Relative Files

Relative files represent the major advantage of disk systems over conventional sequential storage devices (i.e. cassette, etc.). Relative or Random Access files can be accessed in any order by the specification of a record number. Every relative file is composed of a series of records numbered 0 through a maximum of 32767. Realistically, the maximum record number is not obtainable utilizing the disk system available for the 99/4A. In order to create a relative file, the OPEN statement for this file must contain RELATIVE in its file description list; furthermore, since relative files must have a fixed length, the descriptor FIXED must also be included. An example of a correct OPEN statement follows:

```
100 OPEN #1:"DSK1.DATA", RELATIVE, FIXED, INTERNAL
```

When printing to relative file a record option may be specified. This option serves to direct the data to be printed to a specific record in the file. An example of a PRINT STATEMENT to a relative file follows:

```
110 print #1,REC 7:B,A$
```

In the statement above, data consisting of a numeric variable B and a string variable A$ are to be printed to record 7 of the file opened as #1. With the exception of a pending print, each PRINT statement prints a complete record to a file. Since each PRINT statement prints a complete record, care must be taken to ensure that the record size allocated in the OPEN statment is of sufficient size to accomodate the data stored in each record. In the above example the file size is set to the default size of 80 bytes; therefore, the two variables printed to this file must have a combined size of 80 or fewer bytes. In order to facilitate the estimating of record length, the following guidelines may be used:

> For numeric data, 9 bytes of disk storage must be allocated, variable used. Therefore, the variables A,B,C will occupy 3 * 9 or 27 bytes in the file.
>
> String data occupies one byte plus the length of the sring. Represented in BASIC format this would be: BYTES=1+LEN(STRING$) For example, A$="EXAMPLE STRING" would occupy 1+14 or 15 bytes in the file.

With the above limitations in mind it is easy to see why the record length selected is critical. If a data in a PRINT statement will occupy 83 bytes in a file and the file is only allocated for 80 bytes the data written will not be able to be retrieved. It is especially important when using relative records to observe the record size limitations since individual records will be accessed.

It is also very important to remember that the input statement used to retrieve the data must have the same number and type of variables as were PRINTed. For example, if we wish to retrieve the

data output by the PRINT statement on line 110, then an INPUT statement similar to the following must be used:

```
120 INPUT #1,REC 7:NUM, NAME$
```

The data stored on the file consists of a numeric variable and a string variable in that order. The order of the variables in the record is critical, for it is the only means we have for determining what the variable is to be used for. Take note of the fact that different variable names were used when INPUTing the data than were used when PRINTing. As is implied by this, only raw data is stored in the file and not any of the attributes such as variable names or usage.

In the following program, relative records are used exclusively to store the information. Each record can be no longer than 80 chars (the 80 byte default record size is used) and a maximum of 50 records can be set up. A worthwhile experiment would be to increase the record size to 150 chars as well as increasing the number of records allocated to 100. Information on the organization of disk files is located on pages 30-36 of the Disk Memory System manual.

```
100 REM  ********************
110 REM  * THE FOLLOWING    *
120 REM  * PROGRAM DEMON-   *
130 REM  * STRATES RELATIVE*
140 REM  * FILE ACCESS WITH*
150 REM  * DISK DRIVES      *
160 REM  ********************
170 CALL CLEAR
180 PRINT "  PRESS:":::
190 PRINT "    1) TO CREATE A NEW FILE":::
200 PRINT "    2) RELOAD EXISTING FILE":::
210 PRINT "    3) EXIT PROGRAM":::::::::::
220 INPUT "CHOICE: ":ANS
230 ON ANS GOTO 240,440,1240
240 REM   ********************
250 REM    CREATE A FILE
260 REM   ********************
270 CALL CLEAR
280 INPUT "HOW MANY FILES? ":FILNUM
290 PRINT "ENTER FILENAME ONLY "
300 INPUT FILE$
310 PRINT ::
320 OPEN #1:"DSK1."&FILE$,RELATIVE,INTERNAL,FIXED,OUTPUT
330 FOR X=1 TO FILNUM
340 PRINT ::
350 PRINT "ENTRY #";X
360 INPUT "ACCT NAME: ":NAME$
370 INPUT "ACCT NUMBER: ":NUM$
380 INPUT "ACCT BALANCE: ":BAL
390 PRINT #1,REC X:NAME$,NUM$,BAL
400 NEXT X
410 PRINT #1,REC 0:FILNUM
420 CLOSE #1
430 GOTO 170
440 REM  ********************
450 REM   SELECT A FILE
460 REM  ********************
470 CALL CLEAR
480 PRINT "ENTER FILENAME ONLY"
490 INPUT FILE$
500 PRINT ::
510 OPEN #1:"DSK1."&FILE$,RELATIVE,INTERNAL,FIXED
520 REM  *READ # OF RECORDS*
530 INPUT #1,REC 0:FILNUM
540 REM  ********************
550 REM   FILE MANIPULATION
560 REM   MENU
570 REM  ********************
580 CALL CLEAR
590 PRINT "  PRESS:"::
600 PRINT "    1) ADD RECORDS"::
610 PRINT "    2) CHANGE A RECORD"::
620 PRINT "    3) REVIEW RECORDS"::
630 PRINT "    4) RETURN TO MASTER"
640 PRINT "       SELECTION LIST"::::::::
650 INPUT "CHOICE ":ANS
660 IF (ANS<1)+(ANS>4)THEN 540
670 ON ANS GOTO 680,820,960,1220
680 REM  ********************
690 REM   ADD RECORDS
700 REM  ********************
710 CALL CLEAR
720 PRINT "ENTRY #";FILNUM+1
730 INPUT "ACCT NAME: ":NAME$
740 INPUT "ACCT NUMBER: ":NUM$
750 INPUT "ACCT BALANCE: ":BAL
760 REM   *PRINT NEW REC*
770 PRINT #1,REC(FILNUM+1):NAME$,NUM$,BAL
780 REM   *RESET REC NUM*
790 FILNUM=FILNUM+1
800 PRINT #1,REC 0:FILNUM
810 GOTO 540
820 REM  ********************
830 REM   CHANGE RECORDS
840 REM  ********************
850 CALL CLEAR
860 PRINT "ENTER REC # TO CHANGE"
870 PRINT "ENTER REC # FROM 1 TO";FILNUM
880 INPUT X
890 IF (X<1)+(X>FILNUM)THEN 850
900 GOSUB 1090
910 INPUT "NEW ACCT NAME:   ":NAME$
920 INPUT "NEW ACCT NUMBER: ":NUM$
930 INPUT "NEW ACCT BAL:    ":BAL
940 PRINT #1,REC X:NAME$,NUM$,BAL
950 GOTO 540
960 REM  ********************
970 REM   REVIEW RECORDS
980 REM  ********************
990 CALL CLEAR
1000 PRINT ::
1010 PRINT "ENTER REC # FROM 1 TO";FILNUM
1020 PRINT "ENTER 0 TO EXIT"
1030 INPUT X
1040 IF X=0 THEN 580
1050 IF X>FILNUM THEN 1010
1060 IF X<1 THEN 1010
1070 GOSUB 1090
1080 GOTO 960
1090 REM  ********************
1100 REM   BRING RECORD
1110 REM   FROM FILE
1120 REM  ********************
1130 INPUT #1,REC X:NAME$,NUM$,BAL
1140 CALL CLEAR
1150 PRINT :::::"**************************
1160 PRINT "RECORD";X:
1170 PRINT "NAME     ";NAME$
1180 PRINT "ACCT #   ";NUM$
1190 PRINT "ACCT BAL";BAL
1200 PRINT "**************************":::::
1205 INPUT "PRESS 'ENTER' TO CONTINUE":ENT$
1210 RETURN
1220 CLOSE #1
1230 GOTO 170
1240 END
```

DISK FILES

File Statement Definitions

OPEN- general form:
OPEN #file-number:"DSK1.filename",(organization),(file-type),(open-mode),(record-type)

file-number- numeric value assigned to the file. May be any number between 1 and 255, and cannot be duplicated without danger of overwriting existing files.

file-name- any string expression which evaluates to a valid file name. File-name must be enclosed in quotes, or be a valid, predefined string variable.

organization-deals primarily with accessibility to files. May be defined as SEQUENTIAL for sequential accessing, or RELATIVE for random accessing. If you do not specify file organization, file will default to SEQUENTIAL.

file-type- designates format in which the file is stored. Files may be in "easy to read" ASCII form or in machine readable form. The ASCII format, or DISPLAY format, is used if the information is to be printed or displayed. The binary records use space, but cannot be read except by the computer. To specify binary format, enter INTERNAL into the file-type. The default for file-type is DISPLAY. (Note- If using DISPLAY type files, see Addendum)

open-mode- this step instructs the computer to process files in INPUT, OUTPUT, UPDATE, or APPEND mode. INPUT files may only be read, OUTPUT files may only be written to, UPDATE files may be read and written to, and APPEND files may be added to only.

If the file is protected, it may not be written to and can only be opened for input. Also the APPEND mode can only be utilized with VARIABLE length records. If you don't specify an open-mode, the default for disk files is UPDATE.

record-type- specifies the length of the files to be used. A VARIABLE length record usually takes up more space and as the title indicates, it may vary in size. A FIXED length record is processed faster than files saved under VARIABLE record type, and generally take up less space. If you do not specify a record length, the default is 80.

CLOSE- closes communication between program and the device. After this statement is executed the file is no longer available for manipulation unless the file is reopended.

PRINT- allows you to write data onto a disk file. This command may only be used with files opened in OUTPUT, UPDATE, or APPEND mode.

INPUT- allows you to read data from an accessory device. May only be used with files opened in INPUT or UPDATE mode.

RESTORE-allows positioning at a specific record on a file. You may use the RESTORE statements with RELATIVE files.

Program Example

1) Type in the attached program and store it on a diskette using the SAVE command. An example of this statement would be:

```
SAVE DSK1.CHECKBOOK
```

2) Run the program and select the appropriate choice from the menu.
   (1) To create a check file
   (2) To load the check file from diskette

3) When creating a data file you are prompted to input the following:

```
Check # ?
Check Amount ?
Payable To ?
```

The program will then prompt you to type in ten checks you have written and would like to store on a disk file. The following is a few examples of the required input:

```
101
15.00
Texas Instruments

102
45.00
Albertsons

103
185.00
Rent
```

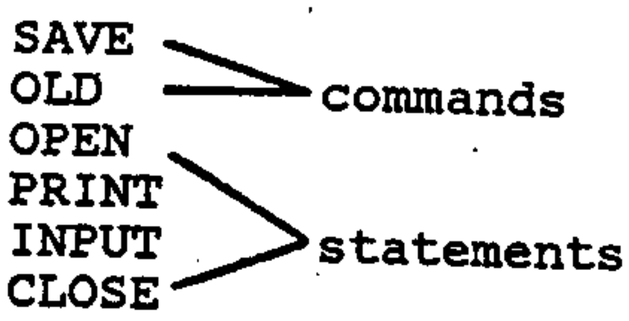
4) After inputting the information for 10 checks, the program will automatically load the input informaion onto the diskette.

5) If (2) is selected from the menu the computer automatically loads the previously stored checks from the diskette.

```
100 CALL CLEAR
110 PRINT "1 OUTPUT CHECKS"::
120 PRINT "2 INPUT CHECKS"
130 PRINT ::::::
140 INPUT "INPUT CHOICE?":Q
150 ON Q GOTO 160,280
160 FOR X=1 TO 10
170 CALL CLEAR
180 INPUT "CHECK #?":A(X)
190 INPUT "CHECK AMOUNT? ":B(X)
200 INPUT "PAYABLE TO? ":A$(X)
210 NEXT X
220 OPEN #1:"DSK1.CHECKS",INTERNAL,FIXED
230 FOR X=1 TO 10
240 PRINT #1:A(X),B(X),A$(X)
250 NEXT X
260 CLOSE #1
270 GOTO 100
280 CALL CLEAR
290 OPEN #2:"DSK1.CHECKS",INTERNAL,FIXED
300 FOR X=1 TO 10
310 INPUT #2:A(X),B(X),A$(X)
320 NEXT X
330 FOR X=1 TO 10
340 PRINT A(X)
350 PRINT B(X)
360 PRINT A$(X)
370 NEXT X
380 CLOSE #2
390 GOTO 100
```

CASSETTE FILES

- File Statements & Commands:



- Definitions (General)

Program Files- Uses OLD and SAVE commands. These commands are not used with line numbers; therefore, they are not actually part of the program. These commands are used only to store and retrieve programs.

Data Files- Uses OPEN, PRINT, INPUT and CLOSE statements. These statements are actually part of the program that is storing data.

File- A collection of records pertaining to a common group of data. (i.e. accounts receivable)

Records- A group of items that pertain to a subject of data group. (i.e. data of a specific account)

Item- A single piece of data within a record. (i.e. name of account, amount owed, etc.)

- Statement Definitions:

OPEN- This statement opens a communication line with the device specified. Parameters of data type are also provided via this statement. The OPEN statement does not move any data to or from the device; it only opens the door for communication.

PRINT- The PRINT statement is used to send data out to the device named in the OPEN statement

INPUT- The INPUT statement is used to load or retrieve data from the device named in the OPEN statement.

CLOSE- This statement closes the line of communication between the console and the device named in the OPEN statement.

- Data File Parameters:

File Number- Entered as a number sign (#) followed by a number between 1 and 255. A file number is not actually stored with the data file, so it is possible to store the data using a file number and retrieve the data with a different file number. The file number only tells the computer which channel is being used.

SEQUENTIAL- Must be used with cassette files. If it is not specified in the OPEN statement the computer assumes SEQUENTIAL access.

FIXED- Specifies the maximum length of each record. The different record lengths which may be specified are 64 (default value), 128 and 192. FIXED must be specified in the OPEN statement parameters. If no numeric value is given, then 64 is assumed. The number refers to the number of bytes per record. (NOTE: In a program, each PRINT or INPUT statement is the equivalent of a record if there are no trailing commas.)

INTERNAL- Refers to the format of data transfer, which is in binary. This is the most efficient method of sending data. When using INTERNAL data records, numeric variables equal 9 bytes each and string variables are 1 byte per character plus 1 length byte. (i.e. 1000= 9 bytes and "Texas" = 6 bytes) This will aid in determining record length with the FIXED parameter.

DISPLAY- Also refers to the format of data transfer. DISPLAY specifies an ASCII format. DISPLAY files can only utilize FIXED records of 64 and 128. If neither INTERNAL nor DISPLAY is specified in the OPEN statement, then DISPLAY is assumed.

OUTPUT- Allows for data to be sent from the console to the device named in the OPEN statement.

INPUT- Allows for data to be loaded into the console from the device named in the OPEN statement.

* NOTE: INPUT or OUTPUT and FIXED must be specified in the OPEN statement. If nothing else is specified the computer assumes SEQUENTIAL and DISPLAY.

- Program Example

1) Type in the attached program and store it on cassette using the SAVE CS1 command. Follow the cassette prompting to record the program.

2) Run the program and select the appropriate choice from the menu. (1) For creating a data file or (2) For loading a data file.

3) When creating a data file you are prompted to input the following:

Acct #
Acct Name
Acct Balance

The program will prompt you to type in five records of information as follows:

001
Texas Instruments
30000

002
General Electric
50,000

003
I.B.M.
75,000

004
Exxon
10,000

005
General Motors
75,000

4) After inputing the above information the program will automatically prompt to record the accounting data.

5) If (2) is selected from the menu the computer automatically starts the prompting for loading the previously saved data file.

*NOTE: When saving the data file, be sure that you do not save the data information over the program which was saved earlier. When saving the data, rewind the tape to a point located after the program file.

```
100 CALL CLEAR
110 PRINT "1 Output Data"::
120 PRINT "2 Input Data"
130 PRINT :::::
140 INPUT "Input Choice ?":Z
150 ON Z GOTO 160,280
160 FOR X=1 TO 5
170 CALL CLEAR
180 INPUT "Acct. # ":A$(X)
190 INPUT "Acct. Name ":B$(X)
200 INPUT "Acct. Balance ":A(X)
210 NEXT X
220 OPEN #1:"CS1",INTERNAL,FIXED,OUTPUT
230 FOR X=1 TO 5
240 PRINT #1:A$(X),B$(X),A(X)
250 NEXT X
260 CLOSE #1
270 GOTO 100
280 CALL CLEAR
290 OPEN #1:"CS1",FIXED,INTERNAL,INPUT
300 FOR X=1 TO 5
310 INPUT #1:A$(X),B$(X),A(X)
320 NEXT X
330 FOR X=1 TO 5
340 PRINT A$(X)
350 PRINT B$(X)
360 PRINT A(X):
370 NEXT X
375 CLOSE #1
380 INPUT "Press enter to continue.":C$
390 GOTO 100
```