dBASE III Plus and the MARC AMC Format: Problems and Possibilities

RONALD J. ZBORAY

Abstract: As archives with smaller holdings adapt to the MARC AMC data format, they will need to explore the utility of various commercially-available, microprocessor-based, data base management systems. This article presents four approaches to using the AMC format in dBASE III Plus, the best selling of these systems. The discussion of the limitations and possibilities of the four approaches demonstrates that while there is no inherent incompatibility between dBASE and the MARC format, the most efficient use of a dBASE-MARC system requires considerable programming.

About the author: Since 1984 Ronald J. Zboray has been the Microfilm Editor of the Emma Goldman Papers at the Institute for the Study of Social Change, University of California at Berkeley. He received his B.A. in history from the University of Bridgeport and the A.M. and Ph.D. in American Civilization from New York University. He has taught American history at the University of Connecticut at Stamford, Pace University, the University of Hartford, and Post College (Waterbury, Conn.). His articles have appeared in ***The International Journal of Micrographics and Video Technology, Film and History, American Quarterly,*** *and* ***Southwest Review.*** *For the latter he won the 1986 DeGolyer prize for best nonfiction essay in American Studies.*

WITH THE INCREASING ACCEPTANCE of the MARC AMC data format by the community of archivists, archives with smaller collections which are able to afford only microcomputers will seek to use the new format for automated control of their holdings. Vendors, such as Cactus Software, have already recognized this small emerging market with records management packages that have the MARC AMC format already built in. Researchers at Michigan State University, with funding from the National Historical Publications and Records Commission, have developed yet another such package. Yet, by and large, archivists have been unhappy with the overall inflexibility and occasional unreliability of these specialized offerings. Some have limited or no networking capability, and none contain a flexible internal programming language as extensive as those in best-selling data base management systems. This dissatisfaction is not surprising; few small software publishers can afford the costs and labor time necessary to develop state-of-the-art data base management systems. Certainly the limited market of archivists who control their collections with microcomputers does not warrant larger publishers undertaking this development.[1]

For the foreseeable future, archives with small holdings will probably rely upon mass market, commercial data base management systems for document control. Many archivists, before they knew much about MARC AMC, used such systems as Revelation, R:Base 5000, and, principally, the industry leader dBASE. Because of the large, highly competitive market these products address, they are generally speedy and dependable, and offer the flexibility of a wide variety of customized reporting options and data manipulations and searches—particularly necessary to the frequently unique characteristics of archival collections.[2]

These limited local applications of data base management systems scarcely required the descriptive depth and standardization demanded by MARC AMC. Few archivists on their own would create a data base file with MARC AMC's seventy-seven variable data fields with each having from one to twenty subfields. Through foresight or sheer luck, a handful of archivists can output data they collected in earlier systems in a form compatible with MARC AMC; far too many others cannot. To ensure compatibility and, above all, to use the new data format most effectively, archivists using microcomputer-based data base management systems should plan for AMC at the input stage when files are created.[3]

Some considerations for archivists planning to use AMC within a commercial data base management system will be presented below. Though the focus is on

[1]MARC (Peoria, Ariz.: Cactus Software, Inc.); MicroMARC:amc (East Lansing, Mich.: Michigan State University Archives and Historical Collections). See also Micro Archives and Research Collections On-Line (MARCON) (Baltimore: Automated and Information Retrieval Systems); Mark H. Kibbey, "Marcon II: Information Control on a Microcomputer," *Library Hi-Tech News* 14 (Summer 1986): 11–21.

[2]Revelation (Seattle: Cosmos, Inc.); R:Base Series 5000 (Bellevue, Wash.: Microrim, Inc.); dBase III Plus (Torrance, Cal.: Ashton-Tate). A good starting point for archivists thinking about implementing data base management systems is Jeanne C. Raudenbusch and Robert B. Shaklee, "Selecting Records Management Software," *Records Management Quarterly* 19 (July 1985): 22–31.

[3]The MARC AMC format appears in Library of Congress, *MARC Formats for Bibliographical Data*, updates nos. 10 and 11 (Washington, D.C.: Library of Congress, 1984, 1985). A more usable guide is Nancy Sahli, *MARC for Archives and Manuscripts: The AMC Format* (Chicago: Society of American Archivists, 1985). Helpful for implementing MARC is Max J. Evans and Lisa B. Weber, *MARC for Archives and Manuscripts: A Compendium of Practice* (Madison: State Historical Society of Wisconsin, 1985). The literature on MARC AMC grows annually; solid introductions to key issues surrounding MARC AMC are presented in three articles in the Winter 1986 edition of the *American Archivist* (vol. 49: 9–40): Nancy A. Sahli, "Interpretation and Application of the AMC Format"; Katherine D. Morton, "The MARC Formats: An Overview"; and Steven L. Hensen, "The Use of Standards in the Application of the AMC Format."

dBASE III Plus, most of the viewpoints can be applied to other systems, since most of them have similar features and, for archivists, similar limitations. dBASE was chosen because it is the best-selling commercial data base management system for microcomputers. In the discussion below, examples will be drawn from the experiences of the Emma Goldman Papers Project at the University of California at Berkeley as it faces the challenge of making dBASE compatible with MARC AMC.[4] After a brief overview of dBASE's capabilities, four different approaches to using dBASE for AMC will be explored. Each approach requires an increasing level of programming sophistication yet yields a more efficient use of hard disk storage space.

The programming required to use dBASE for AMC most efficiently may well be used as an argument against using dBASE at all for this purpose. As the discussion below will demonstrate, dBASE can be adapted to the needs of the AMC format only with difficulty or within severe limitations of disk space usage.

Nevertheless, the installed base of dBASE users among archives will inevitably continue to grow, and many archivists may have little alternative to dBASE for collection control. Though dBASE has a steep learning curve, the program encourages experienced users to dabble in the internal programming language where the true flexibility of the system emerges; the expertise for more sophisticated adaptations of dBASE to the AMC format may thus eventually be found among the staff of an archive. Archivists may also use dBASE for a host of different applications, from keeping addresses of correspondents to keeping track of volunteers or donors. Staff members must master only one system for these different purposes. Because dBASE, despite its limitations, will continue to be used by archivists, the suitability of the four approaches will be discussed below.

Capacity and Capabilities

At first glance dBASE III Plus seems to be adequate to the needs of even moderately large archives. Up to a billion records and a total of 2 billion bytes can be controlled. A maximum of 128 fields promises to fit easily MARC AMC's requirement for 77 fields. Although the maximum record size of 4,000 bytes may barely contain the information required by the MARC format, dBASE permits variable-length memo fields that can run to 512 kilobytes.[5]

In practice, for use on microcomputers, dBASE's specifications make it suited only in archives with small to moderate-sized holdings. dBASE demands that a single, entire data base file

[4]The Emma Goldman Papers, in existence since 1980 under the directorship of Dr. Candace Falk, is an NEH- and NHPRC-sponsored project collecting Goldman's published and unpublished works and correspondence and government documents pertaining to her. Supported by a guide and in-depth, three-volume index, a microform edition of these materials will be published by Chadwyck-Healey, Inc.; completion is expected in 1991.

[5]dBase III Plus differs from dBase II and III in several ways. The Plus version, released in 1986, offers multi-user operation, the ability to control vastly more files, optional menu-driven operation and dozens of other new features. A clumsy copy protection in the first release has recently been removed. Most library and archival literature has not yet had time to catch up with the new upgrade. Since there are strong similarities between the various versions, some of the older literature may still be relevant. See Roger D. Palmer, *dBase II & dBase III: An Introduction for Information Services* (Studio City, Cal.: Pacific Information, 1984); and the various regular columns in *Library Software Review* beginning with Randolph Gadikian, "dBase II in Libraries" 3 (1984): 521–26. Although more and more general sources have appeared for dBase III Plus, one of the best is Luis Castro, Jay Hanson, and Tom Rettig, *Advanced Programmer's Guide Featuring dBASE II, dBASE III, and dBASE III Plus* (Culver City, Cal.: Ashton-Tate, 1985). For good introductions to dBASE III Plus, see Robert A. Byers, *Everyman's Data Base Primer Featuring dBASE III Plus* (Torrance, Cal.: Ashton-Tate, 1986) and Howard Dickler, *dBASE II Plus Trail Guide* (Torrance, Cal.: Ashton-Tate, 1986).

be kept on one disk or diskette. Since Microsoft DOS has difficulty addressing a file greater than 32 megabytes on a single storage media, the total record capacity of dBASE falls far, far short of the theoretical 1 billion limit. That each variable memo field uses a minimum of 512 bytes when data is entered even more severely restricts the total number of records that can be controlled.[6]

The most obvious limitation of dBASE (and nearly all other data base management systems except Revelation) is the fixed-length field. Because the variable-length memo fields cannot be searched or manipulated by dBASE in any way, the fixed-length fields remain the heart of the system. Fixed length fields waste immense amounts of disk space (about one-half to two-thirds of capacity), further limiting the maximum number of records that can be stored on microcomputer with a fixed disk. The business applications for which dBASE is designed will often set a reasonable length on, for example, a personal name field, and truncate names that go beyond that length. Standard archival practice allows for no such shortening; the longest field entry in a file will determine the length of the field, even if that entry is singularly long. Thus for archives, the potential for wasting disk space with the fixed-length field far exceeds that of other applications.

The MARC AMC format itself encourages a waste of disk storage space. Relatively few data control records will use all seventy-seven fields and perhaps only some will use more than a majority. Certainly if an archive settles the first or even second level of AACR2 description, even fewer fields will be used.[7] But such avoidance of the descriptive depth offered by level three in conjunction with MARC AMC may be shortsighted, since points of access considered excessive today too often become expected in the future.

In the discussion below the fullest implementation of MARC AMC will be considered. Control will be assumed to be at the document level and the full AMC format will be used even though most archives would never have need to implement the format to such an extent.

Approach I: Using dBASE Directly Off the Shelf

The ability of very novice users of dBASE to construct and manipulate data files with relative ease accounts for much of dBASE's success. Employing either the "CREATE «filename»" command at the dot prompt or the menu-driven, pull-down "Assistant," the user who wants to design a data base needs to do little more than name fields, determine whether the type of data in each field should be character, numerical, logical, date, or memo (text file), and decide how many characters each fixed-length field will contain.

Table 1 shows the structure of a viable, easily-constructed dBASE file that conforms to the requirements of MARC AMC. The estimated fixed-field lengths range from 12 characters for Tag 007 (category of materials) to 250 characters for Tags 600 and 650 (subject added personal names and topics, respectively). Since all the fields, except one, use character-type data this information has not been displayed. Because dBASE permits only field names of ten characters or less, these names have been mnemonically derived in the table (e.g., "LCNUMBER" in Tag 010 stands for Library of Congress control number).

Six variable length fields do not appear. Tag 002, the subrecord map of the directory, can be generated by the pro-

[6]Editing memo fields adds to their length; the old version is saved in the data base text file and even subsequent edits are saved. dBASE, however, will release the space used by these older versions if the file is copied.
[7]*Anglo-American Cataloging Rules*, 2d. ed. (Chicago: American Library Association, 1978), 15.

gram that outputs the information in the AMC format.[8] The field for character sets specified (Tag 006), not a required field in AMC, has not been used because dBASE stores data in ASCII format. The remaining four fields representing Tags 870 to 873 have not been included because these have fallen into widespread disuse since the Library of Congress's adoption of AACR2 in 1981.

Table 1 shows only forty-eight fields rather than the seventy-seven prescribed by MARC AMC. The difference arises out of combining Tags 500 through 59X—the notes section of the AMC format—into one variable-length field. These basically textual note fields use enormous amounts of space; without combining them, the total number of bytes used by the fixed length exceeds the 4,000 byte per record limit of dBASE by nearly 75 percent. The combined notes field sacrifices little flexibility since the corresponding AMC fields seldom require computer manipulation aside from free text searching. Although dBASE III Plus does not provide free text searching of memo fields, at least one very inexpensive utility program does.[9] One, rather than several, memo fields has been used to save disk space; as mentioned earlier, every time data is entered into a memo a minimum of 512 bytes are used. Users access the memo field by calling up the record to be edited, placing the dBASE blinking cursor on the field filled with the word "memo," and typing control page down ("Ctrl-PgDn"). A text file (in either the Wordstar-based dBASE editing mode or a word processing program of the user's choosing) appears on the screen, and data may be entered in text form with the Tag number identifying each field. Exiting the editing mode or word processor returns the user to the memo field in the controlling data base record.

Even when using memo fields for notes, this off-the-shelf approach wastes space promiscuously. Most fields will be used rarely, yet the possibility of their use requires that they be included. For example, the three fields for conference and meeting information (Tags 111, 611, and 711), which are not always relevant to manuscripts and archives, together use 310 bytes. Shortening these fields might make it impossible to enter the full name of the rare conferences or meetings—which often have very long names—that occur in a collection. The space wasted by such uncommonly-used fields in addition to that lost due to the fixed-length field basis of the system may amount to as much as 95 percent of disk capacity.

The maximum number of records that can be stored on a hard disk using this approach can be estimated easily. The fixed-length fields use up 3,814 bytes per record and the variable-length notes field a minimum of 512 bytes. Since the notes field embraces so many fields archivists commonly use, 1,024 bytes per record is probably a more realistic estimate than the minimum 512. Thus, total number of bytes used by each record equals 4,838 (the use of the memo field permits an excess of the 4,000-byte dBASE maximum). That number divided into the thirty megabytes of the largest single storage media DOS can address suggests an absolute maximum of 6,200 records on a hard disk with that capacity. But the performance of a hard disk slows dramatically as it approaches full disk capacity. For best performance, a 30 megabyte hard disk should hold no more than 5,000 records. For many archives, especially ones con-

[8]Tags Leader/00 to Leader/23 and Directory/00 to Directory/11 also do not appear because they are best generated at the program level during data output.

[9]dBASE III Memo Searcher (San Diego: Data Based Solutions).

trolling materials only at the collection level, this maximum number of records may be ideal.

Such a scheme, however, provides only for the storing of records; to get the most out of dBASE, users must build index files based on the data base. A single index file (e.g., main entry–personal name) uses only about 10 or 20 percent of the size of the related data base, but several such files must be created. Index files are especially necessary for making a quick search for information in the main data base. Seeking indexed information (the first character and up to a hundred characters to the left) takes less than two seconds for a data base of any size; searching for character strings within fields of a moderately-sized unindexed data base can take several minutes. Fortunately, indexes can reside on a separate drive or partition, so one need only add on an extra 20 or 30 megabyte hard disk to make use of this important feature.

Using dBASE directly off the shelf has advantages and disadvantages, the balance of which changes depending upon the nature of the archive and the resources available. Within only an hour or so, users can create a workable MARC AMC-based system and begin to enter data, with little or no support from programmers or consultants. With programmers charging about $50 an hour, such user independence may be well worth the tens of millions of wasted bytes of storage the off-the-shelf approach requires. Storage media increasingly becomes less expensive so that one can anticipate paying for every ten megabytes of storage capacity about $250 or less—a one-time, certain cost that may compare favorably with the unpredictable and chronic costs of customized programming. The chief disadvantage of the off-the-shelf approach is the limited number of records that can be controlled. Even if the user tinkers with field lengths to make the data records most suited to the collection and thus cuts down the number of bytes used by each record, such gains would be minimal. There would still be a limit of about 5,000 records, with no room for further growth, unless the user turns to one of the other approaches explained below.

Approach II: Simple Dependency

The 5,000 record limit of the off-the-shelf approach can be most easily overcome by splitting the single data base outlined in Table 1 into two dependent data bases. The two data bases must be linked through a common field, indexed in at least one. The common field may be the record number or any *unique* finding number in any of the MARC format fields. Whatever the source of the linkage, the numbers or characters must appear in only one record in the data base or the linkage will break down.

Creating dependent data bases requires delving deeply into the dBASE III Plus manual. Although the menu-driven "Assistant" promises to guide the user through the steps to link data bases, there is difficulty writing programs when the data bases are on two different disks.

The first step in creating simple data base dependency consists of constructing the two data bases on two different hard disks. In the data base structure presented in Table 1, a natural division occurs between field number 27 (notes) and field 28 (subject-added personal names). Fields 1 to 27 total an estimated length of 2,478 bytes leaving 2,360 bytes for the remaining fields. If the record number will not provide the linkage between the two data bases, the slightly smaller second data base will need an extra field to duplicate the linking expression in the first. For example, if field 6, "SYSNUMBER" (the local system control number used by many archives to store the accession number) is the link, the same field must appear in the second data base.

With the two data bases constructed on different disks the second step of creating dependency requires creating a format file. The format file will link fields from the two data bases on the CRT screen so that the existence of two data bases will become undetectable during data entry. The dBASE text editor can be used to create the format file (while dBASE runs) with the "MODIFY COMMAND «filename.fmt»" command, or a word processing program may be used as long it will not enter any invisible page formatting characters in the text.

Next the user must tell the program where on the screen to put each field name and the fixed number of spaces for each field. dBASE addresses the screen using "@" with row and column coordinates (e.g., "@ 12,40" represents "at row 12, column 40"). The field names are typed on the screen by using the "SAY" command and the blanks are painted by the "GET" command. When two data bases are being used simultaneously, the "GET" command demands that a pointer ("->") be used to identify the source of the field being read. If the first data base is named MARCAMC1 and the second MARCAMC2, then any number written in the blanks painted by the command "@ 7,12 GET MARCAMC1 -> SYSNUMBER" will be entered only in the first data base.

Figure 1 shows the text of a program that will create a screen form for entering into two data bases the fields presented in Table 1. Although the two data bases may be called MARCAMC1 and MARCAMC2, dBASE permits using a shorter name or "alias" for programming purposes. In Table 2 "ONE" represents the alias for the first data base and "TWO" for the second, thus "ONE -> SYSNUMBER" is used instead of "MARCAMC1 ->SYSNUMBER." Finally, the "SAY" and "GET" command lines are interrupted every so often by a "READ" command that creates page breaks in the data entry form; the user will be able to access the entire form at any time during entry using the "PgUp" and "PgDn" keys.

The final step in creating a simple dependent data base relationship is to write a short program to create the dependency whenever the program is loaded. Again the dBASE text editor can be used to open the program file (do not use any three-character file extension, i.e., filename.*ext*). Commands are entered line by line in dBASE's easy-to-use programming language. For example, assume that there are: 1) two hard disks (C and D); 2) two data base files named MARCAMC1 and MARCAMC2 on C and D respectively; and 3) a format file called MARCAMC.FMT. In this instance, the short program would read:

```
SELECT 1
USE C:MARCAMC1 ALIAS ONE
SELECT 2
USE D:MARCAMC2 ALIAS TWO
SELECT 1
SET RELATION TO RECNO( ) INTO TWO
SET FORMAT TO MARCAMC
RETURN.
```

The program selects two work areas to be open simultaneously: area 1 with MARCAMC1 (under the alias name "ONE") on disk C and area 2 that has MARCAMC2 ("TWO") on D. The "SET RELATION TO . . . " command creates the data base dependency so that when a record pointer moves in one, it moves in the other as well. The example uses the record number ("RECNO( )") as the point of interaction between the two data bases. Finally, the "SET FORMAT TO . . . " command will call up the format file when either the standard "EDIT" or "APPEND" commands are entered at the dot prompt (the prompt reappears after the program issues the "RETURN" command). This program must be run every time the user loads dBASE; to load it at the dot prompt, the user types "DO «program filename»."

Although creating this simple dependency may seem daunting because of the amount of programming involved, most of the programming (i.e., the format file) is redundant and quite simple, even for novices. Much of this programming can be avoided if the user is willing to put up with a more obvious split between the two data bases during data entry. If so, the user would use the same short loading program given above without the "SET FORMAT TO . . . " command. The record in the first data base would be edited and written to the disk and the dot prompt will appear. The user then would change the currently selected work area by typing "SELECT 2." By entering "EDIT," the continuation of the record on the second data base appears. At the dot prompt after completion, "SELECT 1" must be entered again before beginning to edit another record.

Simple data base dependency, however achieved, has one distinct advantage over the off-the-shelf approach. Since dependency allows the forty-eight fields presented in Table 1 to be distributed between two hard disks, the maximum number of records that can be controlled doubles (if the two disks have the same capacity). Thus using the byte-wasting data base structure presented above, a user who creates a simple dependency between two data bases on two 30 megabyte disks or disk partitions will be able to control about 10,000 records (with the various required indexes stored on yet another hard disk).

The simple data base dependency approach to creating MARC AMC records in dBASE probably is best suited to archives with smaller holdings which intend to control materials at the collection level. The archive should expect no dramatic growth since the 10,000 record limit is as fixed as the 5,000 limit of the off-the-shelf approach. Apart from hard disk space, the dependency approach is relatively inexpensive. The limited programming involved can be accomplished by anyone with a working knowledge of dBASE. Even if outside programmers must be hired to set up the dependency, the simplicity of the programming would keep costs down and limit them to the period in which the system is set up.

Approach III: Multiple Dependency

The two approaches outlined above waste large amounts of disk space in order to set up, with minimal programming, small document control systems. More sophisticated programming offers the ability to cut down the number of bytes used by each record as well as to increase dramatically the number of records a system can control. Advanced dBASE programming becomes highly creative at this level, and a myriad of approaches can be pursued to achieve greater efficiency. Since the details of any of these approaches are so highly technical as to confuse most non-programmers, the nature of the discussion below turns from "How-to" to "What-can-be-done."

Advanced dBASE programming can significantly reduce the number of bytes used by records in the MARC AMC format by creating multiple dependent data bases. Rather than splitting a preformatted entry form such as the one presented in Table 1 (done with simple dependency), multiple dependency uses many dependent data bases to replace short codes in a main data base record.[10]

[10]See "Database Design," chap. 9 in Castro, Hanson, and Rettig, *Advanced Programmer's Guide*.

Generally, any field with standardized, recurring information can be rendered more efficiently using codes. MARC AMC has several such fields, ten of which occur in Tags 600 to 69X, the subject indexing fields of MARC. These fields use 1,510 of the 4,838 bytes in the sample record structure presented in Table 1. The fields contain, for the most part, standardized subject entries that appear repeatedly in many archives. The entries can be replaced, as has been the case at the Goldman Papers, by four-letter mnemonically derived codes with a space separating each code for clarity. For example, the Goldman Papers encodes "Theater" as "thea" and "Berkman, Alexander" as "beal" (i.e., the first two letters of the last and first names). Replacing the text-oriented subject fields with codes allows all ten subject fields to be combined into one. A subject field of 150 bytes can contain a maximum of 15 codes—probably more entries that could be contained in the 1,510 byte text-oriented field without codes.[11]

Figure 2 illustrates the manner in which the codes in the main data base record interact with the same codes in a dependent data base of subjects. Each record in the subject data base should contain at least four fields: the four-letter code, the entry itself, a two-letter code for type of entry (i.e., "PN" equals "Personal name"), and a logical field that queries if subfields have been used. The use of subfields requires ten data bases for each type of entry; the subfield and subject data bases also interact through the four-letter code.

One key advantage of using codes for subjects is that the dependent data base acts as an authority file. During final MARC AMC output, a single entry in the data base will replace all the corresponding codes throughout the entire main data base. Such a find-and-replace operation ensures uniform spelling and form for subject entries. Entering a four-letter mnemonic code allows smaller room for error than correctly retyping the same full entries over and over. Still, errors may slip by and codes may be duplicated. Because of these possibilities for human error, the Goldman Papers employs a menu-driven subject assigning program that searches the dependent data base for code duplications and allows any new entries to be appended immediately to the dependent data base.

The amount of disk space such a system saves depends upon the nature of the archive. For control at the document level, as in the Goldman Papers and in most other documentary editing projects where the materials have a common focus, the great repetition of subject entries results in a dependent data base using only 10 percent or less of the bytes that full subject entries in the main record would require. Archives having a wide diversity of materials would employ a greater number of unique subject entries, thus slightly decreasing the amount of disk space saved.

In addition to reducing the space taken up by subject added entries, dBASE programming can also minimize the disk space used by rarely used fields. The greatest savings of this sort can be achieved in Tags 700 to 752, the added entries, which use up 580 bytes in the sample record structure in Table 1. The approach to rarely used fields differs from that of the ubiquitous subjects. Instead of a large field containing codes, rarely used fields can be best treated by creating a logical field (true or false) in the main data base, a field that essentially asks if added entries exist for the record. If so, a dependent data base for added entries can interact with the main data

[11]The Goldman Papers' computer system is described in Ronald J. Zboray, "Microfilm Editions of Personal Papers and Microcomputers: Indexing the Emma Goldman Papers," *International Journal of Micrographics and Video Technology*, 5 (1986) 213-21.

base using the record number, local system control number, or other information unique to the main record. The added entry dependent data base could have more than one record, each with the interacting code, from each main data base record; each dependent data base record should have a field to identify the type of entry (e.g., personal name or uniform title heading).

A similar approach can be followed for fields that tend toward redundancy, such as Tags 100 to 130 for main entries and Tags 240 to 245 containing various forms of titles. These fields have a total of 720 bytes in the sample form in Table 1. A single field each for main entry and title eliminates much of the redundancy. Adding a two byte field for the corresponding field in MARC AMC (e.g., personal or corporate name, uniform title or title statement) addresses any confusion about the nature of the field. Another single byte logical field can ask if other main entry or title fields have been used. If so, a dependent data base can be constructed to store the extra entries, much as was the case with the rarely used fields.

Paring down the number of fields in this manner not only makes data entry easier, but also dramatically reduces the number of bytes each record requires. Table 2 shows a record structure of 32 fields containing only 1,035 bytes in fixed length fields. With the addition of the estimated 1,024 bytes used in the variable length notes field, the total number of bytes amounts to 2,059—only 42 percent of the 4,838 in the off-the-shelf approach previously described. The reduced number of bytes doubles the maximum number of records on a 30 megabyte hard disk from 5,000 to approximately 10,000. Combined with simple dependency (i.e., splitting the main data base record between two disks), the multiple dependent approach can accommodate on two 30 megabyte hard disks 20,000 records—a number adequate to the needs of many archives with moderate-sized holdings which control data at the collection level. The disk space required to store the multiple dependent data bases can be stored on a third hard disk. As suggested above, the number of bytes the dependent data bases require depends on how many unique entries are used in coded fields and how many times rarely used fields are employed. Even the limit of 20,000 records can be far surpassed if the data base is further split up (i.e., the notes field has its own data base) and distributed between the partitions of a 100 megabyte or greater capacity disk or between several hard disks on a local area network.

Against the advantages of increased record capacity must be placed the significant amount of programming necessary to set up and maintain the multiple data bases. Archives with a staff member adept at moderate-level dBASE programming might find the multiple dependency approach suitable to their needs. Hiring outside programmers may not be worth the expense for an archive aiming to control less than 10,000 records, for whom simple dependency might be more suited. Nevertheless, the authority file-type control offered by using multiple data bases—and the consequent savings in input effort and the decrease in the number of errors—may be attractive to even archives with less extensive holdings.

Approach IV: Menu-Driven Programs

A menu-driven approach to using the MARC AMC format in dBASE III Plus offers a wide number of advantages over the other approaches at the cost of a great deal of program development. A menu-driven program usually can be used by staff members and even volunteers with little or no computing experience. Such a program can include error and format checking routines and "help" options (e.g., the text of relevant portions of the MARC AMC manual or AACR2) in order to guide the user with relatively little effort through the maze of rules and procedures. It takes a die-hard computer aficionado to feel comfortable entering "$z" as subfield delimiters in the middle of otherwise text-based materials; a menu-driven program itself generates this type of input without the assistance of the user.

As another advantage, the menu-driven approach uses disk space much more efficiently than the other methods described above. Not only can this approach employ to the fullest extent multiple dependency, it also can address to a limited degree the problem of fixed fields. One way to accomplish this is to create a record with all the field names listed in Table 1 and to characterize the fields as logical (i.e., a one-byte true or false entry). The logical fields will indicate whether information exists in a dependent data base for that field. Then separate dependent data bases for all the fields, including the notes, may be created that will interact with the main record by its record number or some other number unique to it. The length of the fields in the dependent data bases is optional, because the program can combine several records to reconstruct a single entry; the smaller the length the more the system approaches the disk space usage of a variable-length field based system. A sixty-eight character field width may be adequate for most applications; this length is easy to use since it represents one line of characters across the screen (minus the twelve bytes used for a field name). At the time of data entry the program either can paint the screen with a form with very large spaces for information or it can, in an unformatted mode, prompt the user for information for storage to a memory variable, the contents of which the program will eventually split up into standard field lengths and distribute through the relevant dependent data base. While the amount of disk space such a system will use depends upon the particular application and the length of fields in the dependent data bases, most archives could realize 60 to 80 percent usage of disk capacity, compared to an estimated 5 percent for approaches I and II discussed above.

By making the main data base a logical record and with the increase in disk use efficiency, menu-driven programs far outpace the other approaches in the maximum number of records that can be controlled. Seventy-two logical fields in the main document control record would use only seventy-two bytes; conceivably, a thirty megabyte hard disk could hold about 350,000 such records. Of course, the considerable disk space needed for the dependent data bases must be taken into account, which could limit (to various degrees depending on available hard disk storage space) the total number of possible records. Because each field has a separate data base in such a system, the distribution of disk space is flexible and may be changed over time as needs increase. Distributed hard disks on a network or a partitioned disk of over 100 megabytes certainly would guarantee that the capacity of such a system could surpass 100,000 records. In the best of all possible worlds, with an array of networked 100-megabyte-or-greater capacity disks, the 350,000 record limit would apply.[12]

Figure 3 illustrates one path through a menu-driven system developed by the

[12]Sharing disks over a network can dramatically slow down the performance of programs. With time and particularly optical fiber cabling and the adoption of the new 20386 Intel microchip, network performance will become a smaller hindrance to distributed hard disk resources on local area networks. See William Bates and Andres G. Fortino, *dBASE III Plus & Local Area Networks* (Torrance, Cal.: Ashton-Tate, 1986).

Goldman Papers during 1985 and 1986. The user goes through three menus before entering information in a main document data entry form. Data entry is selected at the first menu and customized entry at the second. The "Regularization" option in the second data entry menu deals with the problem of removing editorial interpolations (attributions in brackets and the use of question marks) in order for dBASE to index names in proper order. The selection of "B" for "Customized data entry" brings up a menu that allows various forms of authority work and error and format checking. The use of the file name extension ".dbf" among the options in the third menu indicates a search of a dependent data base, three of which appear on the menu: biographies (of correspondents), bibliography, and institutions. The customized data entry menu leads to a main document entry form, after which subjects are entered, checked in the corresponding data bases, and the field is alphabetized. A series of error checking routines completes the program, and the user is given the option to edit another record.

Such menu-driven programs reflect the most user-friendly and efficient use of dBASE III Plus for creating MARC AMC format records. The extensive, advanced programming involved in developing such a system, however, makes it viable only for projects with an advanced dBASE programmer on their staff or for larger institutions with the need to control a great many records and the funds to support the on-going development of the program. Perhaps some enterprising, commercial developer will someday create a standardized menu-driven program in the MARC AMC format that can run from within a dBASE III Plus environment. Until then, this approach to creating MARC AMC records in dBASE will be available to only a few archives.

Conclusion

The discussion above has attempted to show that although dBASE can be used to create data bases in the MARC AMC format, such use operates within certain restrictions. dBASE can be used directly off the shelf by archives for this purpose, but at the expense of wasting large amounts of disk storage space. Splitting up the main data base records and distributing them between two hard disks can double the number of records that can be controlled but does little to help the disk wastage problem. A system of multiple dependent data bases begins to address the problem, but it requires a commitment to moderate-level programming that may not be affordable by many archives. A full menu-driven approach, while it provides the most efficient use of disk space, also requires extensive, advanced programming.

The four approaches, however, should not be considered mutually exclusive. Elements from any of the four can be blended to make a system most suited to the needs of a particular archive. So, ironically, the MARC AMC format, which many archivists fear may lead to overstandardization, may in its actual implementation on small data base management systems lead archivists to construct, based on knowledge of their holdings, a myriad of computer applications as unique as their own collections.

TABLE 1: A MODEL OF A DBASE III PLUS DATABASE IN THE MARC AMC FORMAT

dBASE FIELD	BYTES	TAG	MARC AMC FIELD
1. CONTRNO	15	001	Control number
2. DATETIME	16	005	Date and time of latest transaction
3. CATEGMAT	12	007/00	Category of material
		007/01	Specific material designation
		007/02	Original versus reproduction aspect
		007/03	Polarity (microforms)
		007/04	Dimensions (microforms)
		007/05-08	Reduction ratio
		007/09	Color (microforms)
		007/10	Emulsion on film
		007/11	Generation
		007/12	Base of film (microforms)
4. DATAELEM	40	008	[Fixed-length data elements]
		008/00-05	Date entered on file
		008/06	Type of date code
		008/07-10	Date 1
		008/15-17	Place of publication, production, or execution code
		008/18-22	Undefined
		008/23	Form of reproduction code
		008/24-34	Undefined
		008/35-37	Language code
		008/38	Modified source code
		008/39	Cataloging source code
5. LCNUMBER	20	010	Library of Congress control number
6. SYSNUMBER	24	035	Local system control number
7. BIBLEVEL	15	039	Level of bibliographic detail and coding detail
8. CATSOURCE	40	040	Cataloging source
9 LANGCODE	25	041	Language code
10. AREACODE	30	043	Geographic area code
11. CHRONCODE	30	045	Chronological code or date/time
12. GEOCODE	30	052	Geographic classification code
13. SUBCATCODE	40	072	Subject category code
14. LOCALLNUMB	15	09X	Local call numbers
15. PERSNAME	100	100	Main entry - personal name
16. CORPNAME	100	110	Main entry - corporate name
17. CONFNAME	80	111	Main entry - conference or meeting
18. TITLEHEAD	80	130	Main entry - uniform title heading
19. UNITITLE	80	240	Uniform title
20. TRANSTITLE	80	242	Translation of title by cataloging agency
21. COLUNITITL	80	243	Uniform title, collective
22. TITLESTATE	120	245	Title statement
23. PUBLDESC	80	260	Publication, distribution, etc. (imprint)
24. PHYSDESCRP	100	300	Physical description
25. MEDIUM	40	340	Medium
26. ORGANIZTN	150	351	Organization and arrangement
27. NOTES [variable-length memo field]	512+	5XX	[Notes]
		500	General note
		502	Dissertation note
		505	Contents note (formatted)
		506	Restrictions on access
		510	Citation note (brief form) /references
		520	Summary, abstract, annotation, scope, etc., note
		521	Users/intended audience note
		524	Preferred citation of described materials
		530	Additional physical form available note
		533	Reproduction note
		535	Location of originals /duplicates
		540	Terms governing use and reproduction
		541	Immediate source of acquisition
		544	Location of associated materials
		545	Biographical or historical note
		546	Language note
		555	Cumulative index/finding aids note
		561	Provenance
		562	Copy and version identification
		565	Case file characteristics note
		580	Linking entry complexity note
		581	Publications note
		583	Actions
		584	Accumulation and frequency of use
		59X	Local notes
28. SUBADDPN	250	600	Subject added entry - personal name
29. SUBADDCN	200	610	Subject added entry - corporate name
30. SUBADDCF	150	611	Subject added entry - conference or meeting
31. *SUBADDUT*	*150*	*630*	*Subject added entry* - uniform title heading
32. SUBADDTH	250	650	Subject added entry - topical heading
33. SUBADDGN	200	651	Subject added entry - geographic name
34. GENRE	30	655	Genre/form heading
35. OCCUPTN	150	656	Index term - occupation
36. FUNCTION	30	657	Index term - function
37. LOCSUB	100	69X	Local subject added entries
38. ADDPN	100	700	Added entry - personal name
39. ADDCP	100	710	Added entry - corporate name
40. ADDCF	80	711	Added entry - conference or meeting
41. ADDUT	100	730	Added entry - uniform title heading
42. ADDTTD	100	740	Added entry - title traced differently
43. ADDPUB	100	752	Added entry - place of publication or production
44. PHYSACCS	25	755	Physical characteristics access
45. HOSTITEM	80	773	Host item entry
46. LOCATION	80	851	Location
47. ALTGRAPH	25	880	Alternate graphic representation
48. FORGNMARC	60	886	Foreign MARC information field

TABLE 2: A MODEL OF MULTIPLE DEPENDENT DBASE III PLUS DATABASES IN THE MARC AMC FORMAT

dBASE FIELD	BYTES	TAG	MARC AMC FIELD
1. CONTRNO	15	001	Control number
2. DATETIME	16	005	Date and time of latest transaction
3. CATEGMAT	12	007/00	Category of material
		007/01	Specific material designation
		007/02	Original versus reproduction aspect
		007/03	Polarity (microforms)
		007/04	Dimensions (microforms)
		007/05-08	Reduction ratio
		007/09	Color (microforms)
		007/10	Emulsion on film
		007/11	Generation
		007/12	Base of film (microforms)
4. DATAELEM	40	008	[Fixed-length data elements]
		008/00-05	Date entered on file
		008/06	Type of date code
		008/07-10	Date 1
		008/15-17	Place of publication, production, or execution code
		008/18-22	Undefined
		008/23	Form of reproduction code
		008/24-34	Undefined
		008/35-37	Language code
		008/38	Modified source code
		008/39	Cataloging source code
5. LCNUMBER	20	010	Library of Congress control number
6. SYSNUMBER	24	035	Local system control number
7. @BIBLEVEL	1	(039)	Level of bibliographic detail and coding detail
8. @CATSOURCE	1	(040)	Cataloging source
9 @LANGCODE	1	(041)	Language code
10. AREACODE	30	043	Geographic area code
11. CHRONCODE	30	045	Chronological code or date/time
12. GEOCODE	30	052	Geographic classification code
13. @SUBCATCODE	1	(072)	Subject category code
14. LOCALLNUMB	15	09X	Local call numbers
15. #MAINENTRY	150	(100-30)	Main entry
16. #METYPE	2		Main entry type: personal name, corporate name, conference or meeting
17. #ADDMAIN	1		Main entry - additions
18 #TITLE	150	(240-45)	Title
19. #TITLETYPE	2		MARC AMC type of title
20. #ADDTITLE	1		Title - additions
21. @PUBLDESC	1	(260)	Publication, distribution, etc. (imprint)

dBASE FIELD	BYTES	TAG	MARC AMC FIELD
22. PHYSDESCRP	100	300	Physical description
23. MEDIUM	40	340	Medium
24. ORGANIZTN	150	351	Organization and arrangement
25. NOTES [variable-length memo field]	512+	5XX	[Notes]
		500	General note
		502	Dissertation note
		505	Contents note (formatted)
		506	Restrictions on access
		510	Citation note (brief form) /references
		520	Summary, abstract, annotation, scope, etc., note
		521	*Users/intended audience note*
		524	Preferred citation of described materials
		530	Additional physical form available note
		533	Reproduction note
		535	Location of originals /duplicates
		540	Terms governing use and reproduction
		541	Immediate source of acquisition
		544	Location of associated materials
		545	Biographical or historical note
		546	Language note
		555	Cumulative index/finding aids note
		561	Provenance
		562	Copy and version identification
		565	Case file characteristics note
		580	Linking entry complexity note
		581	Publications note
		583	Actions
		584	Accumulation and frequency of use
		59X	Local notes
26. %SUBJECTS	150	(600-9X)	Subject added entry
27. @ADDENTRY	1	(700-52)	Added entries
28. PHYSACCS	25	755	Physical characteristics access
29. %HOSTITEM	4	(773)	Host item entry
30. %LOCATION	10	(851)	Location
31. @ALTGRAPH	1	(880)	Alternate graphic representation
32. @FORGNMARC	1	(886)	Foreign MARC information field

dBASE field names begining with @, #, and % differ from those presented in Table 1. Those fields with @ are one byte logical fields (*true or false*) which indicate if information pertaining to the field has been entered in a dependent database. Fields begining with % are filled with four-letter codes which interact with dependent databases. Fields with # are combinations of potentially redundant fields and occur in series of three: 1) the field; 2) code for the corresponding MARC AMC field; and 3) a logical field indicating whether more information has been entered in a dependent database. Tag numbers enclosed by parentheses shows that the *fields represented must be manipulated by a program in order to output information in proper* MARC AMC format.

FIGURE 1: A DBASE III PLUS PROGRAM TO CREATE MARC AMC FORMAT DATA ENTRY FORMS USING SIMPLE DATABASE DEPENDENCY

```
@  0,  0  SAY  "CONTR"
@  0, 12  GET  ONE->CONTR
@  1,  0  SAY "DATETIME"
@  1, 12  GET  ONE->DATETIME
@  2,  0  SAY "CATEGMAT"
@  2, 12  GET  ONE->CATEGMAT
@  3,  0  SAY "DATAELEM"
@  3, 12  GET  ONE->DATAELEM
@  4,  0  SAY "LCNUMBER"
@  4, 12  GET  ONE->LCNUMBER
@ 5,  0  SAY "SYSNUMBER"
@  5, 12  GET  ONE->SYSNUMBER
@  6,  0  SAY "BIBLEVEL"
@  6, 12  GET  ONE->BIBLEVEL
@  7,  0  SAY "CATSOURCE"
@  7, 12  GET  ONE->CATSOURCE
@  8,  0  SAY "LANGCODE"
@  8, 12  GET  ONE->LANGCODE
@  9,  0  SAY "AREACODE"
@  9, 12  GET  ONE->AREACODE
@ 10,  0  SAY "CHRONCODE"
@ 10, 12  GET  ONE->CHRONCODE
@ 11,  0  SAY "GEOCODE"
@ 11, 12  GET  ONE->GEOCODE
@ 12,  0  SAY "SUBCATCODE"
@ 12, 12  GET  ONE->SUBCATCODE
@ 13,  0  SAY "LOCALLNUMB"
@ 13, 12  GET  ONE->LOCALLNUMB
@ 14,  0  SAY "PERSNAME"
@ 14, 12  GET  ONE->PERSNAME
@ 16,  0  SAY "CORPNAME"
@ 16, 12  GET  ONE->CORPNAME
@ 18,  0  SAY "CONFNAME"
@ 18, 12  GET  ONE->CONFNAME
@ 20,  0  SAY "TITLEHEAD"
@ 20, 12  GET  ONE->TITLEHEAD
@ 22,  0  SAY "UNITITLE"
@ 22, 12  GET  ONE->UNITITLE
@ 24,  0  SAY "TRANSTITLE"
@ 24, 12  GET  ONE->TRANSTITLE
READ
@  1,  0  SAY "COLUNITITL"
@  1, 12  GET  ONE->COLUNITITL
@  3,  0  SAY "TITLESTATE"
@  3, 12  GET  ONE->TITLESTATE
@  5,  0  SAY "PUBLDESCR"
@  5, 12  GET  ONE->PUBLDESCR
@  7,  0  SAY "PHYSDESCRP"
@  7, 12  GET  ONE->PHYSDESCRP
@  9,  0  SAY "MEDIUM"
   [CONTINUED IN NEXT COLUMN]
```

```
   [CONTINUED  FROM  PREVIOUS
     COLUMN]
@  9, 12  GET  ONE->MEDIUM
@ 10,  0  SAY "ORGANIZTN"
@ 10, 12  GET  ONE->ORGANIZTN
@ 13,  0  SAY "NOTES"
@ 13, 12  GET  TWO->NOTES
@ 14,  0  SAY "SUBADDPN"
@ 14, 12  GET  TWO->SUBADDPN
@ 18,  0  SAY "SUBADDCN"
@ 18, 12  GET  TWO->SUBADDCN
@ 21,  0  SAY "SUBADDCF"
@ 21, 12  GET  TWO->SUBADDCF
@ 24,  0  SAY "SUBADDUT"
@ 24, 12  GET  TWO->SUBADDUT
READ
@  2,  0  SAY "SUBADDTH"
@  2, 12  GET  TWO->SUBADDTH
@  6,  0  SAY "SUBADDGN"
@  6, 12  GET  TWO->SUBADDGN
@  9,  0  SAY "GENRE"
@  9, 12  GET  TWO->GENRE
@ 10,  0  SAY "OCCUPTN"
@ 10, 12  GET  TWO->OCCUPTN
@ 13,  0  SAY "FUNCTION"
@ 13, 12  GET  TWO->FUNCTION
@ 14,  0  SAY "LOCSUB"
@ 14, 12  GET  TWO->LOCSUB
@ 16,  0  SAY "ADDPN"
@ 16, 12  GET  TWO->ADDPN
@ 18,  0  SAY "ADDCP"
@ 18, 12  GET  TWO->ADDCP
@ 20,  0  SAY "ADDCF"
@ 20, 12  GET  TWO->ADDCF
@ 22,  0  SAY "ADDUT"
@ 22, 12  GET  TWO->ADDUT
@ 24,  0  SAY "ADDTTD"
@ 24, 12  GET  TWO->ADDTTD
READ
@  1,  0  SAY "ADDPUB"
@  1, 12  GET  TWO->ADDPUB
@  3,  0  SAY "PHYSACCS"
@  3, 12  GET  TWO->PHYSACCS
@  4,  0  SAY "HOSTITEM"
@  4, 12  GET  TWO->HOSTITEM
@  6,  0  SAY "LOCATION"
@  6, 12  GET  TWO->LOCATION
@  8,  0  SAY "ALTGRAPH"
@  8, 12  GET  TWO->ALTGRAPH
@  9,  0  SAY "FORGNMARC"
@  9, 12  GET  TWO->FORGNMARC
```

FIGURE 2: A MODEL OF MULTIPLE DATABASE DEPENDENCY OF MARC AMC SUBJECT FIELDS

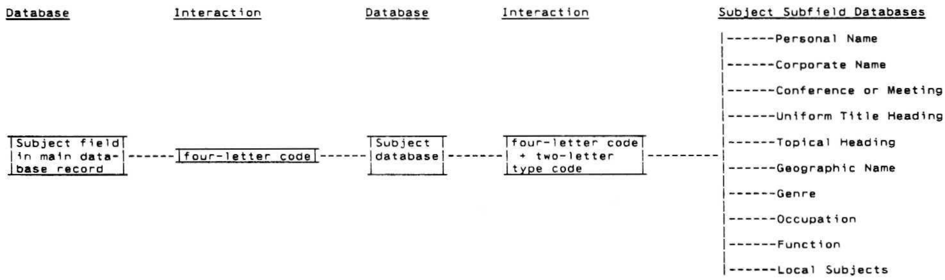

FIGURE 3: SAMPLE SESSION FROM THE EMMA GOLDMAN PAPERS MENU-DRIVEN PROGRAM