

The next option to change is available memory. Since you will receive a default value of 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 (1024 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 you enter YES, 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 50 through 191.

The program continues with:

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

Enter YES or NO. If you enter NO, the program terminates. If you enter YES, the program requests the following:

HOW MANY 12 PAGE BUFFERS DO YOU WANT BEFORE THE WORK SPACE?

Enter 0, 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 you enter 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 you enter 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 you enter NO, the program requests that you specify an exact lower limit address for the work space.

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 you enter 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 you enter NO, the program requests that you specify an exact number limit address for the work space.

NEW UPPER LIMIT?

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

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.

## Chapter V

### Peripherals, An Overview

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 0's, sent over lines from the computer as output. The 1's and 0's are represented by nominal 5 volt and 0 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 C8P 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, "Appliance Control" and "The Home Security Alarms", 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 your readily written BASIC program, based on the examples that follow.

## 1. Appliance Control

Without running any wires your C4P can operate lamps and small appliances when equipped with the AC-12 options! This is accomplished by the BSR X-10<sup>(C)</sup> 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 (110 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 300 watts per lamp) throughout a building. A relay actuated (on-off) appliance switch provides control of larger devices such as lamps (up to 500 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". You have 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 your 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 your directions under computer control, can be specified with simple commands.

In order to run your own AC control programs, we need to borrow support programs from your system disk (OS-65U V3.1 HC).

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.

Your user 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 you can use

```
POKE 56832,1
```

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 for us by the command

```
POKE 548,127
```

```
POKE 549,50
```

Having taken care of the three required POKEs, we can now write device driver programs.

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:

<u>Function</u>	<u>COMMAND</u>
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	70

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

If a light is in the off state, brightening it will result in it 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 3 of the 12 levels of brightness by

ACTL 4,66,66,66 <RETURN>

Another variation of the ACTL command is

ACTL DEVICE1  
ACTL DEVICE2  
:  
ACTL COMMAND  
ACTL COMMAND  
:

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

```
100 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.

We now have 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. When we have examined each of the available devices, we shall put the devices together in a REAL TIME system.

## 2. The Home Security Alarms

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 your vehicles. All alarms report their status by radio-control to the home control module, connected to your computer. 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 for you to disable the alarm when you enter 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 program below:

```

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 : BURGLER=2 : CAR=4 : MISC=8
70 T1=PEEK(ALARM) AND FIRE
80 T2=PEEK(ALARM) AND BURGLER
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 "BURGLER"
140 IF T3=TRIP THEN PRINT "CHECK CAR"
150 IF T4=TRIP THEN PRINT "MISC ALARM"
160 GOTO 70
170 END

```

In later examples, we shall incorporate further alarm response. Your alarm monitoring can be done while other programs are being run. This powerful technique is available when you use the real time monitor, RTMON. Many 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.

## Chapter VI

### General Peripherals, Their Use

The distinguishing characteristic of the home security/control features is the ability to control external devices by use of your C8P computer. But the C8P is still a "personal" computer.

The traditional devices, such as a line printer or modem, can be attached (through standard connectors) to your C8P system ("modulator-demodulator", used to connect the telephone to the computer). The computer signals the modem to generate or receive tones which can be carried by the telephone lines. A line printer provides convenient data logging for record keeping in the home or business. Further, the line printer provides a documentation aid in program development. For these reasons, a line printer is often an early choice in expanding a user's system. More recently, low cost modem circuits have permitted connecting the user's C8P to a telephone interface. This allows conversations between computers. In many environments, students and others can find access to the large mainframe computers at their school or place of employment which allow dialing-in for off-site use. Your modest investment in a C8P system can unlock all the facilities of a large computer center at your convenience, without travel from your chosen location.

The home or business C8P system will also have need of checking switch positions for open or closed status. Home security systems provide an obvious application. We'll use a greenhouse temperature control and alarm system in a later example. Fire alarms, counting events, timing occurrences, regulating intervals, signaling by tone generation, and even voice generation are possible with your C8P system.

The demonstration disks have shown you part of the power of your system. The applications which you will write can bring this power to tasks of your choosing.

Let's look at the characteristics and use of the external devices. The examples which follow will show

- 1) device assignments, including use of printer, modem and disks
- 2) tone and music generation
- 3) joystick and keypad entry of data. (Also, these input devices provide convenient game control as a bonus.)
- 4) voice generation for communications  
and
- 5) control using timers and a real time monitor to supervise the tasks above.

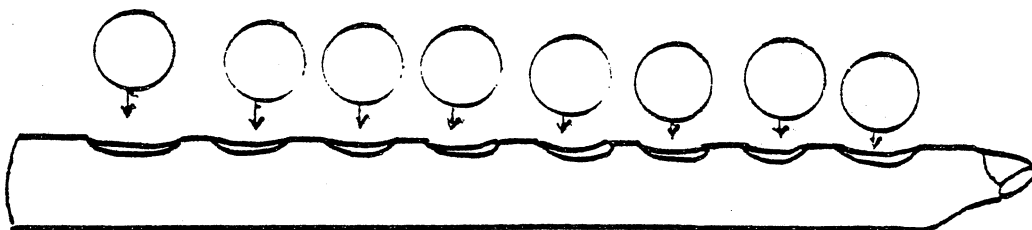
Now, let's examine those applications.

## 1. The Printer, Modem and Other Input/Output Devices

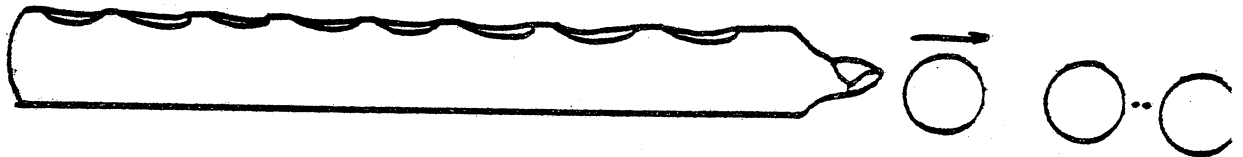
Each character which we store or move is represented by 8 bits (ones or zeros). Normally, we have data on eight data lines (called a data bus), simultaneously. This is convenient when the cost of maintaining multiple lines is low, due to short line lengths. For longer lines, extra circuits for each line 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 Aynchronous Communications Interface Adapter (ACIA) which converts the parallel (simultaneous) data into serial data for transmission (or reverses the process for reception).

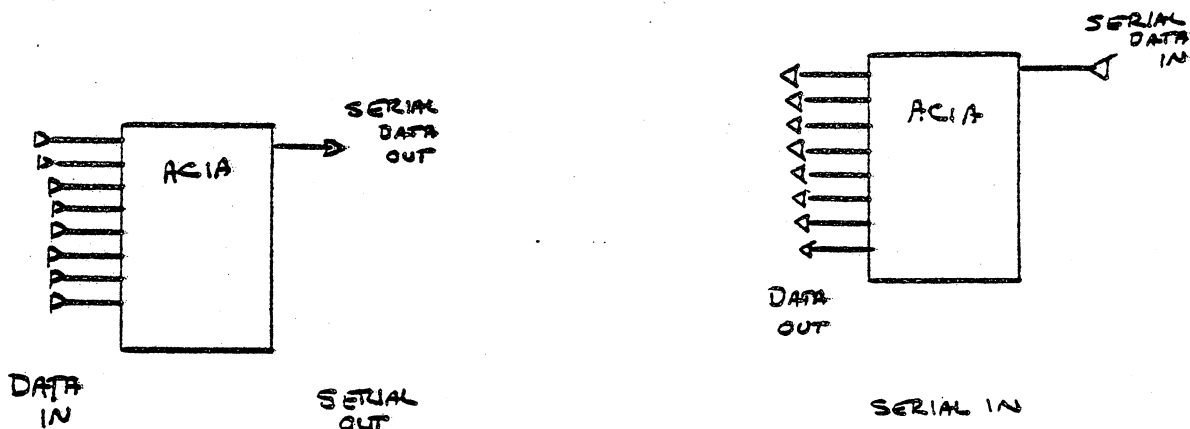
A simple analogy might suggest the function of the ACIA. Consider that the input from a computer is typically 8 parallel, simultaneous, input bits. We can represent the function of the ACIA by a child's flute, where we input data (peas) in parallel



As data is output (removed from the ACIA device) we represent the sequential or serial flow as



Data flow in both directions could be accommodated (though not simultaneously) by reversing the process. The electrical equivalent of the flute analogy would be



where control and timing for the ACIA must be provided, additionally.

This serial (or sequential) handling of bits requires fewer wires for data transmission, but the data handling rates are consequently reduced. This is no disadvantage if the device to which we sent data is limited to low mechanical speeds (such as printers, plotters) or low data rates (telephone lines and their modems).

The system will normally be set up with the information handling rate (baud rate) set at 1200 bits per second (1200 baud). For the modem use, this must be changed to 300 baud. The two choices are given by

POKE 64512,1 : REM 1200 BAUD RATE

or

POKE 64512,2 : REM 300 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 interface to the outside (of the computer) world.

## 2. I/O Distribution

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

IO ,01

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

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 CodeHex mm Output Device Code

00 Null

01 Serial Port (ACIA at FC00)

02 Keyboard on 440/540 Board

04 UART on 430 Board

08 Null

10 Memory

20 Disk Buffer 1

40 Disk Buffer 2

80 550 Board Serial Port

00 Null

01 Serial Port (ACIA at FC00)

02 Video on 440/540 Board

04 UART on 430 Board

08 Line Printer

10 Memory

20 Disk Buffer 1

40 Disk Buffer 2

80 550 Board Serial Port

Each of the device codes listed is a hexadecimal value corresponding to one bit or device. For example, the ACIA (device 01) is given by bits

0000 0001

and the video board (CRT terminal) is device 02, given by bits

0000 0010

We can use both devices simultaneously by specifying the device with a bit pattern :

0000 0011

which is hexadecimal 03. Therefore

IO ,03

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). We could have attached either a serial printer or a telephone line modem to the ACIA output as the external device. However, only one device, the modem or the printer, may be attached at any one time. That is, you can't

have power applied to the printer and modem simultaneously. You may 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. Since the I/O command may specify multiple input devices, a priority rule is established. On input, the lowest numbered devices gets to talk. Other devices are ignored. This gives the modem port high priority.

As an alternative to the I/O command, the ACIA port may be addressed by using the ACIA control register address of FC00 hexadecimal (64512 decimal) and its data register of FC01 hexadecimal (64513 decimal). Reading or writing can be accomplished using the BASIC PEEK and POKE commands.

The simple program

```
5 REM PRINTER PROGRAM
10 POKE 64512,1 : REM SET 1200 BAUD RATE
20 A$="NOW IS THE TIME FOR ALL GOOD MEN"
30 FOR T=1 TO 20 : REM PRINT 20 TIMES
40 FOR K=1 TO LEN (A$)
50 A=ASC(MID$(A$,K,1))
60 FOR DELAY=1 TO 2 : NEXT DELAY
70 REM WE HAD A SLOW PRINTER
80 POKE 64513,A
90 NEXT K : REM MESSAGE COMPLETE
100 POKE 64513,10 : REM LINE FEED PAPER
110 POKE 64513,13 : REM CARRIAGE RETURN
120 NEXT T : REM DO ALL 20 LINES
130 END
```

prints the message

NOW IS THE TIME FOR ALL GOOD MEN

twenty times, illustrating the ACIA function.

This method is less convenient than the I/O command discussed previously and should be used only to overcome new device limitations (such as the need for the additional delay created by a delay loop in line 60).

### 3. 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

<u>Device Number</u>	<u>Input Devices</u>	<u>Device Number</u>	<u>Output Devices</u>
1	Serial Port (ACIA	1	Serial Port (ACIA)
2	Keyboard on 440/540 Board	2	Video on 440/540 Board
3	UART on 430 Board	3	UART on 430 Board
4	Null	4	Line Printer
5	Memory	5	Memory
6	Disk Buffer 1	6	Disk Buffer 1
7	Disk Buffer 2	7	Disk Buffer 2
8	550 Board Serial Port	8	550 Board Serial Port
9	Null	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, we could use the device assignments immediately above.

For example, to read from the keyboard and write on the printer attached to the ACIA, we may use

```
10 INPUT #2,A$ : REM KEYBOARD INPUT
20 PRINT #1,A$ : REM TO PRINTER ON ACIA
30 LIST #1      : REM AND LIST PROGRAM, TOO

RUN
```

We will get 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.

#### 4. 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"
```

You should 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 you "how many 12-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 only 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. You also need not 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

We can use these last two statements 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 we have written and read sequentially. If we modify the program to accept multiple messages, they would be stored sequentially, one after another.

You may inspect the sequential disk file by

```
RUN"SEQLIST"
```

which provides a listing of the file when you give the information requested. The computer responds

```
SEQUENTIAL FILE LISTER
```

```
TYPE A CONTROL C TO STOP
```

```
FILE NAME?
```

You 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 100
```

will indicate completion of the listing.

Caution: if you use 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 (let's say it was stored on track 46) by the CALL statement

```
DISK!"CALL D300=46,1"
```

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 being printed, too.

If you wish 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 you examine 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,0

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.

We may terminate a random (not sequential) input record 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

```

10  REM PROGRAM WRITE TEST
20  REM OPEN THE DISK FILE SCRTCH
30  DISK OPEN, 6, "SCRTCH"
40  REM LOOP THREE TIMES TO END OF LOOP
50  FOR TIME=1 TO 3
60  REM PLACE 128 BYTE RECORDS ON DISK BY
70  REM  (A) POSITION POINTER WITH A GET COMMAND
80  REM  (B) PASSING THE MESSAGE TO THE DISK BY PRINT COMMAND
90  REM  (C) CAUSE THE RECORD TO BE WRITTEN BY PUT COMMAND
100 DISK GET, TIME-1
110 INPUT #2,A$ : REM TYPE IN ANY PHRASE FROM KEYBOARD
120 PRINT #6,A$ : REM PLACE IN MESSAGE BUFFER
130 DISK PUT      : REM TRANSFER MESSAGE BUFFER TO DISK
140 NEXT TIME
150 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 we used for earlier file handling. This is one difference between record handling as compared to file handling.

## 5. More Devices

Memory can also be treated as a device. When we accept 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		<u>Contents</u>
	<u>Dec</u>	<u>Hex</u>	
Input Address	9098	2384	00
	9099	238B	50
Output Address	9105	2391	00
	9106	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.

## 6. Tone Generator Effects

For games or test signals, we often wish to have a tone generated at a specific frequency. This frequency can be heard when the audio output (See Figure 1) of the C4P is connected to the audio input jack of your 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 57089 by

POKE 57089,I

The registers at 56832 and 57089 are write only locations, and cannot be PEEKed.

A familiarization test program which lets you hear the range of tones produced is

```
10 TUNES=57089
20 CST=49152      :REM CONSTANT FOR FREQUENCY CALCULATION
30 FOR I=1 TO 255
40 POKE TUNES,I
50 F=INT(CST/I)   :REM F IS FREQUENCY IN HERTZ (CPS)
60 PRINT I;F
70 NEXT I
80 POKE TUNES,1   :REM BE SURE TO TURN TONE OFF!
90 END
```

To try this computer feature in a more interesting tune, we find the first seven notes of "Twinkle, Twinkle Little Star" have frequencies of 261.6, 261.6, 392.0, 392.0, 440.0, 440.0, 392.0 Hertz (cycles per second). The frequency of the different notes appears in many encyclopedias and handbooks. We can input these data from a BASIC program as

```

5 REM TWINKLE TWINKLE TUNE
10 TUNE=57089
20 FOR T=1 TO 7
30 READ N,BEATS
40 I=INT(49152/N)
50 POKE TUNE,I
60 FOR DELAY=1 TO 500*BEATS
70 NEXT DELAY
80 FOR D=1 TO 50:POKE TUNE,1:NEXT D
90 NEXT T
100 DATA 261.6,1,261.6,1
110 DATA 392.0,1,392.0,1
120 DATA 440.0,1,440.0,1
130 DATA 392.0,2

```

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. We provide a keyboard and note guide in the appendix to help you write your own tunes.

#### Twinkle Tune



## 7. Graphics

High quality graphics have been provided on the C8P 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

<u>N</u>	<u>Characters Per Line</u>	<u>Sound On/Off</u>	<u>Color/ Black &amp; White</u>
0	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).

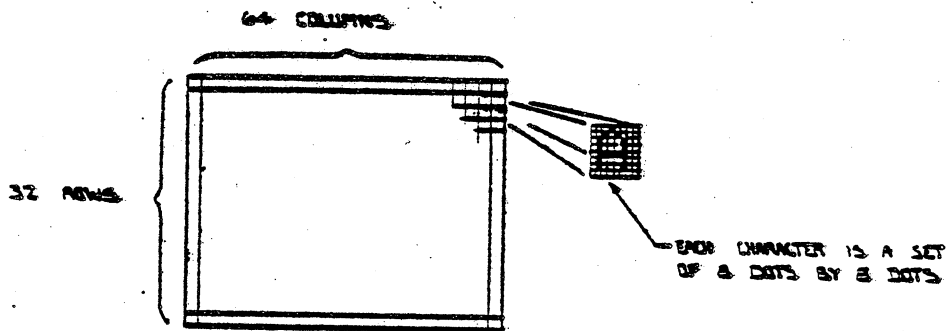
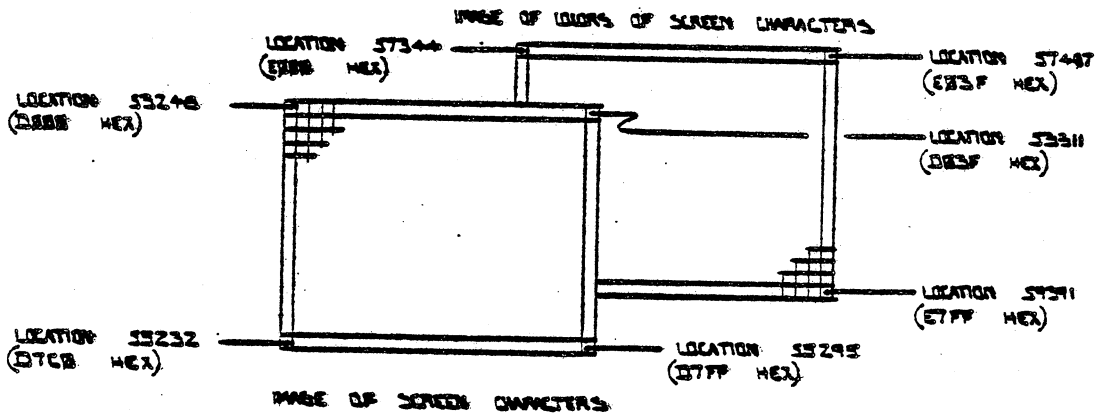
There are 256 selectable characters which are available for your use. The 256 characters, selected from a larger possible set, provide versatile graphics without heavy demands for memory.

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 4096 locations beyond the location for the corresponding symbol. (Since 16 colors are available, only 4 bit (half byte) storage is provided.) You might regard memory as an image of the screen.

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 BASIC program to place a blue "X" in the middle region of a 64 by 32 character display, at location 54302 (D41E hexadecimal) would be

```
10 POKE 56832,5 : REM TURN COLOR ON, SOUND OFF
20 POKE 54302,188 : REM MID SCREEN LOCATION 54303
30 REM SYMBOL 181 IS AN X
40 POKE 58348,8 : REM COLOR NUMBER 8 IS BLUE
```



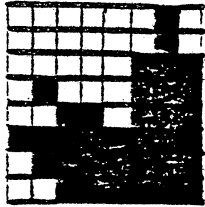
DATA	162	24	240
	14	105	6
	169	1	142
	0	141	0
	141	242	224
	242	2	6
	2	173	220
	169	243	2
	224	2	96
	141	105	0
	243	0	2
	2	141	
	173	243	
	242	2	
	2	201	
		232	

Our color selections must be made from the list

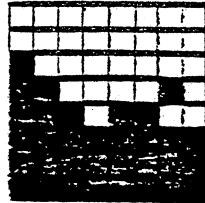
<u>Decimal Value</u>	<u>Color Selected</u>
0	Yellow
1	Inverted Yellow
2	Red
3	Inverted Red
4	Green
5	Inverted Green
6	Olive Green
7	Inverted Olive Green
8	Blue
9	Inverted Blue
10	Purple
11	Inverted Purple
12	Sky Blue
13	Inverted Sky Blue
14	Black
15	Inverted Black (no color)

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.

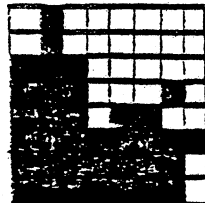
To illustrate the color choices, we shall try a program that places the symbol numbers 181, 182, 179, 180 (the shape of a ship in that order) into adjacent locations.



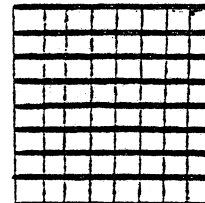
181



182



180



96

We shall display this ship across four columns for 16 times. Each time we shall change the color. Our program would be

```
10 POKE 56832,5 : REM SET UP COLOR ON, SOUND OFF
20 ST=53248 : REM START AT UPPER LEFT
30 C=ST+4096 : REM COLOR AT 4096 BEYOND SCREEN LOCATION
40 FOR RW 0 TO 32 : REM ROW INCREMENT LOOP
50 FOR CM 0 TO 63 STEP 4 : REM COLUMN INCREMENT LOOP
60 D=RW*64+CM : REM COMPUTE SCREEN DISPLACEMENT
70 POKE ST+D+0,181 : REM SHIP USES 4 CELLS
80 POKE ST+D+1,182
90 POKE ST+D+2,180
100 POKE ST+D+3,96
110 FOR I=1 TO 3
120 POKE C+D+I,INT(CM/4) : REM SAME COLOR FOR WHOLE SHIP
130 NEXT I
140 NEXT CM
150 NEXT RW
160 GOTO 20
```

Since we have looped the program on itself, we use <CONTROL C> to exit this program.

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

## 8. 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. If we set (POKE) a row to be examined and look at (PEEK) a desired column, we can tell whether the row-column key is depressed. The values which need to be POKEd and PEEKed are shown below:

		VALUES FOUND WHEN PEEKed							
DECIMAL VALUE		128	64	32	16	8	4	2	1
		C7	C6	C5	C4	C3	C2	C1	C0
DECIMAL VALUE									
128	R7	1	2	3	4	5	6	7	
64	R6	8	9	0	:	-	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	R0	REPT	CTRL	ESC			L SHIFT	R SHIFT	SHIFT LOCK

VALUES TO POKE

The keyboard appears at address 57088. To test for depression of the key "V", for example, the sequence would be

```
10 R4=4:C5=32
```

```
20 POKE 57088,R4
```

```
30 VT=PEEK(57088):REM VT IS TEST FOR V'S COLUMN
```

```
40 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 us to disable the <CONTROL C> feature to avoid problems in identifying which keys are used. This is done by a POKE 2073,96 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 57088, 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. Within BASIC, their effect is the same.
- 4) The "break" key is brought directly to the computer reset circuits. Use of this key restarts the system operation.

We have examined 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 we 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 your system or running BASIC. Unlike a typewriter, however, the numbers will be printed normally. If you wish 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.
- 4) 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 simultaneously pressing C. Program listing or executing is interrupted, and the message  
  
BREAK IN LINE XXX  
  
is printed.
- 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". You 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.

With this agreed notation, let's write a program!

## 9. Keypad

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, we must disable <CONTROL C> , with a POKE 2073,96.

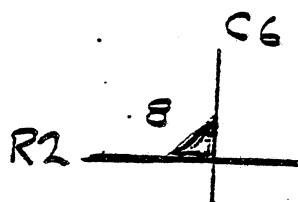
Let's examine how keypad A is connected. Keypad A consists of a set of wires which correspond to keyboard rows shown labeled as R1 to R4. These are shown superimposed on the keyboard rows R0 to R7. In the same manner, the keypad A contains wires corresponding to keyboard columns C5 to C7 out of the total 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.

Values found when PEEKed

		128	64	32	16	8	4	2	1
		C7	C6	C5	C4	C3	C2	C1	C0
Values to POKE	128 R7								
	64 R6								
	32 R5								
	16 R4	1	2	3					
	8 R3	4	5	6					
	4 R2	7	8	9					
	2 R1	*	0	#					
	1 R0								

Keypad A

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



with the key symbol next to the shaded region.

Likewise, keypad B is connected as

Values found when PEEKed

		128	64	32	16	8	4	2	1
		C7	C6	C5	C4	C3	C2	C1	C0
Values To POKE	128: R7								
	64 R6								
	32 R5								
	16 R4				1	2	3		
	8 R3				4	5	6		
	4 R2				7	8	9		
	2 R1				*	0	#		
	1 R0								

Keypad B

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

We can detect the symbol 8 (located at the intersection of Row 2 and Column 6 on keypad A) by setting Row 2 via

```
10 POKE 57088,4
```

where 4 is the value POKEd to activate Row 2. We can sense Column 6 (value associated with column 6 is 64) by

```
20 TEST = PEEK(57088)
```

```
30 IF TEST = 64 THEN GOTO 1000
```

where statement 1000 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:

```
10 REM KEYPAD TEST
20 REM DISABLE <CONTROL C>
30 CTRLC=2073: DISABL=96: POKE CTRLC,DISABL
40 REM NOW SET POINTER TO KEYPAD LOCATION
50 P=57088: 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)=C5 THEN A$="#" : REM ON R1,C5
```

## 10. Joystick

The joysticks provide realistic and convenient input devices for games and control. They are connected to the system as shown in Figure 1. The joysticks provide a digital signal when they are connected and enabled.

Prior to using the joysticks (or keypads) the  $\langle \text{CONTROL C} \rangle$  command must be disabled by

```
POKE 2073,96
```

The enabling of joystick A is done by

```
POKE 57088,128 : REM - ENABLE JOYSTICK A
```

and joystick B is enabled by

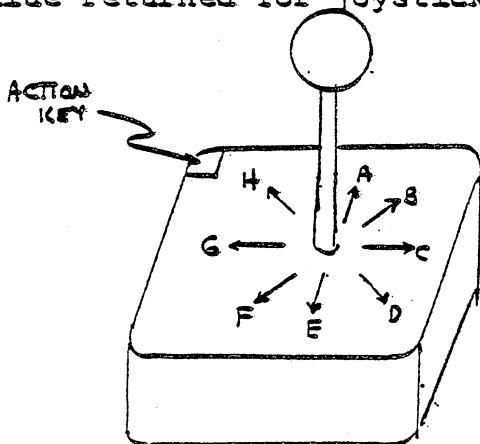
```
POKE 57088,16 : REM - ENABLE JOYSTICK B
```

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 for joystick B. For example

```
APOSIT=PEEK(57088) 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



POSITION 1  
IS THE  
CENTER  
(NEUTRAL)  
POSITION

<u>Joystick Position</u>	<u>Joystick A</u>		<u>Joystick B</u>	
	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	17	32	160
B	20	21	48	176
C	4	5	16	144
D	12	13	80	208
E	8	9	64	192
F	10	11	72	200
G	2	3	8	136
H	18	19	40	168
I	0	1	0	128

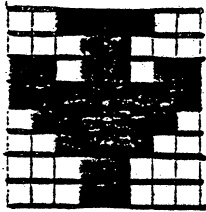
With the action key depressed, 1 has been added to the "action key not depressed" value for joystick A.

When joystick B is enabled, the corresponding values returned are returned to

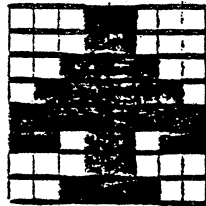
BPOSIT=PEEK(57088) AND 248

The "action key depressed" causes 128 to be added to the "action key not depressed" value for joystick B.

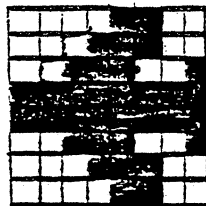
Let's try a sample program. We'll use the airplane figures



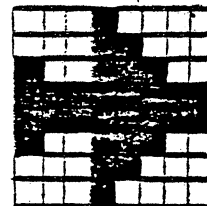
238



236



239



237

to move about the screen. Let's place the plane in the screen center to start at location 53404 (D420 hexadecimal). We'll ignore clearing the screen, for example, simply leaving it in B & W with 64 characters per line and the sound off, by typing

```
10 POKE 56832,1
```

We'll put the original plane on the mid-screen by

```
20 POKE 54304,236
```

Since we are B & W, no color is given. We shall use the "ACTION" button to quit (exit) the program. We shall use the logic shown in Figure 7.

# Flow Chart for Airplane and Joystick

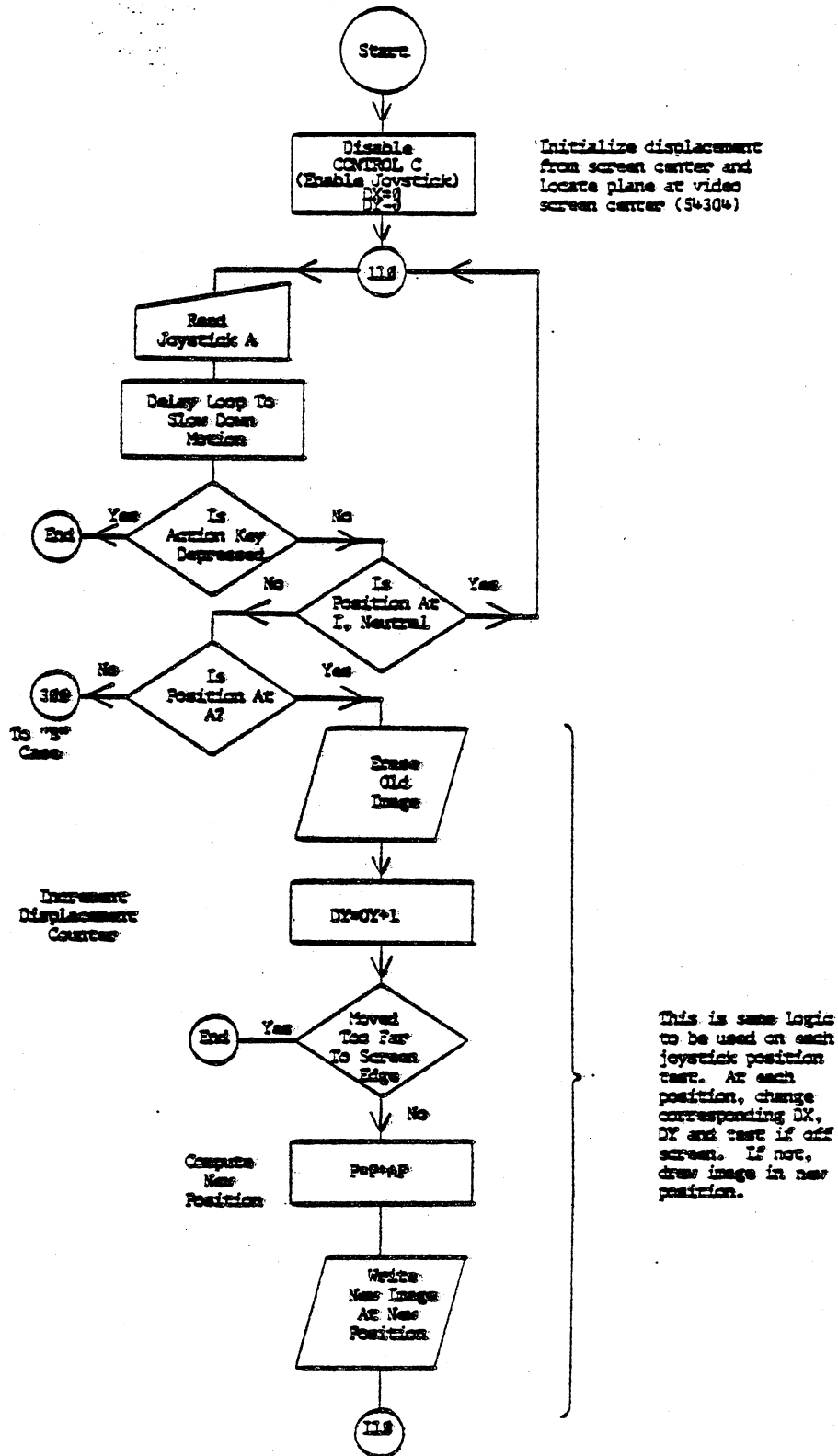


Figure 7.

The program to implement this flowgraph is

```
10 POKE 2073,96 : REM DISABLE <CONTROL C>
20 AP=-64:BP=-62:CP=+1:DP=66 :REM SCREEN POSITION DISPLACEMENTS
30 EP=64:FP=62:GP=-1:HP=-66:IP=0 :REM RESULTING FROM JOYSTICK
    POSITION
35 REM
40 A=16:B=20:C=4:D=12 :REM CODE VALUES FOR
50 E=8:F=10:G=2:H=18:I=0 :REM JOYSTICK POSITION
55 REM
60 POKE 57088,128 :REM ENABLE JOYSTICK A
70 BLANK=96 :REM SCREEN SYMBOL FOR BLANK
80 DX=0:DY=0
90 P=54304 :REM MIDSCREEN START
100 POKE P,236
110 R=PEEK(57088) AND 31
120 FOR K=1 TO 200:NEXT K :REM DELAY LOOP
130 IF(R/2-INT(R/2)) >.1 THEN GOTO 9000 :REM QUIT IF ACTION KEY
135 REM :REM DEPRESSED (ODD VALUE R)
140 IF R=IP THEN GOTO 110
150 IF R =A THEN GOTO 170
160 GOTO 300
170 POKE P, BLANK :REM - ERASE OLD IMAGE
180 DY=DY+1
190 IF ABS(DY) >16 THEN GOTO 9000 :REM IF OFF SCREEN, QUIT
200 P=P+AP
210 POKE P,236 :REM "A" POSITION IS UPWARD PLANE
220 GOTO 110
300 IF R=B THEN GOTO 320 :REM "B" CASE
```

The program to implement this flowgraph is

```
10 POKE 2073,96 : REM DISABLE <CONTROL C>
20 AP=-64:BP=-62:CP=+1:DP=66 :REM SCREEN POSITION DISPLACEMENTS
30 EP=64:FP=62:GP=-1:HP=-66:IP=0 :REM RESULTING FROM JOYSTICK
    POSITION
35 REM
40 A=16:B=20:C=4:D=12 :REM CODE VALUES FOR
50 E=8:F=10:G=2:H=18:I=0 :REM JOYSTICK POSITION
55 REM
60 POKE 57088,128 :REM ENABLE JOYSTICK A
70 BLANK=96 :REM SCREEN SYMBOL FOR BLANK
80 DX=0:DY=0
90 P=54304 :REM MIDSCREEN START
100 POKE P,236
110 R=PEEK(57088) AND 31
120 FOR K=1 TO 200:NEXT K :REM DELAY LOOP
130 IF(R/2-INT(R/2)) >.1 THEN GOTO 9000 :REM QUIT IF ACTION KEY
135 REM :REM DEPRESSED (ODD VALUE R)
140 IF R=IP THEN GOTO 110
150 IF R =A THEN GOTO 170
160 GOTO 300
170 POKE P, BLANK :REM - ERASE OLD IMAGE
180 DY=DY+1
190 IF ABS(DY) >16 THEN GOTO 9000 :REM IF OFF SCREEN, QUIT
200 P=P+AP
210 POKE P,236 :REM "A" POSITION IS UPWARD PLANE
220 GOTO 110
300 IF R=B THEN GOTO 320 :REM"B" CASE
```

```

310 GOTO 400
320 POKE P,BLANK
330 DY=DY+1:DX=DX+1
340 IF ABS(DX) >30 OR ABS(DY) >16 THEN GOTO 9000
350 P=P+BP
360 POKE P,237
370 GOTO 110
400 IF R=C THEN GOTO 420 :REM "C" CASE
410 GOTO 500
420 POKE P,BLANK
430 DX=DX+1
440 IF ABS(DX) >30 THEN 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 repeated use of the same tests and operations, in blocks of less than 10 instructions. We now have a nucleus of programs to implement our own games!

## 11. 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. If you make copies of this disk, you should copy only the version of this AC control program (AC, AC1 or AC2) which you need.

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

To write your own 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 your personal applications.

## 12. 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 3.2 HC operating system. To set the time of day clock, we set hours in location 9480, minutes in 9481, and seconds in 9482. The commands are

POKE 9480,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 0:0:0. Location 9483 holds the count of the number of 24 hour periods (i.e. days) which have been counted.

Time is read by the PEEK command. For example:

```
10 REM INPUT TIME TO SET CLOCK
20 INPUT "HOURS, MINUTES, SECONDS";H,M,S
30 POKE 9480,H:POKE 9481,M:POKE 9482,S
40 REM NOW TO PRINT OUT TIME
50 H=PEEK(9480):M=PEEK(9481):S=PEEK(9482)
60 PRINT H;" ":"M ":"S":"LOCAL TIME"
70 END
```

will permit setting the time, then displaying the time. Replacing statement 70 with

```
70 GOTO 50
```

will continually print the time.

### 13. 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 time of day clock, 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 0 in location 223.

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

```
10 POKE 223,0
20 INPUT "HOURS, MIN, SEC COUNTDOWN";H,M,S
30 POKE 224,H
40 POKE 225,M
50 POKE 226,S
60 REM NOW START TIMER
70 POKE 223,1
```

A program could check the one location, 223, to determine if the hours, minutes, and seconds had elapsed by

```
80 TEST=PEEK(223)
90 IF TEST=0 THEN GOTO 1000
100 GOTO 90
```

The real value of the timer, however, lies in its ability to request the services of the real time monitor, RTMON. RTMON permits inter-

rupting 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 after we have looked at some other devices and features available for home and appliance control.