key is unlocked (not depressed). Normally, in BASIC, the SHIFT LOCK is locked. However, text editing and letter writing will require access to these features.

The foregoing has been a demonstration of a simple method to read the key closures without disturbing the video display. This method can be extended to the keypad and joystick accessories, which are merely extensions of the keyboard.

By using similar programs, interactive games and their displays are easily controlled. The complexity of the most involved game does not require any more than the example just examined.

Some special purpose keys should be mentioned.

1\. SL— the SHIFT LOCK key forces upper case letters to be printed on the CRT. It should be depressed prior to bringing up the system or running BASIC. Unlike a typewriter, however, the numbers will be printed normally. To type the symbols above the numbers, press the <SHIFT> key simultaneously with the desired character. The SHIFT LOCK key is used for normal entry. It should be released only for use of lower case letters, and then reset.

2\. BREAK— resets the computer any time after the system is powered up.

3\. SPACE BAR— provides a space when pressed..

3\. RETURN— must be pressed after a line is typed. The previously typed line is then entered into computer memory.

5\. CONTROL C— press <CONTROL> while simultanously pressing C. Program Listing or executing is interrupted, and the message.

BREAK IN LINE XXX

is printed and XXX=a line number in the program.

6\. SHIFT O— press <SHIFT> first while simultaneously pressing O. The last character typed is erased. By the way, O is the letter "oh"; Ø will represent the number "zero." Do not type the slash. It is just to make reading easier.

7\. SHIFT P— press <SHIFT> first while simultaneously pressing P. The current line being typed will be erased. The symbol '@" will be displayed. The effect will be to erase the line typed and enter a <RETURN> and <LINE FEED>.

8\. D— When pressed after <BREAK>, causes initialization of the computer and boots the operating system from disk.

9\. M— When pressed after <BREAK>, causes initialization of the computer. The computer is then in its machine language monitor.

SECTION 11

JOYSTICKS AND KEYPADS

JOYSTICKS

The joysticks provide realistic and convenient input devices for games and control. They are connected to the system as shown in Figure 12. The joysticks provide a digital signal when they are connected and enabled.

Prior to using the joysticks (or keypads) the <CONTROL C> command must be disabled by

POKE 2Ø73,96

The enabling of joystick A is done by

POKE 57Ø88,128 : REM — ENABLE JOYSTICK A

and joystick B is enabled by

POKE 57Ø88,16 : REM — ENABLE JOYSTICK B

Only one joystick can be enabled at a time.

The joystick position can be read using the PEEK command. The value found using the PEEK command must be ANDed with a constant, depending on which joystick is used, to obtain a value for the specific joystick position. The constants used are 31 for joystick A and 248 joystick B. For example

APOSIT=PEEK(57Ø88) AND 31

will return a value for APOSIT (A's position) which indicates the joystick position. If the "ACTION" KEY is not depressed, the value returned for joystick A will be as indicated in Fig. 13:

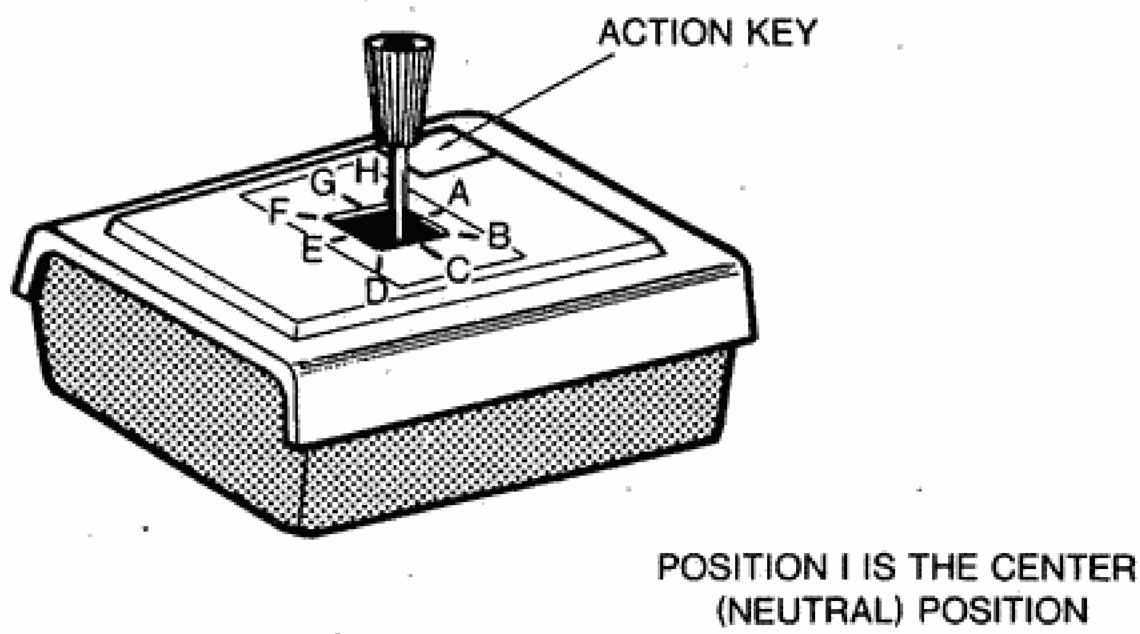


Fig. 12 Joystick

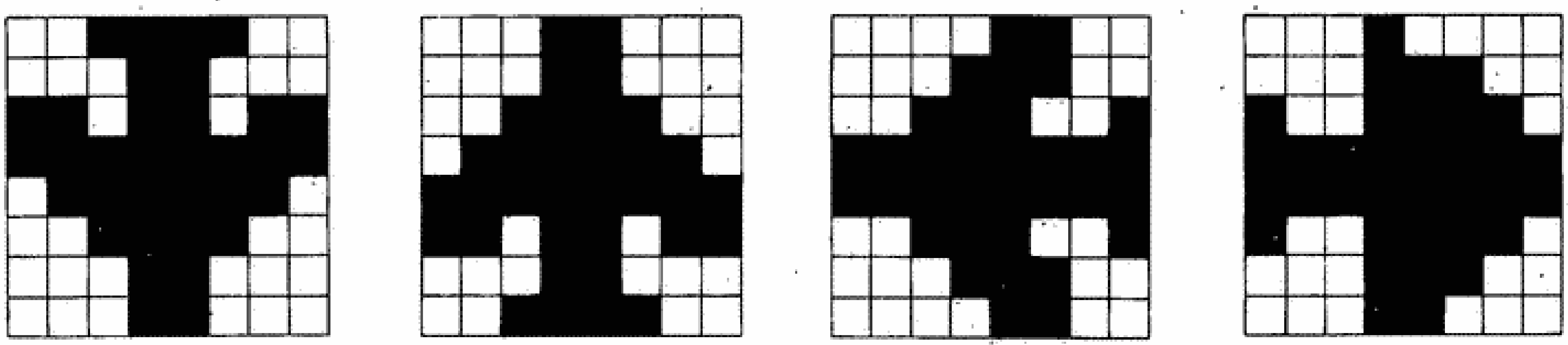
	Joystick A		Joystick B	
Joystick Position	Action Key Not Depressed Decimal Value Returned	Action Key Depressed Decimal Value Returned	Action Key Not Depressed Decimal Value Returned	Action Key Depressed Decimal Value Returned
A	16 4	17 20	32	16Ø
B	2Ø 12	21 28	48	176
C	4 8	5 24	16	144
D	12 9	13 25	8Ø	2Ø8
E	8 1	9 17	64	192
F	1Ø 3	11 19	72	2ØØ
G	2 2	3 18	8	136
H	18 6	19 22	4Ø	168
I	Ø	1 16	Ø	128

Fig. 13 Joystick Values

With the action key depressed, 1 has been added to the "action key depressed" value for joystick A. When joystick B is enabled, the corresponding values are returned to

BPOSIT=PEEK(57Ø88) AND 248

The "action key depressed" causes 128 to be added to the "action key depressed" value for joystick B.



238 236 239 237

Fig. 14 Airplane Display

To try a sample program, cause the airplane figures in Fig. 14 to move about the screen. Place the plane in the screen center to start at location 534Ø4 (D42Ø hexadecimal). Ignore clearing the screen, simply leaving it in B & W with 64 characters per line and the sound off, by typing

1Ø POKE 56832,1

Put the original plane on the mid-screen by

2Ø POKE 543Ø4,236

Since B & W is being used, no color is given. Use the "ACTION" button to quit (exit) the program. Use the logic shown in Fig. 15.

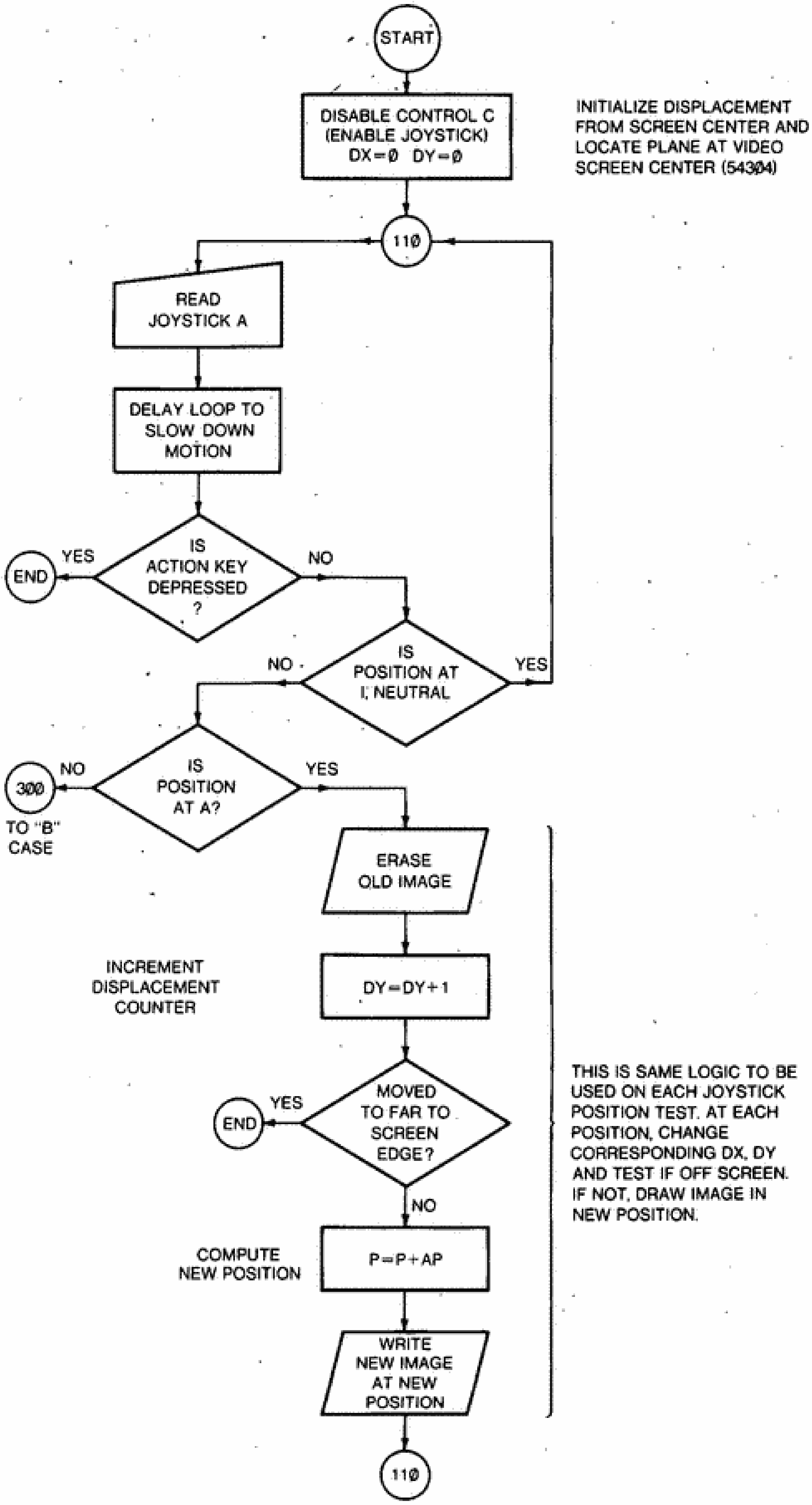


Fig. 15 Flow Chart for Airplane and Joystick.

The program to implement this flowgraph is

```
 1Ø POKE 2Ø73,96                          :REM DISABLE <CONTROL C>
 2Ø AP=-64:BP=-62:CP=+1:DP=66             :REM SCREEN POSITION DISPLACEMENTS
 3Ø EP=64:FP=62:GP=-1:HP=-66:IP=Ø         :REM RESULTING FROM JOYSTICK POSITION
 35 REM
 4Ø A=16:B=2Ø:C=4:D=12                    :REM CODE VALUES FOR
 5Ø E=8:F=1Ø:G=2:H=18:I=Ø                 :REM JOYSTICK POSITION
 55 REM
 6Ø POKE 57Ø88,128                        :REM ENABLE JOYSTICK A
 7Ø BLANK=96                              :REM SCREEN SYMBOL FOR BLANK
 8Ø DX=Ø:DY=Ø
 9Ø P=543Ø4                               :REM MIDSCREEN START
1ØØ POKE P,236
11Ø R=PEEK(57Ø88) AND 31
12Ø FOR K=1 TO 2ØØ:NEXT K                 :REM DELAY LOOP
13Ø IF(R/2-INT(R/2)) > Ø THEN GOTO 9ØØØ   :REM QUIT IF ACTION KEY
135 REM                                   :REM DEPRESSED (ODD VALUE R)
14Ø IF R=IP THEN GOTO 11Ø
15Ø IF R=A THEN GOTO 17Ø
16Ø GOTO 3ØØ
17Ø POKE P, BLANK                         :REM — ERASE OLD IMAGE
18Ø DY=DY+1
19Ø IF ABS(DY) > 16 THEN GOTO 9ØØØ        :REM IF OFF SCREEN, QUIT
2ØØ P=P+AP
21Ø POKE P,236                            :REM "A" POSITION IS UPWARD PLANE
22Ø GOTO 11Ø
3ØØ IF R=B THEN GOTO 32Ø                  :REM "B" CASE
31Ø GOTO 4ØØ
32Ø POKE P,BLANK
33Ø DY=DY+1:DX=DX+1
34Ø IF ABS(DX) > 3Ø OR ABS(DY) > 16 THEN GOTO 9ØØØ
35Ø P=P+BP
36Ø POKE P,237
37Ø GOTO 11Ø
4ØØ IF R=C THEN GOTO 42Ø                  :REM "C" CASE
41Ø GOTO 5ØØ
42Ø POKE P,BLANK
43Ø DX=DX+1
```

```
440 IF ABS(DX) > 30THEN GOTO 9000
450 P=P+CP
460 POKE P,237
470 GOTO 110
500 IF R=D THEN GOTO 520                    :REM "D" CASE
510 GOTO 600
520 POKE P,BLANK
530 DX=DX+1:DY=DY-1
540 IF ABS(DX) > 30 OR ABS(DY) > 16 THEN GOTO 9000
550 P=P+DP
560 POKE P,238
570 GOTO 110
600 IF R=E THEN GOTO 620                    :REM "E" CASE
610 GOTO 700
620 POKE P,BLANK
630 DY=DY-1
640 IF ABS(DY) > 16 THEN GOTO 9000
650 P=P+EP
660 POKE P,238
670 GOTO 110
700 IF R=F THEN GOTO 720                    :REM "F" CASE
710 GOTO 800
720 POKE P,BLANK
730 DX=DX-1:DY=DY-1
740 IF ABS(DX) > 30 OR ABS(DY) > 16 THEN GOTO 9000
750 P=P+FP
760 POKE P,239
770 GOTO 110
800 IF R=G THEN GOTO 820                    :REM "G" CASE
810 GOTO 900
820 POKE P,BLANK
830 DX=DX-1
840 IF ABS(DX) > 30 THEN GOTO 9000
850 P=P+GP
860 POKE P,239
870 GOTO 110
900 IF R=H THEN GOTO 920                    :REM "H" CASE
910 GOTO 110
```

```
920 POKE P,BLANK
930 DX=DX-1: DY=DY+1
940 IF ABS(DX) > 30 OR ABS(DY) > 16 THEN GOTO 9000
950 P=P+HP
960 POKE P,239
970 GOTO 110
9000 END
```

Though the example appears to be long, it is the repeated use of the same tests and operations, in blocks of less than 10 instructions. A nucleus of programs has thus been created with which to implement other games!

KEYPADS

The keypads are merely extensions of the keyboard as are the joysticks. They can be read in the same manner as the keyboard is read by the computer.

Prior to reading the keypad, disable <CONTROL C>, with a POKE 2073,96.

Referring to Fig. 16, examine how keypad A is connected. Keypad A consists of a set of wires which correspond to keyboard rows shown labeled as R1 to R4. These are shown superimposed on the keyboard rows R0 to R7. In the same manner, the keypad A contains wires corresponding to keyboard columns C5 to C7 out of the total keyboard set of columns C0 to C7. When a key is pressed, a connection is made between the row and column where the switch is located.

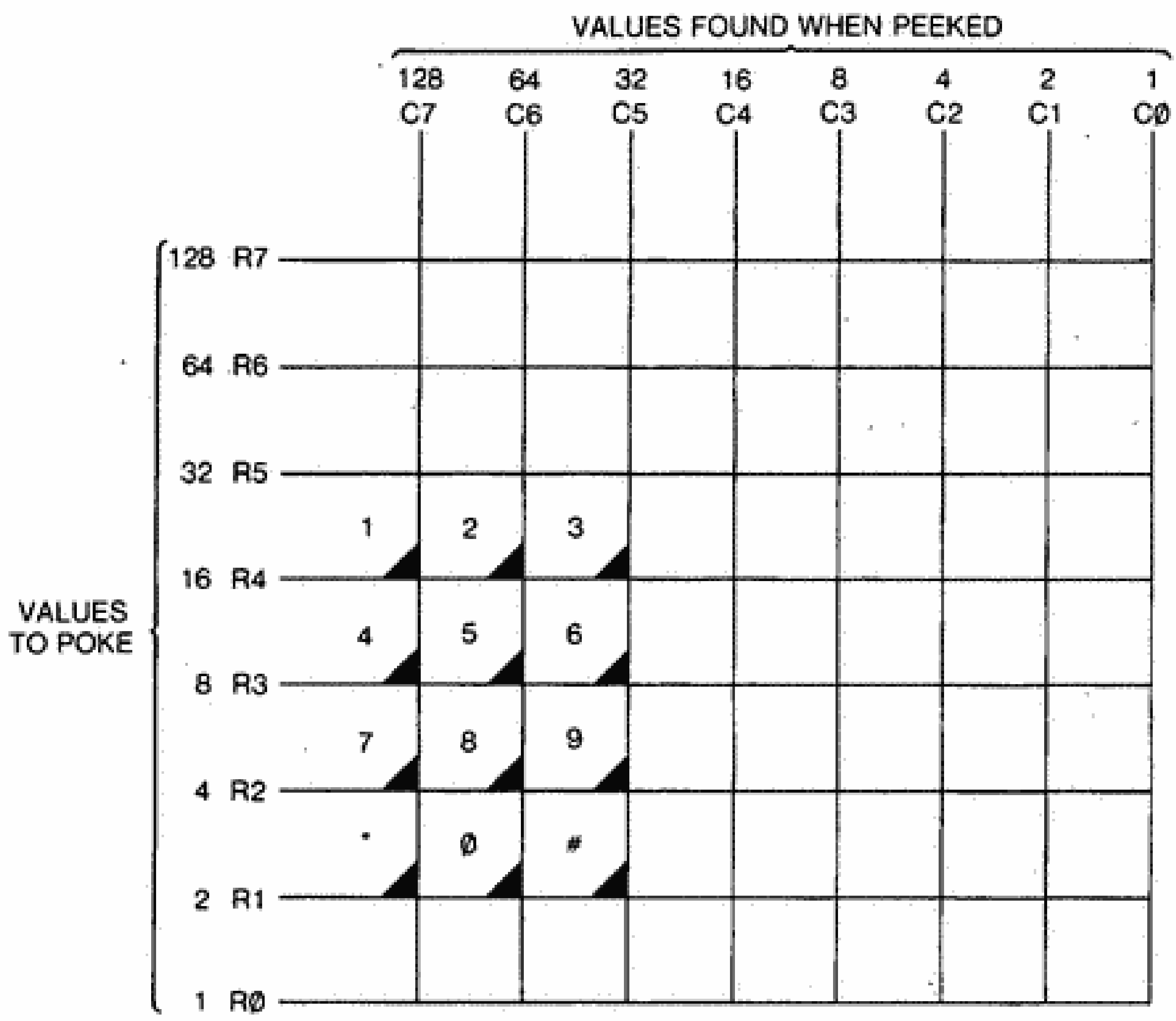
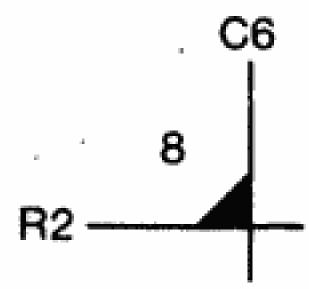


Fig. 16 Keypad A

A cross-over point for keypad A will be drawn as indicated (Row 2 and Column 6 joined when the key for symbol "8" is pressed),



with the key symbol next to the shaded region.
Likewise, keypad B is connected as shown in Fig. 17.

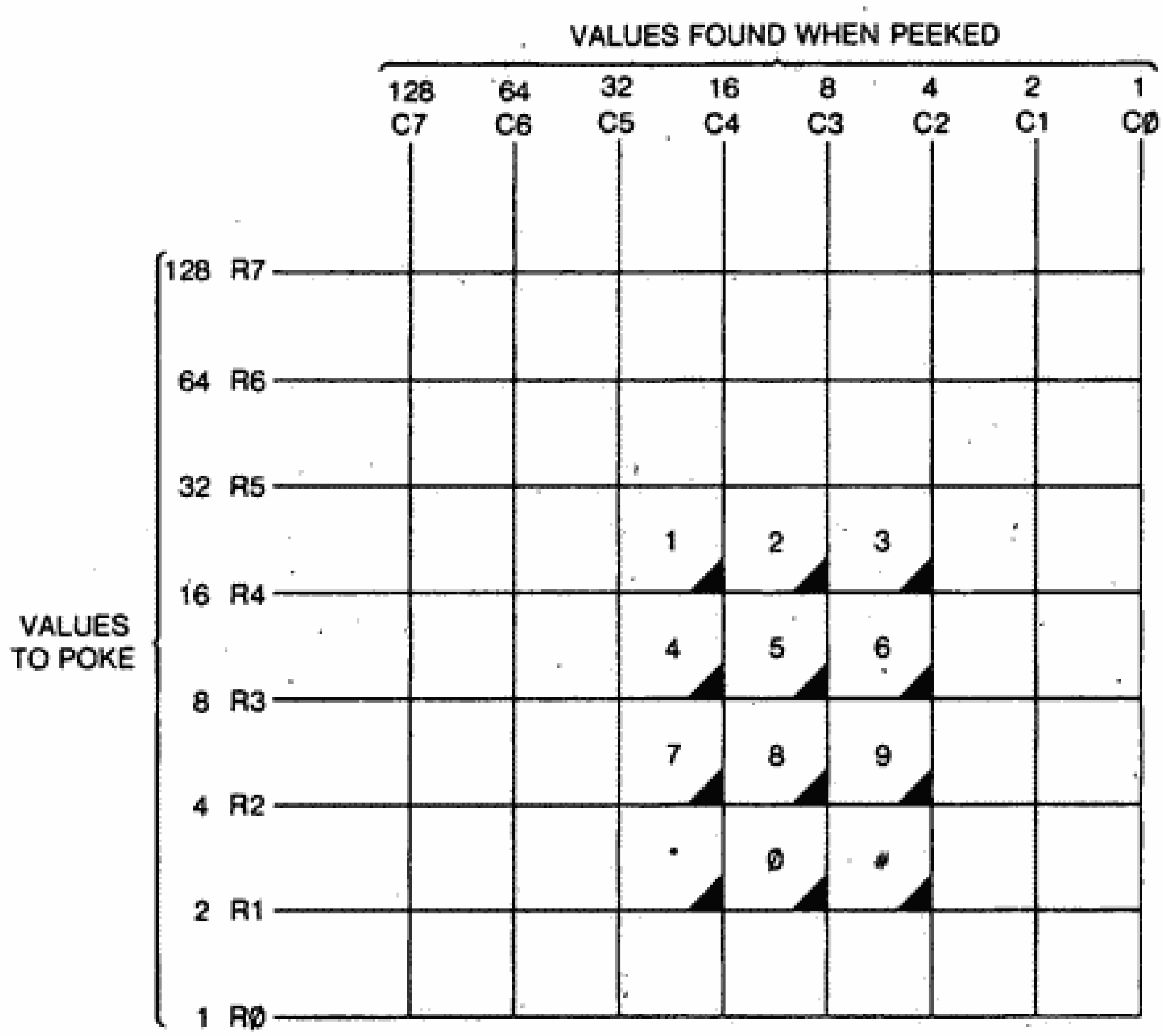


Fig. 17 Keypad B

Since keypad A is connected across R4, R3, R2 and R1, ignore the other rows by examining these lines only. The values of R4, R3, R2, and R1 are 16, 8, 4, and 2, respectively.

It is possible to detect the symbol 8 (located at the intersection of Row 2 and Column 6 on keypad A) by setting Row 2 via

```
1Ø POKE 57Ø88,4
```

where 4 is the value POKEd to activate Row 2. It is then possible to sense Column 6 (value associated with column 6 is 64 by

```
2Ø TEST = PEEK(57Ø88)
3Ø IF TEST = 64 THEN GOTO 1ØØØ
```

where statement 1ØØØ takes care of the case when the 8 value is found.

A short program to read the key "8" or the key "#" and print the respective key is:

```
1Ø REM KEYPAD TEST
2Ø REM DISABLE <CONTROL C>
3Ø CTRLC=2Ø73: DISABLE=96: POKE CTRLC,DISABL
4Ø REM NOW SET POINTER TO KEYPAD LOCATION
5Ø P=57Ø88: R2=4: C6=64: R1=2: C5=32
```

```
100 A$=" "
110 POKE P,R2 : REM TEST FOR 8
120 IF PEEK (P)=C6 THEN A$="8" : REM ON R2,C6
130 POKE P,R1 : REM TEST FOR "#"
140 IF PEEK (P)=CS THEN A$="#" : REM ON R1,C5
```

SECTION 12

AC REMOTE CONTROL, SECURITY

A computer's value over a calculator depends on its ability to change its sequence of computation based on the results already computed. This is particularly important when the values used in computation (decision) are data from or to external devices. External devices use the data, binary 1's and Ø's, sent over lines from the computer as output. The 1's and Ø's are represented by nominal 5 volt and Ø volt levels (TTL logic levels), respectively. Likewise, external devices can send data as input to the computer. Again, standard TTL logic levels are used.

Control, in the C4P, includes being able to turn on/off (and set the level of) AC controlled devices, such as lamps, motors, and appliances. Control also includes being able to supervise security alarms, as well as numerous status switches. All of these capabilities allow device operation while the computer is doing tasks of a more immediate priority.

In the next two sections, the most popular applications are considered. Then, in greater detail, the many possibilities of additional options and capabilities are considered. By combining the capabilities of several features in one program, great flexibility and power can be obtained. All of this is controlled by a readily written BASIC program, based on the examples that follow.

APPLIANCE CONTROL

Without running any wires the C4P can operate lamps and small appliances when equipped with the AC-12 options! This is accomplished by using the BSR X-1Ø ©, a remote AC signaling system. The computer activates the BSR command console which, in turn, sends a signal over the existing home wiring. This signal is sensed at the appropriate device by a small switch module plugged into the AC outlet. The switches are modules which plug into the wall sockets (11Ø volt AC power lines). The appliances are plugged into these modules.

Two types of switches are available, a lamp switch and an appliance switch. A continuously dimmable lamp switch provides adjustable incandescent lighting levels (up to 3ØØ watts per lamp) throughout a building. A relay actuated (on-off) appliance switch provides control of larger devices such as lamps (up to 5ØØ watts), motors (up to 1/3 HP), or current loads of up to 15 amperes.

Each remote switch module has two dials. One selects "house code." There are sixteen choices indicated by the red letters A through P. The "house code" on the remote module must match the "house code" on the control console. The various "house codes" prevent signals from other computers from actuating "non-mated" remote switch modules. Each switch module also has a "unit code" dial (up to 16 units can be addressed), which permits great flexibility in home/office control.

Lights in each room can be put on a different module. Computer control permits turning lights on and off, one room at a time. The timing and sequence, following directions under computer control, can be specified with simple commands.

In order to run AC control programs, the use of support programs from the system disk (OS-65D V3.N AC) is required.

Software control of these remote switches requires running the previously stored program, "AC", by typing

RUN"AC"

This brings the device driver programs from disk. The device drivers permit a relatively simple set of commands to control the more complex functions of the lamp and appliance switch modules. The user's program must contain

a. A POKE to set the display screen state

POKE 249,1

will set a 64 by 32 character B&W (sound off) display in the same manner as

POKE 56832,1

was used to set the display state (discussed in video section)

b. Address 548 (224 hex) and 549 (225 hex) must contain the low and high bytes of the address of the AC driver routines.

These are set by the commands

POKE 548,127

POKE 549,5Ø

Having taken care of the three required POKEs, device driver programs can now be written.

The AC driver routines utilize a new BASIC command, ACTL, with the following format

ACTL DEVICE,COMMAND

where DEVICEs are numbered 1 to 16 and the COMMAND choices are as follows:

Function	COMMAND
Turn on device	65
Increase brightness (lamps only)	66
Turn all lights on (lamps only)	67
Turn off device	68
Decrease (dim) brightness (lamps only)	69
Turn all devices off	7Ø

(The total range of dimming (brightening) is accomplished in 12 steps.)

If a light is in the off state, brightening it will result in its being turned on, first.

This ACTL command can be used to turn on device number 4 (plugged into a module which has had its unit dial set to 4) by

ACTL 4,65 <RETURN>

Multiple devices, for example numbers 4 and 5 can be turned off, using the format

ACTL DEVICE1,DEVICE2, . . . ,COMMAND <RETURN>

as

ACTL 4,5,65 <RETURN>

Similarly, use of the format

ACTL DEVICE,COMMAND,COMMAND, . . . ,COMMAND <RETURN>

permits brightening device #4 through three of the 12 levels of brightness by

ACTL 4,66,66,66 <RETURN>

Another variation of the ACTL command is

ACTL DEVICE1

ACTL DEVICE2

PROGRAM LOOP (LINES 3ØØ-4ØØ IN FOLLOWING EXAMPLE)

ACTL COMMAND (LINE 5ØØ IN EXAMPLE)

ACTL COMMAND (ADDITIONAL COMMANDS, IF DESIRED)

PROGRAM REMAINDER (LINE 6ØØ IN EXAMPLE)

which can be used to slowly brighten device 1 and 2 simultaneously by

1ØØ ACTL1

```
200 ACTL2
300 FOR TIME=1 TO 12
400 FOR DELAY=1 TO 100 : NEXT DELAY
500 ACTL 66
600 NEXT TIME
```

For safety considerations, the command for "all off" (70), which turns off all lamps and appliances, was not matched with an "all on" command. The "all lights on" affects only the lamp modules.

There are now software commands to control one of the peripheral devices on the C4P system. New additions to the peripheral family will be serviced in a similar manner to the devices already described. Now that each of the available devices has been examined, they will be combined in a REAL TIME system.

THE HOME SECURITY

The first level of home security can be met with the home security alarms alone. These devices provide checking for fire, intruders or tampering with vehicles. All alarms report their status by radio-control to the home control module, connected to the C4P computer (on J3 of the C4P back panel, Fig. 1). Each alarm module contains the sensor, battery power, and a radio transmitter to assure a reliable and tamper-resistant operation.

The fire alarm can sense temperature (thermal contact) or smoke (ionization detector). The intruder alarms are silent, magnetically actuated door or window position sensors. By combining these alarms with computerized response, such as automatic dialing of the telephone emergency numbers, a rapid response to critical situations can be managed. The car alarm senses car battery voltage change; a door opening or the radio or lights left on would actuate the alarms. The intrusion and car alarms permit choice of immediate alarm or delaying for 15 seconds prior to actuating (sounding) the alarm. This gives time to disable the alarm when entering the house normally.

Additionally, a hand held alarm is available for handicapped or bedridden persons. All alarms have an effective radius of 200 ft. (60 meters) from the alarm site to the computer home control module.

The alarms are located at the computer address 63232 and the alarm control at 63233. The alarms are enabled (permitted to report back to the computer) by setting locations 63233 and 63234 to the values given in the following program:

```
10 REM PROGRAM AID ; LISTEN FOR HOME SECURITY ALARMS
20 ENABLE=0 : HEAR=0 : TRIP=0
30 ALARM=63232 : CTRL=63233 : START=4
40 POKE CTRL, ENABLE : POKE ALARM, HEAR : POKE CTRL, START
50 REM SET UP TO LISTEN TO ALARM LINES
60 FIRE=1 : BURGLAR=2 : CAR=4 : MISC=8
70 T1=PEEK(ALARM) AND FIRE
80 T2=PEEK(ALARM) AND BURGLAR
90 T3=PEEK(ALARM) AND CAR
100 T4=PEEK (ALARM) AND MISC
110 REM TESTS T1,T2,T3, AND T4 TO CHECK IF ALARM TRIP
120 IF T1=TRIP THEN PRINT "FIRE"
130 IF T2=TRIP THEN PRINT "BURGLAR"
140 IF T3=TRIP THEN PRINT "CHECK CAR"
150 IF T4=TRIP THEN PRINT "MISC ALARM"
160 GOTO 70
170 END
```

In later examples, further alarm responses will be incorporated. Alarm monitoring can be done while other programs are being run. This powerful technique is available by use of the Real Time Monitor, RTMON. Many other computer controlled responses can also be called. For example, AC, Appliance Control, can regulate light levels or sound warnings; automatic telephone dialing can summon aid.

The user has the ability to maintain detailed supervision of home security with the simplicity of conversational instructions in BASIC.

SECTION 13

PARALLEL I/O

EXTERNAL SWITCHES, ALARMS, OR INDICATORS

In AC control and home security systems, there is often need to sense switch openings or closings. Relay contacts might indicate an air-conditioner "on" for an energy management system; an open window might be read as a set of open contacts to a home security system. Individual imagination is the limit.

The C4P system provides (in the AC-21 package) the ability to sense 48 separate remote contact-pairs. Each of these contact-pairs (lines) is to be at either Ø volts or 5 volts (standard TTL levels). When these lines are computer driven (used for output), a maximum of two TTL devices can be driven at a time. If devices other than OSI peripheral devices are used, be cautioned to use good circuit practices in interfacing circuits.

The input lines are grouped as 6 sets of 8 lines (6x8=48), or 6 input registers. Associated with each input register (group of 8 lines) is a mask register (tells which of the 8 lines to ignore) and an active state register (tells whether a 5 volt or Ø volt signal is to be the chosen active state). The state of each line can be sensed by examining the register bit which reflects the state of the connected line. In the case of windows, for example, it might be desired to identify the active state as an open window in one program but in a different program to have the active state reflect a closed window. Which one is desired will depend on the program.

The associated registers, i.e., the mask register and active state register, are used by the real time monitor, RTMON, to systematically scan the input lines. When an input line becomes active, RTMON's services are requested (in the same manner as the count down timer requested service). Once again, discussion of how RTMON uses these associated registers will be put off until after examination of the hardware which is used to support it.

The associated registers are memory locations which are examined to determine how to interpret switch positions. In contrast, the hardware registers directly indicate line status, 5 volts or Ø volts. The hardware registers also indicate whether a set of lines is to receive signals (be read) or whether output signals should be sent to turn on/off devices (to be written to).

External switches which can be used to provide 5 volts or Ø volts are connected (through back panel connectors, Figure 1) to a *P*eripheral *I*nterface *A*dapter (PIA). The PIA presents groups of input lines for input or output of signals. These input or output lines are addressed in groups of 8 lines. The PIA is a single integrated circuit. Its organization and use are best explained in terms of its addressing, i.e., where the computer looks to input or output data. For this purpose, a map is created.

PIA REGISTERS

Map of the hardware registers used for input and output.

Data Register Hex Location	Data Register Decimal Location	7	Ø Bit	Control Register Decimal Location	Control Register Hex Location
C7Ø4	5Ø948	Port 1A			
			CTRL Register For Port 1A	5Ø949	C7Ø5
C7Ø6	5Ø95Ø	Port 1B			
			CTRL Register For Port 1B	5Ø951	C7Ø7
C7Ø8	5Ø952	Port 2A			
			CTRL Register For Port 2A	5Ø953	C7Ø9
C7ØA	5Ø954	Port 2B			
			CTRL Register For Port 2B	5Ø955	C7ØB
C7ØC	5Ø956	Port 3A			
			CTRL Register For Port 3A	5Ø957	C7ØD
C7ØE	5Ø958	Port 3B			
			CTRL Register For Port 3B	5Ø959	C7ØF

Each port A, port B pair is called a Peripheral Interface Adapter or PIA. These ports provide a way to enter data from the outside world into the computer and to respond with computer-generated signals to the outside. The PIA also holds or latches these input and output signals until the computer is ready to receive them (for input) or until the outside devices can utilize them (for output). Each of the two ports on a PIA (port A and port B) contains 8 lines which may be individually used for input or output.

The CA-21 option contains three PIA's. It is connected to the C4P computer by a 16 pin connector, J2, shown in Fig. 1. External devices are connected to the three sets of input port pairs. Since three sets of port A-port B pairs are accommodated (each port 8 bits wide), there are 3*2*8=48 lines available for external connection.

The operating system will initialize the scan of PIA's to include a complete CA-21 option group of PIA's as a default. Scanning fewer PIA's or scanning the PIA at 63232 decimal (F7ØØ hex) will require making the changes (POKEs) just illustrated.

For example, to scan all 48 lines starting at 5Ø948 decimal (C7Ø4 hex), all six data registers (ports 1A, 1B, 2A, 2B, 3A, 3B) must be scanned along with six control registers. Therefore, location 89Ø2 decimal must be loaded with 12−1=11 (the number of scanned registers minus one). These POKEs can be accomplished as

```
POKE 89Ø2,11  : REM LOOK AT ALL 6 DATA AND 6 CONTROL REGISTERS
POKE 89Ø9,4  : REM LOWER HALF OF C7Ø4 PIA PORT ADDRESS
POKE 891Ø,199  : REM SINCE C7 hex=199 decimal
```

(Only decimal values may be used with POKEs.)
With these POKEs, RTMON will check for an active state.

The foregoing has been a review of the connections to the PIA. Now look at the operation of the PIA. The ports (port A and port B) serve two purposes. Each port accomodates input or output signals. Additionally, these port A and port B pairs serve as data direction registers. When serving as a data direction register, the port specifies which bits serve as input and which serve as output bits. The action of the port, whether it serves as an input/output port or as a data direction register, is set by yet another register, called the control register. A control register is associated with each port. If the control register is POKEd with zeros, then the port serves as a data direction register.

When the control register is POKEd with a 4, the port reverts to its data handling function. By using a data port to serve as a data direction register, the number of hardware connections is reduced. But to understand its increased complexity of function requires paying the price of additional work. To illustrate, for example, the use of the PIA to read port 1A at location 5Ø948 (C7Ø4 hex), the steps are

1. POKE 5Ø949,Ø
   This address, one beyond the PIA port 1A address, is the control register for port 1A. A zero in the control register will allow the use of the PIA port 1A address for its alternate use, designating which bits are input or output (called a data direction register). A one indicates output, a zero an input. At the completion of this POKE, the control register contains

   5Ø949 | ØØØØ ØØØØ |

   and the port 1A will serve as a data direction register. Therefore, the command

2. POKE 5Ø948,127
   will place the bit pattern Ø111 1111 into the data direction register. The data direction register will now be

   5Ø948 | Ø111 1111 |

   Bit 7, the leftmost bit of the data direction register contains a Ø indicating that its corresponding line will be an input line. The other register bits (bits Ø to 6) are 1's, indicating that their corresponding data lines will serve as output lines.

3. The PIA port 1A is now ready to revert to its data handling function. This is achieved by

   POKE 5Ø949,4

   which commands the control register for port 1A to perform its I/O function.

4. Bit 7, the leftmost bit, was previously set as an output bit in step 2. This output can be set to a high value by

   POKE 5Ø948,64

   This is a bit pattern 1ØØØ ØØØØ. The data register (the alternate function of the port) will now contain

   5Ø948 | 1ØØØ ØØØØ |

   Likewise bit 7 could have been set to a zero by

   POKE 5Ø948,Ø

5. If it were desired to read bit 6, which was designated as an input bit, the result could be

   BIT6=PEEK (5Ø948) AND 64

   where 64 has a bit pattern Ø1ØØ ØØØØ. The 1 in the bit pattern corresponds to the desired line. To the user, location 5Ø948 appears as

	7	6	5	4	3	2	1	Ø	bit
5Ø948	X	1 or Ø	X	X	X	X	X	X	

   where X indicates that A doesn't care about the value. By ANDing the contents of 5Ø948 with the value

   Ø 1 Ø Ø Ø Ø Ø Ø

   only the value of bit 6 will be examined. If bit 6 of 5Ø948 is a zero, then BIT6=Ø; if bit 6 is 1, then BIT6=64. Testing for zero or non-zero value of BIT6 provides a convenient programming test to determine the bit 6 input line state.

   The socket pin connections are shown in appendix B; socket mating information is also provided.

A short program to make all lines for port 1A into read (input) lines and all lines for port 1B into write (output) lines follows:

```
   5 REM PIA INITIALIZATION SUBROUTINE AT 1ØØØ
  1Ø GOSUB 1ØØØ
  2Ø INPUT "SIDE (A OR B)";C$
  3Ø IF C$="A"GOTO 1ØØ
  4Ø IF C$="B"GOTO 2ØØ
  5Ø GOTO 2Ø
 1ØØ IF A$="I"GOTO 15Ø
 11Ø INPUT "OUTPUT TO A";K
 12Ø POKE X,K
 13Ø GOTO 2Ø
 15Ø PRINT"INPUT TO A IS";PEEK (X)
 16Ø GOTO 2Ø
 2ØØ IF B$="I"GOTO 25Ø
 21Ø INPUT "OUTPUT TO B";K
 22Ø POKE X+2,K
 23Ø GOTO 2Ø
 25Ø PRINT "INPUT TO B IS";PEEK (X+2)
 26Ø GOTO 2Ø
1ØØØ INPUT "STARTING ADDRESS OF PIA";X
1Ø1Ø INPUT "A SIDE I OR O";A$
1Ø2Ø INPUT "B SIDE I OR O";B$
1Ø3Ø POKE X+1,Ø:POKE X+3,Ø : REM SETTING CTRL REGISTER TO ZERO
1Ø4Ø IF A$="I" THEN POKE X,Ø : REM PERMITS SETTING DATA DIRECTION REGISTER
1Ø42 IF A$="I" THEN GOTO 1Ø5Ø
1Ø45 POKE X,255 : REM IF NOT INPUT, THEN SET AS OUTPUT
1Ø5Ø IF B$="I" THEN POKE X+2,Ø
1Ø52 IF B$="I" THEN GOTO 1Ø6Ø
1Ø55 POKE X+2, 255
1Ø6Ø POKE X+1,4:POKE X+3,4 : REM CTRL REGISTER TO FORCE I/O
1Ø7Ø RETURN
```

Multiple lines may be checked at one time.

The home security system addressed at 63232 (F7ØØ hex) is also a PIA port. It is one of two ports. Two ports (of 8 bits each) are available, with the first 4 bits being reserved as:

- Bit 3: Misc. Alarm
- Bit 2: Car Alarm
- Bit 1: Intruder Alarm
- Bit Ø: Fire Alarm

Location	(Hex)	Bit 7 6 5 4 3 2 1 Ø
63232	F7ØØ	Port A
63233	F7Ø1	CTRL A
63234	F7Ø2	Port B
63235	F7Ø3	CTRL B

A program to handle this device is similar to the previous programs. For example, to check for a fire alarm

```
1Ø REM SET PORT A AS INPUT, LOOK AT BIT Ø, THE FIRE ALARM BIT
2Ø POKE 63233,Ø : POKE 63232,1 : POKE 63233,4
3Ø IF PEEK (63232) = Ø THEN GOTO 1ØØ
4Ø GOTO 2Ø
```

This program segment will continually look at the input port and check for the bit assigned by *OSI* to fire alarm checks.

SECTION 14

CONNECTION OF SIXTEEN PIN BUS DEVICES

Ohio Scientific is pleased to introduce a unique new product line—The 16 Pin I/O BUS. With this system, it is possible to add up to eight special function boards while occupying only the backplane slot.

This is made possible by a novel BUS extension method which allows decoding and timing signals plus eight bits of data to be carried on standard, inexpensive 16 pin ribbon cables.

Up to eight inexpensive 16 pin cables with standard DIP connectors may be attached to a single CA-2Ø board which in turn occupies one slot of the standard Challenger backplane. Alternately, one 16 pin I/O BUS cable may be attached to the A-15 board at the rear of all C4P products. Note, in the case of the C4P-MF this allows system expansion beyond the normal four slot backplane.

Currently, five HEAD END CARDS are available for interconnection to the system via the CA-2Ø or CA-15 boards.

COMPUTER INTERFACE TO SIXTEEN PIN I/O BUS

The 16 pin I/O BUS may be attached to the computer via two different boards—the CA-15 or the CA-2Ø. The descriptions of these boards are as follows:

CA-15 BOARD

The CA-15 board is a standard accessory interface installed on the following Ohio Scientific systems: C4P-MF, C4P-DMF, and C8P-DF.

The CA-15 is mounted at the rear of the computer and contains the following interface connection:

Joystick and numeric keypad
Modem and serial printer
Sixteen PIA lines (normally used for the Home Security system—AC-17P)
Sixteen Pin I/O BUS

The interconnect for the Sixteen Pin I/O BUS is simply a 16 pin DIP socket. To use the BUS, the only thing necessary is to attach one end of the 16 pin ribbon cable to the CA-15 board and the other end of the cable to one of the HEAD END CARDS.

Please note that some of the HEAD END CARDS require more power than may be practically carried via the ribbon cable alone. Therefore, some of the cards require auxiliary power supplies.

CA-2Ø BOARD

The CA-2Ø. board contains all the necessary logic to decode eight distinct HEAD END CARD interfaces. The actual interconnect, as with the CA-15, is via simple 16 pin DIP sockets and standard 16 pin ribbon cables.

The CA-2Ø board also requires one slot of the computer's backplane. But remember, from this one slot access is gained to a maximum of eight accessory boards.

The CA-2Ø is recommended for use in the Ohio Scientific C2 series and C3 series computers. It can also be installed in C4P and C8P series systems with some modification to the CA-15 interface.

Since the logic required for the I/O BUS interface is simple, an additional feature was added to the CA-2Ø board—a crystal controlled "time-of-day" clock (hardware) subsystem. The operation of the clock, excepting reading time and setting time, is totally independent of the host computer. As a matter of fact, with the included on-board, auto-recharging, battery back-up, the computer may actually be turned off for several months without losing time.

The features of the clock subsystem are as follows:

Hours, minutes, seconds and 1/1Ø seconds
Day of week

Day of month
Month of year
Four Year calendar

In the C2 and C3 series computers, the CA-2Ø board can actually control the power cycling of the entire computer when equipped with an optional power sequencer package. This means a time (month, day, hour, etc.) may be preset within the clock subsystem and when that preset time agrees with the actual time, A.C. power is applied to the entire computer system through the power sequencer. At a later time, the system's A.C. power may also be removed and the system shut down under software/clock subsystem control.

For applications where the clock subsystem is not required, the CA-2ØA will perform all the Sixteen Pin I/O BUS functions associated with full-feature CA-2Ø.

HEAD END CARDS

HEAD END CARDS is a general name used to describe any or all of the special function boards which attach to the Ohio Scientific Sixteen Pin I/O BUS. There are currently five such boards and, with the exception of the CA-22, they will only interface with the computer via the Sixteen Pin I/O BUS.

Please note, as detailed earlier, a CA-15 or CA-2Ø board must be used at the "computer end" of the Sixteen Pin I/O BUS to complete the interface.

In the following pages a brief product and application description of the currently available HEAD END CARDS will be presented.

THE CA-21 BOARD—BIT SWITCHING AND SENSING

The CA-21 is a 48 line parallel I/O board featuring three 6821 PIAs (peripheral interface adapters) and prototyping/interconnect areas.

The use of PIAs in the design allows for maximum interface versatility as any one of the 48 I/O lines may be configured as either an input or an output. As outputs, each line is capable of driving a minimum of one standard TTL load.

Additional versatility is added because 24 of the lines, when configured as outputs, may simultaneously function as inputs. This feature, although somewhat confusing, is extremely useful for applications such as switch matrix decoding.

Each of the 48 lines is brought out to two foil pads (suitable for wire wrap stakes) as well as a location on one of four 12 pin Molex-type female edge connectors. There are also eight 16 pin DIP socket locations which are intended for use as prototyping areas. Additionally, the 12 PIA "hand-shaking" lines are brought to 12 single foil pads.

The CA-21, with proper buffering, may be used for virtually any computer controlled bit switching or bit sensing application imaginable. With a full complement of eight CA-21s interfaced via the CA-2Ø, a total of 384 individually controllable I/O lines are possible!

An interesting application using one CA-21 board would be a complete, if somewhat slow, emulation of the standard Ohio Scientific BUS.

A more practical application might be augmenting the standard Home Security System (AC-17P) with "hard-wired" sensors.

One type of sensor easily added is a standard window "perimeter detector." This could be done with commercially available adhesive foil tape. A break-in (through a broken window) could then be detected by sensing a break in the foil tape.

Another useful application that could be set up in concert with the AC-12P wireless A.C. Remote Control, is sensing when a room is entered. This could be accomplished with pressure-switch door mats or door switches. When room entry is detected, the lights could be turned on or turned off on exit.

For designing any sort of dedicated control system, the CA-21 is an ideal choice. It is possible to easily sense many types of input (pressure transducers, flow sensors, switches, etc.) while controlling outputs from a simple single LED display to a network of solid state relays controlling A.C. power.

THE CA-23 BOARD—EPROM PROGRAMMER

The CA-23 is an EPROM programmer designed for use with the growing families of 5 volt only EPROMS. With the CA-23 you can program and verify all 1K through 8K byte EPROMs of this type. Note that these parts are often identified as 8K—64K bit EPROMS.

The CA-23 can program (or verify) data in two basic modes—EPROM to/from EPROM or EPROM to/from computer RAM memory. Additionally, EPROM data may be read directly into the computer's RAM memory.

There are four LED indicators on the CA-23. The first is "SOCKET UNSAFE." This means that a programming voltage is present at the socket and if an EPROM is removed or inserted it is likely to be damaged.

The second indicator is "PROGRAMMING." This means that the EPROM is currently being programmed.

The third indicator is "ERROR." This means that somewhere along the line a programming attempt was unsuccessful.

The final indicator is "PROGRAM COMPLETE." This means that the program and verification were successful.

The most intriguing application for this product is the creation of "custom" parts for the computer or peripherals. This could range from a new system monitor to a new high level language. It could even include a new character generator for the CRT or printer. Note, however, tinkering around with the internals of computers and peripherals requires a fairly high degree of technical expertise. Also, most manufacturer's warranties are voided by these types of modifications.

Several OEM (original equipment manufacture) and Research/Development applications will be immediately obvious to those involved in that work.

The CA-23, as previously mentioned, is designed for use with 1K through 8K byte EPROMS. These parts come in various package styles and have various product names. For example, Intel's 2K × 8 part is the 2716, Texas Instruments' part is known as the 2516.

The CA-23 has both 24 pin and 28 pin zero insertion force sockets for reading, programming and verifying the EPROMS.

THE CA-24 BOARD—PROTOTYPING

The CA-24 is a solderless bread-board designed for prototyping, experimental and educational applications.

The bread-boarding is made up of seven solderless plug-strips of the type manufactured by AP Products. Two of the plug-strips contain a connection matrix of 5 by 54 connections and are used as signal distribution points. Another pair of 96 location plug-strips are for powering the bread-board area. The actual experimenter area is comprised of three plug-strips, each with a 1Ø by 64 location connection matrix. Additionally, sixteen LED indicators and sixteen DIP switch positions are provided for signal observation and control functions.

Board I/O is via TTL latches and bi-directional PIA ports as well as direct (buffered) data, signal and control lines from the computer BUS. This method allows you direct interconnection of devices such as 685Ø ACIAs in addition to doing more "isolated" and/or independent circuits.

The CA-24 also contains a "clock" generator which is continuously variable from approximately 25,ØØØ Hz. through 7Ø,ØØØ Hz. It is also possible to connect the clock to an on-board 16 stage divider chain. This allows division of the fundamental frequency by as little as 21 (2) to as much as 2^{16}(65,536).

The applications for the CA-24 are primarily prototyping and experimenting. Parts may be inserted and removed from the terminal strip blocks over and over. Interconnection of parts is accomplished simply through the use of solid, narrow gauge wire jumpers. Errors in design or connection are extremely easy to correct.

The CA-24 lends itself very well to structured experiments that are common in the educational environment. It is an ideal tool to aid in the teaching of computer and computer interface fundamentals.

THE CA-25 BOARD—ACCESSORY INTERFACE

The CA-25 is designed to implement some of the functions normally associated with the CA-15 interface board. It allows direct connection of the Home Security System (AC-17P) and/or the Wireless A.C. Remote Control System (AC-12P) to C2 and C3 series computers. Additionally, those who own an older Ohio Scientific computer can now easily connect these systems to it.

An extremely useful application of the CA-25 is associated with small business systems. Using the CA-25 with the Home Security System, and perhaps a CA-15V (Universal Telephone Interface with speech synthesizer output), the computer could do payroll, inventory, etc. by day and "guard" the shop by night.

THE CA-22 BOARD—ANALOG I/O

The CA-22 is a high speed analog I/O module. Although the CA-22 is classified as a HEAD END CARD, it differs from the rest of the family in that it may also be plugged directly into the computer's standard internal BUS. This allows for maximum flexibility in the use of the CA-22.

The analog input section of the CA-22 consists of a 16 channel analog multiplexer. This means that up to 16 separate signals may be connected directly to the CA-22. Also included is a sample and hold circuit followed by the analog to digital converter circuitry.

The A to D converter is capable of either 8 bit or 12 bit operation. These options are selectable under software control.

The accuracy of the converter is plus or minus one in the least significant bit. The stability of the circuit is rated at one millivolt drift per degree Celsius.

The A to D conversion is extremely fast. It is capable of digitizing up to 66,ØØØ samples per second in the 8 bit conversion mode and 28,ØØØ samples per second in the 12 bit mode. Shannon Sampling Theory states that signals should be sampled at twice the highest frequency present. Therefore, it is possible to convert signals with a frequency greater than 3ØK Hz. Clearly, high fidelity audio is well within the spectrum of the CA-22.

The multiplexer has very high impedance inputs and is capable of accepting inputs in the range of −1Ø volts through +1Ø volts. The input is jumper selectable for other settings including a single sided range of Ø through +1Ø volts.

Due to the indeterminable nature of the actual inputs that may actually be applied to the CA-22, only the multiplexer inputs are brought out. However, a quad op-amp is laid out in foil which may be populated in several different modes to handle some of the more "common" input configurations.

The analog output section of the CA-22 consists of two identical high speed digital to analog converters. Each DAC can convert either 8 bits or 12 bits of data. Data input to the DACs is latched in such a manner that, when in the 8 bit conversion mode, the other four (of the total of twelve) bits are continuously output at a predefined value which may, of course, be defined under software control.

The output of each DAC is buffered with a high speed op-amp capable of changing output voltage at the rate of 2Ø volts per microsecond. The standard configuration of each output is bi-polar with a voltage swing of −1Ø volts through +1Ø volts. This is jumper selectable to allow a uni-polar output of Ø through +1Ø volts.

Some additional I/O capacity is provided on the CA-22. There are three TTL level inputs and six open collector logic outputs. These are strappable to be either standard TTL level outputs or high-voltage outputs.

The CA-22 can be used for a multitude of analog sensing and/or analog controlling applications.

Using the proper transducers and the 16 input channels, it is possible to monitor the temperature in several zones of a home or office. By extending this system with a CA-21, precise temperatures can be maintained by switching the proper controls on and off.

Another interesting, if somewhat obvious application, is in audio processing. Reverberation, phase shifting and echoing are just a few of the uses implementable.

If blocks of RAM were used for data storage, other experiments such as frequency doubling, etc., could be performed.

If more sophisticated software techniques, such as fast Fourier transforms, are applied to store input data, very elaborate signal processing becomes realizable. Projects such as audio spectrum analyzers and speech recognition experiments are certainly practical. Note, in these types of applications, it is likely that some signal pre-processing in hardware is certainly beneficial—if not totally necessary.

Employing both DAC outputs and the on-board unblanking circuit, X-Y oscilloscope plotting is an interesting application. By using these techniques and one or more of the analog inputs, a digital storage scope can be constructed. Note, both of these applications require access to an oscilloscope capable of X-Y input as well as blanking.

SUMMARY

With the introduction of the 16 pin I/O BUS, Ohio Scientific has opened a new world of interfacing capabilities for both the large and the small computer user.

Systems ranging from totally automated sampling and control stations to complete R/D setups to educational lab stations are now available via standard building blocks and standard computer systems.

For pricing and availability, contact the nearest Ohio Scientific dealer.

SECTION 15

MODEM AND TERMINAL COMMUNICATIONS

Each character stored or moved is represented by 8 bits (ones or zeros). Normally, there is data on eight data lines (called a data bus), simultaneously. This is convenient when the cost of maintaining multiple lines is low, due to short line lengths. For longer lines, extra circuits for each line are necessary to maintain data signal fidelity. Also, the cost of maintaining long data lines must be balanced against the speed and convenience of having all data bits simultaneously available.

Certain devices require serial data handling. Serial data handling treats one bit (off-on) at a time, rather than all data bits simultaneously. The serial devices are low speed, with no ability to simultaneously transmit or receive more than one bit at a time. Bits are collected by the serial data device until a complete character is available. Then, when the complete character has been received, it is sent in parallel (all bits simultaneously) to the computer for processing. Serial data is handled by an *A*synchronous *C*ommunications *I*nterface *A*dapter (ACIA) which converts the parallel (simultaneous) data into serial data for transmission (or reverses the process for reception).

A simple analog might suggest the function of the ACIA. Consider that the input from a computer is typically 8 parallel, simultaneous, input bits.

A picture analogous to this process can be seen, as in Fig. 18A, by considering the placing of black and white marbles, simultaneously, into the holes in a pipe.

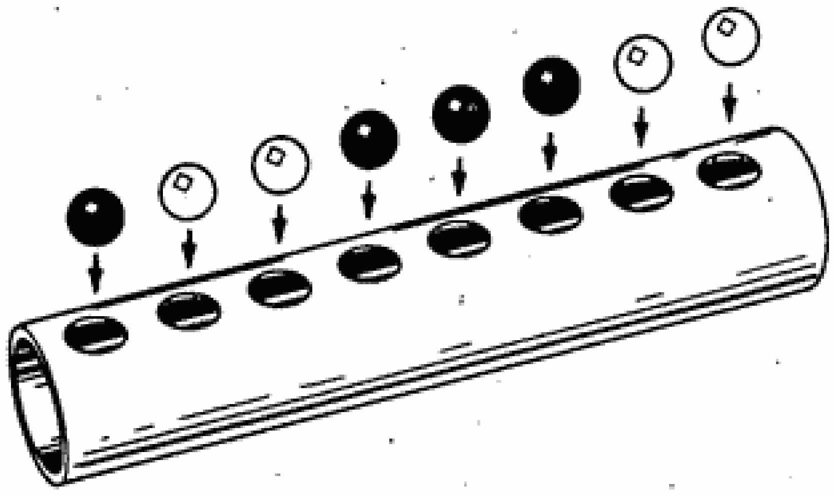


Fig. 18A Marble Placement

If those marbles are now rolled out the left end, a time sequence of marbles is seen, as in Fig. 18B.

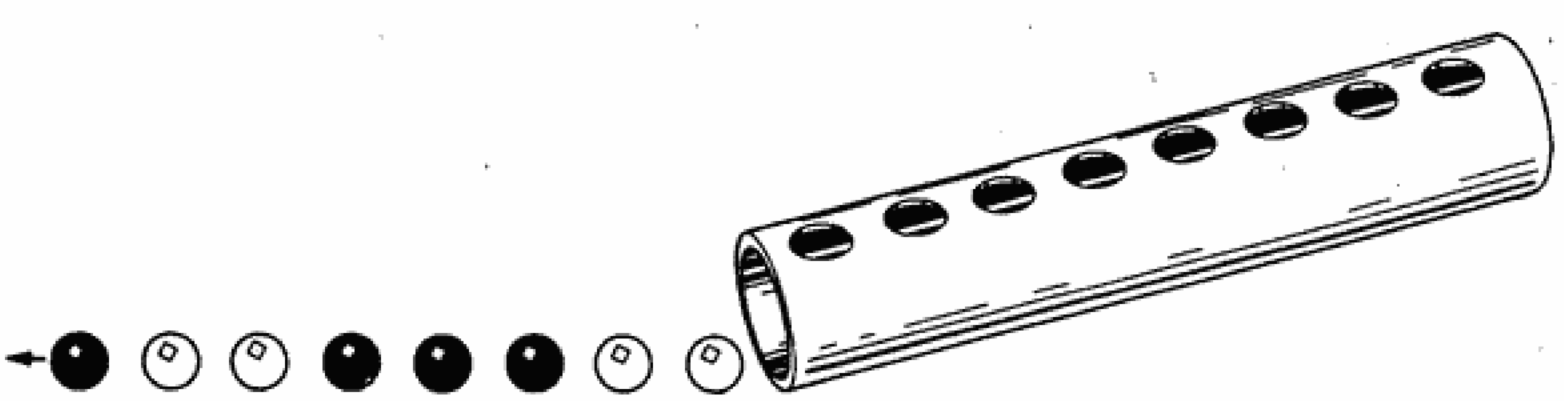


Fig. 18B Marble Time Sequence

The ACIA performs the electrical equivalent of this action. For devices limited to low mechanical speeds (such as printers and plotters) or low data rates (such as telephone lines and modems), this serial (or sequential) handling of data bits can be tolerated. The advantage is the economy of requiring fewer wires (and the circuits to drive them). Whereas parallel transmission requires 8 wire pairs for simultaneous presentation of all 8 data bits, serial transmission is accomplished by the use of only 1 pair, as illustrated in Fig. 19.

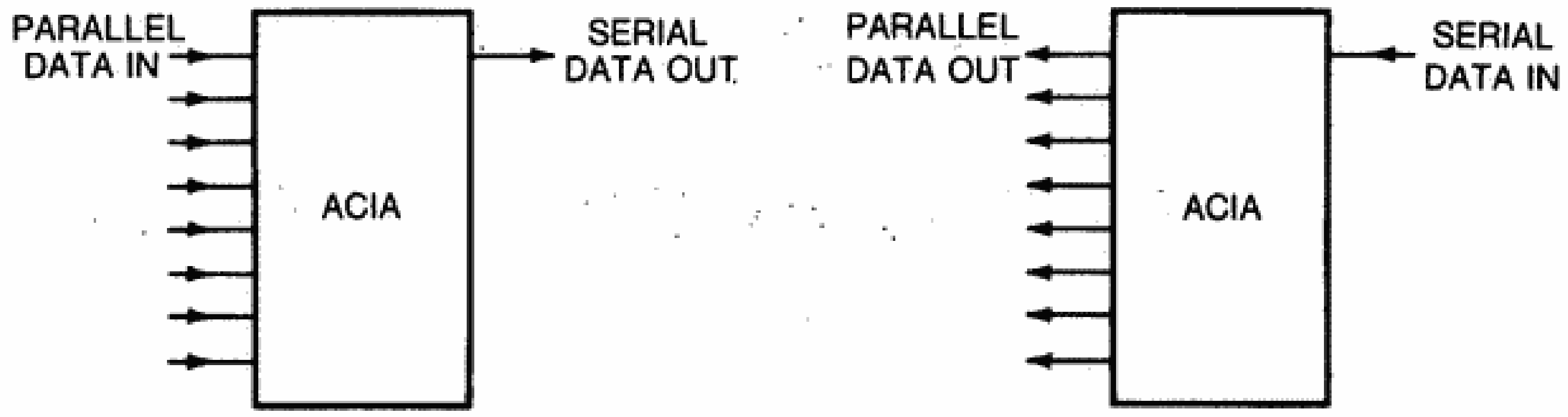


Fig. 19 Parallel Verses Serial Transmissions

The system will normally be set up with the information handling rate (baud rate) set at 12ØØ bits per second (12ØØ baud). For the modem use, this must be changed to 3ØØ baud. The two choices are given by

POKE 64512,1 : REM 12ØØ BAUD RATE

or

POKE 64512,2 : REM 3ØØ BAUD RATE

In contrast to the ACIA, Parallel Interface Adapters (PIA's) handle all 8 bits of a character's data simultaneously. These serve as interfaces to the outside (of the computer) world.

The ACIA's output is available on J8 for the printer output and J9 for the modem connection, as shown in Figure 1.

An ACIA driver program, which is used to drive a slow printer, is shown in the next section.

The following is a general summary of the sequence of steps necessary to use the C4P as a terminal:

1. Connect a modem to the modem port, J9.
2. Load the modem program provided by Ohio Scientific into the C4P. When it is loaded the computer will respond READY.
3. Dial the number of the remote computer. When the number dialed answers there should be heard a high pitched tone. Insert the phone in the modem and follow the instructions displayed on the screen. The computer called will probably require the entering of a user code and password.

The use of acoustic coupled modems extends the resources of the C4P without requiring commitment of available computer slots. The power of the C4P is not limited by this feature.

Ohio Scientific now offers Compuserve packages containing documentation and software designed to enable OSI personal computers equipped with modems to fully access the Compuserve (formerly Micronet) System. Contact your dealer for more information.

SECTION 16

PRINTER COMMUNICATIONS

The printer and the modem are served by the ACIA. Both devices require low data rates, due to limited frequency response of the devices (whether from mechanical reasons or electrical reasons).

Either a serial printer or a telephone line modem may be attached to the ACIA output (J8 for printer, J9 for modem, as specified in Figure 1.). However, only one of those devices may be connected at any one time. That is, power may not be applied to the printer and modem simultaneously. It is possible to store modem data on disk files for later printing, so this is not a difficult restriction. Only one device will have its input accepted at one time.

The C4P Cassette based system, which uses BASIC-in-ROM, can output to the printer in the same manner as output to cassette. If the command SAVE is entered, then all subsequent output which would normally appear on the screen is routed to both the screen and the printer. Output will continue to be routed to the printer as well as the screen until the user enters the following sequence of commands:

LOAD <RETURN>

<SPACE> <RETURN>

These two commands terminate output to the printer in the same way that they terminate output to the cassette recorder when the switch is set for cassette input/output.

For example, a program in the workspace can be listed on the printer by the following series of commands:

SAVE

LIST

LOAD

<SPACE>

As usual, each of these commands should be followed by <RETURN>. The program will begin listing after the command LIST is entered. The command LOAD should be entered after the LISTING is complete. If the printer is not on line or is connected incorrectly (or if the selector switch is turned to printer when no printer is connected) then the computer will stall when the command LIST is entered until the problem is corrected or <BREAK> is depressed.

If a program is RUN after the command SAVE is entered the the results of any PRINT statements are displayed on both the screen and the printer.

9-DIGIT EXTENDED BASIC UNDER OS-65D—PRINTER USE

When OS-65D is being used with C4P MF, output can be directed to the printer by changing the output flag. This is accomplished by a disk operating system command. The following illlustrates the method of accomplishing this:

DISK!"IO , Ø1" — this directs subsequent output to the printer only

DISK!"IO , Ø2" — this directs subsequent output to the screen only

DISK!"IO , Ø3" — this directs subsequent output to both the printer and the screen.

The default mode sets the output flag to send output to the screen. The output flag is automatically reset to "Ø2" (the screen) whenever the computer encounters a snytax error or an abnormal condition such as a CONTROL-C halt to a listing or run of a program.

For the purposes of printer output, setting the output flag to "Ø3" has very much the same effect as entering SAVE when using BASIC-in-ROM. The output to the printer can be terminated by resetting the output flag to "Ø2" with the command DISK!"IO , Ø2."

Under OS-65D the command LIST#1 can be used to list the contents of the workspace on the printer without the necessity of changing the output flag with the DISK!"IO" command. The program is listed only on the printer (not on the screen) when this command is entered.

A further discussion of the I/O capabilities under OS-65D is covered in the appendix.

Alternatively, PEEKS and POKES can be used to address the ACIA port, directly.

The ACIA port may be addressed by using the ACIA control register address of FCØØ hexadecimal (64512 decimal) and its data register of FCØ1 hexadecimal (64513 decimal). Reading or writing can be accomplished using the BASIC PEEK and POKE commands.

The simple program for use of the printer is:

```
5 REM PRINTER PROGRAM
1Ø POKE 64512,1 : REM SET 12ØØ BAUD RATE
2Ø A$="NOW IS THE TIME FOR ALL GOOD MEN"
3Ø FOR T=1 TO 2Ø : REM PRINT 2Ø TIMES
4Ø FOR K=1 TO LEN (A$)
5Ø A=ASC(MID$(A$,K,1))
6Ø FOR DELAY=1 TO 2 : NEXT DELAY
7Ø REM WE HAD A SLOW PRINTER
8Ø POKE 65413,A
9Ø NEXT K : REM MESSAGE COMPLETE
1ØØ POKE 65413,1Ø : REM LINE FEED PAPER
11Ø POKE 64513,13 : REM CARRIAGE RETURN
12Ø NEXT T : REM DO ALL 2Ø LINES
13Ø END
```

prints the message

NOW IS THE TIME FOR ALL GOOD MEN

twenty times, illustrating the ACIA function. This program was designed to overcome device limitations, specifically a slow printer.

The alternate method of addressing the ACIA for printer control is called I/O Commands. It is detailed in the Appendix, as are examples of its applications.

SECTION 17

ADVANCED TOPICS

Advanced topics include extensions of previously examined subjects and introduction to new topics which are of need to very specific users. A high level of support is available from OSI software. Some aspects have been mentioned in prior sections. Details which are required for definitive use can be found in the appendix of this manual or in the specific manuals on each subject. All of these topics are of an advanced nature, beyond the earlier treatments.

PLOT BASIC

In the graphics section, the character set was examined and produced displays for non-text materials. The ability to write programs adequate to plot a curve is well within the skill of the user who has read this far. The convenience of obtaining plotting by using a function, such as SIN(X) for LOG(X), requires adding these new plotting function names to the reserved list of names in BASIC. For this convenience, and the many details it requires, OSI provided an advanced plotting package, PLOT BASIC.

There are nine functions available in PLOT BASIC, named PLOTØ through PLOT8. These functions allow the user to plot single squares, lines or rectangles with any of 256 characters and any of sixteen colors in the 32 x 64 mode. Two functions allow higher resolution point and line plots. Two functions allow the user to call, by name, a previously stored figure to the screen and move it in any of eight directions, off the screen and back on, saving and restoring any background.

For plotting purposes, the squares on the screen are assigned Cartesian coordinates (x,y coordinates) with the default origin at the lower left corner of the screen. The position of the origin may be changed by PLOTØ. PLOTØ through PLOT4 and PLOT7 and PLOT8 are "standard resolution" commands with the x-coordinates of the squares on the screen ranging from Ø to 63 and the y-coordinates from Ø to 31. On most monitors, some of the squares with high or low y-coordinates will be invisible. PLOT5 and PLOT6 allow the user to reference the screen in a higher resolution 64 x 128 mode. Thus, for PLOT5 and PLOT6, the x-coordinates range from Ø to 127 and the y-coordinates from Ø to 63. PLOTØ through PLOT6 do not allow the user to reference coordinates outside the ranges given above. PLOT7 and PLOT8, however, allow the user to reference coordinates off the screen with both x and y ranging from −128 to 255. These exhibit a "wraparound" effect with coordinates that differ by 256 referencing the same point, for example, x = 128 is the same as x = −128 or x = −1 is the same as x = 255.

The PLOTBASIC package provides high resolution while retaining simplicity of use. It can serve applications in business, science, and teaching with equal facility, since the calling function is just another BASIC function.

FILES

In a disk operating system, the efficient use of the disk resource is provided by foresight in file organization. Short, modular programs to perform a specific service should be stored on disk. Each file can be called by name or by disk location. The disk resident programs which are called use the same region of memory as a previously run disk program (called overlaying). The use of disk programs permits large programs, broken into modules, to run without the need of much memory. The price paid is the delay in transferring program from disk to memory. Short, repeatedly used programs should stay memory resident to prevent the waste of time in threshing about from disk to memory. Longer, less frequently used programs should be stored on disk and called in for use as needed.

The specifics of storing and recalling files has been covered in Section 9, "Storing Files on Cassette or Disks." I/O distribution, the use of DOS and BASIC commands for detailed handling is covered in Appendix K.

Examples of embedded file commands are found in Appendix M, "USR(X) Function." Examples which bring in rapidly executing machine code routines, for example, screen clearing routines, require only a single line in the BASIC program in order to be called from disk. In contrast, the equivalent BASIC program would be uncomfortably slow.

The key to efficient use of disk is modularity of programs, where transfer of programs from disk occupies a small fraction of the total program running time.

HOME CONTROL AND REAL TIME OPERATING SYSTEMS

REAL TIME CONTROL OF DEVICES

The heart of AC control lies in being able to run programs of immediate interest while a secondary program sits "in the background" waiting to be run. At periodic intervals, set by a hardware timer, the primary program ("in the foreground" of the computer's attention) is exited, at which time the secondary or background task is serviced. Then the primary task is re-entered and execution picked up where it was previously left. Note that all of this is happening very rapidly.

Background tasks are simple, rapidly computed programs which require periodic attention. Updating a clock display or checking home security status are examples of such a task.

The operating system OS-65D V3.2 HC contains a program "RTMON" which decides which program, foreground or background, should be run.

In addition, there are three programs, AC, AC1 and AC2 which support the use of AC control accessories. The program AC contains no buffers; AC1 contains 1 buffer; AC2 contains 2 buffers. When making copies of this disk, copy only the version of this AC control program (AC, AC1 or AC2) needed.

The demonstration disk will show some examples of the usefulness of AC control.

To facilitate writing personalized programs, the following sections will show the features

1. time of day clock
2. timing events
3. AC control and home security switches

Later sections will show how to integrate these features into a real-time system for personal applications.

REAL TIME CLOCK

TIME OF DAY CLOCK

The clock is a basic building block of a real time control system. The time of day clock does not have to be enabled; it runs continually under the 3.N HC operating system. To set the time of day clock, hours are entered into location 948Ø, minutes in to 9481, and seconds in to 9482. The commands are

```
POKE 948Ø,H (H= number of hours)
POKE 9481,M (M= number of minutes)
POKE 9482,S (S= number of seconds)
```

The clock is a 24 hour clock which resets the time at 23:59:59 back to Ø:Ø:Ø. Location 9493 holds the count of the number of 24 hours periods (i.e., days), which have been counted.

Time is read by the PEEK command. For example:

```
1Ø REM INPUT TIME TO SET CLOCK
2Ø INPUT "HOURS, MINUTES, SECONDS";H,M,S
3Ø POKE 948Ø,H:POKE 9481,M: POKE 9482,S
4Ø REM NOW TO PRINT OUT TIME
5Ø H=PEEK(948Ø):M=PEEK(9481):S=PEEK(9482)
6Ø PRINT H; ":";M;":";S;"LOCAL TIME"
7Ø END
```

will permit setting, then displaying the time. Replacing statement 7Ø with

```
7Ø GOTO 5Ø
```

will continually print the time.

The time entry for 2:4Ø A.M. and 2 seconds, would be

2,4Ø,2 <RETURN>

where a comma separates each numerical entry.

COUNT DOWN TIMER

The count down timer is an event timer which functions like an egg timer. A time count is loaded (set into) the timer which then counts down to zero.

Rather than have to check the current value of the timer count, a flag is raised when the count reaches zero.

To operate the count down timer, the count down timer is loaded with the hours in location 224, the minutes in location 225, and the seconds in location 226.

Starting the count down timer is accomplished by placing a 1 in location 223. Disabling the count down timer (turning it off) requires a Ø in location 223.

A program to set the count down timer and start it running is

```
1Ø POKE 223,Ø
2Ø INPUT "HOURS, MIN, SEC COUNTDOWN";H,M,S
3Ø POKE 224,H
4Ø POKE 225,M
5Ø POKE 226,S
6Ø REM NOW START TIMER
7Ø POKE 223,1
```

A program could check the one variable, "TEST," to determine whether the hours, minutes, and seconds had elapsed by

```
8Ø TEST = PEEK(224) + PEEK(225) + PEEK(226)
9Ø IF TEST=Ø THEN GOTO 1ØØØ
1ØØ GOTO 8Ø
1ØØØØ PRINT "TIME IS UP"
```

The real value of the timer, however, lies in its ability to request the services of the real time monitor, RTMON. RTMON permits interrupting user programs when the count down timer reaches zero. This switching of priorities from one program to an interrupting program allows flexible programming. These uses will be discussed further after looking at some other devices and features available for home and appliance control.

REAL TIME MONITOR, RTMON

The Real Time Monitor, RTMON, acts as a watchdog, responding when either the count down timer counts down to zero or a PIA device is sensed to be "active". The internal computer hardware interrupts processing every 4ØØ milliseconds (.4 seconds) to update the count down timer and the time of day clock.

Should either the count down timer go to zero or a PIA device line go "active", then computer control is immediately passed to the program, RTMON. Within the program RTMON, you may decide what action is to be taken.

A typical RTMON program should deactivate the timer by

POKE 223,Ø

This allows servicing the interrupt without having the timer time out. This would avoid two interrupts occurring simultaneously; however, this uncertainty of occurrence accounts for only a few microseconds. Examining the timer contents and the PIA lines of interest will determine whether a PIA or the timer requested service. Before exiting RTMON the program should

POKE 222,1

to re-enable RTMON so the RTMON can be recalled by future interrupts. If there are no further programs to return to from RTMON, then RTMON can be terminated with a return to BASIC by

```
RUN"BEXEC*" : END
```

The operating system will then turn control over to the BASIC interpreter.

Within the operating system (specifically the OS-65D V3.N HC, Home Control Operating System), certain provisions are made for monitoring and responding to all PIA lines. These special provisions are made for the devices hung on the 48 lines from 50948 to 50958 (C704 to C70E hex) and for the 16 lines at 63232 and 63234 (F700 and F702 hex).

To sense an "active" state on a PIA line, each register of the PIA is matched to two associated registers. A "mask register" (this indicates which bits of the PIA are to be monitored) and an "active state register" (this indicates whether a high level, '1', or a low level, '0,' is the active state). RTMON will be called by the operating system if a bit is not masked out and has reached its alarm state.

These memory locations are illustrated in Fig. 20 as a map

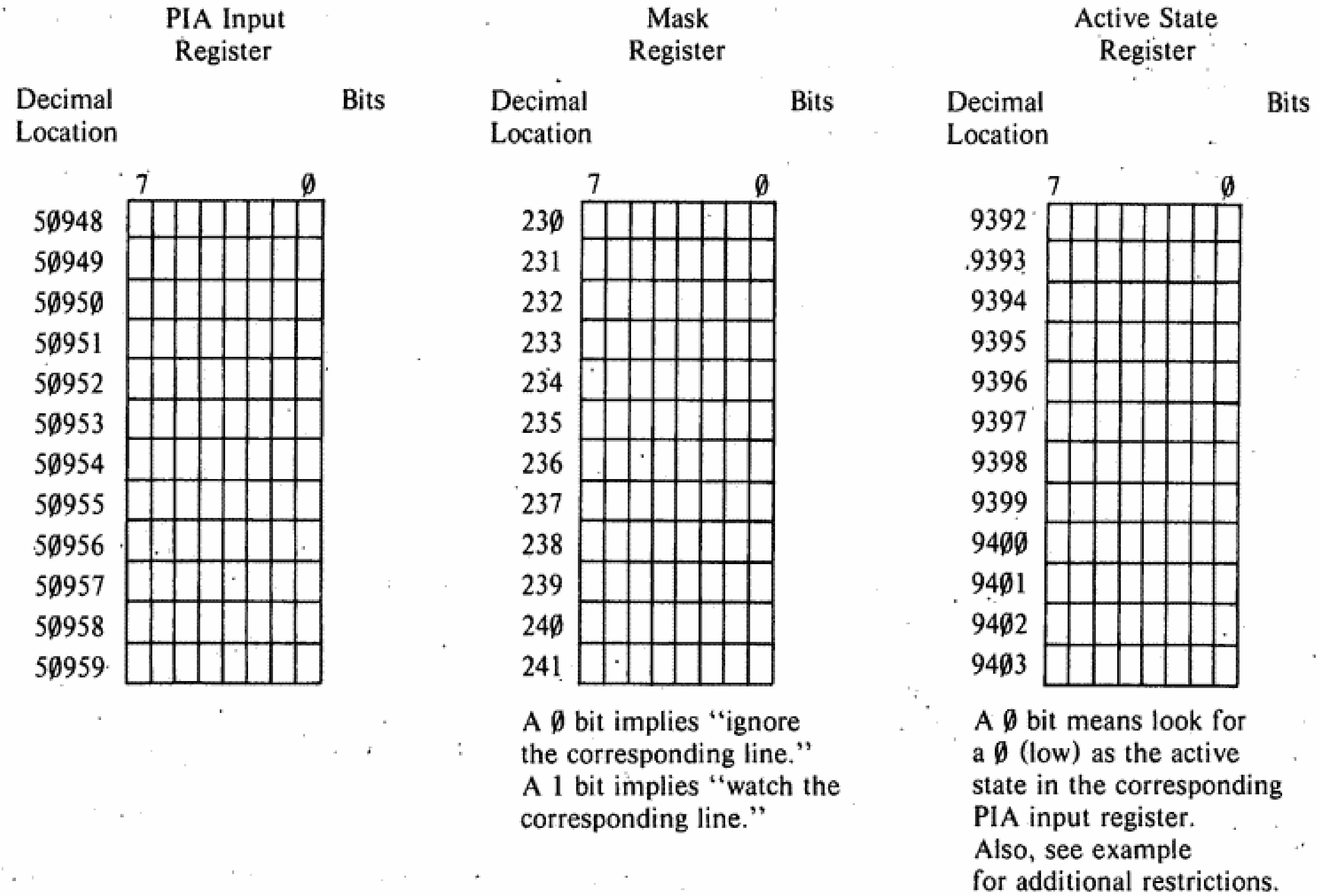


Fig. 20 RTMON Memory Location

Ignore a bit in the PIA data registers when the corresponding bit in the mask register is a 0. If the mask register bit is set to 1, then the corresponding PIA data register bit is examined.

If a bit from a PIA register (data or control register) is to be ignored (by placing a 0 in the corresponding bit position in the mask register), then, a 1 must be placed in the corresponding position of the active state register.

The choice of which registers are to be scanned is made by POKEing (placing) the address of the first register to be scanned in 8909 and 8910. The lower half of the address (low byte) is POKEd in 8909 (22CD hex) and the upper half of the address (high byte) is POKEd in 8910 decimal (22CE hex). Place the number of registers to be scanned (minus one) in location 8902 decimal (22C6 hex).

For example, to examine bit 6 of the PIA port at location 50948 decimal (C704 hex), place the bit pattern 0100 0000 (64 decimal) into the mask register at 230 decimal (E6 hex). This will force the ignoring of all but bit 6. The corresponding active state register at 9392 decimal (24B0 hex) should contain the bit pattern 1011 1111 (183 decimal, B7 hex) in order for a Ø to indicate the active state. If a 1 is to be the bit 6 active state, then the bit pattern should be 1111 1111 (255 decimal, FF hex).

If all 8 bits of a mask register are zero (ignore all data bits) then no special value need be placed in the active state register since it will be totally ignored.

Though examination of the control registers for each port is probably not wanted or needed, this ability is provided (it is possible to examine the interrupt lines of the PIA, for example).

If it is not specified which set of PIA ports to scan, the operating system will choose 5Ø948 decimal (C7Ø4 hex) as the starting value. This is the choice of the CA-12 option PIA's.

A GREENHOUSE EXAMPLE

The following is an AC control example which monitors a home greenhouse. While enjoying normal use of the computer, it is desired that a low temperature alarm be available "in the background." If the temperature should drop below a preset value, the operator is to be informed of the event. Additionally, it is considered advantageous to have an hourly signal sent to the greenhouse to spray the plants.

Both timer and alarm tasks are well suited to the C4P system. These tasks are performed by the real time monitor, RTMON.

A circuit which will accomplish the alarm function is drawn in Fig 21.

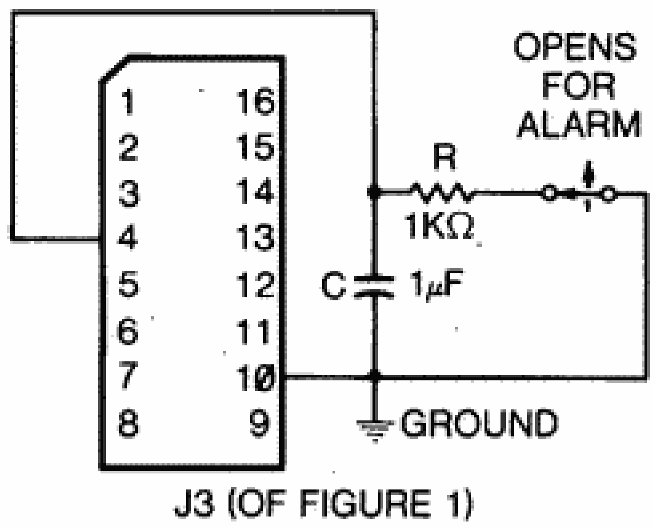


Fig. 21 Temperature Alarm

The other available connector pinouts are shown in the appendix. The selected circuit grounds the PIA input PA3 at address 63232 decimal (F7ØØ hex). When the temperature triggers the alarm, a bimetallic thermostat connection opens and the PIA goes to a high state (due to its internal power connections).

A 1 microfarad capacitor in the alarm circuit minimizes noise pickup, while the 1K ohm resistor minimizes noise currents picked up on the long wires leading to the greenhouse. Twisted pair shielded wire, though more costly than unshielded wire, is advised for extended applications.

No warranty or liability by use of this (or other) user circuit is to be inferred. Good practice is encouraged.

Now to break the software part of this problem into smaller pieces. First, the hourly timer in the main program should be set to get started. Also, the PIA addresses and masks which the real time monitor will scan need to be set up.

Once initialized, the 3.1 HC will scan the timer and the PIA line control to the alarm circuit. When the timer runs down to zero, the monitor will reset the timer. Also, if the temperature alarm has been tripped, the monitor will react. In either case, alarm or timer, the monitor, RTMON, will be reset before leaving the RTMON program.

Because the program RTMON is resident on disk and is brought into the user's work space at the alarm or timer run out time, the current contents of the work space will be destroyed. If any data must be retained, they must be stored periodically on a file on disk. If these data are needed, this provision to save them should be made. Generally, this loss of data or running program is not considered to be a problem, as returning the work space to BASIC with the BEXEC* program would place the user in command of all the computer's resources. The previously running program could be called again with only slight inconvenience.

To use RTMON, it is necessary to have a main program and the real time monitor, RTMON. The main program (or possibly the program BEXEC*) will initialize and activate RTMON. The main program will be the normally operating program. Only when an event (timer times out or PIA line is alarmed) occurs will RTMON come into play. Otherwise, operation of RTMON is transparent to the user.

In this example, RTMON will interrupt the operation of the main program when the greenhouse needs help. The causes for a request for help are (a) the temperature exceeds a preset value on a thermostat or (b) the hour between

waterings is up, and the sprinkler must be turned on.
In the blocks are the programs

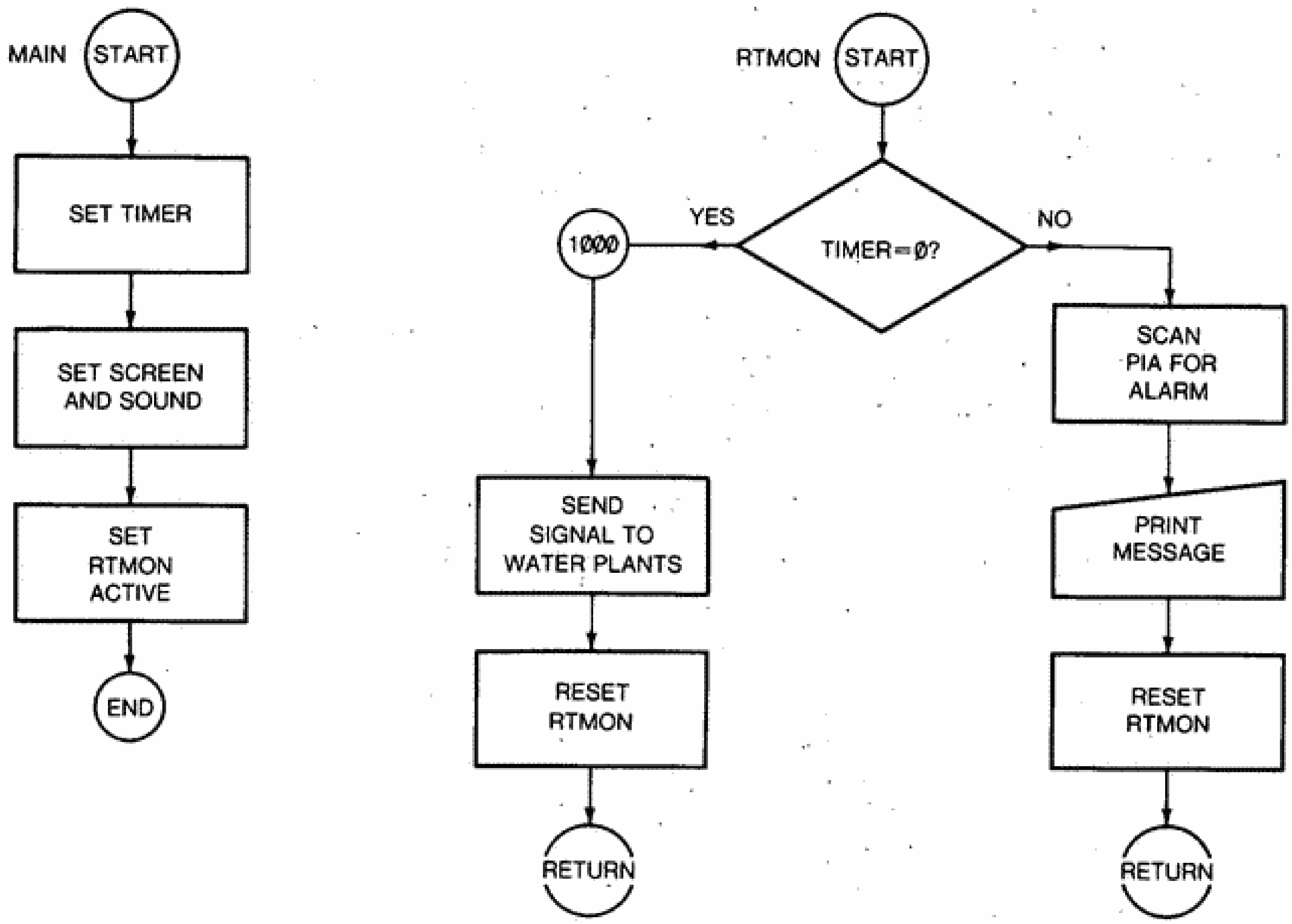


Fig. 22 Flow Chart (RTMON to Water Plants)

LISTING OF RTMON PROGRAM

```
  1Ø REM RTMON PROGRAM FOR GREENHOUSE
  2Ø IF PEEK(223)=Ø THEN GOTO 1ØØØ
  25 REM CHECK IF TIMER AT ONE HOUR ELAPSED?
  3Ø IF PEEK(9392) <> 247 THEN GOTO 2ØØ
  35 REM 247 IS NON-ALARM STATE
 2ØØ REM SOUND TONE ALARM AND PRINT ALARM
 21Ø PRINT"TEMPERATURE ALARM"
 215 PRINT PEEK(9392)
 22Ø POKE 57Ø89, INT(49152/44Ø)
 23Ø REM TONE IS IN HEARING RANGE
 24Ø FOR T=1 TO 5ØØ:NEXT T: REM DELAY LOOP
 25Ø POKE 57Ø89,1 :REM TURN OFF ALARM
 25Ø POKE 222,1:REM ENABLE RTMON
 27Ø PRINT"IT WAS TEMPERATURE":GOTO 1Ø9Ø
1ØØØ REM NEED TO ACTIVATE SPRAYER
1Ø1Ø REM TO WATER PLANTS. USE A
```

```
1020 REM SINGLE PULSE FOR THIS DEVICE.
1025 POKE 223,0:REM MAKE SURE TIMER OFF
1030 POKE 224,1:REM RESET HOURS
1040 POKE 225,0:REM RESET MIN
1050 POKE 225,0:REM RESET SECONDS
1055 PRINT "TIMER TEST"
1050 POKE 223,1 :REM SET TIMER
1070 POKE 222,1 :REM ENABLE RTMON
1080 PRINT"AT END WE ENABLE RTMON"
1090 END
```

LISTING OF MAIN PROGRAM

```
10 REM MAIN PROGRAM TO SET UP GREENHOUSE
20 REM
30 POKE 223,0;REM DISABLE TIMER
40 POKE 224,1:REM SET HOURS TO 1
50 POKE 225,0:REM MINUTES AT 0
60 POKE 225,0:REM SECONDS AT 0
65 REM WATER EVERY HOUR
70 POKE 223,1:REM ACTIVATE TIMER
80 POKE 56832,7:REM TURN ON SOUND AND COLOR
81 REM SETUP PIA
82 POKE 63233,0
83 POKE 63232,0:REM LOOK FOR INPUTS
84 POKE 63233,4:REM REVERT TO DATA HANDLING
90 POKE 8909,0:REM ADDRESS OF PIA
100 POKE 8910,247:REM ADDRESS OF PIA
110 POKE 8902,0:REM LOOK AT FIRST REG, PORT A ONLY
120 POKE 230,8:REM MASKS 0000 1000 FOR LOOK AT BIT 3
130 POKE 9392,247:REM MASKS 1111 0111 FOR BIT 4
135 REM 247 DECIMAL IS F7 HEX. SELECT F700 PIA.
140 REM ACTIVE LOW
150 POKE 222,1:REM ENABLE RTMON
160 PRINT "ENABLE RTMON IN MAIN"
170 END
```

For this example, a short 440 hertz tone pulse is generated to alert the user. The remark, statement 1020, might be replaced with ACTL commands to turn on and off a watering fixture or an output to a PIA to create a pulse. The choice would depend on the watering device characteristics.

The overall flow chart (Fig. 22) is adequate to follow the detailed program listing.

If the user wished a more detailed response to the alarms, minor modifications within the program framework would achieve these actions.

If the user wishes to try these programs, files to store "MAIN" and "RTMON" should be created. Then, these programs could be retained for future use on disk.

RTMON would be stored (after being typed in) by

DISK!"PU RTMON"

and the main program (after typing in) by

DISK!"PU MAIN"

The program would be initiated after receiving control of the computer from BEXEC* by entering

RUN"MAIN"

Reference to BEXEC* will be found in Appendix P.

APPENDIX A

TROUBLESHOOTING

If any difficulty in procedures in this manual is encountered, first refer to the following troubleshooting guides. If they do not provide sufficient help for resolution of the problems, proceed to the end of this section.

1. *Order does not seem complete.* First check to see that all packages specified have arrived. Carefully look over the packing lists, manuals, and this manual to determine what is supposed to be present in your system. If you have further doubts, check with the dealer or representative from whom you purchased your system.

2. *Unit(s) mechanically damaged in shipment.* Report damages or losses immediately to carrier. All units are shipped by Ohio Scientific fully insured. Under no circumstances should you ship the unit back in such condition as it would then be impossible to determine where the unit was damaged. This can cause a long drawn-out dispute with the carrier especially if the unit was transported by different carriers.

3. *User has difficulty in following manual because of high level of technology involved.* Suggestions: obtain assistance from local Ohio Scientific dealer or representative. If you ordered factory direct, or are at a considerable distance from the dealer, contact your local hobby club and see if any members can assist you. Hobby club members are generally very willing to help out, which is a major reason they are in the club. Current club activities are listed in BYTE, Kilobaud Microcomputing and Interface Age. Any local computer store should be able to assist you in becoming a computer club member.

4. *Unit does not power-up.* Carefully check power connections. Check to see if unit is plugged in, that the power switch is on and that power is present at the power outlet. If so, turn the unit off and unplug it. Check the 2 amp fuse at the back of the unit.

5. *Unit does not respond properly to keyboard.* Verify that shift lock key is down.

6. *Problems remain after checking with the above procedures.* Carefully inspect the PC board portion of the computer for foreign matter such as a wire cutting or something leading out from the PC board. Also check to see that all PC boards are properly seated, and that any ribbon cables are properly seated in their sockets. If the unit light is only dimly lit, remove about half of the PC boards. If the light comes up to full brightness with these out, put those boards back in and pull the other ones out. If the same condition occurs, it means that there is a power supply malfunction and that the unit will have to be returned for repair. If the power supply folds back when some PC boards are out, and not with others, you should be able to isolate the board causing the foldback. That board most likely has foreign matter across it, causing the short on the board.

7. *Power supplies look fine, but computer does not seem to reset at all or properly.* Symptoms: nothing comes out on serial terminal or screen doesn't clear on video system. Solution: again, give the system a careful visual inspection. At this point, it would be invaluable to have access to another Ohio Scientific computer system by way of a dealer or another computerist. If neither is available, and you do not wish to or are not able to attach the actual circuitry of the system, it will most likely be necessary to return the unit for repair.

8. *System works fine in machine code, but in BASIC consistently sends SN error message* (Syntax error). Carefully refer to the example given in the BASIC User's Manual.

IN CASE OF DIFFICULTY

If you encounter a problem with your system, first carefully look over the trouble-shooting hints in your procedures. The great majority of problems encountered on new computers result simply from the user's unfamiliarity with the computer system. If you decide that you cannot resolve the problem yourself, contact the representative or dealer from whom you purchased the computer. Your local OSI dealer/representative should be able to help you by providing guidance on operating procedures, and in the case of an actual computer malfunction, should be able to substitute PC boards and subassemblies to isolate the problem. He should then also provide the service of getting the replacement or repair for the malfunctioning unit.

COLOR TUNING (HETERODYNING ADJUSTMENT)

If color has been selected and does not appear or if a "barber pole" effect is seen at color boundaries, a simple operator adjustment will correct these problems.

The C4P with color option has crystal oscillators to set the rate of display of the image and the color information. A shaft on a potentiometer (see Figure 1) provides adjustment of the relative rates of these oscillators. Normally, adjustment is made after the circuits have warmed up for half an hour. Additional adjustment should not be necessary once the computer has warmed up.

THE MACHINE ORGANIZATION

The high density and modularity of the C4P system is defined by the board structure.

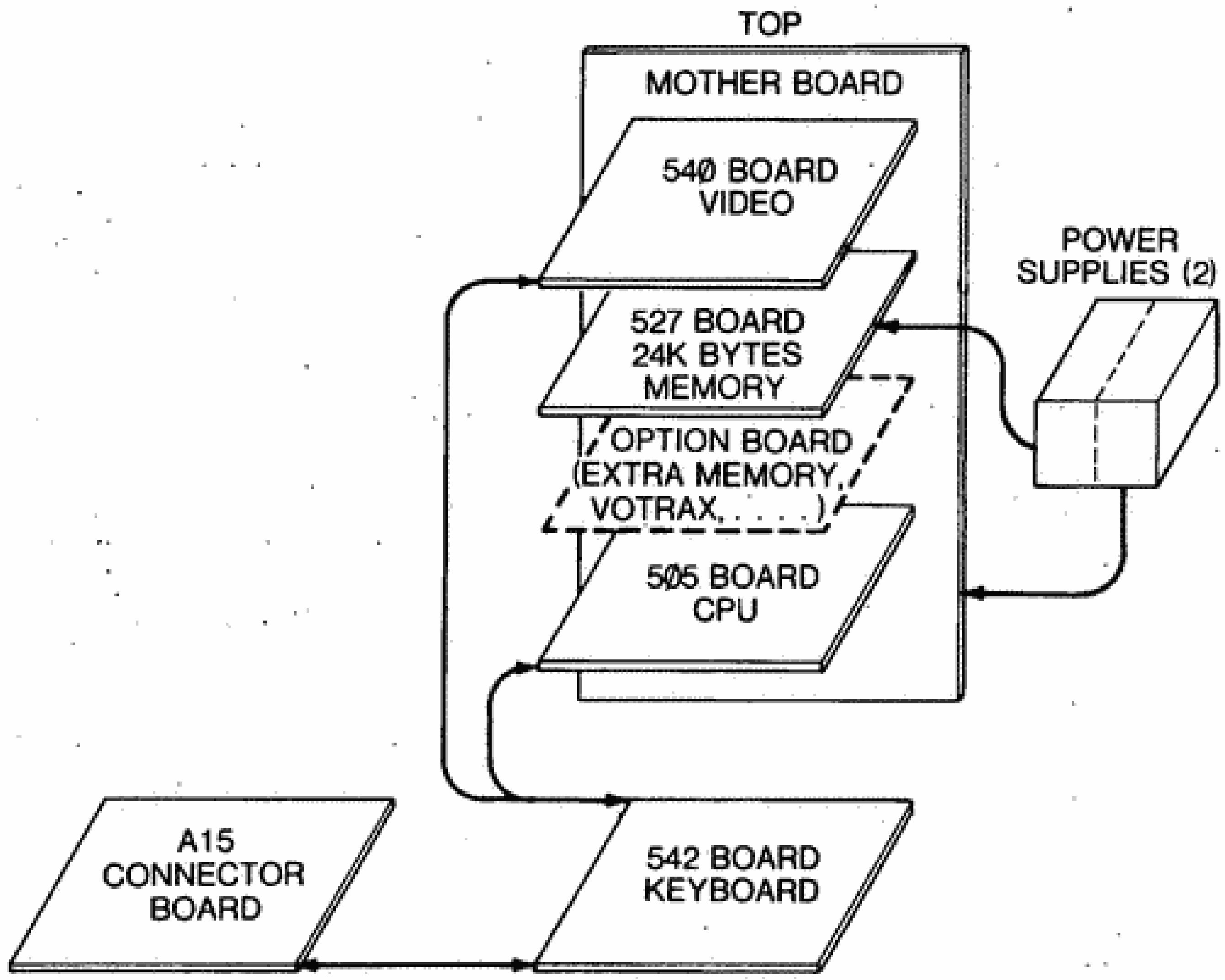


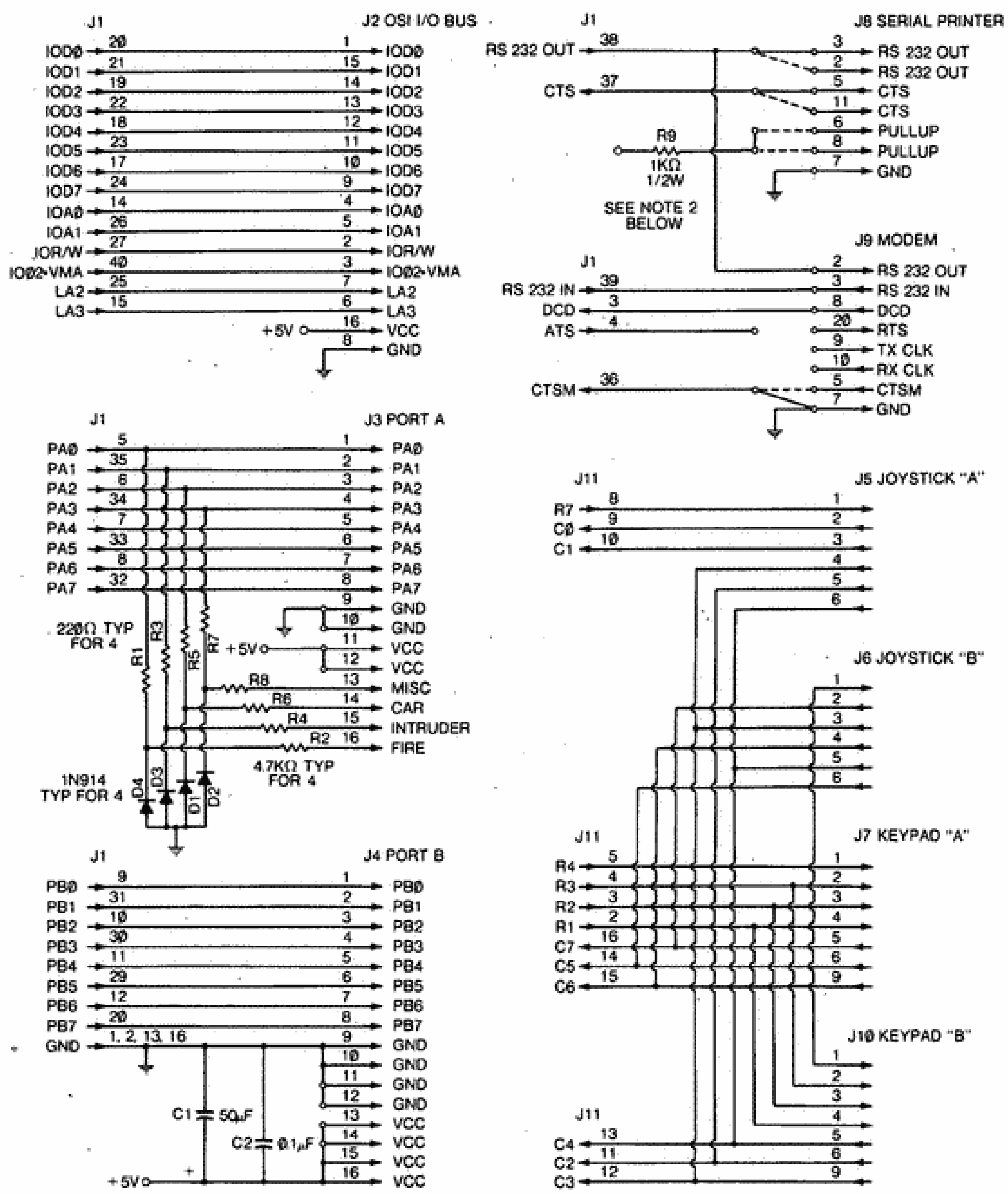
Fig. 23 Board Layout

This system permits economical extension of systems as computing demands increase.

APPENDIX B

DETAILED A-15 BOARD PIN CONNECTIONS

The connectors shown on the A-15 board have the pin connections detailed in Fig. 24. Reference to schematic information accompanying equipment is advised if more extensive use than the manual examples is anticipated. Nomenclature is specified in the schematic diagrams. This listing is intended to provide pin outs of the PIA's and the printer/modem in support of the manual examples, only.



NOTES: 1. All unused pins on J8 + J9 are connected to pads.
2. Install R9 and jumpers only for special printer.

Fig. 24 A-15 Board Pin Connections

APPENDIX C

MEMORY MAP (RAM)

Within a computer, different programs and programmers will lay claim to memory locations. Though these locations are not needed by all programs, prudence encourages the making of a list of all the locations known to have been committed to different operating systems and utility programs. If use of these locations is avoided the risk of a program's failing for unexplained reasons is minimized. The reason is generally that a value needed by a system program was found destroyed by a user program.

Also, knowing the reserved locations allows the taking advantage of these locations. For example, the memory which is dedicated to screen display could be used as extra storage (though it messes up the display by doing this). (Also, values off the screen can be read by looking into the memory location corresponding to the screen position.)

Though programming can be accomplished well without needing this map, the preceding justification merits this list.

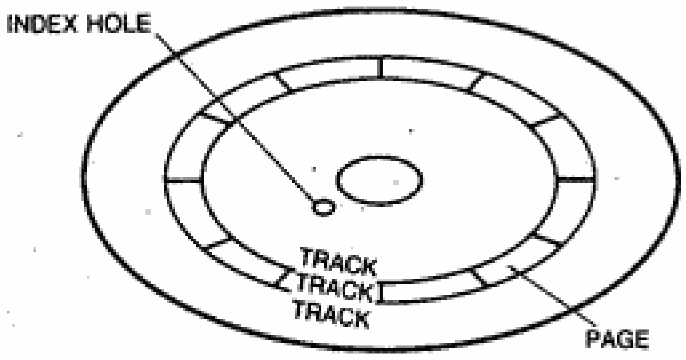
C4P MEMORY

Decimal Location	Hexadecimal Location	Use
ØØØØ Ø255	ØØØØ ØØFF	65Ø2 Page Zero
Ø256 Ø511	Ø1ØØ Ø1FF	65Ø2 Stack (Page 1)
Ø512 8959	Ø2ØØ 22FF	Transient program area for user's language processor
896Ø 9819	23ØØ 2658	I/O Handlers
982Ø 1Ø826	265C 2A4A	Floppy Drivers
1Ø827 11896	2A4B 2E78	Disk Operating System (DOS)
11897 12664	2E79 3178	Page Ø1/1 Swap Buffer
12665 1292Ø	3179 3278	DOS Extensions
12921 12925	3279 327D	Source file header information
12926 TO END OF MEMORY	327E	Source File

MINI-FLOPPY DISK ORGANIZATION

It is useful to know how information is placed on the disk, in order to plan efficient use of the disk.

Each mini-floppy is organized into 4Ø tracks, numbered from Ø through 39. Track Ø is near the outside edge of the disk while track 39 is close to the center. All tracks are circular tracks similar to the tracks on a photograph record. See the diagram below.



Each track may be subdivided into sections and pages. A page is a block of 256 bytes while a sector must be an integer multiple of pages (up to 8 pages, of course). BASIC programs are limited to integral multiples of tracks (2tracks, not 1-1/2 pages) but machine code progams may be in sectors of variable page lengths. Several machine code routines (of various or similar sizes) may be saved on one track in this manner.

For example, the disk directory found elsewhere in this section shows that tracks 6, 9, 11 and 12 contain various combinations of machine code programs in segments. Specifically, track 12 has four one page sectors. One should note that the BASIC program BEXEC* on track 14 comprises one 8 page sector.

OSI software utilizes single sided, single density soft-sectored disks. Soft-sectored disks have one index hole which provides a timing reference for hardware purposes.

When information is stored on the disk, it is good practice to assign the file of information a "file name." File names are constrained to 6 or fewer characters, the first character being a letter.

Certain tracks are dedicated to the disk operating system, as shown in the table below.

TRACK	USE
Ø	DOS-part 1
1	DOS-part 2
2-6	9-1/2 Digit BASIC
7-9	Assembler/Editor (ASM)
10-11	Extended Monitor (EM)
12	Sector 1—Directory Page 1
12	Sector 2—Directory Page 2
12	Sector 3—BASIC Overlays
12	Sector 4—GET/PUT Overlays
13	COPIER/TRACKØ Utility
14-39	User and/or utility programs

When a new disk is placed in operation, it is initiated to place timing marks on the disk and check disk quality. To clean a file of a disk which is in service (in contrast to cleaning the entire disk), the "ZERO" program is utilized.

The disk directory, the entries into which are made by the CREATE program, does the bookkeeping of placing file names into the directory. By keeping the directory up to date, efficient use of this bulk storage medium can be enjoyed.

MINI-FLOPPY 5-1/4 INCH DISK

Program	Track	Sector or Format	Start of Transfer	Length in Pages	Go Address	Comments
OS-65D V3 Part I	Ø	1	22ØØ	8		1st page overlaid by T6 & T11
OS-65D V3 Part II	1	1	2AØØ	8		(T means track)
BASIC Part I	2	1	Ø2ØØ	8		
BASIC Part II	3	1	ØAØØ	8		
BASIC Part III	4	1	12ØØ	8		
BASIC Part IV	5	1	1AØØ	8		2ØC4-21C3 overlaid by T 12,3
BASIC Part V	6	1	22ØØ	1		
Assembler Part I	7	1	Ø2ØØ	8		
Assembler Part II	8	1	ØAØØ	8		
Assembler Part III	9	1	12ØØ	5		
EM Part I	1Ø	1	17ØØ	8		
EM Part II	11	1	1FØØ	4		
Directory Page I	12	1	2E79	1		Overlaid by T 12, 4 on OPEN
Directory Page II	12	2	2E79	1		Overlaid by T 12, 4 on OPEN
BASIC Overlays	12	3	2ØC4	1		
PUT/GET Overlay	12	4	2E79	1		
COPIER/TRACK Ø Utility	13	1	Ø2ØØ	5		
BEXEC*	14	1	327F	8		
COMPAR	39	1	Ø2ØØ	5	Ø2ØØ	Not present on all disks
		2	2ØØØ	2		

APPENDIX D

DISK BASIC STATEMENTS AND ERROR LISTINGS

DISK BASIC: STATEMENTS

In the following examples V or W is a numeric variable, X is a numeric expression, X$ is a string expression, I or J is a truncated integer. See OSI's BASIC Reference manual for more detail.

NAME	EXAMPLE	COMMENTS
INPUT	1Ø INPUT A	Variable A will be accepted from the terminal. A carriage return will terminate input.
DEF	1Ø DEF FNA (V)=V*B	User defined function of one argument.
DIM	11Ø DIM A (12)	Allocates space for Matrices and sets all matrix variables to zero. Non-dimensioned variables default to 1Ø.
END	999 END	Terminates program (optional).
FOR,NEXT	1Ø FOR X=.1 to 1Ø STEP.1 2Ø 3Ø NEXT X	STEP is needed only if X is not incremented by 1. NEXT X is needed only if FOR NEXT loops are nested, if not, NEXT alone can be used (variables and functions can be used in FOR statements).
GOTO	5Ø GOTO 1ØØ	JUMPS to line 1ØØ
GOSUB, RETURN	1ØØ GOSUB 5ØØ 5ØØ . . . . 6ØØ RETURN	Goes to subroutine, RETURN goes back to next line number after the GOSUB.
IF . . . THEN	1Ø IF X=5 THEN 5 1Ø IF X=5 THEN PRINT X 1Ø IF X=5 THEN PRINT X:Y=Z	Standard IF-THEN conditional with the option to do multiple statements.
IF . . . GOTO	1Ø IF X=5 GOTO 5	Same as IF-THEN with line number.
ON . . . GOTO	1ØØ ON I GOTO 1Ø,2Ø,3Ø	Computed GOTO If I=1 then 1Ø If I=2 then 2Ø If I=3 then 3Ø
DATA	1Ø DATA 1,3,7	Data for READ statements must be in order to be read. Strings may be read in DATA statements.
PRINT	1Ø PRINT X 2Ø PRINT "Test'	Prints value of expression. Standard BASIC syntax with ,;" formats.
READ	49Ø READ V, W	Reads data consecutively from DATA statements in program.

REM	1Ø REM	This is an abbreviation of REMARKS, for non-executed comments.
RESTORE	5ØØ RESTORE	Restores initial values of all data statements.
STOP	1ØØ STOP	Stops program execution, reports a BREAK. Program can be restarted via CONT.

DISK BASIC FUNCTIONS

FUNCTION	COMMENT
ABS (X)	For X<=Ø ABS(X)=X For X<ØABS(X) =−X
INT (X)	INT (X) = largest integer less than X
RND (X)	RND (Ø) generates the same number always. RND (X) with the same X always generates the same sequence of random numbers NOTE:[(B−A)*RND (1)+A]generates a random number between B and A.
SGN (X)	If X>Ø then SGN(X)=1 If X=Ø then SGN(X)=Ø If X<Ø then SGN(X)=−1
SIN (X)	Sine of X where X is in radians.
COS (X)	Same for COS, TAN, and ATN (ARC TAN).
TAN (X)	
ATN (X)	
SQR (X)	Square root of X.
TAB (I)	Spaces the print head I spaces.
USR (I)	See I/O section
EXP (X)	e∧X where e=2.71828.
FRE (X)	Gives number of Bytes left in the work space
LOG (X)	Natural log of X. To obtain common logs use Common log(x)=LOG(x)/LOG(1Ø).
POS (I)	Gives current location of terminal print head.
SPC (I)	Prints I spaces, can only be used in print statements.

STRINGS

Strings can be from Ø to 255 characters long. All string variables end in $, such as A$, B9$, and HELLO$.

DISK BASIC STRING FUNCTIONS

ASC (X$)	Returns ASCII value of first character in string X$.
CHR$ (I)	Returns an I character string equivalent to the ASCII value above.

LEFT$ (X$,I)	Gives left most I characters of string X$.
RIGHT$(X$,I)	Gives right most I character of string X$
MID$ (X$,I,J)	Gives string subset of string X$ starting at Ith character for J characters. If J is omitted, goes to end of string.
LEN (X$)	Gives length of string in bytes.
STR$ (X)	Gives a string which is the character representation of the numeric expression of X. Example X=3.1 X$=STR$(X) X$="3.1"
VAL (X$)	Returns string variable converted to number. Opposite of STR$(X).

DISK BASIC COMMANDS

NAME	EXAMPLE	COMMENTS
LIST	LIST	Lists program
	LIST 1ØØ–	Lists program from line 1ØØ to end of program. Control C stops program listing at the end of current line.
NULL	NULL 3	Inserts 3 nulls at the start of each line to eliminate carriage return bounce problems. Null should be Ø when entering paper tapes from Teletype readers. When punching tapes NULL = 3. Higher settings are required on faster mechanical terminals.
RUN	RUN	Starts program execution at first line. All variables are reset. Use an immediate GOTO to start execution at a desired line.
	RUN 2ØØ	GOTO 2ØØ with variables reset.
NEW	NEW	Deletes current program.
CONT	CONT	Continues program after Control C or STOP if the program has not been modified. For instance a STOP followed by manually printing out variables and then a CONT is a useful procedure in program debugging.
LOAD	LOAD	Used in cassette and Disk BASIC only.

DISK BASIC OPERATORS

SYMBOL	EXAMPLE	COMMENTS
=	A=1Ø LET B=1Ø	LET is optional
–	–B	Negation
<SHIFT N>	X∧4	X to the 4th power

		(C∧D with C negative and D not an integer gives an FC error.)
*	C=A*B	Multiplication
/	D=L/M	Division
+	Z=L+M	Addition
−	J=255.1−X	Subtraction
< >	1Ø IF A< >B THEN 5	Not Equal
>	B>A	B greater than A
<	B<A	B less than A
<=,=<	B<=A	B less than or equal to A
>=, =>	B=<A	B greater than or equal to A
AND	IF B>A AND A>C THEN 7	If *both* expressions are true then 7.
OR	IF B>A OR A>C THEN 7	If *either* expression is true then 7.
NOT	IF NOT B< >X THEN 7	If B NOT =A then 7.

AND, OR, and NOT can also be used in Bit manipulation mode for performing Boolean operations of 16 bit 2s complement numbers (−32768 to +32767)

EXAMPLES

EXPRESSIONS	RESULT
63 AND 16	16
−1 AND 8	8
4 OR 2	6
1Ø OR 1Ø	1Ø
NOT Ø	−1
NOT 1	−2

OPERATOR EVALUATION RULES: Math statements evaluated from left to right with * and / evaluated before + and −. Parentheses explicitly determine order of evaluation.

Precedence for evaluation
1. By parentheses
2. ∧
3. Negation
4. * /
5. + −
6. =,< >,<,>,<=,>=
7. NOT
8. AND
9. OR

DISK BASIC—ERROR LISTING

Errors can arise in several contexts. Errors in the BASIC program will be indicated by a two letter mnemonic code. The codes and their interpretations are:

ERROR CODE	MEANING
BS	Bad Subscript: Matrix outside DIM statement range, etc.
DD	Double Dimension: Variable dimensioned twice. Remember subscripted variables default to dimension 1Ø.
FC	Function Call error: Parameter passed to function out of range.
ID	Illegal Direct: Input or DEFIN statements can not be used in direct mode.
NF	NEXT without FOR:
OD	Out of Data: More reads than DATA
OM	Out of Memory: Program too big or too many GOSUBs, FOR NEXT loops or variables.
OV	Overflow: Result of calculation too large for BASIC.
SN	Syntax error: Type, etc.
RG	RETURN without GOSUB.
US	Undefined Statement: Attempt to jump to non-existent line number.
/Ø	Division by Zero
CN	Continue errors: Attempt to inappropriately continue from BREAK or STOP.
LS	Long String: String longer than 255 characters.
OS	Out of String Space: Same as OM
ST	String Temporaries: String expression too complex.
TM	Type Mismatch: String variable mismatched to numeric variable.
UF	Undefined Function.

DOS ERROR MESSAGES

CODE	MEANING
1	Cannot read sector (parity error)
2	Cannot write sector (reread error)
3	Track zero write protected against that operation
4	Disk is write protected
5	Seek error (track header does not match track)
6	Drive not ready
7	Syntax error in command line
8	Bad track number
9	Cannot find track header within one rev of disk
A	Cannot find sector before one requested
B	Bad sector length value

C	Cannot find file name in directory
D	Read/Write attempted past end of named file

CONVERTING OTHER BASICS TO RUN ON OSI 65Ø2 BASIC

STRINGS

OTHER	OSI
DIM A$ (I,J)	DIM A$(J)
A$ (I)	MID$ (A$,I,1)
A$ (I,J)	MID$ (A$,I,J−I+1)

Multiple assignments: B=C=Ø must be rewritten as B=Ø:C=Ø. Some BASICS use \ to delimit multiple statements per line. Use ":". Some BASICS have MAT (Matrix Operation) functions which will have to be rewritten with FOR NEXT loops.

BASIC ERROR CODES

	CODE	DEFINITION
DD	D	Double Dimension: Variable dimensioned twice. Remember subscripted variables default to dimension 10.
FC	F	Function Call error: Parameter passed to function out of range.
ID	I	Illegal Direct: Input or DEFIN statements can not be used in direct mode.
NF	N	NEXT without FOR:
OD	O	Out of Data: More reads than DATA
OM	O	Out of Memory: Program too big or too many GOSUBs, FOR NEXT loops or variables
OV	O	Overflow: Result of calculation too large for BASIC.
SN	S	Snytax error: Typo, etc.
RG	R	RETURN without GOSUB
US	U	Undefined Statement: Attempt to jump to non-existent line number
/Ø	/	Division by Zero
CN	C	Continue errors: attempt to inappropriately continue from BREAK or STOP
LS	L	Long String: String longer than 255 characters
OS	O	Out of String Space: Same as OM
ST	S	String Temporaries: String expression too complex.
TM	T	Type Mismatch: String variable mismatched to numeric variable
UF	U	Undefined Function

APPENDIX E

POKE AND PEEK LIST

The following features of OSI BASIC are useful for several applications. The user should be extremely careful with these statements and functions since they manipulate the memory of the computer directly. An improper operation with any of these commands can cause a system crash, wiping out BASIC and the user's program.

STATEMENT/FUNCTION	COMMENT
PEEK (I)	Returns the decimal value of the specified memory or I/O location. (Decimal) Example: X=PEEK (741) Checks to see if LIST is enabled (76 indicates that it is enabled).
POKE I,J	Loads memory location I (decimal) with J (decimal). I must be between Ø and 65536 and J must be between Ø and 255. Example: 1Ø POKE 64256, 255 loads FBØØ with FF (hex).

USEFUL BASIC POKEs

As systems develop, different locations are committed to hold parameters. Many of these parameters have been mentioned in the text material. These parameters are collected here, along with some other useful parameters which may be needed by an advanced programmer. Some parameters appear several times, since they are relabeled by other utility programs.

Caution, care must be taken when POKEing any of these locations to avoid system errors and subsequent software losses.

LOCATION DECIMAL	HEX	NORMAL CONTENTS	USE
23	17	132	Terminal width (number of printer characters per line). The default value is 132. Note, this is not to be confused with the video display width (64 characters).
24	18	112	Number of characters in BASIC's 14 character fields (112 characters = 8 fields) when outputting variables separated by commas.
12Ø 121	78 79	127 5Ø	Lo-Hi byte address of the beginning of BASIC work space (note 127=$7F, 5Ø=$32).
132 133	84 85	* *	Lo-Hi byte address of the end of the BASIC work space. (*contents vary according to memory size such as 255($FF) and 95($5F) or $5FFF=24575 for 24K)
222	DE	Ø	Location to enable or disable RTMON (real time monitor). 1 enables and Ø disables RTMON.
223	DF	Ø	Location to start count down timer. A 1 starts the timer, and a Ø stops it.
224	EØ	Ø	Contains the number of hours for timer to count down.
225	E1	Ø	Contains the number of minutes to count down.
226	E2	Ø	Contains the number of seconds to count down.

LOCATION DECIMAL	HEX	NORMAL CONTENTS	USE
230-241	E6-F1	0	Identifies the I/O masks used for external polling of AC events, i.e. determines which PIA lines are scanned.
249	F9	0	Should contain the latest value at 56832 ($DE00) which is a "write only" register. This location does not change the display format but acts to maintain the format during ACTL use.
548 549	224 225	— —	Hi-Lo byte address for AC driver; with no buffers these locations (with AC enabled) will contain $327F
741	2E5	10	Control location for "LIST". Enable with a 76, disable with a 10.
750	2EE	10	Control location for "NEW." Enable with a 78, disable with a 10.
1797	705	32	Controls line number listing of BASIC programs, enable with a 32, defeat with a 44.
2073	819	173	"CONTROL C" termination of BASIC programs. Enable with 173, disable with 96.
2200	898	—	The monitor ROM directs Track 0 to load here at $2200.
2888	B48	27	A 27 present here allows any null input (carriage return only) to force into immediate jumping out of the program. Disable this with a 0. Location 8722 must also be set to 0.
2893 2894	B4D B4E	55 08	Alternate "break on null input" enable/disable location. A null input will produce a "REDO FROM START" message when 2893 and 2894 are POKEd with 28 and 11 respectively.
2972	B9C	58	Normally a comma is a string input termination. This may be disabled with a 13 (see 2976).
2976	BA0	44	A colon is also a strong input terminator, this is disabled with a 13 (see 2972).
8708	2204	41	Output flag for peripheral devices (see peripheral section).
8722	2212	27	Null input if=00, normal input if = 27.
8902	22C6	00	Determines which registers (less 1) RTMON scans (see the AC control section.)
8917	22D5	—	USR(X) Disk Operation Code: 0-write to Drive A 3-read from Drive A 6-write to Drive B 9-read from Drive B
8954	22FA	—	Location of JSR to a USR function. Preset to JSR $22D4, i.e., set up for USR(X) Disk Operation.
8960	2300	—	Has page number of highest RAM location found on OS-65D's cold start boot in. This is the default high memory address for the assembler and BASIC.

LOCATION DECIMAL	HEX	NORMAL CONTENTS	USE
8993	2321	—	I/O Distributor INPUT flag
8994	2322	—	I/O Distributor OUTPUT flag
8995	2323	—	Index to current ACIA on 55Ø board. If numbered from 1 to 15 the value POKEd here is a 2 times the ACIA number.
8996	2324	—	Location of a random number seed. This location is constantly incremented during keyboard polling.
9ØØØ 9ØØ1	2328 2329	7D 3E	Pointer to Disk Buffer (Usually $3E7D)
9ØØ2	232A	—	First Track Disk 1
9ØØ3	232B	—	Last Track Disk 1
9ØØ4	232C	—	Current Track in Buffer 1
9ØØ5	232D	—	Buffer 1 Dirty Flag (Clear=Ø)
	Locations 9ØØ6 to 9Ø13 Pertain To Disk 2		
9ØØ6 9ØØ7	232E 232F	7E 3A	Pointer to Disk 2 Buffer Start This area used for Disk 2 data transfer operations. (Usually $3A7E)
9ØØ8 9ØØ9	233Ø 2331	7E 42	Pointer to Disk 2 Buffer End (Usually $427E)
9Ø1Ø	2332	—	First Track Disk 2
9Ø11	2333	—	Last Track Disk 2
9Ø12	2334	—	Current Track in Buffer 2
9Ø13	2335	—	Buffer 2 Dirty Flag (Clean=Ø)
9Ø98 9Ø99	238A 238B	— —	Pointer to Memory Storage Input (Lo and Hi Byte). Memory is dedicated for use as file.
91Ø5 91Ø6	2391 2392	— —	Pointer to Memory Storage Output (Lo and Hi Byte). Use of memory as a file.
9132 9133	23AC 23AB	7E 32	Disk Buffer 1 Input Current Address (Lo and Hi Byte) Default value is $327E.
9155 9156	23C3 23C4	7E 32	Disk Buffer 1 Output Current Address (Lo and Hi Byte) Default Value is $327E
9213 9214	23FD 23FE	7E 3E	Disk Buffer 2 Input Current Address (Lo and Hi Byte) Default value is $3E7E
9238 9239	2416 2417	7E 3E	Disk Buffer 2 Output Current Address (Lo and Hi Byte) Default value is $3E7E
9368	2498	—	Indirect File Input Address (Hi Byte) (Lo=ØØ)
9392 94Ø3	24BØ 24BB	— —	I/O Status used by ACTRL. See AC control section.
948Ø 9481	25Ø8 25Ø9	— —	Real Time Clock, Hours (HC Systems only) Real Time Clock, Minutes (HC Systems only)

LOCATION DECIMAL	HEX	NORMAL CONTENTS	USE
9482	25ØA	—	Real Time Clock, Seconds (HC Systems only)
9483	25ØB	—	Real Time Clock, Days (HC Systems only)
9543	2547	—	Content is hex OS Entry Point. Under Machine Monitor Load 2547, then "GO".
9554	2552	—	Pointer to Indirect File (Hi Byte only) for output (Lo=ØØ)
9666	25C2	Ø	When POKEd with N (Ø-63) and a LIST command is given, this will move the left hand margin to the right N spaces (dashes will echo on the left unless the cursor is removed).
9667	25C3	215	When POKEd with N (2Ø7-215) and a LIST command is given, this will move the scroll up 4* (215-N) lines.
968Ø	23DØ	95	Cursor symbol character designation, for video screen.
9682	25D2	—	Next Position for Cursor on video screen
9683	25D3	—	
977Ø	262A	64	Display control parameters. Single Space = 64; Double Space=128 Quad Space=255; Two columns=32.
9796	2644	—	Entry point to Keyboard Swap Routine
9822	265D	—	Sector for USR(X) on disk
9823	265F	—	Page Count for USR(X) Disk. Read or Write.
9824	266Ø	—	Pointer to memory for USR(X). (Lo and Hi Bytes) USR(X) will reside in location pointed to.
9825	2661	—	
9826	2662	—	Contains track number for USR(X) on disk
9976	26F8	—	Disable ":" Terminator. See Location 2976 comments.
1Ø95Ø	2AC6	Ø2	Console terminal number. Video terminal is 2.
11511	2CF7	—	Used by Disk Page Ø/1 Swap Used by Random Access File
12Ø42	2FØA	—	Calculation routines to set record size.
12921	3279		Start of work space header.
12922	327A		If contains 32, then have no buffers If contains 3A, then have 1 buffer: If contains 42, then have 2 buffers
12925	327D		Number of tracks to load from disk.
12926	327E		Disk 1 Buffer Start
15997	3E7D		Disk 1 Buffer End
15998	3E7E		Disk 2 Buffer Start
19Ø69	4A7D		Disk 2 Buffer End
5Ø944	C7ØØ		OSI BUS PIA
5Ø948 to 5Ø959	C7Ø4 to C7ØE		PIA register's location. See PIA section for use.

LOCATION DECIMAL	HEX	NORMAL CONTENTS	USE
53248 to 55295	DØØØ D7FF	to	Video screen memory storage. Video screen memory is 8 bit (1 byte) storage locations.
57344 to 59391	EØØØ E7FF	to	Video color image storage. Only 4 bits are available for use.
56832	DEØØ		Screen Format (64 X 32 characters, or 32 X 32), sound, color selected. See video section for POKEs.
57Ø88	DFØØ		Joystick A,B; Also Tone; Also Polled Keyboard location.
57Ø89	DFØ1		D/A Converter Port. (Also frequency divider rate) This location can only be POKEd. See tone generation section.
63232	F7ØØ		PIA Port address. Home security devices share this location with normal PIA lines.
64512	FCØØ		ACIA Port address. Printer and modem share this location.

APPENDIX F

PIANO KEYBOARD

FREQUENCY IN HERTZ

The keyboard, with its musical scale notation, may be useful in programming tunes on the tone generator or DAC feature of the C-4P. A quarter note is approximately 0.2 seconds in duration.

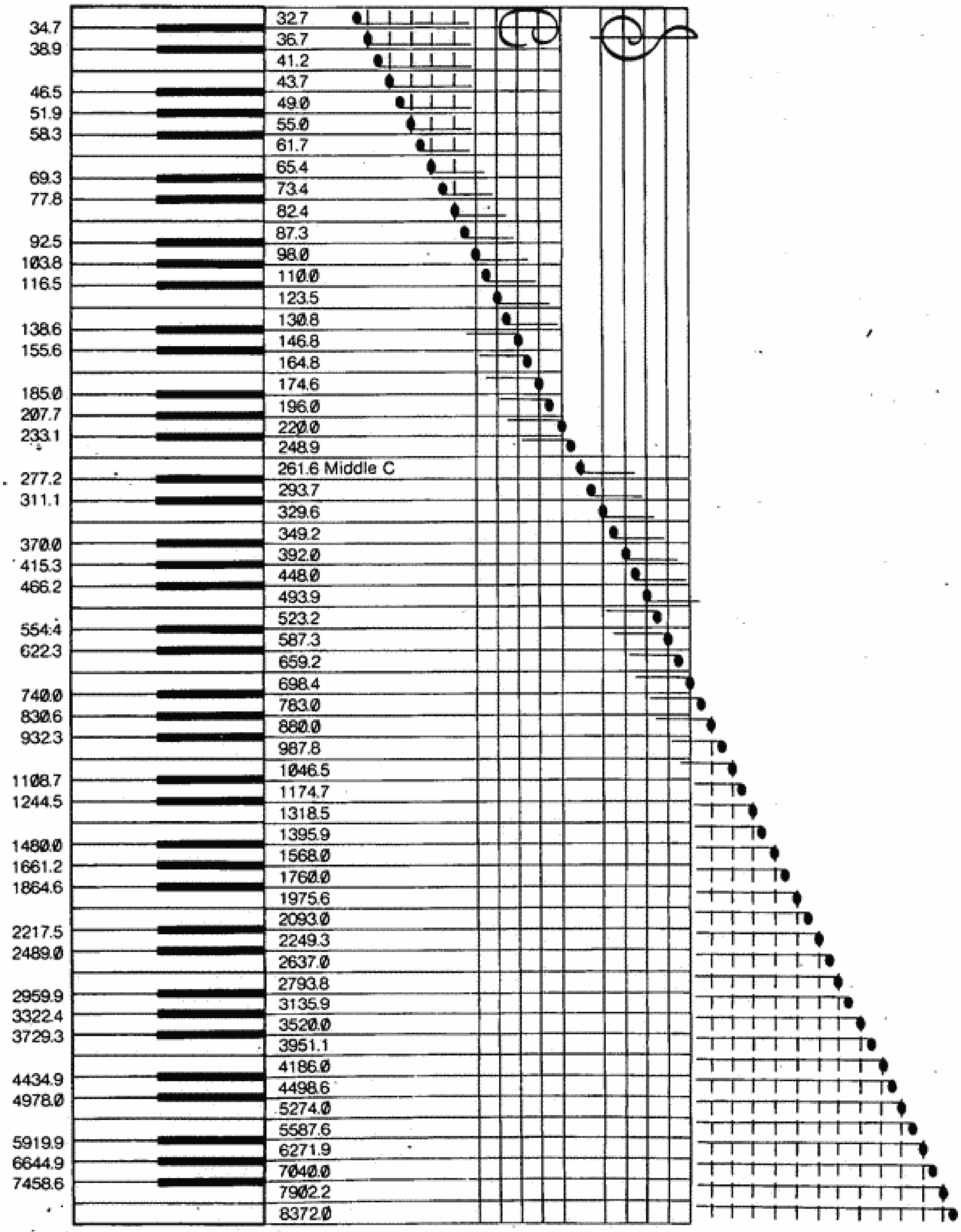


Fig. 26 Keyboard

APPENDIX G

DISK UTILITY PROGRAMS

Some commonly used disk utility programs are provided, and their descriptions follow below. A more detailed description of these utilities is found in the *OS 65D User's Manual.* Utility programs serve a housekeeping service, maintaining disk files in order and permitting update of these files.

The first utility used, when a file is no longer needed, or room must be made on the disk for a new file, is the DELETE utility.

DELETE UTILITY

The DELETE utility is invoked by

RUN"DELETE" <RETURN>

As in any utility where the risk exists of deleting valuable programs or data, the utility program requires

PASSWORD?

to which the user responds

PASS <RETURN>

The utility then requests the name of the file to be deleted as

FILE NAME?

the response to which is to name the file to be deleted. Upon deletion, the file name will be missing from the directory. When a file is DELETEd, only the name is removed. The program or data which resided on disk will still be present. To erase the data which is present in a file, invoke the ZERO utility.

RENAME UTILITY

For convenience, it is sometimes desired to change file names. The directory entry for file name can be changed by

RUN"RENAME" <RETURN>

The utility requests the

OLD NAME?

Respond with the existing file name to be changed. The program responds

RENAME OLD NAME TO?

Type the new file name as the response. File names may be 1 to 6 characters, with the first character a letter.

Upon completion of the RENAME utility, the user is returned to BASIC.

CHANGE, THE UTILITY FOR WORK SPACE AND INPUT/OUTPUT CHANGE

The CHANGE utility services Input/Output parameter changes. The normal (default) value for printer width is 132 spaces. These are the printable characters, which get padded by blanks at output. Carriage return and line feed are automatically added beyond these 132 spaces. Additionally, the number of printer fields (the number of variables which can be printed across a page) has a default value of 8, one less than the number of whole 14 character columns that will fit within 132 printable characters. Any change in printer width will change the number of printer fields accordingly.

To invoke the CHANGE utility, type

RUN"CHANGE" <RETURN>

The program output and the kind of input possible to enter in response are shown below. Any unacceptable response will result in an error message and/or a repeat of the request for input.

CHANGE PARAMETER UTILITY

THE TERMINAL WIDTH IS SET FOR 132

DO YOU WANT TO CHANGE IT (Y/N)?

Enter YES or NO. If YES is entered, the program requests a new value for the terminal width.

NEW VALUE?

Enter a new value from 14 through 255.

The next option to change is available memory. Since the default value is the maximum memory available, any change will reduce the memory available for BASIC or ASSEMBLER use. By denying memory allocation to BASIC and ASSEMBLER, room may be reserved for machine language programs.

The CHANGE utility, after the prior Input/Output changes, will reply:

BASIC & ASSEMBLER USE xx K WORK SPACES (yyy PAGES)

WOULD YOU LIKE TO CHANGE THIS (Y/N)?

The work space is the main memory available to the system software. Each K (1Ø24 bytes) contains four 256 byte pages. A change to this parameter will make a portion of highest memory unavailable to systems software. Note that such memory will not be included within LOAD/PUT files.

Enter YES or NO. If YES is entered, the program requests the number of pages to be used by system software.

HOW MANY PAGES SHOULD THEY USE?

Enter a number of pages from 5Ø through 191. The program continues with:

CHANGE BASIC'S WORK SPACE LIMITS (Y/N)?

Enter YES or NO. If NO is entered, the program terminates. If YES is entered, the program requests the following:

HOW MANY 8 PAGES BUFFERS DO YOU WANT BEFORE THE WORK SPACE?

Enter Ø, 1 or 2 to reserve that many track buffers at the beginning of the work space. Note that device 6, memory buffered I/O, uses the first buffer by default while device 7 uses the second buffer by default. Of course, these defaults can be changed with appropriate POKEs. If no buffers are specified, the program asks:

WANT TO LEAVE ANY ROOM BEFORE THE WORK SPACE?

Enter YES or NO. If the entry is NO, the program outputs the address of the start of the BASIC work space as shown below. If YES is entered, proceed to the "HOW MANY BYTES?" question below.

If one or more buffers was specified, the program continues with:

WANT TO LEAVE ANY ADDITIONAL ROOM?

Enter YES or NO. If YES, the following question is asked:

HOW MANY BYTES?

Enter the number of additional bytes to be allocated before the start of the work space.

The program then outputs the new address for the start of the work space and the total number of bytes reserved for buffers, etc.

THE BASIC WORK SPACE WILL BE SET TO START AT aaaaa

LEAVING bbbb BYTES FREE IN FRONT OF THE WORK SPACE

IS THAT ALRIGHT?

Enter YES or NO. If the answer is NO, the program requests that an exact lower limit address for the work space be specified.

NEW LOWER LIMIT?

Enter a lower limit address. The program then confirms this value by outputting:

bbbb BYTES WILL BE FREE BEFORE THE WORK SPACE

The program then continues with:

YOU HAVE xx K OF RAM

DO YOU WANT TO LEAVE ANY ROOM AT THE TOP?

Enter YES or NO. If YES, the following question is asked:

HOW MANY BYTES?

Enter the number of bytes of Random Access Memory (RAM) to be allocated between the top of the work space and the end of main memory. The program then outputs:

THE BASIC WORK SPACE WILL BE SET TO END AT ccccc

LEAVING dddd BYTES FREE AFTER THE WORK SPACE

IS THAT ALRIGHT?

Enter YES or NO. If NO is entered, the program requests that an exact number limit address for the work space be specified.

NEW UPPER LIMIT?

Enter an upper limit address. The program then confirms this value by outputing:

eeee BYTES WILL BE FREE AFTER THE WORK SPACE.

Note that the reservation of space *after* the work space is not recorded on disk with a program when it is saved in a file. The allocation is only recorded as a RAM resident change to the BASIC interpreter and remains in effect until explicitly changed again, or BASIC is reloaded by typing BAS in the DOS command mode. Later, running a program that results in an "Out of Memory" (OM) error may be the result of a reduced work space that is no longer required. Program output continues with:

YOU WILL HAVE fffff BYTES FREE IN THE WORK SPACE

IS THAT ALRIGHT?

Enter YES or NO. If NO is entered, the Change Parameter Utility Program restarts from the beginning. Otherwise, the requested changes are made, the work space contents are cleared and the program terminates.

DISK COPY

Creating backup copies of disks is a wise precaution. The backup copy provides protection against inadvertently destroying an important program, either by writing over the program or physically damaging the disk. Two utilities are provided for disk copying on the system disk.

Copying a disk requires two disk drives. (If a dual disk system is not owned, the OSI dealer can provide these services.) In a dual disk system, one drive will be labeled "A", the other drive will be labeled "B". Since it is intended that material on one disk be overwritten with material from another disk, extreme caution is urged in following the order of instructions. Otherwise, it is possible to end up with two copies of the wrong disk!

First, select a disk on which to make a copy. This can be a new disk or a spare old disk. This disk should be initialized, a process of placing information on disk for timing purposes. Since this process will overwrite the entire disk, make sure this disk is truly available.

To initialize the disk, enter the operating system (From BASIC, type *EXIT*). Place the disk ONTO WHICH a copy is to be made in drive B. In response to the system prompt, type

A* SE B <RETURN>

Reply to the system response by

B* INIZ <RETURN>

The system will ask

ARE YOU SURE?

If this is affirmative, then type

YES

If any error message is reported, discard the disk as damaged or faulty. No errors will be reported for successful initialization. Now when the system prompt is shown, return to use Drive A by replying

B* SE A

Before using the master disk, caution encourages the covering of the rectangular notch on the side of the disk with a piece of black electrical tape. This will "WRITE PROTECT" the disk against inadvertently overwriting data and programs to be kept. This tape may be removed later. Now the master disk is ready to be copied.

Place the master disk in drive A. The already initialized disk (ONTO which the copy is made) should be in drive B. CALL in the copy utility from disk by typing

A* CALL Ø2ØØ=13.1 <RETURN>

This will load the copy routine at location Ø2ØØ hex. To execute the copy routine, type

GO Ø2ØØ <RETURN>

The result will be the choice

SELECT ONE:

1. COPIER

2. TRACK Ø READ/WRITE

Respond

?1 <RETURN>

to select the copier routine. (The TRACK Ø READ/WRITE is used to restore track Ø. This is typically needed if one powers down a disk drive with a disk in the drive.)
The question will be asked

FROM DRIVE (A/B/C/D)?

Reply

A

The dialog continues:

TO DRIVE (A/B/C/D)?

Reply

B

Tracks are selected by replying to the prompt

FROM TRACK?

by

Ø

TO TRACK (INCLUSIVE)?

39

Since adequate care has been taken to this point, the response to

ARE YOU SURE?

is

YES <RETURN>

Each track number, as it is copied, is displayed on the video screen.

CREATE A DISK FILE

It is useful to be able to name a region of disk for program or data storage. The CREATE utility is set up for this purpose. It reserves room on the disk for user programs and enters the file name into the directory for future reference.

To illustrate CREATE, turn the computer on and bring up the disk operating system (OS-65D V3.N). This process is called "booting up" the system. When the BASIC prompt

OK

appears, type

RUN"DIR" <RETURN>

Respond to the question

LIST ON LINE PRINTER INSTEAD OF DEVICE #2?

by answering

NO <RETURN>

A listing of the disk directory appears. A typical directory listing follows:

```
            OS-65D VERSION 3.N
              —DIRECTORY—

FILE NAME                    TRACK RANGE
OS65D3                           Ø —12
BEXEC*                           14—14
CHANGE                           15—16
CREATE                           17—19
DELETE                           2Ø—2Ø
DIR                              21—21
DIRSRT                           22—22
RANLST                           23—24
RENAME                           25—25
SECDIR                           26—26
SEQLST                           27—28
TRACE                            29—29
ZERO                             3Ø—31
ASMPL                            32—32

        5Ø ENTRIES FREE OUT OF 64
```

The 1Ø directory files use up 1Ø of the 64 available directory entries. Fifty (5Ø) entries remain free.

If any track between Ø and 39 does not have a file name, the user can use that track for his purposes. Now it is suggested that a file called SCRTCH be created. It is a good idea to have such a file for storing programs during development stages.) File names consist of six or fewer characters; the first character must be a letter. Type

RUN"CREATE" <RETURN>

When asked for a password, respond with

PASS <RETURN>

Then, the computer will respond with

FILE NAME?

Respond with

SCRTCH <RETURN>

The computer response

FIRST TRACK OF FILE?

will be answered with

39

(or whatever track was clear)
Assuming there is only 1 track to copy, the prompt

NUMBER OF TRACKS IN FILE?

is replied with

1

Now when

RUN"DIR"

is typed you will see this new file "SCRTCH" on the disk.

It is common practice to create a scratch file "SCRTCH." It is possible to store 2K bytes (approximately 2ØØØ characters) on a track. Take the memory size in Kbytes and subtract 12K (the approximate system requirements), leaving the BASIC work space size. For example, a 24K system needs 24K − 12K = 12K bytes of storage. Since 2K bytes fit on a track, the entire BASIC work space could be stored on 6 tracks. Small programs will obviously require far less disk storage.

APPENDIX H

HEX TO DECIMAL TUTOR

Within computers, calculations are made in zeros and ones, a binary system. This representation of numbers is more convenient than on traditional base 1Ø (decimal) system. For compact notation, the binary representation is often written by grouping multiples of 2, specifically powers of 2*2*2*2*=16. This notation, base 16, is called a hexadecimal number system.

The manual's illustrations of the ASC and CHR$ commands can be used to write a program to convert decimal numbers (counting in base 1Ø) to hexadecimal numbers (counting in base 16).

To count in the base 1Ø numbering system, the symbols Ø, 1, 2, 3, 4, 5, 6, 7, 8, and 9 are used, 1Ø symbols in all. A place holder notation is employed to represent a number, so that

123 = 1*1Ø∧2 + 2*1Ø∧1 + 3*1Ø∧Ø = 1ØØ+ 2Ø + 3

= 1*1ØØ + 2*1Ø + 3*1

= 1ØØ + 2Ø + 3

(where ∧ indicates "to the power").

In the other case, base 16 (hexadecimal) counting will require 16 symbols. By common agreement the symbols are Ø, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F. Here A hexadecimal corresponds to 1Ø decimal, B hexadecimal corresponds to 11 decimal, etc. Therefore, the number

123 hexadecimal = 1*16∧2 + 2*16∧1 + 3*16∧Ø decimal

= 1*256 + 2*16 + 3*1 decimal

= 256 + 32 + 3 decimal = 291 decimal

Similarly, the number 3A hexadecimal is

3A hexadecimal = 3*16∧1 + 1Ø*16∧Ø decimal

= 3*16 + 1Ø*1 decimal

= 48 + 1Ø decimal

= 58 decimal

This much calculation is a sure candidate for a computer program. Also, in some of the advanced programming techniques, it will be necessary to be able to convert from one system to another. This problem of number system conversion provides a chance to use the ASCII conversion commands in the programming. Moreover, this program is readily modified to permit data entry into programs in either hexadecimal or decimal. For occasional conversions, there is also provided a decimal to hexadecimal conversion table elsewhere in the appendix. Now look at the ASCII code table in Appendix I.

Symbols Ø through 9 have ASCII codes of 48 to 57 decimal. By subtracting 48 from this ASCII decimal code, the results are the numerals in the range Ø to 9. For example, the ASCII code for 3 is given as:

ASCII Code for symbol "3" = 51

If we subtract 48 from 51 (the ASCII code value of the number 3), we get the numeric value, 3.

ASCII code for symbol "3" = 48 = 51−48 = 3

This observation permits the change of the code representation of numbers Ø to 9 into the numbers, themselves.

Similarly, the symbols A to F are represented by ASCII codes of 65 to 7Ø decimal. By subtracting 55 from this code, the decimal value which the hexadecimal notation implies can be obtained.

In summary,

1. the ASCII code for the symbol "A" = 65
2. the number A hexadecimal = 1Ø decimal
3. thus the ASCII code for "A" − 55 = 65−55 = 1Ø

permit the conversion of values for the ASCII symbols A to F. This conversion can be used to complete the algorithm for conversions from hexadecimal to decimal.

To go from decimal to hexadecimal (the reverse direction), note how remainders from division yield the separate digit's representation. For example, in base 1Ø, for the number 123, do successive divisions, and observe the remainder;

```
1Ø / 123
1Ø /  12 + remainder 3 ∧
1Ø /   1 + remainder 2 ∧
       Ø + remainder 1 ∧
```

yields the base 1Ø representation when read in the direction of the arrow. Trying this in base 16 to find the hexadecimal value of 2Ø decimal

```
16 / 2Ø
16 /  1 + remainder 4 ∧
      Ø + remainder 1 ∧
```

gives the hexadecimal value of 14 when read in the direction of the arrow. This checks since

```
1*16∧1 + 4*16∧Ø = 2Ø
```

Slightly harder is converting 28 decimal

```
16 / 28
16 /+1 + remainder 12 = B hexadecimal ∧
     Ø + remainder 1 = 1 hexadecimal ∧
```

giving the hexadecimaal value of 1B. Next, combine these two conversion algorithms in a flow chart, Fig. 27, shown in overall form.

It is common practice to indicate hex numbers by use of a leading $, for example, DEØØ hex = $DEØØ.

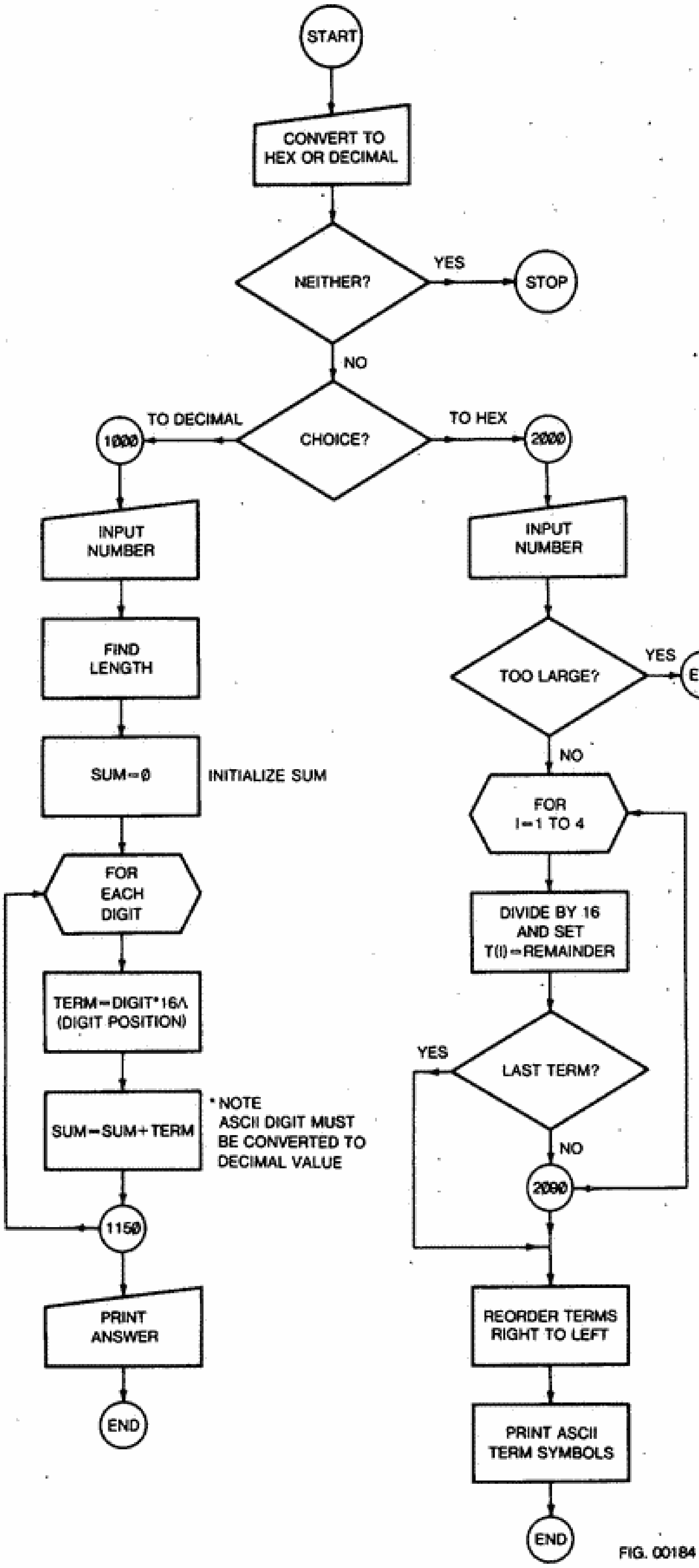


Fig. 27 Flow Chart (Hexadecimal to Decimal)

```
1Ø REM HEX AN OSI PROGRAM TO CONVERT
2Ø REM 1) HEXADECIMAL (BASE 16) TO DECIMAL OR
3Ø REM 2) DECIMAL TO HEXADECIMAL: L ROEMER 28 MAY 1979
35 PRINT" TYPE ":PRINT" 1 FOR HEX TO DECIMAL
36 PRINT" 2 FOR DECIMAL TO HEX"
4Ø INPUT "YOUR CHOICE IS"; CHOICE
5Ø IF CHOICE=1 THEN GOSUB 1ØØØ: REM HEX TO DECIMAL
6Ø IF CHOICE=2 THEN GOSUB 2ØØØ: REM DECIMAL TO HEX
7Ø IF CHOICE <> 1 AND CHOICE <> 2 THEN GOSUB 3ØØØ
8Ø END
1ØØ REM CONVERT EACH CHARACTER TO ASCII CODE
1ØØØ REM HEX INPUT TO DECIMAL OUTPUT
1Ø1Ø INPUT "YOUR HEX NUMBER IS"; A$
1Ø2Ø L=LEN(A$)
1Ø3Ø SUM=Ø
1Ø4Ø REM WHEN EXAMINE CHARACTERS, LOW POSITION
1Ø5Ø REM IS AT RIGHT HAND
1Ø6Ø FOR K=1 TO L
1Ø7Ø M=L+1-K
1Ø8Ø T2=ASC(MID$(A$,M,1))
11ØØ S1=SUM+16*(K-1)*(T2-55)
111Ø S2=SUM+16*(K-1)*(T2-48)
113Ø IF T2> 64 THEN SUM=S1:REM CHECK IF HEX CHAR> 9
114Ø IF T2 <64 THEN SUM=S2:REM OR <9
115Ø NEXT K
116Ø PRINT "DECIMAL VALUE IS"; SUM
117Ø RETURN
118Ø END
2ØØØ REM DECIMAL INPUT WITH HEX OUTPUT
2Ø1Ø INPUT "YOUR DECIMAL IS"; D
2Ø2Ø IF D> 65535 THEN GOTO 26ØØ
2Ø3Ø T(Ø)=D
2Ø4Ø FOR I=1 TO 4
2Ø5Ø T(I)=INT(T(I-1)/16)
2Ø6Ø CI(I)=T(I-1)-T(I)*16
2Ø7Ø K=I
```

```
2Ø8Ø IF INT(T(I))=Ø THEN GOTO 22ØØ
2Ø9Ø NEXT I
22ØØ FOR I=1 TO K
221Ø REM: REVERSE ORDER OF DIGITS FOR PRINTING
222Ø CH$(K+1-I)=CHR$(48+CI(I))
223Ø IF CI(I)> 9 THEN CH$(K+1-I)=CHR$(55+CI(I))
224Ø NEXT I
225Ø ZIPS/=" "
226Ø FOR I=1 TO K
227Ø ZIP$=ZIP$+CH$(I)
228Ø NEXT I
229Ø PRINT "HEX"; ZIP$
23ØØ RETURN
231Ø END
26ØØ PRINT "TOO LARGE A VALUE"
261Ø END
3ØØØ PRINT "YOUR CHOICE SHOULD BE 1 OR 2"
3Ø1Ø PRINT "RUN AGAIN IF YOU WISH CHOICE"
3Ø2Ø RETURN
3Ø3Ø END
```

HEXADECIMAL-DECIMAL CONVERSION

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
000	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
010	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
020	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
030	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
040	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79
050	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95
060	96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111
070	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127
080	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143
090	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159
0A0	160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175
0B0	176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191
0C0	192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207
0D0	208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223
0E0	224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239
0F0	240	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255
100	256	257	258	259	260	261	262	263	264	265	266	267	268	269	270	271
110	272	273	274	275	276	277	278	279	280	281	282	283	284	285	286	287
120	288	289	290	291	292	293	294	295	296	297	298	299	300	301	302	303
130	304	305	306	307	308	309	310	311	312	313	314	315	316	317	318	319
140	320	321	322	323	324	325	326	327	328	329	330	331	332	333	334	335
150	336	337	338	339	340	341	342	343	344	345	346	347	348	349	350	351
160	352	353	354	355	356	357	358	359	360	361	362	363	364	365	366	367
170	368	369	370	371	372	373	374	375	376	377	378	379	380	381	382	383
180	384	385	386	387	388	389	390	391	392	393	394	395	396	397	398	399
190	400	401	402	403	404	405	406	407	408	409	410	411	412	413	414	415
1A0	416	417	418	419	420	421	422	423	424	425	426	427	428	429	430	431
1B0	432	433	434	435	436	437	438	439	440	441	442	443	444	445	446	447
1C0	448	449	450	451	452	453	454	455	456	457	458	459	460	461	462	463
1D0	464	465	466	467	468	469	470	471	472	473	474	475	476	477	478	479
1E0	480	481	482	483	484	485	486	487	488	489	490	491	492	493	494	495
1F0	496	497	498	499	500	501	502	503	504	505	506	507	508	509	510	511
200	512	513	514	515	516	517	518	519	520	521	522	523	524	525	526	527
210	528	529	530	531	532	533	534	535	536	537	538	539	540	541	542	543
220	544	545	546	547	548	549	550	551	552	553	554	555	556	557	558	559
230	560	561	562	563	564	565	566	567	568	569	570	571	572	573	574	575
240	576	577	578	579	580	581	582	583	584	585	586	587	588	589	590	591
250	592	593	594	595	596	597	598	599	600	601	602	603	604	605	606	607
260	608	609	610	611	612	613	614	615	616	617	618	619	620	621	622	623
270	624	625	626	627	628	629	630	631	632	633	634	635	636	637	638	639
280	640	641	642	643	644	645	646	647	648	649	650	651	652	653	654	655
290	656	657	658	659	660	661	662	663	664	665	666	667	668	669	670	671
2A0	672	673	674	675	676	677	678	679	680	681	682	683	684	685	686	687
2B0	688	689	690	691	692	693	694	695	696	697	698	699	700	701	702	703
2C0	704	705	706	707	708	709	710	711	712	713	714	715	716	717	718	719
2D0	720	721	722	723	724	725	726	727	728	729	730	731	732	733	734	735
2E0	736	737	738	739	740	741	742	743	744	745	746	747	748	749	750	751
2F0	752	753	754	755	756	757	758	759	760	761	762	763	764	765	766	767
300	768	769	770	771	772	773	774	775	776	777	778	779	780	781	782	783
310	784	785	786	787	788	789	790	791	792	793	794	795	796	797	798	799
320	800	801	802	803	804	805	806	807	808	809	810	811	812	813	814	815
330	816	817	818	819	820	821	822	823	824	825	826	827	828	829	830	831
340	832	833	834	835	836	837	838	839	840	841	842	843	844	845	846	847
350	848	849	850	851	852	853	854	855	856	857	858	859	860	861	862	863
360	864	865	866	867	868	869	870	871	872	873	874	875	876	877	878	879
370	880	881	882	883	884	885	886	887	888	889	890	891	892	893	894	895
380	896	897	898	899	900	901	902	903	904	905	906	907	908	909	910	911
390	912	913	914	915	916	917	918	919	920	921	922	923	924	925	926	927
3A0	928	929	930	931	932	933	934	935	936	937	938	939	940	941	942	943
3B0	944	945	946	947	948	949	950	951	952	953	954	955	956	957	958	959
3C0	960	961	962	963	964	965	966	967	968	969	970	971	972	973	974	975
3D0	976	977	978	979	980	981	982	983	984	985	986	987	988	989	990	991
3E0	992	993	994	995	996	997	998	999	1000	1001	1002	1003	1004	1005	1006	1007
3F0	1008	1009	1010	1011	1012	1013	1014	1015	1016	1017	1018	1019	1020	1021	1022	1023
400	1024	1025	1026	1027	1028	1029	1030	1031	1032	1033	1034	1035	1036	1037	1038	1039
410	1040	1041	1042	1043	1044	1045	1046	1047	1048	1049	1050	1051	1052	1053	1054	1055
420	1056	1057	1058	1059	1060	1061	1062	1063	1064	1065	1066	1067	1068	1069	1070	1071
430	1072	1073	1074	1075	1076	1077	1078	1079	1080	1081	1082	1083	1084	1085	1086	1087
440	1088	1089	1090	1091	1092	1093	1094	1095	1096	1097	1098	1099	1100	1101	1102	1103
450	1104	1105	1106	1107	1108	1109	1110	1111	1112	1113	1114	1115	1116	1117	1118	1119
460	1120	1121	1122	1123	1124	1125	1126	1127	1128	1129	1130	1131	1132	1133	1134	1135
470	1136	1137	1138	1139	1140	1141	1142	1143	1144	1145	1146	1147	1148	1149	1150	1151
480	1152	1153	1154	1155	1156	1157	1158	1159	1160	1161	1162	1163	1164	1165	1166	1167
490	1168	1169	1170	1171	1172	1173	1174	1175	1176	1177	1178	1179	1180	1181	1182	1183
4A0	1184	1185	1186	1187	1188	1189	1190	1191	1192	1193	1194	1195	1196	1197	1198	1199
4B0	1200	1201	1202	1203	1204	1205	1206	1207	1208	1209	1210	1211	1212	1213	1214	1215
4C0	1216	1217	1218	1219	1220	1221	1222	1223	1224	1225	1226	1227	1228	1229	1230	1231
4D0	1232	1233	1234	1235	1236	1237	1238	1239	1240	1241	1242	1243	1244	1245	1246	1247
4E0	1248	1249	1250	1251	1252	1253	1254	1255	1256	1257	1258	1259	1260	1261	1262	1263
4F0	1264	1265	1266	1267	1268	1269	1270	1271	1272	1273	1274	1275	1276	1277	1278	1279

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
500	1280	1281	1282	1283	1284	1285	1286	1287	1288	1289	1290	1291	1292	1293	1294	1295
510	1296	1297	1298	1299	1300	1301	1302	1303	1304	1305	1306	1307	1308	1309	1310	1311
520	1312	1313	1314	1315	1316	1317	1318	1319	1320	1321	1322	1323	1324	1325	1326	1327
530	1328	1329	1330	1331	1332	1333	1334	1335	1336	1337	1338	1339	1340	1341	1342	1343
540	1344	1345	1346	1347	1348	1349	1350	1351	1352	1353	1354	1355	1356	1357	1358	1359
550	1360	1361	1362	1363	1364	1365	1366	1367	1368	1369	1370	1371	1372	1373	1374	1375
560	1376	1377	1378	1379	1380	1381	1382	1383	1384	1385	1386	1387	1388	1389	1390	1391
570	1392	1393	1394	1395	1396	1397	1398	1399	1400	1401	1402	1403	1404	1405	1406	1407
580	1408	1409	1410	1411	1412	1413	1414	1415	1416	1417	1418	1419	1420	1421	1422	1423
590	1424	1425	1426	1427	1428	1429	1430	1431	1432	1433	1434	1435	1436	1437	1438	1439
5A0	1440	1441	1442	1443	1444	1445	1446	1447	1448	1449	1450	1451	1452	1453	1454	1455
5B0	1456	1457	1458	1459	1460	1461	1462	1463	1464	1465	1466	1467	1468	1469	1470	1471
5C0	1472	1473	1474	1475	1476	1477	1478	1479	1480	1481	1482	1483	1484	1485	1486	1487
5D0	1488	1489	1490	1491	1492	1493	1494	1495	1496	1497	1498	1499	1500	1501	1502	1503
5E0	1504	1505	1506	1507	1508	1509	1510	1511	1512	1513	1514	1515	1516	1517	1518	1519
5F0	1520	1521	1522	1523	1524	1525	1526	1527	1528	1529	1530	1531	1532	1533	1534	1535
600	1536	1537	1538	1539	1540	1541	1542	1543	1544	1545	1546	1547	1548	1549	1550	1551
610	1552	1553	1554	1555	1556	1557	1558	1559	1560	1561	1562	1563	1564	1565	1566	1567
620	1568	1569	1570	1571	1572	1573	1574	1575	1576	1577	1578	1579	1580	1581	1582	1583
630	1584	1585	1586	1587	1588	1589	1590	1591	1592	1593	1594	1595	1596	1597	1598	1599
640	1600	1601	1602	1603	1604	1605	1606	1607	1608	1609	1610	1611	1612	1613	1614	1615
650	1616	1617	1618	1619	1620	1621	1622	1623	1624	1625	1626	1627	1628	1629	1630	1631
660	1632	1633	1634	1635	1636	1637	1638	1639	1640	1641	1642	1643	1644	1645	1646	1647
670	1648	1649	1650	1651	1652	1653	1654	1655	1656	1657	1658	1659	1660	1661	1662	1663
680	1664	1665	1666	1667	1668	1669	1670	1671	1672	1673	1674	1675	1676	1677	1678	1679
690	1680	1681	1682	1683	1684	1685	1686	1687	1688	1689	1690	1691	1692	1693	1694	1695
6A0	1696	1697	1698	1699	1700	1701	1702	1703	1704	1705	1706	1707	1708	1709	1710	1711
6B0	1712	1713	1714	1715	1716	1717	1718	1719	1720	1721	1722	1723	1724	1725	1726	1727
6C0	1728	1729	1730	1731	1732	1733	1734	1735	1736	1737	1738	1739	1740	1741	1742	1743
6D0	1744	1745	1746	1747	1748	1749	1750	1751	1752	1753	1754	1755	1756	1757	1758	1759
6E0	1760	1761	1762	1763	1764	1765	1766	1767	1768	1769	1770	1771	1772	1773	1774	1775
6F0	1776	1777	1778	1779	1780	1781	1782	1783	1784	1785	1786	1787	1788	1789	1790	1791
700	1792	1793	1794	1795	1796	1797	1798	1799	1800	1801	1802	1803	1804	1805	1806	1807
710	1808	1809	1810	1811	1812	1813	1814	1815	1816	1817	1818	1819	1820	1821	1822	1823
720	1824	1825	1826	1827	1828	1829	1830	1831	1832	1833	1834	1835	1836	1837	1838	1839
730	1840	1841	1842	1843	1844	1845	1846	1847	1848	1849	1850	1851	1852	1853	1854	1855
740	1856	1857	1858	1859	1860	1861	1862	1863	1864	1865	1866	1867	1868	1869	1870	1871
750	1872	1873	1874	1875	1876	1877	1878	1879	1880	1881	1882	1883	1884	1885	1886	1887
760	1888	1889	1890	1891	1892	1893	1894	1895	1896	1897	1898	1899	1900	1901	1902	1903
770	1904	1905	1906	1907	1908	1909	1910	1911	1912	1913	1914	1915	1916	1917	1918	1919
780	1920	1921	1922	1923	1924	1925	1926	1927	1928	1929	1930	1931	1932	1933	1934	1935
790	1936	1937	1938	1939	1940	1941	1942	1943	1944	1945	1946	1947	1948	1949	1950	1951
7A0	1952	1953	1954	1955	1956	1957	1958	1959	1960	1961	1962	1963	1964	1965	1966	1967
7B0	1968	1969	1970	1971	1972	1973	1974	1975	1976	1977	1978	1979	1980	1981	1982	1983
7C0	1984	1985	1986	1987	1988	1989	1990	1991	1992	1993	1994	1995	1996	1997	1998	1999
7D0	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015
7E0	2016	2017	2018	2019	2020	2021	2022	2023	2024	2025	2026	2027	2028	2029	2030	2031
7F0	2032	2033	2034	2035	2036	2037	2038	2039	2040	2041	2042	2043	2044	2045	2046	2047
800	2048	2049	2050	2051	2052	2053	2054	2055	2056	2057	2058	2059	2060	2061	2062	2063
810	2064	2065	2066	2067	2068	2069	2070	2071	2072	2073	2074	2075	2076	2077	2078	2079
820	2080	2081	2082	2083	2084	2085	2086	2087	2088	2089	2090	2091	2092	2093	2094	2095
830	2096	2097	2098	2099	2100	2101	2102	2103	2104	2105	2106	2107	2108	2109	2110	2111
840	2112	2113	2114	2115	2116	2117	2118	2119	2120	2121	2122	2123	2124	2125	2126	2127
850	2128	2129	2130	2131	2132	2133	2134	2135	2136	2137	2138	2139	2140	2141	2142	2143
860	2144	2145	2146	2147	2148	2149	2150	2151	2152	2153	2154	2155	2156	2157	2158	2159
870	2160	2161	2162	2163	2164	2165	2166	2167	2168	2169	2170	2171	2172	2173	2174	2175
880	2176	2177	2178	2179	2180	2181	2182	2183	2184	2185	2186	2187	2188	2189	2190	2191
890	2192	2193	2194	2195	2196	2197	2198	2199	2200	2201	2202	2203	2204	2205	2206	2207
8A0	2208	2209	2210	2211	2212	2213	2214	2215	2216	2217	2218	2219	2220	2221	2222	2223
8B0	2224	2225	2226	2227	2228	2229	2230	2231	2232	2233	2234	2235	2236	2237	2238	2239
8C0	2240	2241	2242	2243	2244	2245	2246	2247	2248	2249	2250	2251	2252	2253	2254	2255
8D0	2256	2257	2258	2259	2260	2261	2262	2263	2264	2265	2266	2267	2268	2269	2270	2271
8E0	2272	2273	2274	2275	2276	2277	2278	2279	2280	2281	2282	2283	2284	2285	2286	2287
8F0	2288	2289	2290	2291	2292	2293	2294	2295	2296	2297	2298	2299	2300	2301	2302	2303
900	2304	2305	2306	2307	2308	2309	2310	2311	2312	2313	2314	2315	2316	2317	2318	2319
910	2320	2321	2322	2323	2324	2325	2326	2327	2328	2329	2330	2331	2332	2333	2334	2335
920	2336	2337	2338	2339	2340	2341	2342	2343	2344	2345	2346	2347	2348	2349	2350	2351
930	2352	2353	2354	2355	2356	2357	2358	2359	2360	2361	2362	2363	2364	2365	2366	2367
940	2368	2369	2370	2371	2372	2373	2374	2375	2376	2377	2378	2379	2380	2381	2382	2383
950	2384	2385	2386	2387	2388	2389	2390	2391	2392	2393	2394	2395	2396	2397	2398	2399
960	2400	2401	2402	2403	2404	2405	2406	2407	2408	2409	2410	2411	2412	2413	2414	2415
970	2416	2417	2418	2419	2420	2421	2422	2423	2424	2425	2426	2427	2428	2429	2430	2431
980	2432	2433	2434	2435	2436	2437	2438	2439	2440	2441	2442	2443	2444	2445	2446	2447
990	2448	2449	2450	2451	2452	2543	2454	2455	2456	2457	2458	2459	2460	2461	2462	2463
9A0	2464	2465	2466	2467	2468	2469	2470	2471	2472	2473	2474	2475	2476	2477	2478	2479
9B0	2480	2481	2482	2483	2484	2485	2486	2487	2488	2489	2490	2491	2492	2493	2494	2495
9C0	2496	2497	2498	2499	2500	2501	2502	2503	2504	2505	2506	2507	2508	2509	2510	2511
9D0	2512	2513	2514	2515	2516	2517	2518	2519	2520	2521	2522	2523	2524	2525	2526	2527
9E0	2528	2529	2530	2531	2532	2533	2534	2535	2536	2537	2538	2539	2540	2541	2542	2543
9F0	2544	2545	2546	2547	2548	2549	2550	2551	2552	2553	2554	2555	2556	2557	2558	2559

HEXADECIMAL-DECIMAL CONVERSION

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A00	2560	2561	2562	2563	2564	2565	2566	2567	2568	2569	2570	2571	2572	2573	2574	2575
A10	2576	2577	2578	2579	2580	2581	2582	2583	2584	2585	2586	2587	2588	2589	2590	2591
A20	2592	2593	2594	2595	2596	2597	2598	2599	2600	2601	2602	2603	2604	2605	2606	2607
A30	2608	2609	2610	2611	2612	2613	2614	2615	2616	2617	2618	2619	2620	2621	2622	2623
A40	2624	2625	2626	2627	2628	2629	2630	2631	2632	2633	2634	2635	2636	2637	2638	2639
A50	2640	2641	2642	2643	2644	2645	2646	2647	2648	2649	2650	2651	2652	2653	2654	2655
A60	2656	2657	2658	2659	2660	2661	2662	2663	2664	2665	2666	2667	2668	2669	2670	2671
A70	2672	2673	2674	2675	2676	2677	2678	2679	2680	2681	2682	2683	2684	2685	2686	2687
A80	2688	2689	2690	2691	2692	2693	2694	2695	2696	2697	2698	2699	2700	2701	2702	2703
A90	2704	2705	2706	2707	2708	2709	2710	2711	2712	2713	2714	2715	2716	2717	2718	2719
AA0	2720	2721	2722	2723	2724	2725	2726	2727	2728	2729	2730	2731	2732	2733	2734	2735
AB0	2736	2737	2738	2739	2740	2741	2742	2743	2744	2745	2746	2747	2748	2749	2750	2751
AC0	2752	2753	2754	2755	2756	2757	2758	2759	2760	2761	2762	2763	2764	2765	2766	2767
AD0	2768	2769	2770	2771	2772	2773	2774	2775	2776	2777	2778	2779	2780	2781	2782	2783
AE0	2784	2785	2786	2787	2788	2789	2790	2791	2792	2793	2794	2795	2796	2797	2798	2799
AF0	2800	2801	2802	2803	2804	2805	2806	2807	2808	2809	2810	2811	2812	2813	2814	2815
B00	2816	2817	2818	2819	2820	2821	2822	2823	2824	2825	2826	2827	2828	2829	2830	2831
B10	2832	2833	2834	2835	2836	2837	2838	2839	2840	2841	2842	2843	2844	2845	2846	2847
B20	2848	2849	2850	2851	2852	2853	2854	2855	2856	2857	2858	2859	2860	2861	2862	2863
B30	2864	2865	2866	2867	2868	2869	2870	2871	2872	2873	2874	2875	2876	2877	2878	2879
B40	2880	2881	2882	2883	2884	2885	2886	2887	2888	2889	2890	2891	2892	2893	2894	2895
B50	2896	2897	2898	2899	2900	2901	2902	2903	2904	2905	2906	2907	2908	2909	2910	2911
B60	2912	2913	2914	2915	2916	2917	2918	2919	2920	2921	2922	2923	2924	2925	2926	2927
B70	2928	2929	2930	2931	2932	2933	2934	2935	2936	2937	2938	2939	2940	2941	2942	2943
B80	2944	2945	2946	2947	2948	2949	2950	2951	2952	2953	2954	2955	2956	2957	2958	2959
B90	2960	2961	2962	2963	2964	2965	2966	2967	2968	2969	2970	2971	2972	2973	2974	2975
BA0	2976	2977	2978	2979	2980	2981	2982	2983	2984	2985	2986	2987	2988	2989	2990	2991
BB0	2992	2993	2994	2995	2996	2997	2998	2999	3000	3001	3002	3003	3004	3005	3006	3007
BC0	3008	3009	3010	3011	3012	3013	3014	3015	3016	3017	3018	3019	3020	3021	3022	3023
BD0	3024	3025	3026	3027	3028	3029	3030	3031	3032	3033	3034	3035	3036	3037	3038	3039
BE0	3040	3041	3042	3043	3044	3045	3046	3047	3048	3049	3050	3051	3052	3053	3054	3055
BF0	3056	3057	3058	3059	3060	3061	3062	3063	3064	3065	3066	3067	3068	3069	3070	3071
C00	3072	3073	3074	3075	3076	3077	3078	3079	3080	3081	3082	3083	3084	3085	3086	3087
C10	3088	3089	3090	3091	3092	3093	3094	3095	3096	3097	3098	3099	3100	3101	3102	3103
C20	3104	3105	3106	3107	3108	3109	3110	3111	3112	3113	3114	3115	3116	3117	3118	3119
C30	3120	3121	3122	3123	3124	3125	3126	3127	3128	3129	3130	3131	3132	3133	3134	3135
C40	3136	3137	3138	3139	3140	3141	3142	3143	3144	3145	3146	3147	3148	3149	3150	3151
C50	3152	3153	3154	3155	3156	3157	3158	3159	3160	3161	3162	3163	3164	3165	3166	3167
C60	3168	3169	3170	3171	3172	3173	3174	3175	3176	3177	3178	3179	3180	3181	3182	3183
C70	3184	3185	3186	3187	3188	3189	3190	3191	3192	3193	3194	3195	3196	3197	3198	3199
C80	3200	3201	3202	3203	3204	3205	3206	3207	3208	3209	3210	3211	3212	3213	3214	3215
C90	3216	3217	3218	3219	3220	3221	3222	3223	3224	3225	3226	3227	3228	3229	3230	3231
CA0	3232	3233	3234	3235	3236	3237	3238	3239	3240	3241	3242	3243	3244	3245	3246	3247
CB0	3248	3249	3250	3251	3252	3253	3254	3255	3256	3257	3258	3259	3260	3261	3262	3263
CC0	3264	3265	3266	3267	3268	3269	3270	3271	3272	3273	3274	3275	3276	3277	3278	3279
CD0	3280	3281	3282	3283	3284	3285	3286	3287	3288	3289	3290	3291	3292	3293	3294	3295
CE0	3296	3297	3298	3299	3300	3301	3302	3303	3304	3305	3306	3307	3308	3309	3310	3311
CF0	3312	3313	3314	3315	3316	3317	3318	3319	3320	3321	3322	3323	3324	3325	3326	3327
D00	3328	3329	3330	3331	3332	3333	3334	3335	3336	3337	3338	3339	3340	3341	3342	3343
D10	3344	3345	3346	3347	3348	3349	3350	3351	3352	3353	3354	3355	3356	3357	3358	3359
D20	3360	3361	3362	3363	3364	3365	3366	3367	3368	3369	3370	3371	3372	3373	3374	3375
D30	3376	3377	3378	3379	3380	3381	3382	3383	3384	3385	3386	3387	3388	3389	3390	3391
D40	3392	3393	3394	3395	3396	3397	3398	3399	3400	3401	3402	3403	3404	3405	3406	3407
D50	3408	3409	3410	3411	3412	3413	3414	3415	3416	3417	3418	3419	3420	3421	3422	3423
D60	3424	3425	3426	3427	3428	3429	3430	3431	3432	3433	3434	3435	3436	3437	3438	3439
D70	3440	3441	3442	3443	3444	3445	3446	3447	3448	3449	3450	3451	3452	3453	3454	3455
D80	3456	3457	3458	3459	3460	3461	3462	3463	3464	3465	3466	3467	3468	3469	3470	3471
D90	3472	3473	3474	3475	3476	3477	3478	3479	3480	3481	3482	3483	3484	3485	3486	3487
DA0	3488	3489	3490	3491	3492	3493	3494	3495	3496	3497	3498	3499	3500	3501	3502	3503
DB0	3504	3505	3506	3507	3508	3509	3510	3511	3512	3513	3514	3515	3516	3517	3518	3519
DC0	3520	3521	3522	3523	3524	3525	3526	3527	3528	3529	3530	3531	3532	3533	3534	3535
DD0	3536	3537	3538	3539	3540	3541	3542	3543	3544	3545	3546	3547	3548	3549	3550	3551
DE0	3552	3553	3554	3555	3556	3557	3558	3559	3560	3561	3562	3563	3564	3565	3566	3567
DF0	3568	3569	3570	3571	3572	3753	3574	3575	3576	3577	3578	3579	3580	3581	3582	3583
E00	3584	3585	3586	3587	3588	3589	3590	3591	3592	3593	3594	3595	3596	3597	3598	3599
E10	3600	3601	3602	3603	3604	3605	3606	3607	3608	3609	3610	3611	3612	3613	3614	3615
E20	3616	3617	3618	3619	3620	3621	3622	3623	3624	3625	3656	3657	3628	3629	3630	3631
E30	3632	3633	3634	3635	3636	3637	3638	3639	3640	3641	3642	3643	3644	3645	3646	3647
E40	3648	3649	3650	3651	3652	3653	3654	3655	3656	3657	3658	3659	3660	3661	3662	3663
E50	3664	3665	3666	3667	3668	3669	3670	3671	3672	3673	3674	3675	3676	3677	3678	3679
E60	3680	3681	3682	3683	3684	3685	3686	3687	3688	3689	3690	3691	3692	3693	3694	3695
E70	3696	3697	3698	3699	3700	3701	3702	3703	3704	3705	3706	3707	3708	3709	3710	3711
E80	3712	3713	3714	3715	3716	3717	3718	3719	3720	3721	3722	3723	3724	3725	3726	3727
E90	3728	3729	3730	3731	3732	3733	3734	3735	3736	3737	3738	3739	3740	3741	3742	3743
EA0	3744	3745	3746	3747	3748	3749	3750	3751	3752	3753	3754	3755	3756	3757	3758	3759
EB0	3760	3761	3762	3763	3764	3765	3766	3767	3768	3769	3770	3771	3772	3773	3774	3775
EC0	3776	3777	3778	3779	3780	3781	3782	3783	3784	3785	3786	3787	3788	3789	3790	3791
ED0	3792	3793	3794	3795	3796	3797	3798	3799	3800	3801	3802	3803	3804	3805	3806	3807
EE0	3808	3809	3810	3811	3812	3813	3814	3815	3816	3817	3818	3819	3820	3821	3822	3823
EF0	3824	3825	3826	3827	3828	3829	3830	3831	3832	3833	3834	3835	3836	3837	3838	3839

HEXADECIMAL-DECIMAL CONVERSION

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
F00	3840	3841	3842	3843	3844	3845	3846	3847	3848	3849	3850	3851	3852	3853	3854	3855
F10	3856	3857	3858	3859	3860	3861	3862	3863	3864	3865	3866	3867	3868	3869	3870	3871
F20	3872	3873	3874	3875	3876	3877	3878	3879	3880	3881	3882	3883	3884	3885	3886	3887
F30	3888	3889	3890	3891	3892	3893	3894	3895	3896	3897	3898	3899	3900	3901	3902	3903
F40	3904	3905	3906	3907	3908	3909	3910	3911	3912	3913	3914	3915	3916	3917	3918	3919
F50	3920	3921	3922	3923	3924	3925	3926	3927	3928	3929	3930	3931	3932	3933	3934	3935
F60	3936	3937	3938	3939	3940	3941	3942	3943	3944	3945	3946	3947	3948	3949	3950	3951
F70	3952	3953	3954	3955	3956	3957	3958	3959	3960	3961	3962	3963	3964	3965	3966	3967
F80	3968	3969	3970	3971	3972	3973	3974	3975	3976	3977	3978	3979	3980	3981	3982	3983
F90	3984	3985	3986	3987	3988	3989	3990	3991	3992	3993	3994	3995	3996	3997	3998	3999
FA0	4000	4001	4002	4003	4004	4005	4006	4007	4008	4009	4010	4011	4012	4013	4014	4015
FB0	4016	4017	4018	4019	4020	4021	4022	4023	4024	4025	4026	4027	4028	4029	4030	4031
FC0	4032	4033	4034	4035	4036	4037	4038	4039	4040	4041	4042	4043	4044	4045	4046	4047
FD0	4048	4049	4050	4051	4052	4053	4054	4055	4056	4057	4058	4059	4060	4061	4062	4063
FE0	4064	4065	4066	4067	4068	4069	4070	4071	4072	4073	4074	7075	4076	4077	4078	4079
FF0	4080	4081	4082	4083	4084	4085	4086	4087	4088	4089	4090	4091	4092	4093	4094	4095

HEXADECIMAL-DECIMAL INTEGER CONVERSION TABLE

HEXADECIMAL	DECIMAL
01000	4096
02000	8192
03000	12288
04000	16384
05000	20480
06000	24576
07000	28672
08000	32768
09000	36864
0A000	40960
0B000	45056
0C000	49152
0D000	53248
0E000	57344
0F000	61440
10000	65536
11000	69632
12000	73728
13000	77824
14000	81920
15000	86016
16000	90112
17000	94208
18000	98304
19000	102400
1A000	106496
1B000	110592
1C000	114688
1D000	118784
1E000	122880
1F000	126976
20000	131072

37025=909F

159

144

APPENDIX I

ASCII CODE CHART

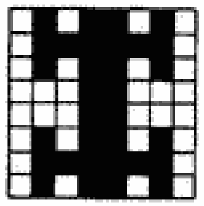
The most common ASCII code values for printed characters are:

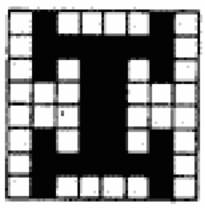
DECIMAL VALUE	HEX VALUE	SYMBOL
32	2Ø	Space
33	21	!
34	22	"
35	23	#
36	24	$
37	25	%
38	26	&
39	27	'
4Ø	28	(
41	29	)
42	2A	*
43	2B	+
44	2C	,
45	2D	–
46	2E	.
47	2F	/
48	3Ø	Ø
49	31	1
5Ø	32	2
51	33	3
52	34	4
53	35	5
54	36	6
55	37	7
56	38	8
57	39	9
58	3A	:
59	3B	;
6Ø	3C	<
61	3D	=
62	3E	>
63	3F	?
64	4Ø	@
65	41	A
66	42	B
67	43	C
68	44	D
69	45	E
7Ø	46	F
71	47	G
72	48	H
73	49	I
74	4A	J
75	4B	K

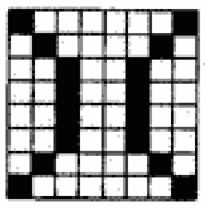
76	4C	L
77	4D	M
78	4E	N
79	4F	O
8Ø	5Ø	P
81	51	Q
82	52	R
83	53	S
84	54	T
85	55	U
86	56	V
87	57	W
88	58	X
89	59	Y
9Ø	5A	Z
91	5B	[
92	5C	\
93	5D	]
94	5E	Λ
95	5F	—

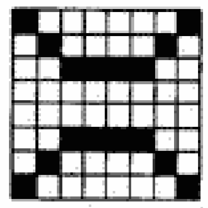
APPENDIX J

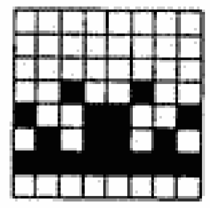
CHARACTER GRAPHICS AND VIDEO SCREEN LAYOUT

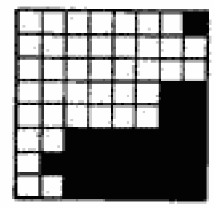

Ø $Ø

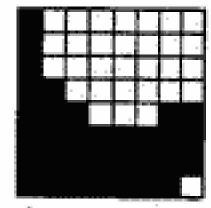

1 $1

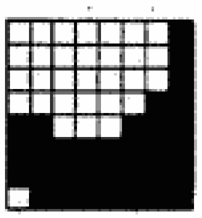

2 $2

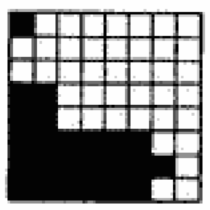

3 $3

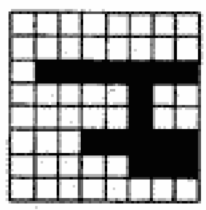

4 $4

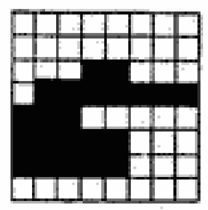

5 $5

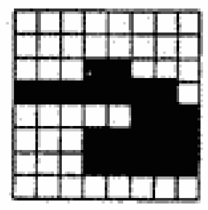

6 $6

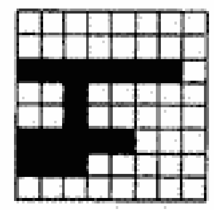

7 $7

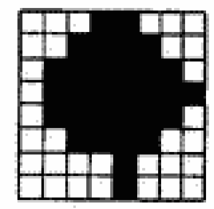

8 $8

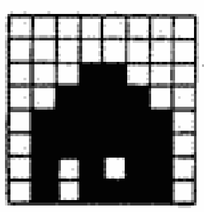

9 $9

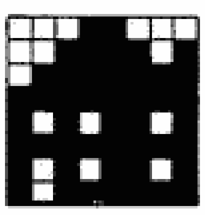

1Ø $A

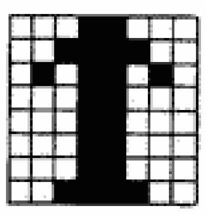

11 $B

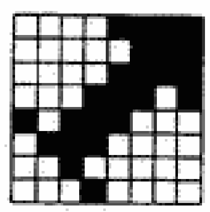

12 $C

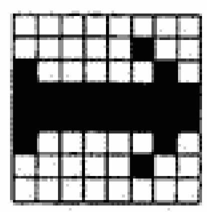

13 $D

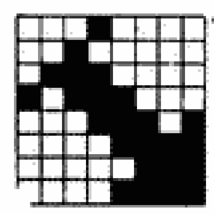

14 $E

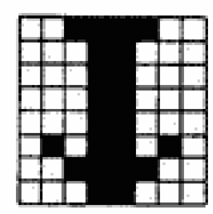

15 $F

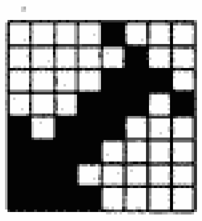

16 $1Ø

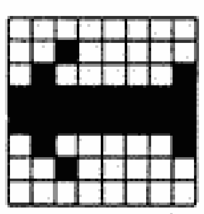

17 $11

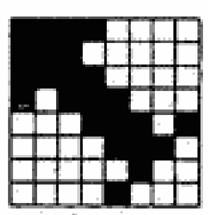

18 $12

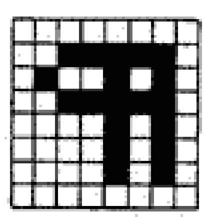

19 $13

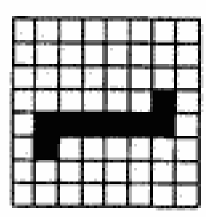

20 $14

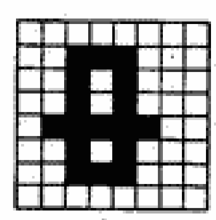

21 $15

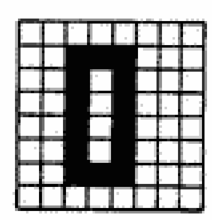

22 $16

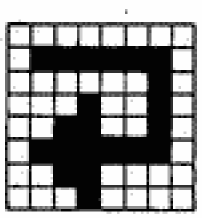

23 $17

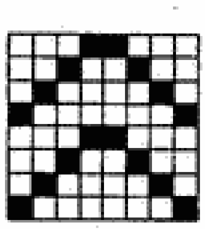

24 $18

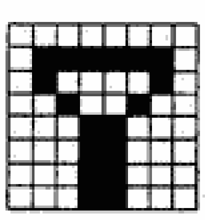

25 $19

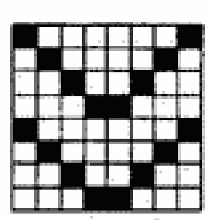

26 $1A

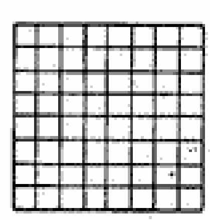

27 $1B

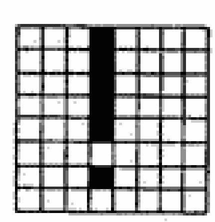

28 $1C

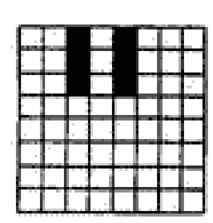

29 $1D

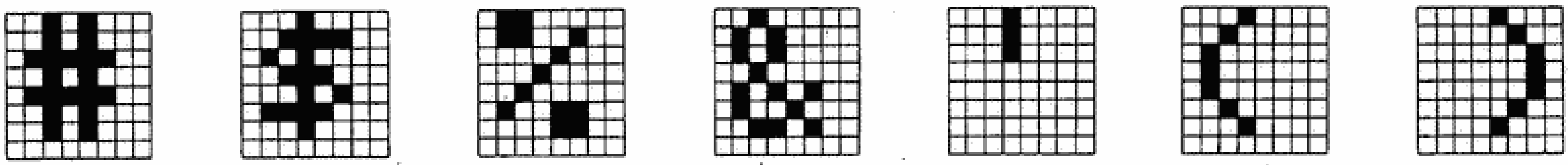

3Ø $1E


31 $1F

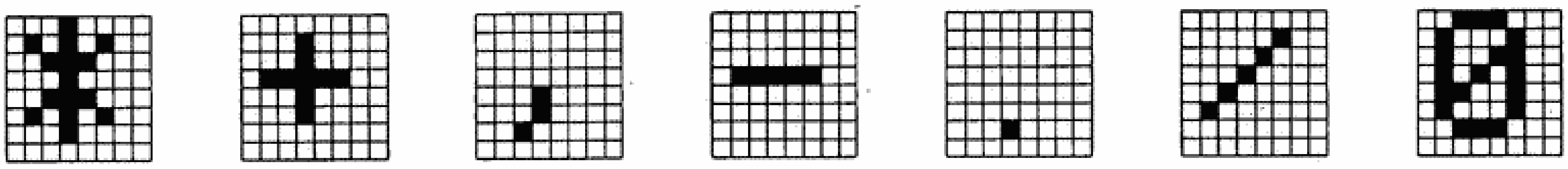

32 $2Ø


33 $21

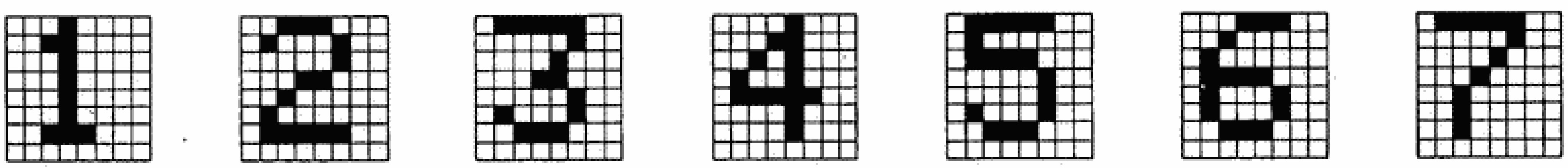

34 $22



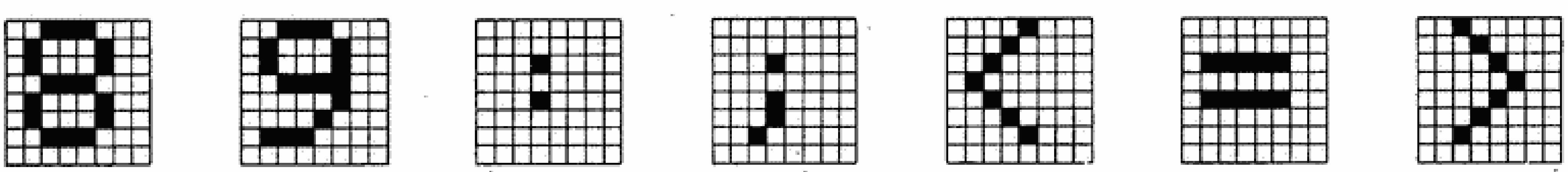
35 $23	36 $24	37 $25	38 $26	39 $27	4Ø $28	41 $29



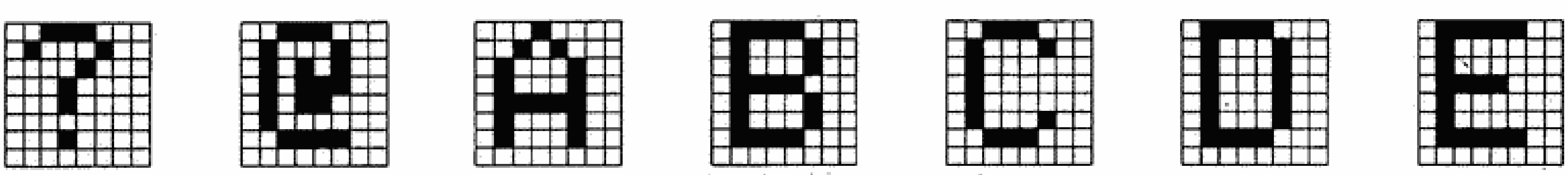
42 $2A	43 $2B	44 $2C	45 $2D	46 $2E	47 $2F	48 $3Ø



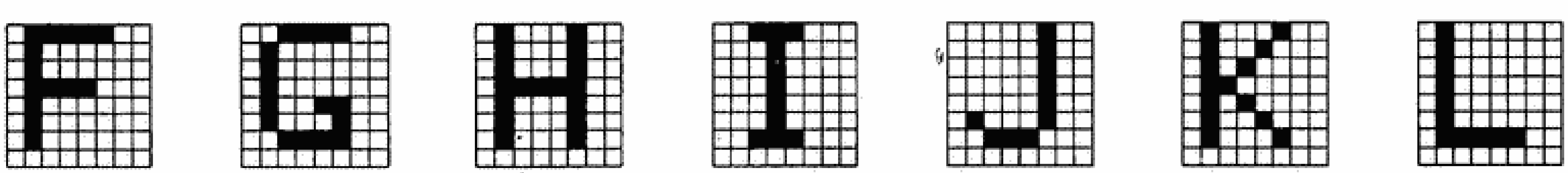
49 $31	5Ø $32	51 $33	52 $34	53 $35	54 $36	55 $37



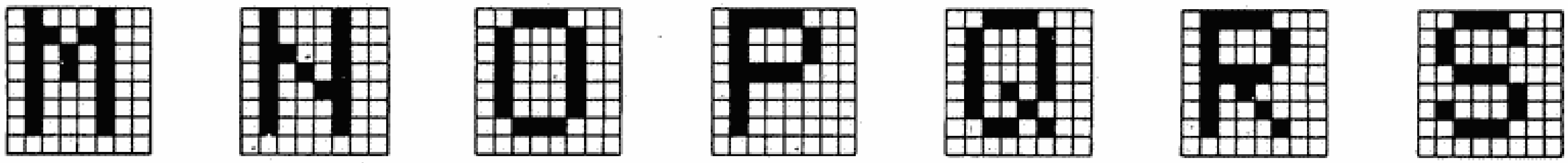
56 $38	57 $39	58 $3A	59 $3B	6Ø $3C	61 $3D	62 $3E



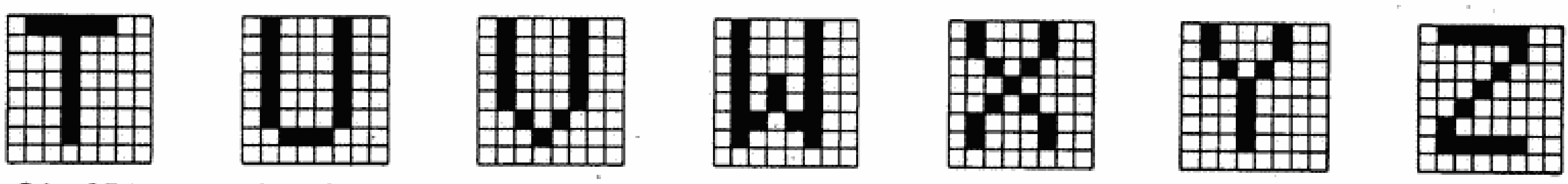
63 $3F	64 $4Ø	65 $41	66 $42	67 $43	68 $44	69 $45



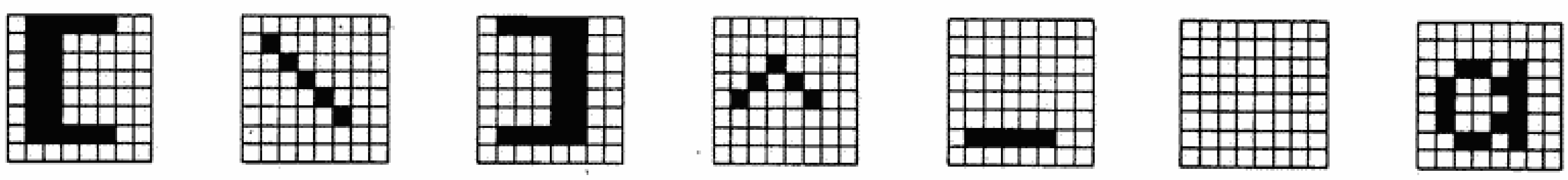
7Ø $46	71 $47	72 $48	73 $49	74 $4A	75 $4B	76 $4C



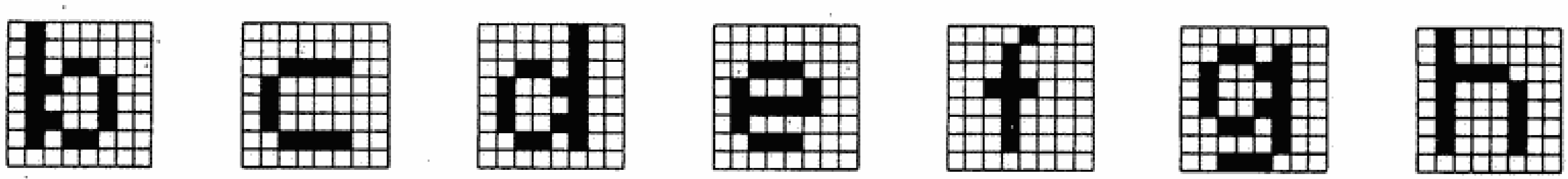
77 $4D | 78 $4E | 79 $4F | 8Ø $5Ø | 81 $51 | 82 $52 | 83 $53



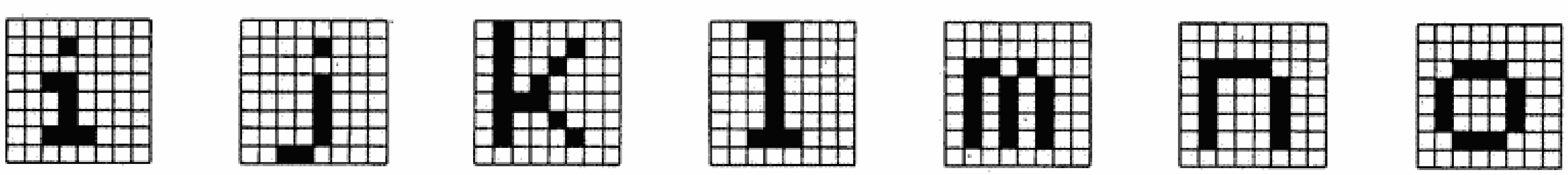
84 $54 | 85 $55 | 86 $56 | 87 $57 | 88 $58 | 89 $59 | 9Ø $5A



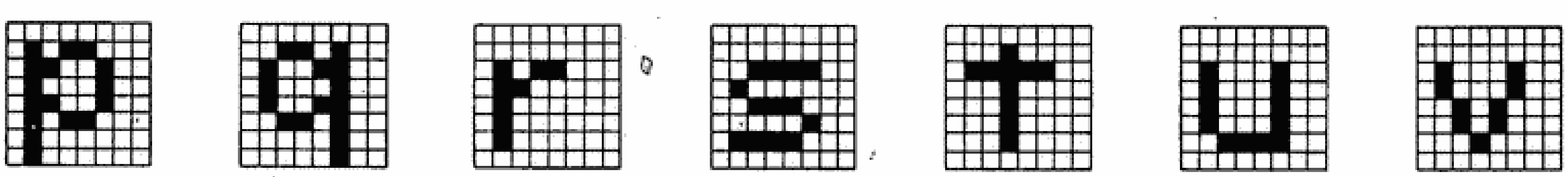
91 $5B | 92 $5C | 93 $5D | 94 $5E | 95 $5F | 96 $6Ø | 97 $61



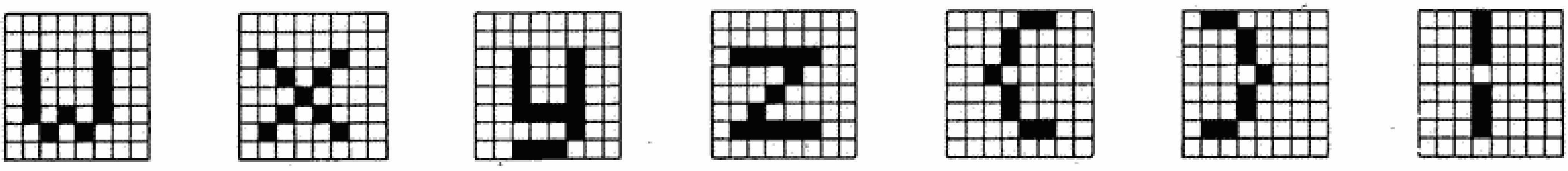
98 $62 | 99 $63 | 1ØØ $64 | 1Ø1 $65 | 1Ø2 $66 | 1Ø3 $67 | 1Ø4 $68



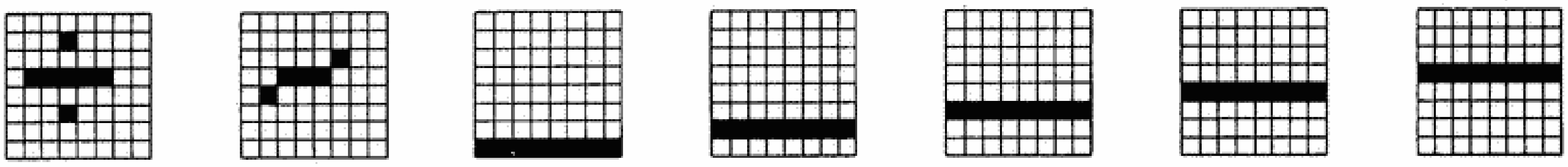
1Ø5 $69 | 1Ø6 $6A | 1Ø7 $6B | 1Ø8 $6C | 1Ø9 $6D | 11Ø $6E | 111 $6F



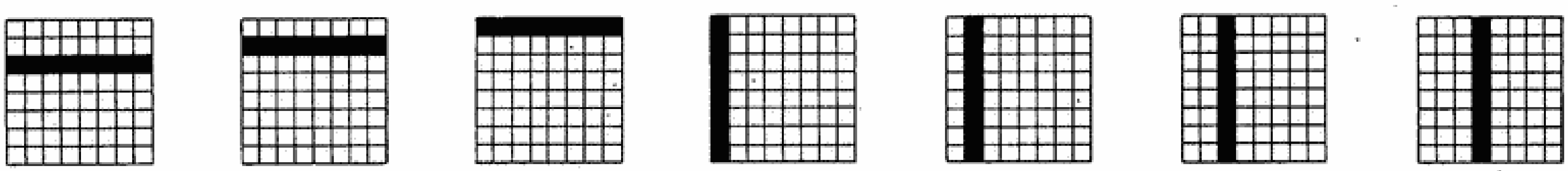
112 $7Ø | 113 $71 | 114 $72 | 115 $73 | 116 $74 | 117 $75 | 118 $76



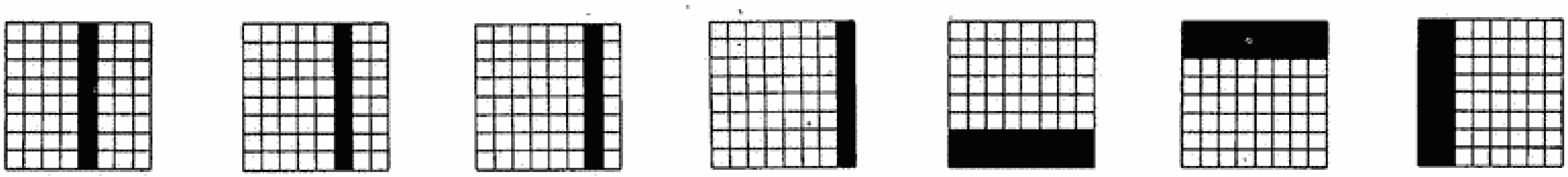
119 $77 | 12Ø $78 | 121 $79 | 122 $7A | 123 $7B | 124 $7C | 125 $7D



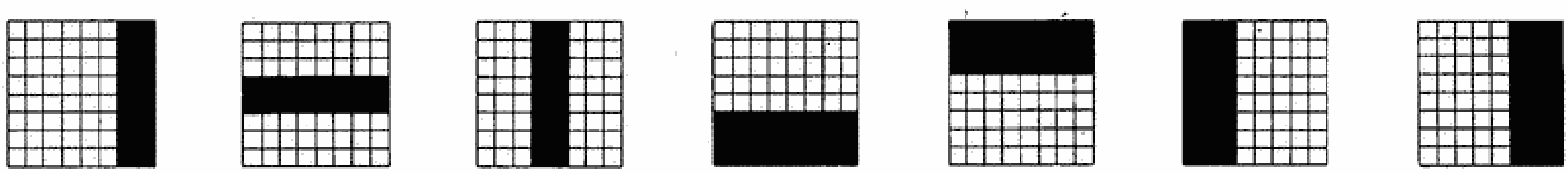
126 $7E | 127 $7F | 128 $8Ø | 129 $81 | 13Ø $82 | 131 $83 | 132 $84



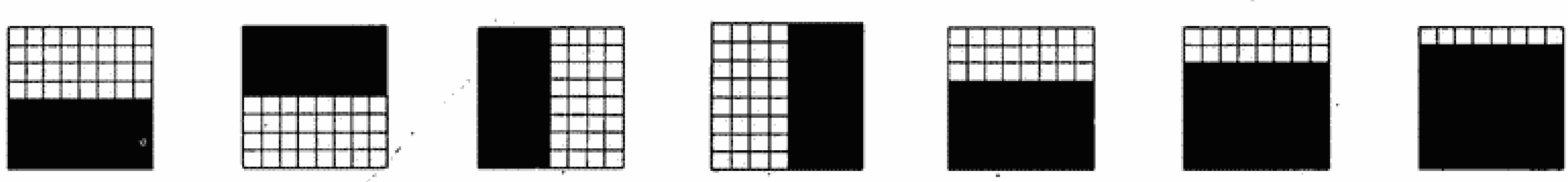
133 $85 | 134 $86 | 135 $87 | 136 $88 | 137 $89 | 138 $8A | 139 $8B



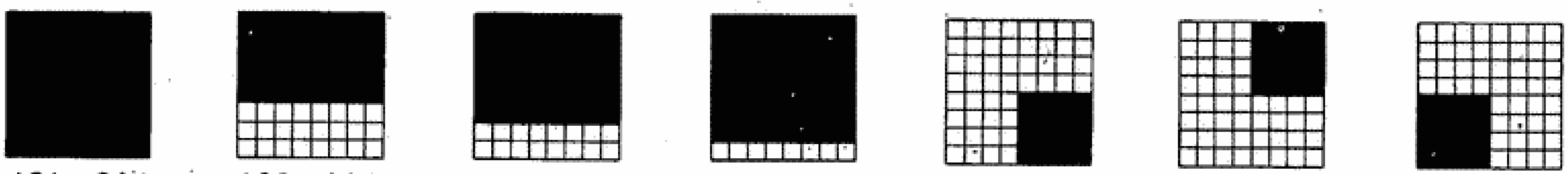
14Ø $8C | 141 $8D | 142 $8E | 143 $8F | 144 $9Ø | 145 $91 | 146 $92



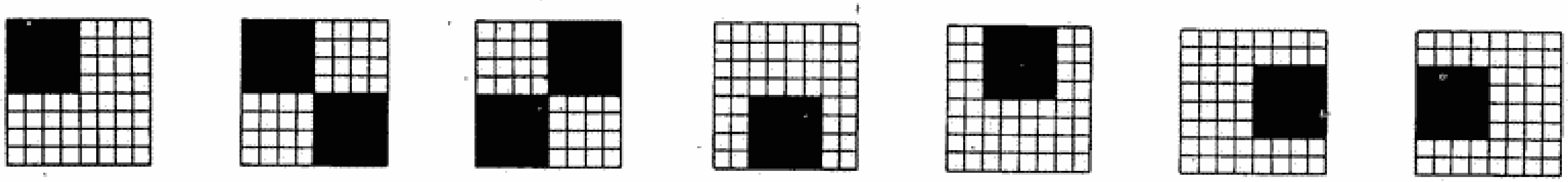
147 $93 | 148 $94 | 149 $95 | 15Ø $96 | 151 $97 | 152 $98 | 153 $99



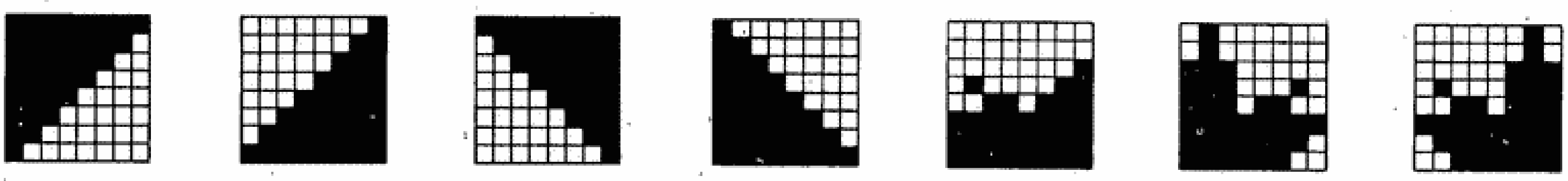
154 $9A | 155 $9B | 156 $9C | 157 $9D | 158 $9E | 159 $9F | 160 $AØ



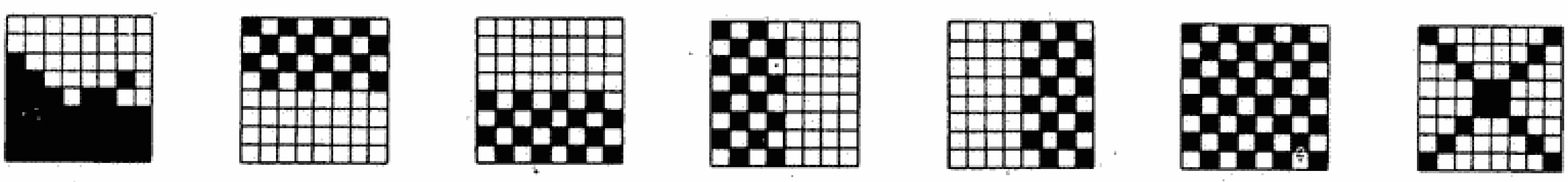
161 $A1	162 $A2	163 $A3	164 $A4	165 $A5	166 $A6	167 $A7



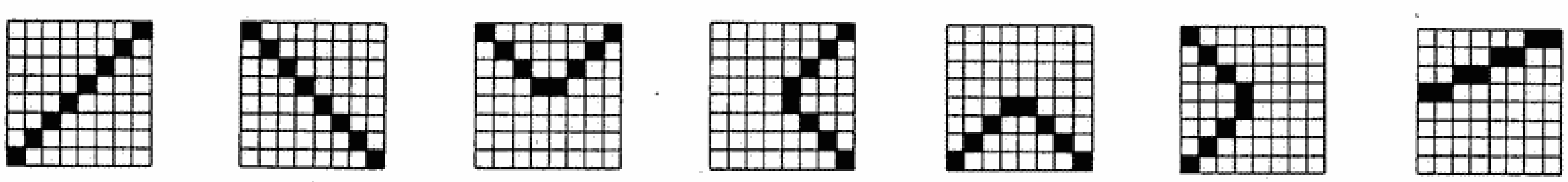
168 $A8	169 $A9	17Ø $AA	171 $AB	172 $AC	173 $AD	174 $AE



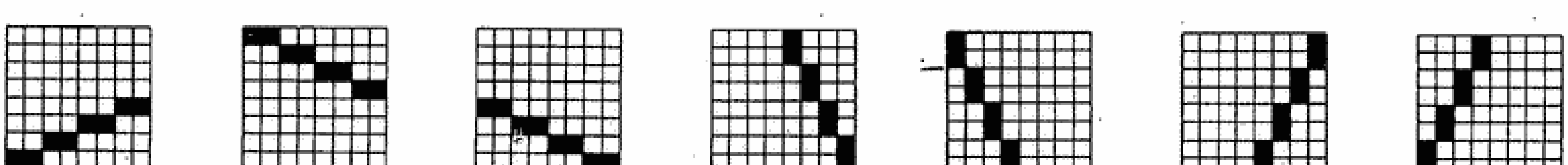
175 $AF	176 $BØ	177 $B1	178 $B2	179 $B3	18Ø $B4	181 $B5



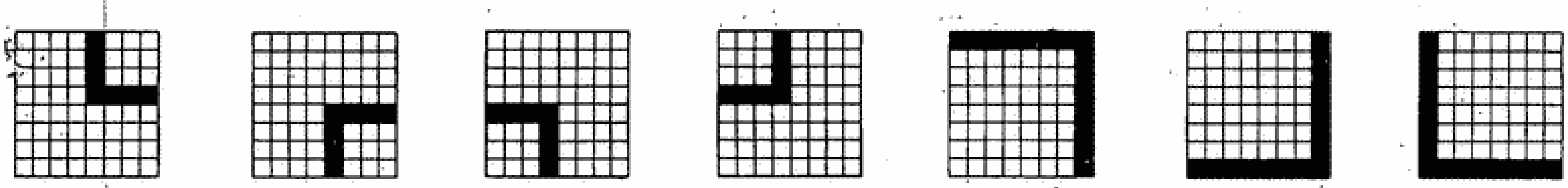
182 $B6	183 $B7	184 $B8	185 $B9	186 $BA	187 $BB	188 $BC



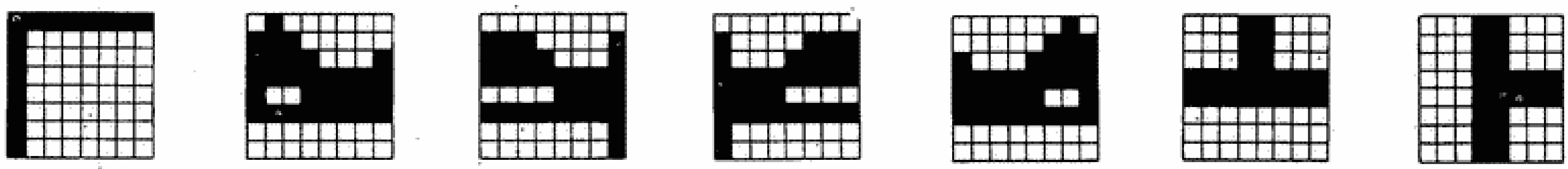
189 $BD	19Ø $BE	191 $BF	192 $CØ	193 $C1	194 $C2	195 $C3



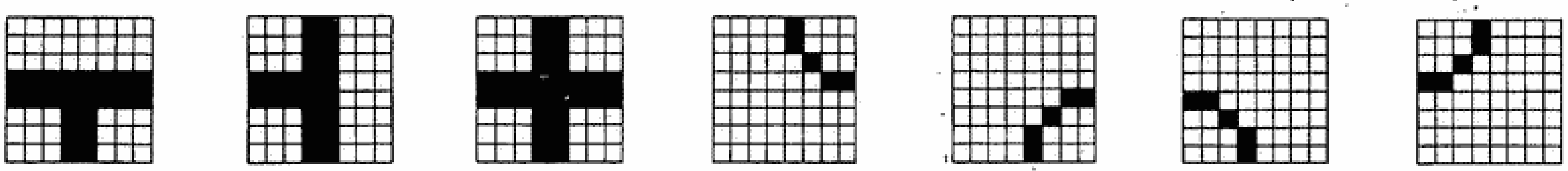
196 $C4	197 $C5	198 $C6	199 $C7	2ØØ $C8	2Ø1 $C9	2Ø2 $CA



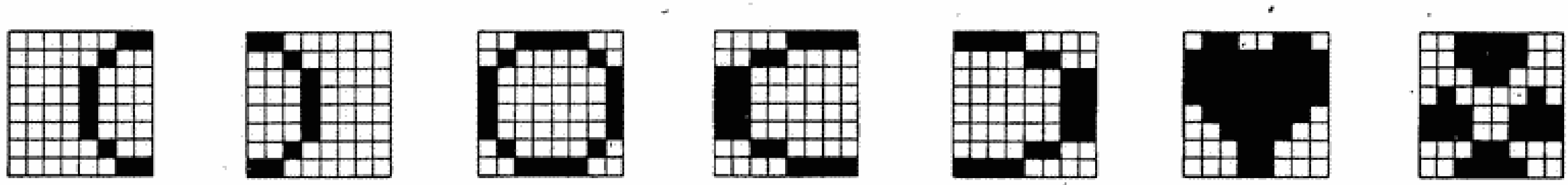
203 $CB | 204 $CC | 205 $CD | 206 $CE | 207 $CF | 208 $DØ | 209 $D1



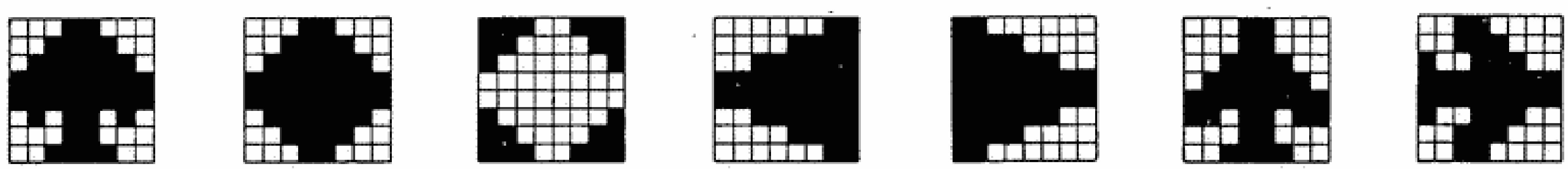
21Ø $D2 | 211 $D3 | 212 $D4 | 213 $D5 | 214 $D6 | 215 $D7 | 216 $D8



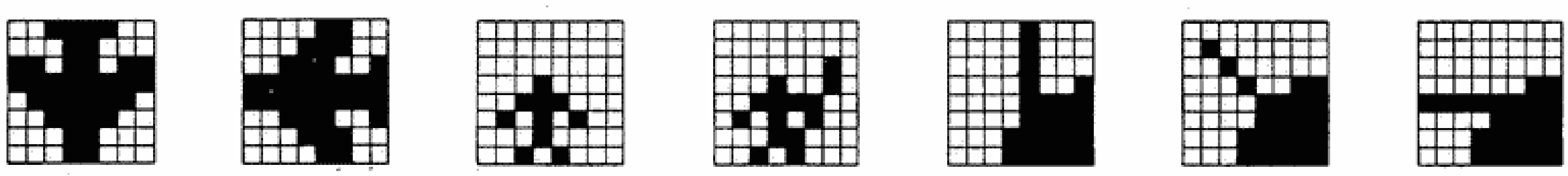
217 $D9 | 218 $DA | 219 $DB | 22Ø $DC | 221 $DD | 222 $DE | 223 $DF



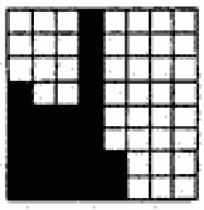
224 $EØ | 225 $E1 | 226 $E2 | 227 $E3 | 228 $E4 | 229 $E5 | 23Ø $E6



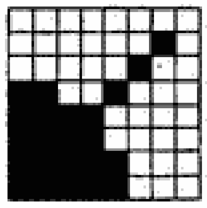
231 $E7 | 232 $E8 | 233 $E9 | 234 $EA | 235 $EB | 236 $EC | 237 $ED



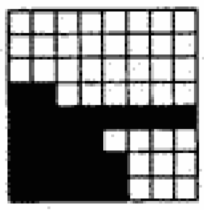
238 $EE | 239 $EF | 24Ø $FØ | 241 $F1 | 242 $F2 | 243 $F3 | 244 $F4



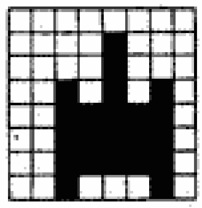
245 $F5



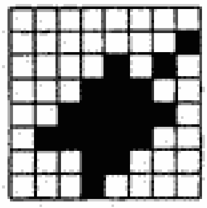
246 $F6



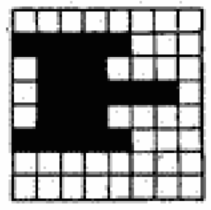
247 $F7



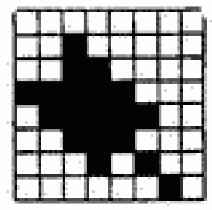
248 $F8



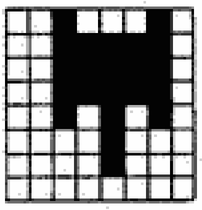
249 $F9



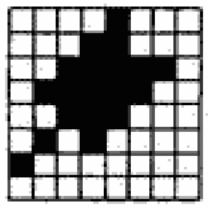
25Ø $FA



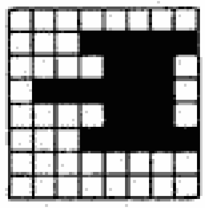
251 $FB



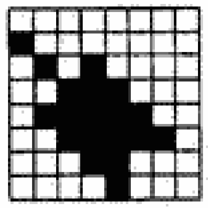
252 $FC



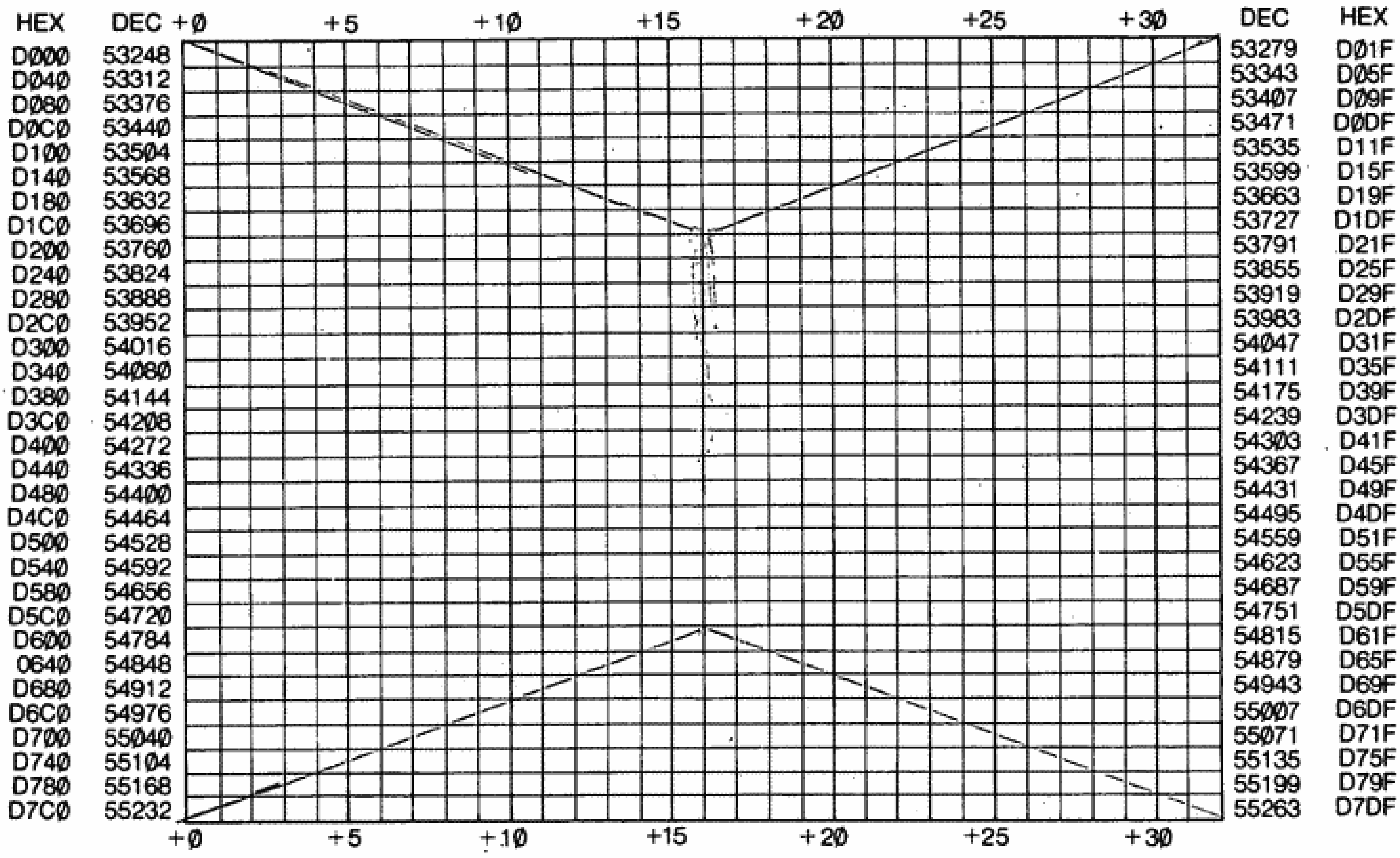
253 $FD



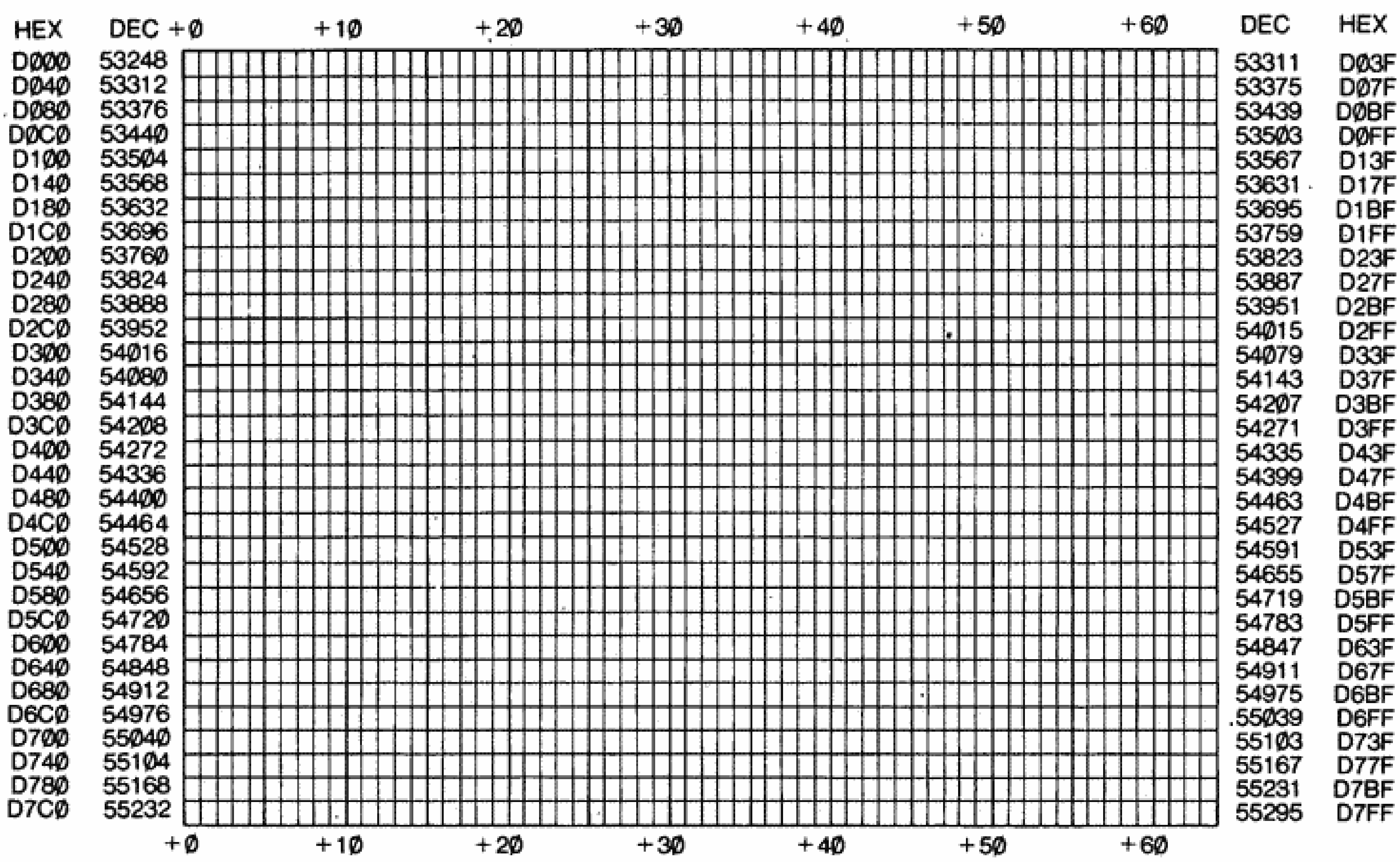
254 $FE



255 $FF



Video Map (32×32 Format)



Video Map (32×64 Format)

APPENDIX K

OS-65D USER'S GUIDE

This section is intended to be used as a quick reference guide only for complete details on OS-65D please refer to the OS-65D User's Manual.

COMMANDS

ASM	Load the assembler and extended monitor. Transfer control to the assembler.
BASIC	Load BASIC and transfer control to it.
CALL NNNN=TT,S	Load contents of track, "TT" sector, "S" to memory location "NNNN".
DIR NN	Print sector map directory of track "NN".
EM	Load the assembler and extended monitor. Transfer control to the extended monitor.
EXAM NNNN=TT	Examine track. Load entire track contents, including formatting formation, into location "NNNN".
GO NNNN	Transfer Control <GO> to location "NNNN".
HOME	Reset track count to zero and home the current drive's head to track zero.
INIT	Initialize the entire disk, i.e., erase the entire diskette (except track Ø) and write new formatting information on each track.
INIT TT	Same as "INIT", but only operates on track "TT".
IO NN,MM	Changes the input I/O distributor flag to "NN", and the output flag to "MM".
IO ,MM	Changes only the output flag.
IO NN	Changes only the input flag.
LOAD FILNAM	Loads named source file, "FILNAM" into memory.
LOAD TT	Loads source file into memory given starting track number "TT".
MEM NNNN,MMMM	Sets the memory I/O device input pointer to "NNNN", and the output pointer to "MMMM".
PUT FILNAM	Saves source file in memory on the named disk file "FILNAM."
PUT TT	Saves source file in memory on track "TT", and following tracks.
RET ASM	Restart the assembler.
RET BAS	Restart BASIC.
RET EM	Restart the Extended Monitor.
RET MON	Restart the Prom Monitor (via RST vector).
SAVE TT,S=NNNN/P	Save memory from location "NNNN" on track "TT" sector "S" for "P" pages.

SELECT X	Select disk drive, "X" where "X" can be, A, B, C, or D. Select enables the requested drive and homes the head to track Ø.
XQT FILNAM	Load the file, "FILNAM" as if it were a source file, and transfer control to location $327E.

NOTE:

—Only the first 2 characters are used in recognizing a command. The rest up to the blank are ignored.

—The line input buffer can only hold 18 characters including the return.

—The DOS can be reentered at 9543 ($2547).

—File names must start with an "A" to "Z" and can be only 6 characters long.

—The dictionary is always maintained on disk. This permits the interchange of diskettes.

—The following control keys are valid:

CONTROL — Q	continue output from a CONTROL-S
CONTROL — S	stop output to the console
CONTROL — U	delete entire line as input
BACKARROW	delete the last character typed.
SHIFT — O	delete the last character (polled keyboards)

ERROR NUMBERS

1—Can't read sector (parity error).

2—Can't write sector (reread error).

3—Track zero is write protected against that operation.

4—Diskette is write protected.

5—Seek error (track header doesn't match track).

6—Drive not ready.

7—Syntax error in command line.

8—Bad track number.

9—Can't find track header within one rev of diskette.

A—Can't find the sector before the one requested.

B—Bad sector length value.

C—Can't find that name in directory.

D—Read/Write attempted past end of named file!

MEMORY ALLOCATION

ØØØØ—22FF	BASIC or Assembler/Extended Monitor
22ØØ—22FE	Cold start initialization on boot
23ØØ—265B	Input/Output handlers
265C—2A4A	Floppy disk drivers
2A4B—2E78	OS-65D V3.Ø Operating system kernel
2E79—2F78	Directory buffer
2F79—3178	Page Ø/1 swap buffer

3179–3278	DOS extensions
3279–327D	Source file header
327E–	Source File

DISKETTE ALLOCATION

Ø–1	OS-65D V3. N bootstrap-loads to $22ØØ for 8 pages).
2–6	9-1/2 Digit Microsoft BASIC.
7–9	Assembler-Editor (if present)
10–11	Extended Monitor (if present)
12	Sector 1—Directory, page 1.
	Sector 2—Directory, page 2.
	Sector 3—BASIC overlays.
	Sector 4—GET/PUT overlays.
13	TrackØ/Copier utility (loads to $Ø2ØØ for 5 pages).
14–38	User programs and OS-65D utility BASIC programs.
39	Compare routine, on some disks only.

I/O FLAG BIT SETTINGS

INPUT:

Bit Ø—ACIA on CPU board (terminal).

Bit 1—Keyboard on 54Ø board.

Bit 2—UART on 55Ø board.

Bit 3—NULL.

Bit 4—Memory input (auto incrementing).

Bit 5—Memory buffered disk input.

Bit 6—Memory buffered disk input.

Bit 7—55Ø board ACIA input. As selected by index at location $2323 (8995 decimal).

OUTPUT:

Bit Ø—ACIA on CPU board (terminal).

Bit 1—Video output on 54Ø board.

Bit 2—UART on 55Ø board.

Bit 3—Line printer interface.

Bit 4—Memory output (auto incrementing).

Bit 5—Memory buffered disk output.

Bit 6—Memory buffered disk output.

Bit 7—55Ø board ACIA output. As selected by index.

9 DIGIT BASIC EXTENSIONS

INPUT # (DEVICE NUMBER) — (input is set to new device, output is set to null device. If device number > 3, null inputs are ignored.

INPUT "TEXT";# (DEVICE NUMBER) — (print "TEXT" at current output device, then function as above).

PRINT # (DEVICE NUMBER): — (print output for this command at new device).

LIST # (DEVICE NUMBER) — (list program or segments of program to new device).

WHERE (DEVICE NUMBER) FOR OUTPUT IS:

1—ACIA terminal

2—54Ø video terminal

3—55Ø ACIA UART port

4—Line printer

5—Memory output

6—Memory buffered disk output (bit 5).

7—Memory buffered disk output (bit 6).

8—55Ø ACIA output

9—Null output

(DEVICE NUMBER) FOR INPUT IS:

1—ACIA terminal

2—54Ø keyboard

3—55Ø ACIA UART port

4—Null device

5—Memory input

6—Memory buffered disk input (bit 5).

7—Memory buffered disk input (bit 6).

8—55Ø ACIA input

9—Null Input

EXIT — Exit to OS-65D V3. N

RUN (STRING) — Load and run file with name in (STRING).

DISK ! (STRING) — Send (STRING) to OS-65D V3. N as a command line.

DISK OPEN, (DEVICE), (STRING) — Open sequential access disk file with file name, (STRING) using memory buffered disk I/O distributor device number 6 or 7. Reads first track of file to memory and sets up the memory pointers to start of buffer.

DISK CLOSE, (DEVICE) — Forces a disk write of the current buffer contents to current track.

DISK GET, (RECORD NUMBER) — Using last file opened on the LUN (logical unit number) 6 device, a calculated track is read into memory. Where that track is: INT (REC.NUM)/24+base track given in last open command.

DISK PUT — It also sets both memory pointers to: 128*(REC. NUM.) −INT(REC. NUM.)/24))+base buffer address for LUN 6 device. Write device 6 buffer out to disk. The effect is the same as a "DISK CLOSE,6".

EXTENSIONS TO ASSEMBLER

For more details refer to the OSI Assembler Editor and Extended Monitor Reference Manual.

E	Exit to OS-65D V3.N
H(HEX NUM)	Set high memory limit to (HEX NUM).
M(HEX NUM)	Set memory offset for A3 assembly to (HEX NUM).
!(CMD LINE)	Send (CMD LINE) to OS-65D V3 as a command to be executed and then return to assembler.
CONTROL-I	Tab 8 spaces. Also: CONTROL-U 7 spaces. CONTROL-Y 6 spaces. CONTROL-T 5 spaces. CONTROL-R 4 spaces. CONTROL-E 3 spaces.
CONTROL-C	Abort current operation.

EXTENDED MONITOR

For more details refer to the OSI Assembler Editor and Extended Monitor Reference Manual.

!TEXT	Send "TEXT" to OS-65D V3 as a command.
@NNNN	Open memory location "NNNN" for examination. Subcommands: LF—Open next location. CR—Close location. DD—Place "DD" into location. "—Print ASCII value of location. /—Reopen location. Uparrow—Open previous location.
A	Print AC from breakpoint.
BN,LLLL	Place breakpoint "N" (1-8) at location, "LLLL".
C	Continue from last breakpoint.
DNNNN,MMMM	Dump memory from "NNNN" to "MMMM".
EN	Eliminate breakpoint "N".
EXIT	Exit to OS-65D V3. N
FNNNN,MMMM=DD	Fill memory from "NNNN" to "MMMM"−1 with "DD".
GNNNN	Transfer control to location "NNNN".

HNNNN,MMMM(OP)	Hexadecimal calculator prints result of "NNNN"(OP)"MMMM" where (OP) is + − * / .
I	Print break information for last breakpoint.
K	Print stack pointer from breakpoint.
L	Load memory from cassette.
MNNNN=MMMM,LLLL	Move memory block "MMMM" to "LLLL" −1 to location "NNNN" and up in memory.
NHEX)NNNN,MMMM	Search for string of bytes "HEX" (1-4) between memory location 'NNNN" and "MMMM"-1.
O	Print overflow/remainder from hex calculator.
P	Print processor status word from breakpoint.
QNNNN	Disassemble 23 lines from location "NNNN". A linefeed continues disassembly for 23 more.
RMMMM=NNNN,LLLL	Relocate "NNNN" to "LLLL"−1 to location "MMMM"
SMMMM,NNNN	Save memory block, "MMMM" to "NNNN"−1 on cassette.
T	Print breakpoint table.
V	View contents of cassette.
WTEXT)MMMM,NNNN	Search for ASCII string "TEXT" between "MMMM" and "NNNN"−1
X	Print X index register from last break.
Y	Print Y index register from last break.

NOTE: All commands are line buffered by OS-65D. Thus only 18 characters per line are allowed and CONTROL-U and BACKARROW apply.

SOURCE FILE FORMAT

RELATIVE DISK ADDRESS	MEMORY ADDRESS	USAGE
Ø	$3279	Source start (low)
1	$327A	Source start (high)
2	$327B	Source end (low)
3	$327C	Source end (high)
4	$327D	Number of tracks req.
5 and on . . .	$327 and on . .	Source text

DIRECTORY FORMAT

Two sectors (1 and 2) on track 12 hold the directory information. Each entry requires 8 bytes. Thus there are a total of 64 entries between the two sectors. The entries are formatted as follows:

Ø-5 ASCII 6 character name of file

6 BCD first track of file

7 BCD last track of file (included in file).

TRACK FORMATTING

The remaining tracks are formatted as follows:

- 1Ø millisecond delay after the index hole
- a 2 byte track start code, $43 $57
- BCD track number
- a track type code, always a $58

There can be any mixture of various length sectors hereafter. The total page count cannot exceed 8 pages if more than one sector is on any given track.

—Each sector is written in the following format:

- —previous sector length (4 if none before) times 8ØØ microseconds of delay
- —sector start code, $76
- —sector number in binary
- —sector length in binary
- —sector data

DISKETTE COPIER

The diskette copy utility is found on track 13, sector 1. It should be loaded into location 2ØØ with a "CA Ø2ØØ=13,1. To start it, type "GØ Ø2ØØ". To select the copier type a "1". Destination disks must be initialized prior to copying.

TRACK Ø READ/WRITE UTILITY

This utility permits the reading of data on track Ø anywhere into memory. Also the capability is available to write any block of memory to track Ø specifying a load address and page count. The track zero format is as follows:

- —1Ø millisecond delay after the index hole
- —the load address of the track in high-low form
- —the page count of how much data is on track zero

APPENDIX L

MACHINE MONITOR, 65V

The machine monitor provides a simple way to examine and modify memory contents. Data or programs are entered using hexadecimal (base 16) notation. Programs must be entered in machine code using hexadecimal notation. A thorough treatment of the Machine Monitor and its uses is found in OSI's 65V Primer.

The machine monitor provides a simple command structure. The machine monitor is entered after typing <BREAK> when the C4P gives the prompt

H/D/M?

Then type

M

The machine then responds with

ØØØØ XX

where XX are two hexadecimal characters. The computer is now in the machine monitor mode displaying the contents of location ØØØØ.

To load a given location (address) with data or program, type a period:

.

This will select the addressing mode. If the machine were already in the addressing mode, it will remain in the addressing mode. Now type the desired address. If an entry error is made, reentering the address will remove the old value.

To enter data into the selected memory location, A transfer to the data entry mode is required. This is done by typing a slash:

/

Data may now be entered as two hexadecimal characters. As in the address mode, an incorrect entry can be corrected by typing the correct value. To increment to the next sequential location, press

<RETURN>

Upon completion of loading, the program may be executed at its starting address (for illustration, hexadecimal address Ø2ØØ); type the starting address and then the Letter "G" as

.Ø2ØØG

(The period entry caused a return to the address mode.) The program will start executing. (The machine monitor goes to Ø2ØØ to start.)

ILLUSTRATION

Load a program which places grahics character 25Ø (hexadecimal FA) into mid video screen location 5432Ø (Hexadecimal D43Ø)—An assembly language program and its machine code would be

HEX LOCATION	MACHINE CODE	ASSEMBLY CODE	COMMENT
Ø2ØØ	A9	LDA #$FA	FA is symbol for eastward tank
Ø2Ø1	FA		
Ø2Ø2	8D		

HEX LOCATION	MACHINE CODE	ASSEMBLY CODE	COMMENT
Ø2Ø3	3Ø	STA $D43Ø	Tank to midscreen
Ø2Ø4	D4		
Ø2Ø5	EA	NOP	
Ø2Ø6	4C	JMP $Ø2Ø5	Jump back to NOP
Ø2Ø7	Ø5		
Ø2Ø8	Ø2		

This program should place an eastward point tank (character 25Ø) near mid video screen. The machine monitor instructions would be

<BREAK>

.Ø2ØØ

/A9 <RETURN>

FA <RETURN>

8D <RETURN>

3Ø <RETURN>

D4 <RETURN>

EA <RETURN>

4C <RETURN>

Ø5 <RETURN>

Ø2 <RETURN>

.Ø2ØØG

At this point, the tank should appear mid video screen.

For the cassette user, the command L permits loading program from cassette. Upon typing L, all ASCII commands are accepted from the audio cassette rather than the keyboard. Cassettes are prepared with a auto-loading program at their beginning. Examples of this are the Extended Machine Code Monitor cassette and the Assembler/Editor cassette. When the program is loaded, the cassette playback unit may be rewound and turned off.

In summary, the Machine Monitor commands are

/—Use Data Mode

.—Use Address Mode

G—Start execution at the address presently displayed on video screen.

L—Transfer control to the audio casette.

Some of the hexadecimal locations which the Machine Monitor uses are

FEØØ—Start of Monitor (restart location)

FEØC—Restart with clear video screen, other Machine Monitor parameters unchanged

FE43—Entry into Address Mode, with initialization bypassed

FE77—Entry into Data Mode, with initialization bypassed

These entry points may be useful to incorporate into other programs.

A more comprehensive discussion of the 65V Monitor is included in the 65V Primer, OSI's introduction to 6502 assembler coding.

APPENDIX M

USR(X) ROUTINE

The speed of machine code execution can be combined with the simplicity of BASIC by using the USR(X) function. The linking of machine code and BASIC programs is accomplished by the single BASIC statement

X=USR(X)

The USR(X) function permits leaving the BASIC program, executing a machine language routine, and then returning to the original BASIC program. To call the USR(X) routine in BASIC, a pointer to the location of the USR(X) routine must have been stored. In disk BASIC, these pointers are at 22FC hexadecimal (8956 decimal) for the low half of the hexadecimal address and 22FB hexadecimal (8955 decimal) for the high half of the hexadecimal address.

Cassette based C-4P systems, using BASIC-in-ROM, use ØØØB hexadecimal (11 decimal) and ØØØC hexadecimal (12 decimal) to store the low and high half of the USR(X) routine address, respectively.

Typically, the operator will want to protect the machine language (code) program by placing it in high memory. If BASIC's "end of memory" pointer is moved to a value at least two pages (512 decimal words) down from the physical value of "end of memory," this memory area can be saved from use by any other routine. For example, on a 24K system (24576 decimal, 6ØØØ hex) these limits would be

```
24576
- 512
24Ø64
```

The equivalent calculation in hex is

```
6ØØØ
-2ØØ
5EØØ
```

Therefore, setting 5EØØ hex as "end of memory" will give a 512 byte clear region for calculations. This "end of memory" value should be stored with the high order two hex digits in location 23ØØ hex (896Ø decimal) i.e., POKE 896Ø,94.

Since the "end of memory" value will need to be stored with a POKE command in BASIC, first convert 5EØØ hex to 94ØØ decimal.

```
5E    ØØ    hexadecimal
94    ØØ    decimal
```

Since the address of end of memory requires two bytes for storage, two POKEs are necessary. The POKE command requires decimal values as operands. Therefore, each half of the hex address must be converted into decimal, one half at a time. Conversion was accomplished by looking up the decimal conversion in the table provided in the appendix. The high order hex equivalent digits are stored by

```
POKE    896Ø,                94
end of memory pointer     high memory boundary
```

The lower half of the "end of memory" is assumed at the page end (ØØ).

Next, choose the lower end of this now protected memory (above the official "end of memory") to store the USR(X) routine. Place the address of USR(X) in the location pointer to where BASIC expects the USR(X) address. The address of USR(X) can be loaded by using POKEs. The two address parts of USR(X) can be POKEd into the location which stores USR(X)'s address by

POKE 8955,ØØ = REM—LOW BYTE OF USR(X) ADDRESS

POKE 8956,94 = REM—HI BYTE OF USR(X) ADDRESS

REM INTO USR(X) POINTER

Now a program, USR(X), needs to be written to be stored in memory starting at 5EØØ hex (24Ø64 decimal). Please note that this last decimal value is the result of converting all four hex digits of 5EØØ at one time, rather than finding the decimal equivalent of each half of the address. The earlier conversions of half of the address were for storage convenience, and were not for evaluating the whole address value.

EXAMPLE: A SCREEN CLEARING ROUTINE

To illustrate the USR(X) routine, a routine to clear the CRT terminal screen will be written. The letter "A" will be placed at each screen position, sequentially to illustrate the speed of this routine. Of course, replacing the letter "A" with the symbol for a blank would produce a general screen clearing. This program is described by a flow chart in Fig. 29 which is reduced to assembly language in Fig. 28 and restated without comments to show sequential locations in Fig. 30. In this example, the last statement is an RTS (return from subroutine), which returns from the subroutine to the calling BASIC program.

In the example, the 65Ø2 microprocessor's accumulator will be used as the register for data transfer. The X-register and the Y-register will be used as counter registers. This usage will be economical in terms of data transfer time, since the accumulator is the central point for transfer purposes. The X- and Y-registers are serviced with increment and decrement commands to aid counting operations.

HEX LOCATION	DECIMAL LOCATION	MACHINE CODE	ASSEMBLER CODE	COMMENT
			*=$5EØØ	Set program counter on 5EØØ
5EØØ	24Ø64	A9 41—	LDA #$41	Load accumulator with ASCII A
5EØ2	24Ø66	AØ Ø8— (Ø4 FOR VIC 20)		Load page count
5EØ4	24Ø68	A2 ØØ—	LDX #$ØØ	Load column counter at zero
5EØ6	24Ø7Ø	9D ØØ DØ	STA $DØØØ,X (1EØØ FOR VIC 20)	Store "A" at each screen position
5EØ9	24Ø73	E8— —	INX	Increment column on screen
5EØA	24Ø74	DØ FA—	BNG $5EØ6 (*+6)	If columns not complete, loop to store "A" again
5EØC	24Ø76	EE Ø8 5E	INC $5EØ8 (*+8)	If columns complete, increment page (4 line) counter
5EØF	24Ø79	88— —	DEY	Decrement page count
5E1Ø	24Ø8Ø	DØ F4	BNG $5EØ6 (*+6)	If not complete page count, loop to store "A" again
5E12	24Ø82	A9 DØ—	LDA #$DØ (1E FOR VIC 20)	If page complete, then reset screen address
5E14	24Ø84	8D Ø8 5E	STA $5EØ8 (*+8)	Restore operand of page count
5E17	24Ø87	6Ø— —	RTS	Go back to calling program

Fig. 28 Screen Clearing Assembly Language

* = STARTING ADDRESS

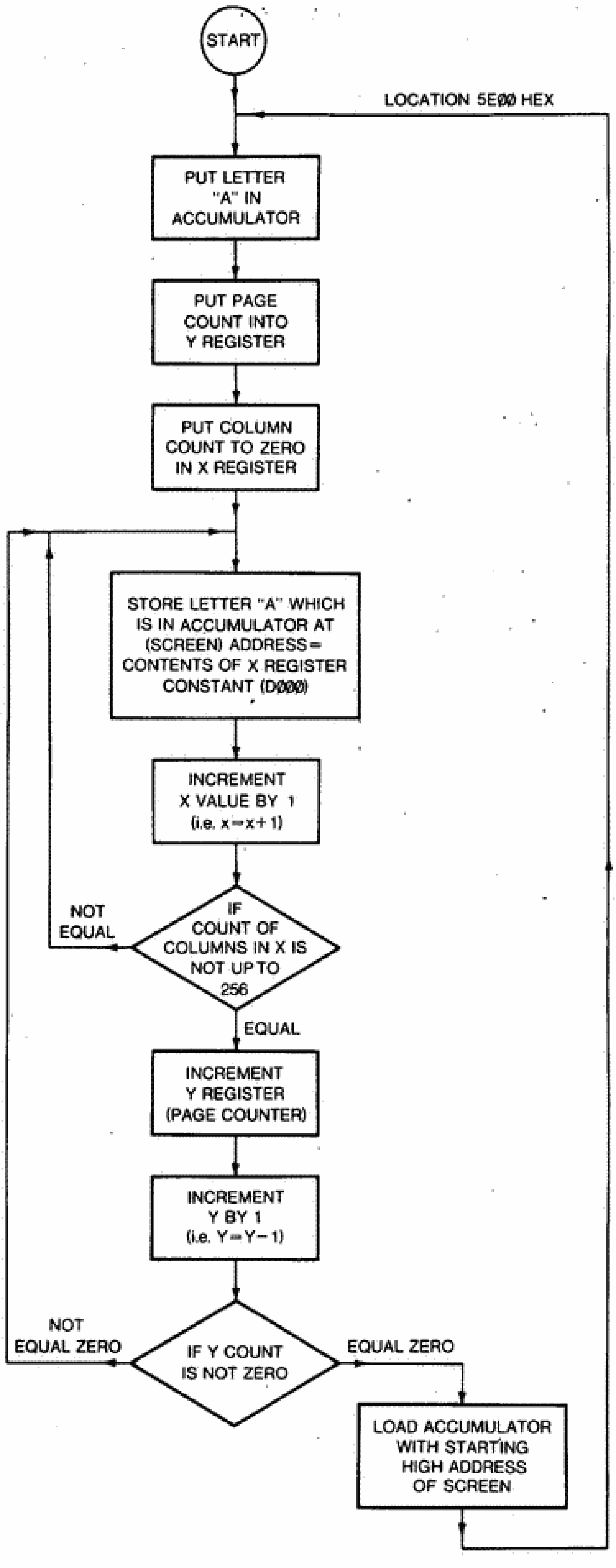


Fig. 29 Flow Chart (Screen Clearing Routine)

HEX LOCATION	DECIMAL LOCATION	MACHINE CODE (HEXADECIMAL)	MACHINE CODE (DECIMAL)
5E00	24064	A9	169
5E01	24065	41	65
5E02	24066	A0	160
5E03	24067	08	8
5E04	24068	A2	162
5E05	24069	00	0
5E06	24070	9D	157
5E07	24071	00	0
5E08	24072	D0	208
5E09	24073	E8	232
5E0A	24074	D0	208
5E0B	24075	FA	250
5E0C	24076	EE	238
5E0D	24077	08	8
5E0E	24078	5E	94
5E0F	24079	88	136
5E10	24080	D0	208
5E11	24081	F4	244
5E12	24082	A9	169
5E13	24083	D0	208
5E14	24084	8D	141
5E15	24085	08	8
5E16	24086	5E	94
5E17	24087	60	96

Fig. 30 Screen Clearing Assembly Language Showing Sequential Locations

By converting the hexadecimal machine code into decimal values, the code can be POKEd into the desired memory locations. This is a handy method to enter machine code routines while in BASIC. A BASIC program to store this machine code at the required locations is

```
5 REM CLEAR SCREEN PROGRAM
10 RESTORE : REM SETS START OF DATA LIST
20 P=24064 : REM START AT 5E00 HEX
30 FOR I=1 TO 24
40 READ M : POKE P,M
50 P=P+I
60 NEXT I
70 DATA 169,65,160,8,162
80 DATA 0,157,0,208,232
90 DATA 208,250,238,8,94
100 DATA 136,208,244,169,208
110 DATA 141,8,94,96
120 END
RUN <RETURN>
```

Running this program places the desired machine code routine in memory. Now exit from BASIC by typing

EXIT <RETURN>

At this time, the machine code routine can be SAVEd in high memory on disk. For example, use track 39 of the disk, starting at sector 1, and respond to the prompt

A*

with

SAVE 39,1=5E00/1 <RETURN>

This saves the program located at 5E00 hexadecimal, starting on track 39 at sector 1 for 1 page (256 bytes). This program can be reloaded from disk by responding to the prompt

A*

with

CALL 5E00=39,1

The machine code routine would thus be read off track 39, sector 1 into RAM at 5E00. This screen clearing routine may be run as follows, reloading the program under BASIC. This reloading under BASIC may be done by typing

DISK!"CALL 5E00=39,1"

Therefore the BASIC program segment

```
:
90 POKE 8955,0 : POKE 8956,94: REM SET USR(X) ENTRY POINT
100 DISK!"CALL 5E00=39,1" : REM USR(X) STORED EARLY IN PROGRAM
:
1000 X=USR(X): REM SCREEN CLEARING ROUTINE INVOKED
```

including USR(X), would provide a screen clear at a far faster rate than possible with the BASIC program.

An additional feature of USR(X) is the ability to transfer parameters between a BASIC program and a machine language program.

PASSING PARAMETERS

The machine language routine begins by calling a routine the starting address of which is a $0006. This routine converts the argument X into a 16 bit two's complement number which is then stored. The storage location of this number depends upon the BASIC used, as follows:

HIGH BYTE	LOW BYTE	BASIC USED
$00AE	$00AF	ROM BASIC
$00B1	$00B2	65D

The value of X is now available for the machine language routine.

The machine language routine ends by placing the value to be returned to the BASIC program in the accumulator (high byte) and the Y register (low byte); then calling a subroutine that starts at $0008. This subroutine will pass the value to the BASIC program as USR(X) and then return control to the BASIC program.

EXAMPLE

An example is given in this section of a program in 65D BASIC and a machine language routine that are linked by and have parameters passed by the USR function. In the example, the argument of the USR function is an integer H between Ø and 255. The value of H is passed to the machine language routine which then returns as USR(H) the number of times the character whose ASCII value is H appears on the video screen.

The BASIC program:

```
1Ø POKE 574,Ø
2Ø POKE 575,64
3Ø INPUT "ENTER CHARACTER";A$
4Ø H=ASC(A$)
5Ø N=USR(H)
6Ø PRINT N
7Ø END
```

The machine language routine:

```
1Ø                          ;passing parameters to USR function
2Ø                          ;K=USR(C)
3Ø                          ;C=character number Ø<=C<=255
4Ø                          ;K=count of how many times the character
5Ø                          ;  appears on the screen 6Ø    ;
7Ø   3FFC                   *=$3FFC
8Ø   3FFC 6CØ6ØØ  CALL      JMP (6)
9Ø
1ØØ  4ØØØ                   JSR CALL
11Ø  4ØØØ 2ØFC3F  START     JSR CALL          integerize C
17Ø  4ØØ3 A5B2              LDA $B2           the result
18Ø  4ØØ5 A2DØ              LDX #$DØ
19Ø  4ØØ7 8E194Ø            STX COMP+2        screen addr (hi)
2ØØ  4ØØA A2ØØ              LDX #Ø
21Ø  4ØØC 8E184Ø            STX Comp+1        screen addr (lo)
22Ø  4ØØF 8E364Ø            STX COUNT
23Ø  4Ø12 8E374Ø            STX COUNT +1      initialize counter
24Ø  4Ø15 AØØ8              LDY #8            this many pages per screen
25Ø  4Ø17 DDFFFF  COMP      CMP $FFFF,X       dummy address
26Ø  4Ø1A DØØ8              BNE END
27Ø  4Ø1C EE374Ø            INC COUNT+1       count it
28Ø  4Ø1F DØØ3              BNE END
29Ø  4Ø21 EE364Ø            INC COUNT         do this if lo half rolls over
```

```
300   4024 E8     END     INX
310   4025 D0F0           BNE COMP
320   4027 EE1940         INC COMP+2
330   402A 88             DEY
340   402B D0EA           BNE COMP
350   402D AD3640         LDA COUNT
360   4030 AC3740         LDY COUNT+1
370   4033 6C0800         JMP (8)
380   4036 00     COUNT   BYTE 0,0
380   4041 00
```

These two programs can be combined into the following one; the machine language routine is directly POKED into memory after converting each hex instruction to its decimal equivalent.

```
  2 FOR I=0 TO 2
  4 READ V
  6 POKE 16380+I,V
  8 NEXT
 10 FOR I=0 TO 55
 20 READ V
 30 POKE 16384+I,V
 40 NEXT
 50 POKE 574,0
 60 POKE 575,64
 70 INPUT"ENTER CHARACTER";A$
 80 H=ASC(A$)
 90 N=USR(H)
100 PRINT N
110 DATA 108,6,0
120 DATA 32,252,63,165,178,162,208
130 DATA 142,25,64,162,0,142,24,64
140 DATA 142,54,64,142,55,64,160,8
150 DATA 221,255,255,208,8,238,55
160 DATA 64,208,3,238,54,64,232,208
170 DATA 240,238,25,64,136,208,234
180 DATA 173,54,64,172,55,64
190 DATA 108,8,0,0,0
```

USING THE ASSEMBLER

The preceding USR(X) program was shown in Assembly language. The C4P system supports an assembler. The Assembler/Editor could have been used for creating the program module which was SAVEd on disk.

To use the Assembler/Editor, boot up the system. Once in BASIC, request (after the OK prompt)

```
EXIT <RETURN>
```

Type (after the operating system prompts, shown underlined)

```
A* ASM <RETURN>
```

to get the Assembler, and enter the program (the same USR(X) program as before) after the Assembler prompt.

```
.1Ø *=$5EØØ
.2Ø LDA #$41
.3Ø LDY #$Ø8
.4Ø LDX #$ØØ
.5Ø STA $DØØØ,X
.6Ø INX
.7Ø BNE $5EØ6
.8Ø INC $5EØ8
.9Ø DEY
.1ØØ BNE $5EØ6
.11Ø LDA #$DØ
.12Ø STA $5EØ8
.13Ø RTS
.A
```

The Assembler file will assemble the program and store it at 5EØØ hexadecimal (24Ø64 decimal). The machine code program has again been stored in memory at 5EØØ hexadecimal.

At this point, the use of the operating system to SAVE the program on disk would be the same as shown in the previous section, i.e., typing

```
SAVE 39,1=5EØØ1 <RETURN>
```

would place the machine code on disk. Running the previous BASIC program segment

```
90 POKE 8955,Ø : POKE 8956,94
100 DISK!"CALL 5EØØ=39,1"
1ØØØ X=USR(X)
RUN
```

will result in the same screen clearing routine to be run.

The Assembly language listing provided the machine code needed for the USR(X) loading. Even if the Assembler is not used to create the USR(X) program module, the extensive editing routines of the Assembler/Editor encourage its use.

Note, for more detail on the Assembler/Editor see the Ohio Scientific Assembler/Editor Manual.

Finally, an often used USR(X) routine to color the video background is given. This illustrates the brevity and simplicity of USR(X).

Example: Color Background

This BASIC program sets up an ASSEMBLER subroutine under the USR(X) function. The subroutine changes the background color of the entire screen. Note, if a disk system is not used then the BASIC code; DISK!"CA 4FDO=36,1"; must be removed from the program.

To save the assembler program (created by this BASIC program) on disk, type DISK!"SA 36,1=4FDO/1" after running the program. This will allow the calling of the program from disk in any other BASIC program by the command DISK!"CA 4FDO=36,1" instead of running this BASIC code.

Use the following code in BASIC (after the assembler program is loaded into memory) to execute the assembler routine. NOTE: this must be done after the subroutine is in memory.

```
POKE8955,2Ø8:POKE8956,79
```

This is the high and low addresses to tell the computer where the USR(X) function is located in memory.

```
POKE2Ø433,(choice, Ø-16)
```

This is choice of color background.

```
X=USR(X)
```

This is the BASIC command for jumping to an assembler subroutine specified by the previous POKEs.

```
1ØØ FOR I=2Ø432TO2Ø473:READ X:POXE I,X:NEXT
2ØØ DATA162,14,169,Ø,141,242,79,169,224,141,243,79,173,242,79
21Ø DATA24,1Ø5,1,141,242,79,173,243,79,1Ø5,Ø,141,243,79,2Ø1,232
22Ø DATA24Ø,6,142,Ø,224,76,22Ø,79,96,Ø,2
```

Use of this code or the method should increase the versatility of the computer, both in the speed of its response and the ease of use.

APPENDIX N

EXECUTING A DISK RESIDENT MACHINE LANGUAGE PROGRAM

To access a desired machine language program, there is an alternative to use of the BASIC routine

X=USR(X)

Assume there is a machine code program stored on a disk file named "FILE." The alternate method is used under the DOS. The response should be

A* XQT FILE <RETURN>

where FILE is the name of the machine language program on disk (or it can be the track number where it is stored).

Under BASIC, this is accomplished by

DISK!"XQT FILE"

In order to use the XQT command, however, some computer housekeeping is required first.

The XQT command brings a machine code program from disk and stores it at location 12921 decimal (3279 hexadecimal). When the machine code is stored on disk, some housekeeping is done. The first four bytes on the file used will contain a "header" which is labeling information provided by the assembler. The next (fifth) byte will contain how many tracks are to be loaded to contain the program. Then, from the sixth byte to the end of the file, the machine code program is stored.

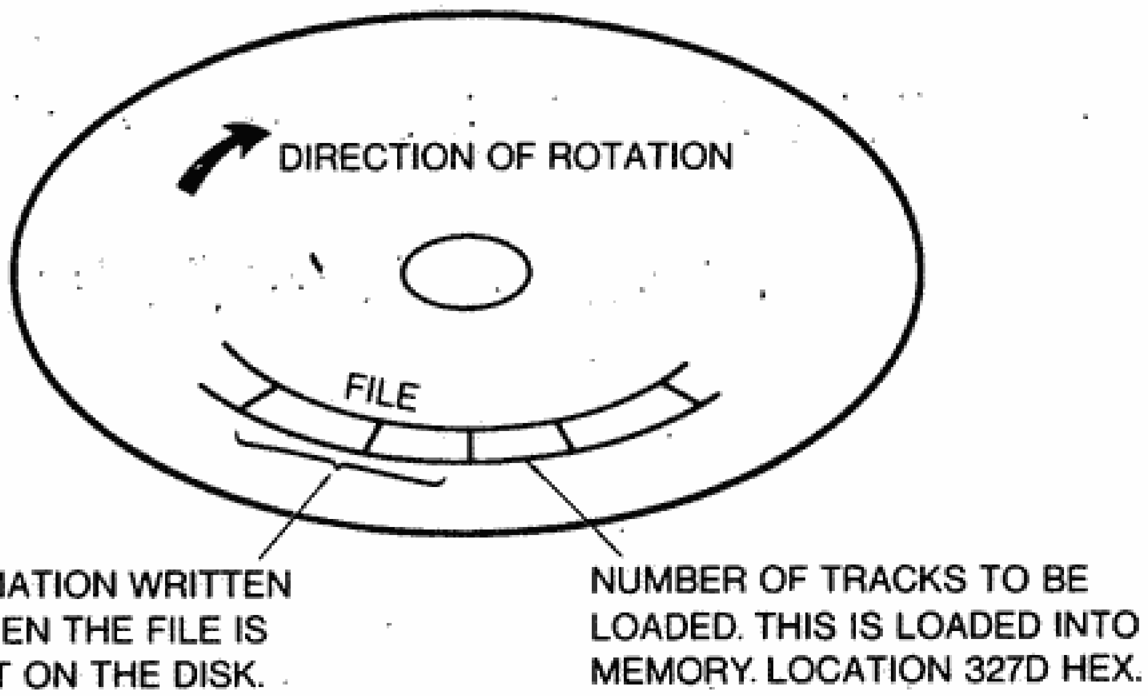
When a machine code file is loaded by the XQT command into memory starting at 12921 decimal (3279 hexadecimal), program control will have to skip over the header and track length information and start with the program stored at 12926 decimal (327E hexadecimal).

The following is a map of how the program is expected to appear on disk. Also, a map of how the file will be stored in memory.

XQT FILE STORAGE IN MEMORY

DECIMAL LOCATION	HEX LOCATION	CONTENTS
12921	3279	
12922	327A	
12923	327B	File header created by Assembler
12924	327C	
12925	327D	Number of disk tracks to be loaded
12926	327E	Start of first program instruction
12927	327F	

XQT FILE STORAGE ON DISK



With the housekeeping conventions established, start by creating a file called FILE1 which will contain an assembly language code. This program has not been converted into machine code yet. The program shown will store the message "ANY ASCII CHARACTERS" at locations D74Ø hexadecimal (551Ø4 decimal) which is in the lower left hand side of the video screen. Enter the program as follows

A* ASM <RETURN>

The computer will reply

OSI 65Ø2 ASSEMBLER

COPYRIGHT 1976 BY OSI

Then enter the assembly language code.

```
.1Ø           *=$327E                         { SET ORIGIN
.2Ø           LDX #Ø                          { SYMBOL COUNTER INITIALIZED
.3Ø LBL1      LDA MSG,X
.4Ø           BEQ LBL2
.5Ø           STA $D74Ø,X
.6Ø           INX
.7Ø           BNE LBL1
.8Ø LBL2      JMP LBL2
.9Ø MSG       .BYTE 'ANY ASCII CHARACTERS'
.1ØØ          .BYTE Ø
.11Ø          .END
```

This can be stored in the previously created file—FILE1—by typing

!PUT FILE1

When this file is already on disk it could be recalled by typing

!LOAD FILE1

In either case, the source program is not yet ready to be assembled, i.e., converted into machine code. When it is converted to machine code, the assembled (converted machine code) program will be stored at a location (address)

2ØØØ hexadecimal bytes displaced from the assembly language program. A memory displacement or offset, arbitrarily chosen here as 2ØØØ hexadecimal (valid for 24K machines), must be established in order to be within memory available and above the region needed by the assembler program, by typing

M2ØØØ

and then

A3

The Assembler/Editor will now assemble the program and leave it at a location offset by 2ØØØ hexadecimal from the intended program origin. Now exit the assembler by typing

EXIT

The assembled (machine code) program should now be placed at the final destination of 327E hexadecimal, which is where the XQT command will place the first machine code program step. The Extended Monitor provides the means of relocating the program from location offset by 2ØØØ hex above the destination of 327E. The previously used region (327E hex and up) is no longer needed by the Assembler/Editor.

To invoke the extended monitor from the DOS type

EM

The extended monitor prompt is a colon. Type

: M 327E=527E,5298

The difference between the first two numbers is the offset value previously used. The last number is one more than the last memory location required, all in hexadecimal. The Assembler/Editor provides the address of each instruction in the listing. By subtracting the last address from the first address in the listing, the hexadecimal length of the machine code (not including the last instruction) can be calculated. Shorter programs, of course, would require less memory.

The integer number of tracks to store the machine code program needs to be determined. Each disk track can store 2K bytes of code (length of approximately 2ØØØ decimal).

Since the example is 19 hexadecimal in length (25 decimal), far less than one track is required (even if the five locations needed for the header are added). The information about the track requirement is put in location 327D by responding to the colon prompt by

: @327D

The @ symbol is <SHIFT P>. The Extended Monitor permits the storing of data in 327D following the prompt

327D/Ø1

Reply with

Ø1

the number (two hexadecimal digits) of tracks required. The next response is

: EXIT <RETURN>

In earlier examples in the manual, files (called scratch files) were created for incidental use. Now is the time to use one of those files named "SCRTCH" to store the machine code program. This machine code program is stored by responding to the prompt

A* with PUT FILE2

The XQT command can now be verified by responding to the prompt

A* with XQT FILE2

The message "ANY ASCII CHARACTERS" should appear on the screen.

The details of this section have been rather involved. By using machine code, the housekeeping responsibilities within the computer have had to be accepted. In return, considerably faster running programs are obtained. Storage requirements of the programs are also reduced. If the speed and compactness of machine code is needed within the convenience of BASIC programming, the XQT command may prove worth the demands on the user.

APPENDIX O

INDIRECT FILES

The indirect file is an uncommonly powerful mechanism to manipulate and combine separate programs.

The need for the indirect file arises from two characteristics of the operating system. First, in order to do editing, the editor needs to know where a given statement resides in memory. When Assembly language programs are stored, a somewhat compressed form (tokenized) is used to save memory. This makes it difficult to know where a given statement is located in memory. Second, in order to load two BASIC programs (assumed to have compatible statement numbers), i.e., the same statement numbers haven't been used in both programs, the operating system would wipe out the first program when it loaded the second.

These potential problems encourage the placing of the ASCII coded text sequentially into a single file in memory (similar to a file on disk). Also, it is desirable to be able to keep the two loaded modules (programs) together, contiguous, without garbage in between. The disk file handling routines do not give the fine control that the indirect file does. In an indirect file, the individual characters can be pointed to in a string of text. For these reasons, indirect file handling has been developed under the OS-65D V3.N system. The indirect file provides a method of temporarily storing ASCII code.

The indirect file is stored in high memory. The address of the indirect file is stored in 9554 (high byte only). The low half of the indirect file address is assumed to be Ø. For a 24K system, the POKE to store the high address byte is

```
POKE 9554,8Ø
```

The high byte of the indirect file address, for different memory configurations is

Memory Size	POKE 9554 with Decimal
24K	8Ø
32K	96
4ØK	112
48K	128

These suggested memory allocations provide a balance between indirect file size and available user work space. In a 24K system, this allocation of memory allows 4K bytes for the indirect files. Additionally, the indirect file input address must be POKEd at location 9368 with the same table value. For a 24K system this is

```
POKE 9368,8Ø
```

FIRST EXAMPLE: COMBINING TWO PROGRAMS

The goal is to take the first of two programs and temporarily store it in the indirect file. Then it will be desired to enter a second program into the BASIC work space, but the LOAD command normally causes overwriting of the first program.

In order to avoid overwriting of one program by another, indirect files allow the use of the steps:

1. clean out the work space by typing

   ```
   NEW
   ```

2. enter a program from the keyboard or a disk file
3. store the newly entered program in an indirect file
4. clear the work space again. This time, it is done only to illustrate that the old program is removed.
5. enter a new program (with statement numbers that do not conflict with the first program).
6. bring the indirect file back into the work space. Now both programs are in the work space and have been merged together.

Now to apply these steps in a short example.
The commands to combine two short programs would be

POKE 9554,8Ø : REM SET INDIRECT FILE OUTPUT FOR 24K SYSTEM

POKE 9368,8Ø : REM INDIRECT FILE INPUT FOR 24K SYSTEM

The first program is then typed

1Ø PRINT"TEST1" : REM SHORT EXAMPLE!

The program is transferred to indirect file by typing

LIST <SHIFT K> <RETURN> Note: at the same time pressing
<SHIFT K>=[

The listing will appear on the video screen and the program will be transferred to the indirect file in upper memory.
Now close the indirect file by typing

SHIFT M<RETURN> Note: at the same time pressing
<SHIFT M>=]

The symbols

]]

will be displayed, along with an error message

?SN ERROR

which should be ignored.
Typing

NEW

will assure that the program is removed from the BASIC work space.
Now enter the second program

2Ø PRINT"TEST2"

The command

LIST

will assure that only statement 2Ø is in the work space.
Typing

<CONTROL X>

will transfer the indirect file back into the work space. Either the RUN command or the LIST command shows that both programs are now resident in the BASIC work space.

This example has been extremely short. Be cautioned that a large program in the BASIC work space could overwrite the indirect file.

SECOND EXAMPLE: CREATING A BUFFER FOR A BUFFERLESS PROGRAM

This example illustrates adding a buffer to a previously written program which lacked a necessary buffer. The original program could be loaded from its file, say FILE1, by

DISK!"LO FILE1"

Note: at this point PEEKs could be done to verify that no buffer was in front of the program, FILE1. Again, POKE the indirect file I/O addresses for 24K systems

POKE 9554,8Ø

POKE 9368,8Ø

Typing

LIST <SHIFT K> <RETURN>

and

<SHIFT M> <RETURN>

writing FILE1 into the indirect file and closing that file.
Type

NEW

to remove FILE1 from the BASIC work space. Run the program "CHANGE" to create the needed buffer. Now, reload FILE1 from the indirect file by typing

<CONTROL X> <RETURN>

The original program with its newly acquired buffer is now resident in the BASIC work space. This program can be stored with the PUT command back on its original disk file (caution, the program is now larger by the buffer size, one or two tracks) by

DISK!"PUT FILE1"

This completes the examples. Since the indirect file stores its data as ASCII characters, it may be useful for file manipulation programs. There is a potential for greater utility than these examples with other applications. The indirect ASCII file is a subtle but powerful tool for experienced programmers.

APPENDIX P

BEXEC*

BEXEC* is the program which links the operating system and the end user programs. It is run by the operating system prior to turning control of the computer over to the user. BEXEC* typically provides setting critical parameters, such as specifying the input and output devices, and disabling or enabling certain entries, such as the <CONTROL C> entry to permit interrupting user programs. The demonstration disks and the operating system disks each have a program called BEXEC*. These versions may be used by copying the BEXEC* program for use in the users program development. However, it will often be desired to set some initial parameter (i.e., POKE some location) or run some initial program (such as a screen clearing program) prior to reverting to input to the BASIC system.

To start with an example of one:

```
   10 REM BASIC EXECUTIVE
   20 REM
   24 REM SETUP INFLAG & OUFLAG FROM DEFAULT
   25 X=PEEK (10950): POKE 8993, X: POKE 8994, X
   30 PRINT : PRINT "BASIC EXECUTIVE FOR OS-65D VERSION 3.N : PRINT
   40 PRINT
   50 GOTO 100
   60 PRINT : INPUT "FUNCTION";A$
   70 IF A$="CHANGE" THEN RUN "CHANGE"
   80 IF A$="DIR" THEN RUN "DIR"
   90 IF A$="UNLOCK" THEN 10000
  100 PRINT
  110 PRINT "FUNCTIONS AVAILABLE:"
  120 PRINT "CHANGE—ALTER WORKSPACE LIMITS"
  130 PRINT "DIR—PRINT DIRECTORY"
  140 PRINT "UNLOCK—UNLOCK SYSTEM FROM END USER MODIFICATIONS"
  150 GOTO 60
10000 REM
10010 REM UNLOCK SYSTEM
10020 REM
10030 REPLACE "NEW" AND "LIST"
10040 POKE 741, 76 : POKE 750, 78
10050 REM
10060 REM ENABLE CONTROL—C
```

```
1ØØ7Ø POKE 2Ø73, 173
1ØØ8Ø REM
1ØØ9Ø REM DISABLE "REDO FROM START"
1Ø1ØØ POKE 2893, 55 : POKE 2894, 8
1Ø11Ø PRINT : PRINT "SYSTEM OPEN" : END
```

The BEXEC* program shown sets the input and output devices to be the keyboard and video display and prompts the user to use the DIRectory or CHANGE utilities. If these utilities are not requested, the editing and debugging features of "NEW", "LIST", and <CONTROL C> are enabled. In certain programs (such as the example used in the section on Joystick use), the user may wish to disable these optional utilities prior to running programs. BEXEC* provides the ideal time to take care of these housekeeping functions.

Demonstration or game disks often require special provisions to be made. BEXEC* provides the opportunity to make these changes, including the guiding of the user by program prompts. To simplify use of demo or game disk, it is often convenient to start the user in his/her program. For example, to run a program (here called DEMO), the last statement in BEXEC* could be

```
RUN"DEMO"
```

In this manner, BEXEC* can take care of routine keyboard entries and simplify user response. As in most endeavors, simple is better.

APPENDIX Q

I/O DISTRIBUTION

Use of multiple input and output devices can be accomodated without the need for specialized PEEKs and POKEs, by using the I/O distribution system which is available under the DOS and BASIC. The following is an illustrative example using the ACIA.

The simplest way to send data to the ACIA is to inform the Disk Operating System (DOS) that the ACIA is to be an output port. The command, responding to the DOS prompt

A*

is

<u>IO ,Ø1</u>

This assigns the ACIA as the sole system output port.

The general form of I/O distribution is

IO ,nn to assign input devices only

IO ,mm to assign output devices only

IO ,nn,mm to assign both input and output devices

A blank must be used in the command, as illustrated. Note that these numbers, nn, mm, are in hexadecimal (base 16). Each device number assignment must be a two digit number selected from the following list:

HEX NN INPUT DEVICE CODE	HEX MM OUTPUT DEVICE CODE
ØØ Null	ØØ Null
Ø1 Serial Port (ACIA at FCØØ)	Ø1 Serial Port (ACIA at FCØØ)
Ø2 Keyboard on 54Ø Board	Ø2 Video on 54Ø Board
Ø4 UART on 55Ø Board	Ø4 UART on 55Ø Board
Ø8 Null	Ø8 Line Printer
1Ø Memory	1Ø Memory
2Ø Disk Buffer 1	2Ø Disk Buffer 1
4Ø Disk Buffer 2	4Ø Disk Buffer 2
8Ø 55Ø Board Serial Port	8Ø 55Ø Board Serial Port

Each of the device codes listed is a hexadecimal value corresponding to one bit or device. For example, the ACIA (device Ø1) is given by bits

ØØØØ ØØØ1

and the video board (CRT terminal) is device Ø2, given by bits

ØØØØ ØØ1Ø

Both devices can be used simultaneously by specifying the device with a bit pattern

ØØØØ ØØ11

which is hexadecimal Ø3. Therefore

IO ,Ø3

Will send data to the CRT terminal and the device on the ACIA port, simultaneously. Multiple output devices may be used (in contrast to only single input devices).

OTHER DEVICES

For other devices, it is probably easier to accept the device handlers built into the BASIC programs. Under BASIC, the devices are numbered sequentially, 1 to 9. This renumbering is distinct from the previous I/O command example. Under BASIC, the devices which are available are

DEVICE NUMBER	INPUT DEVICES
1	Serial Port (ACIA)
2	Keyboard on 54Ø Board
3	UART on 55Ø Board
4	Null
5	Memory
6	Disk Buffer 1
7	Disk Buffer 2
8	55Ø Board Serial Port
9	Null

DEVICE NUMBER	OUTPUT DEVICES
1	Serial Port (ACIA)
2	Video on 54Ø Board
3	UART on 55Ø Board
4	Line Printer
5	Memory
6	Disk Buffer 1
7	Disk Buffer 2
8	55Ø Board Serial Port
9	Null

The DOS I/O command previously discussed remains in effect until it is reset or an error occurs. If an error occurs, the default value is set (start up value). In contrast, the device numbers above can be assigned for each input/output operation as needed. For devices other than those set up by the DOS I/O command, the device assignments immediately above could be used.

For example, to read from the keyboard and write on the printer attached to the ACIA, the following instructions could be used:

```
1Ø INPUT #2,A$ : REM KEYBOARD INPUT
2Ø PRINT #1,A$ : REM TO PRINTER ON ACIA
3Ø LIST #1     : REM AND LIST PROGRAM, TOO
RUN
```

Yielding the input prompt

?

After typing a message (72 characters or less) and a <RETURN>, the message and the program will be printed on the serial printer.

DISK USE

As an input/output device, disks can be used in a similar manner.

However, prior to using the disk, the user should provide for protecting his buffer areas by running the CHANGE program as

RUN"CHANGE"

Respond to the terminal width change with

NO <RETURN>

and respond to a request to change the BASIC and ASSEMBLER use of memory by

NO <RETURN>

but respond to the work space limit change by

YES <RETURN>

The CHANGE program will ask "how many 8-page buffers before the work space." (Remember each page contains 256 characters.) There are only two valid responses here (1 and 2)

1. if only one file is to be used
2. two files must be open simultaneously

For the example that follows, 1 is sufficient. No additional room is required, so respond

NO <RETURN>

to that question. It is also not necessary to request any room at the top for this example.

The small differences between a disk and other devices are the need to open a disk file by name as

DISK OPEN,6, "FILE1"

and to close the file when finished by

DISK CLOSE,6

These last two statements can be used to store a string received from the modem. The input from the modem would be

INPUT #1,A$

where the string A$ must have as its last character <RETURN>.

Combining these three statements into a program to write a single message on disk

```
10 DISK OPEN,6, "FILE1"   :REM OPENS DISK (W/ONE BUFFER)
20 INPUT #1,A$            :REM LISTENS TO MODEM
30 PRINT #6,A$            :REM ECHOS TO DISK
40 DISK CLOSE,6           :REM CLOSES DISK FILE
50 END
```

Likewise, we could later recover the data by the program

```
10 DISK OPEN,6, "FILE1"
20 INPUT #6,A$
30 PRINT #2,A$
40 DISK CLOSE,6
50 END
```

In this problem, writing and reading was done sequentially. Modifying the program to accept multiple messages requires that they be stored sequentially.

It is possible to inspect the sequential disk file by

RUN"SEQLST"

which provides a listing of the file when the information requested is given. The computer responds

SEQUENTIAL FILE LISTER

TYPE A CONTROL C TO STOP

FILE NAME?

Respond with the file name of a sequential file

FILE1

and a listing of the file will be printed. Upon reaching the end of the disk file, the message

ERR #D ERROR IN 1ØØ

will indicate completion of the listing.

Caution: be aware that when using the SEQLST utility to inspect files which have BASIC programs stored in them, the display will look different than the original text. The reason for this is that the BASIC program stores BASIC source programs in a shorthand, called a tokenized form.

Another popular way is to transfer the disk file (say it was stored on track 39) by the CALL statement

DISK!"CALL D3ØØ=39,1"

which writes the file contents onto the middle of the CRT screen. Note that some apparent garbage will be additionally printed here due to the unused portion of the disk file's being printed, too.

To handle data in a random order, for example extracting the 20th data item from a file, it is not necessary to read the 19 prior data items. The use of random data items, also called records, is particularly useful when examining a large set of data. Such data might be a set of customer accounts, a checking account history, or even temperature records for given days. In all these cases, the need arises to extract a specific record, without looking at all the prior records.

To aid in understanding the handling of random records, visualize a pointer which marks the start of a record. The GET command moves this pointer at the start of a given record. For example,

DISK GET,Ø

places the pointer in front of the first record. Similarly,

DISK GET,5

places the pointer in front of the sixth record. This method makes it easy to locate a record on the disk, however, it is wasteful of disk storage capability.

Each record uses a large disk area (128 bytes). The value of 128 bytes is preset by the operating system.

A random (not sequential) input record may be terminated by the PUT command. This will close the present record from further input.

A simple program to write three records on disk file "SCRTCH" and then GET the second record from that file, would be

```
 1Ø REM PROGRAM WRITE TEST
 2Ø REM OPEN THE DISK FILE SCRTCH
 3Ø DISK OPEN, 6, "SCRTCH"
 4Ø REM LOOP THREE TIMES TO END OF LOOP
 5Ø FOR TIME=1 TO 3
 6Ø REM PLACE 128 BYTE RECORDS ON DISK BY
 7Ø REM (A) POSITION POINTER WITH A GET COMMAND
 8Ø REM (B) PASSING THE MESSAGE TO THE DISK BY PRINT COMMAND
 9Ø REM (C) CAUSE THE RECORD TO BE WRITTEN BY PUT COMMAND
1ØØ DISK GET, TIME-1
11Ø INPUT #2,A$ : REM TYPE IN ANY PHRASE FROM KEYBOARD
12Ø PRINT #6,A$ : REM PLACE IN MESSAGE BUFFER
13Ø DISK PUT     : REM TRANSFER MESSAGE BUFFER TO DISK
14Ø NEXT TIME
15Ø REM END OF LOOP
```

```
160 RCRD=2
170 DISK GET,RCRD-1 : REM POINTER AT START OF RECORD 2
180 INPUT #6,A$     : REM READ DISK's SECOND RECORD
190 PRINT #2,A$     : REM AND OUTPUT TO CRT (TERMINAL)
200 DISK CLOSE,6
210 END
```

The use of sequential and random disk files permits simpler control and bookkeeping than the CALL and SAVE or LOAD and PUT commands which were used for earlier file handling. This is one difference between record handling as compared to file handling.

MORE DEVICES

Memory can also be treated as a device. When accepting data from memory (Random Access Memory or RAM) as the input device, the DOS uses the address found in locations 238A (low address half) hexadecimal and 238B (high address half) hexadecimal to determine what memory region to use. After each input, the address is incremented by one location. Memory, as an output device, is specified by the contents of 2391 (low address half) hexadecimal and 2392 (high address half) hexadecimal.

To load the address of memory to be used as an input device into 238A and 238B, and also load the address memory to be used as an output device into 2391 and 2392, DOS provides the command

```
MEM mmmm,nnnn
```

mmmm is the address of memory to be regarded as an input device (its starting address) and nnnn is the address of memory to be regarded as an output device (its starting address). For example,

```
MEM 5000,5500
```

would load the locations

	LOCATION		
	DEC	**HEX**	**CONTENTS**
Input	9098	238A	00
Address	9099	238B	50
Output	9105	2391	00
Address	9105	2392	55

which establishes memory locations 5000 and up to be used as an input device and locations 5500 and up to be used as an output device. No end of these memory regions is specified, so the user is cautioned in their use.

Finally, a device called the null device is provided. The null device permits writing programs without having to worry about the physical device characteristics. For example, a program could be tested which would normally print on the printer; by assigning the null device, no paper would be wasted while the program is checked out.

INDEX

A

AIS Board 77
AC 68
AC1 68
AC2 68
AC12P 60
AC17P 60
AC21 54
Accessory Interface 61
ACIA 33
ACTL 51
Action Button 43
Advanced
  Features 40
  Topics 67
Alarm Operations 54
Analog
  I/O 61
  To Digital Conversion 62
Appliance Control 50
ASCII
  Code 25
  Example 99
  Table 106
ASM (Assembler) 38, 133
ASM Extensions 121

B

Back Arrow 119, 123
Backpanel Connectors 3
Barber Pole 76
BASIC Commands 38, 81
BASIC Conversions 86
BASIC Errors
  BASIC-in-ROM (Cassette) 85
  BASIC (Disk) 84
BASIC Extensions 65, 120
BASIC Programming 12
Baud Rate 64
Bit Switching and Sensing 60
BEXEC 11, 71, 141
Block 78
Break 5, 6, 41
Breakpoint 121
BUS, 16 Pin I/O 59, 60, 62
Bytes Free 95

C

CA-15 59
CA-20 59
CA-21 60
CA-22 61
CA-23 60
CA-24 61
CA-25 61
Calculator Mode 13
Cassette
  BASIC-in-ROM 14
  Cold Start 5
  Control Shift 8, 35
  Data Recovery 35
  Data Storage 35
  Load 9, 33
  Save 34
Change Utility 10, 94
Character Graphics 109
Character Manipulation 15
Character String 24, 82
Clock, Time of Day 68
Code, Machine Language 31
Cold Start 20
Color Graphics 27
Color, Inverted 29
Color's 10
Color Tuning 76
Combining Programs 139
Computer Setup
  Cassette 4
  Disk 4
Computer Interface to 16 Pin I/O Bus 59
Conditional Statement 15, 24

Connections
Closed Circuit Video 2
Pin 77
Video 2
Control C 41, 122
Control E 122
Control O 119
Control Q 119
Control R 122
Control S 119
Control Shift 8
Control T 122
Control U 122, 123
Control Y 122
Conversions
Analog to Digital 62
Digital to Analog 31, 62
Hexadecimal to Decimal Chart 104
Languages 38
Copier 124
Copy 95
Countdown Timer 69
Create 97

D

D- 41
Data Register (PIA) 54, 59, 60, 71
Default 24
Delete (files) 93
Device Numbers 120
Devices 144, 147
Digital to Analog (D/A) Converter 30, 31, 62
Dictionary 119
Dimension (DIM) 24, 81
Direct Video (modifications) 2
Directory
DIR 8, 10, 11, 97
Disk Directory Listings 11, 97
Format 11, 97, 123
DIR SRT (Sorted directory) 11, 97
Disk
Allocations 79, 80, 120, 123
BASIC Commands 6, 81-84
Care of 1
Copier 124
Copy 95
Extensions 65, 121
I/O (OPEN, GET, PUT, CLOSE) 146
Organization 38
Programs 7, 36, 37
Read/Write 36
Systems 10
Track Formatting 80, 122
Track Ø Read/Write 94
Utilities 94
DOS (disk operating system)
Commands 38
Errors 85

E

EM Command 38
Entries Free 97
Error 60
BASIC-in-ROM 85
Disk BASIC 84
DOS 85
Numbers 118
Extended Monitor 38, 121
Extensions to Disk BASIC 65, 119
Extensions to Assembler 121
External Switches 54

F

File 67
Access 10, 33, 139, 142
Create 97
Delete 93
Indirect 139
Source Formatting 123
Flag Bit Setting, I/O 119
Flow Chart Explanation 17
Frequency (piano keyboard) 92

G

Generator, Tone 30
GET/PUT Overlays 80
Graphics 108
Greenhouse Example 71

H

Head End Carps 59, 60
Heterodyning 76
Hexadecimal to Decimal
Tables 104
Tutor and Conversions 99
Home Security 52

I

Indirect Files . . . 138
Inverted Video . . . 29
I/O (input/output)
  Analog . . . 61
  Disk . . . 119
  Distribution . . . 143
  Flag Bit Setting . . . 119

J

Joysticks . . . 42

K

Keyboard . . . 40, 92
Keypads . . . 47
Key Usage . . . 41

L

Labeling . . . 13
Languages . . . 31, 38, 67, 124, 133, 135
Left$ . . . 16
List . . . 83, 121
Loops, Explanation of . . . 20
LUN . . . 120

M

M- . . . 41
Machine Code Language . . . 31, 124, 135
Machine Organization . . . 76
Memory
  Allocation . . . 118
  Map . . . 79
  Move . . . 82
  Video Maps . . . 28
MID$ . . . 16
Modem . . . 60, 63
Monitor, Extended . . . 121
Monitor (machine) . . . 124
  Real Time (RTMON) . . . 71
Multiplexer . . . 61, 62

N

NEW . . . 124
Notations . . . 7
Null Device . . . 147

O

Organizations
  Machine . . . 76
  Operating System . . . 38
  Floppy Disk . . . 78
OS-65D User's Guide . . . 116

P

Page . . . 78
Passing Parameters . . . 131
Peripherals . . . 63, 65, 67
PIA Data . . . 54, 55, 60, 71
Piano Keyboard . . . 92
Plot BASIC . . . 67
POKEs and PEEKs . . . 87
Pound Sign . . . 120
Power
  Down . . . 8
  Up . . . 6
PRINT . . . 81
Printer . . . 65
Program Mode . . . 13
Prototyping . . . 61

R

RAM . . . 33, 78
Read/Write
  Cassette . . . 35
  Disk . . . 36
  Track Ø . . . 96
Real Time
  Clock . . . 68
  Control . . . 68
  Monitor (RTMON) . . . 54, 69, 70, 72
Remarks . . . 15
RENAME . . . 93
Return . . . 5, 6, 9, 41
RF Modulator/Standard TV . . . 2
Right$ . . . 16
ROM . . . 14
RUN . . . 8, 9, 13

S

Screen Cleaning Routine (USR-X) 127
SCRTCH 36, 98
Sector 78
Security 52
SEQLST (sequential file lister) 145
Shift (O) 14, 41
Shift (P) 41
SL 41
Source File Format 123
Sound 30
Space Bar 41
Strings 15
Subscripted Variables 21, 24
Summary 62
Switches (external) 54
Syntax (Error) 75
System Organization 38

T

Telephone Interface 63
Terminal Communications 63
Time
  Control 68
  Of Day Clock 68
  Monitor (RTMON) 69
Timer (countdown) 69
Tone Generator 30
Track Formatting 80, 123
Track Ø Read/Write 96, 124
Troubleshooting 75
Tutor, Hexadecimal to Decimal 99

U

Unlock 7, 19, 12
Up Arrow 123
USR(X) Function 67, 126

V

Video
  Connections 2
  Close Circuit 2
  Direct (modifications) 2
  RF modulator 2
  Memory Map 78, 115
  Screen Layout 110

X

XQT
  File Storage 135
  Filnam Command 118

Z

Zero Utility 80

NOTES

OHIO SCIENTIFIC

1333 S. Chillicothe Road - Aurora, OH 44202

Phone: (216)562-3101