HEWLETT PACKARD

2100 series computers

using IMAGE/2100 data bases

HP MANUAL PART NO. 02100-90139
MICROFICHE PART NO. 02100-90197

October 1973

Printing History

First Edition . . . . . October 1973

HP Computer Museum
www.hpmuseum.net

For research and education purposes only.

Preface

This document describes the Data Base Management System (DBMS) for the Hewlett-Packard Information Management System for HP 2100 Computers (IMAGE/2100). Other documents of interest to the user are:

- Designing IMAGE/2100 Data Bases (02100-90138)
- QUERY: On-Line Access to IMAGE/2100 Data Bases (02100-90142)
- DOS III: Disc Operating System (02100-90136)
- Decimal String Arithmetic Routines (02100-90140)

Section I of this manual describes the use and structure of DBMS, including the common work areas. Section II details use of the Data Base Open Subroutine (DBOPN). The Data Base Close Subroutine (DBCLS) is explained in Section III. Section IV shows the Data Base Read Subroutine (DBGET), while Section V explains the Data Base Update Subroutine (DBUPD). The Data Base Write Subroutine (DBPUT) is outlined in Section VI. The Data Base Delete Subroutine (DBDEL) is detailed in Section VII. The Data Base Find Subroutine (DBEND) is explained in Section VIII, while Section IX explains the Data Base Information Subroutine (DBINF). Appendix A lists all of the error messages that may occur during DBMS execution.

Contents

FIGURES

TABLES

SECTION I
Introduction

The Data Base Management System (DBMS) is a part of the Information Management System for HP 2100 Computers (IMAGE/2100). DBMS consists of eight subroutines designed to access data stored in an IMAGE/2100 data base. The structure of the data base (known as the root file) is created by the IMAGE/2100 Data Base Definition System (DBDS).

The subroutines perform such functions as opening the data base for access by verifying that the user possesses the proper security codes, reading data entries, writing new data entries into the data base, updating entries by changing the values of non-key data items in data entries already present in the data base, and other functions necessary to access and manipulate information in the data base.

The names and functions of the eight subroutines are:

DBOPN	Transfers the root file into a DBMS work area known as the Run Table, and sets various bits in the table to inhibit access of specific data items in the data base, depending upon the level access word specified by the user in the DBOPN calling sequence. DBOPN also specifies which operation types are allowed in general for access of the entire data base, depending upon the mode of access number specified by the user in the DBOPN calling sequence.
DBCLS	Closes or purges a data base to ensure the integrity of the data base root file and to prevent accidental or purposeful access of the data base.
DBGET	Reads data entries from data sets in the data base, and opens the data sets prior to access if necessary.
DBUPD	Updates currently existing data entries in the data sets by changing the value of non-key data items within the data entries.
DBPUT	Enters new data entries into data sets of the data base.
DBDEL	Deletes data entries from data sets in the data base.
DBFND	Initializes information in the DBMS Run Table (described below) to implement chain accesses to a data set of the data base. (See Section IV, Data Base Read Subroutine.)
DBINF	Returns information about the data base to the user. The information can include the length (in words) of data entries in data sets, type and length of data items, etc.

Along with the eight subroutines described above, DBMS also maintains a work area which includes a Run Table and buffer space. The Run Table contains information about each data set including the relative record address of the last entry accessed in a data set. When a data base is opened using DBOPN, the Run Table includes a copy of the data base root file, with the appropriate bits set to identify which data items in the data base may be accessed. The DBMS routines access the Run Table to locate desired data entries during their execution.

The data base files exist as a group of records independent of data entries. Data records in a data set not containing data entries initially contain all zeros. It is possible under some conditions to access an empty record. DBMS so informs the user through status information sent to the user in the user status area defined for each subroutine.

USING DBMS

Execution of the DBMS subroutines occurs in response to calling sequences appearing in a user source language program (either HP FORTRAN or HP 2100 Assembly Language). When a calling sequence is encountered during execution of a user program, the DBMS subroutine mentioned in the calling sequence executes according to the value of parameters in the calling sequence, and returns information to user areas defined in the parameter list. The number, type, and meaning of the parameters depends upon the subroutine and is described in detail in the section of the manual devoted to the particular subroutine.

For FORTRAN, the calling sequence used is:

CALL name(param,param, ... ,param)

where:

name is the name of the subroutine.

param is an identifier name (name of an array or variable), corresponding in form to the form of the host language. It is the user's responsibility to define and use the parameters in a manner corresponding to the rules of usage of the host language.

For Assembly Language, the calling sequence is:

```
JSB   name
DEF   *+n+1
DEF   param
DEF   param
  .
  .
  .
DEF   param
```

where:

n is the number of parameters in the calling sequence.

name is the name of the subroutine.

param is the same as defined previously for FORTRAN.

DBMS WORK AREA

DBMS subroutines require a work area in memory to hold pointers to data in the data base and to provide temporary buffer space for the transfer of data entries to and from the data base. These blocks of memory are reserved for the sole use of DBMS subroutines and must never be accessed by user programs.

Each user program using DBMS subroutines must reserve work area space in common with all other user programs and the DBMS subroutines. The actual name of each area is not important; however, the areas must be defined with the same size and in the same order as shown below.

The work area definition in a FORTRAN program consists of a COMMON statement. For example,

```
COMMON ISORT(1024),ISLGT(1),IRUN(1175),IOPN(128)
```

The work area definition in an Assembly Language program consists of a COM statement. For example,

```
COM SORT(1024),SLGT(1),RUN(1175),OPEN(128)
```

DBMS EXAMPLE

Figure 1-1 below is a program written in FORTRAN IV which uses the COMMON statement to define the DBMS work area. The program itself opens the data base (by asking for the proper data base name, security code, level and mode of access), reads and prints an entry from a master data set based upon the value of the data entry key item, requests another key item value from the user, and when finished, closes the data base.

```
      PROGRAM U100
C     DESCRIPTION OF FIELDS
C          ISORT      1024   EFMP TEMP. BUFFER
C          ISLGT      1      IMAGE COUNTER
C          IRUN       1175   ROOT FILE RUN TABLE
C          IOPN       128    EFMP OPEN FILE TABLE
C
C          IBASE      3      DATA BASE (ROOT FILE) NAME
C          ILEVL      3      IMAGE LEVEL ACCESS WORD
C          ISCOD      1      DATA BASE SECURITY CODE
C          IMODE      1      IMAGE MODE OF ACCESS CODE
C          ISTAT      4      STATUS WORD
C          IDSET      3      DATA SET NAME(SAME AS EFMP FILE NAME)
C          IBUF       31     USER DATA RECORD BUFFER
C          IARG       6      KEY DATA ITEM VALUE
C          JANS       2      ANSWER TO QUESTION
C          JTEST      1      TEST VALUE FOR ANSWER "YE" (S)
```

Figure 1-1. Sample Program

```
      COMMON   ISORT(1024),ISLGT(1),IRUN(1175),IOPN(128)
      COMMON   IBASE(3),ILEVL(3),ISCOD(1),IMODE(1),ISTAT(4)
      COMMON   IDSET(3),IBUF(31),IARG(6),JANS(2)
      DATA JTEST/2HYE/
C     OPEN THE DATA BASE
100   WRITE(1,110)
      READ(1,120)IBASE
      WRITE(1,130)
      READ(1,120)ILEVL
      WRITE(1,140)
      READ(1,*)ISCOD(1)
      WRITE(1,150)
      READ(1,*)IMODE(1)
110   FORMAT("DATA BASE=?")
120   FORMAT(3A2)
130   FORMAT("LEVEL=?")
140   FORMAT("SECURITY CODE=?")
150   FORMAT("MODE=?")
      CALL DBOPN(IBASE,ILEVL,ISCOD,IMODE,ISTAT)
      IF (ISTAT(1) .EQ. 0 ) GO TO 200
      WRITE (1,160)ISTAT(1)
160   FORMAT ("ERROR" ,I5, "ON READ")
      PAUSE
C     READ ROUTINE
C          1.  GET DATA SET NAME
C          2.  GET KEY ITEM VALUE
C          3.  READ THE RECORD OF THE MANUAL MASTER
C          4.  WRITE THE RECORD ON THE TERMINAL
C          5.  ASK FOR ANOTHER RECORD KEY ITEM VALUE
C          6.  READ ANSWER
C          7.  DO 1,2,3,4,5,6 OR CLOSE THE DATA BASE
200   WRITE(1,210)
      READ(1,220)IDSET
      WRITE(1,230)
      READ(1,240)IARG
210   FORMAT("DATA SET=?")
220   FORMAT(3A2)
230   FORMAT("ENTER KEY ITEM VALUE")
240   FORMAT(6A2)
      IMODE = 4
      CALL DBGET(IDSET,IMODE,ISTAT,IBUF,IARG)
      IF (ISTAT(1) .EQ. 0) GO TO 260
      WRITE(1,250)ISTAT(1)
250   FORMAT("ERROR" ,I5, "ON OPEN")
      CALL DBCLS(0,0)
      PAUSE
260   CALL EXEC(2,1B,IBUF(4),28)
      WRITE(1,280)
```

Figure 1-1. Sample Program (cont.)

```
280     FORMAT("MORE?")
        READ(1,290)JANS
290     FORMAT(2A2)
        IF (JANS(1) .EQ. JTEST) GO TO 200
        CALL DBCLS(0,0)
        END
```

Figure 1-1. Sample Program (cont.)

SECTION II

DBOPN - Data Base Open

DBOPN opens a data base to a user by transferring a copy of the root file into the DBMS Run Table. The DBOPN calling sequence contains the name of the data base, (name of the root file), a level access word, data base security code, and a mode of access code. DBOPN verifies that the data base name is the root file of an IMAGE/2100 data base, that the security code is proper for that data base. DBOPN then sets flags in the Run Table to identify which data items may be accessed by the user according to the level access word given in the calling sequence, and checks that the mode of access is valid.

If DBOPN opens the data base successfully, the first word of the status word (ISTAT in the calling sequence) contains a zero, and the second word contains a level number corresponding to the level access word given in the user calling sequence. The number is binary between 0 and 15. Level access words and numbers are defined during creation of the data base root file by the Data Base Definition System. If the data base was not successfully opened, DBOPN stores a non-zero integer in the first word of ISTAT. Consult Appendix A, *DBMS Error Messages* for the meaning of the error number.

If the data base being opened does not contain any level words and numbers in the root file, then the user can specify any value for the level access word in the user calling sequence (ILEVL) and DBOPN returns 15 in the second word of ISTAT.

The user is cautioned never to access an IMAGE/2100 data base through any means except the DBMS or QUERY/2100 systems. The user must never attempt to access a data base through Extended File Management Package (EFMP) or damage to the data base may result. User programs may access data from an IMAGE/2100 data base through DBMS routines, and then store that data on other EFMP files only if the user closes the data base (using the DBCLS routine) prior to using EFMP to manipulate the accessed data entries.

CALLING SEQUENCES

The FORTRAN calling sequence of DBOPN is:

```
CALL DBOPN(IBASE,ILEVL,ISCOD,IMODE,ISTAT)
```

The Assembly Language calling sequence is:

```
JSB   DBOPN
DEF   *+6
DEF   IBASE
DEF   ILEVL
DEF   ISCOD
DEF   IMODE
DEF   ISTAT
```

DBOPN PARAMETERS

The following parameters appear in the DBOPN calling sequence:

IBASE A one-dimensional integer array at least three computer words long, containing the data base (root file) name. The name must be stored in IBASE in ASCII format, two characters per computer word. The data base name must be from 1 to 6 characters long. If the name is less than six characters, the remaining characters must be blank.

ILEVL A one-dimensional integer array at least three computer words long, containing the level access word choosen by the user to access the data base. The name must be stored in ILEVL in ASCII format, two characters per computer word. The access level word must be from 1 to 6 characters long, and if less than six characters, the remaining characters must be blank.

ISCOD Is an integer variable one computer word long, containing the data base security code as defined in the data base schema stored in the root file.

IMODE Is an integer variable one computer word long, containing the code for the mode of operation desired by the user. Table 2-1 below shows the allowed values for IMODE and their meaning.

ISTAT A one-dimensional integer array at least two computer words long, used by DBOPN to signal completion or error conditions to the user. If the data base is opened successfully, DBOPN stores a zero in the first word of the ISTAT and the level number corresponding to the level access word given in the calling sequence in the second word of ISTAT. If the data base is not opened due to some error condition, DBOPN stores a non-zero integer in the first word of ISTAT. Consult Appendix A *DBMS Error Messages* for the meaning of the error number.

Table 2-1. IMODE Values

Value	Meaning
1	Read values from the data base only. Do not add or modify data in the data base.
2	Update data entries already existing in the data base. Data entries cannot be added or deleted from the data base, although non-key data item values can be changed.
3	Add, delete, or modify (update) data entries in the data base, including modifying the values of key items in a data entry. A user calling DBOPN with IMODE=3 must also specify a level access word as the contents of ILEVL corresponding to the level access number of 15 (as defined in the data base schema stored in the root file). If no levels are defined in the root file, the user may specify any value as the contents of ILEVL, and MODE 3 will be granted.
5	Initializes the data base to enable usage of all data sets. Once the data base has been defined and the data base root file created and stored on the disc (by the IMAGE/2100 Data Base Definition System), the user must open the data base with IMODE=5 the very first time the data base is accessed, and prior to entering any data entries into the data base. IMODE=5 allows all of the features of IMODE=3. Once the data base has been opened with IMODE=5, the user must never use IMODE=5 again unless he first purges the data base using DBCLS in purge mode.

DBOPN EXAMPLE

Figure 2-1 below shows part of a data base schema defining an IMAGE/2100 data base. Figure 2-2 below shows the calling sequence (in FORTRAN) necessary to open the data base described in Figure 2-1.

```
EFMP PACK ID PN001;100; ← security code
BEGIN DATA BASE CABCO; ← data base
LEVELS:
        1   CLERK;
        4   SUPERV;
        8   MANAGE;
        15  OWNER; ← level access word
    .
    .
    .
    .
    .
END.
```

Figure 2-1. Data Base Schema

```
      DIMENSION IBASE(6),ILEVL(6),ISTAT(2)
      READ(6,10)IBASE                    (Read the name of the data base
                                          rootfile [CABCO])
      READ(6,10)ILEVL                    (Read level word [OWNER])
10    FORMAT(3A2)
      ISCOD = 100                        (Assign data base security code.)
      IMODE = 3                          (Assign the desired mode of access.)
      CALL DBOPN(IBASE,ILEVL,ISCOD,IMODE,ISTAT
```

Figure 2-2. Sample Program Opening a Data Base

SECTION III
DBCLS - Data Base Close

DBCLS performs one of two functions:

1. Closes the data base files including the data base root file.
2. Purges (removes) all data sets from the data base, while keeping the data base root file intact.

IMAGE/2100 allows only one open data base at a time. DBCLS accesses the Run Table in the DBMS work area to find the name of the currently open data base. The root file (a copy of which is stored in the Run Table) contains the name of the files containing actual data entries.

Users attempting to purge the data base must have previously opened the data base using the level 15 access word as defined in the data base root file. If the data base schema (stored in the root file) does not define any level words for the data base, then no restriction is made for purging the data sets. DBCLS removes all data sets from the data base, but keeps the data base root file intact.

To access a data base once it has been purged, the user must open the data base (using the DBOPN routine described in Section II) with the mode of access parameter equal to 5. Subsequent openings of the data base can then occur using a lower-numbered mode of access number.

CALLING SEQUENCES

The FORTRAN calling sequence for DBCLS is:

```
CALL DBCLS (IMODE,ISTAT)
```

The Assembly Language calling sequence is:

```
JSB  DBCLS
DEF  *+3
DEF  IMODE
DEF  ISTAT
```

DBCLS PARAMETERS

The following parameters appear in the DBCLS calling sequence:

IMODE — Is an integer variable one computer word long, containing either the value 1 or 0.

- IMODE = 0 indicates that DBCLS is to close the data base files (including the data base root file), but leave the data base entries intact.
- IMODE = 1 indicates that DBCLS is to purge the data entries stored in the data base.

ISTAT — Is an integer array at least one computer word long, used by DBCLS to signal completion of closing (or purging) or to signal an error condition. If the data base was successfully closed (or purged), DBCLS stores a zero in the first word of ISTAT. If an error condition occurs, DBCLS stores the appropriate error number in ISTAT. Consult Appendix A, *DBMS Error Messages* for the meaning of the error number.

DBCLS EXAMPLE

Figure 3-1 below shows an example of a FORTRAN calling sequence used to close a data base.

```
DIMENSION ISTAT(1)          User defines the array for reporting status.
IMODE = 0                   (User sets IMODE = 0 to signify closing of data base without
                            purging.)
CALL DBCLS(IMODE,ISTAT)     (User closes DATA BASE.)
```

Figure 3-1. Sample DBCLS Execution

SECTION IV
DBGET - Data Base Read

DBGET reads data entries from various sets in the data base by locating the desired data set entry and loading that entry into a user storage buffer defined in the DBGET calling sequence parameter list. DBGET locates desired data entries in one of four different ways: through a chained read, serial read, directed read, or keyed read.

- Chain Read

  A chain read occurs on detail data sets only, and requires that the user call the DBFND routine prior to calling DBGET. DBFND accepts the value of a key item as an input parameter from the user and locates the data entry in the master data set which contains that key item value. DBFND then loads the pointer to the first data entry in the detail data set for the chain defined by the key item value into the DBMS work area.

  When DBGET is called for a chain read, DBGET references the Run Table in the DBMS work area for the detail data set and reads the pointer to the entry in the chain. After loading that entry into the user buffer, DBGET updates the Run Table to point to the next entry in the chain.

  If the user wishes to access the next entry in the same chain, he calls DBGET again without first calling DBFND. However, to access entries from a different chain, the user must first call DBFND with the appropriate key item value prior to calling DBGET for a chained read.

- Serial Read

  For a serial read, DBGET references the Run Table for the data set specified by the user in the DBMS work area to find the last accessed data entry in the data set. If the data set has not been accessed since the data base was opened, the relative record address is zero. DBGET searches the next relative record address and continues to search one record at a time until a non-empty record containing a data entry is discovered, or until the end of the data set occurs. The data entry is stored in the user buffer.

- Directed Read

  DBGET performs a directed read by locating the relative record location for the data set supplied by the user in the DBGET calling sequence. If the record location contains a data entry (is non-empty), DBGET loads that entry into the user buffer. If the record address contains no data entry, DBGET returns an error code to the user status word (ISTAT). If the user specifies a relative record address of zero, DBGET resets the current access record counter (which specifies the last record accessed in the data set) to zero without performing a read.

- Keyed Read

  DBGET performs a keyed read to locate a data entry in a master data set whose key item value matches the key item value specified by the user in the DBGET calling sequence. DBGET searches the master data set specified and loads the data entry with the matching key item value into the user buffer. If DBGET cannot find an entry in the master data set with a matching key item, then DBGET loads the appropriate error number into the status word (ISTAT).

Data buffers defined by the user for purposes of storing data entries read by DBGET from the data base must be long enough to contain not only the data in the data entry but the *media record* part of the data entry as well. The media record consists of pointer information, and appears in front of the actual data entry containing data values.

DBGET loads data item values into the user buffer if and only if the read level of that item (as defined in the data schema) is equal to or less than the access level corresponding to the access level word specified by the user when the data base was opened using the DBOPN routine. If the read level number is greater than the user's specified level, DBGET fills the data space in the user buffer with all zeros in place of the data item value.

DBGET CALLING SEQUENCE

The FORTRAN calling sequence for DBGET is:

```
CALL DBGET (IDSET,IMODE,ISTAT,IBUF,IARG)
```

The Assembly Language calling sequence is:

```
JSB   DBGET
DEF   *+6
DEF   IDSET
DEF   IMODE
DEF   ISTAT
DEF   IBUF
DEF   IARG
```

DBGET PARAMETERS

The following parameters appear in the DBGET calling sequence:

IDSET — Is a one-dimensional integer array at least three words long containing the name of a master or detail data set. The names must be in ASCII format, two characters per word. The name must be from one to six characters long, and if less than six characters the remaining characters must be blank.

IMODE Is an integer variable one computer word long, containing an integer value ranging from 1 to 4, indicating the reading method. The values and their meaning appear in Table 3-1 below:

Table 3-1. IMODE Values for DBGET

Value	Meaning
1	Perform a chain read in the data set specified by IDSET. The value of IARG (defined below) is ignored. The next data entry in the chain to be read by DBGET is specified in the Run Table for the data set.
2	Perform a serial read in the data set specified by IDSET. The value of IARG is ignored. The record location to be read is specified in the Run Table for the data set. DBGET reads record addresses until a data entry is found or until the end of the data set is reached.
3	Perform a directed read in the data set specified by IDSET. The value of IARG is treated as a positive integer specifying the relative record number (the first record is number one) in the data set where DBGET is to seek a data entry. If the record location contains a data entry, it is loaded into the buffer defined by the value of IBUF (defined below). If the record address is empty, DBGET returns an error code in ISTAT (as described below).
4	Perform a keyed read in the master data set specified by IDSET. The value in IARG is treated as a key item value. DBGET either retrieves the data entry containing the key item value, or returns an error code to ISTAT if the key item value does not exist in the master data set.

ISTAT — A one-dimensional integer array at least four computer words long used by DBGET to signal completion of a read or to report an error condition. Table 3-2 describes the four words and their meaning:

Table 3-2. DBGET Status Word (ISTAT)

Word 1 (Condition Word) For all four types of read (IMODE = 1,2,3, or 4), DBGET loads a zero into the Condition Word to signify successful completion of the read function. If an error occurs, DBGET loads an error code into the Condition Word. Consult Appendix A, *DBMS Error Messages.*
Word 2 (Record Address) Contains the record address read by DBGET. The record address consists of a relative record number of the data set accessed. The first record address is a number 1. One logical data entry (including the media record containing IMAGE/2100 data pointers) is stored in one record.
Word 3 (Chain Length) DBGET loads zero into Word 3 for IMODE = 2, 3, and 4. For a chain read, (IMODE = 1) DBGET stores the number of entries in the currently-accessed chain. The DBFND routine locates the head of the chain prior to executing DBGET.
Word 4 (Forward Address) DBGET loads zero into Word 4 for IMODE = 2, 3, and 4. For a chain read (IMODE = 1), DBGET stores the relative record address of the next record in the chain following the currently-accessed record.

IBUF — A one-dimensional integer array in which DBGET stores the data entry retrieved from the data set. The array must be large enough to contain both the media record and the data record parts of the data entry. Media and data record length for any data set can be determined through the DBINF routine (see Section IX) or a copy of the processed schema. (See *Designing IMAGE/2100 Data Bases*, 02100-90138.)

IARG — Is a one-dimensional integer array used in one of two ways: for a directed read or a keyed read (IMODE = 3 or IMODE = 4 respectively). For a directed read, the user must place a relative record number in the first word of IARG. DBGET reads the data entry at the record address specified. For a keyed read, the user stores the key item value used as a search value into as many words of IARG as necessary. DBGET searches the desired master data set to find the data entry with a key item value equal to the value stored in IARG.

DBGET ignores the contents of IARG for a chain or serial record (IMODE = 1 or IMODE = 2 respectively.)

DBGET EXAMPLE

Figure 4-1, below, indicates an example of the DBGET routine used to perform a chain read on a detail data set called PTIENT (PATIENT). The program attempts to find all detail data entries containing the value BLUE for the key item named EYE-C (eye color), and prints the entire data entry (including media record pointers) on the system terminal device.

First, the program opens the data base using DBOPN. Then, the program uses DBFND to access the manual master data set called EYES to find a master data entry with the value BLUE for the key item named C-EYES. DBFND loads information about the first detail data entry in the chain consisting of EYE-C equal in value to BLUE. The program also uses the number of entries in the chain (reported by DBFND) as a counter for a DO loop and reads and prints the detail data entries in the chain. The program then closes the data base.

```
C       THIS PROGRAM PERFORMS A CHAIN READ.
C       DECLARE DBMS COMMON WORK AREA TO SET UP RUN TABLES
        COMMON ISORT(1024),ISLGT(1),IRUN(1175),IOPN(128)
C       DECLARE NECESSARY PARAMETERS FOR DBGET,DBOPN,DBFND
        DIMENSION ISET(3),IMODE(1),ISTAT(4),IBUF(25),IARG(3)
        DIMENSION ID1(3),ILEVL(3),ISCOD(1),IM1(1),ISTT1(2)
        DIMENSION ID2(3),ISTT2(4),IPATH(3),IARG1(3),ID3(3)
C       INITIALIZE THE PARAMETERS USING DATA STATEMENTS
        DATA ID1/2HCL,2HIE,2HNT/,ILEVL/2HOW,2HNE,2HR  /
        DATA ISCOD/1234/,IM1/3/,ISTT1/2*0/
        DATA ID2/2HPT,2HIE,2HNT/,ISTT2/4*0/
        DATA IPATH/2HEY,2HE-,2HC  /,IARG1/2HBL,2HUE,2H    /
        DATA ISET/2HPT,2HIE,2HNT/,IMODE/1/,ISTAT/4*0/
        DATA IBUF/25*0/,IARG/3*0/
        DATA ID3/2HEY,2HES,2H    /

C       OPEN THE DATA BASE USING DBOPN
        CALL DBOPN(ID1,ILEVL,ISCOD,IM1,ISTT1)
        IF (ISTT1(1)  .EQ.  0) GO TO 50
        WRITE(1,*) ISTT2(1)
        STOP 1

C       PREPARE FOR A CHAIN READ BY CALLING DBFND

50      CALL DBFND(ISTT2,ID2,IPATH,IARG1)
        IF(ISTT2(1)  .EQ.  0) GO TO 60
        WRITE (1,*)ISTT2(1)
        STOP 2

C       PERFORM THE CHAIN READ. ISTT2(3) CONTAINS THE NUMBER
C       OF ENTRIES IN THE CHAIN AS RETURNED BY DBFND
```

Figure 4-1. DBGET Example

```
60      DO 70 I=1,ISTT2(3)
        CALL DBGET(ISET,IMODE,ISTAT,IBUF,IARG)
        IF(ISTAT(1) .EQ. 0) GO TO 62
        WRITE(1,*)ISTAT(1)
        STOP 3
C       WRITE ENTIRE RECORD (MEDIA+DATA)
62      WRITE(1,65)IBUF

C       THE FORMAT STATEMENT BELOW PRINTS THE MEDIA RECORD
C       AS THREE INTEGER VALUES AND THE DATA RECORD AS
C       A CONTIGUOUS CHARACTER STRING

65      FORMAT(I,X,I,X,I,X,22A2)
70      CONTINUE

C       CLOSE THE DATA BASE

        IMODE=0
        CALL DBCLS(IMODE,ISTAT)

        IF (ISTAT(1) .EQ. 0) GO TO 80
        WRITE(1,*)ISTAT(1)
        STOP 4

80      CONTINUE
        END.
```

Figure 4-1. DBGET Example (cont.)

SECTION V
DBUPD - Data Base Update

DBUPD is used to change the values of individual data items within an entry of a data set. To do so, DBUPD accesses the DBMS Run Table for the data set specified by the user and reads the last accessed record reported in the Run Table. The last record read for each data set is updated (in the Run Table) by the DBGET routine whenever it is executed. Consequently, the user must call DBGET to store the desired record address in the Run Table sometime prior to calling DBUPD.

Once DBUPD is called, the routine reads the record specified in the Run Table, modifies the data entries in the entry, stores a copy of the updated entry in the buffer specified by the user in the DBUPD calling sequence, and places the modified entry back into the data set. DBUPD handles both the media record and the data record part of the data entry; the buffer length specified by the user must be long enough to accomodate both the media and the data record.

The data items to be modified in the entry are specified by number. To discover these numbers prior to calling DBUPD, use the DBINF routine (see Section IX). In addition, each data item in the data set has associated with it a read and write level. (See *Designing IMAGE/2100 Data Bases*, 02100-90138.) Any data item specified for updating by DBUPD must have a write level equal to or less than the level specified by the user when the data base was opened using the DBOPN routine. If the user specifies a data item to be updated which has a write level higher than the user level, no updating takes place at all for any item in the entry, and DBUPD loads an error code into the first word of the status area for the routine. Under no circumstances can a key item value be changed. Any attempt to do so results in an error.

DBUPD can be invoked only if the mode of access to the data base specified by the user in the DBOPN calling sequence is equal to 2, 3, or 5. If the user attempts to invoke DBUPD with a mode of access equal to 1 for the data base, DBUPD returns an error code in the first word of the status area for the routine.

DBUPD CALLING SEQUENCES

The FORTRAN calling sequence for DBUPD is:

```
CALL DBUPD(IDSET,ISTAT,INBR,IVALU,IBUF)
```

The Assembly Language calling sequence is:

```
JSB  DBUPD
DEF  *+6
DEF  IDSET
DEF  ISTAT
DEF  INBR
DEF  IVALU
DEF  IBUF
```

DBUPD PARAMETERS

The following parameters appear in the DBUPD calling sequence:

IDSET — Is a one-dimensional integer array at least three words long containing the name of a master or detail data set. The name must be in ASCII format, two characters per word. The name must be from one to six characters long, and if less than six characters, the remaining characters must be blanks.

ISTAT — Is a one-dimensional integer array at least one word long used by DBUPD to signal completion of the update or to report an error condition. A zero in the status word indicates completion, while a positive integer indicates an error condition. Consult Appendix A, *DBMS Error Messages* for the meaning of the error code.

INBR — Is a one-dimensional integer array. The first word of the array contains the number of data items to be updated. Remaining words of the array contain the number of the data item to be updated, one data item number per word of the array. The data item number is determined by the order the data item was defined in the data schema for the data base. The first data item defined in the *item part* of the data schema is item number one. (See *Designing IMAGE/2100 Data Bases*, 02100-90138.) The number of individual data items can be discovered programmatically through use of the DBINF routine (see Section IX).

IVALU — Is a one-dimensional integer array containing the values to be placed in the data items specified in INBR. These values are concatenated together in the array. Each value must occupy the full amount of space defined for the data item in the data schema for the data base. For example, a character data item defined as U8 (four words long) requires that the user supply an eight-character value for the array specified in IVALU, perhaps by padding the significant characters with trailing or leading blanks.

IBUF — Is a one-dimensional array large enough to contain the data entry to be updated. The data entry length must include the length of the media record as well as the length of the data record. This information can be obtained through the DBINF routine (see Section IX).

DBUPD EXAMPLE

Figure 5-1, below, shows the data schema used to define a data set in a data base. Figure 5-2 displays a FORTRAN program which uses the Data Base Management Subsystem routines to update a data entry in the data set.

First, the program declares the common area necessary for DBMS operation (not to be accessed by the user program.) The program then defines and initializes all the parameters used in the DBMS routine calling sequences. The data base is opened using DBOPN with the level 15 word (defined in ILEVL). DBGET is then called to store the record number of the desired entry in the Run Table of the data set. DBUPD then updates the data entry with new information. The data base is then closed.

```
EFMP PACK ID PN001;100;
BEGIN DATA BASE EMPLY;

LEVELS:
        1           CLERK;
        4           SUPERV;
        8           MANAGE;
        15          OWNER;

ITEMS:
        L-NAME    U20   (1,8);    <<LAST NAME>>
        F-NAME    U10   (1,8);    <<FIRST NAME>>
        ADDRESS   U20   (1,8);    <<ADDRESS>>
        CITY      U10   (1,8);    <<CITY>>
        STATE     U6    (1,8);    <<STATE>>
        ZIP       U6    (1,8);    <<ZIP CODE>>
        E-NUMB    U4    (15,15);  <<EMPLOYEE NUMBER>>

SETS:
NAME: PERSON,MANUAL;
ENTRY:
        L-NAME,
        F-NAME,
        ADRESS,
        CITY,
        STATE,
        ZIP,
        NUMBER(1);

CAPACITY: 400;
.
.
.
```

Figure 5-1. Data Schema for DBUPD Example

```
C       THIS PROGRAM UPDATES A DATA ENTRY OF A MASTER DATA SET
C       DEFINE THE DBMS COMMON AREA
        COMMON ISORT(1024),ISLGT(1),IRUN(1175),IOPN(128)
C       DEFINE PARAMETERS FOR THE DBMS CALLING SEQUENCES
        DIMENSION IBASE(3),ILEVL(3),ISCOD(1),IMODE(1),ISTAT(2)
        DIMENSION IDSET(3),IM1(1),ISTT1(4),IBUF1(42),IARG1(2)
        DIMENSION ID1(3),ISTT2(1),INBR(8),IVALU(38),IBUF2(42)
C       INITIALIZE PARAMETERS WITH VALUES
        DATA IBASE/2HEM,2HPL,2HY  /,ILEVL/2HOW,2HNE,2HR  /,ISCOD/100/,
     *       IMODE/3/,ISTAT/2*0/,IDSET/2HPE,2HRS,2HON/,IM1/4/,
     *       ISST1/4*0/,IBUF1/42*0/,IARG1/2H12,2H34/,
     *       ID1/2HPE,2HRS,2HON/,ISTT2/0/,INBR/3,3,4,5*0/,
     *       IVALU/2H76,2H2  ,2HE.,2H  C,2HHA,2HRL,2HES,2HTO,2HN  ,2HRD/
C       OPEN THE DATA BASE
        CALL DBOPN(IBASE,ILEVL,ISCOD,IMODE,ISTAT)
C       CHECK FOR PROPER OPENING
        IF(ISTAT(1)  .EQ.  0) GO TO 10
        WRITE(1,*)ISTAT(1)
        PAUSE 1
C       READ (KEYED) FROM DATA SET TO PREPARE FOR UPDATE
10      CALL DBGET(IDSET,IM1,ISTT1,IBUF1,IARG1)
C       CHECK FOR PROPER READ
        IF (ISTT1(1)  .EQ.  0) GO TO 20
        WRITE(1,*)ISTT1(1)
        PAUSE  2
C       UPDATE THE DATA ENTRY PREVIOUSLY READ
20      CALL DBUPD(ID1,ISTT2,INBR,IVALU,IBUF2)
C       CHECK FOR PROPER UPDATE
        IF (ISTT2(1)  .EQ.  0) GO TO 30
        WRITE(1,*)ISTT2(1)
        PAUSE  3
C       WRITE UPDATED RECORD FOR THE USER TO EXAMINE
30      WRITE(1,40)IBUF2
40      FORMAT(2(10A2,X,5A2,X),2(3A2,X),2A2)
C       CLOSE THE DATA BASE
        IMODE=0
        CALL DBCLS(IMODE,ISTAT)
C       CHECK THAT DATA BASE CLOSED PROPERLY
        IF (ISTAT(1)  .EQ.  0) GO TO 50
        WRITE(1,*)ISTAT(1)
        PAUSE  4
END.
```

Figure 5-2. DBUPD Example

SECTION VI
DBPUT - Data Base Write

DBPUT enters new data entries into Manual or Detail data sets. Users of DBPUT must open the data base using DBOPN with a mode of access equal to either 3 or 5. To open the data base in either of the two modes, the user must specify a level word (as defined in the data base schema stored in the root file) corresponding to level number 15.

When entering a new data entry using DBPUT, the user specifies the number of data items in the entry that will contain values. This number can be less than the number of actual data items in the entry.

The data items in the entry to be given values are specified by number in the DBPUT calling sequence. To discover the numbers of the data items prior to calling DBPUT, use the DBINF routine (see Section IX).

Along with the numbers of the data items in the entry to receive values, the user supplies the data item values stored in an array, concatenated together. The user supplies the values in the order that the data item numbers appear in the DBPUT calling sequence. The values supplied in the array must be the same length as the data item length defined in the data schema. Key item values must always be supplied for the data entry. Any data item in the entry not supplied with a value by the user is filled with binary zeros.

When DBPUT is called for a detail data set, DBPUT verifies that the data set is not full, that the necessary manual master data set entries already exist for the key item values specified in the new data entry, that the necessary automatic master data set entries exist, and if they do not exist that the master data sets are not full. If any of these criteria fail, DBPUT stores an error code in the first word of the status area defined for DBPUT in the user calling sequence. If all of the above are true, then DBPUT stores the entry in the detail data set, stores the required entries in the automatic data sets, performs all linkage maintenance, and stores a zero in the first word of the status area.

For a manual master data set, DBPUT verifies that the data set is not full and that no data entry exists in the data set with a key item value identical to the key item value of the new data entry to be entered. If one of the above tests fail, DBPUT stores an error code in the first word of the status area defined for DBPUT in the user calling sequence. If the above tests are true, DBPUT stores the new entry in the data set and stores a zero in the user-defined status area.

DBPUT CALLING SEQUENCE

The FORTRAN calling sequence for DBPUT is:

```
CALL DBPUT(IDSET,ISTAT,INBR,IVALU,IBUF)
```

The Assembly Language calling sequence is:

```
JSB  DBGET
DEF  *+6
DEF  IDSET
DEF  ISTAT
DEF  INBR
DEF  IVALU
DEF  IBUF
```

DBPUT PARAMETERS

The following parameters appear in the DBPUT calling sequence:

IDSET — Is a one-dimensional integer array at least three words long containing the name of a manual master or detail data set. The name must be in ASCII format, two characters per word. The data base name must be from one to six characters long, and if less than six characters, the remainder must be blank.

ISTAT — Is a one-dimensional integer array at least one word long used by DBPUT to signal completion or error conditions. If DBPUT successfully stores a data entry in the desired data set, DBPUT stores a zero in the first word of ISTAT. If the entry is not stored due to some error condition, DBPUT stores the appropriate error code in the first word of ISTAT. Consult Appendix A *DBMS Error Messages* for the meaning of the error code.

INBR — Is a one-dimensional integer array. The first word of the array contains the number of data items in the entry which are to be given values. (This number must be equal to or less than the number of data items in the entry.) Remaining words in the array contain the number of the data item to be given a value, one data item number per word of the array. The number of a data item is determined by the order in which the data item was defined in the data schema for the data base. The first data item defined in the *item part* of the data schema is item number one. (See *Designing IMAGE/2100 Data Bases*, 02100-90138.) The number of individual data items can be discovered programmatically through use of the DBINF routine (see Section IX).

IVALU — Is a one-dimensional integer array containing the values to be placed in the data items specified in the INBR parameter. These values are concatenated in the array. Each value must occupy the full amount of space defined for the data item in the data schema for the data base. For example, a character data item defined as U8 (four words long) requires that the user supply an eight-character value for the array specified in IVALU, perhaps by padding the significant characters with trailing or leading blanks.

IBUF — Is a one-dimensional integer array large enough to contain a data entry from the data set being accessed. The data entry length must include the length of the media record as well as the length of the data record. The length of these records can be obtained programmatically through the DBINF routine (see Section IX) or through a copy of a processed data base schema.

DBPUT EXAMPLE

Figure 6-1 shows how DBPUT is used to load a new data entry into a data set of the data base defined in the data schema pictured in Figure 5-1. Like previous examples, the DBMS common area is defined first. Then, DBOPN opens the data base using a mode of access equal to 3 and with the level 15 code word as defined in Figure 5-1. Then the complete data entry is loaded into the data set using DBPUT. Finally, the data base is closed using DBCLS.

```
C       THIS PROGRAM LOADS A NEW RECORD INTO A DATA SET
C       DEFINE THE COMMON AREA FOR DBMS FIRST
        COMMON ISORT(1024),ISLGT(1),IRUN(1175),IOPN(128)
C       DEFINE PARAMETERS FOR THE DBMS CALLING SEQUENCES
        DIMENSION IBASE(3),ILEVL(3),ISCOD(1),IMODE(1),ISTAT(2),
      *           ID1(3),ISTT1(1),INBR(8),IVALU(38),IBUF2(42)
C       INITIALIZE PARAMETERS WITH VALUES
        DATA IBASE/2HEM,2HPL,2HY  /,ILEVL/2HOW,2HNE,2HR  /,ISCOD/100/,
      *       IMODE/3/,ISTAT/2*0/,ID1/2HPE,2HRS,2HON/,ISTT1/0/,
      *       INBR/7,1,2,3,4,5,6,7/,
      *       IVALU/2HJO,2HHN,2HST,2HON,6*2H   ,2HFR,2HAN,2HK  ,2*2H   ,
      *       2H20,2H20,2H  L,2HAT,2HHA,2HM  ,2HST,2H. ,2*2H   ,
      *       2HPA,2HLO,2H  A,2HLT,2HO  ,2HCA,2HLI,2HF.,2H94,2H50,2H1  ,
      *       2H09,2H44/,IBUF2/42*0/
C       OPEN THE DATA BASE
        CALL DBOPN(IBASE,ILEVL,ISCOD,IMODE,ISTAT)
C       CHECK FOR PROPER OPENING
        IF((ISTAT(1) .EQ. 0) GO TO 10
        WRITE(1,*)ISTAT(1)
        PAUSE 1
C       LOAD THE NEW ENTRY INTO THE DATA SET
10      CALL DBPUT(ID1,ISTT1,INBR,IVALU,IBUF2)
C       CHECK THAT THE ENTRY LOADED CORRECTLY
        IF (ISTT1(1) .EQ. 0) GO TO 20
        WRITE(1,*)ISTT1(1)
        PAUSE 2
C       CLOSE THE DATA BASE
20      IMODE(1)=0
        CALL DBCLS(IMODE,ISTAT)
C       CHECK THAT THE DATA BASE CLOSED PROPERLY
        IF (ISTAT(1) .EQ. 0) GO TO 30
        WRITE(1,*)ISTAT(1)
        PAUSE 3
30      CONTINUE
END.
```

Figure 6-1. DBPUT Example

SECTION VII

DBDEL - Data Base Delete

DBDEL removes data entries from the master or detail data set named in the DBDEL calling sequence. The data entry removed is the last data entry accessed by a previous call to DBGET, DBPUT, or DBUPD, as indicated in the Run Table for the data set in the DBMS work area. Any data entry in a detail data set can be removed without restriction. To delete a master data set entry, the user must first remove all entries in detail data sets on the chains defined by the master entry, (i.e., all the chains on the path defined by the master data entry values are empty).

To use DBDEL, the user must open the data base with a mode of access defined in the DBOPN calling sequence equal to 3.

DBDEL CALLING SEQUENCE

The FORTRAN calling sequence for DBDEL is:

```
CALL DBDEL (IDSET,ISTAT)
```

The Assembly Language calling sequence is:

```
JSB  DBDEL
DEF  *+3
DEF  IDSET
DEF  ISTAT
```

DBDEL PARAMETERS

The following parameters appear in the DBDEL calling sequence:

IDSET — Is a one-dimensional integer array at least three words long containing the name of a master or detail data set. The name must be in ASCII format, two characters per word. The name must be from one to six characters long, and if less than six characters, the remainder must be blank.

ISTAT Is a one-dimensional integer array at least one word long used by DBDEL to signal completion of the deletion or to report an error condition. A zero in the status word indicates completion, while a positive integer indicates an error condition. Consult Appendis A, *DBMS Error Messages* for the meaning of the error code.

DBDEL EXAMPLE

Figure 7-1, below, shows an example of the use of the DBDEL routine to delete a data entry from the data base. This example is similar to the one shown in Figure 4-1. Like Figure 4-1, the program first calls DBGET to read a data entry from a data base. Once having done so, the program calls DBDEL to remove the data entry last accessed from the data set.

```
C     THIS PROGRAM PERFORMS A CHAIN READ AND THEN DELETES.
C     DECLARE DBMS COMMON WORK AREA TO SET UP RUN TABLES
      COMMON ISORT(1024),ISLGT(1),IRUN(1175),IOPN(128)
C     DECLARE NECESSARY PARAMETERS FOR DBGET,DBOPN,DBFND
      DIMENSION ISET(3),IMODE(1),ISTAT(4),IBUF(25),IARG(3)
      DIMENSION ID1(3),ILEVL(3),ISCOD(1),IM1(1),ISTT1(2)
      DIMENSION ID2(3),ISTT2(4),IPATH(3),IARG1(3),ID3(3)
C     INITIALIZE THE PARAMETERS USING DATA STATEMENTS
      DATA ID1/2HCL,2HIE,2HNT/,ILEVL/2HOW,2HNE,2HR  /
      DATA ISCOD/1234/,IM1/3/,ISTT1/2*0/
      DATA ID2/2HPT,2HIE,2HNT/,ISTT2/4*0/
      DATA IPATH/2HEY,2HE-,2HC  /,IARG1/2HBL,2HUE,2H    /
      DATA ISET/2HPT,2HIE,2HNT/,IMODE/1/,ISTAT/4*0/
      DATA IBUF/25*0/,IARG/3*0/
      DATA ID3/2HEY,2HES,2H    /

C     OPEN THE DATA BASE USING DBOPN
      CALL DBOPN(ID1,ILEVL,ISCOD,IM1,ISTT1)
      IF (ISTT1(1) .EQ. 0) GO TO 50
      WRITE (1,*) ISTT2(1)
      PAUSE 1
C     PREPARE FOR A CHAIN READ BY CALLING DBFND

50    CALL DBFND(ISTT2,ID2,IPATH,IARG1)
      IF(ISTT2(1) .EQ. 0) GO TO 60
      WRITE (1,*)ISTT2
      PAUSE 2

C     PERFORM THE CHAIN READ.

      CALL DBGET(ISET,IMODE,ISTAT,IBUF,IARG)
      IF(ISTAT(1) .EQ. 0) GO TO 62
      WRITE(1,*)ISTAT(1)
      PAUSE 3
```

Figure 7-1. DBDEL Example

```
C       WRITE THE ENTIRE RECORD (MEDIA+DATA)
62      WRITE(1,65)IBUF
C       THE FORMAT STATEMENT BELOW PRINTS THE MEDIA RECORD
C       AS THREE INTEGER VALUES AND THE DATA RECORD AS
C       A CONTIGUOUS CHARACTER STRING
65      FORMAT(3(I,X)22A2)
C       DELETE THE RECORD ACCESSED BY DBGET
        CALL DBDEL(ISET,ISTAT)
C       CHECK THAT THE RECORD WAS DELETED
        IF (ISTAT(1) .EQ. 0) GO TO 70
        WRITE(1,*)ISTAT(1)
        PAUSE 4
C       CLOSE THE DATA BASE
70      IMODE=0
        CALL DBCLS(IMODE,ISTAT)
C       CHECK THAT THE DATA BASE CLOSED PROPERLY
        IF (ISTAT(1) .EQ. 0) GO TO 80
        WRITE(1,*)STAT(1)
        PAUSE 5
80      CONTINUE
        END.
```


Figure 7-1. DBDEL Example (cont.)

SECTION VIII

DBFND - Data Base Find

DBFND locates master data set entries containing the key item value specified in the calling sequence. If no data entry exists with a key item value equal to the one specified by the user, DBFND stores an error code in the first word of the status area (defined by the user in the calling sequence). If DBFND does locate a data entry with the same key item value, DBFND stores zero in the first word of the status area, and the relative record number of the accessed master data entry in the second word. The length (number of entries) of the chain in a detail data set linked to the master through the key item value is stored in the third word, and the relative record number of the first detail entry in the chain accessed by DBFND, is stored in the fourth word.

DBFND is used primarily to set up a chain read using the DBGET routine. When DBFND accesses the master data entry, it takes the pointer to the head of the chain in the detail data set and stores that pointer in the Run Table of the DBMS work area. When the user requests a chain read using DBGET, the routine accesses the first entry of the chain by using the relative record address stored in the Run Table by DBFND.

CALLING SEQUENCES

The FORTRAN calling sequence for DBFND is:

```
CALL DBFND(ISTAT,IDSET,IPATH,IARG)
```

The Assembly Language calling sequence is:

```
JSB   DBFND
DEF   *+5
DEF   ISTAT
DEF   IDSET
DEF   IPATH
DEF   IARG
```

DBFND PARAMETERS

The following parameters appear in the DBFND calling sequence:

ISTAT — Is a one-dimensional integer array at least four computer words long used by DBFND to signal completion of a read or to report an error condition. Table 8-1 describes the four words and their meaning:

Table 8-1. DBFND Status Word (ISTAT)

Word 1 (Condition Word) DBFND loads a zero into the Condition Word to signify that the data entry was found in the master data set. If a data entry with the specified key item value is not found, or some other error condition occurs DBFND loads an error code into the Condition Word. Consult Appendix A, *DBMS Error Messages.*
Word 2 (Record Address) Contains a relative record address of the data entry found by DBFND. The first record address is number 1. One data entry (logical record plus media record) is stored in one physical record.
Word 3 (Chain Length) DBFND stores the number of entries in the currently-accessed chain into Word 3.
Word 4 (Forward Address) Contains the relative record address of the first entry in the currently-accessed chain.

IDSET — Is a one-dimensional integer array at least three words long containing the name of the detail data set linked to the master data set through the key item named in IPATH and the key item value named in IARG. The name must be in ASCII format, two characters per word. The name must be from one to six characters long, and if less than six characters, the remaining characters must be blank.

IPATH — Is a one-dimensional integer array at least three words long containing a data item name which is a key item in the data set specified in IDSET. The name must appear in IPATH in the same form as specified for the data set name in IDSET.

IARG — Is a one-dimensional integer array containing the value of the master data set key item linked to the detail data set key item. The length of IARG depends upon the type and length of the desired key item value.

DBFND EXAMPLE

DBFND is used primarily to set up the Run Table for a data set prior to performing a chain read on the first entry of a data set using the DBGET routine. Figure 4-1 in Section IV of this manual shows an example of DBFND used to initialize a data set for a chain read. DBFND searches for the master data entry containing the value BLUE for the key item named EYE-C. DBGET then performs a chain read for all the entries in the detail data set with the value BLUE for the detail data set key item linked to the master data set specified by the DBFND calling sequence.

SECTION IX
DBINF - Data Base Information

Some DBMS routines such as DBUPD and DBPUT require that the user refer to data items in the data set by number. The DBINF routine searches the data base root file (data schema) and returns information about data sets and specific data items to the user. The type and order of information given is described under *DBINF Parameters* in this section.

The first data set defined in the data schema is set number 1; the second data set is set number 2, and so on. The first data item defined in the data schema *item part* is item number 1, the second item defined is number 2, and so on. For more information about the data base schema, consult *Designing IMAGE/2100 Data Bases*, 02100-90138.

CALLING SEQUENCES

The FORTRAN calling sequence for DBINF is:

```
CALL DBINF(ITYPE,IMODE,ID,IBUF)
```

The Assembly Language calling sequence is:

```
JSB   DBINF
DEF   *+5
DEF   ITYPE
DEF   IMODE
DEF   ID
DEF   IBUF
```

DBINF PARAMETERS

The following parameters appear in the DBINF calling sequence:

ITYPE — Is an integer variable containing two ASCII characters. The value "I" followed by a blank indicates that DBINF is to return information about a data item, while the value "S" followed by a blank indicates that DBINF is to return information about a data set. The value of ITYPE along with the value of the IMODE parameter indicates the exact information returned to the user by DBINF (see below).

IMODE — Is an integer variable containing an integer value from 1 to 5. This number in combination with the value of ITYPE determines the information returned to the user in the array defined by the IBUF parameter.

ID — Is a one-dimensional integer array which either contains an integer number in the first word of the array, or the name of a data item or data set stored in ASCII format two characters per word. If the data item or data set name is less than five characters, the remainder must be blanks.

IBUF — Is a one-dimensional integer array which contains the information requested by the user. The information returned to the user depends upon the values of ITYPE, IMODE, and ID. In any case, the first word of IBUF is used as a condition word. If the call is successful, DBINF loads zero into the first word of IBUF. If an exceptional condition occurs, DBINF loads an error code into the first word of IBUF. Consult Appendix A, *DBMS Error Messages* for the meaning of the codes.

The following describes the contents of IBUF as determined by the values of IMODE, ITYPE, and ID.

IMODE = 1

ITYPE = "IΔ" (Δ is an ASCII blank.)

ID — is the name of a variable containing the number of the desired data set. The data sets are numbered from 1 to *n* as they are defined in the data schema *set part* (see *Designing IMAGE/2100 Data Bases*, 02100-90138).

IBUF — contains the number of data items in a data entry of the specified data set, followed by data item numbers, one data item number per word of IBUF. Data item numbers are either positive or negative. A positive number indicates that the level code word used to open the data base (using the DBOPN routine) is sufficient only to read the data item but not to write into it, while a negative number indicates that the level code word is sufficient to both read and write into the data item. The data item number is omitted from IBUF if the level code word is not sufficient to read or write the item. Figure 9-1 shows the contents of IBUF.

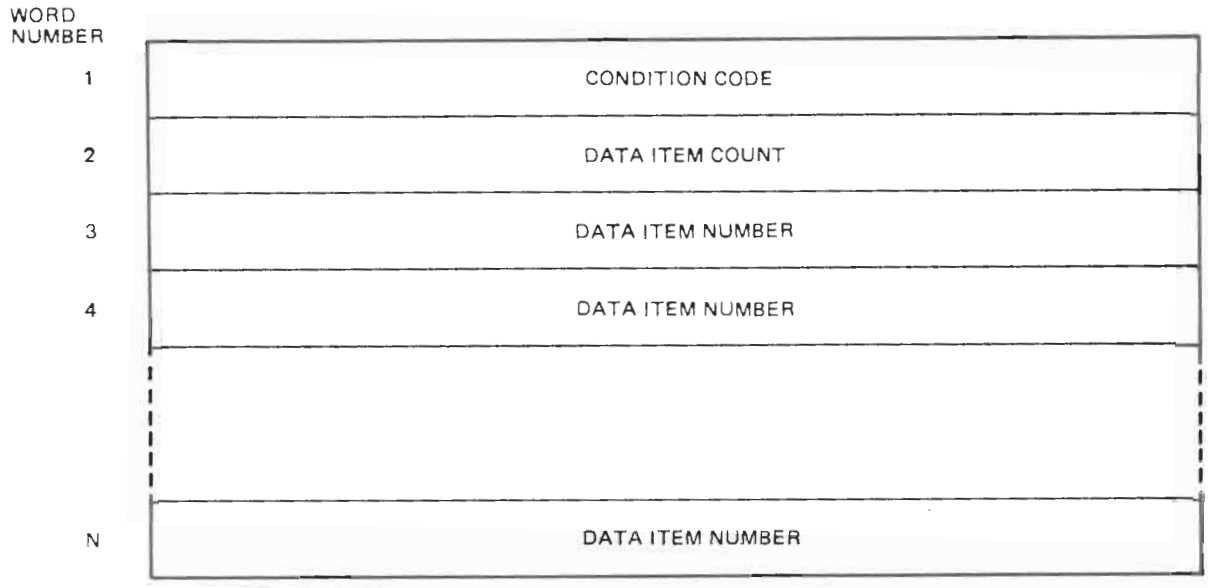

Figure 9-1. IBUF Contents With IMODE = 1 And ITYPE = "IΔ"

IMODE = 2

ITYPE = "IΔ"

ID is the name of an integer variable containing a data item number. Data items are numbered from 1 to *n* as they are defined in the *item part* of a data schema (see *Designing IMAGE/2100 Data Bases*, 02100-90138).

IBUF contains (after the condition code):

1. DATA ITEM NAME in the next six bytes of IBUF (a byte is one-half of a 16-bit computer word). If the item name is less than six characters, the remaining bytes of the character string are set to blanks.

2. SEARCH TYPE in the next byte of IBUF. If the data item is a search (key) item of the data set which contains it, the SEARCH TYPE is set to 1. If the data item is not a key item, SEARCH TYPE is set to 0.

3. ITEM TYPE in the next byte of IBUF is set to "I" if the data item is an integer, "R" if the data item is real, or "U" if the data item is character.

4. ITEM LEVEL in the next word of IBUF. The left byte of the word contains the data item read level as defined in the data schema, while the right byte of the word contains the data item write level as defined in the data schema.

5. ITEM LENGTH in the next word of IBUF. This binary value gives the length of the data item in words. An integer data item is length one, a real value of length two, and an ASCII value of the length defined in the schema.

6. ITEM OFFSET in the next word of IBUF. This value gives the location of the data item in the data entry relative to the beginning word of the entry. In the physical record containing the entry, the media record is located in front of the data record. The first word of the entry is number 1, the second is number 2, etc.

7. DATA SET NUMBER in the next word of IBUF. This is the number of the data set in which the data item is located. Data sets are numbered from 1 to *n* in the order they are defined in the data schema.

Figure 9-2 below shows the contents of IBUF.

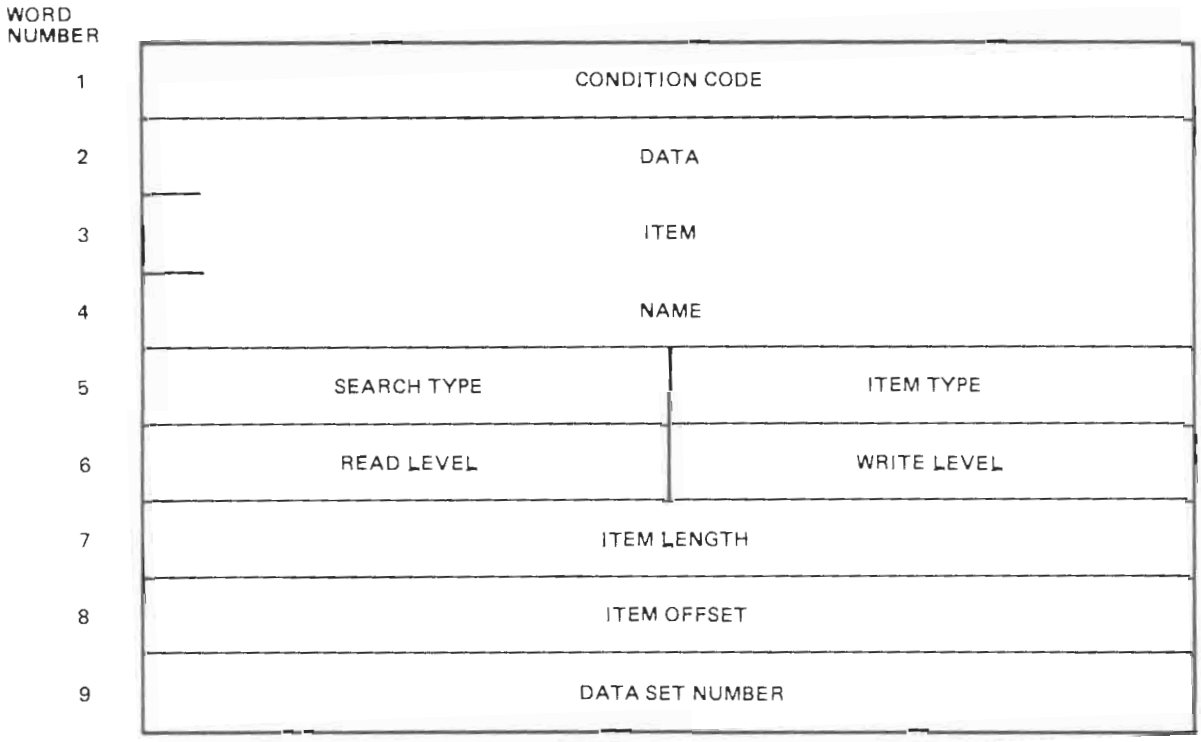

Figure 9-2. IBUF Contents With IMODE = 2 And ITYPE = "I△"

IMODE = 2

ITYPE = "S△"

ID is the name of an integer variable containing a data set number. Data sets are numbered from 1 to *n* as they are defined in the *set part* of a data schema.

IBUF contains (after the condition code):

1. DATA SET NAME in the next six bytes of IBUF (a byte is one-half of a 16-bit computer word). If the data set name is less than six characters, the remaining bytes of the character string are set to blanks.

2. DATA SET TYPE in the next byte of IBUF. The value "A" stands for an automatic master, "M" stands for a manual master, and "D" stands for a detail.

3. CAPACITY in the next word of IBUF. This value indicates the maximum number of entries that can be held by the data set.

4. ENTRY LENGTH in the next word of IBUF. This value is the length in words of an entry in the data set, including both the media and data record.

Figure 9-3 below shows the contents of IBUF.

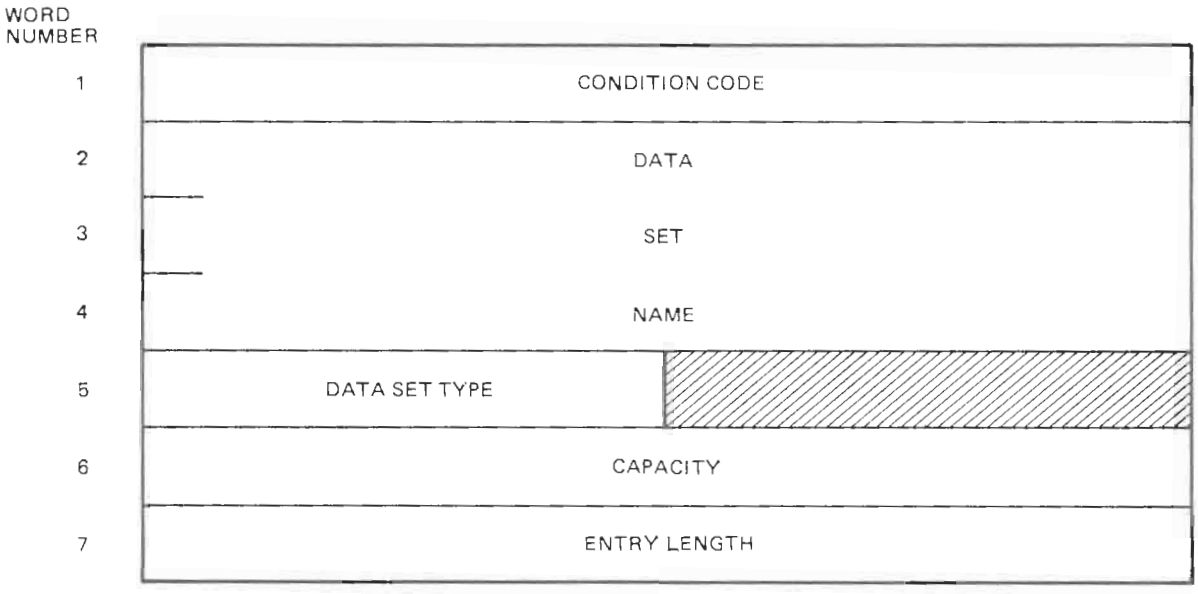

Figure 9-3. IBUF Contents With IMODE = 2 And ITYPE = "S△"

IMODE = 3

ITYPE = "I△"

ID is a data set number of the desired data set. Data sets are numbered from 1 to *n* as they are defined in the *set part* of a data schema.

IBUF contains a count of data items in an entry of the data set that are key items. Remaining words of IBUF contain the data item numbers of the keys, one data item number per word of IBUF.

Figure 9-4 below shows the contents of IBUF.

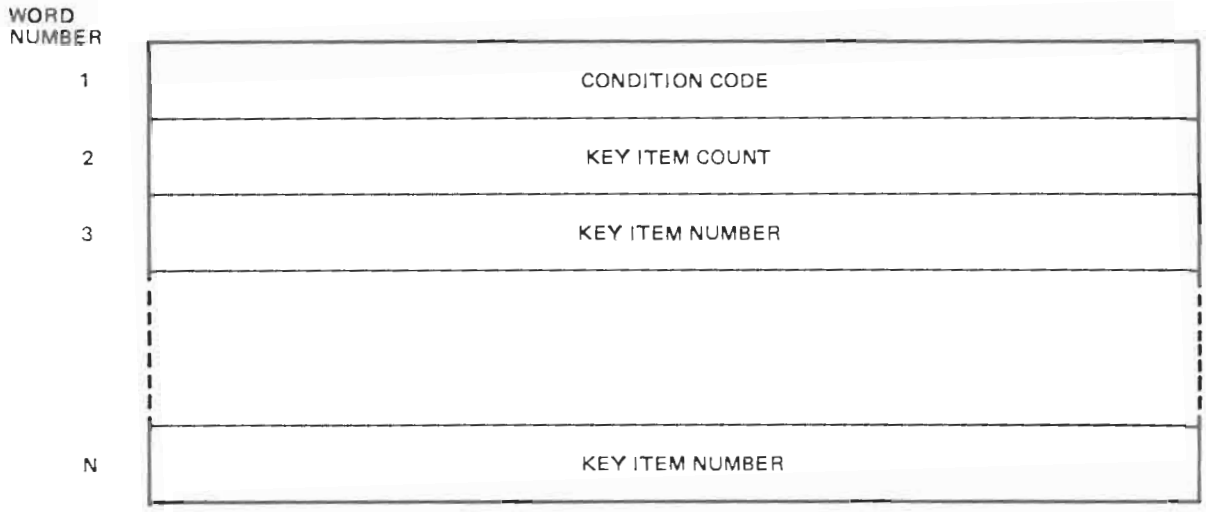

Figure 9-4. IBUF Contents With IMODE = 3 And ITYPE = "I△"

IMODE = 4

ITYPE = "SΔ"

ID is the name of a variable containing a data item number. The data item referenced must be a key item of a data set.

IBUF contains the data set numbers of all data sets related to a specific data set through the key item defined in ID. The first word of IBUF (after the condition code) is a count of the number of data sets linked to a specific data set (containing the key item in ID). Following that word, IBUF contains a pair of values (one value per word) indicating the data set number and the key item number of the data set linked to the specific data set through the key item defined in ID. As many data set/data item pairs follow the first word of IBUF as were specified in the first word of IBUF.

Figure 9-5 shows the contents of IBUF.

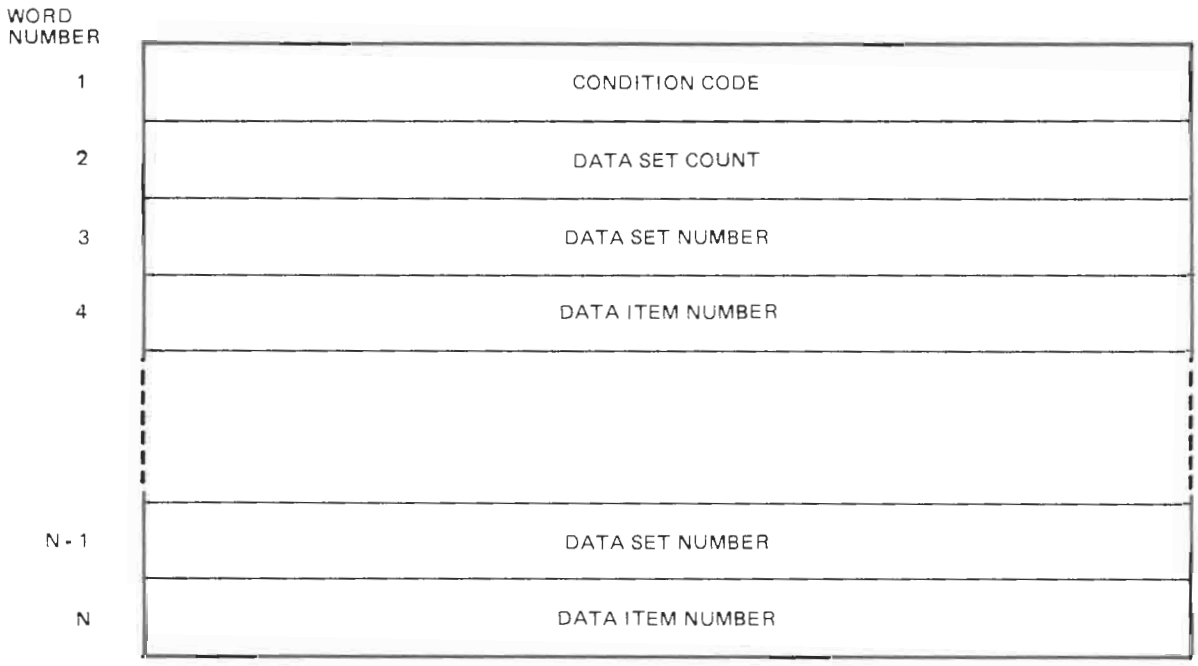

Figure 9-5. IBUF Contents With IMODE = 4 And ITYPE = "SΔ"

DBINF EXAMPLE

Figure 9-9 below shows part of a data base schema defining one master data set and two detail data sets that are linked to the master through the master data key item named A. Figure 9-10 shows part of a user program which calls DBINF with IMODE = 4 and ITYPE = "SΔ". ID contains the data item number of the master data set key item named A. Notice that A is defined first in the *item part* of the data schema in Figure 9-9, so the data item number in ID is 1.

IMODE = 4 and ITYPE = "SΔ" indicates that DBINF is to return the number of data sets linked to the data set whose key item is indicated in ID, along with the data set number and the key item number for the data sets linked to the key item in ID. Figure 9-11 shows the contents of IBUF after the call to DBINF is executed. The second word contains the data set number of data set BAKER (2) and the third word contains the data item number of the key item of BAKER linked to A (2). The fourth word contains the data set number of the next set linked to ABLE (3) and the fifth word contains the data item number of the key item C linked to A (3).

```
EFMP PACK ID PN001;100;
BEGIN DATA BASE CABCO
.
.
.
ITEMS:
   A, I1(1,15);
   B, I1(1,15);
   C, I1(1,15);
SETS:

NAME: ABLE, M;
ENTRY: A(3);
CAPACITY: 100;

NAME: BAKER, D;
ENTRY: B(ABLE);
CAPACITY: 100;

NAME: CHARLIE, D;
ENTRY: C(ABLE);
CAPACITY: 100;

END.
```

Figure 9-9. Sample Data Base For DBINF Example

```
.
.
.
.
.
DATA ITYPE/2HS  /,IMODE/4/,ID/1/,IBUF/5*0/
.
.
.
.
.
.
.
.
.
CALL DBINF(ITYPE,IMODE,ID,IBUF)
```

Figure 9-10. DBINF Call

WORD NUMBER	IBUF
1	0
2	2
3	2
4	2
5	3
6	3

Figure 9-11. IBUF Contents After DBINF Call

Error Code	Description
111	The user has specified an invalid record number. The routine has discovered an invalid record number. (DBUPD,DBDEL,DBGET).
112	The user has attempted to update a data item that is defined as a key item. (DBUPD).
113	The user has attempted to delete a master data set that is linked to a chain in a detail that still contains data entries linked to the master. (DBDEL).
114	The record accessed by the user is empty (does not contain a data entry.) (DBUPD,DBDEL,DBGET).
115	The mode of execution mentioned in the calling sequence is invalid. (DBOPN, DBCLS,DBGET,DBINF).
116	The root file referenced does not contain a valid data base schema. (DBOPN).
117	The security code referenced in the calling sequence does not equal the data base security code. (DBOPN).
118	The access level specified in the calling sequence is invalid. (DBOPN).
119	The root file data base name referenced in the calling sequence does not exist. (DBOPN,DBCLS).
120	The data set referenced in the parameter list is not a detail data set. (DBFND).
121	The detail data set referenced in the calling sequence is not linked to any masters. (DBFND).
122	The path count for a chain read using the DBGET routine is zero. DBFND must be called prior to a chain read. (DBGET)
123	The data set referenced in calling sequence for a keyed read is not a master data set.
124	The value of ITYPE in the calling sequence is invalid. (DBINF).
125	The value of the variable referenced by ID in the calling sequence is invalid. (DBINF).
126	The data base being opened in already open. (DBOPN).

02100—90139