DLOAD : AIM Memory Loader

Joel Swank
Rockaway, OR

The AIM 65 monitor gives you the ability to save and load non-contiguous blocks of memory on cassette tape, paper tape, or a user device. This handy feature allows you to save a program, along with any vectors or data values it needs to execute, and then to load it all with just one command. The AIM assembler uses the same format for its object files. You can assemble several programs at different locations and load them all with one load command.

But the AIM load command is missing a couple of useful features. When loading a file with the AIM load command there is no way to tell which memory locations are being loaded. After assembling a program there is no way, without a listing, to tell where the program ends. You might also want to assemble a program at one address and load it into memory at a different address, as in the case of a program that is to reside in ROM. It would also be convenient to be able to save data from one area of memory and load it back to a different area. The AIM load command cannot do this.

DLOAD is a modified version of the AIM load command that adds these two missing features. DLOAD works like the AIM load command except that it first requests an offset with the 'OFFSET=' prompt. This hexidecimal number is input with the AIM subroutine ADDIN. ADDIN is the same routine that inputs your reply to the 'FROM=' and 'TO='

```
0000
0000          ;  DLOAD : LOAD AIM OBJECT FILE AND DISPLAY THE
0000          ;           ADDRESSES OF THE DATA LOADED. OPTIONAL
0000          ;           OFFSET LOAD.

0000          ;   AIM SUBROUTINES

0000          WHEREI =$E848          ;OPEN INPUT
0000          INALL  =$E993          ;INPUT A CHAR FROM AOD
0000          CLRCK  =$EB4D          ;CLEAR CHECKSUM
0000          CHEKAR =$E54B          ;INPUT HEX BYTE
0000          CKERR  =$E385          ;ERROR RETURN
0000          RBYTE  =$E3FD          ;READ OBJECT BYTE
0000          STBYTE =$E413          ;STORE OBJECT BYTE (ADDR)
0000          DU13   =$E520          ;CLOSE TAPE INPUT FILE
0000          ADDIN  =$EAAE          ;INPUT ADDRESS FROM KBD
0000          RCHEK  =$E907          ;CHECK FOR INTERRUPT
0000          WRITAZ =$E2DB          ;DISPLAY CONTENTS OF ADDR
0000          COMIN  =$E1A1          ;NORMAL RETURN TO AIM
0000          NUMA   =$EA46          ;DISPLAY BYTE IN HEX
0000          OUTPUT =$E97A          ;DISPLAY ACCUM
0000          CRLF   =$E9F0          ;SEND CR AND LF

0000          ;   AIM RAM

0000          ADDR   =$A41C          ;OBJECT LOAD POINTER
0000          CKSUM  =$A41E          ;CHECKSUM STORAGE
0000          CURAD  =$A41C          ;ADDRESS INPUT BUFFER

0000          ;   ZERO PAGE

0000          OFFL   =0              ;OFFSET SAVE AREA
0000          OFFH   =1
0000          POINTL =2              ;DUPLICATE LOAD POINTER
0000          POINTH =3
0000          RECLEN =4              ;RECORD LENGTH SAVEAREA

0000                 *=$200

0200 A000     DLOAD  LDY #OFFMSG-LITS
0202 20D502           JSR KEPX        ;DISPLAY 'OFFSET='
0205 20AEEA           JSR ADDIN       ;INPUT ADDRESS
0208 B0F6             BCS DLOAD       ;ERROR - TRY AGAIN
020A AD1EA4           LDA CKSUM       ;ANY ENTERED?
020D F005             BEQ SAVOFF      ;YES, SAVE IT
020F A900             LDA #0
0211 8D1DA4           STA CURAD+1     ;NO, USE ZERO
0214 AD1CA4   SAVOFF  LDA CURAD       ;COPY CURAD TO OFFSET
0217 8500             STA OFFL
0219 AD1DA4           LDA CURAD+1
021C 8501             STA OFFH

021E 2048E8   OPFIL   JSR WHEREI      ;OPEN INPUT DEVICE
0221 207C02           JSR STREC       ;START RECORD
0224 20A602           JSR PSTART      ;DISPLAY START ADDRESS
0227 4C4502           JMP BYTLUP

022A 207C02   RECLUP  JSR STREC       ;START RECORD
022D A604             LDX RECLEN      ;ZERO LENGTH RECORD?
022F F037             BEQ FINISH      ;YES, END
0231 AD1CA4           LDA ADDR        ;IS NEW ADDRESS EQUAL
0234 C502             CMP POINTL      ;TO OLD ADDRESS?
0236 D007             BNE NEWLOC      ;NO, NEW BLOCK OF MEMORY
0238 AD1DA4           LDA ADDR+1
023B C503             CMP POINTH
023D F006             BEQ BYTLUP

023F 20BC02   NEWLOC  JSR PEND        ;DISPLAY END OF LAST RECORD
0242 20A602           JSR PSTART      ;AND START OF THIS ONE

0245 20FDE3   BYTLUP  JSR RBYTE       ;INPUT AN OBJECT BYTE
0248 2013E4           JSR STBYTE      ;STORE IT
024B E602             INC POINTL      ;BUMP POINTER
024D D002             BNE NOCY
024F E603             INC POINTH
0251 C604     NOCY    DEC RECLEN      ;COUNT BYTE
0253 D0F0             BNE BYTLUP      ;DO NEXT BYTE

0255          ; END OF RECORD
0255 20FDE3           JSR RBYTE       ;GET CHECKSUM
0258 CD1FA4           CMP CKSUM+1     ;AND COMPARE
025B D008             BNE ERROUT      ;ERROR IF NOT EQUAL
025D 20FDE3           JSR RBYTE
0260 CD1EA4           CMP CKSUM

0263 F0C5             BEQ RECLUP      ;CHECKSUM OK - NEXT RECORD

0265 4C85E3   ERROUT  JMP CKERR       ;ERROR EXIT

0268 20BC02   FINISH  JSR PEND        ;PRINT ADDRESS OF LAST RECORD
026B A205             LDX #5
```

prompts, so the syntax is the same. The hexadecimal number you enter is added to the starting address of each block of memory in the file. For example, a block that was saved from location $200 can be loaded back at location $1000 by replying 'E00' to the 'OFFSET=' prompt. You can calculate the proper offset by: $1000-$200=$E00. You can also load a file to a location lower in memory by adding $10000 to the desired load address before performing the calculation. A file dumped from location $B000 can be loaded back at $200 as follows: $10200-$B000=$5200. Enter '5200' in response to the 'OFFSET=' prompt. If the file contains multiple blocks, then the offset is added to the starting address of all blocks. This means you must take care when loading a file containing vectors or zero page data. These blocks will also be displaced by the offset you entered. You may load a file to its original address by entering a space or return in response to the 'OFFSET=' prompt.

DLOAD next issues the standard AIM 'IN=' prompt to open the input device. You respond as you normally would when using the AIM load command. DLOAD then displays the start and end addresses of each contiguous block of memory as it is loaded. If you are using an offset, the addresses displayed are those at which the data is being stored and not the addresses in the file. DLOAD calls the AIM RCHEK subroutine at the start of each data block so that you can stop or cancel the program. DLOAD used zero page memory locations 0-4, so be sure not to try to load anything there. Included is a listing of DLOAD assembled at location $200. DLOAD can be executed from ROM.

```
026D 20FDE3  FLUP   JSR RBYTE       ;READ END OF LAST RECORD
0270 CA              DEX
0271 D0FA            BNE FLUP
0273 2093E9          JSR INALL
0276 2020E5          JSR OUT3       ;CLOSE TAPE
0279 4CA1E1          JMP COMIN      ;RETURN TO MONITOR

027C            ;    END OF MAINLINE

027C            ;    SUBROUTINES FOLLOW

027C            ;   STREC : INPUT BEGINNING OF RECORD

027C 2007E9  STREC  JSR RCHEK       ;CHECK FOR INTERRUPT
027F 2093E9         JSR INALL       ;SEARCH FOR ';'
0282 C93B           CMP #';'
0284 D0F6           BNE STREC
0286 204DEB         JSR CLRCK       ;CLEAR CHECKSUM
0289 204BE5         JSR CHEKAR      ;GET RECORD LENGTH
028C 8504           STA RECLEN      ;SAVE IT
028E 204BE5         JSR CHEKAR      ;GET RECORD ADDRESS
0291 8D1DA4         STA ADDR+1      ;AND SAVE
0294 204BE5         JSR CHEKAR
0297 18             CLC
0298 6500           ADC OFFL        ;ADD OFFSET
029A 8D1CA4         STA ADDR
029D AD1DA4         LDA ADDR+1
02A0 6501           ADC OFFH
02A2 8D1DA4         STA ADDR+1
02A5 60             RTS

02A6            ;   PSTART : DISPLAY STARTING ADDRESS OF MEMORY BLOCK

02A6 20F0E9  PSTART JSR CRLF        ;NEW LINE
02A9 A007           LDY #STMSG-LITS
02AB 20D502         JSR KEPX        ;DISPLAY 'START='
02AE 20DBE2         JSR WRITAZ      ;DISPLAY ADDRESS
02B1 AD1CA4         LDA ADDR        ;COPY ADDR TO POINT
02B4 8502           STA POINTL
02B6 AD1DA4         LDA ADDR+1
02B9 8503           STA POINTH
02BB 60             RTS

02BC            ;   PEND : DISPLAY ENDING ADDRESS OF MEMORY BLOCK

02BC A00E    PEND   LDY #ENDMSG-LITS
02BE 20D502         JSR KEPX        ;DISPLAY ' END='
02C1 38             SEC
02C2 A502           LDA POINTL      ;DECREMENT LAST ADDRESS
02C4 E901           SBC #1
02C6 8502           STA POINTL
02C8 A503           LDA POINTH
02CA E900           SBC #0
02CC 2046EA         JSR NUMA        ;DISPLAY ADDRESS
02CF A502           LDA POINTL
02D1 2046EA         JSR NUMA
02D4 60             RTS

02D5            ;   KEPX : DISPLAY MESSAGE FROM LITERAL TABLE

02D5 B9E102  KEPX   LDA LITS,Y      ;GET A BYTE
02D8 F006           BEQ RETURN      ;QUIT ON NULL
02DA 207AE9         JSR OUTPUT      ;DISPLAY IT
02DD C8             INY             ;NEXT CHARACTER
02DE D0F5           BNE KEPX
02E0 60      RETURN RTS

02E1            ;   LITERAL TABLE

02E1            LITS   =*

02E1 4F46    OFFMSG .BYTE 'OFFSET',0
02E3 00

02E8 5354    STMSG  .BYTE 'START=',0
02EA 00
02EF 2045    ENDMSG .BYTE ' END=',0
02F1 00

02F2                .END
02F2          ERRORS= 0000
```