27
9-21-77
WNT IS
20

MASTER

UCID-17417

Lawrence Livermore Laboratory

A DATA BASE MANAGEMENT SYSTEM (ADBMS) USERS GUIDE

Version 2.5

Edward W. Birss
Data Management Research Project

February 1977

This is an informal report intended primarily for internal or limited external distribution. The opinions and conclusions stated are those of the author and may or may not be those of the laboratory.

Prepared for U.S. Energy Research & Development Administration under contract No. W-7405-Eng-48.

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency Thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

DISCLAIMER

Portions of this document may be illegible in electronic image products. Images are produced from the best available original document.

This is an informal report intended primarily for internal or limited external distribution. The opinions and conclusions stated are those of the author and may or may not be those of the laboratory.

LAWRENCE LIVERMORE LABORATORY
University of California / Livermore, California

ADBMS USERS GUIDE

Edward W. Birss

February 10, 1977

NOTICE

THIS report was prepared as an account of work sponsored by the United States Government. Neither the United States nor the United States Energy Research and Development Administration, nor any of their employees, nor any of their contractors, subcontractors, or their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness or usefulness of any information, apparatus, product or process disclosed, or represents that its use would not infringe privately owned rights.

Prepared for U.S. Energy Research & Development Administration under Contract
No. W-7405-Eng-48

FOREWORD

The data base management system described here has been designed from the specifications in the CODASYL Data Base Task Group report but with several important modifications, most of which are restrictions, but some of which are extensions.

The basic design consideration of ADBMS was to produce a data base management system for a Problem Statement Analyzer developed by the ISDOS project at the University of Michigan. It has since been modified by the Data Translation Project at the University of Michigan for internal DBMS support for their Data Translator. Although developed for internal use, it can be used independently as a data base management system. It is attractive because of its small memory requirements, its ability to use FORTRAN or COBOL as the host language, and its ability to define and use network structures.

This manual documents the version of ADBMS which was received from the Defense Communication Agency as part of the Data Translation software which was developed under government contract.

To get copies of this report:

ELF RSL .075238:ADBMS:LLL ADBMSOUT

To get copies of programs:

ELF RSL .075238:ADBMS:LLL DDLE DBINE DBLIB DBBLK-O

As requested in "CODASYL COBOL - Data Base Facility Proposal" the following is included here:

This proposal has been submitted to the CODASYL Programming Language Committee for consideration and ultimate approval as the COBOL Data Base Facility. It is based on the concepts and facilities outlined in the CODASYL DBTG April 1971 Report.

The Programming Language Committee will defer action in this proposal until after July 15, 1973, to permit comments, suggestions, and criticisms from the data processing community to be received and evaluated. Such inputs are urgently requested and should be sent to the address above (CODASYL, Box 124, Monroeville, Pennsylvania 15146).

The reader is reminded that this document neither reflects COBOL language specifications nor any implied support by members of the Programming Language Committee or their sponsors, but merely is being offered for possible consideration to the Programming Language Committee. However, the information has been made available to establish better communication between the Programming Language Committee activities and that segment of the public interested in the COBOL language development.

Anyone reproducing this release is requested to include this introduction where meaningful. He is also requested to make reference when pertinent to the introduction on each page of the release.

M. L. O'Connell, Chairman
Data Base Language Task Group

CONTENTS

	Page
1. Background and Terminology	1
Review of DBTG Terms	2
Schema.	2
2. Description of the Schema	7
Example	8
Format of the Schema Description	9
Schema Description Rules	11
3. Database Control System Subroutines	12
Initiation	12
Example	12
Creation and Addition	12
Example	13
Retrieval	13
Example	13
Example	14
Update and Delete	14
Currency	14
Example	14
Test	15
Miscellaneous	15
Termination	15
Alphabetical Listing of User Callable ADBMS Routines	16
4. Helpful Hints	21
4.1 Writing Your Program	21
5. Example	22
References	32
Appendix A	
Detailed DBCS Subroutine Descriptions	33
AMS -- Add Member to Set based on current record	34
AMSK -- Add a Member to a Set based on Key	35
AMSM -- Add a Member to a Set based on current Member.	36
AMSO -- Add a Member to a Set based on current Owner.	37
CARD -- Get the Cardinality of a Set	38
CLOS -- Close a data base	39
CMK -- Check Membership based on Key	40
CMM -- Check Membership based on Member	41
CMO -- Check Membership based on Owner	42
CMR -- Check Membership based on Record	43
CMT -- Check the current Member Type	44
COK -- Check Ownership based on Key	45
COM -- Check Ownership based on Member	46
COO -- Check Ownership based on Owner	47
COR -- Check Ownership based on Record	48
COT -- Check the current Owner Type	49
CR -- Create a Record	50
CRS -- Create a Record and Store data	51
DBST -- Get Statistics About Data Base System	52
DELS -- Delete an entire Set	53
DEND -- Close all data bases and the data base system	54
DINT -- Data Base System initialization	55
DR -- Delete a Record	56
DRK -- Delete a Record based on Key	57

DRM -- Delete the Current Member of a Set	58
DRO -- Delete the Current Owner of a Set	59
FFM -- Find First Member of a set	60
FLM -- Find Last Member of a set	61
FMSK -- Find a Member of a Set based on Sort Key	62
FNM -- Find Next Member of a set	63
FNSK -- Find Next Member of a Sorted Set Based on Sort Key	64
FPM -- Find Previous Member of a set	65
GETCDB -- Get the Current Data Base's identifier	66
GETK -- GET data from record based on Key	67
GETM -- GET data from current Member	68
GETO -- GET data from current Owner	69
GETR -- GET data from current Record	70
GFK -- Get Field based on Key	71
GFM -- Get Field from current Member of a set	72
GFO -- Get Field from current Owner of a set	73
GFR -- Get Field from Record	74
GKM -- Get Key of current Member of a set	75
GKO -- Get Key of current Owner of a set	76
GKR -- Get Key of current Record	77
GTK -- Get record Type based on Key	78
GTM -- Get record Type of current Member	79
GTO -- Get record Type of current Owner	80
ICFK -- Check Field Based on Key	81
ICFM -- Check Field Based on Current Member	82
ICFO -- Check Field Based on Current Owner	83
ICFR -- Check Field Based on Current Record	84
IDOP -- Open a Data base	85
RM -- Remove the current Member from a set	86
RS -- Remove the entire Set	87
SETCDB -- SET a data base to the Current Data Base	88
SETK -- SET the data of a record based on Key	89
SETM -- SET the data of a record based on current Member	90
SETO -- Set the data of a record based on current Owner	91
SETR -- SET the data of a Record based on current Record	92
SFK -- Set Field in record based on Key	93
SFM -- Set Field in the current Member of a set	94
SFO -- Set Field in current Owner of a set	95
SFR -- Set Field in the current Record	96
SKFM -- Set owner based on Key and Find Member	97
SM -- Set the current Member of a set based on the current record ...	98
SMFM -- Set owner based on current Member, and Find Member based on sort key	99
SMK -- Set the current Member of a set based on Key	100
SMM -- Set the current Member of a set based on the current Member ..	101
SMO -- Set the current Member of a set based on current Owner	102
SO -- Set the current Owner of a set based on a current record	103
SOFM -- Set Owner based on current Owner and Find Member based on sort key	104
SOK -- Set the current Owner of a set based on Key	105
SOM -- Set the current Owner of a set based on the current Member of a set	106
S00 -- Set the current based on the current Owner of a set	107
SRFM -- Set owner based on current Record and Find Member based on sort key	108
SRK -- Set current Record based on Key	109

SRM -- Set current Record based on current Member of a set	110
SRO -- Set current Record based on current Owner of a set	111
USE -- Set up error recovery procedure	112
3.3 Errors	114
3.4 Summary of Currency Indicators	115
3.5 Summary of Errors	117
Appendix B	
String Manipulation Subroutine Descriptions	119
Subroutine BLNK	120
Subroutine CSTOI	121
Integer Function ICNC1	122
Integer Function ICSC1	123
Integer Function ISCOMP	124
Subroutine ITOC	125
Subroutine ITOCS	126
Subroutine SMOVE	127
Index	128

ADBMS USERS GUIDE

1. BACKGROUND AND TERMINOLOGY

ADBMS is an implementation of a subset of the Codasyl Data Base Task Group (DBTG) Specifications. It was developed to provide data storage and manipulation capabilities to FORTRAN programs and to maintain program portability. The characteristics of the CODASYL subset complement FORTRAN since dynamic data structures can easily be handled by ADBMS.

ADBMS manages data stored in data bases. Data bases are similar to files, in that they reside on disk, and are composed of records. Unlike sequential files, however, where a record has only one logically "next" record, an ADBMS data base has potentially several "next" records. This capability is achieved by defining relationships between records. A program can utilize the different relationships to find or navigate to the "next" record.

In addition to managing records, ADBMS manages items, or fields within the record. The items are defined and referred to by name. If a user program only needs one item from a record, the program merely requests the item; the entire record is not presented to the user.

To use the features of ADBMS, the FORTRAN application program issues calls to the data base manipulation routines. For many applications these CALLs replace FORTRAN READ and WRITE statements. Usually though, the data manipulation routines do significantly more sophisticated processing than an input/output operation. They provide facilities for interrogation, modification, deletion, and addition of information to the data base. Factoring this data manipulation capability to the data base system tends to decrease the programming effort for certain applications.

This factoring of data management to ADBMS also results in a greater degree of data independence; that is, fewer changes (ideally no changes) have to be made to user programs if access or storage methods are changed, or if the logical structure of the data is modified. For example, because a user program does not receive a whole record, but just the requested items, if an item is added to a record, the buffer space in old programs does not have to be enlarged to accommodate the larger record. Hence, user programs are less dependent on the data stored in the data base.

To use ADBMS, the user must describe the structure of the data that will be stored in the data base. The structure of the data is described in a Data Definition Language (DDL). After the DDL is written and analyzed, the data base must be formatted by running a small formatting program. Finally, the user compiles his or her program, and loads it with a library of data base routines. The user programs, through calls to ADBMS, create and add records to the data base, and retrieve, delete, and modify records stored in the data base.

Review of DBTG Terms

To understand ADBMS, one should understand the CODASYL DBTG terminology. The March 1976 issue of ACM Computing Surveys [1] contains an excellent DBTG tutorial. For completeness, however, a review of the terminology is presented below.

ADBMS is a library of FORTRAN callable subroutines which provide data-structuring and data-manipulating operations. These subroutines are the run-time data base support or the Data Base Control System (DBCS). The DBCS is called by a user application program through a CALL statement. The DBCS manipulates data stored in a data base, and the logical structure of the stored data is described in a

Schema.

The description is fixed before any application programs are executed, and the application program cannot modify the schema.

The schema, the logical structure of the data, consists of item types, record types, and Set types. An item type is the smallest unit of information handled by the data base management system. Although items are logical quantities, they are usually things that can be stored as integers, real numbers, or character strings. Record types are collections of item types, and Set types are relationships between record types.

For example, if we wish to store data about the presidents of the United States, we could define a record type president containing the item types last name, first name, middle initial, and date of birth. To include the occupations of the presidents in the data base, we could define a record type occupation which contains the item type job. A set type can then be defined between the record type president and the record type occupation. We will call this set has-occupations (figure 1).

Figure 1: A Schema

Figure 1 shows an example of a Bachman diagram. It provides a short-hand notation for visualizing data base structures. In these diagrams, records are represented by boxes, and sets by arrows.

To manipulate data, a program creates record instances or occurrences. A president record type filled with information about president George Washington becomes a president record instance. Since Washington was a soldier, a surveyor, and a farmer, three occupation record instances would be related to Washington's president record instance with a has-occupation set instance (figure 2).

Figure 2: A Set Instance

The relationship between the president and occupation record types has some additional terminology associated with it. Since the president record type can be said to "own" the has-occupation set type, the president record-type is called the Owner record type of set type has-occupation. Similarly, the occupation record type is called a Member record type for the has-occupation set

type. In the Bachman diagrams (figure 1) the arrows representing sets always point from owner to member.

Notice that in figure 2 a set instance is depicted. It clearly shows that a set represents a one-to-many relationship between one owner record instance and 0 or more member record instances (i.e. one president has many occupations).

In ADBMS one or more record types may be owners of a given set type, and one or more record types may be members of a given set type. A record type can even be both an owner and a member of the same set type. However, by far the most common usage is a set that has one owner record type and one member record type. The member of that set type can in turn be defined as an owner of another set type; thereby establishing a hierarchy (figure 3). That member record could also be defined as a member of another set, to establish a network or graph structure (figure 4). However, a member record instance may not participate as a member instance more than once in any instance of a given set type. It may participate as a member instance in several different set types but never more than once in any instance of one set type.

Figure 3: A Hierarchical Structure

Figure 4: A Network or Graph Structure

Every set type has an order defined for its member record occurrences, so a user program, in retrieving record instances, receives them in a specific order. ADBMS offers six different set orders: FIFO, LIFO, NEXT, PRIOR, IMMAT, and SORTED. FIFO (first in first out) means new member record instances are inserted at the logical end of the set, so on retrieval, the program receives the first stored record instances first. LIFO (last in first out) stores new record instances at the logical beginning of the set instance. NEXT stores the records following the current member of the set (see currency discussion below), PRIOR stores the record before the current member record, IMMATerial stores member records in the easiest manner for the DBCS, and SORTED maintains member records in sorted order based on one item within the member record type.

ADBMS maintains three types of currency indicators: current record of each record type, current member of each set, and current owner of each set. These types of "current records" are used by the DBCS to identify records.

For example, to find the next member of a set, the DBCS must know the current owner of the set, and the current member of the set. The current owner identifies the set instance, and the current member defines the reference point from which the logically next record may be found.

The currency indicators can be thought of as markers which keep track of the navigation through the network.

Every record in the data base is assigned a unique identifier by the DBCS when the record instance is created. This identifier is called a data base key. Data base keys can be used to immediately find a particular record; no searching through set is necessary.

The DBCS maintains the data base keys which remain invariant throughout the lifetime of the data base. A user application program may inspect data base keys, but may not modify them, and only equality or nonequality comparisons on data base keys are meaningful.

Although a user application program could store data base keys for use on a subsequent run, such behavior is dangerous if an unload and reload of the data base occurs between runs, because an unload/reload usually reassigns data base keys. It is wise to assume data base keys remain constant only throughout a single execution of a program.

2. DESCRIPTION OF THE SCHEMA

The schema may be considered to be a template for the logical structure of the data base, since it describes "building blocks" and legal connections among record and set types. A user program references the set types, record types, and item types defined in the schema, and passes them as parameters to DBCS subroutines.

The user specifies the schema in the Data Definition Language (DDL). The DDL is analyzed by a DDL analyzer, and an encoded version of the schema, the Data Base Tables (DBT), are output. It is the DBT that the DBCS uses in the course of processing data requests. In version 2.5 of ADBMS, up to five data bases, each with their own DBT, can be concurrently manipulated.

To describe the schema in the DDL, the user must specify the items, records, and sets in the data base:

The item description consists of named atomic quantities of information. Items are of two basic types, single items, and repeating items. A repeating item can have a fixed number of repetitions (e.g. an array of length five), or a variable number of repetitions (like the COBOL depending-on option). Variable repeating items must have a depending-on item, an item in the same record which contains the current number of instances of the repeating item.

A record description consists of naming the record and defining the items contained within it. Record type names must be unique within the data base, but item type names need only be unique within a record type. The DBCS maintains a special record type named SYSTEM. There is a single instance of the SYSTEM record and it serves as an entry point into the data base. This record is always current, and it is an owner record type for all singular sets in the data base (singular sets are sets that connect the SYSTEM record with a user's logically topmost records). The SYSTEM record should not be manipulated by the user.

The set description consists of naming the legal owner and the legal members and defining the set order. To define a set, the Set description record names the set and defines the set order. The owner description records name the legal owner record types, and the member description records name the legal member record types.

The DDL is also used to specify the number of pages in the data base. A page is the unit of physical storage (1024 words by default) used by ADBMS.

Example

RECORD	PRES		
ITEM	LASTN	CHAR	10
ITEM	FIRSTN	CHAR	10
ITEM	INIT	CHAR	10
ITEM	DOB	CHAR	10
RECORD	OCCUP		
ITEM	JOB	CHAR	10
SET	HASOCC	FIFO	
OWNER	PRES		
MEMBER	OCCUP		
SET	SPRES	FIFO	
OWNER	SYSTEM		
MEMBER	PRES		
NPAGES			20

Format of the Schema Description

RECORD description:

columns 1 - 6 RECORD
columns 8 - 13 record type name

ITEM description:

columns 1 - 4 ITEM
columns 8 - 13 item name
columns 15 - 20 item type

legal types:

INTEG - integer
REAL - floating point
BINARY - bit string
DBKEY - data base key
LOGIC - logical variable
CHAR - character string

columns 22 - 24 item size

legal size:

type	max.size	units
INTEG	60	bits
REAL	60	bits
BINARY	256	bytes
DBKEY	-	always fullword
LOGIC	-	always fullword
CHAR	256	bytes

columns 26 - 31 depended-on item (optional)

columns 33 - 35 max replication factor

SET description:

columns 1 - 3 SET
columns 8 - 13 set type name
columns 15 - 20 order

legal orders:

FIFO - first in first out
LIFO - last in first out
NEXT - after current record
PRIOR - before current record
IMMAT - don't care
SORTED - ascending order by key

columns 26 - 31 sort key (if order = SORTED)

OWNER description:

columns 1 - 5 OWNER
columns 8 - 13 owner record type name

MEMBER description:

columns 1 - 6 MEMBER

columns 8 - 13 member record type name

NPAGES description:

columns 1 - 6 NPAGES

columns 22 - 24 data base size (in pages - legal range is between
1 and 99999).

Schema Description Rules

Items must be defined immediately after the record description in which they belong.

The order of item descriptions defines the order in which they are stored by the DBCS.

Specifying a replication field in an item description defines a repeating item.

A repeating item must be the last item in the record.

The depending-on item field is used to specify the item in the record which is to contain the number of occurrences of the repeating item.

The depending-on item must be an integer item, and must be defined before the repeating item is defined.

The maximum replication field must be specified if a depending-on item is used.

There must be a minimum of one owner description and one member description for each set description.

All the owner descriptions must follow the set descriptions, and the member descriptions must follow the last owner description.

A record must be defined as a record prior to being referenced in an owner or member description.

Sort keys must be items within the member record types, and if more than one member type is defined, then the sort keys must have the same type, displacement, and size in every member record type.

A sort key item must be integer or character.

If SYSTEM is defined as an owner type, it must be the sole owner. No other owner record may be defined for that set.

3. DATABASE CONTROL SYSTEM SUBROUTINES

Data manipulation in ADBMS is accomplished by calls to DBCS routines. Most of these routines are names using a convention: the first two to three letters indicate the routine's function, while the last letter designates the criteria used (which currency indicator) to identify the record instance. For example, there are four routines available to get an item's value from a record: GFK, GFM, GFO, and GFR. They stand for Get Field from a record based on Key, current Member, current Owner, and current Record respectively.

The data manipulation routines are broken down into several categories: initiation, creation and addition, retrieval, updates and delete, currency, test, miscellaneous, and termination. Each of these categories is described below.

Initiation

To begin to process an ADBMS data base, one must first initialize the data base system using a call to DINT, and then open the data base(s) using call(s) to IDOP.

Example

```
CALL DINT(0, 59, IERR)
I=IDOP(2, 3, 1, IERR)
```

Creation and Addition

Once the data base has been opened, a typical task is to fill it with data. This is done by a consortium of three basic groups of routines: CR (create), SET (set), and AMS (add member to set). The CR group consists of CR (create record instance) and CRS (create record instance and store data). The CRS routine requires the user to know the exact format of the data, and therefore it should not be used by novice users.

After CR has created a record instance, the user must set the data values within the record. This is accomplished by several SET data routines: SETK, SETM, SETO, SETR, SFK, SFM, SFO, and SFR. The SETx routines require the user to know the format of the data into the data base and, hence, are not recommended for general use. The SFx (set field) routines permit naming of items, thus unburdening the user from needing to know the storage format. They provide the best way to set item values.

Finally, when the record has been filled with data, it may be added as a member to a set instance by use of AMS, AMSK, AMSM, and AMSO.

Example

```
CALL CR("PRES ", KEY, IERR)
CALL SFR("LASTN ", "PRES ", "WASHINGTON", IERR)
CALL SFR("FIRSTN", "PRES ", "GEORGE", IERR)
CALL SFR("DOB   ", "PRES ", "12/16/1723", IERR)
CALL AMS("SPRES ", "PRES ", IERR)
```

The call to CR creates a PRES record occurrence, the SFR calls set the data values of items in the PRES record instance, and the AMS call adds the PRES record instance to the singular set SPRES.

Retrieval

Retrieval in ADBMS is accomplished through FIND operations: FFM(find first member), FNM (find next member), FPM (find previous member), FLM (find last member), FMSK (find member based on sort key), and FNSK (find next member based on sort key). Find first member always find the first member of a set. It is usually followed by a loop containing FNM calls. All finds should check for a return code of -1 which indicates the end of a set. The user application program usually does not have to check for positive return codes, because the DBCS traps these errors and writes them out to the error message unit specified in the DINT call.

Finds are usually followed by gets, which get data from the record. Gets come in many flavors: GETK, GETM, GETO, GETR, GFK, GFM, GFO, GFR, GKM, GKO, GKR, GTK, GTM, and GTO. These gets get data, fields, keys, and record types from record instances.

Example

```
c  print all presidents in the data base
   CALL FFM("SPRES ", IERR)
   IF (IERR .EQ. -1) GO TO EMPTYSET
10  CONTINUE
   CALL GFM("LASTN ", "SPRES ", DATA, IERR)
   print presidents name
   CALL FNM("SPRES ", IERR)
   IF (IERR .EQ. -1) GO TO ENDOFSET
   GO TO 10
```


To print all the presidents in the data base, FFM is called to get the first president. GFM gets the president's last name, and then the data retrieved is printed. To get the next record, FNM is called. Next we loop back to GFM to get data. The loop is terminated when a -1 is returned indicating an end-of-set condition.

FMSK is a different type of manipulation. It can only be used with sorted sets. A value is provided and the routine finds the record with that value.

Example

```
C  EXAMPLE ASSUMES SET SPRES IS SORTED ON LASTN
    CALL FMSK("SPRES ", "WASHINGTON", IERR)
    IF(IERR .EQ. -1) GO TO NOTFOUND
```

Update and Delete

To update and delete records, ADBMS provides SET, DR (delete record), DELS (delete all members of a set), RM (remove a record), and RS (remove all members of a set). The SET commands have been discussed previously. DR comes in the four standard flavors, DR, DRK, DRM, and DRO, which delete a record based on currency indicators. The distinction between remove and delete is that remove unhooks a record from a set but does not destroy the record. Delete removes the record from the set and destroys the record.

Currency

In order to navigate through the data base, the user program must manipulate currency indicators. To do this, ADBMS provides a different flavor of set commands: Set current member (SM, SMR, SMM, SMO), set current owner (SO, SOK, SOM, SOO), and set record (SRK, SRM, SRO).

Example

```
C  FIND ALL OCCUPATION OF ALL PRESIDENTS
    CALL FFM("SPRES ", IERR)
5  CONTINUE
    CALL SOM("HASOCC", "SPRES ", IERR)
10 CONTINUE
    CALL GFM(...)
    print occupation
    CALL FNM("HASOCC", IERR)
```

```

      IF (IERR .EQ. -1) GO TO 15
      GO TO 10
15  CONTINUE
      CALL FNM("SPRES ", IERR)
      IF (IERR .EQ. -1) GO TO END OF ALL OCCUPATIONS
      GO TO 5

```

This example is similar to the "print all presidents" example, except we must drop down a level in the hierarchy. To do this, a call to SOM is employed. SOM sets the current owner of set HASOCC equal to the current member of set SPRES. This then identifies the HASOCC set occurrence properly so the FFM (and subsequent FNM's) on set HASOCC are defined.

Test

Test routines check fields within record (without necessitating a get and an if statement): ICFK, ICFM, ICFO, and ICFR. Checks are also provided to see if a record is a member or an owner of a set: CMK, CMM, CMO, CMR, COK, COM, COO, COR, COT.

Miscellaneous

Several miscellaneous routines are: CARD, get the cardinality (number of members) in a set, GETCDB, get currency indicator for the data base, SETCDB, set currency indicator for the data base, DBST, get data base statistics, SKFM, SOFM, and SRFM which are combinations of other routines, and USE which currently is not implemented.

Termination

Two routines are provided for terminations: CLOS which closes one data base, and DEND which closes all data bases and the data base system.

The following pages provide a brief listing of the subroutines available in the DBCS. A more complete description of each subroutine is found in the appendix A.

Alphabetical Listing of User Callable ADBMS Routines

Subroutine AMS(set-type, record-type, IERR)
Add Member to Set based on current record

Subroutine AMSK(set-type, db-key, IERR)
Add a Member to a Set based on Key

Subroutine AMSM(set-type1, set-type2, IERR)
Add a Member to a Set based on current Member

Subroutine AMSO(set-type1, set-type2, IERR)
Add a Member to a Set based on current Owner

Subroutine CARD(set-type, data, IERR)
Get the CARDinality of a set

Subroutine CLOS(IERR)
CLOSE the current data base

Subroutine CMK(set-type, db-key, sw, IERR)
Check Membership based on Key

Subroutine CMM(set-type1, set-type2, sw, IERR)
Check Membership based on current Member

Subroutine CMO(set-type1, set-type2, sw, IERR)
Check Membership based on current Owner

Subroutine CMR(set-type, record-type, sw, IERR)
Check Membership based on current Record

Subroutine CMT(set-type, record-type, sw, IERR)
Check the current Member Type

Subroutine COK(set-type, db-key, sw, IERR)
Check Ownership based on Key

Subroutine COM(set-type1, set-type2, sw, IERR)
Check Ownership based on current Member

Subroutine COO(set-type1, set-type2, sw, IERR)
Check Ownership based on current Owner

Subroutine COR(set-type, record-type, sw, IERR)
Check Ownership based on current record of Record type

Subroutine COT(set-type, record-type, sw, IERR)
Check the current Owner Type

Subroutine CR(record-type, key, IERR)
Create a Record occurrence

Subroutine CRS(record-type, data, key, IERR)

Create a Record occurrence and Store data

Subroutine DBST(sw, array, IERR)
Get Data Base Statistics

Subroutine DELS(set-type, IERR)
DElete an entire Set

Subroutine DEND(IERR)
Close all data bases and the data base system

Subroutine DINT(nbuf, lio, IERR)
Initialize the Data base system

Subroutine DR(record-type, IERR)
Delete a Record Occurrence

Subroutine DRK(db-key, IERR)
Delete a Record Occurrence based on Key

Subroutine DRM(set-type, IERR)
Delete a Record which is the current Member of a set

Subroutine DRO(set-type, IERR)
Delete Record which is the current Owner of a set

Subroutine FFM(set-type, IERR)
Find First Member of a set

Subroutine FLM(set-type, IERR)
Find Last Member of a set

Subroutine FMSK(set-type, key-value, IERR)
Find a Member of a set based on Sort Key

Subroutine FNM(set-type, IERR)
Find Next Member of a set

Subroutine FNSK(set-type, sort-key, IERR)
Find Next member of a sorted set based on Sort Key

Subroutine FPM(set-type, IERR)
Find the Previous Member of a set

Subroutine GETCDB(dbid, ierr)
Get the id of the current data base

Subroutine GETK(db-key, data, IERR)
GET data from record based on Key

Subroutine GETM(db-key, data, IERR)
GET data from current Member

Subroutine GETO(set-type data, IERR)
GET data from current Owner

Subroutine GETR(record-type, data, IERR)

GET data from current Record

Subroutine GFK(item-type, db-key, data, IERR)
Get Field from a record based on Key

Subroutine GFM(item-type, set-type, data, IERR)
Get Field from current Member of a set

Subroutine GFO(item-type, set-type, data, IERR)
Get Field from current Owner of a set

Subroutine GFR(item-type, record-type, data, IERR)
Get Field from current Record

Subroutine GKM(set-type, db-key, IERR)
Get Key of current Member of a set

Subroutine GKO(set-type, db-key, IERR)
Get Key of current Owner of a set

Subroutine GKR(record-type, db-key, IERR)
Get Key of current Record

Subroutine GTK(db-key, type, IERR)
Get record Type based on Key

Subroutine GTM(set-type, type, IERR)
Get record Type of current Member of a set

Subroutine GTO(set-type, type, IERR)
Get record Type of current Owner of a set

Integer Function ICFK(item-type, db-key, data, IERR)
Check Field based on Key

Integer Function ICFM(item-type, set-type, data, IERR)
Check Field Based on current Member

Integer Function ICFO(item-type, set-type, data, IERR)
Check Field Based on current Owner

Integer Function ICFR(item-type, record-type, data, IERR)
Check Field based on current Record

Subroutine IDOP(liodbf, liodbt, use, IERR)
Open a data base

Subroutine RM(set-type, IERR)
Remove the current Member from a set

Subroutine RS(set-type, IERR)
Remove the entire Set

Subroutine SETCDB(IERR)
Set the current data base

Subroutine SETK(db-key, data, IERR)

SET data in record based on Key

Subroutine SETM(set-type, data, IERR)

SET data in record based on current Member

Subroutine SETO(set-type, data, IERR)

SET data in record based on current Owner

Subroutine SETR(record-type, data, IERR)

SET data in record based on current Record

Subroutine SFK(item-type, db-key, data, IERR)

Set Field in record based on Key

Subroutine SFM(item-type, set-type, data, IERR)

Set Field in the current Member of a set

Subroutine SFO(item-type, set-type, data, IERR)

Set Field in the current Owner of a set

Subroutine SFR(item-type, record-type, data, IERR)

Set Field in the current record of Record type

Subroutine SKFM(set-type, db-key, data, IERR)

Set owner based on Key and Find Member based on sort key

Subroutine SM(set-type, record-type, IERR)

Set the current Member of a set based on the current record

Subroutine SMFM(set-type1, set-type2, data, IERR)

Set owner based on current Member; Find Member based on sort key

Subroutine SMK(set-type, db-key, IERR)

Set the current Member of a set based on Key

Subroutine SMM(set-type1, set-type2, IERR)

Set current Member of a set based on current Member of a set

Subroutine SMO(set-type1, set-type2, IERR)

Set current Member of a set based on current Owner of a set

Subroutine SO(set-type, record-type, IERR)

Set the current Owner of a set based on the current record

Subroutine SOFM(set-type1, set-type2, data, IERR)

Set owner based on current Owner; Find Member based on sort key

Subroutine SOK(set-type, db-key, IERR)

Set the current Owner of a set based on Key

Subroutine SOM(set-type1, set-type2, IERR)

Set current Owner of a set based on current Member of a set

Subroutine SOO(set-type1, set-type2, IERR)

Set current Owner of a set based on current Owner of a set

Subroutine SRFM(set-type, record-type, data, IERR)

Set owner based on current Record; Find Member based on sort key

Subroutine SRK(db-key, IERR)

Set current Record based on Key

Subroutine SRM(set-type, IERR)

Set current Record based on current Member of a set

Subroutine SRO(set-type, IERR)

Set current Record based on current Owner of a set

Subroutine USE(error-number, subroutine name, IERR)

Set up error recovery procedure

4. HELPFUL HINTS

To use ADBMS for a particular application, there are seven basic steps that must be followed.

1. Design your Data Structure and write your DDL
2. Run the DDL analyzer
3. Run the Data Base Initializer
4. Write your application program
5. Compile your program
6. Load the DBCS Block Data, your program, and the DBCS Library
7. Execute your program

4.1 Writing Your Program

When writing your program, it is not always necessary to check IERR. IERR is automatically checked by the DBCS prior to returning to your program, and a message is written to the terminal indicating any error other than a minus 1. Note however that the DBCS traps the error, but does not indicate where in your program the bad call occurred.

When passing names of items, records or sets to DBCS routines, the strings must be six characters long, padded with trailing blanks, if necessary.

5. EXAMPLE

A data base containing information about Presidents of the United States and their occupations, marriages, elections, opponents, administrations, and Vice Presidents is presented.

STEP 1: Write The Data Definition Language:

```

NPAGES                                45
RECORD PRES
ITEM  LASTN  CHAR    10
ITEM  FIRSTN CHAR    10
ITEM  INIT   CHAR     1
ITEM  MONTHB CHAR    10
ITEM  DAYB   INTEG   59
ITEM  YEARB  INTEG   59
ITEM  NATSTA CHAR    10
RECORD OCCUP
ITEM  JOB    CHAR    10
RECORD MARRIA
ITEM  SPOUSE CHAR    10
ITEM  MONTHM CHAR    10
ITEM  DAYM   INTEG   59
ITEM  YEARM  INTEG   59
ITEM  NUMCHD INTEG   59
RECORD ELECT
ITEM  EYEAR  INTEG   59
ITEM  VICTOR CHAR    10
ITEM  PARTY  CHAR    10
ITEM  EVOTES INTEG   59
RECORD LOSER
ITEM  LOSERN CHAR    10
ITEM  PARTY  CHAR    10
ITEM  EVOTES INTEG   59
RECORD ADMIN
ITEM  NUMBER INTEG   59
ITEM  MONTHI CHAR    10
ITEM  DAYI   INTEG   59
ITEM  YEARI  INTEG   59
RECORD VPRES
ITEM  VFIRST CHAR    10
ITEM  VLAST  CHAR    10
SET   SPRES  SORTED    LASTN
OWNER SYSTEM
MEMBER PRES
SET   OCCUPS FIFO
OWNER PRES
MEMBER OCCUP
SET   MRGS   FIFO
OWNER PRES
MEMBER MARRIA
SET   ELECTS FIFO

```

```

OWNER  PRES
MEMBER ELECT
SET    LOSERS FIFO
OWNER  ELECT
MEMBER LOSER
SET    ADMINS FIFO
OWNER  PRES
MEMBER ADMIN
SET    VPS      FIFO
OWNER  ADMIN
MEMBER VPRES

```

STEP 2: Run the DDL Analysis Program (DDLE)

```

DDLE / t v
ddl list dbt:      PROGDDL LIST PRESDBT

ALL DONE

```

STEP 3: Run the Data Base Initialization Program (DBINE)

```

DBINE / t v
dbt dbf:  PRESDBT PRESDB
1data base initialized with      45 pages.
ALL DONE

```

STEP 4: Write the load program

The program provided here is a load program. This load program operates off of an input data file -- presdata. The program also uses the Data Base Table file created in step 2, and the Data Base File created in step 3.

To aid the reader in understanding the data base program, below is a listing of the input data file presdata.

```

PRES      ADAMS      JOHN      OCTOBER      30      1735MASS.
OCCUPATIONLAWYER
OCCUPATIONTEACHER
MARRIAGE  ABIGAIL    OCTOBER      25      1764      5
ELECTION  1796J. ADAMS  FEDERALIST  71

```

LOSER	BURR	DEM-REP	30			
LOSER	JEFFERSON	DEM-REP	68			
LOSER	PINCKNEY	FEDERALIST	59			
LOSER	9 OTHERS		48			
ADMIN	A3	MARCH	4	1797	ADAMSJ	
VICE-PRES	THOMAS	JEFFERSON				
PRES	ADAMS	JOHN Q.		JULY	11	1767MASS.
OCCUPATION	LAWYER					
OCCUPATION	SECRETARY					
MARRIAGE	LOUISA	JULY	26	1797	4	
ELECTION	1824	J.Q. ADAMS	DEM-REP	84		
LOSER	CLAY	INDEP.	37			
LOSER	CRAWFORD	INDEP.	41			
LOSER	JACKSON	INDEP.	99			
ADMIN	A10	MARCH	4	1825	ADAMSJQ	
VICE-PRES	JOHN	CALHOUN				
PRES	ARTHUR	CHESTER A.		OCTOBER	5	1830VERMONT
OCCUPATION	LAWYER					
OCCUPATION	TEACHER					
MARRIAGE	ELLEN	OCTOBER	25	1859	3	
ADMIN	A28	SEPTEMBER	20	1881	ARTHUR	
PRES	BUCHANAN	JAMES		APRIL	23	1791PA.
OCCUPATION	LAWYER					
ELECTION	1856	BUCHANAN	DEMOCRATIC	174		
LOSER	FILLMORE	AMERICAN	8			
LOSER	FREMONT	REPUBLICAN	114			
ADMIN	A20	MARCH	4	1857	BUCHANAN	
VICE-PRES	JOHN	BRECKRIDGE				
PRES	CLEVELAND	GROVER		MARCH	18	1837NEW JERSEY
OCCUPATION	CLERK					
OCCUPATION	TEACHER					
MARRIAGE	FRANCES	JUNE	2	1886	5	
ELECTION	1884	CLEVELAND	DEMOCRATIC	219		
LOSER	BLAINE	REPUBLICAN	182			
ELECTION	1892	CLEVELAND	DEMOCRATIC	277		
LOSER	HARRISON	REPUBLICAN	145			
LOSER	WEAVER	PEOPLES	22			
ADMIN	A29	MARCH	4	1885	CLEVELAN	
ADMIN	A31	MARCH	4	1893	CLEVELAN	
VICE-PRES	ADLAI	STEVENSON				
VICE-PRES	THOMAS	HENDRICKS				
PRES	COOLIDGE	CALVIN		JULY	4	1872VERMONT
OCCUPATION	LAWYER					
MARRIAGE	GRACE	OCTOBER	4	1905	2	
ELECTION	1924	COOLIDGE	REPUBLICAN	382		
LOSER	DAVIS	DEMOCRATIC	136			
LOSER	LAFOLLETTE	PROGRESS.	13			
ADMIN	A40	AUGUST	3	1923	COOLIDGE	
ADMIN	A41	MARCH	4	1925	COOLIDGE	
VICE-PRES	CHARLES	DAWES				
PRES	EISENHOWER	DWIGHT D.		OCTOBER	14	1890TEXAS
OCCUPATION	SOLDIER					
MARRIAGE	MAMIE	JULY	1	1916	2	
ELECTION	1952	EISENHOWER	REPUBLICAN	442		

Now a listing of the program...

NOTE: the string manipulation subroutines referenced by this program and used internally by ADBMS are given in appendix B.

```
PROGRAM X(TAPE59,PRESDB,TAPE2=PRESDB,PRESDBT,TAPE3=PRESDBT
1,PRESDATA,TAPE5=PRESDATA)
COMMON/DDL/PRES, LASTN, FIRSTN, INIT, MONTHB, DAYB, YEARB, NATSTA,
2OCCUP, JOB, MARRIA, SPOUSE, MONTHM, DAYM, YEARM, PARTY,
3NUMCHD, ELECT, EYEAR, VICTOR, EVOTES, LOSER, LOSERN, ADMIN,
4NUMBER, MONTHI, DAYI, YEARI, VPRES, VFIRST, VLAST,
5SPRES, OCCUPS, MRGS, LOSERS, ADMINS, VPS, ELECTS

C
C
DATA PRES/"PRES"/, LASTN/"LASTN"/, FIRSTN/"FIRSTN"/, INIT/"INIT"/,
1MONTHB/"MONTHB"/, DAYB/"DAYB"/, YEARB/"YEARB"/, NATSTA/"NATSTA"/,
2HEIGHT/"HEIGHT"/, PARTY/"PARTY"/, UNIVER/"UNIVER"/, HERIT/"HERIT"/
DATA PARTY/"PARTY"/, OCCUP/"OCCUP"/,
5JOB/"JOB"/, MARRIA/"MARRIA"/, SPOUSE/"SPOUSE"/, MONTHM/"MONTHM"/,
7YEARM/"YEARM"/, NUMCHD/"NUMCHD"/, ELECT/"ELECT"/, EYEAR/"EYEAR"/,
8VICTOR/"VICTOR"/, EVOTES/"EVOTES"/, LOSER/"LOSER"/, LOSERN
1/"LOSERN"/, DAYM/"DAYM"/
DATA
&ADMIN/"ADMIN"/, NUMBER/"NUMBER"/, MONTHI/"MONTHI"/, DAYI/"DAYI"/,
&YEARI/"YEARI"/, VPRES/"VPRES"/, VFIRST/"VFIRST"/, VLAST/"VLAST"/,
&SPRES/"SPRES"/, OCCUPS/"OCCUPS"/,
&MRGS/"MRGS"/, LOSERS/"LOSERS"/, ADMINS/"ADMINS"/, PRCO/"PRCO"/,
&VPS/"VPS"/, PC/"PC"/, ELECTS/"ELECTS"/
COMMON/IBUFC/IBUF(9)

C
C PRESIDENTIAL DATABASE LOAD
C INITIALIZE THE DATA BASE SYSTEM WITH A CALL TO DINT
C
CALL DINT(0,59,IERR)
C OPEN THE DATA BASE BY CALLING IDOP
IDD=IDOP(2,3,1,IERR)
C
1 CONTINUE
READ(5,6) IBUF
IF(EOF,5) GO TO 1000
6 FORMAT(9A10)
C WRITE OUT THE FIRST 3 WORDS OF IBUF SO WE KNOW WHERE WE ARE
C
WRITE(59,15)(IBUF(I),I=1,3)
15 FORMAT(1X,3A10)
C CHECK FOR VARIOUS RECORD TYPES
IF(ISCOMP(IBUF,1,"PRES",1,4) .EQ. 0) GO TO 100
IF(ISCOMP(IBUF,1,"OCCUPATION",1,10) .EQ. 0) GO TO 200
IF(ISCOMP(IBUF,1,"MARRIAGE",1,8) .EQ. 0) GO TO 300
IF(ISCOMP(IBUF,1,"ELECTION",1,8) .EQ. 0) GO TO 400
IF(ISCOMP(IBUF,1,"LOSER",1,5) .EQ. 0) GO TO 500
IF(ISCOMP(IBUF,1,"ADMIN",1,5) .EQ. 0) GO TO 600
IF(ISCOMP(IBUF,1,"VICE-PRES",1,9) .EQ. 0) GO TO 700

GO TO 999

C
```

```

C
C   PROCESS THE VARIOUS RECORD TYPES
C
C   100 CONTINUE
C   CREATE PRES RECORD
C       CALL CR(PRES,KEY,IERR)
C   FILL UP THE RECORD
C       CALL PRESFL
C   ADD IT TO SET SPRES
C       CALL AMS(SPRES,PRES,IERR)
C       GO TO 1
C
C   OCCUPATION RECORD
C
C   200 CONTINUE
C       CALL SOM(OCCUPS,SPRES,IERR)
C       CALL CR(OCCUP,KEY,IERR)
C       CALL OCFL
C       CALL AMS(OCCUPS,OCCUP,IERR)
C       GO TO 1
C
C   MARRIAGE RECORD
C
C   300 CONTINUE
C       CALL SOM(MRGS,SPRES,IERR)
C       CALL CR(MARRIA,KEY,IERR)
C       CALL MARRFL
C       CALL AMS(MRGS,MARRIA,IERR)
C       GO TO 1
C
C   ELECTION RECCORD
C
C   400 CONTINUE
C       CALL SOM(ELECTS,SPRES,IERR)
C       CALL CR(ELECT,KEY,IERR)
C       CALL ELECFL
C       CALL AMS(ELECTS,ELECT,IERR)
C       GO TO 1
C
C   LOSER RECORD
C
C   500 CONTINUE
C       CALL SOM(LOSERS,ELECTS,IERR)
C       CALL CR(LOSER,KEY,IERR)
C       CALL LOSFL
C       CALL AMS(LOSERS,LOSER,IERR)
C       GO TO 1
C
C   ADMIN RECORD
C
C   600 CONTINUE
C       CALL SOM(ADMINS,SPRES,IERR)
C       CALL CR(ADMIN,KEY,IERR)
C       CALL ADFL
C       CALL AMS(ADMINS,ADMIN,IERR)
C       GO TO 1

```

```

C
C VICE PRES REC
C
700 CONTINUE
    CALL SOM(VPS,ADMS,IERR)
    CALL CR(VPRES,KEY,IERR)
    CALL VPFL
    CALL AMS(VPS,VPRES,IERR)
    GO TO 1

C
C
C
C ERROR PROCESSING AND TERMINATION
C
998 FORMAT ("ERROR--IDENTIFICATION",/,7A10)
999 WRITE(59,998) (IBUF(I),I=1,7)
    GO TO 1
1000 CONTINUE
    CALL DEND(IERR)
    STOP
    END
    SUBROUTINE PRESFL

C
C
    COMMON/DDL/PRES, LASTN, FIRSTN, INIT, MONTHB, DAYB, YEARB, NATSTA,
    2OCCUP, JOB, MARRIA, SPOUSE, MONTHM, DAYM, YEARM, PARTY,
    3NUMCHD, ELECT, EYEAR, VICTOR, EVOTES, LOSER, LOSERN, ADMIN,
    4NUMBER, MONTHI, DAYI, YEARI, VPRES, VFIRST, VLAST,
    5SPRES, OCCUPS, MRGS, LOSERS, ADMS, VPS, ELECTS

C
    COMMON/IBUFC/IBUF(9)

C
    CALL SMOVE(ITEMP,1,IBUF,11,10)
    CALL SFR(LASTN,PRES,ITEMP,IERR)
    CALL SMOVE(ITEMP,1,IBUF,21,10)
    CALL SFR(FIRSTN,PRES,ITEMP,IERR)
    CALL SMOVE(ITEMP,1,IBUF,31,10)
    CALL SFR(INIT,PRES,ITEMP,IERR)
    CALL SMOVE(ITEMP,1,IBUF,41,10)
    CALL SFR(MONTHB,PRES,ITEMP,IERR)
    CALL CSTOI(IBUF,51,10,INUM)
    CALL SFR(DAYB,PRES,INUM,IERR)
    CALL CSTOI(IBUF,61,10,INUM)
    CALL SFR(YEARB,PRES,INUM,IERR)
    CALL SMOVE(ITEMP,1,IBUF,71,10)
    CALL SFR(NATSTA,PRES,ITEMP,IERR)
    RETURN

C
C
    ENTRY OCFL
    CALL SMOVE(IOBUF,1,IBUF,11,10)
    CALL SFR(JOB,OCCUP,IOBUF,IERR)
    RETURN

C
C
    ENTRY MARRFL
    CALL SMOVE(IOBUF,1,IBUF,11,10)

```



```

CALL SFR(SPOUSE,MARRIA,IOBUF,IERR)
CALL SMOVE(IOBUF,1,IBUF,21,10)
CALL SFR(MONTHM,MARRIA,IOBUF,IERR)
CALL CSTOI( IBUF,31,10,NUMB)
CALL SFR(DAYM,MARRIA,NUMB,IERR)
CALL CSTOI( IBUF,41,10,NUMB)
CALL SFR(YEARM,MARRIA,NUMB,IERR)
CALL CSTOI( IBUF,51,10,NUMB)
CALL SFR(NUMCHD,MARRIA,NUMB,IERR)
RETURN

```

C
C

```

ENTRY ELECFL
CALL CSTOI( IBUF,11,10,NUMB)
CALL SFR(EYEAR,ELECT,NUMB,IERR)
CALL SMOVE(IOBUF,1,IBUF,21,10)
CALL SFR(VICTOR,ELECT,IOBUF,IERR)
CALL SMOVE( IOBUF,1,IBUF,31,10)
CALL SFR(PARTY,ELECT,IOBUF,IERR)
CALL CSTOI( IBUF,41,10,NUMB)
CALL SFR(EVOTES,ELECT,NUMB,IERR)
RETURN

```

C
C

```

ENTRY LOSFL
CALL SMOVE( IOBUF,1,IBUF,11,10)
CALL SFR(LOSERN,LOSER,IOBUF,IERR)
CALL SMOVE( IOBUF,1,IBUF,21,10)
CALL SFR(PARTY,LOSER,IOBUF,IERR)
CALL CSTOI( IBUF,31,10,NUMB)
CALL SFR(EVOTES,LOSER,NUMB,IERR)
RETURN

```

C
C

```

ENTRY ADPL
IE=ICSC1( IBUF,12," ",9)
IE=IE-1
CALL CSTOI( IBUF,12,IE,NUMB)
CALL SFR(NUMBER,ADMIN,NUMB,IERR)
CALL SMOVE( IOBUF,1,IBUF,21,10)
CALL SFR(MONTHI,ADMIN,IOBUF,IERR)
CALL CSTOI( IBUF,31,10,NUMB)
CALL SFR(DAYI,ADMIN,NUMB,IERR)
CALL CSTOI( IBUF,41,10,NUMB)
CALL SFR(YEARI,ADMIN,NUMB,IERR)
RETURN

```

C
C

```

ENTRY VPFL
CALL SMOVE( IOBUF,1,IBUF,11,10)
CALL SFR(VFIRST,VPRES,IOBUF,IERR)
CALL SMOVE( IOBUF,1,IBUF,21,10)
CALL SFR(VLAST,VPRES,IOBUF,IERR)
RETURN

```

C
C

```

END

```

STEP 5: compile and load the program

To compile and load the program, one must compile the program and load it with a block data routine and the data base library in addition to certain system libraries. In some installations, the block data must be loaded first.

```
CHAT (DBBLK-O$ DBLIB, URLIB, ORDERLIB, ↑ ) PROG,PROGE,PROG-O$,,BOX / t v
```

ALL DONE

STEP 6: Execute the program

```
PROGE / t v
```

ALL DONE

STEP 7: Write a Retrieval Program

The following program lists all the presidents and their occupations.

```
PROGRAM X(TAPE59,PRESDB,TAPE2=PRESDB,PRESDBT,TAPE3=PRESDBT
1,PRESDATA,TAPE5=PRESDATA)
COMMON/DDI/PRES, LASTN, FIRSTN, INIT, MONTHB, DAYB, YEARB, NATSTA,
2OCCUP, JOB, MARRIA, SPOUSE, MONTHM, DAYM, YEARM, PARTY,
3NUMCHD, ELECT, EYEAR, VICTOR, EVOTES, LOSER, LOSERN, ADMIN,
4NUMBER, MONTHI, DAYI, YEARI, VPRES, VFIRST, VLAST,
5SPRES, OCCUPS, MRGS, LOSERS, ADMINS, VPS, ELECTS
C
C
DATA PRES/"PRES"/, LASTN/"LASTN"/, FIRSTN/"FIRSTN"/, INIT/"INIT"/,
1MONTHB/"MONTHB"/, DAYB/"DAYB"/, YEARB/"YEARB"/, NATSTA/"NATSTA"/,
2HEIGHT/"HEIGHT"/, PARTY/"PARTY"/, UNIVER/"UNIVER"/, HERIT/"HERIT"/
DATA PARTY/"PARTY"/, OCCUP/"OCCUP"/,
5JOB/"JOB"/, MARRIA/"MARRIA"/, SPOUSE/"SPOUSE"/, MONTHM/"MONTHM"/,
7YEARM/"YEARM"/, NUMCHD/"NUMCHD"/, ELECT/"ELECT"/, EYEAR/"EYEAR"/,
8VICTOR/"VICTOR"/, EVOTES/"EVOTES"/, LOSER/"LOSER"/, LOSERN
1/"LOSERN"/, DAYM/"DAYM"/
DATA
&ADMIN/"ADMIN"/, NUMBER/"NUMBER"/, MONTHI/"MONTHI"/, DAYI/"DAYI"/,
&YEARI/"YEARI"/, VPRES/"VPRES"/, VFIRST/"VFIRST"/, VLAST/"VLAST"/,
&SPRES/"SPRES"/, OCCUPS/"OCCUPS"/,
```

```

      &MRGS/"MRGS"/,LOSERS/"LOSERS"/,ADMINS/"ADMINS"/,PRCO/"PRCO"/,
      &VPS/"VPS"/,PC/"PC"/,ELECTS/"ELECTS"/
C
C PRESIDENTIAL DATABASE RETRIEVAL PROGRAM
C THIS PROGRAM RETRIEVES THE PRESIDENTS FIRST NAME MIDDLE INITIAL
C AND LAST NAME AND FOR EACH PRESIDENT PRINTS A LIST OF HIS
C OCCUPATIONS
C
C INITIALIZE THE DATA BASE SYSTEM WITH A CALL TO DINT
C
      CALL DINT(0,59,IERR)
C OPEN THE DATA BASE BY CALLING IDOP
      IDD=IDOP(2,3,0,IERR)
C
C FIND THE FIRST PRESIDENT
C
      CALL FFM(SPRES,IERR)
C
C CHECK FOR END OF SET
C
      IF(IERR .NE. -1) GO TO 1
      WRITE(59,99)
      99 FORMAT(1X,"DATA BASE IS EMPTY")
      STOP
      1 CONTINUE
C
C GET THE PRESIDENTS NAME
C
      CALL GFM(FIRSTN,SPRES,IF,IERR)
      CALL GFM(INIT,SPRES,II,IERR)
      CALL GFM(LASTN,SPRES,IL,IERR)
C
C PRINT OUT THE PRESIDENTS NAME
C
      WRITE(59,100) IF,II,IL
      100 FORMAT(1X,"PRESIDENT",1X,A10,1X,A1,".",1X,A10)
C
C GET THE OCCUPATIONS
C
C SET THE CURRENT MEMBER OF SET SPRES TO BE THE CURRENT OWNER
C OF THE SET OCCUPS
C
      CALL SOM( OCCUPS,SPRES,IERR)
C
C GET THE FIRST OCCUPATION
C
      CALL FFM(OCCUPS,IERR)
C CHECK FOR END OF SET
      IF(IERR .EQ. -1) GO TO 10
      20 CONTINUE
C
C GET JOB
C
      CALL GFM(JOB,OCCUPS,IJ,IERR)
C
C WRITE OUT THE OCCUPATION
C

```

```

        WRITE(59,101)IJ
101 FORMAT(10X,"OCCUPATION: ",A10)
C
C GET THE NEXT OCCUPATION RECORD
C
        CALL FNM(OCCUPS,IERR)
        IF(IERR .EQ.-1) GO TO 10
        GO TO 20
10 CONTINUE
C
C FIND THE NEXT PRESIDENT RECORD
C
        CALL FNM(SPRES,IERR)
        IF(IERR .EQ. -1 ) GO TO 1000
        GO TO 1
C
C
C 1000 CONTINUE
C
C CLOSE UP SHOP
C
        CALL DEND(IERR)
        STOP
        END

```

STEP 8: Compile and execute this program as performed in steps 5 and 6.

REFERENCES

1. Taylor, R. W., and Frank, R. L., "CODASYL Data-Base Management Systems", ACM Computing Surveys, Volume 8, Number 1, March 1976, pp. 67-103.
2. Hershey E. A., and Messink, P. W., "A Data Base Management System For PSA Based on DBTG 71", ISDOS Working Paper No. 88, The University of Michigan, 1975.
3. CODASYL Data Base Task Group, April 1971 Report, ACM, New York.
4. Hershey, Dissen, Foley, "Implementation and Modification Instructions for PSA in Source Form," ISDOS Working Paper Number 111, University of Michigan, August 1975.
5. Birss, Edward W., "ADBMS Installation Guide", Lawrence Livermore Laboratory, February 1977.

APPENDIX A
DETAILED DBCS SUBROUTINE DESCRIPTIONS

In the following appendix, the string $\rightarrow$ indicates input parameters or input currency indicators, while the string $\leftarrow$ indicates output parameters or currency indicators which are set by the DBCS.

These Subroutine Descriptions are largely taken from [2].

AMS -- Add Member to Set based on current record

AMS(set-type, record-type, IERR)

Arguments:

set-type -> character
record-type<- character

IERR-> fullword integer

Currency Indicators:

current record of record-type<-
current owner of set-type<-
current member of set-type<-

current member of set-type->

Comments:

The current record of the given record-type is added to the current set as identified by the current owner of that set-type.

The position of the new member in the set is determined by the ordering criteria given in the DDL Set Description for the given set-type.

The new member becomes the current member of the given set-type.

Return Codes:

- 01. Data base not open
- 02. Invalid set type
- 03. Invalid record type
- 06. Invalid member type for this set type
- 08. No current owner of set type
- 10. No current of record type
- 11. Record already member of set
- 99. Catastrophe

AMSK -- Add a Member to a Set based on Key

AMSK(set-type, db-key, IERR)

Arguments:

set-type -> character
db-key -> data base key

IERR <- full word integer

Currency Indicators:

Current Member of a Set ->
Current Owner of a Set ->

Current Member of a Set <-

Comments:

The record identified by the given data base key is added to the given set-type identified by the current owner of the set-type.
GETCDB(dbid,ierr) Get the id of the current data base

The new member becomes the current member of the given set-type.

Return Codes:

- 01. Data base not open
- 02. Invalid set type
- 06. Invalid member type for this set type
- 07. Invalid data base key
- 08. No current owner of set type
- 11. Record already member of set
- 99. Catastrophe

AMSM -- Add a Member to a Set based on current Member.

AMSM(set-type1, set-type2, IERR)

Arguments:

set-type1 -> character
set-type2 -> character

IERR <- fullword integer

Currency Indicators:

Current Owner of set-type1 ->
Current Member of set-type1 ->
Current Member of set-type2 ->

Current Member of set-type1 <-

Comments:

The current member of set-type2 is added to set-type1 as identified by the current owner of set-type1. The position of the new member is determined by the ordering criteria given in the DDL Set Description for set-type1.

The new member becomes the current member of set-type1.

Return Codes:

- 01. Data base not open
- 02. Invalid set type (1 or 2)
- 06. Invalid member type for set-type1
- 08. No current owner of set-type1
- 09. No current member of set-type2
- 11. Record already a member of set-type1
- 99. Catastrophe

AMSO -- Add a Member to a Set based on current Owner.

AMSO(set-type1, set-type2, IERR)

Arguments:

```
set-type1<- character
set-type2 -> character

IERR <- fullword integer
```

Currency Indicators:

```
Current Owner of set-type1 ->
Current Owner of set-type1 ->
Current Member of set-type2 ->

Current Member of set-type1 <-
```

Comments:

The current owner of set-type2 is added to set-type1 as identified by the current owner of set-type1. The position of the new member is determined by the ordering criteria given in the DDL Set Description for set-type1.

The new member becomes the current member of set-type1.

Return Codes:

```
01. Data base not open
02. Invalid set type (1 or 2)
06. Invalid member type for this set-type
08. No current owner of set-type (1 or 2)
11. Record already a member of set-type1
99. Catastrophe
```

CARD -- Get the Cardinality of a Set

CARD(set-type, data, IERR)

Arguments:

set-type -> character

data <- fullword integer

IERR <- fullword integer

Currency Indicators:

Current Owner of set-type

Comments:

The number of members of the given set type is returned in data.

Return Codes:

- 01. Data base not open
- 02. Invalid set type
- 08. No current owner of set type
- 99. Catastrophe

CLOS -- Close a data base

Subroutine CLOS(IERR)

Arguments:

IERR ← integer

Currency Indicators:

None.

Comments:

This routine closes the current data base. If this routine is used, a call to DEND must also be coded before program termination. However, since DEND closes all the data bases and closes the data base system, if DEND is used, a call to CLOS is usually unnecessary.

Return Codes:

- 01. Data Base not initialized
- 21. No current data base

CMK -- Check Membership based on Key

CMK(set-type, db-key, sw, IERR)

Arguments:

set-type -> character
db-key -> data base key

sw <- fullword integer
IERR <- fullword integer

Currency Indicators:

None.

Comments:

The record type identified by the given data base key is checked to see if it is a member of the given set type.

SW is set to -1 if the record type identified by db-key is not a member of the given set type; 0 if the record type is a legal member but is not a member of any instance of the set type; +1 if the record type is a legal member of the set type and is a member of some instance of the set type.

Return Codes:

- 01. Data Base not open
- 02. Invalid set type
- 07. Invalid data base key
- 99. Catastrophe

CMM -- Check Membership based on Member

CMM(set-type1, set-type2, sw, IERR)

Arguments:

```
set-type1 -> character
set-type2 -> character

sw <- fullword integer
IERR <- fullword integer
```

Currency Indicators:

Current member of set-type2 ->

Comments:

The current member of set-type2 is checked to see if it is a member of set-type1.

SW is set to -1 if the current member of set-type2 is not a member of set-type1; 0 if the current member is a legal member of set-type1 but is not a member of any instance of set-type1; +1 if the current member is a legal member of set-type1 and is a member of some instance of set-type1.

Return Codes:

- 01. Data Base not open
- 02. Invalid set type (1 or 2)
- 09. No current member of set-type
- 99. Catastrophe

CMO -- Check Membership based on Owner

CMO(set-type1, set-type2, sw, IERR)

Arguments:

set-type1 -> character
set-type2 -> character

sw <- fullword integer
IERR <- fullword integer

Currency Indicators:

Current owner of set-type2 ->

Comments:

The current owner of set-type2 is checked to see if it is a member of set-type1.

SW is set to -1 if the current owner of set-type2 is not a member of set-type1; 0 if the current owner is a legal member of set-type1 but is not a member of any instance of set-type1; +1 if the current owner is a legal member of set-type1 and is a member of some instance of set-type1.

Return Codes:

- 01. Data Base not open
- 02. Invalid set type (1 or 2)
- 09. No current owner of set-type2
- 99. Catastrophe

CMR -- Check Membership based on Record

CMR(set-type, record-type, sw, IERR)

Arguments:

set-type -> character
record-type -> character

sw <- fullword integer
IERR <- fullword integer

Currency Indicators:

Current record of record type ->

Comments:

The current record of record type is checked to see if it is a member of set-type.

SW is set to -1 if the current record of record type is not a member of set-type; 0 if the current record is a legal member of set-type but is not a member of any instance of set-type; +1 if the current record is a legal member of set-type and is a member of some instance of set-type.

Return Codes:

- 01. Data base not open
- 02. Invalid set type
- 03. Invalid record type
- 10. No current record of record type
- 99. Catastrophe

CMT -- Check the current Member Type

CMT(set-type, record-type, sw, IERR)

Arguments:

set-type -> character
record-type -> character

sw <- fullword integer
IERR <- fullword integer

Currency Indicators:

Current Member of set-type ->

Comments:

The record type of the current member of the given set-type is compared against the given record-type.

SW is set to -1 if the current member type is less than the given record-type; 0 if the current member type is equal to the given record-type; +1 if the current member type is greater than the given record-type.

No currency indicators are changed.

Return Codes:

- 01. Data Base not open
- 02. Invalid set type
- 09. No current member of set type
- 99. Catastrophe

COK -- Check Ownership based on Key

COK(set-type, db-key, sw, IERR)

Arguments:

set-type -> character
db-key -> data base key

sw <- fullword integer
IERR <- fullword integer

Currency Indicators:

None.

Comments:

The record type identified by the given data base key is checked to see if it is an owner of the given set type.

SW is set to -1 if the record type identified by db-key is not an owner of the given set type; 0 if the record type is a valid owner but is not an owner of any instance of the set type; +1 if the record type is a valid owner of the set type and is an owner of some instance of the set type.

Return Codes:

01. Data Base not open
02. Invalid set type
07. Invalid data base key
99. Catastrophe

COM -- Check Ownership based on Member

COM(set-type1, set-type2, sw, IERR)

Arguments:

set-type1 -> character
set-type2 -> character

sw <- fullword integer
IERR <- fullword integer

Currency Indicators:

Current member of set-type2 ->

Comments:

The current member of set-type2 is checked to see if it is an owner of set-type1.

SW is set to -1 if the current member of set-type2 is not an owner of set-type1; 0 if the current member is a legal owner of set-type1 but is not an owner of any instance of set-type1; +1 if the current member is a legal owner of set-type1 and is an owner of some instance of set-type1.

Return Codes:

- 01. Data Base not open
- 02. Invalid set type (1 or 2)
- 09. No current member of set-type
- 99. Catastrophe

COO -- Check Ownership based on Owner

COO(set-type1, set-type2, sw, IERR)

Arguments:

set-type1 -> character
set-type2 -> character

sw <- fullword integer
IERR <- fullword integer

Currency Indicators:

Current owner of set-type2 ->

Comments:

The current owner of set-type2 is checked to see if it is an owner of set-type1.

SW is set to -1 if the current owner of set-type2 is not an owner of set-type1; 0 if the current owner is a legal owner of set-type1 but is not an owner of any instance of set-type1; +1 if the current owner is a legal owner of set-type1 and is an owner of some instance of set-type1.

Return Codes:

- 01. Data Base not open
- 02. Invalid set type (1 or 2)
- 09. No current owner of set-type2
- 99. Catastrophe

COR -- Check Ownership based on Record

COR(set-type, record-type, sw, IERR)

Arguments:

set-type -> character
record-type -> character

sw <- fullword integer
IERR <- fullword integer

Currency Indicators:

Current record of record type ->

Comments:

The current record of record type is checked to see if it is an owner of set-type.

SW is set to -1 if the current record of record type is not an owner of set-type; 0 if the current record is a legal owner of set-type but is not an owner of any instance of set-type; +1 if the current record is a legal owner of set-type and is an owner of some instance of set-type.

Return Codes:

- 01. Data base not open
- 02. Invalid set type
- 03. Invalid record type
- 10. No current record of record type
- 99. Catastrophe

COT -- Check the current Owner Type

COT(set-type, record-type, sw, IERR)

Arguments:

set-type -> character
record-type -> character

sw <- fullword integer
IERR <- fullword integer

Currency Indicators:

Current Owner of set-type ->

Comments:

The record type of the current owner of the given set-type is compared against the given record-type.

SW is set to -1 if the current owner type is less than the given record-type; 0 if the current owner type is equal to the given record-type; +1 if the current owner type is greater than the given record-type.

No currency indicators are changed.

Return Codes:

- 01. Data Base not open
- 02. Invalid set type
- 09. No current member of set type
- 99. Catastrophe

CR -- Create a Record

CR(record-type, KEY, IERR)

Arguments:

record-type -> character

KEY <- data base key

IERR <- fullword integer

Currency Indicators:

Current record of record-type <-

Comments:

Space is allocated in the data base for an instance of the given record-type. The amount of space allocated is equal to the maximum size the record can be.

No items are initialized, i.e. the contents of all items in this record are garbage until set by calls to appropriate DBCS routines.

The new record becomes the current record of the given record-type, and the data base key of the newly created record is returned in KEY.

Return Codes:

- 01. Data base not open
- 03. Invalid record type
- 17. No more space in the data base
- 99. Catastrophe

CRS -- Create a Record and Store data

CRS(record-type, data, KEY, IERR)

Arguments:

record-type -> character
data -> depends on DDL

KEY <- data base key
IERR <- fullword integer

Currency Indicators:

Current record of record type <-

Comments:

A new record of the given record type is created, and initialized with the given data. The data is assumed to be in the same order and alignment as the items in the DDL for the given record type.

There is no check made that data is of the right types, alignments, or sizes. It is the responsibility of the user to insure that such parameters are consistent.

If any item type in the given record type is a depending item, the location of the depended-on item is assumed to contain the correct value to determine the number of instances of the depending-on item to store.

The data base key of the newly created record is returned in KEY.

Return Codes:

- 01. Data base not open
- 03. Invalid record type
- 13. Depended-on item too large or negative
- 17. No more space in data base
- 99. Catastrophe

DBST -- Get Statistics About Data Base System

DBST(sw, array, IERR)

Arguments:

```
sw -> fullword integer

array <- fullword integer array
IERR <- fullword integer
```

Currency Indicators:

None.

Comments:

The low order (right hand) five bits of sw are used as bit switches to independently request the following data base statistics:

sw Decimal Value	Bit Switches and Index into Array	What to Return
1	1 array(1)	Number of reads from data base
2	2 array(2)	Number of writes to data base
4	3 array(3)	Number of records created
8	4 array(4)	Number of records deleted
16	5 array(5)	Number of calls to DBKFND

The value of sw should be the sum of the decimal values of the bit switches for the statistics requested. Each bit is checked, and if non-zero the appropriate statistic is returned in array. Array should be dimensioned to five.

Return Codes:

```
01. Data base not open
99. Catastrophe
```

DELS -- Delete an entire Set

DELS(set-type, IERR)

Arguments:

set-type -> character

IERR <- fullword integer

Currency Indicators:

Current member of set type ->

Current member of set type <-

Other currency indicators as indicated below

Comments:

Every member of the given set-type is logically deleted from the data base.

DELS is logically equivalent to the following, for every member of the set:

CALL GKM(set-type, db-key, IERR)

CALL DRK(db-key, IERR)

See the description of these routines for more specific information.

Return Codes:

- 01. Data base not open
- 02. Invalid set type
- 08. No current owner of set
- 99. Catastrophe

DEND -- Close all data bases and the data base system

Subroutine DEND(IERR)

Arguments:

IERR ← integer

Currency Indicators:

None.

Comments:

Subroutine DEND closes all data bases that are open, and also closes the data base system. DINT must be called again if any data bases are to be opened following the call to DEND.

Return Codes:

01. Data base not initialized

DINT -- Data Base System initialization

Subroutine DINT(nbuf,llo,IERR)

Arguments:

nbuf -> integer

lio -> integer

IERR <- integer

Currency Indicators:

None.

Comments:

The data base system is initialized. Error messages are written onto device lio. Nbuf is the number of buffers used during execution of the program, however in this version this parameter is ignored, and nbuf is set by the block data subprogram.

Return Codes:

19. DINT already called

DR -- Delete a Record

DR(record-type, IERR)

Arguments:

record-type -> character

IERR <- fullword integer

Currency Indicators:

Current record of record type ->

Current record of record type <-
Other currency indicators as described below.

Comments:

The current record of the given record type is logically deleted from the data base.

DR is logically equivalent to:

CALL GKR(record-type, db-key, IERR)
CALL DRK(db-key, IERR)

See the descriptions of these routines for more specific information.

Return Codes:

- 01. Data base not open
- 03. Invalid record type
- 10. No current record of record type
- 99. Catastrophe

DRK -- Delete a Record based on Key

DRK(db-key, IERR)

Arguments:

db-key -> data base key

IERR <- fullword integer

Currency Indicators:

All indicators which become null and those which change
(see description)

Comments:

The record identified by the given data base key is logically deleted from the data base.

For all sets of which the record to be deleted is an owner, the set is deleted, i.e. the members of the set are no longer a member of any set of that type. The members are still logically in the data base however.

For all sets of which the record to be deleted is a member, the record is removed from the set, i.e. the previous member is linked to the next member relative to the deleted member.

For all set types of which the deleted record was the current owner, the current owner and current member currency indicators are set to null.

For all set types of which the deleted record was the current member, the current member is set to the logically next member of that set. (note that it is possible to reach the end of the set this way without it being indicated via IERR).

If the deleted record was the current of its record type, the current of that record type is set to null.

Return Codes:

- 01. Data base not open
- 07. Invalid data base key
- 99. Catastrophe

DRM -- Delete the Current Member of a Set

DRM(set-type, IERR)

Arguments:

set-type -> character

IERR <- fullword integer

Currency Indicators:

Current owner of set type ->

Current member of set type ->

Current member of set type <-

Other currency as indicated below

Comments:

The current member of the given set-type is logically deleted from the data base.

DRM is logically equivalent to:

CALL GKM(set-type, db-key, IERR)

CALL DRK(db-key, IERR)

See the descriptions of these routines for more specific information.

Return Codes:

- 01. Data Base not open
- 02. Invalid set type
- 08. No current owner of set
- 09. No current member of set
- 99. Catastrophe

DRO -- Delete the Current Owner of a Set

DRO(set-type, IERR)

Arguments:

set-type -> character

IERR <- fullword integer

Currency Indicators:

Current owner of set type ->

Current owner of set type <-

Current member of set type <-

Other currency as indicated below

Comments:

The current owner of the given set-type is logically deleted from the data base.

DRO is logically equivalent to:

CALL GKO(set-type, db-key, IERR)

CALL DRK(db-key, IERR)

See the descriptions of these routines for more specific information.

Return Codes:

- 01. Data Base not open
- 02. Invalid set type
- 08. No current owner of set
- 99. Catastrophe

FFM -- Find First Member of a set

FFM(set-type, IERR)

Arguments:

set-type -> character

IERR <- fullword integer

Currency Indicators:

Current owner of set type ->

Current member of set type <-

Comments:

The logically first member of the given set-type associated with the current owner of that set type is made the current member of that set type.

If the set contains no members, IERR is set to -1.

Return Codes:

- 01. Data base not open
- 02. Invalid set type
- 08. No current owner of set type
- 99. Catastrophe
- 1. End of set

FLM -- Find Last Member of a set

FLM(set-type, IERR)

Arguments:

set-type -> character

IERR <- fullword integer

Currency Indicators:

Current owner of set type ->

Current member of set type <=

Comments:

The logically last member of the given set-type associated with the current owner of that set type is made the current member of that set type.

If the set contains no members, IERR is set to -1.

Return Codes:

- 01. Data base not open
- 02. Invalid set type
- 08. No current owner of set type
- 99. Catastrophe
- 1. End of set

FMSK -- Find a Member of a Set based on Sort Key

FMSK(set-type, key-value, IERR)

Arguments:

set-type -> character
key-value -> depends on DDL
IERR <- fullword integer

Currency Indicators:

Current owner of set ->
Current member of set <-

Comments:

The current set of the given set type is searched for the first member with the given sort-key value. If such a member is found, it is made the current member. If such a member is not found, the current member of the set remains unchanged and IERR is set to -1.

Return Codes:

01. Data base not open
02. Invalid set type
08. No current owner of set type
18. Set type not sorted
99. Catastrophe
-1. End of set

FNM -- Find Next Member of a set

FNM(set-type, IERR)

Arguments:

set-type -> character
IERR <- fullword integer

Currency Indicators:

Current owner of set type ->
Current member of set type ->
Current member of set type <-

Comments:

The logically next member of the given set-type associated with the current owner of that set type is made the current member of that set type.

If there is no next member (the current member is the last, or the set is empty) IERR is set to -1 and the current member of the given set is set to null.

If there is no current member of the given set-type, then this is the same as FFM (Find First Member).

Return Codes:

01. Data base not open
02. Invalid set type
08. No current owner of set type
99. Catastrophe
-1. End of set

FNSK -- Find Next Member of a Sorted Set Based on Sort Key

FNSK(set-type, sort-key, IERR)**Arguments:**

set-type -> character
sort-key -> depends on DDL

IERR <- fullword integer

Currency Indicators:

Current member of set-type ->
Current owner of set-type ->

Current member of set-type <-

Comments:

The next member of set-type is made the current member of set-type. If the current member (on input) was the last member, IERR is set to -1 and a return is made; otherwise, the value of the given sort key is compared with the sort key of the new current member. If they are different, IERR is set to -1.

Return Codes:

- 01. Data base not open
- 02. Invalid set type
- 08. No current owner of set type
- 09. No current member of set type
- 18. Set type not sorted
- 1. End of set

FPM -- Find Previous Member of a set

FPM(set-type, IERR)

Arguments:

set-type -> character

IERR <- fullword integer

Currency Indicators:

Current owner of set type ->

Current member of set type ->

Current member of set type <-

Comments:

The logically previous member of the given set-type associated with the current owner of that set type is made the current member of that set type.

If there is no previous member (the current member is the last, or the set is empty) IERR is set to -1 and the current member of the given set is set to null.

If there is no current member of the given set-type, then this is the same as FLM (Find Last Member).

Return Codes:

- 01. Data base not open
- 02. Invalid set type
- 08. No current owner of set type
- 99. Catastrophe
- 1. End of set

GETCDB -- Get the Current Data Base's identifier

Subroutine GETCDB(dbid,IERR)

Arguments:

dbid -> integer

IERR <- integer

Currency Indicators:

Current Data Base

Comments:

This routine gets the identifier associated with the current data base. This identifier is the one returned by IDOP when the current data base was opened.

Return Codes:

21. No current data base

GETK -- GET data from record based on Key

GETK(db-key, data, IERR)

Arguments:

db-key -> data base key

data <- depends on DDL

IERR <- fullword integer

Currency Indicators:

None.

Comments:

The values of all items associated with the record with the given db-key are returned in data. The values are returned in the same order and with the same alignment as the items in the DDL for the given record type.

There is no check made that data is of the right types, alignments, or sizes.

If any item type in the record indicated by the given db-key is a depending item, the value of the associated depended-on item in the record is used to determine the number of instances of the depending-on item to return.

Return Codes:

- 01. Data Base not open
- 07. Invalid data base key
- 13. Depended-on item too large or negative
- 99. Catastrophe

GETM -- GET data from current Member

GETM(set-type, data, IERR)

Arguments:

set-type -> character
data <- depends on DDL
IERR <- fullword integer

Currency Indicators:

Current owner of set type ->
Current member of set type ->

Comments:

The values of all items associated with the current member of the given set type are returned in data. The values are returned in the same order and with the same alignment as the items in the DDL for the given record type.

There is no check made that data is of the right types, alignments, or sizes.

If any item type in the current member of the given set-type is a depending item, the value of the associated depended-on item in the record is used to determine the number of instances of the depending-on item to return.

Return Codes:

- 01. Data Base not open
- 08. No current owner of set type
- 09. No current member of set type
- 13. Depended-on item too large or negative
- 99. Catastrophe

GETO -- GET data from current Owner

GETO(set-type, data, IERR)

Arguments:

set-type -> character

data <- depends on DDL

IERR <- fullword integer

Currency Indicators:

Current owner of set type ->

Comments:

The values of all items associated with the current owner of the given set type are returned in data. The values are returned in the same order and with the same alignment as the items in the DDL for the given record type.

There is no check made that data is of the right types, alignments, or sizes.

If any item type in the current owner of the given set-type is a depending item, the value of the associated depended-on item in the record is used to determine the number of instances of the depending-on item to return.

Return Codes:

- 01. Data Base not open
- 08. No current owner of set type
- 13. Depended-on item too large or negative
- 99. Catastrophe

GETR -- GET data from current Record

GETR(record-type, data, IERR)

Arguments:

record-type -> character

data <- depends on DDL

IERR <- fullword integer

Currency Indicators:

Current record of record type ->

Comments:

The values of all items associated with the current record of the given record type are returned in data. The values are returned in the same order and with the same alignment as the items in the DDL for the given record type.

There is no check made that data is of the right types, alignments, or sizes.

If any item type in the current record of the given record-type is a depending item, the value of the associated depended-on item in the record is used to determine the number of instances of the depending-on item to return.

Return Codes:

- 01. Data Base not open
- 10. No current of record type
- 13. Depended-on item too large or negative
- 99. Catastrophe

GFK -- Get Field based on Key

GFK(item-type, db-key, data, IERR)

Arguments:

item-type -> character
db-key -> data base key

data <- depends on DDL
IERR <- fullword integer

Currency Indicators:

None.

Comments:

The value of the given item-type in the record identified by the given db-key is returned in data.

There is no check made that data is of the right type, alignment, or size.

If the item type is a depending-on item, the depended-on item in the record is used to determine the number of instances of the item to return.

Return Codes:

- 01. Data base not open
- 04. Invalid item type for this record type
- 07. Invalid data base key
- 13. Depended-on item too large or negative
- 99. Catastrophe

GFM -- Get Field from current Member of a set

GFM(item-type, set-type, data, IERR)

Arguments:

item-type -> character
set-type -> character

data <- depends on DDL
IERR <- fullword integer

Currency Indicators:

Current owner of set type ->
Current member of set type ->

Comments:

The value of the given item-type in the current member of the given set-type is returned in data.

There is no check made that data is of the right type, alignment, or size.

If the item type is a depending-on item, the depended-on item in the record is used to determine the number of instances of the item to return.

Return Codes:

- 01. Data base not open
- 02. Invalid set type
- 04. Invalid item type for this record type
- 08. No current owner of set type
- 09. No current member of set type
- 13. Depended-on item too large or negative
- 99. Catastrophe

GFO -- Get Field from current Owner of a set

GFO(item-type, set-type, data, IERR)

Arguments:

item-type -> character
set-type -> character

data <- depends on DDL
IERR <- fullword integer

Currency Indicators:

Current owner of set type ->

Comments:

The value of the given item-type in the current owner of the given set-type is returned in data.

There is no check made that data is of the right type, alignment, or size.

If the item type is a depending-on item, the depended-on item in the record is used to determine the number of instances of the item to return.

Return Codes:

- 01. Data base not open
- 02. Invalid set type
- 04. Invalid item type for this record type
- 08. No current owner of set type
- 13. Depended-on item too large or negative
- 99. Catastrophe

GFR -- Get Field from Record

GFR(item-type, record-type, data, IERR)

Arguments:

item-type -> character
record-type -> character

data <- depends on DDL
IERR <- fullword integer

Currency Indicators:

Current of record type ->

Comments:

The value of the given item-type in the current record of the given record-type is returned in data.

There is no check made that data is of the right type, alignment, or size.

If the item type is a depending-on item, the depended-on item in the record is used to determine the number of instances of the item to return.

Return Codes:

- 01. Data base not open
- 03. Invalid record type
- 04. Invalid item type for this record type
- 10. No current record of record type
- 13. Depended-on item too large or negative
- 99. Catastrophe

GKM -- Get Key of current Member of a set

GKM(set-type, db-key, IERR)

Arguments:

set-type -> character
db-key <- data base key
IERR <- fullword integer

Currency Indicators:

Current owner of set type ->
Current member of set type ->

Comments:

The value of the data base key associated with the current member of the given set-type is returned in db-key. If the current member of the given set-type is null, a data base key value of zero is returned.

See section on data base keys for description of restrictions and assumptions which apply to legal operations with data base keys.

Return Codes:

01. Data base not open
02. Invalid set type
99. Catastrophe

GKO -- Get Key of current Owner of a set

GKO(set-type, db-key, IERR)

Arguments:

set-type -> character
db-key <- data base key
IERR <- fullword integer

Currency Indicators:

Current owner of set type ->

Comments:

The value of the data base key associated with the current owner of the given set-type is returned in db-key. If the current owner of the given set-type is null, a data base key value of zero is returned.

See section on data base keys for description of restrictions and assumptions which apply to legal operations with data base keys.

Return Codes:

- 01. Data base not open
- 02. Invalid set type
- 99. Catastrophe

GKR -- Get Key of current Record

GKR(record-type, db-key, IERR)

Arguments:

record-type -> character

db-key <- data base key

IERR <- fullword integer

Currency Indicators:

Current record of record.type ->

Comments:

The value of the data base key associated with the current record of the given record-type is returned in db-key. If the current record of the given record-type is null, a data base key value of zero is returned.

See section on data base keys for description of restrictions and assumptions which apply to legal operations with data base keys.

Return Codes:

- 01. Data base not open
- 03. Invalid record type
- 99. Catastrophe.

GTK -- Get record Type based on Key

GTK(db-key, type, IERR)

Arguments:

db-key -> data base key
type <- character
IERR <- fullword integer

Currency Indicators:

None.

Comments:

The record type name of the record specified by the given db-key is returned in type.

The record type name is returned in character format, and is padded to the maximum name length with trailing blanks.

Return Codes:

01. Data base not open
07. Invalid data base key
99. Catastrophe

GTM -- Get record Type of current Member

GTM(set-type, type, IERR)

Arguments:

set-type -> character
type <- character
IERR <- fullword integer

Currency Indicators:

Current owner of set type ->
Current member of set type ->

Comments:

The record type name of the current member of the given set-type is returned in type.

The record type name is returned in character format, and is padded to the maximum name length with trailing blanks.

Return Codes:

01. Data base not open
02. Invalid set type
08. No current owner of set type
09. No current member of set type
99. Catastrophe

GTO -- Get record Type of current Owner

GTO(set-type, type, IERR)

Arguments:

set-type -> character
type <- character
IERR <- fullword integer

Currency Indicators:

Current owner of set type ->

Comments:

The record type name of the current owner of the given set-type is returned in type.

The record type name is returned in character format, and is padded to the maximum name length with trailing blanks.

Return Codes:

- 01. Data base not open
- 02. Invalid set type
- 08. No current owner of set type
- 99. Catastrophe

ICFK -- Check Field Based on Key

Integer Function ICFK(item-type, db-key, data, IERR)

Arguments:

item-type -> character
db-key -> data base key
data -> depends on DDL

IERR <- fullword integer

Currency Indicators:

None.

Comments:

The value of the given item type in the record identified by the given data base key is compared against the given data value.

The returned value is -1 if the value of the item is less than data; 0 if the value of the item is equal to data; +1 if the value of the item is greater than data.

There is no check made that the data is of the right type, alignment, or size.

Return Codes:

- 01. Data base not open
- 04. Invalid item type for the record type
- 07. Invalid data base key
- 99. Catastrophe

ICFM -- Check Field Based on Current Member

Integer Function ICFM(item-type, set-type, data, IERR)

Arguments:

item-type -> character
set-type -> character
data -> depends on DDL

IERR <- fullword integer

Currency Indicators:

Current Member of set type

Comments:

The value of the given item type in the record identified by the current member of the given set-type is compared against the given data value.

The returned value is -1 if the value of the item is less than data; 0 if the value of the item is equal to data; +1 if the value of the item is greater than data.

There is no check made that the data is of the right type, alignment, or size.

Return Codes:

- 01. Data base not open
- 02. Invalid set type
- 04. Invalid item type for the record type
- 09. No current member of set
- 99. Catastrophe

ICFO -- Check Field Based on Current Owner

Integer Function ICFO(item-type, set-type, data, IERR)

Arguments:

item-type -> character
set-type -> character
data -> depends on DDL
IERR <- fullword integer

Currency Indicators:

Current Owner of set type

Comments:

The value of the given item type in the record identified by the current owner of the given set-type is compared against the given data value.

The returned value is -1 if the value of the item is less than data; 0 if the value of the item is equal to data; +1 if the value of the item is greater than data.

There is no check made that the data is of the right type, alignment, or size.

Return Codes:

- 01. Data base not open
- 02. Invalid set type
- 04. Invalid item type for the record type
- 08. No current owner of set
- 99. Catastrophe

ICFR -- Check Field Based on Current Record

Integer Function ICFR(item-type, record-type, data, IERR)

Arguments:

item-type -> character
record-type -> character
data -> depends on DDL

IERR <- fullword integer

Currency Indicators:

Current record of record type ->

Comments:

The value of the given item type in the record identified by the current record of the given record-type is compared against the given data value.

The returned value is -1 if the value of the item is less than data; 0 if the value of the item is equal to data; +1 if the value of the item is greater than data.

There is no check made that the data is of the right type, alignment, or size.

Return Codes:

- 01. Data base not open
- 02. Invalid set type
- 04. Invalid item type for the record type
- 10. No current record of record type
- 99. Catastrophe

IDOP -- Open a Data base

Function IDOP(liodbf, liodbt, use, IERR)

Arguments:

liodbf -> integer
liodbt -> integer
use -> integer

IERR <- integer

Currency Indicators:

None.

Comments:

IDOP returns an integer which contains an identifier for the data base. This identifier is used by GETCDB and SETCDB to determine and set the current data base. (All subroutines operate on the current data base.) When IDOP is called the newly opened data base becomes the current data base.

Use is 0 for read only data base access, 1 for read/write data base access.

Return Codes:

- 01. Data base not initialized
- 14. Data Base already opened
- 15. DDL tables inconsistent
- 17. No more space in data base
- 22. Data base number exceeded
- 23. Illegal Logical unit for DBF or DBT

RM -- Remove the current Member from a set

RM(set-type, IERR)

Arguments:

set-type -> character

IERR <- fullword integer

Currency Indicators:

Current owner of set type ->

Current member of set type ->

Current member of set type <-

Comments:

The current member of the given set-type is logically removed from the set i.e., the previous member is linked to the next member.

The next member of the set becomes the current member of the given set-type unless the removed member was the last member (in order, in which case the current member is set to null and IERR is set to -1.

Note that the record is only removed from the set; it is not deleted from the data base.

Return Codes:

- 01. Data base not open
- 02. Invalid set type
- 08. No current owner of set type
- 09. No current member of set type
- 99. Catastrophe
- 1. End of set

RS -- Remove the entire Set

RS(set-type, IERR)

Arguments:

set-type -> character

IERR <- fullword integer

Currency Indicators:

Current owner of set type ->

Current member of set type <-

Comments:

The current set is set to have no members, i.e. all members of the set are logically removed from the set.

The current member of the given set-type becomes null.

Note that the records removed from the set are not deleted from the data base, nor is the owner of the given set-type deleted from the data base.

Return Codes:

- 01. Data base not open
- 02. Invalid set type
- 08. No current owner of set type
- 99. Catastrophe

SETCDB -- SET a data base to the Current Data Base

Subroutine SETCDB(dbid,IERR)

Arguments:

dbid -> integer

IERR <- integer

Currency Indicators:

Current data base

Comments:

This routine sets the current data base to be that data base identified by dbid. Dbid must have a value previously returned by IDOP.

Return Codes:

21. No current data base

SETK -- SET the data of a record based on Key

SETK(db-key, data, IERR)

Arguments:

db-key -> data base key

data -> depends on DDL

IERR <- fullword integer

Currency Indicators:

None.

Comments:

The value of data is stored in the record identified by the given data base key.

There is no check made that data is of the right type, alignment, or size.

The value of data is stored beginning at the first item. The value of data is taken as a stream of data values, which are stored in all the items belonging to the record type, beginning with the first and continuing to the last (including all instances of a depending item, if present).

Return Codes:

01. Data base not open

07. Invalid key

99. Catastrophe

SETM -- SET the data of a record based on current Member

SETM(set-type, data, IERR)

Arguments:

set-type -> character
data -> depends on DDL

IERR <- fullword integer

Currency Indicators:

Current member of a set ->

Comments:

The value of data is stored in the current member of the given set-type.
SETM is logically equivalent to:

CALL GKM(set-type, db-key, IERR)
CALL SETK(db-key, data, IERR)

See the descriptions of these routines for specific information.

Return Codes:

- 01. Data Base not open
- 02. Invalid set type
- 09. No current member of set type
- 99. Catastrophe

SETO -- Set the data of a record based on current Owner

SETO(set-type, data, IERR)

Arguments:

set-type -> character
 data -> depends on DDL
 IERR <- fullword integer

Currency Indicators:

Current owner of a set ->

Comments:

The value of data is stored in the current owner of the given set-type.
 SETO is logically equivalent to:

CALL GKO(set-type, db-key, IERR)
 CALL SETK(db-key, data, IERR)

See the descriptions of these routines for specific information.

Return Codes:

- 01. Data Base not open
- 02. Invalid set type
- 08. No current owner of set type
- 99. Catastrophe

SETR -- SET the data of a Record based on current Record

SETR(record-type, data, IERR)

Arguments:

record-type -> character

data -> depends on DDL

IERR <- fullword integer

Currency Indicators:

Current record of record-type ->

Comments:

The value of data is stored in the current record of the given record-type.

SETR is logically equivalent to:

CALL GKR(record-type, db-key, IERR)

CALL SETK(db-key, data, IERR)

See the descriptions of these routines for specific information.

Return Codes:

- 01. Data Base not open
- 03. Invalid record type
- 10. No current record of record type
- 99. Catastrophe

SFK -- Set Field in record based on Key

SFK(item-type, db-key, data, IERR

Arguments:

item-type -> character
db-key -> data base key
data -> depends on DDL

IERR <- fullword integer

Currency Indicators:

None.

Comments:

The value of data is stored into the given item-type of the record identified by the given db-key.

There is no check made that data is of the right type, alignment, or size.

If the item-type is a depending item, the depended-on item value in the record is used to determine the number of instances of the given item-type to store.

Return Codes:

- 01. Data base not open
- 04. Invalid item type for this record type
- 07. Invalid data base key
- 13. Depended-on item too large or negative
- 17. No more space in data base
- 99. Catastrophe

SFM -- Set Field in the current Member of a set

SFM(item-type, set-type, data, IERR)

Arguments:

item-type -> character
set-type -> character
data -> depends on DDL

IERR <- fullword integer

Currency Indicators:

Current owner of set type<-
Current member of set type<-

Comments:

The value of data is stored into the given item-type of the current member of the given set-type.

There is no check made that data is of the right type, alignment, or size.

If the item-type is a depending item, the depended-on item value in the record is used to determine the number of instances of the given item-type to store.

Return Codes:

- 01. Data base not open
- 02. Invalid set type
- 04. Invalid item type for this record type
- 08. No current owner of set type
- 09. No current member of set type
- 13. Depended-on item too large or negative
- 17. No more space in data base
- 99. Catastrophe

SFO -- Set Field in current Owner of a set

SFO(item-type, set-type, data, IERR)

Arguments:

item-type -> character
set-type -> character
data -> depends on DDL

IERR <- fullword integer

Currency Indicators:

Current owner of set type<-

Comments:

The value of data is stored into the given item-type of the current owner of the given set-type.

There is no check made that data is of the right type, alignment, or size.

If the item-type is a depending item, the depended-on item value in the record is used to determine the number of instances of the given item-type to store.

Return Codes:

- 01. Data base not open
- 02. Invalid set type
- 04. Invalid item type for this record type
- 08. No current owner of set type
- 13. Depended-on item too large or negative
- 17. No more space in data base
- 99. Catastrophe

SFR -- Set Field in the current Record

SFR(item-type, record-type, data, IERR)

Arguments:

item-type -> character
record-type -> character
data -> depends on DDL

IERR <- fullword integer

Currency Indicators:

Current record of record type<-

Comments:

The value of data is stored into the given item-type of the current record of the given record-type.

There is no check made that data is of the right type, alignment, or size.

If the item-type is a depending item, the depended-on item value in the record is used to determine the number of instances of the given item-type to store.

Return Codes:

- 01. Data base not open
- 03. Invalid record type
- 04. Invalid item type for this record type
- 10. No current record of record type
- 13. Depended-on item too large or negative
- 17. No more space in data base
- 99. Catastrophe

SKFM -- Set owner based on Key and Find Member

SKFM(set-type, db-key, data, IERR)

Arguments:

set-type -> character
 db-key -> data base key
 data -> fullword integer or character

 IERR (ouput) fullword integer

Currency Indicators:

Current Owner of set ->

 Current Owner of set <=
 Current Member of set <=

Comments:

The current owner of the given set-type is set based on the given data base key. If the set is empty, IERR is set to -1 and a return is made; other wise set-type is searched for a member with the given sort key value. If no such member is found, the value of IERR is set to -1. This subroutine is logically equivalent to the following:

```
CALL SOK(set-type, db-key, IERR)
IF(IERR.LT. 0 ) RETURN
CALL FMSK(set-type, data, IERR)
```

See the description of these routines for specific information.

Return Codes:

- 01. Data base not open
- 02. Invalid set type
- 05. Invalid owner type for this set type
- 07. Invalid data base key value
- 08. No current owner of this set type
- 18. Set not sorted
- 99. Catastrophe
- 1. End of set

SM -- Set the current Member of a set based on the current record

SM(set-type, record-type, IERR)

Arguments:

set-type -> character
record-type -> character

IERR <- fullword integer

Currency Indicators:

Current record of record type ->

Current owner of set type <-
Current member of set type <-

Comments:

The current record of the given record type becomes the current member of the given set type. The current owner of the given set type becomes the owner associated with the new current member.

Return Codes:

- 01. Data base not open
- 02. Invalid set type
- 03. Invalid record type
- 06. Invalid member type for this set type
- 10. No current record of record type
- 12. Record not member of set
- 99. Catastrophe

SMFM -- Set owner based on current Member, and Find Member
based on sort key

SMFM(set-type1, set-type2, data, IERR)

Arguments:

set-type1 -> character
 set-type2 -> character
 data -> fullword integer or character

 IERR <- fullword integer

Currency Indicators:

Current owner of set-type1 ->
 Current owner of set-type2 ->
 Current member of set-type2 ->

 Current owner of set-type1 <-
 Current member of set-type1->

Comments:

The current owner of set-type1 is set based on the current member of set-type2. If set-type1 is empty, IERR is set to -1, and a return is made; otherwise set-type1 is searched for a member with the given sort key value. If no such member is found, the value of IERR is set to -1. This subroutine is logically equivalent to:

```
CALL SOM(set-type1, set-type2, IERR)
IF(IERR .LT. 0) RETURN
CALL FMSK(set-type1, data, IERR)
```

See the descriptions for these routines for specific information.

Return Codes:

- 01. Data base not open
- 02. Invalid set-type
- 05. Invalid owner type for set type
- 08. No current owner type for set type
- 09. No current member type for set type
- 18. Set not sorted
- 99. Catastrophe
- 1. End of set

SMK -- Set the current Member of a set based on Key

SMK(set-type, db-key, IERR)

Arguments:

set-type -> character
db-key -> data base key

IERR <- fullword integer

Currency Indicators:

Current owner of set type <-
Current member of set type <-

Comments:

The record identified by the given db-key becomes the current member of the given set type. The current owner of the given set type becomes the owner associated with the new current member.

Return Codes:

- 01. Data base not open
- 02. Invalid set type
- 06. Invalid member type for this set type
- 07. Invalid data base key
- 12. Record not member of set
- 99. Catastrophe

SMM -- Set the current Member of a set based on the current Member

SMM(set-type1, set-type2, IERR)

Arguments:

set-type1 -> character
set-type2 -> character

IERR <- fullword integer

Currency Indicators:

Current owner of set-type2 ->
Current member of set-type2 ->

Current owner of set-type1 <-
Current member of set-type1 <-

Comments:

The current member of set-type2 is also made the current member of set-type1. The current owner of set-type1 is set to the owner of the new current member of set-type1.

Return Codes:

- 01. Data base not open
- 02. Invalid set type
- 06. Invalid member type for this set type
- 08. No current owner of set type
- 09. No current member of set type
- 12. Record not member of set
- 99. Catastrophe

SMO -- Set the current Member of a set based on current Owner

SMO(set-type1, set-type2, IERR)

Arguments:

set-type1 -> character
set-type2 -> character

IERR <- fullword integer

Currency Indicators:

Current owner of set-type2 ->

Current owner of set-type1 <-
Current member of set-type1 <-

Comments:

The current owner of set-type2 is also made the current member of set-type1. The current owner of set-type1 is set to the owner of the new current member of set-type1.

Return Codes:

01. Data base not open
02. Invalid set type
06. Invalid member type for this set type
08. No current owner of set type
12. Record not member of set
99. Catastrophe

SO -- Set the current Owner of a set based on a current record

SO(set-type, record-type, IERR)

Arguments:

set-type -> character
 record-type -> character

 IERR <- fullword integer

Currency Indicators:

Current record of record type ->

 Current owner of set type <-
 Current member of set type <-

Comments:

The current record of the given record type becomes the current owner of the given set type. The current member of the given set type becomes the owner associated with the new current member.

If the set is empty, the current member of the given set-type becomes null, and IERR is set to -1.

Return Codes:

- 01. Data base not open
- 02. Invalid set type
- 03. Invalid record type
- 05. Invalid owner type for this set type
- 10. No current record of record type
- 99. Catastrophe

SOFM -- Set Owner based on current Owner and Find Member

based on sort key

SOFM(set-type1, set-type2, data, IERR)

Arguments:

set-type1 -> character
 set-type2 -> character
 data -> fullword integer or character

 IERR <- fullword integer

Currency Indicators:

Current owner of set-type1 ->
 Current owner of set-type2 ->

 Current owner of set-type1 <-
 Current member of set-type1->

Comments:

The current owner of set-type1 is set based on the current owner of set-type2. If set-type1 is empty, IERR is set to -1, and a return is made; otherwise set-type1 is searched for a member with the given sort key value. If no such member is found, the value of IERR is set to -1. This subroutine is logically equivalent to:

```
CALL SOO(set-type1, set-type2, IERR)
IF(IERR.LT. 0) RETURN
CALL FMSK(set-type1, data, IERR)
```

See the descriptions for these routines for specific information.

Return Codes:

01. Data base not open
 02. Invalid set-type
 05. Invalid owner type for set type
 08. No current owner type for set type
 18. Set not sorted
 99. Catastrophe
 -1. End of set

SOK -- Set the current Owner of a set based on Key

SOK(set-type, db-key, IERR)

Arguments:

set-type ->
db-key -> data base key

IERR <- fullword integer

Currency Indicators:

Current owner of set type <-
Current member of set type <-

Comments:

The record identified by the given db-key becomes the current owner of the given set type. The current member of the given set type is taken to be the first member of the set associated with the new current member.

If the set is empty, the current member of the given set-type becomes null, and IERR is set to -1.

Return Codes:

- 01. Data base not open
- 02. Invalid set type
- 05. Invalid owner type for this set type
- 07. Invalid data base key
- 99. Catastrophe

SOM -- Set the current Owner of a set based on the current

Member of a set

SOM(set-type1, set-type2, IERR)

Arguments:

set-type1 -> character

set-type2 -> character

IERR <- fullword integer

Currency Indicators:

Current owner of set-type2 ->

Current member of set-type2 ->

Current owner of set-type1 <-

Current member of set-type1 <-

Comments:

The current member of set-type2 is made the current owner of set-type1.
The current member of set-type1 is set to the first member of that set.
If the new owner of set-type1 has no members, the current member of
set-type1 is set to null, and IERR is set to -1.

Return Codes:

- 01. Data base not open
- 02. Invalid set type
- 05. Invalid owner type for set type
- 08. No current owner of set type
- 09. No current member of set type
- 99. Catastrophe

S00 -- Set the current based on the current

Owner of a set

S00(set-type1, set-type2, IERR)

Arguments:

set-type1 -> character
 set-type2 -> character

 IERR <- fullword integer

Currency Indicators:

Current owner of set-type2 ->

 Current owner of set-type1 <-
 Current member of set-type1 <-

Comments:

The current owner of set-type2 is made the current owner of set-type1.
 The current member of set-type1 is set to the first member of that set.
 If the new owner of set-type1 has no members, the current member of
 set-type1 is set to null, and IERR is set to -1.

Return Codes:

01. Data base not open
 02. Invalid set type
 05. Invalid owner type for set type
 08. No current owner of set type
 09. No current member of set type
 03. Invalid record type

SRFM -- Set owner based on current Record and Find Member

based on sort key

SRFM(set-type, record-type, data, IERR)

Arguments:

set-type -> character
record-type -> character
data -> fullword integer or character

IERR <- fullword integer

Currency Indicators:

Current owner of set-type ->
Current record of record-type ->

Current owner of set-type <-
Current member of set-type->

Comments:

The current record of set-type is set based on the current record of record-type. If set-type is empty, IERR is set to -1, and a return is made; otherwise set-type is searched for a member with the given sort key value. If no such member is found, the value of IERR is set to -1. This subroutine is logically equivalent to:

```
CALL SO(set-type, record-type, IERR)
IF(IERR .LT. 0) RETURN
CALL FMSK(set-type, data, IERR)
```

See the descriptions for these routines for specific information.

Return Codes:

- 01. Data base not open
- 02. Invalid set-type
- 03. Invalid record type
- 05. Invalid owner type for set type
- 08. No current owner type for set type
- 10. No current record of record type
- 18. Set not sorted
- 99. Catastrophe
- 1. End of set

SRK -- Set current Record based on Key

SRK(db-key, IERR)

Arguments:

db-key -> data base key

IERR <- fullword integer

Currency Indicators:

Current record of record type <-

Comments:

The record identified by the given data base key is made the current record of that record type.

Return Codes:

- 01. Data base not open
- 07. Invalid data base key
- 99. Catastrophe

SRM -- Set current Record based on current Member of a set

SRM(set-type, IERR)

Arguments:

set-type -> character

IERR <- fullword integer

Currency Indicators:

Current owner of set type ->

Current member of set type ->

Current record of record type <-

Comments:

The current member of the given set-type is inspected to determine its record type, then the current record of that record type is set to current member of the given set type.

Return Codes:

- 01. Data base not open
- 02. Invalid set type
- 08. No current owner of set type
- 09. No current member of set type
- 99. Catastrophe

SRO -- Set current Record based on current Owner of a set

SRO(set-type, IERR)

Arguments:

set-type -> character

IERR <- fullword integer

Currency Indicators:

Current owner of set type ->

Current member of set type ->

Current record of record type <-

Comments:

The current owner of the given set-type is inspected to determine its record type, then the current record of that record type is set to current member of the given set type.

Return Codes:

- 01. Data base not open
- 02. Invalid set type
- 08. No current owner of set type
- 99. Catastrophe

USE -- Set up error recovery procedure

USE(error-number, subroutine-name, IERR)

Arguments:

error-number -> fullword integer
subroutine-name -> external name or fullword integer

IERR <- fullword integer

Currency Indicators:

None.

Comments:

Tells the DBCS the name of the subroutine to call if a certain type of error occurs, or if any error occurs.

Subroutine-name must be declared as EXTERNAL in the calling program.

If error-number is zero, then the given subroutine-name is called for any kind of error.. If USE is subsequently called with a nonzero error-number, then the default subroutine given when USE was called with error number set to zero is overridden for that error type only. A subsequent call to USE with an error-number of zero resets all to the given default.

The call to subroutine-name is of the following form: CALL sub (error-number, routine-number) where error-number is an integer variable containing the error number, and routine-number is an integer variable containing the routine number during which the error occurred.

If the given subroutine returns normally when called by the DBCS, control is returned to the routine which called the DBCS with the error flagged in IERR as if the subroutine had not been called.

If the subroutine name is zero, then standard error processing will take place for the given error-number. Note that if error-number is also zero the standard error processing is in effect for all errors. OPEN, in effect, calls USE with error-number equal to zero and subroutine name equal to zero.

Return Codes:

- 01. Data base not open
- 19. Invalid error number
- 99. Catastrophe

3.3 Errors

- 01. Data base not open
- 02. Invalid set type
- 03. Invalid record type
- 04. Invalid item type for this record type
- 05. Invalid owner type for this set type
- 06. Invalid member type for this set type
- 07. Invalid data base key
- 08. No current owner of set type
- 09. No current member of set type
- 10. No current record of record type
- 11. Record already member of set
- 12. Record not member of set
- 13. Depended-on item too large or negative
- 14. Data base already open
- 15. DDL tables inconsistent
- 16. Data base file inconsistent
- 17. No more space in Data base
- 18. Set not sorted
- 19. DINT already called
- 20. Invalid number of buffers
- 22. Data Base number exceeded
- 23. Illegal logical unit for dbf or dbt (zeros not legal)
- 99. Catastrophe
- 1. End of set

3.4 Summary of Currency Indicators

INPUT

a = current record of record type
 b = current owner of set
 c = current member of set

OUTPUT

d = current record of record type
 e = current owner of set
 f = current member of set

	a	b	c	d	e	f
	-----			-----		
1. AMS	X	X	X	X		
2. AMSK	X	X		X		
3. AMSM	X	X		X		
4. AMSO	X	X		X		
5. CARD		X				
6. CLOS						
7. CMK						
8. CMM		X				
9. CMO			X			
10. CMR			X			
11. CMT		X				
12. COK						
13. COM		X				
14. COO			X			
15. COR			X			
16. COT		X				
17. CR						X
18. CRS						X
19. DBST						
20. DELS		X	X	X		
21. DR			X			X
22. DRK				X		
23. DRM		X	X	X		
24. DRO			X	X	X	
25. FFM			X	X		
26. FLM			X	X		
27. FMSK			X	X		
28. FNM		X	X	X		
29. FNSK		X	X	X		
30. FPM		X	X	X		
31. GETK						
32. GETM		X	X			
33. GETO			X			
34. GETR			X			

INPUT

a = current record of record type
 b = current owner of set
 c = current member of set

OUTPUT

d = current record of record type
 e = current owner of set
 f = current member of set

	a	b	c	d	e	f
	-----			-----		
35. GFK						
36. GFM	X	X				
37. GFO		X				
38. GFR			X			
39. GKM	X	X				
40. GKO		X				
41. GKR			X			
42. GTK						
43. GTM	X	X				
44. GTO		X				
45. ICFK						
46. ICFM						
47. ICFO						
48. ICFR						
49. OPEN						X
50. RM	X	X			X	
51. RS		X			X	
52. SETK						
53. SETM	X					
54. SETO		X				
55. SETR			X			
56. SFK						
57. SFM	X	X				
58. SFO		X				
59. SFR			X			
60. SKFM		X			X	X
61. SM			X		X	X
62. SMFM	X	X			X	X
63. SMK					X	X
64. SMM	X	X			X	X
65. SMO		X			X	X
66. SO			X		X	X
67. SOFM		X			X	X
68. SOK					X	X
69. SOM	X	X			X	X
70. SOO		X			X	X
71. SRFM		X	X		X	X
72. SRK						X
73. SRM	X	X				X
74. SRO		X				X
75. USE						

3.5 Summary of Errors

	1 1 1 1 1 1 1 1 1 1 2 - 9																		
	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9
1. AMS	X	X	X			X		X	X	X									X
2. AMSK	X	X				X	X	X		X									X
3. AMSM	X	X				X	X	X	X	X									X
4. AMSO	X	X				X		X		X									X
5. CARD	X	X						X											
6. CLOS	X																		
7. CMK	X	X						X											
8. CMM	X	X							X										X
9. CMO	X	X						X											X
10. CMR	X	X	X							X									X
11. CMT	X	X							X										
12. COK	X	X						X											
13. COM	X	X							X										X
14. COO	X	X						X											X
15. COR	X	X	X							X									X
16. COT	X	X						X											
17. CR	X		X														X		
18. CRS	X		X									X					X		
19. DBST	X																		
20. DELS	X	X						X											
21. DR	X		X							X									X
22. DRK	X							X											
23. DRM	X		X					X	X										X
24. DRO	X	X						X											X
25. FFM	X	X						X											X
26. FLM	X	X						X											X
27. FMSK	X	X						X								X			X
28. FNM	X	X						X											X
29. FNSK	X	X						X	X							X			X
30. FPM	X	X						X											X
31. GETK	X						X					X							
32. GETM	X	X						X	X			X							X
33. GETO	X	X						X				X							X
34. GETR	X		X							X		X							X

		1	2	3	4	5	6	7	8	9	1	1	1	1	1	1	1	1	1	1	2	-	9
		1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	9
35. GFK	X			X				X						X									
36. GFM	X	X		X					X	X				X									X
37. GFO	X	X		X					X					X									X
38. GFR	X		X	X							X			X									X
39. GKM	X	X																					
40. GKO	X	X																					
41. GKR	X		X																				
42. GTK	X							X															
43. GTM	X	X							X	X													X
44. GTO	X	X							X														X
45. ICFK	X			X				X															
46. ICFM	X	X		X						X													X
47. icfo	x	x		x					x														x
48. ICFR	X		X	X							X												X
49. OPEN															X	X	X				X		
50. RM	X	X							X	X												X	X
51. RS	X	X							X														X
52. SETK	X							X															
53. SETM	X	X								X													X
54. SETO	X	X							X														X
55. SETR	X		X								X												X
56. SFK	X			X				X						X				X					
57. SFM	X	X		X					X	X				X				X					X
58. SFO	X	X		X					X					X				X					X
59. SFR	X		X	X							X			X				X					X
60. SKFM	X	X			X			X	X										X			X	
61. SM	X	X	X			X					X		X										X
62. SMFM	X	X			X				X	X									X			X	
63. SMK	X	X						X	X					X									
64. SMM	X	X						X		X	X			X									X
65. SMO	X	X						X		X				X									X
66. SO	X	X	X		X						X											X	X
67. SOFM	X	X			X				X										X			X	
68. SOK	X	X			X			X														X	X
69. SOM	X	X			X				X	X												X	X
70. SOO	X	X			X				X													X	X
71. SRFM	X	X	X		X				X		X								X			X	
72. SRK	X							X															
73. SRM	X	X							X	X													
74. SRO	X	X							X														
75. USE	X																				X		

APPENDIX B
STRING MANIPULATION SUBROUTINE DESCRIPTIONS

These Subroutine descriptions are largely taken from [4].

Subroutine BLNK

Description:

Blank out a buffer

Definition:

CALL BLNK(BUF, BPTR, LEN)

Arguments:

BUF(output) buffer to be blanked
BPTR -> beginning index to start blanking
LEN -> number of characters to blank out

Discussion:

BLNK blanks out the buffer starting at BPTR for length LEN.

Subroutine CSTOI

Description:

Convert Character string to integer

Definition:

CALL CSTOI(CH, PTR, LEN, NUM)

Arguments:

CH -> Character form of number
PTR -> index into ch where number starts
LEN -> length of the number

NUM <- binary value of number

Discussion:

CSTOI converts the number beginning at character position PTR in CH for length LEN into a binary number NUM.

Integer Function ICNC1

Description:

Find the first character that is not the one specified.

Definition:

$I = ICNC1(BUF, BPTR, CH, LEN)$

Arguments:

BUF → integer array containing character string
BPTR → integer
CH → single character e.g. "A" or 1HA
LEN → integer

Discussion:

ICNC1 looks at BUF starting at position BPTR up to LEN and returns the index into BUF of the first character that is not the same as CH. IF all characters are the same as CH, or if LEN is less than BPTR, ICNC1 returns 0. Character are numbered from the left starting at one.

Example:

ICNC1("ABCDE",3,"C",5) would return 4 since the forth character in the character string "ABCDE" is the first character (when you begin looking at position three) not equal to "C".
ICNC1("abbb",2,"C",4) would return 0.

Integer Function ICSC1

Description:

Find the first character that equals the one provided.

Definition:

$I = ICSC1(BUF, BPTR, CH, LEN)$

Arguments:

BUF → integer array containing character string

BPTR → integer

CH → single character e.g. "A" or 1HA

LEN → integer

Discussion:

ICSC1 looks at BUF starting at position BPTR up to LEN and returns the index into BUF of the first character that is the same as CH. If all characters are not equal to CH, or if LEN is less than BPTR, ICSC1 returns 0. Characters are numbered from the left starting at one.

Example:

ICSC1 ("ABCD" , 1, "C", 4) would return 3 since the third character of the BUF is a "c".

Integer Function ISCOMP

Description:

Character string compare

Definition:

I = ISCOMP (STR1,S1,STR2,S2, LEN)

Arguments:

STR1 -> character string
S1 -> integer
STR2 -> character string
S2 -> integer
LEN -> integer

Discussion:

ISCOMP compares the string STR1 starting at S1 with STR2 starting at S2, for a length LEN. ISCOMP returns a value denoting their equivalence or their relative alphabetic ordering. the value returned is :

-1 if STR1 < STR2
0 if STR1 = STR2
+1 if STR1 > STR2
0 if LEN < 1

Example:

ISCOMP ("AAAA",1,"ZZZZ",1,4) returns +1
ISCOMP ("AAAA",2,"BAACCD",2,2) returns 0

Subroutine ITOC

Description:

Convert integer character

Definition:

CALL ITOC(NUM,BUF,PTR,LEN)

Arguments:

NUM -> integer
BUF <- integer array
PTR -> integer
LEN -> integer

Discussion:

ITOC converts the integer NUM into a right-justified hollerith of length LEN and puts the hollerith into BUF starting at PTR.

Example:

I = 5
CALL ITOC (I,BUF,2,5)
would make the second through sixth characters of BUF become "bbbb5"
where b = 1 blank.

Subroutine ITOCS

Description:

Convert a number to a character string

Definition:

```
CALL ITOCS( NUM, CH, PTR, LEN)
```

Arguments:

NUM -> The number to be converted to character form
CH -> Place to put character representation
PTR -> Index into ch where character representation starts

LEN <- Length of character form returned

Discussion:

ITOCs converts a number to a character string using only as many characters as is necessary.

Subroutine SMOVE

Description:

Move character strings

Definition:

CALL SMOVE (TO,TPOS,FROM,FPOS,LEN)

Arguments:

TO <- character string e.g. integer array
TPOS -> integer
FROM -> character string e.g. integer array
FPOS -> integer
LEN -> integer

Discussion:

SMOVE replaces LEN number of characters in TO, starting with the character at TPOS. Characters are moved from FROM starting at FPOS.

INDEX

AMS, 33
AMSK, 34
AMSM, 35
AMSO, 36

BLNK, 120

CARD, 37
CLOS, 38
CMK, 39
CMM, 40
CMO, 41
CMR, 42
CMT, 43
COK, 44
COM, 45
COO, 46
COR, 47
COT, 48
CR, 49
CRS, 50
CSTOI, 121

DBST, 51
DELS, 52
DEND, 53
DINT, 54
DR, 55
DRK, 56
DRM, 58
DRO, 58

FFM, 59
FLM, 60
FMSK, 61
FNM, 62
FNSK, 63
FPM, 64

GETCDB, 65
GETK, 66
GETM, 67
GETO, 68
GETR, 69
GFK, 70
GFM, 71
GFO, 72
GFR, 73

GKM, 74
GKO, 75
GKR, 76
GTK, 77
GTM, 78
GTO, 79

ICFK, 80
ICFM, 81
ICFO, 82
ICFR, 83
ICNC1, 122
ICSC1, 123
IDOP, 84
ISCOMP, 124
ITOC, 125
ITOCS, 126

RM, 85
RS, 86

SETCDB, 87
SETK, 88
SETM, 89
SETO, 90
SETR, 91
SFK, 92
SFM, 93
SFO, 94
SFR, 95
SKFM, 96
SM, 97
SMFM, 98
SMK, 99
SMM, 100
SMO, 101
SMOVE, 127
SO, 102
SOFM, 103
SOK, 104
SOM, 105
SOO, 106
SRFM, 107
SRK, 108
SRM, 109
SRO, 110

USE, 111

NOTICE

This report was prepared as an account of work sponsored by the United States Government. Neither the United States nor the United States Energy Research & Development Administration, nor any of their employees, nor any of their contractors, subcontractors, or their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness or usefulness of any information, apparatus, product or process disclosed, or represents that its use would not infringe privately-owned rights.

NOTICE

Reference to a company or product name does not imply approval or recommendation of the product by the University of California or the U.S. Energy Research & Development Administration to the exclusion of others that may be suitable.

Printed in the United States of America
Available from
National Technical Information Service
U.S. Department of Commerce
5285 Port Royal Road
Springfield, VA 22161
Price: Printed Copy \$; Microfiche \$3.00

Page Range	Domestic Price	Page Range	Domestic Price
001-025	\$ 3.50	326-350	10.00
026-050	4.00	351-375	10.50
051-075	4.50	376-400	10.75
076-100	5.00	401-425	11.00
101-125	5.50	426-450	11.75
126-150	6.00	451-475	12.00
151-175	6.75	476-500	12.50
176-200	7.50	501-525	12.75
201-225	7.75	526-550	13.00
226-250	8.00	551-575	13.50
251-275	9.00	576-600	13.75
276-300	9.25	601-up	*
301-325	9.75		

*Add \$2.50 for each additional 100 page increment from 601 to 1,000 pages;
add \$4.50 for each additional 100 page increment over 1,000 pages.