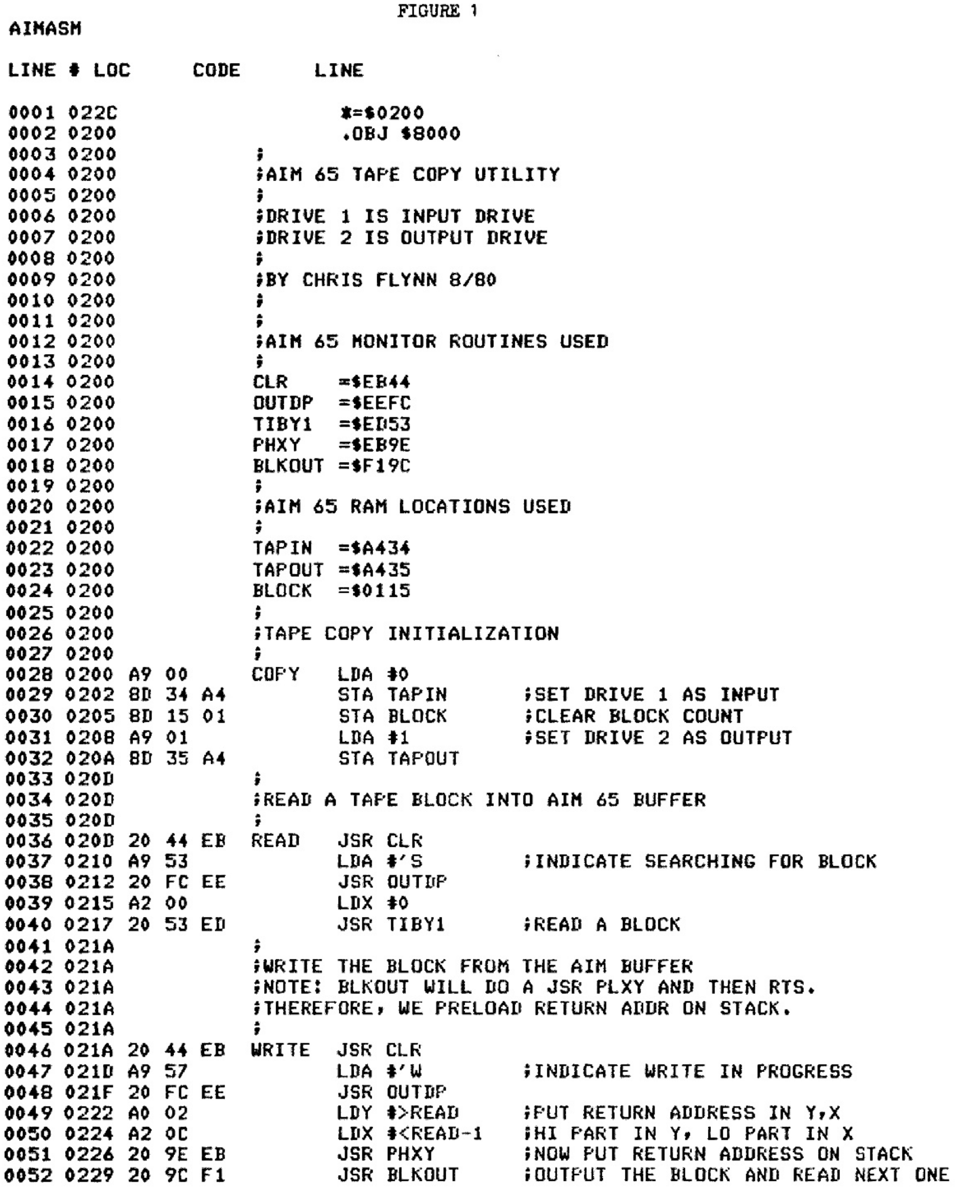
FIGURE 1



```
AIMASM

LINE # LOC      CODE        LINE

0001 022C                     *=$0200
0002 0200                     .OBJ $8000
0003 0200              ;
0004 0200              ;AIM 65 TAPE COPY UTILITY
0005 0200              ;
0006 0200              ;DRIVE 1 IS INPUT DRIVE
0007 0200              ;DRIVE 2 IS OUTPUT DRIVE
0008 0200              ;
0009 0200              ;BY CHRIS FLYNN 8/80
0010 0200              ;
0011 0200              ;
0012 0200              ;AIM 65 MONITOR ROUTINES USED
0013 0200              ;
0014 0200              CLR    =$EB44
0015 0200              OUTDP  =$EEFC
0016 0200              TIBY1  =$ED53
0017 0200              PHXY   =$EB9E
0018 0200              BLKOUT =$F19C
0019 0200              ;
0020 0200              ;AIM 65 RAM LOCATIONS USED
0021 0200              ;
0022 0200              TAPIN  =$A434
0023 0200              TAPOUT =$A435
0024 0200              BLOCK  =$0115
0025 0200              ;
0026 0200              ;TAPE COPY INITIALIZATION
0027 0200              ;
0028 0200 A9 00        COPY   LDA #0
0029 0202 8D 34 A4            STA TAPIN       ;SET DRIVE 1 AS INPUT
0030 0205 8D 15 01            STA BLOCK       ;CLEAR BLOCK COUNT
0031 0208 A9 01               LDA #1          ;SET DRIVE 2 AS OUTPUT
0032 020A 8D 35 A4            STA TAPOUT
0033 020D              ;
0034 020D              ;READ A TAPE BLOCK INTO AIM 65 BUFFER
0035 020D              ;
0036 020D 20 44 EB     READ   JSR CLR
0037 0210 A9 53               LDA #'S         ;INDICATE SEARCHING FOR BLOCK
0038 0212 20 FC EE            JSR OUTDP
0039 0215 A2 00               LDX #0
0040 0217 20 53 ED            JSR TIBY1       ;READ A BLOCK
0041 021A              ;
0042 021A              ;WRITE THE BLOCK FROM THE AIM BUFFER
0043 021A              ;NOTE: BLKOUT WILL DO A JSR PLXY AND THEN RTS.
0044 021A              ;THEREFORE, WE PRELOAD RETURN ADDR ON STACK.
0045 021A              ;
0046 021A 20 44 EB     WRITE  JSR CLR
0047 021D A9 57               LDA #'W         ;INDICATE WRITE IN PROGRESS
0048 021F 20 FC EE            JSR OUTDP
0049 0222 A0 02               LDY #>READ      ;PUT RETURN ADDRESS IN Y,X
0050 0224 A2 0C               LDX #<READ-1    ;HI PART IN Y, LO PART IN X
0051 0226 20 9E EB            JSR PHXY        ;NOW PUT RETURN ADDRESS ON STACK
0052 0229 20 9C F1            JSR BLKOUT      ;OUTPUT THE BLOCK AND READ NEXT ONE
```

The listing in Figure 1 shows the assembly language code for the tape copy program. The only tricky part of the program is the JSR to BLKOUT. BLKOUT is really a part of the AIM subroutine TOBYTE ($F18B). A problem arises because the tape copy program calls TOBYTE at a point other than its normal entry point.

The first and last two statements of TOBYTE are:

JSR PHXY
●
●

```
●
JSR PLXY
RTS
```

Notice that TOBYTE saves the X and Y registers on the stack. When TOBYTE is called in the middle, the X and Y registers do not get saved. So, when TOBYTE finishes, the JSR PLXY does not pick up X and Y. Instead, it removes the return address from the stack. Therefore, the RTS picks up garbage from the stack and the AIM hangs!

To get around this problem, the tape copy program preloads X and Y before calling BLKOUT. The values loaded into X and Y represent the return address. X and Y are then stored on the stack. Lastly, the JSR to BLKOUT is done.

Figure 1 shows the way X and Y are loaded. The most significant byte of (return address - 1) is placed in Y. The least significant byte of (return address - 1) is placed in X. One is subtracted from the return address in order to mimic the way the 6502 stores return addresses on the stack. If you relocate this program, you will have to load X and Y with the appropriate values.

Summary

This article has described a simple tape copy utility for the AIM 65. I hope that you find it both useful and easy to use. ©

AIM 65 Tape Copy Utility

Christopher J. Flynn

Introduction

If you're an AIM 65 user, you've probably stored your favorite programs and important data bases on cassette tape. Have you thought about making backup copies of your tapes? I didn't until my tape recorder ate my only copy of a 1000 line assembly language program that I was writing.

You may be thinking it is too much trouble to make backup tapes on the AIM. Each file has to be loaded into memory and then written back out. If you have machine language programs, Basic programs, and text files, then you have to follow three different load and dump procedures. Machine language programs are the worst to copy. Sure, it is very easy to load them into memory. Have you tried dumping such a program when you've lost the little piece of paper that had the memory addresses on it?

Well, here is a little 44 byte program that will make tape copying easy. All you do is put the tape to be copied in drive 1 and a blank tape in drive 2. Then, position the tapes and let the program do the rest. The program will copy any kind of AIM file. It will even copy multiple input files from the same tape. So now, none of us should have any excuse for not having backup copies of our important tapes.

Hardware Required

First of all, I'll assume that you have an AIM. An AIM with just 1K of RAM will do fine.

Next, you'll need to attach two cassette recorders to your AIM. Chances are you already have one. If nothing else, maybe this article will give you an excuse to buy a second one. By the way, the versatility of the AIM definitely improves with the second recorder.

Finally, you should connect the remote control circuits to each of the recorders. You should experiment with the setting of GAP ($A409) as described in the AIM manual. Pick a value of GAP that lets you record on one device and play back on the other reliably. I have found that the default value of $08 works well. It only worked, however, after I modified my recorders (Radio Shack) so that their electronics would remain on even when the motor was toggled off.

Tape Copy Procedure

Let's go through the step by step procedure of copying a tape.

1. Load the tape copy program into the AIM's memory starting at $0200. The program is easily relocated, but you'll have to observe the cautions described in a later section.
2. Place the tape to be copied in drive 1. This program assumes that drive 1 is used only in the playback mode.
3. Place a blank tape in drive 2. This program assumes that drive 2 is used only in the record mode.
4. Position the tapes.
   a. Position the tape in drive 1 to a point just beyond the leader. Use the "1" monitor command to toggle drive 1 off.
   b. Position the tape in drive 2 to a point about 4 turns beyond the leader. Use the monitor "2" command to toggle drive 2 off.
5. Start the tape copy program.
   a. Use the monitor "*" command to set the AIM's program counter to $0200.
   b. Use the monitor "G" command to begin the program.
6. Watch the AIM display. The display will alternately show an "S" and a "W". The "S" means that the program is searching for the next block. The "W" means that the program is in the process of writing a block to drive 2.
7. Hit reset to stop the copy program when a steady display of "S" appears without any intervening "W"s.
   a. Drive 1 will be on and you can rewind and remove the input tape.
   b. Drive 2 will be off. This allows you to stack additional programs or data on the same output tape. You will have to toggle drive 2 with the "2" command when you are ready to rewind the output tape.

That's all there is to copying a tape. Notice that at no time did the AIM ask you "IN = " or "OUT = ". It did not even ask you for the input and output file names.

By the way, you should probably verify the first few tape copies that you make just to be sure that the program works and that GAP is set properly.

How It Works

The Tape copy program makes use of subroutines in the AIM monitor. Basically, the program reads a data block from drive 1 (subroutine TIBY1 at $ED53) into the AIM's tape buffer. The data block is then written from the buffer to drive 2 by an AIM subroutine beginning at $F19C which I've called BLKOUT. In between data blocks, the program writes either an "S" or a "W" to the AIM display. This process of reading and writing a block continues forever or until reset is pushed or the plug is pulled.