Cassette I/O With AIM 65 BASIC

Michael Rathbun
Polar Solutions
Kodiak, Alaska

The AIM 65 is one of the few micro systems I have worked with which was packed with PLEASANT surprises. Its monitor, assembler, and BASIC do things I didn't expect from a piece of equipment in its price range. After a while, however, I found myself wishing that the excellent AIM cassette system could be used with the BASIC on the system for data input and output, instead of just for SAVE and LOAD. It turns out that, because BASIC uses certain monitor routines to interface the keyboard and display/printer, BASIC cassette file I/O is not all that difficult.

Monitor Routines

For those who haven't spent an exciting evening or two reading the assembly listing of the monitor which Rockwell provides, here is a brief summary of the I/O routines which BASIC uses.

Most of the AIM 65 functions which get data from the keyboard (i.e. Editor, BASIC, and even Assembler) do so by calling a monitor routine called INALL. INALL, however, is not just for accessing the keyboard. It will get a byte of data from ANY input device. Which device it goes to is determined by the contents of a memory location labelled INFLG, which is located at $A412. If this location contains a RETURN character ($OD) then the input will be from the keyboard. If INFLG contains an ASCII "T" (54), then INALL will look to the cassettes for data.

How does this location come to contain the proper value? The functions which allow a selection of input devices also make use of a subroutine from the monitor called WHEREI; it is this subroutine which displays the familiar "IN = " prompting message after the BASIC LOAD command is entered. If you respond to "IN = " with "T", the WHEREI routine then also asks for a file name ("F = ") and then finds out which cassette you will use ("T = "). From this time on, any time INALL is called, a byte of data from the specified tape file will be returned.

Output works in a similar fashion; there is a subroutine called OUTALL which will output a byte of data to any AIM 65 output device, depending on the contents of a location labelled OUTFLG, which is located at $A413. This location is set to the desired value by a subroutine called WHEREO, which is the one which generates the "OUT = " prompt.

Utilization

Making your BASIC programs read from cassettes is quite simple--most of the work has been done for you by the program logic used by the LOAD command. When you type LOAD and give the cassette file information, BASIC simply takes its input data from the tapes instead of from the keyboard, continuing to do this until a CONTROL Z character ($1A) is read from the tape. The CONTROL Z causes control to return to the keyboard. If your program contains a step with the LOAD command (for example, 100 LOAD) then when this step is executed, you will see the "IN = " message. If you specify input from a cassette file, then from that point on, until a CONTROL Z is read, or until INFLG is changed to a RETURN character, every INPUT statement in your program will take data from the tapes instead of from the keyboard.

When you reach a point in your program when you wish to switch input back to keyboard, simply POKE a RETURN into INFLG. If you want to intermix INPUTs from keyboard and tape, you can change the input device back and forth at any time by changing the contents of INFLG. Remember, though, that if your program bombs with an error while INFLG points to the tapes, the system will go on trying to get its data from the tape file; you will have to use the RESET button to regain control of the situation.

For writing data to cassettes, the procedure is a little more complex; there is no BASIC command which will change OUTFLG. The SAVE command will access the tapes, all right, but all it does is LIST the program and return to keyboard control. However, this sequence of steps will work:

1. POKE the address of the WHEREO routine into locations 4 and 5.
2. Execute a USR(X) statement. This will cause BASIC to call WHEREO.
3. Output data is required using regular PRINT statements.
4. When output is finished, you will need to close the file properly. Do this by PRINTING CHR$(13) and CHR$(26). This puts an AIM Editor end-of-file mark on the tape, followed by a CONTROL Z, just to be safe. Then POKE the address of the routine called DU11 (see table of locations) into locations 4 and 5, and execute a USR(X) statement. This will end the cassette file properly, and also will restore output to the display.

Notes and Cautions

If the OUTFLG is set to send output to tapes, and your program bombs with an error message for some reason, you will never see the error message — it will have been written to tape! For this reason, it is a good idea to debug programs using regular keyboard input and display output before using cassettes; also, it might be wise to "turn off" the cassettes when not actually reading or writing, by POKEing a RETURN into INFLG or OUTFLG after a statement which accesses tape. This allows you to intermix keyboard-display and cassette operation.

You can use both input and output in the same program, but unfortunately, NOT AT THE SAME TIME. The reason for this restriction is as follows: the monitor cassette routines store data on tape in 80-byte blocks. The data going to or from a block on tape is stored temporarily in a buffer area in memory. If INFLG and OUTFLG are both "T", then the cassette write routine uses a different buffer from that used by the read routine. This buffer is located on page zero, right in the middle of the area BASIC uses for its math operations. Therefore, if the same program is going to do both reading and writing, it must finish completely with one operation before it initializes the other. A procedure which eliminates this restriction (but requires assembly-language routines and some memory overhead) was reported in the first issue of Rockwell's new publication INTERACTIVE. The method used here is considerably simpler, but limits you to read-only or write-only at any given instant.

Sample Programs

The two sample programs were developed to fill a need in a project I was working on. The first writes a table of about 600 prime numbers to tape; the second program reads this table from tape into an integer array, and uses this array to print the factors of a number entered from the keyboard. While not elegant examples of the programmer's art, they do show the implementation of the procedures detailed here.

Location Table

Label	Hex	Decimal	Function
INFLG	A412	42002	Defines input device
OUTFLG	A413	42003	Defines output device
WHEREI	E848	59464 Low byte = 72 High byte = 232	Initialize INFLG
WHEREO	E871	59505 Low byte = 113 High byte = 232	Initialize OUTFLG
DU11	E50A	58634 Low byte = 10 High byte = 229	Close active tape file

List

```
0 REM SET UP OUTPUT TAPE FILE.
1 POKE 4,113: POKE 5,232
2 N = USR(N)
5 UL = 600: REM DEFINE TABLE LIMIT HERE
10 DIM X%(UL)
20 X%(1) = 2: X%(2) = 3
30 L = 2
90 N = 3
100 I = 1
110 IF INT(N/X%(I)<>N/X%(I) THEN 200
120 N = N + 2 GOTO 100
200 IF X%(I) = >SQR(N) THEN 300
210 I = I + 1: GOTO 110
300 L = L + 1: X%(L) = N
309 REM OUTPUT TO TAPE.
310 PRINT N
314 REM ALSO SHOW NUMBER ON DISPLAY.
315 POKE 42003,13: PRINT N: POKE 42003,ASC("T)
320 IF L<>UL THEN N = N + 2: GOTO100
321 REM
329 REM WRITE END-OF-FILE MARK ON TAPE
330 PRINT CHR$(13); CHR$(26)
331 REM
339 REM CLOSE TAPE WITH DU11 ROUTINE.
340 POKE 4,10: POKE 5,229
350 N = USR(N)
360 PRINT" DONE."

10 DIM A%(600)
20 A%(1) = 2
90 REM SET UP TAPE INPUT.
100 LOAD
115 REM READ DATA FROM TAPE TO ARRAY.
120 FOR I = 2 TO 597: INPUT A%(I): NEXT
125 REM TURN OFF TAPE.
130 POKE 42002,13
200 INPUT X
205 PRINT! "***";X
210 Q = 1
220 IF INT(X/A%(Q) = X/A%(Q) THEN 230
225 Q = Q + 1: GOTO 240
230 PRINT! A%(Q): X = A/A%(Q)
240 IF SQR(X) = >A%(Q) THEN 220
250 PRINT! X: GOTO 200
```

©

WANT YOUR COMPUTER BACK?

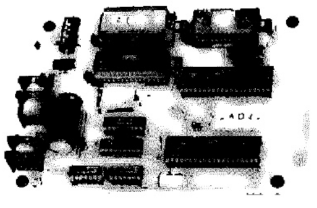
Let the MICROsport™ Microcomputer (MMC) take over any dedicated task.

It is the affordable alternative – kits from $89.00, application units from only $119.00 (assembled and tested).

It is user-oriented – complete in-circuit emulation allows program development on ANY 6502 based system. It is compact (4½" x 6½" pc board) but powerful (32 I/O lines; 20 mA full duplex, 1K RAM + EPROM socket 4/16 bit counters; 6503 CPU) and works off any AC or DC power supply.

Turn your present 6502 based system into a complete development system with:

1 MMC/03D Microcomputer with ZIF sockets
1 MMC/031CE In-circuit emulator for the 6503 CPU
1 MMC/03EPA EPROM Programmer complete with software driver.



For more info call or write

R. J. BRACHMAN ASSOCIATES, INC.
P.O. Box 1077
Havertown, PA 19083
(215) 622-5495