pology-based approach were made by Drs. U. Doelling and I. Wilhelm and M. Jaeger.

REFERENCES AND NOTES

(1) International Documentation in Chemistry, Otto Volger Strasse 19, 6231 Sulzbach/Taunus, Federal Republic of Germany.
(2) Morgan, H. L. "The Generation of a Unique Machine Description for Chemical Structures—a Technique Developed at Chemical Abstracts Service". *J. Chem. Doc.* **1965**, *5*, 107–113.
(3) Fugmann, R.; Kusemann, G.; Winter, J. H. "The Supply of Information on Chemical Reactions in the IDC System". *Inf. Process. Manage.* **1979**, *15*, 303–323. Fugmann, R. "The IDC System". In *Chemical Information Systems*; Ash, J. E., Hyde, E., Eds.; Ellis Horwood: Chichester, U.K., 1975; p 195.
(4) See also: Valls, J.; Schier, O. "Chemical Reaction Indexing". In *Chemical Information Systems*; Ash, J. E., Hyde, E., Eds.; Ellis Horwood: Chichester, U.K., 1975; p 255. Valls, J. "Reaction Documentation". In *Computer Representation and Manipulation of Chemical Information*; Wipke, W. T., Heller, St. R., Hyde, E., Eds.; Wiley: New York, 1973; p 92.
(5) See also: McGregor, J. J.; Willett, P. "Use of a Maximal Common Subgraph Algorithm in the Automatic Identification of the Ostensible Bond Changes Occurring in Chemical Reactions". *J. Chem. Inf. Comput. Sci.* **1981**, *21*, 137–140. (Here, the algorithmic identification of reaction sites is considered "sufficiently precise".)
(6) Vogt, F. BASF Report, 1968, unpublished.

Data Access Subroutine Package for Spectrometric Data Bases

CHENG QIAN[‡] and CHARLES L. WILKINS*

Department of Chemistry, University of California, Riverside, California 92521

Received April 14, 1987

A data access subroutine package for spectrometric data bases has been developed. The subroutines are high-level language callable. They can transform logical records of a data file to virtual records consisting of selected data fields with user-specified data format for further processing or, conversely, fill a record of a data file with data transformed from a virtual record. By careful balance of its functions and the overheads, the package has been made very compact, resulting in low development cost and reasonable execution efficiency. Application examples are given to show its versatility and usefulness.

INTRODUCTION

During the past decade, a number of multispectrometric data-base systems have been implemented. Some examples are CIS of NIH/EPA,[1,2] Pluridata in the DARC system,[3] and a retrieval system developed by Koptyung and co-workers.[4] These systems integrate large spectrometric data bases of tens of thousands of spectra as well as related structural data bases. They can retrieve the data in an interactive mode by using their built-in search and retrieval software. The software is very user-friendly. However, this software works in a "closed" environment where users can search data using only procedures dictated by the software and retrieve the data only in a predetermined format. Molecular Design Ltd. has developed a more flexible system, called MACCS, that can be used for structural and associated textual databases[5] but is still end user oriented, although it, too, is a closed environment because of its proprietary nature.

In some research laboratories, users work in an "open" environment, where they may actually be developers of new library search algorithms or other application programs in which data retrieval is only the first step for further program operations. Therefore, more flexible data-base software is needed. Such software should function as an interface between user application programs and data bases, keeping both independent from each other. This is, in fact, a part of the goals of various data base management systems (DBMS).[6] Using a relational DMBS, Morffew and others have handled molecular geometry data in a molecular graphics system.[7] However, there is still a lack of technically and economically suitable commercial DBMS for large multispectrometric and structural data-base management. Most commercial DBMS are based upon algorithms that combine a set of individual simple keyword searches by Boolean operations. They can hardly support spectrometric and structural data bases, owing to the special characteristics of queries required for these data bases. Table I lists various types of queries and typical search methods, showing that present DBMS support the queries listed in columns 1 and 3. In contrast, the queries raised frequently with respect to spectrometric and structural data bases are those in columns 4 and 5, sometimes simplified to those listed in column 2 for presearch purposes. Thus, the DBMS are of limited utility for spectral library searches. Moreover, because they usually require much CPU time, they can hardly meet the time-response criteria necessary to make possible searching of large spectral and structural libraries on supermini computers that are popular in chemistry laboratories. In other words, the use of commercially available DBMS for spectrometric and structural data bases is generally expensive and ineffective, if not impossible. Therefore, we have developed a flexible software tool that provides data access facilities to user programs and includes the potential for future expansion of searching features. This is the idea behind the data access subroutine package (DASP) described here.

DASP's approach is to provide a set of compact and efficient application program callable subroutines for convenient data access while maintaining data-program independence and allowing search algorithms of all varieties to be realized in the user's programs. Because variable-length fields of binary data are used extensively in spectrometric and structural data-base organization, equal attention has been paid to data processing of fixed-length as well as variable-length fields. Furthermore, DASP is carefully designed to fit the needs of integrating and reorganizing multisource data bases, such as those that are required to support integrated gas chromatography–infrared mass spectrometry systems.[8]

PRINCIPAL FEATURES OF DASP

Data Model. Spectrometric or structural data often aggregate as units of "spectra" or "structures". Data from different spectra or structures are seldom related, in terms of either data organization or application. On the other hand,

[‡]Present address: Chemical Abstracts Service, P.O. Box 3012, Columbus, Ohio 43210.

Table I. Queries and Data-Base Search Methods

		n-dimensional			
query type	1D simple	*n* 'YES'/'NO'	*n* separable	one *n*-D query	complex query
commonly used expression	single keyword	existence table or saturated bar graph	Boolean expression	parameter table or vector bar graph	graph or pattern expressions
essential search algorithm	inverted file search	screen search	inverted file search	sequential search, some inverted file, or screen presearch	sequential search, some inverted file, or screen presearch
discrimination basis	keyword match	screen match	keyword match and set operations	*n*-D distance, inner product, or other *n*-D function	various pattern recognition methods, including A-A match
examples of application	author index search	functional group search	multikeyword bibliographic search	spectrometric library search	structure/substructure search, image pattern search

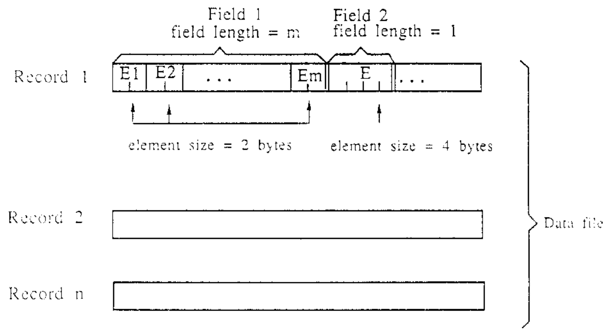

Figure 1. Record-based data model.

a file record is probably the storage unit most conveniently accessible by using either high-level language I/O functions or operating system services. Therefore, it is quite natural that a logical record of physical data files and their memory counterparts, virtual records (or "views" of the physical records), are adopted to be the basic data units that DASP reads, writes, and transforms.

A DASP record consists of data fields. Each of them accommodates one or more data elements of the same data type and the same element size. The current version of DASP accepts integer and character data types. The sizes of integer and character elements range from 1 to 8 bytes and from 1 to 32 767 bytes, respectively. Records with identical data field configurations comprise a data file that completes DASP's relational data model (Figure 1).

Record Operations. DASP's basic record operations contain read/write operations and record transformation–projections. A read/write operation is simply a data transfer between a logical record and a virtual record. A projection, carried out between two virtual records, is a key function of DASP software. According to the requirements of a destination record, the DASP projection module selects data fields from a source record, resequences them, transforms their formats to conform with the destination record format, and finally files them in the destination record.

The data format transformations performed by DASP are truncation, padding, binary data element repacking, and character string translations between EBCDIC and ASCII. Truncation/padding of a field or an element is automatically effected when corresponding destination and source fields are of different field lengths or they have unequal element sizes. Binary data repacking and character code translations are undertaken at the user's request.

DASP Control Information. When DASP subroutines are called by a user program to access data, they use the information provided by the user in control files. For either a source or a destination virtual record involved in a projection, a control file called the record description file (RDF) is required. It contains each data field's attributes, such as data type, element size, field length (number of elements in the field), and word alignment. Another control file is the transformation description file (TDF). A TDF specifies the correspondences between source and destination fields of a projection and the translations or repackings to be performed for each field. In DASP control files, a virtual record is identified by its RDF's name, and data fields are identified by their identification numbers, which appear in the RDF. When DASP operations involve logical record accesses, the physical file containing the records should be opened. Hence, a control file describing the file's characteristics (FAF) is also needed. The FAF keeps information about file names, file organization, file status, file record type, etc. Examples of the three types of control files are shown in Appendix I with brief explanations.

Before a user program calls any DASP module to access records, it should invoke a specific DASP initialization module to set up a "channel" for the operations. Setting up a channel requires defining some data structures as the channel's control tables and reading data from the control files into the tables. There are three tables for each channel: a file description table (FDT), a source record description table (SDT), and a destination record description table (DDT). These tables save the control data in a format convenient for later reference by other DASP modules. Although a channel supports different DASP molecules for different record operations, as long as the same set of control information is involved, a user program can initialize several channels. Each channel is identified by its channel number.

Fixed- and Variable-Length Records. The sequence of data fields appearing in a record is in accordance with their identification numbers. Field 1 starts at the first byte of the record. Each successive field begins at the location immediately following the previous field. Obviously, the relative addresses of the fields in a record can be calculated if the byte count of each field is known. However, the byte count of each field equals the field's length times the element size, plus some optional word alignment bytes, designated by the field's word alignment attribute in the RDF. Consequently, it is necessary to know the length of each field in a record. The length of a fixed field is specified by the absolute value of its field length attribute, while for a variable-length field this value points to a field of which the first element is the actual length specifier. The sign of the field length attribute denotes whether the field is fixed or variable (i.e., a negative sign means a variable-field length).

With this mechanism, lengths of variable fields are dynamically acquired by DASP from the record being accessed. Relative addresses of all fields are also calculated dynamically. This arrangement is adaptable to most spectrometric and structural data organizations, though it imposes the following limitations: (a) nested variable-length fields are not allowed; (b) in a record, a variable-length field must be located behind its length specifier.

Shared Accesses of Files and Records. Two types of file sharing are provided by DASP. For a channel initialized for reading, its associated data file can be either read or written by other channels or user programs. For a channel initialized

Table II. Summary of DASP Subroutines

module name	module level		control table
DASP_OPEN	primary	initializes a DASP channel, defines its control tables; for a channel involving logical record accesses, opens the associated data file	FDT, SDT, DDT loaded
DASP_READ	primary	copies data from a logical record of a physical data file to a virtual record; source records can be accessed sequentially, directly, or by key, so long as the access mode is compatible with the file's organization	FDT used
DASP_WRITE	primary	writes or rewrites a logical record of a physical data file by sequential, direct, for keyed access	FDT used
DASP_TRANS	primary	projects a source virtual record to a destination virtual record with necessary data format transformations	SDT, DDT used
DASP_CLOSE	primary	resets a DASP channel	FDT, SDT, DDT removed
DASP_GET	secondary	equivalent to DASP_READ plus DASP_TRANS	FDT, SDT, DDT used
DASP_PUT	secondary	equivalent to DASP_TRANS plus DASP_WRITE	FDT, SDT, DDT used

for writing, the associated data file can only be shared for reading. Once writing to a record is begun by a user program through one channel, the record may not be accessed by other channels or programs until the writing operation finishes. This interlock function guarantees data integrity within a record. However, users still must provide for data integrity. If a data integrity problem involves more than one record, an optional DASP read/write lock function is available.

DASP IMPLEMENTATION

DASP Modules. DASP is a modular software package. For the current version, seven application program callable subroutines have been built. Two of them are secondary subroutines in which some primary subroutines are called. All these modules are programmed in VAX FORTRAN 77 and have been implemented on both VAX 11/785 and MicroVax II computers operating under the VMS operating system. The functions of the Digital Equipment Record Management Services (RMS) software are extensively utilized. Generally, VMS/RMS services are called implicitly through FORTRAN I/O statements. However, there are some cases where a DASP module calls VMS or RMS services explicitly in a "call" statement of FORTRAN. All the DASP modules are placed in an objective library and need to be linked to application programs by the VAX link utility. DASP modules take about 100–200 kB of memory, depending on how many and which modules being called.

Arguments are passed to and from DASP subroutines in compliance with VAX FORTRAN argument passing conventions; i.e., character arguments are passed by descriptors, and numerical arguments are passed by reference. By observing these conventions, application programs written in languages other than FORTRAN can easily call DASP subroutines by means of VMS-provided special functions, such as %REF, %VAL, %DESCR, and LOC. Any DASP subroutine module also can be called as an integer function. Upon completion, it returns a completion code indicating successful execution (code = 0) or an error message (code $\neq$ 0), which is convenient for program debugging.

Table II summarizes the features of DASP modules. The secondary subroutines DASP_GET and DASP_PUT are combinations of DASP_READ with DASP_TRANS and DASP_TRANS with DASP_WRITE. In DASP_GET, DASP_READ is first called to copy a whole source logical record of a physical data file to an intermediate virtual record without any data format transformations. Subsequently, DASP_TRANS is called to transfer the intermediate record to the destination virtual record. DASP_PUT works similarly but in the reverse direction. However, the intermediate record is transparent to users. What users see is a logical record being translated to a virtual record (its view). In this context, a projection is also applicable to a logical record and so is an RDF.

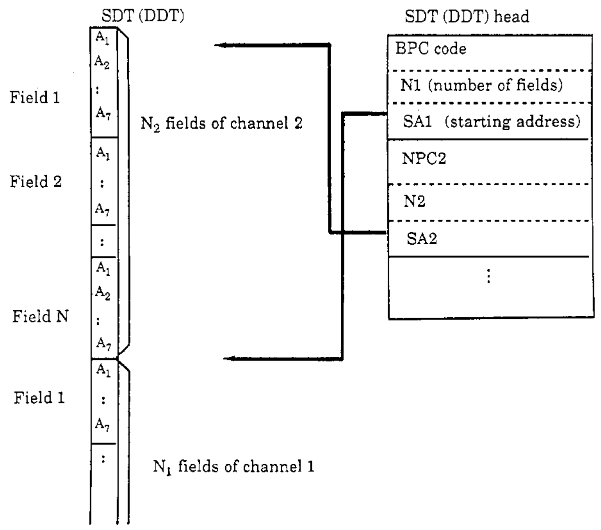

Figure 2. SDT(DDT) structures.

The primary subroutine DASP_READ is responsible for all read accesses of data records. The access mode can be sequential, direct, or keyed. A sequential file allows sequential and direct accesses to its records. An index file permits keyed accesses and sequential accesses. In reading an index file in keyed access mode, a keyword value should be specified for a field that is one of the physical index fields of the file. An additional access mode of DASP_READ is "rewind", which resets the current record number to the first record of the sequential file so that it can be read from the beginning.

Data Structure. Table II also shows that every DASP subroutine works with some of the previously mentioned control tables. The data structures of these control tables have significant importance for the performance of DASP modules.

The control table FDT has a fixed-length data structure, containing 15 information items about the data file to be accessed. If the file is a new index file, index key descriptors are also included in the FDT. Although DASP FDT structures support creation of simple index files with only one primary and one secondary key, any existing index files that are acceptable to RMS can be read or written by DASP modules without limitation. To create more complicated index files, the RMS utility must be invoked in advance.

The SDT and DDT control tables have variable-length structures. SDT and DDT table heads keep their table starting addresses, numbers of fields in each table, and the binary data packing convention code (IBM/DEC). In an SDT or DDT table, each data field is described by seven attributes, including data type, element size, field length, etc. A DDT table contains field correspondences, data format transformation types of the defined projection, and destination field attributes as well (Figure 2). Obviously, FDT and SDT accommodate information from control files FDF and source RDF, respectively, and DDT accommodates that from both destination RDF and TDF.

APPLICATION OF DASP

(1) ^{13}C NMR Library Search. DASP has been used for carbon 13 nuclear magnetic resonance (^{13}C NMR) library search applications. The particular library used was that compiled through the cooperative efforts of the United States Environmental Protection Agency (EPA) and the National Institutes of Health.[2] The version used in our laboratory contains about 11 800 spectra. In this library, each spectrum is represented by one record that contains the compound's serial number, the Chemical Abstracts Service (CAS) Registry number, the number of carbon resonances, chemical shifts, multiplicities, and some other miscellaneous numeric and textual data. Because the reverse algorithm for mixture analysis being tested required only four data elements (serial number, CAS Registry number, number of resonances, and chemical shift values), those fields were extracted from the larger file by using DASP. This permitted tight coupling of the search programs without restructuring the original library file. Previously, any changes of data format in the file required search program modification. By utilizing DASP subroutines, the revised program can read the compressed EPA ^{13}C NMR library and get virtual records consisting of only the four data fields in the required data format. By means of the DASP interfacing, the program is now independent from the format of data in the library; this significantly improves program development efficiency and data-base maintenance. The cost for this improved flexibility provided by DASP is CPU time overhead. Compared to 84 s for the previous search program, a sequential search of the library of 11 682 spectra now takes about 160 s of CPU time on the MicroVax II. However, the 7 ms/record CPU overhead is very reasonable, demonstrating fairly good DASP efficiency. It is even close to meeting the real-time response requirements.

(2) Mass Spectrometric Data Compression. A mass spectral library from the National Bureau of Standards was transformed from an exchange format to a compressed internal format for storage, search, and retrieval. The usage of DASP modules has largely simplified the programming. Another advantage is that the program can be used for some other data exchange formats without program modification, recompiling, or linking. The only thing that need be done is to modify the control files (RDF and TDF).

DISCUSSION AND CONCLUSIONS

DASP works as an interface between an application program and spectrometric data bases. It frees application programs from data access procedures and allows concentration on more essential functions. Meanwhile, it maintains programs and data bases independent from each other. An application program can easily access a data base comprised of spectral libraries from different sources and having different data formats by simply calling the DASP_GET module to project them to virtual records of identical formats (Figure 3). The good performance of DASP modules and the low cost of their development are achieved by carefully balancing the DASP functions to be implemented and the overhead. Versatility is somewhat reduced in exchange for higher efficiency and lower cost. However, it is adequate for spectrometric and structural data bases. DASP uses a simplified file record based data model; it supports only projection for record transformations; it works on temporary and localized data control tables, instead of a global data dictionary; and it utilizes DEC RMS for I/O support. All these contribute to the improvement of run-time performance and to simplification of the programming using the package. Unfortunately, these features require some re-

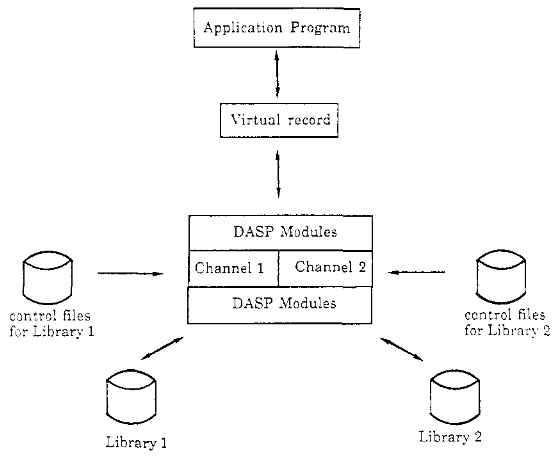

Figure 3. Accesses to multisource data.

strictions of application and software transportability.[9] The application results discussed here provide some measure of the package's performance and overhead. However, comparison of the search time of any specific program using DASP with other existing search systems specifically optimized for particular libraries is unrealistic, because search time is highly dependent upon both system hardware and software, as well as the detailed data and file structures. On the other hand, achievement of a search time only double that of an optimized search program for the present applications, considering the generality of the DASP approach, is encouraging. Additional improvements in speed can be visualized: first, the input–output parameters used by the RMS routines could be optimized (especially those for looking up definition tables and mapping source/destination fields); second, utility programs for the generation of definition tables could be added. A natural and logical extension would be addition of some commonly used search algorithm features and a set of library search modules, specific for certain spectrometric data bases, such as mass, infrared, and NMR spectra.

ACKNOWLEDGMENT

We thank Dr. David Laude and John Cooper, who devoted their time and contributed C-13 NMR and mass spectral library search programs to help test the DASP package. We also thank Steven Steiner for very constructive discussions. Partial support was received under Cooperative Agreement CR-811370 between the University of California, Riverside, and the Environmental Protection Agency Environmental Monitoring Systems Laboratory, Las Vegas, and Cooperative Agreement CR-813714 between the University of California, Riverside, and the Environmental Protection Agency Laboratory, Research Triangle, NC. Although the research described in this paper has been supported wholly or in part by the United States Environmental Protection Agency, it has not been subjected to Agency review and therefore does not necessarily reflect the views of the Agency, and no official endorsement should be inferred.

APPENDIX I

File description file, record description file (MASS1.RDF, source RDF in the TDF shown), record description file (MASSCMPR.RDF, destination RDF in the TDF shown), and transformation description file are shown in Tables III–VI, respectively.

APPENDIX II

DASP Return Codes.

(1) Fail to open a sequential data file.
(2) Fail to open an index data file.

Table III. File Description File (FDF)

```
    $FILE_ATTRIBUTE[a]
    DATA_FILE_NAME='DISK1:[DATABANK,MASS]
      MASSCHPR.DAT'[b]
      FILE_ORGAN='INDEXED'[c]
      REC_TYPE='VARIABLE'[d]
      REC_FORM='UNFORMATED'[e]
      REC_LENGTH=918[f]
      FILE_STATUS='NEW'[g]
      FILE_ACCESS='KEYED'[h]
      FILE_INIT=4000[i]
      FILE_EXT=1000[j]
      FILE_OPERATION='W'[k]
      MULTI_BLOCK=2[l]
      MULTI_BUFFER=4[m]
      KEY=1,4,5,8[n]
      KEY_TYPE='INTEGER','INTEGER'[o]
    $END[a]
```

[a] Begin and End statements. [b] Name of the data file, 70 characters maximum. [c] File organization ('INDEXED'; 'SEQUENTIAL', 'RELATIVE'; default 'SEQUENTIAL'). [d] Record type ('FIXED', 'VARIABLE'; default 'VARIABLE'). [e] Record format ('FORMATED', 'UNFORMATED'; default 'UNFORMATED'). [f] Record length, integer number smaller than 32 767, default 80. [g] File status ('NEW', 'OLD', 'UNKNOWN'; default 'OLD'). [h] File access model ('SEQUENTIAL', 'DIRECT', 'KEYED'; default 'SEQUENTIAL'). [i] File initial size, integer number, default 40 blocks. [j] File extent size, integer number, default 40 blocks. [k] File access operation, ('R' for read only, 'W' for read/write; default 'R'). [l] Block count for physical transfer, integer number, RMS system default. [m] Buffer count for physical transfer, integer number, RMS system default. [n] Primary and secondary index per field specification: primary start, primary finish, secondary start, secondary finish (byte address). [o] Primary and secondary index key type specifications ('INTEGER', 'CHARACTER').

(3) A file organization not accepted by DASP (e.g., relative).
(5) Fail to open a record description file (RDF).
(6) Fail to open a file description file (FDF).
(7) Fail to open a transformation description file (TDF).
(8) Format error in a record description file (RDF).
(9) Format error in a transformation description file (TDF).
(10) Try to initialize a channel already initialized.
(11) Specified data transformation not relevant to the data type.
(12) Data file read error.
(13) Specified record not found in direct or keyed access.
(14) Record to be rewritten not found.
(15) Data file write error.
(16) Error in specifying the length of variable fields.
(17) File description file (FDF) or transformation description file (TDF) required for channel initialization not available.
(18) Channel to be closed is not an initialized channel.
(19) Channel control table overflow.
(20) Too large channel ID number.
(100) End of file encountered in sequential access.

APPENDIX III

DASP Subroutine Summary. (1) DASP-OPEN (CHANNEL, FILE-NAME, TRANS-NAME, IOS).

Function. Initialization of a DASP channel. Optionally, open an associated data file.

Arguments.

CHANNEL: Channel ID number. A 4-byte integer.

FILE-NAME: Name of the file to be opened, which accommodates logical records associated with the channel. A string of maximum 70 characters (optional).

TRANS-NAME: Name of a transformation description file (TDF) that defines the channel's record transformation. A string of maximum 70 characters (optional).

IOS: A 4-byte integer. File opening status provided by Digital RMS.

(2) DASP_READ (CHANNEL, MODE, KEY_READ, KEY_VALUE, IKEY_VALUE, BUF, BUFSIZE, LREAD, IER).

Function. Transfer of data from a logical record (physical data file record) to a virtual record. The source record can be accessed sequentially, directly, or by index key, as long as the access mode conforms to the file organization.

Arguments.

CHANNEL: Channel ID number. A 4-byte integer.

MODE: Access mode (S for sequential access; D for direct access; K for keyed access; and SL, DL, or KL for sequential, direct, or keyed access with automatic record lock if the channel is initialized for read/write operations). A string of one or two characters.

KEY_RECID: Key ID of an index file to be accessed or serial number of the logical record to be accessed for direct access. A 4-byte integer.

KEY_VALUE: Value of a character key for keyed read of an index file record. A string of maximum 255 characters.

IKEY_VALUE: Value of an integer key for keyed read of an index file record.

BUF: Array to accommodate the virtual destination record. A variable-length byte array of size specified by BUFSIZE.

BUFSIZE: Size of array named BUF. A 4-byte integer.

LREAD: Actual size of the transferred destination record. A 4-byte integer.

IER: File reading status provided by Digital Equipment RMS. A 4-byte integer.

(3) DASP_WRITE (CHANNEL, MODE, KEY_RECID, Key_VALUE, IKEY_VALUE, BUF, RECL, IOS).

Function. Write or rewrite a logical record of a physical data file by sequential, direct, or keyed access.

Arguments.

CHANNEL: Channel ID number. A 4-byte integer.

MODE: Access mode (S for writing a new record of a sequential file; K for writing a new index file record; D for writing or rewriting a sequential file by direct access; R for rewriting a record of an index file; and SL, DL, KL, RL for writing with automatic lock of accessed record until access of another record through the same channel). A series of one or two characters.

KEY_RECID: Index key ID for keyed write of an index file or serial number of a logical record for direct write. A 4-byte integer.

DKEY_VALUE: Value of a character key for keyed rewrite of an index file record.

IKEY_VALUE: Value of an integer key for keyed rewrite of an index file record.

BUF: Array to accommodate the virtual source record. A variable-length byte array of size defined by BUFSIZE.

BUFSIZE: Size of array named BUF. A 4-byte integer.

RECL: Size of the source virtual record to be written or rewritten.

IOS: File writing sections provided by Digital RMS. A 4-byte integer.

(4) DASP__TRANS (CHANNEL, SOURCE__BUF, SBSIZE, DES_BUF, DBSIZE, CURRET).

Function. Projecting a source virtual record to a destination virtual record according to the channel's control data.

Arguments.

CHANNEL: Channel ID number. A 4-byte integer.

SOURCE-BUF: Array to accommodate the source record. A variable-length byte array.

Table IV. Record Description File (RDF): MASS1.RDF (Source RDF in the TDF Shown)

! DEFINITION FILE FOR MASS SPECTROMETRIC DATA BASE FROM NBS[a]	
IBM[b]	TAPE ON IBM MACHINE[c]
IO4[d] 1,[e]	1[f] LENGTH OF RECORD THAT FOLLOWS, IN WORDS[g]
IO4 1,	2 LENGTH OF HEADER THAT FOLLOWS, IN WORDS
IO4 1,	3 SPECTRUM NUMBER
IO4 1,	4 CAS NUMBER
IO4 1,	5 EPA ACCESSION NUMBER
EBC 3,	6 COLLECTION LETTERS
EBC 1,	7 OWNER FLAG
IO4 1,	8 COLLECTION NUMBER
EBC 10,	9 INSTRUMENT
EBC 10,	10 INLET TYPE
IO2 1,	11 MOLECULAR WEIGHT
IO2 1,	12 QUALITY INDEX
IO2 1,	13 INLET TEMPERATURE (DEGREE, C)
IO2 1,	14 SOURCE TEMPERATURE (DEGREE, C)
IO2 1,	15 ELECTRON VOLTAGE (eV)
IO2 1,	16 PRESSURE (MANTISSA)
IO2 1,	17 PRESSURE (EXPONENT)
IO2 1,	18 ION ACCELERATING VOLTAGE (kV×100)
IO2 1,	19 TRAP CURRENT (micro A)
IO2 1,	20 unused
IO2 1,	21 NUMBER OF COUNT FIELDS (2 BYTES EACH)
IO2 1,	22 LENGTH OF FEATURE CODE
IO2 1,	23 COUNT OF PEAKS
IO2 1,	24 COUNT OF MOLECULAR FORMULA (BYTES)
IO2 1,	25 COUNT OF WLN (BYTES)
IO2 1,	26 LENGTH OF NOMENCLATURE (BYTES)
IO2 1,	27 LENGTH OF CONTRIBUTOR (BYTES)
IO2 1,	28 unused
4IO4[h] -23,[i]	29 MASS/INTENSITY PAIRS
4EBC -24,	30 MOLECULAR FORMULA
4EBC -26,	31 NOMENCLATURE
4EBC -27,	32 CONTRIBUTOR
4EBC -22,	33 FEATURE CODE
4EBC -25,	34 WLN
! END OF DEFINITION	

[a] Comment line denoted by an exclamation mark in the first place. [b] Binary data packing conversion (IBM/DEC). [c] Trailing comment. [d] Data type and element size in bytes of the field (character string: EBC/ACS, Binary: I/R/L/B for integer, real, logical and byte). [e] Field length (number of elements in the field). [f] ID number of a data field. [g] Data field description of a source field. [h] Number of bytes designating the word alignment bound of this field. [i] Negative value of field length means variable length data field and its absolute value points to another field in which the element specifies the actual field length.

Table V. Record Description File (RDF): MASSCMPR.RDF (Destination RDF in the TDF Shown)

! DEFINITION FILE FOR COMPRESSED MASS SPECTROMETRIC DATA BASE	
DEC	DISK FILE ON MICRO-VAX
IO4 1,	1 SPECTRUM NUMBER
IO4 1,	2 CAS NUMBER
IO4 1,	3 EPA ACCESSION NUMBER
ASC 3,	4 COLLECTION LETTERS
ASC 1,	5 OWNER FLAG
IO4 1,	6 COLLECTION NUMBER
ASC 10,	7 INSTRUMENT
ASC 10,	8 INLET TYPE
IO2 1,	9 MOLECULAR WEIGHT
IO2 1,	10 QUALITY INDEX
IO2 1,	11 INLET TEMPERATURE (DEGREE, C)
IO2 1,	12 SOURCE TEMPERATURE (DEGREE, C)
IO2 1,	13 ELECTRON VOLTAGE (eV)
IO2 1,	14 PRESSURE (MANTISSA)
IO2 1,	15 PRESSURE (EXPONENT)
IO2 1,	16 ION ACCELERATING VOLTAGE (kV×100)
IO2 1,	17 TRAP CURRENT (micro A)
IO2 1,	18 COUNT OF PEAKS
IO2 1,	19 COUNT OF MOLECULAR FORMULA (BYTES)
IO2 1,	20 LENGTH OF NOMENCLATURE (BYTES)
IO4 -18,	21 MASS/INTENSITY PAIRS
ASC -19,	22 MOLECULAR FORMULA
ASC -20,	23 NOMENCLATURE
! END OF DEFINITION	

SUBSIZE: Size of the source array. A 4-byte integer.

DES_BUF: Array to accommodate the destination record. A variable-length byte array.

DBSIZE: Size of the destination array.

CURRENT: Effective size of the destination record in bytes.

(5) DASP_GET (CHANNEL, MODE, KEY_RECID, KEY_VALUE, IKEY_VALUE, BUF, BSIZE, LENGTH, IOS).

Function. ReadΔ a logical record and projectΔ it to a virtual record according to the channel's control data.

Arguments.

CHANNEL: Channel ID number. A 4-byte integer.

MODE: Access mode (S for sequential access; D for direct access; K for keyed access; and SL, DL, or KL for sequential, direct, or keyed access with automatic record lock if the channel is initialized for read/write operations). A string of one or two characters.

KEY_RECID: Key ID of an index file to be accessed or serial number of the logical record to be accessed for direct access. A 4-byte integer.

KEY_VALUE: Value of a character key for keyed read of an index file record. A string of maximum 255 characters.

IKEY_VALUE: Value of an integer key for keyed read of an index file record.

BUF: Array to accommodate the virtual destination record. A variable-length byte array of size specified by BUFSIZE.

SIZE: Size of array named BUF. A 4-byte integer.

LENGTH: Actual size of the transformed destination record. A 4-byte integer.

Table VI. Transformation Description File (TDF)

! TRANSFORMATION TABLE FROM MASS1.DAT TO MASSCMPR.DAT[a]
MASS1.RDF[b]
MASSCMPR.RDF[c]
! DEFINITION FILE FOR COMPRESSED MASS SPECTROMETRIC DATA BASE

3,RV[d,e]	1[f] SPECTRUM NUMBER[g]
4,RV	2 CAS NUMBER
5,RV	3 EPA ACCESSION NUMBER
6,CT	4 COLLECTION LETTERS
7,CT	5 OWNER FLAG
8,RV	6 COLLECTION NUMBER
9,CT	7 INSTRUMENT
10,CT	8 INLET TYPE
11,RV	9 MOLECULAR WEIGHT
12,RV	10 QUALITY INDEX
13,RV	11 INLET TEMPERATURE (DEGREE, C)
14,RV	12 SOURCE TEMPERATURE (DEGREE, C)
15,RV	13 ELECTRON VOLTAGE (eV)
16,RV	14 PRESSURE (MANTISSA)
17,RV	15 PRESSURE (EXPONENT)
18,RV	16 ION ACCELERATING VOLTAGE (kV×100)
23,RV	18 COUNT OF PEAKS
24,RV	19 COUNT OF MOLECULAR FORMULA (BYTES)
26,RV	20 LENGTH OF NOMENCLATURE (BYTES)
29,RV	21 MASS/INTENSITY PAIRS
30,CT	22 MOLECULAR FORMULA
31,CT	23 NOMENCLATURE

! END OF DEFINITION

[a]Comment is denoted by an exclamation mark in the first position. [b]Name of the source record definition file (RDF). [c]Name of the destination record definition file (RDF). [d]ID number of a field in source record, which defines a source-definition correspondence for the record's transformation. [e]Type of data transformation to be performed for the field (RV, reverse the sequence of bytes in binary data repacking; CT, character string code translation; NT, data field to be transferred without any code or sequence changes). [f]ID number of a field in destination record. [g]Data field description of a destination field.

IOS: File reading status provided by Digital RMS. A 4-byte integer.

(6) DASP-PUT (CHANNEL, MODE, KEY-RECID, KEY-VALUE, IKEY-VALUE, BUF, BSIZE, LENGTH, IOS).

Function. Projecting and writing (rewriting) a virtual record to a physical data file according to the control data of the channel.

Arguments.

CHANNEL: Channel ID number. A 4-byte integer.

MODE: Access mode (S for writing a new record of a sequential file; K for writing a new index file record; D for writing or rewriting a sequential file by direct access; R for rewriting a record of an index file; and SL, DL, KL, RL for writing with automatic lock of accessed record until access of another record through the same channel). A series of one or two characters.

KEY_RECID: Index key ID for keyed write of an index file or serial number of a logical record for direct write. A 4-byte integer.

DKEY_VALUE: Value of a character key for keyed rewrite of an index file record.

IKEY_VALUE: Value of an integer key for keyed rewrite of an index file record.

BUF: Array to accommodate the virtual source record. A variable-length byte array of size defined by BUFSIZE.

BUFSIZE: Size of array named BUF. A 4-byte integer.

LENGTH: Size of the source virtual record to be written or rewritten.

IOS: File writing sections provided by Digital RMS. A 4-byte integer.

(7) DASP-CLOSE (CHANNEL).

Function. Reset of a initialized channel and removing its defined control tables.

Arguments.

CHANNEL: Channel ID number. A 4-byte integer.

REFERENCES AND NOTES

(1) Milne, G. W. A.; Heller, S. R. "NIH/EPA Chemical Information System". *J. Chem. Inf. Comput. Sci.* **1980**, *20*, 204–211.

(2) Dalrymple, D. L.; Wilkins, C. L.; Milne, G. W. A.; Heller, S. R. "A Carbon-13 Nuclear Magnetic Resonance Spectra Data Base and Search Systems". *Org. Magn. Reson.* **1978**, *11*, 535–540.

(3) Dubois, J. E.; Bonnet, J. C. "The DARC Pluridata System: The ^{13}C NMR Data Bank". *Anal. Chim. Acta* **1979**, *112*, 245–252.

(4) Koptyung, V. A.; Bochkarev, V. S.; Derendyaev, B. G.; Nekoroshev, S. A.; Piottnkh-Peletskii, V. N.; Podgornaya, M. I.; Ulŷanov, G. P. "Use of Computer for the Solution of Structural Problems in Organic Chemistry by the Methods of Molecular Spectroscopy". *Zh. Strukt. Khim.* **1977**, *18*, 440–459.

(5) Marson, S. "Evolution of Graphic Software for Chemical Information Management". *Abstracts of Papers*, 187th National Meeting of the American Chemical Society, St. Louis, MO; American Chemical Society: Washington, DC, 1984; CHINF 42.

(6) Fry, J. P.; Sibley, E. H. "Evolution of Data Base Management Systems". *ACM Comput. Surv.* **1986**, *8*, No. 1.

(7) Morffew, A. J.; Todd, S. J. P.; Sudgrove, M. J. "The Use of a Relational Data Base for Holding Molecular Data in a Molecular Graphic System". *Comput. Chem.* **1983**, *7(1)*, 9–16.

(8) Wilkins, C. L. "Linked Gas Chromatography–Infrared Mass Spectrometry". *Anal. Chem.* **1987**, *59*, 571A–581A.

(9) A program listing can be obtained by writing to Cheng Qian at Chemical Abstracts Service, Columbus, OH.