A A R D V A R K

DATA SHEET

SUBJECT: USE OF 65D DISK DATA FILES FOR EDITING BASIC PROGRAMS

INTRODUCTION

You are not limited to using the "indirect file" for merging or otherwise editing BASIC programs on 65D systems. By using the LIST#<device#>,<line spec> command and naming "6" or "7" as the device#, you can LIST any sub set of program lines from the work space to a disk data file - in the same manner as you would LIST to the console or to a printer. You are not limited to one LIST command either, as you are with the indirect file. You can invoke multiple LIST commands from one or more programs LOADed into the work space. You can also invoke PRINT commands to write data to the disk file. Each LIST and PRINT command to device# 6 or 7 will append the associated data lines to the data file, as long as the number of tracks assigned to the file will hold the data.

In the reverse direction, you can enter BASIC statements and BASIC commands from a disk data file to the work space by invoking a DISK!"IO 20" or a DISK!"IO 40" command depending on whether the data file was "OPENED" on device# 6 or 7 respectively. The DISK!"IO 20" command makes the data file assigned to device# 6 the console input device. It is the same as if you entered the contents of the data file directly from the normal console keyboard. The computer will continue to take its "commands" from the disk device until another DISK!"IO xx" command or an invalid BASIC command is encountered. Any invalid command read from the data file will cause the computer to take all subsequent commands from the permanent console input device, namely the console keyboard.

GENERAL PROCEDURE

The basic steps required for program editing using 65D disk data files are as follows:

1. Establish a disk file memory buffer that will not conflict with the editing process.
2. Build an edited set of program lines on a disk data file.
3. Merge the edited data file back into the work space and "save" the results.

In the subsequent discussions, device# 6 will be used as the the disk device for input/output to disk data files. Device# 7 could have been used just as well.

ESTABLISH A DISK BUFFER

The normal location for the disk memory buffer for device# 6 is the beginning of the BASIC work space. If each of the programs that you may want to include in your editing is created and "saved" with at least one disk memory buffer, then your editing will not conflict with the disk buffer. However, there is no need to worry about what programs do or do not use disk buffers. If you use a buffer for editing that is located outside of the work space, then any program can be included in the edit operations.

Appendix A includes a listing of a program that sets up a disk memory buffer for device# 6 located at the top of your usable RAM. By RUNning this program with option "0" at the beginning of an edit session and RUNning it with option "1" at the end, you can edit to your heart's content without concern for buffer interference.

The followings procedures assume that a device# 6 buffer has been established outside of the work space.

BUILD AN EDITED FILE

The step by step procedure to build a data file of edited program lines for reentry into the work space is as follows:

#	Step Description	Command Syntax
1.	Open a data file	DISK OPEN,6,"<filename>"
2.	Load a program for LISTing	DISK!"LO <program name>"
3.	List a sub set of lines to the data file	LIST#6 or LIST#6,<line spec>
4.	Repeat #3 until all sub sets for the LOADed file are listed.	
5.	Repeat #2 thru #4 until all programs involved are processed.	
6.	Print an invalid BASIC command as an "end of file" mark.	PRINT#6,".END"
7.	Assure the disk buffer is written to the disk.	DISK CLOSE,6
8.	Edit the data file contents, as necessary, using the AARDVARK ASCII FILE/TEXT editor.	

MERGE EDITED DATA FILE INTO THE WORK SPACE

Once the data file contains the program lines that are to replace the "old" program or that are to be merged with the "old" program, then we are ready to "merge" the file into the work space. The steps to accomplish this are as follows:

#	Step Description	Command Syntax
	*	
1.	Load the program to be replaced or modified	DISK!"LO <program name>"
2.	Open the data file	DISK OPEN,6,"<filename>"
3.	If the program is to be replaced, then clear the work space	NEW or <NULL> *
4.	Enter the data file contents into the work space	DISK!"IO 20"
5.	When the entry is complete, "save" the results.	DISK!"PUT <program name>"

* It is wise to always LOAD the old program even if it is going to be replaced by the contents of the edited data file, in order to assure the same allocation of disk buffers. This procedure is not necessary, of course, if the original disk buffer allocation was wrong. In that case, RUN "CHANGE", in place of step #1 above, to assign the proper number of buffers before invoking the NEW command.

After all your editing is done, remember to restore the normal device# 6 disk buffer by RUNning the program shown in Appendix A with option "1". There is no problem with having the disk buffer at the top of memory, as long as it is outside the work space and the buffer area is not used by any of your programs. Re-booting the system will also restore the normal buffer assignments.

EDITING EXAMPLES

Once you have mastered the basic mechanics for using 65D disk data files for program editing, the following may be helpful in solving specific editing problems.

BLOCK LINE DELELETES - by Omission

Problem: Delete lines 200-399

Solution: Command Sequence

```
DISK!"LO PROG
DISK OPEN,6,"SCRDAT"
LIST#6,-199
LIST#6,400-
PRINT#6,".END"
DISK CLOSE,6
NEW
DISK OPEN,6,"SCRDAT"
DISK!"IO 20"
DISK!"PUT PROG
DISK CLOSE,6
```

BLOCK LINE DELETES - by Commission

Problem: Delete lines 200-399

Solution: Command Sequence

```
DISK OPEN,6,"SCRDAT
FOR I=200TO399:PRINT#6,I:NEXTI:PRINT#6,".END
DISK CLOSE,6
DISK!"LO PROG
DISK OPEN,6,"SCRDAT
DISK!"IO 20"
DISK!"PU PROG
DISK CLOSE,6
```

ADD COMMON SUBROUTINES TO A NEW PROGRAM

EXAMPLE COMMAND SEQUENCE:

```
DISK!"LO SUBLIB":REM GET SUBROUTINE LIBRARY
DISK OPEN,6,"SCRDAT
LIST#6,1000-1999:REM 1ST SUBROUTINE USED
LIST#6,4000-4999:REM 2ND SUBROUTINE USED
  .
  .
 etc.
PRINT#6,".END
DISK CLOSE,6
DISK!"LOAD NEWPRG":REM GET THE NEW MAIN PROGRAM
DISK OPEN,6,"SCRDAT
DISK!"IO 20":REM MERGE THE SUBROUTINES INTO MAIN PROGRAM
DISK!"PUT NEWPRG":REM SAVE THE RESULTS
DISK CLOSE,6
```

CORRECT A SUBROUTINE USED BY MANY PROGRAMS

EXAMPLE COMMAND SEQUENCE:

```
DISK!"LO SUBR":REM GET DEFECTIVE SUBROUTINE
(CORRECT THE SUBROUTINE IN THE WORK SPACE)
DISK!"PUT SUBR":REM SAVE THE CORRECTED VERSION
DISK OPEN,6,"SCRDAT
LIST#6:REM LIST THE WHOLE SUBROUTINE TO THE DATA FILE
PRINT#6,<line number>:REM RECORD ANY DELETED LINES
  .
  .
 etc.
PRINT#6,".END
DISK CLOSE,6
DISK!"LOAD MAINx":REM FIX EACH MAIN PROGRAM
DISK OPEN,6,"SCRDAT
DISK!"IO 20"
DISK!"PUT MAINx":REM SAVE THE CORRECTED RESULT
DISK CLOSE,6
(repeat from LOAD to CLOSE for each affected program)
```

Note that this last example task might be done quicker using the indirect file - once the corrected subroutine is LISTed to the indirect file, it requires only three commands per affected program: 1. DISK!"LO MAINx" 2. <CTRL-X> and 3. DISK!"PUT MAINx". However, you cannot "PRINT" to the indirect file, so you need another remedy to remove any lines that were deleted to correct the subroutine. One way is to replace all deleted lines with "REM" lines, which is not too satisfactory. Or, you could use the "data file" method above.

APPENDIX A

```
2 X=PEEK(9006)+PEEK(9007)*256
4 IF X=15742THENBASE=12670:SIZE=3072:GOTO10
6 IF X=14974THENBASE=12926:SIZE=2048:GOTO10
8 PRINT"PROGRAM ANOMOLY IN LINE 8":END
10 POKE2888,0
20 INPUT"DEVICE#6 BUFFER: 0=MOVE IT, 1=RESTORE IT";X
30 IFX=0GOTO100
50 MEM=BASE:GOTO300
100 MEM=(PEEK(8960)+1)*256
200 MEM=MEM-SIZE
300 I=8998:GOSUB1000
400 MEM=MEM+SIZE:I=9000:GOSUB1000
450 PRINT"BUFFER LOCATION:";PEEK(8998)+PEEK(8999)*256
500 POKE2888,27:END
1000 HI=INT(ME/256):LO =MEM-HI*256
1100 POKEI,LO:POKEI+1,HI:RETURN
1200 END
```