CPB 00810

A standard multidimensional, easy-access data file structure for Apple II computers

Jakob H. Waterborg and Rodney E. Harrington

Department of Biochemistry, University of Nevada, Reno, NV, U.S.A.

A random access file structure was designed for Apple II microcomputers that allows data storage of more than 65 500 data values at 170 per Kbyte with a dynamic range of nearly 5 orders of magnitude. All or part of the data are easily accessible from BASIC under ProDOS operating conditions. The file structure accommodates single or multiple data sets in a single data file. Data values within a file with one set of data may interrelate by equal spacing along a second coordinate, such as time or space. Multiple data sets in a file can be independent, parallel or interdependent. Each interdependent data set defines the position of a data point along its coordinate in a two- or multidimensional registration system. The lowest and highest values of each data set are separately recorded to allow easy manipulation of even part of the data, e.g. for graphical presentation. The possibility of storing large numbers of data values in a single file facilitates high-resolution recording of events and simple mathematical manipulation.

Apple II   Microcomputer   BASIC   ProDOS   Random-access text file   Multidimensional data file

1. Introduction

The amount of memory accessible in a 64 or 128K Apple II computer limits severely its use for analyzing data sets with large numbers of data values. With a small program and its variables loaded and occupying 15K bytes of memory, a single data array with a maximum of only 4000 real values is possible. When such data must be displayed graphically on screen, a similar program can use only 2000 to 2500 data values. Commercially available programs show this limitation very clearly by their limit of 300 to 1000 values. This clearly indicates that data storage for large numbers of data values must be disk-based. Most disk-based systems for data storage employ sequential text files for their ease of data entry. However, this choice limits the accessibility for data retrieval severely and often requires entry of all data values in active computer memory before any data handling becomes possible.

Although multidimensional data arrays can be used inside the Apple computer to handle data points defined by multiple values in multiple dimensions, the one-dimensional structure of a sequential data file is not suited for storage of such data so that often interdependent data sets are stored in separate data files. This clearly further limits their accessibility for data analysis.

We propose a data file structure that allows a very large number of data values to be recorded in a single file and that allows easy retrieval of any number of individual values. It fully supports data storage and retrieval of multidimensional data types.

2. File structure requirements

A file structure for data storage in Apple II computers was developed with a number of specific criteria in mind.

Correspondence: J.H. Waterborg, Department of Biochemistry, University of Nevada, Reno, NV 89557, U.S.A.

(1) It should allow storage of large numbers of data, but take as small a file size as possible.
(2) It should allow a rapid access to a limited part of the data since often only part of the data need to be analyzed.
(3) It should be in a format readily accessible from a high-level programming language without the need for machine language routines for storage or retrieval to allow flexible and adjustable use of stored data by various program modules.
(4) The data file structure must support storage of one-, two- and multidimensional data.
(5) It should allow the dynamic recording of at least 4 orders of magnitude, including negative values, to accommodate storage of measurements ranging from −0.500 to +3.500, a range often obtained during spectroscopic measurements or during densitometric scanning of gels and films [1].

3. Choice of file type and programming language

Study of options in Apple operating systems and available file structures, and the need for flexible, large and fast storage options of hard disk and RAM disk, lead to the choice of Apple's ProDOS operating system which allows full access to a tree-structured directory and file system [2]. The wide availability of commercial and public domain machine language and ampersand-linked routines for use from BASIC of additional RAM capacity and the double-high resolution screen of 192 × 560 pixels for graphical presentation of data, supported this choice [3,4]. A random-access file structure was chosen to allow rapid access to any part of the data, in any order, and its standard format is shown in Fig. 1. The limit of a ProDOS random access file is 65 536 records, i.e. maximally approximately 65 500 data values can be stored in a single standard file.

```
Bytes     : 1 - 6   7-12  13-18 19-24 25-30 31-36 37-42 43-48 49-54 ...
Record    :   0      1     2     3     4     5     6     7     8    ...
Content   : RECORD$ V$(1) V$(2) V$(3) V$(4) V$(5) V$(6) V$(7) V$(8) ...
Example 1:  20100   20000 20010 20100 21000 30000 45000 88000 50000 ...
Example 2:  54000   19000 24000 19010 25000 19020 26000 19030 27000 ...


B: .. [ general formula : Bytes = Record# * 6 + 1 to Record# * 6 + 6 ].
R: ..   n-3     n-2     n-1     n    n+1   n+2   n+3   n+4   n+5   n+6   n+7   n+8   n+9  ...
C: .. V$(n-3) V$(n-2) V$(n-1) V$(n) TYPE  C$(1) C$(2) C$(3) C$(4) C$(5) C$(6) C$(7) C$(8) ...
E1:..  20000   19500   19000  18500 20001 20005 18500 88000
E2:..  20090   65000   20100  66000 20002 20003 19000 20100 19996 24000 66000
```

Fig. 1. Standard data file structure for one- and two-dimensional data. *Example 1* is a one-dimensional data file with 100 data points (NUMBER = 100, TYPE = 1, RECORD = 100), a decimal displacement value of +5 (C(1)), a minimum data value of −1500 (C(2)) and a maximum one of 68,000 (C(3)), and thus real data limits of −0.015 and +0.680. *Example 2* is a two-dimensional data file with 17000 data values in each data set and thus 34000 records with data values in the file (NUMBER = 17000, TYPE = 2, RECORD = 34000). Data set 1 has a decimal displacement value of +3 (C(1)), a minimum data value of −1000 (C(2)) and a maximum one of 100 (C(3)), and thus real data limits of −1.0 and +0.1. Data set 2 has a decimal displacement value of −4 (C(4)), a minimum data value of 4000 (C(5)) and a maximum one of 46000 (C(6)), and thus real data limits of 40000000 and 460000000.

4. Choice of a standard data file structure

An essential feature of the data file is that all values are stored as strings which are at least one byte shorter than the record length. The last byte in each record remains empty, or contains a carriage return (Control-M ASCII character) in case of strings of maximum size, to signify the end of a recorded value. To minimize record size and maximize data storage capacity, we chose to limit our strings to represent only positive integer values.

A standard offset is used to easily accommodate negative data. Storage of high-precision optical density data requires approximately 4 to 5 orders of magnitude. We chose to represent values from −2.0000 to +7.9999 OD as integers from 0 to 99999 by applying a shift of 4 positions for the decimal position in the real value, applying integer rounding and a subsequent offset of +20 000 (Fig. 2 *). Thus 0.101 OD, multiplied by 10^4 becomes 1010, rounding gives again 1010 (i.e. *no* loss of accuracy), and addition gives 21010. The string format of this number is then stored. Similarly

* The algorithms for data storage (Fig. 2) and data retrieval (Fig. 3) from a standard data file are valid for files with any number of data values (NUMBER) and any number of data sets (TYPE). The examples specify and dimension for a data file with two sets of data values (TYPE = 2) with 1000 data values per set (NUMBER = 1000). The choice for decimal shift values was made prior to data collection and storage (C(1) and C(4)). The limits of data values (C(2)–C(3) and C(5)–C(6)) were determined during data storage (Fig. 2, 60–80).

```
10  DIM VALUE(1000, 2), C(6)
  : NUMBER = 1000
  : TYPE = 2
  : RECORD = NUMBER * TYPE
  : C(1) = 4
  : C(2) = 99999
  : C(3) = 99999
  : C(4) = - 2
  : C(5) = 99999
  : C(6) = 99999
  : REM DIMENSION DATA ARRAYS;
    SET NUMBER OF DATA VALUES;
    SET NUMBER OF DATA SETS OR DATA TYPE CODE;
    SET DECIMAL DISPLACEMENT VALUES;
    INITIALIZE LOWEST/HIGHEST DATA VALUES AT 99999
20  PRINT CHR$(4) "OPEN" FILENAME ",L6"
  : REM CREATE AND OPEN A RANDOM ACCESS DATA FILE WITH A
    RECORD LENGTH OF 6 BYTES
30      FOR I = 1 TO NUMBER STEP TYPE
40          FOR J = 1 TO TYPE
50          VALUE = INT ( VALUE(I,J) * 10 ^ C(3 * J - 2))
          : REM SHIFT DECIMAL PLACES AND ROUND
60          IF C(3 * J - 1) = 99999 THEN C(3 * J - 1) = VALUE
          : C(3 * J) = VALUE
          : REM SET VALUE LIMITS AT FIRST PASS
70          IF VALUE < C(3 * J - 1) THEN C(3 * J - 1) = VALUE
          : REM SET MIN VALUE
80          IF VALUE > C(3 * J) THEN C(3 * J) = VALUE
          : REM SET MAX VALUE
90          VALUE$ = STR$ ( VALUE + 20000)
100         PRINT CHR$(4) "WRITE" FILENAME ",R" (I - 1) * TYPE + J
110         PRINT VALUE$
          : REM ENTER STRING IN SPECIFIED RECORD NUMBER
120         NEXT J
130     NEXT I
140 PRINT CHR$(4) "WRITE" FILENAME ",RO"
150 PRINT STR$ ( RECORD + 20000)
  : REM ENTER TOTAL NUMBER OF RECORDS WITH DATA VALUES
160 PRINT CHR$(4) "WRITE" FILENAME ",R" RECORD + 1
170 PRINT STR$ ( TYPE + 20000)
  : REM ENTER NUMBER OF DATA SETS OR DATA CODE
180     FOR I = 1 TO 3 * TYPE
190     C$(I) = STR$ ( C(I) + 20000)
200     PRINT CHR$(4) "WRITE" FILENAME ",R" RECORD + 1 + I
210     PRINT C$(I)
      : REM ENTER CONTROL VALUES FOR DECIMAL DISPLACEMENT AND DATA
        LIMITS
220     NEXT I
230 PRINT CHR$(4) "CLOSE"
  : REM CLOSE THE DATA FILE
```

Fig. 2. Example of data storage.

−0.402 and +3.303 OD values are stored as 15980 and 53030. This choice easily covers the normal range from −0.500 to +3.500 OD with a extra decimal position to spare for added accuracy, not present under normal conditions of spectroscopy but usable when data averaging may yield higher accuracy. Thus data accuracy is fully maintained while a defined, invariable data length is specified that allows easy retrieval of data (Fig. 3).

In most biological and biochemical systems, the dynamic range of nearly 5 orders of magnitude was generally found to be more than sufficient for data storage from sources other than spectroscopy without any loss of accuracy. The step that shifts decimal positions before integer rounding allows the use of Standard Units to be carried through calculations because positive and negative real

```
10  DIM VALUE (1000,2), C(7)
  : REM DIMENSION DATA ARRAYS
20  PRINT CHR$(4) "OPEN" FILENAME
  : REM OPEN THE EXISTING RANDOM ACCESS DATA FILE
30  PRINT CHR$(4) "READ" FILENAME ",RO"
40  INPUT RECORD$
50  RECORD = VAL ( RECORD$) - 20000
  : REM GET TOTAL NUMBER OF RECORDS WITH DATA VALUES
60  PRINT CHR$(4) "READ" FILENAME ",R" RECORD + 1
70  INPUT TYPE$
80  TYPE = VAL ( TYPE$) - 20000
  : REM GET NUMBER OF DATA SETS OR CODE FOR DATA TYPE
90      FOR I = 1 TO 3* TYPE
100     PRINT CHR$(4) "READ" FILENAME ",R" RECORD + 1 + I
110     INPUT C$(I)
120     C(I) = VAL ( C$(I)) - 20000
      : REM GET CONTROL VALUES FOR DECIMAL DISPLACEMENT AND
        DATA LIMITS
130     NEXT I
140 NUMBER = RECORD / TYPE
  : REM CALCULATE NUMBER OF DATA VALUES PER DATA SET
150     FOR I = 1 TO NUMBER STEP TYPE
160         FOR J = 1 TO TYPE
170         PRINT CHR$(4) "READ" FILENAME ",R" (I - 1) * TYPE + J
180         INPUT VALUE$
190         VALUE(I,J) = ( VAL ( VALUE$ ) - 20000) * 10 ^ - C(3 * J - 2)
          : REM CONVERT ROUNDED INTEGER INTO REAL DATA VALUE
200         NEXT J
210     NEXT I
220 PRINT CHR$(4) "CLOSE"
  : REM CLOSE THE DATA FILE
```

Fig. 3. Example of data retrieval.

numbers ranging from 10^{-33} to 10^{+33}, almost the maximum range that the Apple processor can handle, can easily be accommodated.

5. The use of control values

The control values of a data file stored in the first record (record 0) and in a limited number of records immediately after the last data value, allow for increased flexibility in handling all or part of the data. Record 0 stores the number of records filled by data (RECORD), after the standard offset of 20 000 (Fig. 2). This value is derived from two other values, the number of data values per data set (NUMBER) and the number of data sets in the data file (TYPE). RECORD is limited by the ProDOS file limit of 65 535 so that the stored string generally will vary between 20 001 and 85 550. The use of the first record for this value allows easy access to the size of a data file, and also to the type of file and its limits because these values are stored in the records immediately following those with the last data value (Fig. 1).

The first of these records will contain a code number that specifies the type of data stored (TYPE), and is generally used to note the number of data sets or dimensions. Thus one-dimensional scanning data contain a value string TYPE$ of

20001, signifying 1 data set with an implicit even spacing between data points (Fig. 1). In such a file the first data value is located in record 1, the second in 2, and the *n*th data in record *n*. Two-dimensional data store the values of data set 1 in records 1, 3, 5, ..., $2 \times n - 1$, and the values of data set 2 in records 2, 4, 6, ..., $2 \times n$ (Fig. 1). Multiple data sets follow a similar rule.

Record number RECORD + 2 stores in string C$(1) the number of powers of 10 to be applied to store real values of the first data set as rounded integers (Fig. 2). For optical density scanning values it contains string 20004 so that $0.103\ \mathrm{OD} \times 10^4$ will give the value 1030, to be rounded to the nearest integer, if required. The maximum range of C$(1) is 19967 to 20033 because the Apple processor is limited to real numbers between $\pm 1 \times 10^{38}$ and $\pm 3 \times 10^{-39}$. Strings similar to C$(1) are stored in record RECORD + 5 for the second data set, in record RECORD + 8 for the third one, and so on.

Record RECORD + 3 stores the string with the lowest value of the first data set and record RECORD + 4 stores the string with the highest value. These lower and upper limits of the data set are recorded during initial data storage (Fig. 2) and allow easy graphical manipulation of the data because it obviates the need to read all data file values before even a small part of the data can be graphed. Thus standardized graphical handling without loss of information outside the screen limits is possible. The same upper and lower value limits of additional data sets are stored at RECORD + 6 and RECORD + 7, in RECORD + 9 and RECORD + 10, etc.

6. One-dimensional data files

This type of one-dimensional data file is used for situations where the value of a single variable is collected at a equidistant interval such as time or distance. Such data can be recordings over time or temperature or of optical density measured by a densitometer while a polyacrylamide gel, stained for protein by Coomassie, is passed at a constant rate between lamp and photomultiplier tube [1]. In such a system the equal spacing in time of recording optical density is transformed to a equal spacing in space in the gel along the line of scanning. Thus by only recording the variable dimension, a greater information density in the data file is possible, reducing the need for memory expansion or storage capacity.

7. Two- and multidimensional data files

A significantly different type of data values is obtained when the change in a first variable results in a change of a second variable. A data point is then defined by the coordinates of the first and second variable. In this case two independent sets of data values will define an array of data points defined by 2 coordinates (2 dimensions). To allow easy access to the two values that define a data point, we chose to place both values next to each other (Fig. 1). This greatly facilitates retrieval of only part of the data points. One can use almost any type of analysis as an example for this type of data file, e.g. dose–response results in pharmacological experiments or changes in optical density in a DNA preparation which is melted by a slow and gradual increase in temperature.

This choice of arranging corresponding values from more than one data set together requires a decision on the number of data sets that will be collected, recorded in the control value TYPE, prior to file storage of any data. Such a choice is generally known already before data collection is started. What is often not known, or not precisely known, is the number of data values that will be or need to be collected in a data set. The choice to keep corresponding data values together has the clear advantage that it leaves open the choice of the number of data values to be stored until one chooses to terminate data collection. Of course, the limits of 65 536 records per file and possibly the available disk storage capacity always apply.

The file structure will also allow data storage of several independent data sets, for instance in a situation where both high and low sensitivity detectors are used to record the same variable. In such a case, the independent but time-correlated data values are next to each other in a data file and can more easily be studied together. In other experimental systems two independent variables are changed and are recorded together with the resulting variable. Such a combination of three interdependent data sets defines a data point in a three coordinate system. Such multidimensional, interdependent data sets are also easily accommodated within the data file structure, and access to any part of the data is maintained. The one choice that must be made is the number entered for the control value TYPE. In the examples presented, we used the TYPE value purely to indicate the number of data sets in a data file. It can also be a coded value, for instance TYPE = 55 can signify a dose–response two-dimensional interdependent data set with recording of data values once every minute, so that in fact a third dimension, time, is recorded without any requirement for data storage capacity.

8. Available software

The data file structure proposed has been implemented in a set of integrated program modules that were developed primarily to record and analyze optical density patterns of stained polyacrylamide gels or of their fluorographic images on X-ray film [1]. The number of data values that can be stored in a data file as proposed is so large that band resolution and quantitation of 50 μm or less is attainable for gels as long as 15 or 30 cm. If the noise level in the recorded data is relatively high, simple data averaging over a very large number of initially recorded values can increase data accuracy while maintaining spatial resolution between closely spaced protein bands. The large number of data points that generally span a single stained protein band allow for a simple but still accurate determination of peak size by simple data value addition, so that mathematical approximations of expected band or peak shape can be avoided. In case of very high resolution band analysis, e.g. by application of overlapping Gaussian curves to a patterns of partially overlapping protein bands [1], a sufficient number of data values are generally present within a data file, so that one can decide whether such an analysis might be required at a much later stage than otherwise would be possible.

The integrated set of programs that employ the described data file structure are collectively named SCAN.GRAPH. They can be obtained from the authors for a nominal cost and registration fee. Several program modules employ fast analog–digital convertors for multichannel data collection, while others allow manual data entry or editing. Utility programs are included that will convert non-standard data in a sequential text file to the standard file format, or do the reverse. Double-high resolution data graphs at 180 × 540-dot resolution allow visual inspection of collected data and graphical editing by background subtraction, selection of part of the data and mathematical data filtering. Quantitation of peak areas at single-value screen resolution is supported. All programs are written in BASIC and can easily be adapted to specific hardware conditions. They will function on Apple IIe and //c machines with 128K RAM and can be run from 5.25-inch floppy disks, 3.5-inch micro disks and from any hard disk under ProDOS.

9. Additional options and alternatives

In specific cases which require a greater dynamic range than 100 000, a longer storage string could be used with a record length (,L# in line number 20 of Fig. 2) one byte longer than the maximum length of the storage string. If this increases significantly the length of each record, it may be easier to store strings of real values instead of rounded ones, because gain in reduced storage requirements may not be offset by the number of additional mathematical manipulations required to store real values as rounded integers (Fig. 2). For storage of real values, e.g. $-1.2345678E+22$, a record length of 15 must be specified to assure that the last byte of a record remains empty or contains a Carriage Return to signify the boundary with the next record. Compared to the standard file, such a file would have a 2.5 times larger storage requirement. Data storage and retrieval would be slightly faster, and control records with the position of the decimal point shift like C(1) would not be required.

If multidimensional data arrays, each with large numbers of data values, are to be stored, the ProDOS limit of 65 536 records, and thus data values, may be too limiting. An option, still essentially retaining the proposed standard file structure, would be to increase the number of fields per record from 1 in the standard file structure, to *n* with *n* being the number of dimensions. Input and output of data for each record would be more complicated and slower but still possible. In addition to using ,R♯ to specify the record to write to or read from, one must also use ,F♯ or ,B♯ to specify the field or byte number within the record to specify the start of writing or reading. In fact, each record is converted to a sequential type of mini–file with all the inherent problems of data retrieval. Such a change would limit the number of data values for *each* dimension to approximately 65 500. The number of dimensions would be unlimited as long as the total file size does not exceed 16M bytes.

Acknowledgements

This work was supported in part by United States Public Health Service Grant GM 33435 and by Hatch Project 131 through the College of Agriculture, University of Nevada Reno.

References

[1] J.H. Waterborg and H.R. Matthews, Patterns of histone acetylation in *Physarum polycephalum*. H2A and H2B acetylation is functionally distinct from H3 and H4 acetylation, Eur. J. Biochem., 142 (1984) 329–335.

[2] BASIC Programming with ProDOS, Product number A2L2013 (Apple Computer Inc., 20525 Mariani Avenue, Cupertino, CA 95014, U.S.A.)

[3] Ramdrive^{TM}, by R. Kraemer (Applied Engineering, P.O. Box 798, Carrollton, TX 75006, U.S.A.)

[4] Beagle Graphics^{TM}, by M. Simonsen (Beagle Bros Micro Software, Inc., 3990 Old Town Avenue, San Diego, CA 92110, U.S.A.)