UNIVAC 1100 SERIES

DATA MANAGEMENT SYSTEM (DMS 1100)

SYSTEM SUPPORT FUNCTIONS

PROGRAMMER REFERENCE

SPERRY UNIVAC

A

UP-7909 Rev. 1

This document contains the latest information available at the time of publication. However, the Univac Division reserves the right to modify or revise its contents. To ensure that you have the most recent information, contact your local Univac Representative.

UNIVAC is a registered trademark of the Sperry Rand Corporation.

PRINTED IN U.S.A.

PAGE STATUS SUMMARY

ISSUE: UP–7909 Rev. 1

Section	Page Number	Update Level
Cover/Disclaimer		Orig.
PSS	1	Orig.
Acknowledgment	1	Orig.
Preface	1	Orig.
Contents	1 thru 4	Orig.
1	1	Orig.
2	1 thru 5	Orig.
3	1	Orig.
4	1, 2	Orig.
5	1, 2	Orig.
6	1 thru 11	Orig.
7	1 thru 20	Orig.
Appendix A	1 thru 6	Orig.
Appendix B	1 thru 5	Orig.
Appendix C	1 thru 3	Orig.
Appendix D	1	Orig.
Appendix E	1	Orig.
UCS		

Section	Page Number	Update Level

Section	Page Number	Update Level

ACKNOWLEDGMENT

Univac wishes to acknowledge the efforts of the CODASYL Programming Language Committee (PLC) Data Base Task Group (DBTG). The DBTG produced two reports containing specifications for a standardized Data Management facility containing a DATA DESCRIPTION LANGUAGE and a DATA MANIPULATION LANGUAGE. The DBTG issued its first report in October 1969, the second, revised, and expanded, in April 1971. Univac is a member of the DBTG and participated in the development of the Data Management specifications. Univac's implementation of the 1100 Series Data Management System (DMS 1100) is based upon the DBTG specifications.

PREFACE

This document is the Programmer Reference Manual for the UNIVAC 1100 Series Data Management System (DMS 1100) SYSTEM SUPPORT FUNCTIONS. It is one of a series of manuals covering DMS 1100. It is intended that this Programmer Reference Manual be used to provide support functions to databases defined using the *DMS 1100 Schema Definition Data Administrator Reference, UP-7907* (current version).

CONTENTS

FIGURES

1. INTRODUCTION

1.1. GENERAL

The UNIVAC 1100 Series Data Management System (DMS 1100) is comprised of three basic areas of responsibility:

(1) Schema Definition.

(2) Data Manipulation.

(3) System Support.

While the system support functions are largely independent of the schema definition and data manipulation components, some DDL and DML clauses have influence on the operation of these functions. Where such interaction occurs, the involved clauses are referenced.

The general characteristics of the DMS 1100 system which might have impact upon concurrent users of the system are described in Section 2.

The four categories of system support functions which currently exist are as follows:

(1) The DMS 1100 system's generation function (see Section 3) which enables the Data Administrator to create a unique version of DMS 1100 which reflects his particular processing requirements.

(2) The system initialization function which must be performed before the first run unit may access the data base through DMR (see Section 4).

(3) The system supplied routines for ensuring the integrity of the data base (see Section 6).

(4) The DMS 1100 utility processor through which a variety of database utility functions may be performed (see Section 7).

2. DMS 1100 CHARACTERISTICS

2.1. GENERAL

The DMS 1100 enables multiple run units to simultaneously access a common database. The multi-thread characteristics encompass all of the control necessary to allow multiple run unit access of the database. These are as follows:

- Reentrant processor control,
- Queuing and deadlock detection, and
- Locks.

2.2. REENTRANT PROCESSOR CONTROL

The Reentrant Processor (REP) control consists of two phases:

(1) User interface, and

(2) Systems interface.

2.2.1. User Interface

Since the DMR is a REP, all EXEC considerations for REP's apply (see section on REP's *UNIVAC 1100 Series Operating System Programmer Reference, UP-4144* (current version)). Also, CALC routines must be collected in the programs D-bank (see *UNIVAC 1100 Series Data Management System (DMS 1100) Schema Definition Reference, UP-7907* (current version) Appendix F).

The communication between the run unit and the DMR REP is performed by the LINKER. The LINKER is collected with each program. The DML preprocessor generates a call to the LINKER for each DML command, in a COBOL-DML program.

The LINKER, upon entry from a command sequence does the validity checks required at the LINKER level and then calls the DMR.

Return from the DMR is made through the LINKER. A normal return causes the LINKER to return in line to the main program, i.e., to the COBOL program's next verb.

Error or contingency conditions are handled as follows:

On execution of the IMPART command, the LINKER registers a contingency address with the EXEC.

- If a contingency occurs the contingency routine will be entered.
- If the contingency occurred while in the user program, a DEPART WITH ROLLBACK is performed and return is made to the run units rollback-error-paragraph.
- If the contingency occurs in the DMR, either run unit Rollback or QUICK RECOVERY is necessary.

If an error is found which does not require run unit Rollback, return is to either the error paragraph named in the COBOL main program, or if none exists, in-line as expected by COBOL. If run unit Rollback is executed, return is made to the rollback error address registered by the main program. The rollback error address is used for all commands except 'DEPART WITH ROLLBACK'. Return after execution of DEPART WITH ROLLBACK is in-line as ROLLBACK *is* the successful execution of the command.

The type of errors requiring run unit Rollback are as follows:

- Deadlock between run units,
- The run unit has an error after alteration of the database has occurred, but it has not specified command Quick-Before looks,
- DMR internal error, and
- DMR is performing a system rollback.

Validity checking by the LINKER on IMPART command execution includes checking for, or if necessary, establishing a valid run unit identity. The run unit establishes its identity in the Data Management Communication Area (DMCA). If none is established, the LINKER will create one using the RUN-ID from the program PCT. In any case, after the run unit name is established DMS 1100 further qualifies it by attaching the IMPART time. This combined identity/time stamp is used by the Locking and Recovery mechanisms.

2.2.2. Systems Interface

Since the DMR's systems buffers are maintained with the DMR, it is necessary that the DMR REP be locked in core whenever a run unit is active so that tables and pages are not overlaid. This function is performed by the TIMER, which is an independent program that gets started when the first run unit enters the DMR. The TIMER performs this function by linking to the DMR and raising itself to real-time, thereby locking the DMR REP in main storage until all run units have DEPARTED at which time the DMR is released from main storage.

The TIMER is assigned the audit trail tape and transfer of information to the audit trail is performed under the control thread of the TIMER. A run unit can place information on the audit trail by means of the LOG command. The information is transferred from the run units record delivery area to the DMR's systems buffer, where the TIMER acquires it for the write to the audit trail tape. The TIMER maintains the Systems file, checking it on original entry and updating it as necessary.

Also, the TIMER times out run units based on a system's generation parameter.

2.3. QUEUES AND DEADLOCK DETECTION

In a multi-thread environment several facilities must be shared. In some instances, one run unit may have a facility which another run unit needs and cannot proceed without. Facilities in this sense include main storage buffers, data pages, control tables, etc. The DMR recognizes when a condition such as this exists and queues the run unit, that is, it stops execution of the run unit while the facility remains locked. When the facility becomes available, the DMR recognizes this, takes the run unit off the queue and it continues.

An outgrowth of this capability is 'deadlock'. Deadlock is defined as a situation in which two or more run units are queued, each holding a facility required by another queued run unit. In this instance, none can complete until another releases a facility, but since none are active, no facility is released. All run units are then inactive and the system has stopped.

The DMR recognizes the occurrence of deadlock and takes action to break it and allows the run units to continue. The action taken is to select a run unit and roll it back. Selection of a run unit is based on the number of alterations made to the database unless overridden by a user specified parameter. The run unit having made the least number of alterations is selected for rollback.

The run units have the ability to specify a parameter which will reflect run unit priority for rollback selection. This parameter is established in the DMCA, either by a PRIORITY statement, or by a COBOL MOVE into the PRIORITY word prior to issuing each command. Since no individual run unit knows its priority relative to other existing runs units, it is envisioned that the basis for this priority scheme could be the number of external messages involved, or some value arrived at by an algorithm, or imposed by a Supervisory routine. In order to use the PRIORITY statement, a system's generation parameter must be set to indicate that the DMR selection process is not to be used in resolving deadlock.

2.4. LOCKS

Locks are applied for a run unit at two levels as follows:

(1) Page

(2) Area.

The Page Lock is the exclusive right to access a page. An Area Lock is the exclusive right to access an Area. Exclusive access means that no other run unit can access the locked facility while the lock is in effect. In addition to exclusive access to an AREA, a PROTECTED access is available.

2.4.1. Page Locks

Progressive locks imply that when a run unit alters a page, a lock is applied to the page and that lock remains in effect for the life of the run unit, or until a FREE command is issued. This facility is required in order

to provide an economical run unit rollback capability. Page alteration occurs during each of the following commands:

- KEEP,
- STORE,
- MODIFY,
- INSERT,
- REMOVE, and
- DELETE.

A temporary lock is applied when a FIND command is issued which selects a new 'current of run unit'. This lock is removed on either the next FIND or STORE command.

The two commands uniquely involved with page locks are KEEP and FREE. The purpose of a KEEP command is to place a page lock on the page that is "current of run unit", even though no alteration has been made to the page.

The FREE command releases all page locks and Quick-Before looks to enable other run units to access these pages. Usage of the FREE requires extreme care, since as a result of a FREE all Recovery and Rollback functions handle the point in time at which the FREE was performed as though a DEPART has been performed. This means that on run unit Rollback, the database is restored to the state which existed when the FREE command was issued. Also, for Long Recovery, when run units active at a Recovery Point are backed out of the system, they are backed out to the point of the most immediate FREE in front of the Recovery Point if any FREE was issued.

Example:

The use of the KEEP command.

Assume run unit 1 accesses Record Z through a 'FETCH Z' command, moves the record to its working storage, inspects the record, and then wants to 'FIND X' to use an item in Record X in conjunction with items in Record Z. If Run Unit 1 next issued a FETCH X, Record Z is unlocked and could be altered by another run unit. To prevent another run unit from intervening, Run Unit 1 would execute a KEEP.

All locks applied in conjunction with page alterations by the run unit remain in effect until the run unit executes a FREE or DEPARTs. In either case, all locks and Quick-Before looks established by the run units are released.

In DMS 1100, there are many situations in which a record may be related to records on other pages through pointers. It is a fundamental premise that when pointers are changed this is an alteration of the page on which the record resides. An alteration such as this is sufficient to cause the page to be locked. Therefore, when a record that is current of run unit is the object of an INSERT, REMOVE, MODIFY, STORE or DELETE, other records can be altered with resultant page locks.

Example:

Assume that records A, B and C are on pages 1, 2 and 3, as in Figure 2–1 following.

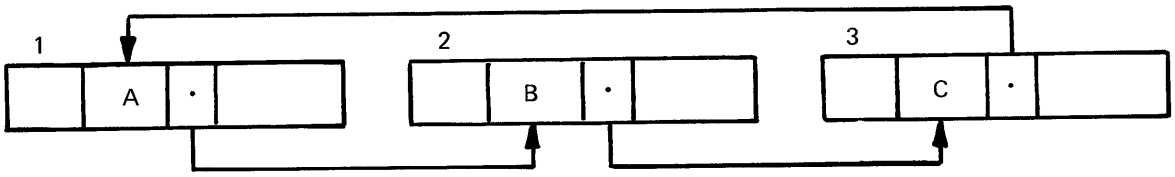


Figure 2–1. KEEP Command Example

If B is REMOVEd from this ring, the pointer in A must be changed to point to C. Therefore, both page 1 and page 2 are locked.

2.4.2. Area Lockout

When a run unit issues an EXCLUSIVE OPEN of an AREA or OPEN FOR INITIAL LOAD, no other run unit can access that AREA. Also, when a PROTECTED OPEN is issued, no update of the AREA is allowed by a concurrent run unit although retrieval is allowed. This condition remains until either the run unit DEPARTs or issues a CLOSE for the AREA.

When a run unit alters pages in an AREA, the pages are locked. These pages remain locked until either a DEPART or a FREE command is issued, even though the run unit issues a CLOSE for the AREA. Therefore, after a CLOSE, and prior to issuance of a DEPART or FREE, only the unaltered pages of the AREA are available to other run units. Altered pages are protected by the Page Locks established for the run unit.

While one run unit has exclusive use of an AREA, any other run unit attempting to access it will be queued. When an AREA is OPENED with a USAGE-MODE of EXCLUSIVE or PROTECTED, the KEEP command is not required since no update of the AREA can be made by concurrent run units.

2.4.3. Locks Implementation

The concept of a page lock is implemented in DMS 1100 using modifications to the page header. Each run unit has a RUID (Run Unit Identity) and a Time Stamp taken when the run unit executes its IMPART (or a FREE). When a page is to be locked, the RUID and Time Stamp are stored in the page header. When a run unit accesses a page, its RUID and Time Stamp are compared to the RUID of the altering run unit in the page header. If they are not the same, and the run unit that altered the page is still active, the requesting run unit is temporarily denied access to the page, and placed on a queue.

When the lock is released, either through issuance of a FREE command by the locking run unit, or by termination of the locking run unit, the queued run unit is activated and given the page.

3. SYSTEM GENERATION

3.1. GENERAL

System's generation parameters are constants which the Data Administrator can change to alter operational or structural characteristics of DMS 1100. These values are in the PDP element DMSSGP. These parameters are subject to change when new releases of DMS are issued taking into account new features. The Data Administrator should consult the appropriate Release Bulletin which accompanies a particular DMS release for a complete description of the system's generation parameters, their names and usage, and the procedure to redefine their standard values.

4. SYSTEM INITIALIZATION

4.1. GENERAL

After the system has been generated and loaded, system initialization must proceed. System initialization consists of the establishment of the System File and the Quick-Look Files. This is accomplished by activating the run streams discussed in Appendix B.1.

4.2. INITIALIZATION OF THE SYSTEM FILE

The System File contains information which is maintained and used by the DMS. The information contained in this file is accessed when the TIMER is activated and written out by the TIMER as required. It serves as the vehicle for retaining information about the system, which would otherwise be lost when a fresh copy of the DMR is brought into main storage.

In order to prepare this file before its first reference by the TIMER, (*i.e., before the first run unit accesses the database*) the System File must be initialized. A utility routine is provided to accomplish this function. Appendix B.1 gives the run stream to be used which executes the absolute element SFINIT. When executed, this program will properly establish all system status flags and other data contained in the System File. Additionally, it will interrogate the system's generation element DMSSGP to determine if an audit trail tape is to be used. If so, it will solicit the appropriate equipment type, catalog the audit trail tape, place the tape servos reserved by the data administrator in DMSSGP into the system file and solicit the tape numbers to be used from the computer operator.

After successful completion of this activity, the operator will be informed that the system file has been initialized.

The System File is revised to reflect changes in the system over time. Hence, once a database has been established, the System File should not be re-initialized by SFINIT. The System File is protected by the recovery routines of DMS. If long recovery is required to restore the database, the System File will be restored. If the System File is lost, Long Recovery must be initiated.

4.3. INITIALIZATION OF THE QUICK-LOOK FILE(S)

The Quick-Look Files are defined in DMSSGP and are used to contain the Quick-Before-Looks of database pages, which have been altered by run units which are still active. After a Run Unit Departs, these looks are no longer required. An internal table describing what portion of the Quick-Look File is used by a run unit is overwritten by a status flag, when it Departs to indicate that a portion of the Quick-Look File may be reused. These status

flags are therefore required initially to indicate that all portions of the Quick-Look File are not in use. The run stream shown in Appendix B.1 which executes QLINIT accomplishes this initialization. Note that the cataloging of the Quick-Look Files must be performed prior to executing QLINIT.

The Quick-Look File is a cataloged file on a random access device. It has valuable information only when there are run units accessing the database through the DMR, or when a system failure has occurred and recovery has not been completed. If an installation uses DMS only during a portion of the day, it may be desirable to Delete the Quick-Look File after all DMS activity is completed for the day and then re-initialize the Quick-Look File prior to the start of the next day's activity. Unlike System File initialization, Quick-Look File initialization may be initiated any number of times during the life of a database.

When all segments of the Quick-Look Files have been successfully initialized a message to this effect will be displayed on the operator's console.

5. LOOKS

5.1. GENERAL

A 'look' is a copy of a database page which is taken in order to provide a means for preserving the integrity of the database.

Looks are not available for access by a run unit, but can only be accessed by the recovery or rollback routines. The three types of looks are as follows:

(1) Before looks

(2) After looks

(3) Quick-Before looks

The first two types of looks are written to an audit trail tape. The third type of looks is written to a random access file.

A page is not considered successfully stored in the database until all required write operations for these looks are successfully completed. The Data Administrator and run units must select which of the looks are to be taken in a given system.

5.2. BEFORE LOOKS

The basic information in a Before look is a copy of the data page prior to alteration. The Before look is written to the audit trail which is a sequential file (*e.g., tape*). The DMR writes control information along with the copy of the page for the Before look.

Only one copy of a page is written to the audit trail for the run unit. That is, when the run unit is about to alter the page for the first time only, the DMR will take a Before look. All subsequent alterations for that page do not require a Before look to be taken.

5.3. AFTER LOOKS

The basic information in an After look is a copy of the data page after the alteration has been made. In addition to the data page, the DMR appends control information and writes the After look to the audit trail tape. This write is performed in conjunction with the write of the page to the database. An After look is taken whenever

the modified page is returned to the database, not everytime the page is modified. If a page is not altered, it is overlaid without being written back to the database. A page is only written back to the database when it has been modified and one of the following conditions exist:

- It must be swapped to make room for another page.
- The run unit terminates.
- A FREE is issued.

5.4. QUICK-BEFORE LOOKS

The Quick-Before look contains the same information as the Before look. However, the Quick-Before look is written to a file placed on a fast device (*e.g., FH 1782, FH 432, type 8440 disc file*). This file is segmented by run unit and is specifically available to provide a capability for command and run unit rollback.

5.5. DATA ADMINISTRATION SPECIFICATION OF LOOKS

Since it is a responsibility of the Data Administrator to guarantee integrity of the database, it is necessary that provisions to save the required data be under his control. As a result of this, the ability is provided in the DDL to specify which type of looks are to be taken for each AREA.

To provide this ability, two AREA LOOKS clauses have been added to the Schema Language.

(1) The first is a Global Clause in which the Data Administrator can select Before, After and/or Quick-Before looks for all areas.

(2) The second is a clause which specifies looks to be taken for each individual area, and is in each AREA description.

5.6. PROGRAMMER SPECIFICATION OF LOOKS

A run unit can select either Command Quick-Before looks, or Run Unit Quick-Before looks. These two options are available in the SAVE DATA clause of the SCHEMA section of the Data Division of the run unit. These looks are taken so that the database will always be restored to an accurate condition at the termination of the run unit, or at the termination of a command.

If a run unit has selected command rollback capability by specifying command Quick-Before looks, then a Quick-Before look must be taken prior to the first alteration of the page by each command. If command rollback is not selected, then only run unit rollback is required and only one Quick-Before look prior to the first alteration of each page is required for the run unit.

6. PRESERVING THE INTEGRITY OF THE DATABASE

6.1. GENERAL

A major objective in a database implementation is to preserve the accuracy of the data from possible destruction. Individual items, records, pointers, or areas may possibly be destroyed or incorrectly modified by hardware, software, or operational errors. DMS 1100 provides capabilities for the restoration of the database after such errors have occurred.

The vehicle for providing database integrity is the page "look". A look is a copy of a database page which is written to a file for use at a later point in time, if it becomes necessary to re-establish the contents of the database. The point in time a look is taken, its disposition, and later usage comprise the DMS 1100 database protection system.

The type of database protection to be utilized is a decision which is to be made by the Data Administrator. The options available range from no protection for the database, to complete protection by the usage of all supplied routines. The type of protection is specified at system's generation and by the types of looks to be taken for areas, as specified in the source schema. Looks may be taken prior to alteration of a page (Before looks) and following the page alteration (After looks). The Before looks are used to move backward over time to remove the effects of an error on the database. After looks are used to move forward over time to re-establish the database to a stable state.

The type of looks to be taken is specified on an area basis in the schema. The Data Administrator is responsible for assuring areas which are spanned by sets have the same level of looks. The Schema Report of the DDL may be used as a guide for this analysis. If different levels of looks are specified for such areas, an unstable database may result after recovery.

The Data Administrator may request that Quick-Before looks be taken for specific areas or all areas. This requires previous allocation for Quick-Look files at system generation. The names of the Quick-Look files, their sizes, and the type of random access, sector-addressable, on-line devices on which they will reside are all specified by the Data Administrator at system's generation. One or more files may be specified for receiving Quick-Before looks by this manner. The Quick-Look files are cataloged as part of the Quick-Look files initialization (see 4.3) and must be assigned as catalogued files by user (DML) programs. The Quick-Look files contain only Before looks. These files are used by the Rollback mechanism and for Quick Recovery. Rollback is the procedure whereby the changes made to a database by a run unit, or a command of a run unit, are reversed. This may be caused by user program request, by the DMR detection of a user error, or by the DMR deadlock resolution mechanism. Quick Recovery is a procedure which is initiated by an Executive run stream. This procedure causes all Before looks in the Quick-Look files to be applied to the database, thereby reversing the effects of all run units which have IMPARTed, but not DEPARTed.

The Data Administrator may request an Audit Trail tape be assigned to the TIMER activity. This cataloged tape file is used to store After and/or Before looks for those areas, which have been designated by the Data Administrator in the source schema as requiring such looks. The establishment of an Audit Trail tape file is made through system's generation and the System File initialization procedure (see 4.2). The looks contained on the Audit Trail tape file are applied to a reloaded database by the Long Recovery procedure, to re-establish the database to a specified point in time. This technique requires that the database, or portions of the database be periodically dumped to tape or removable disc, to be used at a later point in time in the event that Long Recovery be needed to re-establish the database.

Note that, the emphasis of integrity is at the database level as opposed to the run unit level. Consequently, the responsibility for determining what types of database protection are to be employed and what special actions are required (see 6.5) are determined at the Data Administrator level (*system's generation and schema description*). Only limited control is left to the user program, e.g., specification of COMMAND QUICK-BEFORE-LOOKS vs RUN-UNIT QUICK-BEFORE-LOOKS and usage of the LOG and DEPART WITH ROLLBACK commands.

Note that DMS 1100 areas are Executive files, and all FUR/PUR routines and SECURE can be used under Executive control to aid the Data Administrator in protecting the database.

6.2. ROLLBACK

ROLLBACK is the procedure which reverses the effects of a run unit, or command of a run unit on the database and on the internal tables of DMR. In order for ROLLBACK to occur the Quick-Look files must be:

(1) Defined at system's generation,

(2) Cataloged by Quick-Look file initialization,

(3) Assigned to the run unit involved, and

(4) Must contain QUICK-BEFORE looks for all altered areas as specified in the source schema.

The determination of whether the entire run unit, or only the particular command which is found to be in error is to be reversed, is based upon the user programs specification to the DML Preprocessor. Rollback may also be caused in order to resolve a facilities deadlock condition. In this latter case, run unit Rollback is performed regardless of the user's program specification. Rollback will occur under the previous conditions without any further action on the part of the Data Administrator, operator, or run unit.

Rollback is accomplished in two phases.

(1) External Rollback is that phase which applies the Before looks from the Quick-Look files to the database.

(2) Internal Rollback releases buffer space which has been acquired by the run unit, or command of a run unit, and re-establishes all internal tables of DMR which were altered by its processing.

While Rollback is being performed on a specific run unit, other run units may be accessing the database through the DMR.

A Rollback Paragraph is required in all DML programs. Control will be returned to this paragraph after run unit Rollback has been performed. When control reaches the Rollback Paragraph, a "DEPART" has taken place internal to the DMR as far as the user program is concerned.

A user error for a command which has not involved an alteration of the database will cause control to go to the Error Paragraph if specified. If none is specified, control will be returned in-line. The appropriate error status is returned in either case. If previous commands have caused alteration to the database, the user program is responsible for determining if DEPART WITH ROLLBACK is required.

A user error on a command which is detected after database alteration has occurred will cause command or run unit Rollback to occur. If QUICK-BEFORE looks do not exist for all areas altered, the database will be left in an inaccurate state. Internal Rollback takes place regardless of whether or not external Rollback occurred.

When it becomes necessary to resolve a facility deadlock, the run unit having the lower priority, either explicity stated in the user program or through the default priority technique, will be rolled back. This Rollback will be run unit Rollback regardless of whether or not command Before looks are specified in the user program. If Quick-Before looks do not exist for all areas altered, the database will be left in an unstable state. Internal Rollback will always take place and control will be returned to the user program's Rollback Paragraph with a status indicating it was rolled back due to deadlock resolution. An effective "DEPART" will have been performed before the return is made to the Rollback Paragraph. If the user program wishes to reattempt access to the database it must issue an IMPART.

The user program may gain control in his Rollback Paragraph under other conditions. One such condition is the case where the DMR has detected an error condition which is not caused by any particular run unit. Certain I/O error conditions, for example, must be thought of as "global" in nature and require that all run units be removed from the DMR. Under these conditions, control is returned to each run unit at its Rollback Paragraph after an effective "DEPART" has been performed. Flags are set in the DMR to prohibit any IMPART command from being processed until the effects of all the run units have been reversed. This reversal is accomplished by giving the operator a message requesting the initiation of Quick Recovery (see 6.3) to accomplish the restoration of the database, while the restoration of the internal tables and buffer areas of DMR is accomplished by a reload of the DMR REP from mass storage.

NOTE:

This reload will take place under these conditions even if the parameter requesting that the REP be unconditionally held in main storage was specified at systems's generation.

As in previous discussion, an inaccurate database will result if pages were altered for which no Quick-Before looks existed.

6.2.1. Run Unit Rollback

Run Unit Rollback is the mechanism which reverses the effects that a run unit previously had on the database and the internal tables of DMR since its IMPART or last FREE command. Run Unit Rollback for a particular run unit may take place without impairing the access to the database by other run units. Control is returned to the user program at its Rollback Paragraph after an effective "DEPART" has been performed by DMR. The returned error status will indicate the reason run unit Rollback was performed.

Run Unit Rollback could be performed for three reasons:

(1) An error was detected in a command after alteration of the database had occurred and command Before looks were not specified by the run unit,

(2) Run Unit Rollback was performed to resolve a facility deadlock with another run unit, and

(3) The operator elected to remove the run unit from the DMR via an II keyin.

In this latter case, such action may have been prompted by a run unit time-out message from the DMS TIMER activity to the operator console.

In order for the database to be stable after run unit Rollback has occurred, Quick-Before looks must be specified in the source schema for all areas altered by the run unit. If Quick-Before looks are not specified for all such areas, no indication will be given as to which areas are not reestablished. If run unit Rollback is in progress and an unresolvable I/O error occurs during the read of the Quick-Look file or the write to the database, the operator is given a message to initiate Quick Recovery.

6.2.2. Command Rollback

Command Rollback is the mechanism which reverses the effects that a run unit previously had on the database and the internal tables of DMR since the issuance of a particular command which is found in error. Command Rollback for a particular run unit may take place without impairing the access to the database by other run units. Control is returned to the run unit at its Error Paragraph if one is specified, or in-line if no Error Paragraph is specified. An error status will be available to the user program in either case. When the user program regains control after command Rollback has occurred, it is possible to issue another command to DMR. This is opposed to run unit Rollback whereby an internal "DEPART" has been performed before the user program regains control.

The only conditions under which external command Rollback will be successfully performed are as follows:

(1) The Quick-Look files exist,

(2) The schema has directed that Quick-Before looks be taken,

(3) The user program has specified that command Quick-Before looks be taken, and

(4) An error was detected in a command after alteration of the database had occurred.

Internal command Rollback will be performed in any event and the database may be in an unstable state if any of the above conditions are not present.

In order for the database to be stable after command Rollback has occurred, Quick-Before looks must be specified in the source schema for all areas altered by the command. If Quick-Before looks are not specified for all such areas, no indication will be given as to which areas are not reestablished.

Specification of the command Quick-Before looks may cause additional looks to be taken. For run unit Rollback, one copy of a page is taken before its first alteration by the run unit. No other copy of that page will be taken even though it may be altered several times by subsequent commands in the run unit. If command Quick-Before looks are specified by a user program, a copy of a page is taken before its first alteration by each command. At the end of each command, the duplicate copies of the page are not saved, if it is determined

that Rollback of the command is not required. Only the first look of the page (*the same one that is saved for run unit Rollback*) is saved.

NOTE:

If run unit Rollback is required when command looks exist in the Quick-Look files, only that copy of a page which was taken prior to its first alteration will be applied to the database.

If command Rollback is in progress and an unresolvable I/O error occurs during the read of the Quick-Look file or the write to the database, the operator is given a message to initiate Quick Recovery.

6.3. QUICK RECOVERY

Quick Recovery is a procedure to be used when a DMR internal error occurs (e.g., the DMR detects an unresolvable I/O error while attempting to access the database), or when a system failure is encountered. The results of a Quick Recovery are to restore the database to its original state prior to the operation of all run units which were active at the time of the failure. Quick Recovery applies the Quick-Before looks of these run units to the database. Only the first Quick-Before look of any page is used, as is done for run unit Rollback. Note that only those areas having Quick-Before looks are restored by Quick Recovery. As the database is restored to a stable state, the restored pages are written to the Audit Trail as After looks.

Quick Recovery is initiated external to the DMR. It is initiated from a separate EXEC run stream as outlined in Appendix B.2. This run stream should be placed into the cataloged file DMRMT and it may be STARTed from the operator's console. In the event of an internal DMR error, a message is given to the operator directing him to initiate Quick Recovery. If a system failure occurs, the first run unit to IMPART after the system has been rebooted will cause the TIMER activity to detect that run units were active at the time of the failure by inspection of the System file, and then will generate an internal START of the run stream which initiates Quick Recovery. Any run unit attempting to IMPART while Quick Recovery is in progress will have control returned to its Rollback Paragraph, including the first run unit after a system failure, previously described. Consequently, it may be desirable to establish an operating procedure to initiate Quick Recovery immediately after the system has been rebooted and before the first run unit attempts to IMPART.

Quick Recovery will not be allowed to alter the database or Quick-Look files except under the preceding conditions. If it is improperly initiated, an error status will be displayed on the operator's console and Quick Recovery will be terminated.

Quick Recovery notifies the operator of the run unit ID's of all run units which were backed out of the system.

NOTE:

In the event of a system failure, the run units are not available to receive this information from DMR.

The DMR will release itself from main storage, and the next run unit to IMPART will cause a fresh copy of the DMR to be loaded. At this point the system is again in operation.

6.4. LONG RECOVERY

Long Recovery is the means of restoring a database from a dump of the database up to a specified point in time. To use Long Recovery, dumps of all or selected areas of the database must be taken and an audit trail tape must be employed which contains After looks, Before looks, and Recovery Points, or After looks and Checkpoints.

Long Recovery may be used to reestablish a database after one or more files (*areas*) have been lost due to hardware, software, or operational failure. It may also be used to reestablish one or more areas which have been improperly modified by a run unit. Partial Long Recovery for reversing run unit error must take into account any impact that bad data might have had on run units processed after the error occurred, i.e., later run units may have operated on bad data.

Long Recovery is accomplished by loading a database dump and initiating the Long Recovery routine. This routine will apply After looks from the audit trail tape to the reloaded database up to a designated Recovery Point or Checkpoint. The Before looks for all run units active if a Recovery Point is designated are then applied to the database. A list of all run units being "backed out" of the system is given to the operator. Since Checkpoints are placed on the Audit Trail when run units are not active, "backing up" is not required if they are employed and Before looks need not be taken. After the completion of Long Recovery, processing of run units can be reinitiated. It is possible to go back any number of database dumps and use Long Recovery to any Recovery Point or Checkpoint without necessitating the loading of any intervening dumped files.

6.4.1. Database Dumps for Recovery

A DMS area is synonymous with an Executive file. Several processors are available for moving the text of a file from random access mass storage to tape or vice versa. Consequently, no special DMS utility routines are provided for this purpose. The SECURE processor of the Executive can be effectively used in saving and restoring the database or any areas within the database.

In order to use Long Recovery, the database dump must be taken at a point in time where run units are not accessing the database. Two DMS utility routines are provided to ensure that run units are not accessing the database while a dump is in progress and to "open up" DMS to run units after the dump has been completed.

An example run stream is given in Appendix B.3, which shows the sequence of activities necessary for taking a database dump. Three basic steps necessary are as follows:

(1) The execution of DTBSDMP1,

(2) The movement of the text of DMS areas to tape, and

(3) The execution of DTBSDMP2.

When DTBSDMP1 is executed, DMR is informed that database dumps are to be taken. DMR allows all active run units to continue processing, but will return an error status to any new run unit attempting to IMPART. When all the run units which were active have completed, DMR updates the System file and returns control to DTBSDMP1. DTBSDMP1 obtains a new audit trail tape reel number and writes it into the System file. An End of Reel block will be written to the previous audit trail tape reel in the audit trail which stemmed from the previous database dump. All additional information in the System file pertaining to the audit trail tape is updated before DTBSDMP1 terminates.

The movement of areas from random access mass storage to tape may be accomplished by use of the SECURE processor or @COPY,G. The Data Administrator is responsible for maintaining the reel numbers involved with a database dump for use when the database is reloaded for Long Recovery. The entire database or any number of selected areas may be dumped. If a set spans several areas, these areas should always be treated as a unit for the purposes of taking dumps and reloading these dumps. The System file must always be dumped, since it contains updated information from DTBSDMP1 which is required for Long Recovery. For a complete discussion on the SECURE processor, the user is referred to the *UNIVAC 1100 Series Operating System Programmer Reference, UP-4144* (current version).

The execution of DTBSDMP2 following the taking of the dump is required to inform DMR that it may accept new run units once again.

The routines DTBSDMP1 and DTBSDMP2 should only be used when the dump is required to reload the database for Long Recovery. These routines are not to be thought of as a general means to "stop" and "start" DMR from accepting run units, since they modify the System file with the assumption that a dump is being taken for use in the event that Long Recovery is necessary. Note that the EXEC run ID of the run which takes the dump must be DATDUM, as this is checked for validity when DTBSDMP1 makes a special entry to DMR.

The Data Administrator is responsible for determining the policy to be followed in dumping the database. It may be decided to dump the entire database at periodic intervals, or to dump the entire database at longer intervals and only the highly "active" areas at shorter intervals. Maintaining the tapes containing dumps and a record of what areas were dumped, and when they were dumped is the responsibility of the Data Administrator. As previously mentioned, every dump must contain the System file, and areas spanned by sets must be treated similarly. For additional considerations concerning database dumps, see 6.5.

6.4.2. Audit Trail Tape

The audit trail tape is a cataloged tape file. The initial cataloging is accomplished by the System File initialization routine (see 4.2), if the system's generation parameters specify that an audit trail tape is to be used. The audit trail tape is thereafter assigned to the TIMER activity which does the I/O. The audit trail tape contains Before and After looks as directed by the Data Administrator in the schema. Additionally, a Start and End of run unit sentinel is written to the audit trail tape by an IMPART and DEPART, respectively. The FREE command causes a special block which effectively acts as an End of run unit, followed by a Start of run unit to be written to the audit trail tape. The audit trail tape may also contain Recovery Point sentinels or Checkpoint sentinels.

6.4.3. Recovery Points

A Recovery Point is a point in time to which the system can return during the Long Recovery operation. When the recovery operation is complete, the system has regenerated an accurate database. A Recovery Point record is a sentinel written on the audit trail. Run units are active when a Recovery Point record is written.

A Recovery Point may be generated by a run unit via the LOG command, or may be placed on the audit trail by the automatic generation of Recovery Point feature of DMS 1100. The latter method requires the Data Administrator to specify, in the schema, the frequency of Recovery Points on the audit trail. The value specified will be the number of blocks written to the audit trail between successive Recovery Points. A schema language clause is available for this function. Additionally, the system's generation parameter which causes the code for automatic Recovery Points to be assembled into the DMR must have been on when the system was built.

6.4.4. Checkpoints

A Checkpoint is similar to a Recovery Point except that it is written when run units are not active in the DMR. Long Recovery may be requested to recover to a Checkpoint in the same manner as to a Recovery Point. When Long Recovery reaches a Recovery Point, however, it "backs out" the run units which were active at the time it was written by use of Before looks. Since no run units are active when Checkpoints are written, Before looks are not referenced by Long Recovery on recovery to a checkpoint.

It is suggested that either Recovery Points or Checkpoints, but not both, be employed at an installation. The decision as to which to use is dependent upon the impact of temporarily stopping database accessability in order to write a Checkpoint, versus the impact of writing Before looks to the Audit Trail and using Recovery Points.

A separate program, CHECKPOINT, is provided on the DMS 1100 release tape for writing a Checkpoint on the Audit Trail. This program is initiated from within a run stream as shown in Appendix B.5. This run stream must be initiated each time a Checkpoint is written on the Audit Trail.

When CHECKPOINT is executed it interfaces with DMR such that all currently active run units are allowed to continue processing to completion and any new run units are queued when they attempt to IMPART. This forces all run units to be drained from the system. When the last run unit completes, a Checkpoint sentinel is written to the Audit Trail and all run units which were queued are dequeued.

6.4.5. Running Long Recovery

Long Recovery is activated by a run stream and must assign those files (areas) which are to be recovered. Appendix B.4 shows an example of the run stream required to initiate Long Recovery. After the files to be recovered have been assigned, three basic tasks must be performed by this run stream:

(1) DMR must be informed that Long Recovery is to be initiated,

(2) The appropriate database dumps must be loaded, and

(3) The Long Recovery routine must be executed.

In addition to the system's generation requirements for the audit trail tape, two other system's generation parameters must be defined for the Long Recovery routine.

(1) MNAREA must be given a value equal to or greater than the number of areas for which Before and/or After looks are taken.

(2) BUFSZR is the system's generation parameter which must be given a value at least as large as the length of the largest look (*page*), which will be read by the Long Recovery routine.

If either of these parameters are changed the Long Recovery routine, LONGRECOVERY, must be reassembled and collected.

DMR is informed that Long Recovery is to be initiated by the execution of the routine DMRACT. If run units are active at this time, they will be given an error status and returned at their Rollback Paragraph. While the Long Recovery procedure is in progress, from the execution of DMRACT to the completion of LONGRECOVERY, any run unit attempting to IMPART will be given an error status and returned to its Rollback Paragraph.

As was the case with the taking of database dumps, the loading of database dumps is accomplished by the use of EXEC processors, such as SECURE or COPY. The System File must always be reloaded from the dump. Reloading database areas is a decision of the Data Administrator. However, ensure that if looks are to be applied to an area, (*i.e., the corresponding file is assigned in the Long Recovery run stream*) that area should be reloaded from the dump, and conversely, if an area is reloaded its looks should be applied.

When LONGRECOVERY is executed, it internally assigns the audit trail tape and causes the mount message for the proper reel number to be generated. The reel number required is obtained from the reloaded System File. LONGRECOVERY then applies After looks from the audit trail tape to the database. If a look is found for an area which is not assigned to the Long Recovery run stream, a message will be given to the operator asking if Long Recovery should be continued or aborted. If the operator answers to continue, that look, and all other looks for the area, will be ignored. No additional messages will be generated for that area. Areas which were not reloaded from the database dump should not be assigned to the run stream which initiates Long Recovery, since an unstable database may occur if they are assigned.

LONGRECOVERY will apply After looks from the audit trail tape to the database for assigned areas (*files*) until the specified Recovery Point is reached. It then applies Before looks by moving the tape backwards to "back out" those run units which were active at the time that the Recovery Point was taken. These run units are backed up to their Start of run unit sentinel or to the last FREE sentinel which they caused on the audit trail tape. The run unit ID's for all run units backed out are displayed on the operator console. If a Checkpoint is specified, the "back out" procedure is not required (see 6.4.4).

Note that LONGRECOVERY does not attempt to determine if both Before and After looks for a given area are on the audit trail tape. This is the responsibility of the Data Administrator. A diagnostic message is generated in the DDL if a schema definition specifies After looks but no Before looks, or vice versa. If an area does not have After looks on the audit trail tape, it will not be recovered. If it does not have Before looks, the run units which were active at a Recovery Point may not be properly "backed out". The latter case is permissable if Checkpoints are employed.

The Recovery Point or Checkpoint to which Long Recovery is directed to return the database may be specified as a data image immediately following the @XQT LONGRECOVERY statement. If this directive is not given in the Long Recovery run stream or if it is improperly formatted, the operator is requested to supply this directive. As each Recovery Point or Checkpoint is written to the audit trail tape, the time at which it was written and its sequence number relative to the last database dump is displayed on the operator console. The directive given to LONGRECOVERY from the run stream or from the operator may specify either the time or sequence number of the required Recovery Point or Checkpoint.

If the Recovery Point or Checkpoint is specified on the basis of time, the format of the directive is:

MMDDYYHHMMSS

and must begin in column one,

where:

MM IS THE MONTH (01, 02, . . . 12)
DD IS THE DAY (01, 02, . . . 31)
YY IS THE YEAR (72, 73, . . .)
HH IS THE HOUR (00, 01, . . . 23)
MM IS THE MINUTE (00, 01, . . . 59)
SS IS THE SECOND (00, 01, . . . 59)

If this specification is used, the specification is validated ($0 < MM \leq 12$, etc.) and the application of After looks to the database is initiated. If database dumps were taken between the time of the loaded dump and the specified Recovery Point or Checkpoint time, they will be by-passed without requiring any intervention by the operator. Hence, it is possible to go back further than the last database dump for the "starting point" of Long Recovery. When a Recovery Point is reached which has the time specified in the directive, the active run units are "backed out" as previously discussed.

If a Recovery Point is reached which has a time that is greater than that specified in the directive (e.g., Recovery Points exist at 2:00 and 4:00 for the specified day and the directive specifies 3:00), the operator is asked if the later time should be accepted. If this time is not acceptable, Before looks are applied to get back to the earlier Recovery Point. In either case, the run units active at the selected Recovery Point are "backed out". This procedure cannot be used if Checkpoints are employed and Before looks are not specified.

If the specification of the Recovery Point or Checkpoint is to be made on the basis of sequence number, the format to be used is:

SEQnnnnnnAd

This specification must start in column one.

nnnnnn — is the sequence number of the required Recovery Point or Checkpoint. Six characters are required. The number may be left justified and blank filled, or right justified and zero-filled. Hence, sequence number 12 may be specified as 12bbbb or as 000012.

d — is the number of dumps to be by-passed before LONGRECOVERY becomes sensitive to Recovery Points or Checkpoints. (*The sequence number is relative to the previous dump.*) 0 must be specified if no dumps are to be by-passed, e.g., the last dump which was taken is the dump which was loaded.

If an I/O error occurs while LONGRECOVERY is applying After looks to the database (*moving forward over time*) the operator is asked to which Recovery Point it should back up. If for example, the directive to LONGRECOVERY had specified a Recovery Point by time which was greater than the last Recovery Point on the audit trail tape, this message may be answered by specifying that last Recovery Point.

NOTE:

This will not be possible if Checkpoints are used and no Before looks exist on the Audit Trail. LONGRECOVERY should be reinitiated specifying an earlier Checkpoint.

Once LONGRECOVERY has reached the specified Recovery Point and is in the process of backing out the run units which were active by applying Before looks to the database, an I/O error causes the abort of LONGRECOVERY. If this occurs, the Long Recovery run stream must be re-initiated with a directive to recover to an earlier Recovery Point.

After the completion of LONGRECOVERY, it is not mandatory that a database dump be initiated. LONGRECOVERY will initiate a new reel number of the audit trail tape when the specified Recovery Point or Checkpoint is reached. If a Recovery Point was specified, as the Before looks of the active run units are applied to the database, these Before looks are written to the new reel of the audit trail tape as After looks. For either Recovery Points or Checkpoints, an End-of-Reel mark is written to the old reel following the Recovery Point or Checkpoint which was used by Long Recovery. By this procedure, the audit trail tape is updated such that

it may be "reused" at any later point in time. Run units active at a Recovery Point and all run units which were initiated after the specified Recovery Point or Checkpoint must be re-initiated.

At completion of LONGRECOVERY, the System File is updated to reflect the new status of the audit trail tape and to enable DMR to accept new run units.

Three additional points on Long Recovery are as follows:

(1) LONGRECOVERY takes no special action based upon whether a Recovery Point was placed on the audit trail tape via a LOG command or by the automatic generation of Recovery Points feature of DMS.

(2) The run stream which initiates Long Recovery must have the EXEC run ID of LONREC, as a validity test is made on this ID before Long Recovery is initiated.

(3) If both Checkpoints and Recovery Points are employed on an Audit Trail the procedures governing Recovery Points should be followed.

6.5. ADDITIONAL RECOVERY CONSIDERATIONS

The DMS 1100 has been designed and implemented with database integrity in mind. The Data Administrator is left with the responsibility of determining what level of protection is necessary for areas, what is the frequence of Recovery Points/Checkpoints on the audit trail (*if one is employed*) and the method and frequency of taking database dumps.

This section presents some additional considerations for the Data Administrator in his task of ensuring database integrity.

6.5.1. Initial Load of Areas

Prior to level 4 of DMS 1100, several special recovery considerations were necessary when areas were initially loaded. These have been eliminated. The only remaining consideration is that when pages within an area are initialized (*either by the INITIALIZE utility command or by dynamic initialization*) no test will be made to determine if germane data existed on the mass storage file prior to establishing the pages.

With level 4 and later DMS 1100 levels, it is not necessary to dump areas after they are initially loaded. LONGRECOVERY will properly establish such areas even when started from a database dump which occurred prior to the existence of such areas, provided they are assigned to the LONGRECOVERY run stream.

It is always possible to re-initiate processing from the point of the last FREE command if run unit rollback or quick recovery is performed on a run unit which has areas OPEN for INITIAL LOAD, or if LONGRECOVERY is performed to a Recovery Point which was taken while an INITIAL LOAD was in progress.

6.5.2. Usage of the Page Expansion Utility

Since the DMS 1100 recovery system is based upon the concept of page looks, and the page expansion utility function changes the size of pages, the procedures described in Section 7.5 should be followed to ensure database integrity.

7. DMS 1100 UTILITY PROCESSOR

7.1. GENERAL

The DMS 1100 Utility Processor consists of a collection of routines for performing a variety of utility functions upon a database as directed by the user. A Data Management Utility Language is provided for the specification of the functions to be performed and the portion of the database which is to be operated upon.

The utility processor must be interfaced with a particular database before any functions can be used. (See 7.2.) This interface need only be performed once for any schema. The utility processor may then be called (see 7.3), to perform the various functions discussed in Section 7.5.

7.2. INTERFACING THE UTILITY PROCESSOR WITH A DATABASE

The routines within the utility processor are designed to interface with the DMR to ensure proper data locking and recovery. This interface essentially consists of establishing the collection of utility routines as a DMR run unit. This run unit is capable of performing the utility functions as "special" commands. The code required for these special commands is contained in the D-bank of the established run unit as opposed to the DMR itself. Hence, the processor requires main storage only when one or more of the utility functions are being performed.

In order to use the utility functions, the utility routines must be established as a DMR run unit. The procedure to accomplish this is to modify the skeleton DML program, MAIN, which is supplied on the DMS 1100 release tape to incorporate schema dependent information into the INVOKE. The run stream required is given in Appendix B.6.

The supplied version of MAIN specifies that COMMAND QUICK-BEFORE-LOOKS are to be taken. If rollback is required from within the utility processor, the database will be returned to the status it was in prior to the issuance of the last utility command for all areas which have QUICK-BEFORE-LOOKS specified in the schema.

7.3. CALLING THE UTILITY PROCESSOR

After interfacing the utility processor with a database (see 7.2), the utility functions are requested from within a run stream which follows the same conventions as a run stream of a user developed run unit. The following is an example of such a run stream:

```
@RUN
<Control statements to assign and load DMU into a file if not part of a
    system library>
@ASG,A )
   .    }  @ASG of areas, schema, quick-look files
   .    }
   .    )
@file.DMU,options   element-name
<utility processor commands>
@FIN
```

The options allowed on the processor call are as follows:

I	– Source input is from cards or card images in the control stream.
U	– Source input is to be updated producing new cycle of input element.
Neither I nor U	– Source input is to be updated, however, no new cycle of element will be created.
L or S	– These options may be specified, however a listing of (updated) source input will be produced if the N option is not specified.
N	– Suppress listing of source input.

The element-name specifies a program file element in standard EXEC format.

7.4. SYNTAX COMPONENTS AND NOTATION

The syntax of the Data Management Utility (DMU) Processor consists of basic syntax clauses and combinations of these clauses. These combinations, through definition and convention, are used to communicate requests for utility processing of the database.

This section introduces the components of these clauses and the notation used to indicate allowable syntactical combinations of these clauses. The DMU examines all clauses for syntactical completeness and prints a syntax error message should a violation occur.

7.4.1. DMU Syntax Components

This section describes the basic component of a clause (*the character set*) and continues the description through the clause to the most complex structure (*the DMU command*).

7.4.1.1. Character Set

The DMU character set contains 40 characters that consist of letters, digits, symbols, punctuation marks, and the space. The following list itemizes the 40 character set :

0,1,. . . ,9	digits
A,B,. . . ,Z	letters
␣	space
-	hyphen
,	comma
.	period

7.4.1.2. Word

A DMU word is a contiguous (*no embedded spaces*) sequence of not more than 30 characters. Each character in a word must be from the set 'A', 'B',. . . ,'Z', '0', . . . ,'9', and '—', and neither the first nor last character in a word can be '—'. The following are words: 'HERE-AFTER', '22', 'A9C8F7J', and 'A'. The following are not words: 'A,BC', '-AT-THE', '␣', and 'AB0␣'.

7.4.1.3. Name

A DMU name is a word that names or identifies an entity within a clause. The use of a name and its meaning is a function of its syntactical position. DMU has three types of names which are as follows:

(1) record-name

(2) area-name

(3) set-name

7.4.1.4. Reserved Word

Any DMU word whose syntactical usage is restricted and cannot be supplied by the user is a DMU reserved word. A complete list of DMU reserved words is given in Appendix D.

7.4.1.5. Literal

A DMU literal is a word, supplied by the user whose purpose is to convey explicit information, as opposed to identification information such as names. Thus, a literal is a string of characters whose value is implied by the ordered set of characters, of which the literal is composed. DMU literals are used to specify numeric values in DMU syntax.

Example:

'2' represents the value two.

7.4.1.6 Clause

A DMU clause is an ordered sequence of words and characters; the last character is an optional punctuation mark, space or period. Clauses are the basic units of DMU syntax that communicate functional pieces of information.

7.4.1.7. Command

A DMU command consists of one or more clauses and constitutes a syntactically complete description of a request for the performance of a utility function.

7.4.2. DMU Syntax Notation

This section describes the notational symbols and rules that are used to guide the construction of syntactically correct DMU commands. These conventions apply to all DMU syntax skeletons specified in Section 7.5.

- The elements which make up a syntax skeleton are upper case words, characters (*spaces included*) and notational symbols. Their skeleton order, left to right, and next line, must be followed in forming DMU commands.

- All upper case words represent DMU reserved words. Underlined upper case words are required in each specification of the syntax component. Non-underlined upper case words are individually optional. Each may be included or omitted in source language specifications.

- Lower case words (*in italics*) are generic terms, names, or literal types, that must be replaced in the actual source specification with a name or literal. Generic terms sometimes have integers appended; this is to avoid confusion in case two generic terms in one command are of the same type.

- The use of a period is required to denote the completion of a DMU command specification. The commas and spaces in a skeleton must be used as indicated by the skeleton.

- When one or more syntax components are enclosed by special notational symbols, they should be interpreted as follows:

  [a / b / c] — A source specification does not require a syntax in this position, but can include either a, b, or c.

  {a / b / c} — A source specification must include exactly one of the components a, b, or c.

  . . . — Repeated choices may be made and included in a source specification, one after the other.

- All underlined **(LARGE BOLD FACE)** upper case words must begin on a new card. The first character may appear in any column. All words must begin and end between card columns 1 and 72. More than one card can be used to specify the words between two **(LARGE BOLD FACE)** underlined words. The content of column one of a continuation statement is considered to immediately follow the last non-blank character of the previous statement. A period must terminate a command and the indicated word order must be preserved.

7.5. DATA MANAGEMENT UTILITY LANGUAGE (@DMU)

Any number of DMU commands may be specified following the DMU processor call statement. No physical restrictions are imposed upon the order in which these commands appear. Clauses within commands, however, must appear in the order specified in the syntax skeleton.

The utility processor will perform the utility functions in the order in which the commands appear. Additionally, the utility processor will OPEN all areas required for a specific function prior to its execution and CLOSE these areas at its completion.

If an error is detected in a command specification, a diagnostic message will be given and all subsequent commands will be analyzed but not executed.

NOTE:

In the demand mode, the processor call statement should be resubmitted in this case.

If an error is detected during the performance of a utility function upon a database, the utility processor may be directed to proceed to the next command or to stop accepting commands by an ON ERROR clause of the statement. Since the DMU interfaces with the database as a run unit, the same level of database protection is used as for any other run unit (see 7.2).

The utility processor will issue an IMPART to the DMR prior to accepting the first command and a DEPART following completion of the last command.

The complete command skeleton is presented in the following section. Subsequent sections present detailed information on each individual command. Diagnostic messages and their descriptions are given in Appendix C of this document.

7.5.1. Complete Syntax Skeleton

```
COMPACT { DATABASE                                                  } [ OUTPUT STATUS ]
        { { AREA  } area-name–1 [ , area-name–2 ] ... }
        { { AREAS }                                   }

        [ ON ERROR { STOP    } ] .
                   { PROCEED }

EXPAND    AREA  area-name [ EXISTING PAGES integer ]

        [ ON ERROR { STOP    } ] .
                   { PROCEED }
```

```
INITIALIZE AREA area-name [BEGIN AT PAGE page-number]
        [ ON ERROR { STOP   } ] .
                   { PROCEED}

PATCH AREA area-name PAGE page-number WORD displacement
        [ { RELATIVE } TO { TOP          } ]
          { REL      }    { BOTTOM       }
                          { RECORD record-number }

          { REPLACING } integer–1 WITH integer–2
          { REP       }

        [ ON ERROR { STOP   } ] .
                   { PROCEED}

PRINT AREA area-name [ { integer } WORDS ] [ DATA IS { ASCII     } ]
                     { ALL     }               { FIELDATA }

        [BEGIN AT PAGE page-number–1] [END AT PAGE page-number–2]
        [ ON ERROR { STOP   } ] .
                   { PROCEED}
```

Format 1:

```
VERIFY DATABASE [ { AREA  } area-name–1 [OWNER] [ , area-name–2 [OWNER] ] . . . ]
                { AREAS }

        [MAXREC OF integer] [ ON ERROR { STOP   } ] .
                                       { PROCEED}
```

Format 2:

```
VERIFY SET set-name { AREA  } area-name–1 [OWNER] [ , area-name–2 [OWNER] ] . . .
                    { AREAS }

        [MAXREC OF integer] [ ON ERROR { STOP   } ] .
                                       { PROCEED}
```

Format 3:

```
VERIFY SET-OCCUR OF set-name { AREA  } area-name–1 [OWNER] [ ,area-name–2 [OWNER] ] . . .
                             { AREAS }

        WITH DBK AREA area-name-k1 PAGE page-number-k1 RECORD record-number-k1

        [ ,AREA area-name-k2 PAGE page-number-k2 RECORD record-number-k2 ] . . .

        [MAXREC OF integer] [ ON ERROR { STOP   } ] .
                                       { PROCEED}
```

7.5.2. Page Compaction

The page compaction utility provides an "off-line" removal of deleted records from within pages. When records are deleted from a database they are flagged as being deleted, but are not physically removed from the database until space is needed for a new record on that same page. A page will automatically be compacted by the DMR when deleted records exist and space is required, however, this is done "on-line", that is, when the new record is being stored. The page compaction utility is provided to allow this function to be performed during time periods of low database activity.

The command may also request that detailed statistics on the record space utilization of pages be displayed for each page of the area.

Total and average utilization statistics for all pages will always be displayed for each area which is compacted.

7.5.2.1. Command Syntax

The command syntax for page compaction is:

```
COMPACT { { DATABASE                                                    }
        { { AREA     } area-name–1 [,area-name–2] . . .                 }
        { { AREAS    }                                                  }

[OUTPUT STATUS] [ ON ERROR { STOP    } ] .
                           { PROCEED }
```

A specification of DATABASE will cause the physical removal of deleted records and record space compaction on all pages in all areas of a database. A selected list (*one or more*) of areas may be specified through the AREA clause. If statistics are required on the record space utilization of each page, the OUTPUT clause should be specified. The ON ERROR clause operates as described in Section 7.5.

7.5.2.2. Output

The normal completion of the page compaction utility will cause the message:

PAGE COMPACTION SUCCESSFULLY COMPLETED.

If errors are detected in command syntax or while performing the compaction function, diagnostic messages will be produced. Appendix C describes all such diagnostics which could occur. If the OUTPUT clause is specified on the COMPACT command, the following output will be given for each area operated upon:

AREA USAGE REPORT FROM PAGE COMPACTION

AREA area-name-1

PAGE NUMBER	TOTAL NO. OF WORDS	UNUSED WORDS	USED WORDS	PERCENT UNUSED
1	t_1	a_1	u_1	p_1
2	t_2	a_2	u_2	p_2
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.

SUMMARY STATISTICS	NO. OF PAGES	TOTAL NO. OF WORDS	AVERAGE UNUSED PER PAGE	AVERAGE USED PER PAGE	AVERAGE PERCENT UNUSED PER PAGE
DATA PAGES	n_d	t_d	a_d	u_d	p_d
OVERFLOW PAGES	n_o	t_o	a_o	u_o	p_o
ALL PAGES	n_a	t_a	a_a	u_a	p_a

AREA area-name–2
.
.
.

The numbers in the table reflect only those words which are in the user data portion of a page. If t_d, t_o, or $t_a \geqslant 1{,}000{,}000$ the print will truncate the last three digits and suffix the number with a K.

If the OUTPUT clause is not specified, only the SUMMARY STATISTICS section will be printed.

7.5.2.3. Examples

Example 1:

COMPACT DATABASE ON ERROR STOP.

This command will cause the compaction of every page in the database as described by the INVOKEd schema (see 7.2). Only summary statistics will be printed for each area. If any error is detected while performing this activity, subsequent DMU commands will be ignored. The user is cautioned about the amount of activity which may be required to perform compaction for a specification of DATABASE.

Example 2:

COMPACT AREA ABC,DEF OUTPUT STATUS.

This command will cause the compaction of every page in areas ABC and DEF. Additionally, a detailed report will be given on the record space utilization for these areas as described in the previous section. Since no ON ERROR clause is specified, the default PROCEED is assumed.

7.5.3. Page Expansion

The page expansion utility provides a means whereby the record storage capacity of an area may be increased without altering the number of pages allocated, overflow specification, Database Keys, Area Keys, or pointers. Consequently, such an expansion will not necessitate changes to any run units.

Prior to requesting this function, the schema must be changed to reflect the new page size for an area and the user must ensure that the Executive mass storage file which contains this area is sufficiently large to contain the enlarged pages.

The utility will lengthen each existing page of the specified area to the page size which it finds in the updated absolute schema (*as INVOKEd in MAIN, see 7.2*). No scratch files are required.

It must be noted that old audit trails and database dumps of expanded areas are not usable for recovery purposes after expansion. It is recommended that expansion be performed during a time when the user is certain of the integrity of his database and that a database dump be taken after the expansion is completed.

NOTES:

(1) It is not possible to use old database dumps and audit trails for recovering up to the point at which the expansion was performed, then re-expand the areas, and then apply post-expansion looks to the database to reach a desired recovery point.

(2) Also, since the page sizes in the "old" area are not the same as described by the updated absolute schema, rollback will not function properly. If QUICK-BEFORE-LOOKS are specified in the schema, they will be ignored during expansion. The user should dump all areas to be expanded prior to requesting expansion, and reload these areas if any error is encountered. The utility will print a message reminding the user of this "rollback override" if any error condition occurs.

7.5.3.1. Command Syntax

The command syntax for page expansion is:

EXPAND AREA *area-name* [EXISTING **PAGES** *integer*]

$$\left[\text{ON } \textbf{ERROR} \left\{\begin{matrix}\text{STOP} \\ \text{PROCEED}\end{matrix}\right\}\right] .$$

Separate commands are required for each area which is to have its pages expanded. The EXISTING clause is required only when the updated schema increased the number of pages in an area as well as the page size, i.e., the existing number of pages in the physical area is different from the logical number ALLOCATEd to that area in the updated schema. The INITIALIZE command is required to prepare the new pages to receive data. The EXISTING clause must be specified in the EXPAND command regardless of the order in which the INITIALIZE and EXPAND functions are performed.

The ON ERROR clause operates as described in Section 7.5.

7.5.3.2. Output

The output from page expansion consists only of a message informing the user of activity completion:

PAGE EXPANSION HAS BEEN SUCCESSFULLY COMPLETED FOR AREA area-code.

If errors are detected in command syntax or while performing the page expansion function, diagnostic messages will be produced. Appendix C describes all such diagnostics which could occur.

7.5.3.3. Examples

Example 1:

EXPAND AREA ABC ON ERROR STOP.

This command will cause the length of each page in area ABC to be expanded from the length it was prior to the execution of the request to the length specified in the updated absolute schema.

NOTE:

The user responsibilities given in Section 7.5.3 may cause this command to be embedded in a run stream which is more extensive than the one shown in Section 7.3.

Since no EXISTING clause is specified, it is assumed that the updated schema did not change the number of pages in area ABC. If any error is detected while performing the expansion, subsequent DMU commands will be ignored.

Example 2:

EXPAND AREA DEF.

This command will cause an activity similar to that of Example 1 except that since no ON ERROR clause is specified, the default PROCEED is assumed.

Example 3:

EXPAND AREA ABC EXISTING PAGES 200 ON ERROR STOP.

This command is similar to Example 1 except that it would be used if the updated schema changed the number of pages in ABC from 200 to some number greater than 200. This command could either precede or follow an

INITIALIZE AREA ABC BEGIN AT PAGE 201.

command (see 7.5.4).

7.5.4. Area Initialization

The area initialization utility provides for an "off-line" initialization of pages within an area, or portion of an area, prior to loading records into those pages. Page initialization consists of building a page header in main storage, clearing out the remainder of the page, and writing this page to the mass storage area.

Initialization of the pages within an area may occur in one of two ways:

(1) when an area is OPENed for INITIAL LOAD, pages are initialized and written to the area as required, or

(2) by requesting this utility function.

If this utility is used to initialize pages of an area, then the ALLOCATE clause in the schema should specify PRE-INITIALIZED to avoid the duplication of effort which would occur during an OPEN for INITIAL LOAD.

NOTE:

The existence or non-existence of the PRE-INITIALIZED clause has no influence on this utility function.

The utility allows for initialization to begin at any page within an area. The utility will always initialize all pages from the specified starting point to the end of the area as defined by the schema.

Note that the utility does not attempt to determine if the specified pages within the area contain germane data prior to over-writing those pages.

7.5.4.1. Command Syntax

The command syntax for initialization is:

INITIALIZE AREA *area-name* [BEGIN AT PAGE *page-number*]

[ON ERROR {STOP / PROCEED}] .

If the BEGIN clause is omitted, page number one is assumed which will cause all pages allocated to the specified area in the schema to be initialized. The ON ERROR clause operates as described in Section 7.5. A separate INITIALIZE command must be specified for each area to be initialized.

7.5.4.2. Output

The output from area initialization consists only of a message informing the user of activity completion:

INITIALIZATION SUCCESSFULLY COMPLETED

If errors are detected in command syntax or while performing the page expansion function, diagnostic messages will be produced. Appendix C describes all such diagnostics which could occur.

7.5.4.3. Examples

Example 1:

```
INITIALIZE AREA ABC ON ERROR STOP.
```

This command will cause the initialization of every page allocated to area ABC. If any error is detected while performing this activity, subsequent DMU commands will be ignored.

Example 2:

```
INITIALIZE AREA DEF BEGIN AT PAGE 100.
```

This command will cause the initialization of pages 100, 101, . . ., n; where n is the number of pages allocated to DEF in the schema. If n $<$ 100 a diagnostic message will be given and no action will be performed on the database. Since no ON ERROR clause is specified, the default PROCEED is assumed.

7.5.5. Database Patching

The patch utility provides the user with the ability to modify any word in any page of a database.

The user of the patch utility supplies specifications as to where the object word *(that word which is to be modified)* is located in the database and its current and desired values. The patch routine verifies that the actual value of the object word is the same as the user's specification to prevent an incorrect modification. The result of the requested patch operation is communicated back to the user. If the requested patch was not performed, diagnostic messages are given explaining the reason for the rejection of the request.

When one or more modifications to a page are made via the patch utility, a bit in the page header (PH0020, EQUF 8,,S1) is set to aid in future debugging.

The patch utility will cause database recovery looks to be taken as specified by the schema.

Each word which is to be modified requires a separate command.

7.5.5.1. Command Syntax

The command syntax for a patch request is as follows:

PATCH AREA *area-name* PAGE *page-number* WORD *displacement*

[{ RELATIVE / REL } **TO** { TOP / BOTTOM / RECORD *record-number* }]

{ REPLACING / REP } *integer–1* **WITH** *integer–2*

[ON **ERROR** { STOP / PROCEED }] .

The AREA and PAGE clauses are used by the utility routine to obtain the page containing the object word. The specific word within the page is specified as a displacement from the TOP or BOTTOM of the page, or from the beginning of a specified record of the page. If the displacement is specified relative to a record, the page number in the PAGE clause must be the logical page number of that record. The value supplied for displacement must be an integer which is greater than or equal to zero. A specification of zero for displacement means that the object word is that which is indicated by the RELATIVE clause, for example, a displacement of zero RELATIVE TO TOP establishes the first word of the page header as the object word of the default case is RELATIVE TO TOP.

The REPLACING clause specifies the existing octal value of the object word as a string of 12 octal characters preceded by the letter o. The WITH clause specifies the desired value of the object word in the same manner.

The ON ERROR clause operates as described in Section 7.5.

7.5.5.2. Output

The normal completion of the patch activity will cause the message

PATCH EXECUTED SUCCESSFULLY.

If errors are detected in command syntax or while performing the patch function, diagnostic messages will be produced.

Appendix C describes all such diagnostics which could occur.

7.5.5.3. Examples

Example 1:

PATCH AREA ABC PAGE 9 WORD 39 RELATIVE TO TOP REPLACING O007146243564 WITH O007146243565 ON ERROR STOP.

This command will cause word 39 from the beginning of the ninth page *(the first word of the page header is considered the beginning, i.e., word 0)* of area ABC to be changed from its current value of octal 007146243564 to octal 007146243565. If area ABC has less than 9 pages, or if a page has less than 40 words, or if the existing value is not octal 007146243564, a diagnostic will be given and no database alteration will take place. If these or any other errors are detected while performing this activity, the ON ERROR STOP clause will cause subsequent DMU commands to be ignored.

Example 2:

```
PATCH AREA ABC PAGE 9 WORD 2 RELATIVE TO RECORD 3 REPLACING O007146243564 WITH
O007146243565 ON ERROR STOP.
```

This command is similar to that of Example 1 except that the object word is specified as word 2 of record number 3 *(the record header is considered the beginning, i.e., word 0)*. Note that the object word may not be in the ninth page if the database key formed from the supplied area, page, and record information is for a foreign record.

7.5.6. Data Printing

The print utility provides a capability to print out the contents of data pages from a DMS 1100 data area. Through user specifications, the data printed may be varied from all data on all pages to selected data on a selected page. The format used for the print out is automatically handled by the print routine. No output format specifications are required from the user.

The data contained in the records on printed pages may be suppressed or printed by user request. If printed, any number of words ranging from only the first through all words of the record may be requested.

These may be displayed in either FIELDATA or ASCII format.

7.5.6.1. Command Syntax

The command syntax for a print request is:

```
PRINT AREA area-name [ {integer} WORDS ] [ DATA IS {ASCII    } ]
                     {ALL    }             {FIELDATA}

[BEGIN AT PAGE page-number–1 ] [END AT PAGE page-number–2]

[ ON ERROR {STOP   } ].
           {PROCEED}
```

The first two optional clauses are used to specify the number of words of data records to be printed out and the format to be used. A specification of ALL WORDS will cause entire data records to be printed for every record on the page regardless of type. A specification of n WORDS will cause the first n words of data records to be printed for every record on the page regardless of type. A default value of zero words will be used if the clause is omitted. If the DATA IS clause is not specified, ASCII format will be assumed.

The BEGIN and END clauses specify the first and last pages which will be printed. If the BEGIN clause is omitted, print will start with the first page of the area. If the END clause is omitted, print will continue to the last page of the area as specified by the ALLOCATE clause in the schema.

The ON ERROR clause operates as described in Section 7.5.

7.5.6.2. Output

The PRINT utility will print out the following general information for each page:

- AREA NAME – 12 character area-name.
- AREA CODE – decimal number code from DDL clause 'AREA CODE IS integer'.
- PAGE NUMBER – decimal page number of page to be printed.
- PAGE LENGTH – decimal page length in words of page to be printed.
- NUMBER SLOTS – total decimal number of slots allocated in the page.
- NUMBER UNUSED – total decimal number of unused words in page *(will include deleted record space).*
- LENGTH VACANT – decimal number of words of vacant space at the end of the page where new records will be placed before compaction must take place.

The PRINT utility will then print out the following information for each slot on the page:

- SLOT NUMBER – slot number of the record in decimal.
- RECORD NUMBER – record number of the record controlled by this slot in decimal.
- RECORD CODE – octal and decimal record code *(left and right)* of this record.
- RECORD LENGTH – total record length on page *(including slot and header)* in decimal.
- RECORD DISPL – relative record displacement on page in decimal from beginning of page.
- TYPE – type of slot (*i.e., DIRECT, CALC, SET, DELETED, DISPL/PG. NO., FOREIGN).* If the type is DISPL/PG. NO., then the RECORD CODE, RECORD LENGTH, and RECORD NAME are voided.
- RECORD NAME – first 12 characters of record-name from DDL clause 'RECORD NAME IS record-name'.

The PRINT utility will print out for each slot that controls a record on the current page the following owner, auto, and manual pointer information:

- RELATIVE POS – relative position from record header of pointer in decimal.
- TYPE – NEXT, PRIOR, or OWNER.

- SET NAME — first 12 characters of set-name of set that pointer refers to.
- SET CODE — decimal set code of set from DDL clause 'SET CODE IS integer'.
- AREA CODE — decimal area-code of pointer.
- PAGE NUMBER — decimal page number of pointer.
- SLOT NUMBER — decimal slot number of pointer.

The PRINT utility will print out the following information for each record on a page:

- The first n or ALL words of user data *(according to command specification)* with the following format:

  USER DATA – first n or ALL words of user data in ASCII or FIELDATA according to user specification.

- The PRIME PAGE NUMBER in decimal with following format:

  PRIME PAGE NO – page-number

- The CALC CHAIN POINTER in same format as pointers.

If errors are detected in command syntax or while performing the print function, diagnostic messages will be produced. Appendix C describes all such diagnostics which could occur.

7.5.6.3. Examples

Example 1:

```
PRINT AREA ABC ALL WORDS DATA IS FIELDATA.
```

This command will cause the print out of all pages of area ABC. The entire data record will be printed in FIELDATA format for all records on each page. Since no ON ERROR clause is specified, the default PROCEED is assumed.

Example 2:

```
PRINT AREA ABC.
```

Same as Example 1 except no data record information will be printed.

Example 3:

```
PRINT AREA ABC 10 WORDS BEGIN AT PAGE 4 END AT PAGE 4 ON ERROR STOP.
```

This command will cause the fourth page of area ABC to be printed. The first 10 words of each data record will be printed in ASCII format.

NOTE:

If any record on the page has 10 or less words the entire record is printed.

If any error is detected while performing this activity, subsequent DMU commands will be ignored.

7.5.7. Set Verification

This utility function provides a means of verifying set relationships within a data-base. The utility performs this function by traversing records of a set and examining their type and their pointers.

The set verification utility may be requested to vary its domain of examination from a single set occurrence, to all occurrences of a specified set, to all sets of a database through the specification of the appropriate format of the VERIFY command.

For each set being examined, the following error conditions will be detected:

- A pointer points to an undefined area or page number.
- A pointer points to a deleted or non-existent record.
- A pointer points to a record of a type not possible within the set.
- The manual flag control word has incorrect bits set or does not have the bit set which indicates membership in the set being checked.
- An owner record is encountered which is not the owner record at which the check started.
- A specific member record is encountered twice in a set occurrence.

Note that the set verification function never attempts to correct an error it detects. This responsibility remains with the data Administrator. The PATCH utility function may often be used to correct errors which are detected by VERIFY.

7.5.7.1. Command Syntax

The set verification command has three formats.

- The first format presented is to be used when all sets of all areas in a database are to be verified. The user is cautioned of the amount of processing which is required to accomplish such a function.
- The second format is used when all set occurrences of a given set are to be verified.
- The third format is used to direct the verification of a list of specific set occurrences.

Format 1:

```
VERIFY DATABASE [ {AREA  } area-name–1 [OWNER] [, area-name–2 [OWNER]] . . . ]
                {AREAS }

[MAXREC OF integer] [ ON ERROR {STOP    } ] .
                               {PROCEED }
```

A set verification request using this format will cause all set occurrences of all sets to be verified.

If the AREA clause is omitted, all areas will be searched for owners. If a list of areas is specified in the AREA clause, set verification will restrict its search for owners to only those areas designated as OWNER. Areas which contain members but no owners need not be specifed for this format, since an OPEN ALL will be performed.

The MAXREC clause is used to specify the maximum number of record occurrences in any set occurrence being verified. This value is used by set verification in loop detection. The value specified may be greater than the actual value, however an excessively large value will cause additional processing time in the event that a loop exists. If this clause is omitted, a default value specified as a system's generation parameter will be used.

The ON ERROR clause operates as described in Section 7.5. Note that the detection of a problem in set integrity is not considered an error in this sense, since it is the purpose of the function.

Format 2:

```
VERIFY SET set-name { AREA  } area-name–1 [OWNER] [ , area-name–2 [OWNER] ] . . .
                    { AREAS }

    [MAXREC OF integer] [ ON ERROR { STOP    } ] .
                                   { PROCEED }
```

A set verification request using this format will cause all set occurrences of a specified set to be verified.

A list of areas must be specified which contains all owner and member records of the specified set. By specifying OWNER after all area names containing set occurrence owners, the set verification routine will restrict its search for owners to those areas.

The MAXREC and ON ERROR clauses operate as described for *Format 1.*

Format 3:

```
VERIFY SET-OCCUR OF set-name { AREA  } area-name–1 [OWNER] [ , area-name–2 [OWNER] ] ...
                             { AREAS }

    WITH DBK AREA area-name-k1 PAGE page-number-k1 RECORD record-number-k1

        [ , AREA area-name-k2 PAGE page-number-k2 RECORD record-number-k2 ] . . .

        [MAXREC OF integer] [ ON ERROR { STOP    } ] .
                                       { PROCEED }
```

A set verification request using this format will cause a list of specified set occurrences of a given set to be verified.

The set containing the set occurrences to be verified is specified in the SET-OCCUR OF clause. If set occurrences of more than one set are to be verified, separate commands are necessary.

The AREA clause operates as described for *Format 2* except that a specification of OWNER is superfluous and will be ignored.

The DBK clause is used to specify a list of database keys of set occurrence owners which are to be verified. A key is specified by supplying the area name and page number on which the record logically resides, and its record number within that page. Any number of keys may be specified within the DBK clause.

The MAXREC and ON ERROR clauses operate as described for *Format 1.*

7.5.7.2. Output

If the requested set verification is performed without detection of set integrity errors, the message

SET VERIFICATION COMPLETE – NO ERRORS DETECTED

will be printed.

The following information is printed if any error is encountered by set verification:

- Set code
- Set name
- Set occurrence owner's DBK
- Record code of record in error
- Record name of record in error

In addition, one of the following messages is printed.

- NEXT POINTER <dbp> ON RECORD WITH DBK <dbk> POINTS TO UNDEFINED SPOT.
- RECORD WITH DBK <dbk> POINTS TO RECORD OF TYPE <type> WHICH IS NOT A VALID MEMBER OF THE SET BEING CHECKED.
- LOOP ENCOUNTERED. BAD NEXT POINTER IN RECORD WITH DBK <dbk>.
- OWNER POINTER <dbp> IN RECORD WITH DBK <dbk> IS INCORRECT. POINTER SHOULD BE <dbp>.
- PRIOR POINTER <dbp> IN RECORD WITH DBK <dbk> IS INCORRECT. POINTER SHOULD BE <dbp>.

- DIFFERENT OCCURRENCES OF OWNER RECORD TYPE ENCOUNTERED THAN THE ONE STARTED ON. BAD NEXT POINTER <dbp> IN RECORD WITH DBK <dbk>. NEXT POINTER SHOULD BE <dbp>.

- PROPER BIT IN MANUAL FLAG CONTROL WORD NOT SET.

- BIT SET IN MANUAL FLAG CONTROL WORD BEYOND THE POINT WHICH IS VALID. THIS WILL NOT CAUSE IMMEDIATE PROBLEMS BUT MAY INDICATE OTHER ERRORS.

- PRIOR POINTER ERROR INDICATES POSSIBLE ERROR IN MANUAL FLAG CONTROL WORD.

If errors are detected in command syntax or while performing set verification *(other than set integrity errors as cited previously)*, diagnostic messages will be produced as described in Appendix C.

7.5.7.3. Examples

Example 1 (Format 1):

VERIFY DATABASE AREAS ABC OWNER, DEF OWNER MAXREC 1000 ON ERROR STOP.

This command will cause the verification of all set occurrences of all sets in the database whose owners occur in areas ABC and DEF. If more than 1000 records are examined in attempting to traverse any set occurrence, a loop analysis routine will be called to determine the source of the loop and generate the appropriate output. If any error other than one involving set integrity is detected while performing this activity *(such as I/O errors)*, subsequent DMU commands will be ignored.

Example 2 (Format 2):

VERIFY SET XYZ AREAS ABC, DEF OWNER.

This command will cause the verification of all set occurrences of the set XYZ whose owners occur in area DEF. Members may occur in both areas ABC and DEF. Loop detection is based upon the default value specified at system's generation. Since no ON ERROR clause is specified, the default PROCEED is assumed.

Example 3 (Format 3):

VERIFY SET-OCCUR OF XYZ AREAS ABC, DEF WITH DBK AREA DEF PAGE 5 RECORD 6, AREA DEF PAGE 7 RECORD 2.

This command will cause the verification of two set occurrences of the set XYZ. The owners of the two occurrences are both found in area DEF; on page 5 record 6 and on page 7 record 2.

NOTE:

The page numbers and record numbers refer to their logical positions and may or may not be the same as their physical placement.

Loop detection is based upon the default value specified at system generation. Since no ON ERROR clause is specified, the default PROCEED is assumed.

A. INTERNAL ERROR CODES

A.1. GENERAL

In addition to the external error codes returned to the user as explained in the *UNIVAC 1100 Series Data Management System (DMS 1100) American National Standard COBOL (Fieldata) Data Manipulation Reference, UP-7908* (current version) section on ERROR RECOVERY TECHNIQUES, the DMR has provisions for detecting execution time problems. These internal errors may result from inconsistencies in the user program which escaped prior detection or may be an indication of an error in DMR coding itself.

This appendix lists the possible error codes and their explanation. When such an error occurs, the DMR will initiate a diagnostic dump and will then abort. All run units in the system at the time of the internal error will receive a rollback error code 03 (*internal DMR error*) and will be rolled back.

A.2. CONTINGENCY ERRORS

If a contingency occurs between an IMPART and DEPART the contingency routine within the LINKER will receive control and will print out the following information about the contingency in the Users Print File *(using the Executive Proc. L$SNAP).* See the following Example 2 on contingency errors.

(1) The first line contains the word 'CONTIN' plus time and date information.

(2) The X, A, and R registers are printed out on the following lines.

(3) Following the registers is a line containing (*at the left side of page 1 or 2*) an absolute address (*address in the LINKER*) and going across the page to the right, 2 sequential words containing the contingency information. The format of the contingency information is shown in Example 1.

Example 1:

1	2	3	4
ERROR TYPE	ERROR CODE	CONT. TYPE	ERROR ADDRESS
T/S FOR ESI	NOT USED	NOT USED	PACKET ADDRESS
5	6	7	8

1 — ERROR TYPE — further qualifies contingency type, e.g., 'I/O error' within contingency type for 'ERR MODE'.

2 — ERROR CODE — further qualifies error type, e.g., 'UNASIGNED AREA' within error type for 'I/O ERROR'.

3 — CONTINGENCY TYPE — type of contingency received, e.g., ERR-MODE=12, GUARD MODE=02, etc.

4 — ERROR ADDRESS — absolute address where contingency occurred.

5 — T/S FOR ESI — test and set location for ESI activities.

6 — NOT USED

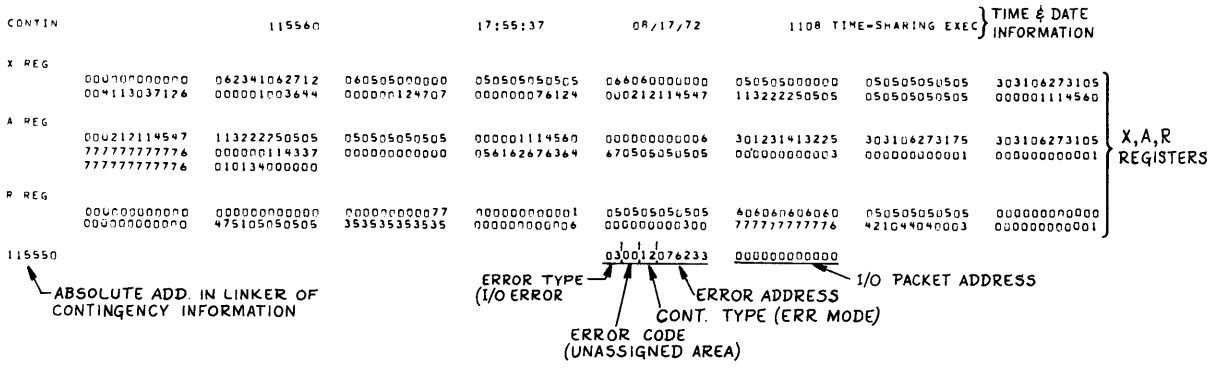
7 — NOT USED

8 — PACKET ADDRESS — absolute packet address for contingencies occurring while doing an EXECUTIVE REQUEST (ER) using a packet, e.g., ER IOW$.

For a further explanation of codes used see technical description for LINKER or *UNIVAC 1100 Series Operating System Programmer Reference, UP-4144* (current version) section on CONTINGENCIES.

Example 2:

CONTINGENCY PRINT-OUT



A.3. EXPLANATION OF THE GS07 DUMP

The GS07 routine is used by the DMR when an internal DMR error occurs. The EXEC proc L$SNAP is used five times to:

(1) Print out the contents of the registers and the absolute address where the abort was initiated.

  (a) The first line contains the word 'ABORT$', plus time and date information.

  (b) The X, A and R registers are printed out on the following lines.

  (c) Following the registers is a line which contains two addresses. The second word contains the absolute address in the DMR where the abort was initiated.

(2) Print out the error status.

  (a) The first line contains the word 'ERRST', plus time and date information.

  (b) The next line contains two words. The second word on the line contains the error status in the left 1/3 of the word in octal, and the rollback error status in the right 1/3 of the word in FIELDATA code.

(3) Print out the contents of the DMR system buffer area.

  (a) The first line contains the word 'BUFFER', plus time and date information.

  (b) The following lines are an octal dump of the whole DMR buffer area (i.e., fixed and variable table area, page area).

(4) Print out the contents of all DMR instructions.

  (a) The first line contains the word 'DMCODE', plus time and date information.

  (b) The following lines are an octal dump of all the DM instructions.

(5) Print out the contents of the run units D$WORK.

  (a) The first line contains the word 'D$WORK', plus time and date information.

  (b) The following lines are an octal dump of all D$WORK for the run unit.

A.4. INTERNAL ERROR CODES

Error Code	Error Description and Correction
0001	ILLEGAL COMMAND CODE.
	Run unit is using a command code which is not implemented.
0002	RECORD ENTRY ON PAGE WITH NO SLOT.
	No slot on page could be found which points to this record.

Error Code	Error Description and Correction
0005	OCCURS LOST FOR VARIABLE LENGTH RECORD. A record is flagged as a variable length record, but a search of the items fails to turn up the item having the OCCURS DEPENDING clause.
0011	USAGE, FORMAT, OR LENGTH PARAMETER MISSING. The DMR internal routine which compares two similar items used as search keys could not find a description of either the usage, format, or length of one of the items.
0012	RECORD CODE DOES NOT MATCH MEMBER CODE IN SET DESCRIPTION TABLE. A record with an alias clause is implied to be a member of a particular set. However, the record code for the record does not match the record code for any records defined to be members of the set in the Set Description Table.
0013	THE DATABASE-KEY IS DEFINED AS SORT KEY BUT NO OBJECT RECORD GIVEN. In the Item Extraction Table the database-key for a record already stored is being used as a sort key. However, a required pointer to the record in the database is missing.
0014	NO. OF ALIAS DATA-NAMES NOT SAME AS NO. OF GIVEN SORT KEYS FOR OBJECT RECORD TYPE.
0015	THE NUMBER OF KEYS SPECIFIED TO LOCATE A RECORD DIFFERS FROM THE NUMBER OF KEYS DEFINED FOR THE RECORD TYPE. In the Item Extraction Table the keys specified to locate a record occurrence do not agree in number with the number of keys defined for the record type being searched.
0017	UNDEFINED POINTER LINKAGE. The record description table indicates no recognizable pointer linkage (*i.e., not next, prior, owner, or a combination of them).*
0070	CONTRADICTION IN INDEX SLOT SWITCH SETTING. A page index entry simultaneously indicated foreign record placement and address field containing a displacement rather than new page address (*Bits 33 and 34 both on*).
0071	RECORD FOUND USING DBK IS VACANCY ENTRY. On a FIND operation, the record retrieved using the database-key turned out to be a vacancy entry.
0072	SEARCH FOR FOREIGN RECORD SLOT FAILED. The DMR is searching an overflow page to find the foreign record no. of a record which was modified and expanded such that it had to be moved to the overflow page. The foreign record could not be found.

Error Code	Error Description and Correction
0074	SYSTEM GENERATION PARAMETER 'HIARCH' EXCEEDED IN BUILDING A PATH. The Data Administrator has the option of specifying how many levels of sets are allowed in a database hierarchy. This is controlled by the System Generation Parameter 'HIARCH'. Unless changed, it is set to 10. (*See Section 3.*)
0076	DUPLICATES ARE PRESENT AT AN INTERMEDIATE LEVEL IN THE HIERARCHY. The DMR does not allow duplicates in any set of a hierarchy except the lowest. While tracing a path through the hierarchy the DMR found a violation of this rule.
0100	SET OCCURRENCE SELECTION IS LOCATION MODE OF OWNER BUT THE SET ORDER IS NEXT OR PRIOR. In developing a path through the hierarchy, the DMR found an intermediate level set whose set occurrence selection is through location mode of owner and whose owner record is a member of a set whose order is NEXT or PRIOR. Consequently owner record could not be uniquely determined.
0103	NO SPACE FOUND TO STORE RECORD WITH LOCATION MODE VIA SET. The DMR considered all areas or partial areas listed on WITHIN clauses and could not find enough space to store this record VIA SET.
0104	RECORD HAS INTERVAL CLAUSE BUT NO WITHIN CLAUSE. The DMR has found a record which is defined in the schema having a location mode via set with an INTERVAL clause but the required WITHIN clause is missing.
0106	NO ENOUGH SPACE IN LAST RECORD INTERVAL TABLE. The System Generation Parameter, LRINUM, limits the number or record types which may contain an interval clause. An attempt was made to exceed this limit. Unless changed, it is set to 10 (*see Section 3*).
0107	AREA NAME CONFLICT BETWEEN LAST RECORD INTERVAL TABLE ENTRY AND RDT. The run unit tried to store a record whose location mode is via set with an INTERVAL clause but failed. An area entry in the Last Record Interval table contradicted the areas listed on the WITHIN clause. Internal error.
0110	SET ORDER NOT SORTED – RECORD RECEIVED DOES NOT MATCH RECORD CODE IN PATH. While tracing a path through an unordered set, the DMR retrieved the next record in logical order and it does not match the record expected.
0111	EMPTY SET ENCOUNTERED WHEN ASKING FOR NEXT OR PRIOR MEMBER. While tracing a path through the hierarchy an empty set was encountered. A request for NEXT or PRIOR returned the OWNER instead.

Error Code	Error Description and Correction
0112	INTERVAL CLAUSE ON DDL 'LOCATION MODE VIA SET' RECORD CLAUSE DOES NOT AGREE WITH DDL 'ALLOCATE. . . EVERY' CLAUSE. While determining proper page to store a record via set, a page was selected which turned out to be an overflow page.
0113	BAD I/O STATUS RETURNED. After an I/O operation from within the DMR, a bad I/O status was returned.
0114	AFTER DELETE, DMR COULD NOT RETURN TO TOP OF HIERARCHY. After deleting a record following a path through the hierarchy, the DMR could not retrace its path to the top of the hierarchy.
0115	RECORD FOUND VIA SET LOGIC NOT DEFINED AS MEMBER OF PROPER SETS. A record which was found via set was either implied to be a member of a manual set and not defined as such or does not contain links of the type expected by the DMR.
0116	DEALLOCATED 011 MISCELLANEOUS SPACE ADDRESS NOT FOUND IN D$WORK.
0117	RECORD SLOT CANNOT BE FOUND. An attempt was made to find the slot associated with a record header; the search failed.
0120	NO SPACE REMAINING IN D$WORK FOR ANOTHER MISCELLANEOUS SPACE ADDRESS. This indicates a bad MISLEN parameter – not enough words allocated.
0121	SAVED REC. LRI ENTRY NOT FOUND IN LRI TABLE. The LRI entry for this record was saved when the run unit began storing records of this type, but when Rollback of the run unit was needed the record type could not be found in the LRI Table.
0122	NOT ENOUGH PAGE BUFFER SPACE YET AVAILABLE FOR ROLLBACK. This is an internal error status passed back to Rollback routine from core allocation.
0123	ROLLBACK CANNOT FIND THE QUICK LOOK ENTRY IN THE QLSEG TABLE FOR ROLLING BACK THE RUN UNIT.
0124	ROLLBACK BUFFER SIZE IS TOO SMALL. Systems generation parameter PAGLEN is too small to allow Rollback.
0125	RUN UNIT THAT ALTERED PAGE AND IS STILL ACTIVE COULD NOT BE FOUND.

B. STANDARD RUN STREAMS

B.1. SYSTEM INITIALIZATION RUN STREAMS

- The following run stream may be used to initialize the system file as discussed in Section 4.2:

```
@RUN        run-ID, account-no., DMS
@ASG,A      DMRMT.
@ASG,CP     DMS-SYS-FILE.
@FREE       DMS-SYS-FILE.
@ASG,A      DMS-SYS-FILE.
@ASG,CP     AUDIT-TRAIL1.,T/2,XXXX/XXXX
@XQT        DMRMT.SFINIT
@FIN
```

NOTES:

(1) If the Audit Trail tape is not catalogued by this run stream, it will be dynamically catalogued by SFINIT and the reel numbers solicited from the operator.

(2) If the names of the System file or Audit Trail are changed at system generation, the above run stream must be altered accordingly.

- The following run stream may be used to initialize the Quick-Look file as discussed in Section 4.3:

```
@RUN        run-ID, account-no., DMS
@ASG,A      DMRMT.
@ASG,CP     QUICK-LOOK1.,type/reserve/granule/maximum
@ASG,CP     QUICK-LOOK2.,type/reserve/granule/maximum
@FREE       QUICK-LOOK1.
@FREE       QUICK-LOOK2.
@XQT        DMRMT.QLINIT
@FIN
```

NOTES:

(1) If the names of the Quick-Look files are changed at system generation, the above run stream must be altered accordingly.

(2) The size specification on the @ASG's of the Quick-Look files should be in agreement with the specification for the Quick-Look Segment table at system generation.

Example:

The default values supplied in DMSSGP would require

```
@ASG,CP QUICK-LOOK1., type///100
@ASG,CP QUICK-LOOK2., type///200
```

to be specified.

B.2. INITIATING QUICK RECOVERY

The following run stream may be used to initiate Quick Recovery as discussed in Section 6.3:

```
@RUN        QUIREC, account-no., DMS
@ASG,A      DMRMT.
@ASG,A      QUICK-LOOK1.
@ASG,A      QUICK-LOOK2.
@ASG,A      for all database areas
@XQT        DMRMT.QUIREC
@FIN
```

NOTES:

*(1) When this run stream is developed it should be placed into DMS*DMRMT.*

(2) If the names of the Quick-Look files are changed at system's generation, the above run stream must be altered accordingly.

B.3. TAKING DATABASE DUMPS FOR RECOVERY

The following run stream may be used for taking a database dump for recovery as discussed in Section 6.4.1:

```
@RUN        DATDUM,account-no., DMS
@ASG,A      DMRMT.
@XQT        DMRMT.DTBSDMP1
@ASG,NT     OBACKUP,T
@SECURE
    OBACKUP = reel number list
    SAVE FILES schema file name, system file name, areas
    END
@XQT        DMRMT.DTBSDMP2
@FIN
```

NOTES:

(1) For a detailed discussion of the SECURE processor, see UNIVAC 1100 Series Operating System Programmer Reference UP-4144, (current version).

(2) 'areas' in the SAVE FILES statement is the list of areas which are to be dumped.

B.4. INITIATING LONG RECOVERY

The following run stream may be used to initiate Long Recovery as discussed in Section 6.4.5:

```
@RUN          LONREC, account-no., DMS
@DELETE,C     system file name
@DELETE,C     schema file name
@DELETE,C     area name 1 } delete catalogue directories of all areas to be
@DELETE,C     area name 2 } recovered by the LONGRECOVERY routine
              :
              :
              :
@ASG,NT       IBACKUP,T
@SECURE
    IBACKUP = reel number list
    LOAD FILE system-file name FROM IBACKUP
    END
@ASG,A        system file name
@ASG,A        DMRMT.
@XQT          DMRMT.DMRACT
@REWIND       IBACKUP.
@SECURE
    IBACKUP = reel number list
    LOAD FILE all file names except system file FROM IBACKUP
    END
@ASG,A        schema file name
@ASG,A        area name 1 } assignment of all areas to be recovered by the
@ASG,A        area name 2 } LONGRECOVERY routine
              :
              :
              :
@XQT          DMRMT.LONGRECOVERY
MMDDYYHHMMSS or SEQmmmmmmAd
@DELETE,C     Quick look files
@ASG,CP       Quick look files
@FREE         Quick look files
@XQT          DMRMT.QLINIT
@FIN
```

NOTES:

(1) For a detailed discussion of the SECURE processor, see UNIVAC 1100 Series Operating System Programmer Reference UP-4144, (current version).

(2) If the Recovery Point or Checkpoint specification is not given after the @XQT DMRMT.LONG-RECOVERY statement, this information will be solicited from the operators console.

B.5. ESTABLISHING CHECKPOINTS

The following run stream may be used to initiate CHECKPOINT as discussed in Section 6.4.4:

```
@RUN      CHKREQ, account-no., DMS,...
@ASG,A    DMRMT.
@XQT      DMRMT.CHECKPOINT
@FIN
```

NOTES:

(1) The run-ID for this run must be CHKREQ. A validity check is made for this run-ID.

(2) This run stream assumes that the absolute element CHECKPOINT has been placed into the catalogued file DMRMT.

B.6. INTERFACING THE UTILITY PROCESSOR WITH A SCHEMA

The following is a sample run stream for interfacing the Data Management Utility processor with a schema as discussed in Section 7.2. This run stream should be rerun after a DMS system's generation or when the schema is changed. The control stream is numbered to correspond to the explanation following:

```
 1   @RUN          run-id, account-no., project
 2   @FREE         TPF$
 3   @ASG,T        TPF$, F2//POS/10
 4   Control statements for placing required elements into TPF$
 5   @ED,U         MAIN
 6   C /SYSTEM-TEST/schema-name/G
 7   @EOF
 8   @DMLP,BZ      MAIN,COB
 9   @COB,B        COB,MAIN
10   @PREP         TPF$.
11   @MAP          UTMAPMT,DMU
12   @ASG,A        DMS*DMRMT.
13   @COPY,A       DMU,DMS*DMRMT.DMU
14   @FIN
```

Statement Numbers	Explanation
1	Standard Executive run statement.
2–3	Expansion of TPF$ to accommodate all necessary elements.
4	Control statements to: (a) place the absolute element DMLP into TPF$. (b) place into TPF$ either the relocatable DMR for running collected with DMU or the necessary elements to create a DMU which runs with the DMR as a REP. (c) place the utility processor elements of the DMS 1100 release tape into TPF$. *NOTE:* *For additional information, consult the appropriate DMS 1100 Field Support Bulletin.*
5–7	Statements to change the schema file name in the INVOKE command of the symbolic COBOL element MAIN from the dummy name of SYSTEM-TEST to the name used at a particular site. Note that the INVOKE in MAIN specifies COMMAND QUICK-BEFORE-LOOKS.
8–11	Statements to process the updated element MAIN and create an absolute DMU which will run with the DMR as a REP. If it is desired to create a DMU which runs with a collected DMR, the map element UTMAPMT must be changed to UTMAPST.
12–13	Statements to place the utility processor into the catalogued file DMS*DMRMT. These statements may be altered to place DMU into any other user file.
14	Standard Executive FIN statement.

NOTE:

The preceeding assumes that the American National Standard COBOL (Fieldata) compiler and library is being used. If it is desired to use the American National Standard COBOL (ASCII) compiler and library, then

@ED, U MAIN/ASCII

and and

@DMLP,BZ MAIN/ASCII,COB

should be substituted into the sample run stream.

APPENDIX C. DATA MANAGEMENT UTILITY PROCESSOR (DMU) DIAGNOSTICS

C.1. GENERAL

There are two types of errors which may occur within the DMU processor.

- The first type is that of semantic errors.

  These errors are caused by improper command specifications. The error messages will always be printed immediately following the print of the erroneous command. Section C.2 presents a listing of all such error messages. When such errors occur, an attempt will be made by DMU to semantically check all subsequent commands, however, the functions requested by these commands will not be performed.

- The second type of error is that which occurs during the actual performance of the function requested by a semantically correct command.

  These errors will always be printed following the print of the command requesting the function. A listing of all such errors is presented by command type starting in Section C.3. When errors of this type occur, subsequent commands will be ignored if an ON ERROR STOP clause is specified on the command causing the error. If ON ERROR clause is not specified, or if an ON ERROR PROCEED is specified, DMU will attempt to process the next command.

Since the DMU processor interface with the DMR as a run unit it is also possible to obtain an error status from the DMR. If this occurs, control will be returned to the rollback or error paragraphs of the interface COBOL routine MAIN. A message specifying which paragraph was reached and the internal error status will be printed in this case. If this situation occurs, no further DMU commands will be processed.

C.2. COMMAND SEMANTIC ERRORS

AREA LIST CAN NOT BE CREATED

FAILURE IN OBTAINING SCHEMA REFERENCE TABLE

FAILURE TO GET SCHEMA DESCRIPTION TABLE

INVALID AREA/SET/RECORD CODE

INVALID PAGE NUMBER

INVALID RECORD NUMBER

NUMBER OF WORDS TOO LARGE

SCHEMA DESCRIPTION TABLE COULD NOT BE READ

WORD DISPLACEMENT WITHIN PAGE TOO LARGE

C.3. COMPACT ERRORS

AREA CAN NOT BE OPENED

AREA REFERENCE TABLE CAN NOT BE FOUND

ERROR WHILE PACKING THE PAGE

I/O ERROR DURING PAGE COMPACTION

C.4. EXPAND ERRORS

BUFFER FOR READ CAN NOT BE ALLOCATED FOR AREA <area-name >

I/O ERROR ON AREA <area-name >

NEW PAGE SIZE < OLD PAGE SIZE

OPEN HAS FAILED FOR AREA <area-name >

REQUIRED SPACE > ASSIGNED FOR AREA <area-name >

ROLLBACK WAS NOT PERFORMED

THE AREA REFERENCE TABLE CAN NOT BE FOUND FOR AREA <area-name >

WRONG EQUIPMENT ASSIGNED FOR AREA <area-name >

C.5. INITIALIZE ERRORS

Initialize detects no errors other than those which occur within the DMR.

C.6. PATCH ERRORS

AREA <area-name > NOT ASSIGNED

INVALID AREA CODE <area-code >

INVALID DISPLACEMENT WITHIN PAGE BOUNDARIES

INVALID DISPLACEMENT WITHIN RECORD BOUNDARIES

PATCH COULD NOT EXECUTE SUCCESSFULLY

USER SUPPLIED DATA DID NOT MATCH WITH EXISTING DATA

C.7. PRINT ERRORS

ERROR ENCOUNTERED OPENING AREA

PRINT UTILITY HAS BEEN ENTERED WITH BAD COMMAND TYPE

C.8. VERIFY ERRORS (SET VERIFICATION)

In addition to the error messages involving set validity (see 7.5.7.2) the following error messages may occur:

AREA REFERENCE TABLE CAN NOT BE FOUND FOR AREA <area-name>

I/O ERROR ON AREA <area-name>

OPEN HAS FAILED FOR AREA <area-name>

APPENDIX D. DATA MANAGEMENT UTILITY PROCESSOR (DMU) RESERVED WORDS

A list of Data Management Utility Processor (DMU) Reserved Words is as follows:

ALL
AREA
AREAS
ASCII
AT
BEGIN
BOTTOM
CALC
CHAIN
COMPACT
DATA
DATABASE
DBK
DUMP
END
ERROR
EXISTING
EXPAND
FIELDATA
FROM
INITIALIZE
IS
MAXREC
ON
OUTPUT
OWNER
PAGE
PAGES
PATCH
PRINT
PROCEED
RECORD
RECOVER
RECOVERY
REL (RELATIVE)
RELATIVE
REP (REPLACING)
REPLACING
SELECTIVE
SET
SET-OCCUR
START
STATUS
STOP
TAPE
TO
USE
VERIFY
WITH
WORD
WORDS

APPENDIX E. LIST OF ABBREVIATIONS

AUTORP	–	Automatic Recovery Point
CD	–	Communication Descriptions
DDL	–	Data Definition Language
DMCA	–	Data Management Communications Area
DML	–	Data Manipulation Language
DMR	–	Data Management Routine
DMS	–	Data Management System
DMSCALC	–	Data Management System-Supplied Calc
DMSSGP	–	Data Management Systems Support Generation Parameter Element
DMU	–	Data Management Utility Processor
MCS	–	Message Control System

Comments concerning this manual may be made in the space provided below. Please fill in the requested information.

System: _____

Manual Title: _____

UP No: _____ Revision No: _____ Update: _____

Name of User: _____

Address of User: _____

Comments:

CUT

FOLD

FIRST CLASS
PERMIT NO. 21
BLUE BELL, PA.

BUSINESS REPLY MAIL NO POSTAGE STAMP NECESSARY IF MAILED IN THE UNITED STATES

POSTAGE WILL BE PAID BY

UNIVAC

P.O. BOX 500

BLUE BELL, PA. 19422

ATTN: SYSTEMS PUBLICATIONS DEPT.

FOLD

CUT