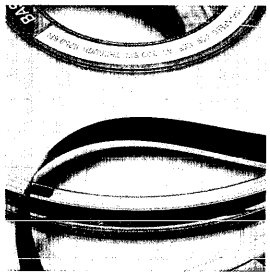
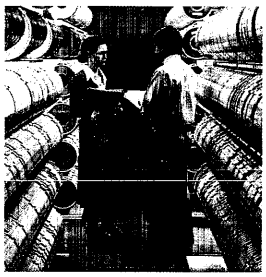
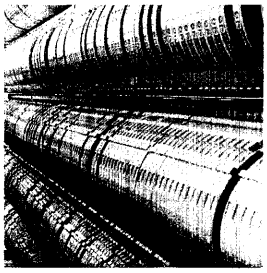
Prime Computer, Inc.

MRU4304-009
Revision 19.1 Software Release Document

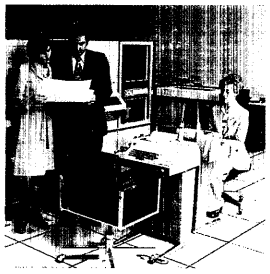
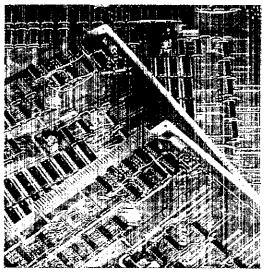












Software Release Document

MRU4304-009

Revision 19.1

by

Emily M. Stone

This document contains information on enhancements and technical changes to Prime user software at Rev. 19.1.

Prime Computer, Inc.
500 Old Connecticut Path
Framingham, Massachusetts 01701

COPYRIGHT INFORMATION

HOW TO ORDER TECHNICAL DOCUMENTS

U.S. Customers

Software Distribution
Prime Computer, Inc.
1 New York Ave.
Framingham, MA 01701
(617) 879-2960 X2053, 2054

Prime Employees

Communications Services
MS 15-13, Prime Park
Natick, MA 01760
(617) 655-8000 X4837

Customers Outside U.S.

Contact your local Prime subsidiary or distributor.

INFORMATION Systems

Contact your Prime INFORMATION system dealer.

PRINTING HISTORY — Software Release Document

Edition	Date	Number	Documents Rev.
	December 1982	MRU4304-009	19.1

SUGGESTION BOX

All correspondence on suggested changes to this document should be directed to:

Emily M. Stone
Technical Publications Department
Prime Computer, Inc.
500 Old Connecticut Path
Framingham, Massachusetts 01701

Contents

About This Book

This book summarizes the changes and new features in Prime's user software at Rev. 19.1 of PRIMOS. One chapter is devoted to each of the following:

- PRIMOS and Utilities (Chapter 2)
- Languages (Chapter 3)
- Data Management Systems (Chapter 4)
- FORMS and FED (Chapter 5)
- Communications (Chapter 6)
- Software Support for New Hardware (Chapter 7)

Within each chapter, the information on each product (e.g., COBOL, MIDAS, RJE) begins on a new right-hand page. Pages can thus be extracted from this book and placed in other manuals as necessary.

Note

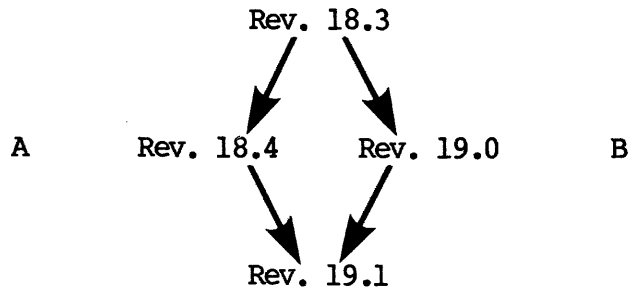
This book is designed to supplement other manuals. Its pages are not replacement pages, and its pagination does not correspond to the pagination of other books.

For each individual product, this book provides the following information (where applicable):

- New features in the software at Rev. 19.1
- Documentation corrections and additions
- Software problems fixed at Rev. 19.1
- Software problems outstanding as of Rev. 19.1

Upgrade Paths to Rev. 19.1

The most common upgrade paths to Rev. 19.1 are from Rev. 18.4 and from Rev. 19.0, as shown below.



Rev. 19.1 introduces a number of new features and corrections to software problems. In addition, Rev. 19.1 incorporates all of the features, corrections, and changes that were introduced at Revs. 18.4 and 19.0.

New Features

This book describes features that are new at Rev. 19.1. If you took upgrade path A (Rev. 18.3 to Rev. 18.4 to Rev. 19.1), you will need to find out about features that were new at at Rev. 19.0. For a summary of Rev. 19.0, refer to the Rev. 19.0 Software Release Document (MRU4304-008).

If you took upgrade path B (Rev. 18.3 to Rev. 19.0 to Rev. 19.1), you will need to find out about features that were new at Rev. 18.4. For a summary of Rev. 18.4, refer to the Rev. 18.4 Software Release Document (MRU4303-007).

Documentation Corrections

This book contains a number of corrections and additions for other Prime software manuals. It is assumed that you have access to our most recent documentation. Lists of new books and updates can be found in the following places:

- The section called NEW TITLES, later in this chapter, for new titles at Rev. 19.1.
- The Rev. 19.0 Software Release Document (MRU4304-008), for new titles at Rev. 19.0.
- The Rev. 18.4 Software Release Document (MRU4304-007), for new titles at Rev. 18.4.

This book does not repeat documentation corrections that were printed in previous Software Release Documents.

Software Problems Fixed

The corrected software problems listed in this book are divided into three categories:

1. Problems fixed at Revs. 18.4 and 19.1
2. Problems fixed at Revs. 19.0 and 19.1
3. Problems first fixed at Rev. 19.1

If you have already used Rev. 18.4, you will be most interested in categories 2 and 3. If you have already used Rev. 19.0, you will be most interested in categories 1 and 3.

Problems that were fixed at both Rev. 18.4 and Rev. 19.0 are not listed in this book. This book also does not list problems that were fixed at Rev. 18.3 or before.

Software Problems Outstanding

This book lists some of the software problems outstanding as of Rev. 19.1. For further information, contact your Prime field engineer.

CHAPTER 1

INTRODUCTION

IN THIS CHAPTER

This chapter provides:

- An overview of Rev. 19.1
- Information on installation of Rev. 19.1
- A list of new titles from Prime's Technical Publications Department

OVERVIEW OF REV. 19.1

New Features and Products

Rev. 19.1 introduces the following new software products:

- MIDASPLUS — a data management product that is compatible with, and improves upon, MIDAS
- Software support for the cartridge tape drive of the Prime 2250
- Software support for the ICS1 Communications Controller
- Software support for the PST 100 terminal

In addition, at Rev. 19.1, new features or enhancements have been added to the following products:

COBOL
COPY_DISK
DBMS
DBMS/QUERY
Editor
EMACS
FED
FORMS
FORTRAN
FORTRAN 77
LOGPRT
MIDAS
PASCAL
PHYSAV/PHYRST
PL/I, Subset G
POWERPLUS
PRIMOS
SEG

Documentation Changes

This book contains documentation corrections or additions for manuals on the following products:

COBOL
COPY_DISK
DBMS
DBMS/QUERY
Editor
EMACS
MAGNET
MIDAS
MIDASPLUS
PL/I, Subset G
POWERPLUS
RJE
RUNOFF
SEG
Subroutines

Software Problems Fixed

Problems in the following products have been corrected:

AVAIL
BASIC/VM
Batch
CMPF
COBOL
COPY
CPL
DBG
DBMS
DBMS/QUERY
DPTX
Editor
EMACS
FIX_DISK
FORMS
FORTRAN
FORTRAN 77
FTS
LOAD
LOGPRT
MAGNET
MAKE
MIDAS
Pascal
PHYSAV/PHYRST
PL/I, Subset G
POWERPLUS
PRIMENET
PRIMOS
PRIMOS II
RJE
RPG
RUNOFF
SEG
SPOOL
Subroutines
VFTNLIB
VRPG

Software Problems Outstanding

This book describes software problems outstanding in the following products:

DBG
DBMS
DBMS/QUERY
EMACS
FORMS
FORTRAN
FORTRAN 77
MIDAS
RJE
SEG
VRPG

INSTALLING REV. 19.1

The Rev. 19.1 Master Disk is in complete Master Disk format, rather than in the update format that is normally used at minor (dot) revisions. The Rev. 19.1 Master Disk contains all Master Disk software, including the software that has not changed since Rev. 19.0. Rev. 19.1 can therefore be installed without prior installation of Rev. 19.0.

For information on installing a Rev. 19 system, refer to the Rev. 19.0 Planning and Installation Guide (DOC6426-190).

Note

The installation and conversion procedure lists in Chapter 2 of the System Administrator's Guide (DOC5037-190) are superseded by the information in the Rev. 19.0 Planning and Installation Guide.

NEW BOOK TITLES

The following new technical publications are available at Rev. 19.1.

The Prime 2250

2250 Site Preparation Guide (DOC6514-191): This manual describes site preparation procedures for Prime 2250 systems. Topics include site selection, HVAC requirements, electrical and cabling requirements, and equipment receiving and inspection. Thus, the book covers the period from the time a customer first places an order for a system to installation day.

Prime 2250 Planning and Installation Guide (DOC6515-191): This guide is intended for the user who is responsible for installing Rev. 19.1 software on a Prime 2250. The book provides guidelines for planning a Rev. 19.1 system. It also discusses administrative issues such as User Profiles, ACLs, and system configuration. The book includes detailed instructions for the initial installation of Rev. 19.1 software on a Prime 2250. Examples of planning and installation sessions are provided.

Using Your Prime 2250 (DOC6516-191): This book provides guidelines for the successful daily operation of the Prime 2250 computer. It is assumed that the reader has no prior knowledge of computers or computer operations. After a general introduction to the Prime 2250, the book explains how to:

- Start up the system
- Use PRIMOS commands
- Work with the magnetic tape drive
- Back up work done on the system
- Communicate with other users and other computers
- Shut down the system
- Maintain the system and perform trouble-shooting

The PST 100 Terminal

PST 100 Primer (DOC6517-001): This book is an introductory manual for all users of the Prime System Terminal (PST 100). The book contains basic information on the components and features of the PST 100, and describes how to use and care for the terminal. Specific topics include start-up procedures, operating modes, menu use, keyboard and screen features, maintenance, and trouble-shooting.

PST 100 User's Guide (DOC6136-001): This manual describes how to develop application programs that communicate with the PST 100 terminal. The reader is assumed to be familiar with PRIMOS, the PST 100, and at least one high-level language such as COBOL or FORTRAN. Topics include screen writing, visual and logical attributes, data transmission, and the PST 100's programmable features and command syntax. Simple examples of COBOL code are provided to illustrate coding techniques.

Languages

Pascal Reference Guide, Second Edition (DOC4303-191): This book is a programmer's reference guide to the Pascal language as implemented on Prime computers. Readers are expected to be familiar with the Pascal language and with programming in general, but not necessarily with Prime computers. Topics covered include:

- Compilation, loading, and execution of Pascal programs
- The fundamental elements of Pascal syntax and program structure
- The use of Pascal data types, expressions, executable statements, procedures, and functions
- Data input and output in Prime Pascal
- Prime restrictions and extensions to standard Pascal
- Guidelines for interfacing Pascal to Prime's other high-level languages

FORTRAN 77 Reference Guide Update (UPD4029-191): This update package contains the first set of changes for the January, 1982 edition of the FORTRAN 77 Reference Guide (DOC4029-183). With this package, the reference guide is brought up to date for Master Disk Revision 19.1. The package consists of a set of change pages. Modifications made to the text since the last printing are identified by vertical bars in the margins. With change pages inserted, the original guide takes on the new part number DOC4029-191.

FORTRAN 77 Reference Guide Programmer's Companion (FDR4030): This book has been updated from Rev. 17 to Rev. 18. The companion is a handy pocket-sized reference that summarizes FORTRAN 77 statements, formats, data types, intrinsic functions, operators, error messages, and more.

Data Management Products

The new DBMS Data Description Language Reference Guide and DBMS Data Manipulation Language Reference Guide, described below, update and replace the following three books:

- DBMS Schema Reference Guide (PDR3044)
- DBMS FORTRAN Reference Guide (PDR3045)
- DBMS COBOL Reference Guide (PDR3046)

DBMS Data Description Language Reference Guide (DOC5717-181): This manual is a complete reference for the use of the DBMS Data Description Language (DDL). This book is intended for the Data Base Administrator and for the user who will implement a DBMS schema or define a subschema for use with application programs.

After briefly describing the place of DDL in DBMS, this book details the rules and syntax coding for the schema and for COBOL and FORTRAN DDL. The book also discusses the compilation of schema and subschemas and provides suggestions for efficient data description. Included in the appendixes are comprehensive listings of schema and subschema DDL syntax, DDL reserved words, and schema compiler error messages. Sample compilation listings for a schema, a COBOL subschema, and a FORTRAN subschema are also provided.

DBMS Data Manipulation Language Reference Guide (DOC5308-190): This manual explains how to use the Data Manipulation Language (DML), a language designed to access Prime's CODASYL compliant Data Base Management System (DBMS). The DML is defined, and some of the fundamental aspects of data base programming are discussed. Topics include compilation of DML programs, communication between DBMS and application programs, and error handling. This book also includes sample COBOL and FORTRAN code; reference listings of DML commands, syntax, reserved words, and error messages; and a Data Description Language (DDL) syntax summary.

Prime Technical Update: MIDASPLUS (PTU98): MIDASPLUS is a new data management product that is is fully compatible with MIDAS but offers substantially improved performance. This Prime Technical Update contains information on:

- The advantages of using MIDASPLUS
- Compatibility between MIDASPLUS and MIDAS
- Programmer-level differences between MIDASPLUS and MIDAS
- Administrator-level differences between MIDASPLUS and MIDAS
- Limits and precautions for MIDASPLUS users
- Internal fatal errors
- The MDUMP utility (also described in the section on MIDAS in Chapter 4 of this Software Release Document)

The MIDASPLUS PTU should be read in conjunction with the MIDAS User's Guide (IDR4558).

INFORMATION Systems

INFORMATION Configuration Guide (DOC6095-051): This guide describes hardware and software configuration requirements for Prime INFORMATION systems. Topics include cable requirements, equipment specifications, and upgrade policies.

INFORMATION Operator's Guide (DOC3907-052): This manual is an illustrated guide for the INFORMATION system operator. It describes all the equipment normally associated with an INFORMATION system and contains directions for system startup, shutdown, backup, and monitoring. The reader is assumed to have some familiarity with computers and with peripherals equipment. However, references for further detail or background are provided.

INFORMATION System Update, Release 5.2 (MRU5964-002): This update lists new features, enhancements, and software problems corrected in INFORMATION software at Release 5.2. (Release 5.2 is based on PRIMOS Rev. 18.4.)

INFORMATION System Update, Release 5.3 (MRU5964-003): This update lists new features, enhancements, and software problems corrected in INFORMATION software at Release 5.3. (Release 5.3 is based on PRIMOS Rev. 19.1.)

Office Automation Systems

Prime Technical Update: New PT65 Keyboard (PTU95): This PTU describes a new keyboard for the PT65 terminal (Administrative Workstation). Changes in keybaord layout and functionality are discussed.

CHAPTER 2

PRIMOS AND UTILITIES

PRIMOS

NEW FEATURES AND CHANGES

> WARNING
>
> Any user software that uses the first word of the three spare words in a directory file header must be changed. As of Rev. 19.1, this word is used by PRIMOS.

I/O Improvements

At Rev. 19.1, I/O throughput has been increased on all Prime computers. Most of the operating system changes that contribute to this improvement are invisible to the user. The visible changes are described in this section.

New CONFIG Directives

Three new configuration directives are available at Rev. 19.1. Two of them, NLBUFS and PRATIO, are described below. The third, ICS JUMPER, is described in the section called ICS1 Controller Support in Chapter 7.

NLBUFS Directive: PRIMOS incorporates a memory-to-disk cache that stores the most recently and most frequently accessed disk records, thus reducing disk I/O. This cache is made up of a number of buffers called LOCATE buffers, or associative buffers. Prior to Rev. 19.1, there were always 64 LOCATE buffers. At Rev. 19.1, the System Administrator can allocate from 8 to 256 LOCATE buffers at system startup. The new LOCATE buffers reside in segments 50 through 53.

LOCATE buffers are allocated by means of the new configuration directive NLBUFS. The format of the directive is:

```
NLBUFS n
```

where n is the number (in octal) of LOCATE buffers desired. The

default is 64 ('100 octal). Allocating more LOCATE buffers decreases disk I/O, but also takes up memory. For example, 256 buffers require one-half megabyte of memory. A paging system with 256 buffers allocated should therefore add at least one-half megabyte of memory to realize any gains.

The optimal number of LOCATE buffers depends upon the applications running on the system. These buffers are most useful when applications access the same file records repeatedly. If the LOCATE miss rate (as reported by the USAGE command) is greater than 10% and the paging rate is low, more buffers should be configured.

PRATIO Directive: Prior to Rev. 19.1, an alternate paging device was only selected if and when the primary paging device became full. At Rev. 19.1, the System Administrator can determine the ratio of alternate device use to primary device use. This ratio depends upon the new configuration directive PRATIO. The format of the directive is:

```
PRATIO n
```

where n is an octal number from 0 to '12. (The default is 5.) The alternate paging device is selected n times in 10. If the selected device (primary or alternate) is full, the other device is used. If both devices are full, an error code is returned.

PRATIO should be selected to balance traffic to the primary and alternate paging disks. The USAGE command can be used to monitor disk traffic. In a heavy paging environment where the primary and alternate devices are on different drives, the flexibility provided by the PRATIO directive can result in improved throughput.

MAXSCH

The variable MAXSCH in SUPCOM controls the amount of overlapped processing performed by the system. MAXSCH is set by the operator command MAXSCH, as described in the System Operator's Guide (DOC5038-190). At Rev. 19.1, PRIMOS uses an improved algorithm to calculate the default value of MAXSCH. The default value now depends upon factors such as the amount of main memory on the system and the type of CPU. This change results in increased throughput.

Disk Queue Control Blocks

PRIMOS uses a queue to control disk I/O requests from user processes. At Rev. 19.1, this queue has been expanded to use up to 17 queue control blocks. The result is an increase in efficiency that is especially apparent when MAXSCH is set to more than 7.

Logging of Priority ACLs

At Rev. 19.1, whenever a priority ACL is added to the system, a PACL event is added to the system log file in LOGREC*. In addition, a message is printed at the supervisor terminal. The message includes the date and time at which the priority ACL was added and the ID and number of the user who added it. This feature is intended to improve system security. No event or message occurs when a priority ACL is removed.

New CREATE Options

At Rev. 19.1, the CREATE command has been enhanced to allow you to:

- Set a quota for the newly created directory
- Protect the new directory with an access category

The command format is:

```
CREATE directory_pathname [-PASSWORD] [-MAX n]
                          [-CATEGORY category_name]
```

The -MAX option creates the directory as a quota directory and sets the quota to n. n must be a decimal integer between 0 and the maximum number of records on the disk partition.

The -CATEGORY option protects the new directory with access category category_name. category_name must reside in the same directory as the newly created directory. The .ACAT suffix is supplied automatically by PRIMOS.

The -MAX and -CATEGORY options may not be specified if -PASSWORD is specified.

For more information on the CREATE command, refer to the PRIMOS Commands Reference Guide (FDR3108-190).

User Control of Execution Priority

Each user on a Prime system is assigned an execution priority level. Priority levels range from 0 through 3. Users with higher priority levels receive more "attention" from the operating system than do users with lower priority levels. The default priority for terminal users is 1.

Before Rev. 19.1, a user's priority could be changed only by means of the operator command CHAP. At Rev. 19.1, two new user commands, CHAP UP and CHAP DOWN, give users limited control over their own priority levels. The operator CHAP command has not changed at Rev. 19.1. CHAP is described in the System Operator's Guide (DOC5038-190).

With the CHAP UP and CHAP DOWN commands, a user may change his or her execution priority level within an assigned range. The low end of the range is always 0. The high end of the range is the priority most recently assigned to the user by the operator CHAP command. Thus, if the operator sets a user's priority to 2, the user may vary his or her priority between 0 and 2. If the CHAP command has not been issued for a user, then by default the high end of that user's range is 1.

The CHAP UP command raises the user's priority by one each time it is issued (up to the top of the range for that user). The CHAP DOWN command lowers the user's priority by one each time it is issued (down to the lowest priority of 0).

Changes Concerning Paging Partitions

At Rev. 19.1, PRIMOS supports an unlimited number of badspots per paging partition. Prior to Rev. 19.1, a paging partition could have a maximum of 16 badspots.

At Rev. 19.1, split paging partitions can be assigned any legal PRIMOS filenames. Prior to Rev. 19.1, split paging partitions had to be called PAGING or ALTPAG unless the command device and the paging device were the same.

External Logout

The initial attach point is now preserved during the execution of an external logout.

STATUS COMM Command

A new option has been added to the STATUS command at Rev. 19.1. The STATUS COMM command provides information about the communications controllers (excluding the PNC) present in a system backplane. The format of the command is:

```
STATUS COMM
```

The command displays a table describing each ICS1, AMLC, MDLC, HSSMLC, and SMLC controller present in a system. The following information is provided:

- Controller name (for example, MDLC, ICS1)
- Controller type (for example, DMQ for AMLC; 5646 for MDLC)
- Device address (the octal device address of the controller)
- Number of operable asynchronous lines on the controller
- Number of operable synchronous lines on the controller

The following is an example of the STATUS COMM command:

```
OK, status comm

                        Device      Lines
 Controller   Type      Address   Async Sync

 ICS1                     36        8    1
 MDLC         5646        50        0    4
 AMLC         DMQ         54       16    0
```

The information displayed in the "Type" field is controller-dependent. The possibilities are as follows:

- For AMLC controllers, DMT indicates a 50xx controller (old style) and DMQ indicates a 51xx controller (new style).
- For MDLC controllers, a 4-digit PROM set id number is displayed. For example, 5646 is displayed for bisynchronous communications.
- For SMLC, HSSMLC, and ICS1 controllers, no information is displayed.

If there are no communications controllers present in a system, STATUS COMM displays the following message:

```
No communications controllers defined
```

If PRIMOS is unable to identify a controller that has a device address reserved for communications controllers, STAT COMM displays the message:

```
Unknown Controller with communications device address
```

followed by a list of unknown controllers' device addresses. For example:

```
Unknown Controller with communications device address

device address      device id
     10
```

The device id field is not used at Rev. 19.1.

USAGE Command

The USAGE command displays three new parameters:

Parameter	Meaning
%ASYNC	CPU percentage used for ICS1 asynchronous support
%SYNC	CPU percentage used for ICS1 synchronous support
%ICS	CPU percentage used for ICS1 interface support

The ICS1 is the new communications controller that is introduced at Rev. 19.1. Refer to Chapter 7 of this book for more information.

In addition, the USAGE command has been moved from the TOOLS directory to the CMDNC0 directory on the Master Disk.

Memory Size Message

In the memory size message that is printed at cold start, memory size is now expressed in bytes rather than words.

Alternate Command Names

The RLS, DMSTK, and REN commands now have alternate names: RELEASE_LEVEL, DUMP_STACK, and REENTER, respectively. The shorter names may be regarded as abbreviations.

Shared Segments

Shared segments in DTAR1 are now allocated from segment '2000 to segment '2377 rather than from '2000 to '2277.

SOFTWARE PROBLEMS FIXED

Fixed at Rev. 18.4 and Rev. 19.1

ADDISK: If the RENAME option of ADDISK fails, the system lock is now released correctly (POLERs #41564, 40114).

AMLC: The setting and clearing of the LWORD bits now work correctly when AMLC lines are assigned and unassigned (POLER #32702).

The performance of T$AMLC has been improved in cases where the OUTPUT BUFFER FULL condition occurs frequently.

Command Files: Superfluous ATTACH commands were sometimes being issued internally when command files were executing. This problem has been corrected.

Disk Drives: A problem that was causing incorrect header check disk errors has been corrected.

Floppy Disks: PRIMOS can now access track 0 of IBM-format floppy disks.

Installation: The INSTALL.ALL.COMI and INSTALL.STD.COMI files have been updated to include VRPG.

Message: The MESSAGE command was sometimes losing the last character of a message. This problem has been corrected.

Passwords: Lowercase UFD passwords now function correctly (POLER #27618).

PRERR: PRERR has been removed from the list of gated PRIMOS routines. The routine PRERR$ performs a similar function. A problem that caused the system to crash when PRERR was supplied with a single nonpositive argument has been corrected (POLER #43324).

Prime 850: The Prime 850 can now be warm started properly.

Printers: XOFF is now processed in time to avoid buffer overruns in NEC spinwriter printers. Output is resumed correctly if XON/XOFF is disabled while output is suspended (POLERs #32445, 42687).

RDLIN$ and WILIN$: Negative count arguments in calls to RDLIN$ and WILIN$ no longer cause network problems. WILIN$ now recognizes a disk full condition and no longer writes over a previously written portion of the file (POLER #45589).

SMLC: Users with user numbers greater than 63 are no longer prevented from assigning synchronous lines (POLER #34096).

Tape Dumps: An error in the tape dump routine has been corrected, and error recovery has been added. In addition, a tape dump may now be sent to any tape unit (POLER #40048).

Versatec Plotter: When one process unassigned the plotter and the next process assigned it, sometimes a line of data was lost. Also, when a blank line was to be printed, sometimes the second character of the previous line would be printed instead. These problems have been corrected (POLER #45555).

Version Stamp: The program VERSION_STAMP.CPL now displays the version of PRIMOS currently running. The program VERSION.CPL extracts the necessary information from COLD.COMI to create VERSION_STAMP.CPL.

Fixed at Rev. 19.0 and Rev. 19.1

AMLTIM CONFIG Directive: The default value for the second argument of the AMLTIM CONFIG directive is now correctly calculated (POLER #82632).

Cold Start: During cold start, if PRIMOS cannot attach to CMDNCO because of a "bad password" error, an error message will be printed (POLER #23089).

COMOUTPUT: Command output files may now be closed only by the COMOUTPUT -END command, not by the OPEN or CLOSE commands.

SMLC: Unassigning an SMLC line will no longer cause the system to crash if another SMLC line, line 4 or greater, is using the same segment as the unassigned line (POLER #29196).

Pathnames: The internal PRIMOS pathname processing routine, TA$, now correctly handles the specification "<volume>filename" by attaching to the MFD, allowing the specified operation to reference filename in the correct place. This affects the LISTING, BINARY, and OPEN commands.

WILIN$: WILIN$ now recognizes a disk full condition and no longer writes over a previously written portion of the file (POLER #45589).

300 LPM Printer/Plotter: When the 300 lpm printer/plotter is nearly done printing the last file in the spool queue, it no longer prints the remaining lines at a rate of one line per minute (POLERs #20655, 32622, 31068).

First Fixed at Rev. 19.1

Echoing: The DUPLX$ subroutine now correctly disables echoing. Programs that disable echoing via DUPLX$ and then use their own echoing no longer receive double echoes when run from CPL or COMINPUT files (POLER #46805).

Wildcards: A problem with wildcard matching has been corrected.

DELETE: Under certain circumstances, DELETE was reporting that it had deleted files when in fact the files were not deleted because the user did not have the required access rights. In particular, this happened when the user had non-owner rights to a passworded directory, and issued the DELETE command with the -NQ option but without the -FORCE option. This problem has been corrected (POLER #57909).

FORCEW Subroutine: In certain cases, if the FORCEW subroutine was used just prior to a system crash, some of the data that was supposed to have been written to disk was lost. This problem has been corrected.

LD: If a user did not have an attach point, the LD command listed the CMDNC0 directory instead. This problem has been corrected (POLER #57900).

An appropriate error message is now supplied if the user tries to invoke LD for a file.

PRIMOS II: When a Rev. 19.0 disk with new directories was added under PRIMOS II, PRIMOS II set a parameter in the UFD header to 0. PRIMOS then interpreted that 0 to mean that no quota handling was supported for the disk. This problem has been corrected (POLER #57978).

STATUS ME Command: The STATUS ME command no longer lists all user ids whose first six characters match those of the invoking user (POLER #57889).

Event Log: The event logger now checks that the date and time have been set, and thus no longer creates files called LOG.IN/AL/**.

PRIMOS II

SOFTWARE PROBLEMS FIXED

Fixed at Rev. 19.0 and Rev. 19.1

Tape Drive Density: PRIMOS II now initializes model 3 1600/6250 BPI Capable Magtape drives so that the front panel density selection switch is enabled.

AVAIL

SOFTWARE PROBLEMS FIXED

Fixed at Rev. 19.0 and Rev. 19.1

AVAIL *: The AVAIL * command now correctly handles a SYSTEM>DISCS file that has no comments field (POLER #45591).

Batch

SOFTWARE PROBLEMS FIXED

Fixed at Rev. 19.0 and Rev. 19.1

Data Base Locking Mechanism: Batch now uses a named semaphore, rather than a sleep loop, to assure prompt service while attaining its internal data base lock.

File in Use Timeouts: The timeout on files that a Batch program (JOB, BATCH, etc.) is trying to open has been raised from 30 seconds to 60 seconds.

Message Accept State: Batch no longer resets the user's message acceptance state when it needs to send a message to the system console. Therefore, the message:

(Batch) I have reset your message state to -ACCEPT.

no longer exists.

Multiple Monitors: The check for the spawning of multiple Batch monitors will now more reliably produce the "Multiple monitors illegal" error message rather than a "File in use" error on the O_LOG file in BATCHQ.

CMPF

SOFTWARE PROBLEMS FIXED

Fixed at Rev. 19.0 and Rev. 19.1

Closing Files: CMPF will now close files by unit number instead of by filename.

COPY

SOFTWARE PROBLEM FIXED

First Fixed at Rev. 19.1

Deleted Files: Under certain error conditions, COPY deleted the source object. When an error message is printed at Rev. 19.1, the source object is not deleted.

COPY_DISK

NEW FEATURE

At Rev. 19.1, COPY_DISK attempts to calculate disk size from the DSKRAT. If disk size cannot be calculated, COPY_DISK asks the user what type of disk is being copied.

Note

COPY_DISK is not intended for use with completely unformatted disks at Rev. 19.0 or Rev. 19.1.

DOCUMENTATION CORRECTION

The following entry should be added to the list of COPY_DISK error messages in Appendix F of the System Operator's Guide (DOC5038-190):

- IF YOU DO NOT WISH TO CONTINUE WITHOUT BADSPOT HANDLING YOU WILL NEED TO RE-MAKE PARTITION xxxxxx

  Indicates that the target partition cannot accomodate the source partition. For example, appears when the source partition is full and the target partition has at least one badspot. Appears in conjunction with the message:

  NO FREE RECORDS ON PARTITION xxxxxx

CPL

SOFTWARE PROBLEMS FIXED

Fixed at Rev. 18.4 and Rev. 19.1

CALC: The CALC function now correctly handles + and - operations following parentheses (POLERs #37520, 42669).

Error Messages: An error message is now supplied if the CPL command fails (POLERs #40711, 41511).

&TTY_CONTINUE: &TTY_CONTINUE now works as documented (POLER #43979).

&ARGS: &ARGS now corrcetly sets a severity code if a faulty argument is used (POLER #40177).

&ARGS now accepts numeric option arguments, as documented (POLER #45173).

&WILD: The WILD function now accepts the options -FILE, -DIR, and -SEGDIR for consistency with the EXIST function (POLER #35777).

-TTY Option for QUERY and RESPONSE: The -TTY option has been added to the QUERY and RESPONSE functions to allow these functions to force on terminal input (POLER #48206). This option is fully documented in the Rev. 18.4 Software Release Document (MRU4304-007).

&EXPAND: The &EXPAND directive caused spaces to be compressed on subsequent lines. This problem has been corrected (POLER #41561).

Non-octal Arguments: CPL programs invoked from the CMDNCO directory did not accept numeric arguments containing non-octal digits. This problem has been corrected (POLER #41514).

Fixed at Rev. 19.0 and Rev. 19.1

Nonlocal &Goto: A CPL nonlocal &GOTO from a CPL on-unit to a "START <address>" command now works (POLER #41507).

Editor

NEW FEATURE

A new mode, INFO, has been added to the Editor at Rev. 19.1. If you issue the command MODE INFO, all subsequent carriage returns in command mode are interpreted as NEXT commands. (To enter INPUT mode, you must type the command INPUT.) The command MODE NOINFO returns you to ordinary operating mode. NOINFO is the default mode.

MODE INFO has been added to the Editor to support Prime INFORMATION users, who are accustomed to working in this mode.

DOCUMENTATION CORRECTIONS

The MODE COUNT command cannot be abbreviated to MODE COU. This correction should be noted in the following places:

- Page E-3 of the Prime User's Guide (DOC4130-190).
- Page 9-8 of Update Package COR3104-002 for The New User's Guide to EDITOR and RUNOFF (FDR3104).

The PRINT, BLANK, and SUPPRESS options of MODE COUNT may be abbreviated P, B, and S, respectively. This correction should be noted on page 9-8 of Update Package COR3104-002.

SOFTWARE PROBLEMS FIXED

Fixed at Rev. 19.0 and Rev. 19.1

FILE Command: The FILE command no longer requires Delete access on the specified file. Before Rev. 19.0, Delete access was required to truncate the file (since it may have become shorter as a result of editing the file). As of Rev. 19.0, the file system allows truncation of a file if the user has Write access; therefore, a FILE command may be issued to a file to which the user has RW (Read and Write) access (POLER #29963).

EMACS

NEW FEATURES

Rev. 19.1 EMACS introduces two sets of changes:

- Changes that affect users who do little or no customization
- Changes that affect people who use the Prime EMACS Extension Language (PEEL) to customize and extend EMACS's functionality

Functionality Changes Visible To All Users

With EMACS 19.1, some of the earlier functionality that was supplied by the PEEL libraries has been moved into the core product.

The PST 100 terminal is supported in both fundamental and Standard User Interface modes. To invoke EMACS on this terminal, use the EMACS -TTP option with an argument of PST100. For example:

```
EMACS -TTP PST100
```

Four other new terminal drivers have also been added. They are the Pericom 6800 series (-TTP PERICOM6, -TTP PERICOM 7), the Lynwood GD1 (-TTP P.GD), the Newbury laboratories 7000/8000 series (-TTP NL7004, -TTP NL8009), and the Televideo 912/902c (-TTP TVI912, -TTP TVI920).

There are three new command line options:

- -SUI invokes the Standard User Interface
- -SUIX invokes the Standard User Interface With Extensions
- -ECHO_CPL causes EMACS to perform screen updates while reading a CPL or COMINPUT file. This option only has meaning if EMACS was invoked by CPL or COMINPUT.

The EMACS Standard User Interface (SUI) is an enhancement of the PT45 Standard Interface. Certain PT45 Standard Interface features which may have been confusing to new users are not included in the new SUI. However, all PT45 Standard Interface features are available in the EMACS Standard User Interface with Extensions (SUIX), which is described later in this section.

The PT45 keyboard overlay, IDR6135-001, should be used with both the PT45 Standard Interface and the new SUI. In addition to the labeled areas on the overlay, SUI supplies two new functions — a "copy region" and an "exchange mark and cursor" function. These functions are accessed by {CONTROL-f14} and {CONTROL-f13}, respectively.

The PST 100 keyboard overlay, to be supplied in the future, supports SUI for PST 100 users. The part number for this overlay is not available at this time.

The same functionality is available in both the PST 100 and PT45 versions of the SUI. The only major differences are:

- The actual locations of the keys
- The online HELP information, which is customized for each of the two terminals

The EMACS Standard User Interface with Extensions (SUIX) is a superset of both the SUI and the PT45 interface. In the SUIX, all functions of the PT45 interface are available and have been improved. Some new functions are available.

The SUIX uses the same keyboard overlays as the SUI and supports the same two terminals. In addition to supplying the standard SUI functions labeled on the template, all of the fundamental and library commands are available except where their keystroke patterns conflict with the characters issued by the PT45/PST 100 function keys. These conflicts are described in the EMACS Standard User Interface Primer.

The PT45 command may still be used. It now invokes SUIX. It also displays an information message in the minibuffer recommending use of either -SUI or -SUIX.

The language modes are not accessible from SUI. Users may access them from SUIX or from Fundamental Mode.

The file EMACS*>INIT.EM has been deleted. The two commands it provided, LOAD and PT45, are now part of the core product. INIT.EM is still looked for and executed if found, but Prime no longer supplies it. If a particular site wishes to have EMACS initialize in a customized way, the System Administrator should add a file EMACS*>INIT.EM. EMACS will find and load_pl_source it by default as part of the initialization sequence.

Installations that do not provide a special INIT.EM for their users should recommend that people use the -NULIB option unless they wish to initialize with private user libraries. Use of the -NULIB option improves initialization because it tells EMACS not to bother looking for EMACS*>INIT.EM.

The -ULIB option has been improved. If the user invokes EMACS with -ULIB MYFILE, EMACS first searches the current directory for the file MYFILE. If MYFILE is found, EMACS performs a load_pl_source on it. If

EMACS fails to find MYFILE, it will search for MYFILE.EM and load_pl_source that file as part of the initialization process. If EMACS fails to find MYFILE.EM, it will search for MYFILE.EFASL and fasload it if it is found. Users can force EMACS to either load_pl_source or fasload an initialization file by specifying the full filename; e.g., -ULIB MYFILE.EFASL finds and fasloads MYFILE.EFASL independently of the existence of MYFILE.EM in the connected directory. Full pathnames can be used.

The EXPLORE command has improved user feedback. The SET_SPOOL user feedback within EXPLORE mode has been improved.

The HELP_ON_TAP command has improved user feedback.

The DESCRIBE command has improved user feedback and now allows exits with "q" and carriage return, upon initial invocation.

The QUERY_REPLACE command has improved user feedback.

Functionality Changes Visible to PEEL Programmers

In order to improve the initialization performance, some of the functionality provided by the standard PEEL library software has been integrated into core EMACS. The converted PEEL modules are therefore obsolete and are kept in EMACS*>EXTENSIONS>OBSOLETE_SOURCES, which is a new directory. The .EFASL version of these files is still in EMACS*>EXTENSIONS to maintain compatibility. These .EFASL files are not used by the core system. They may still be fasloaded, but this is not recommended. Where possible, the System Administrator should delete these files and convert users to using the faster built-in equivalent.

EMACS LIBRARY Customized Library Users

New LOAD Command: EMACS initialization will be faster because your customized interface is built on basic EMACS, which now comes up much more quickly. Furthermore, most customized interfaces call for the fasloading of three large files:

EMACS*>EXTENSIONS>DEFINITIONS.EFASL

EMACS*>EXTENSIONS>AUTOLOAD.EFASL

EMACS*>EXTENSIONS>KEY_ASSIGNMENTS

These files have been brought within EMACS at Release 19.1. While those files are still available for fasloading, there is a new EMACS command, (load), which peforms the same operations faster.

The first thing you should do, therefore, is to inspect all the files in your EMACS environment, looking for code that fasloads any of these files. If you do call one of these three files, remove the references from your file and add the load command. For example, many EMACS customizers have the following commands in their start-up files:

```
(fasload "emacs*>extensions>definitions")

(fasload "emacs*>extensions>autoload")

(fasload "emacs*>extensions>key_assignments")
```

You can now replace these three lines with:

```
(load)
```

The OVERLAY.EM functionality (OVERLAY_ON, OVERLAY_OFF) is available at all times.

The PT45_FUNCTION_KEYS.EM functionality becomes available to PT45 users as part of the PT45, SUI, and SUIX commands.

The DUMP_PACKAGE command in MAINTAIN.EM is obsolete. It does not have to be used on autoload declaration buffers because the DEF_AUTO command is now an EMACS core command.

The SUI and SUIX commands use a special autoloader to bring commands and functions from EMACS*>EXTENSIONS>PRIME_SUI_LIB. System Administrators are strongly advised against changing any of the files in this UFD, or adding new ones, because Prime reserves the right to remove this UFD and convert the PEEL to core functionality.

The language mode libraries no longer load in the COMPILE.EFASL file. The COMPILE.EFASL is not loaded until the COMPILE command is given. The key bindings for lm_next_error$ and lm_prev_error$ are now set up by the compile command.

DOCUMENTATION CORRECTIONS

The following corrections apply to the EMACS Reference Guide (IDR5026).

Explore Mode

On page 16-6, insert the following entry before the entry for the G command:

▶ B

This command moves the cursor to the beginning of the previous line.

On page 16-7, insert the following entry before the entry for the ESCAPE X Set_spool_options command:

▶ {SPACE}

The space character moves the cursor to the beginning of the next line.

User Type

On page A-5, line 16 of text, change "(setq user_type 'type)" to "(setq user_type$ 'type)".

CONTROL-Q

On page 4-1, in the second paragraph of the CONTROL-Q entry, change "stop printing" to "start printing".

CONTROL-S

In the note at the top of page 8-2, change "start printing" to "stop printing".

Buffer Commands

Add the following text to the bottom of page 5-4:

▶ CONTROL-X B {RETURN}

This command returns you to the previous buffer.

CONTROL-_A

In the warning on page A-2, change "APROPOS" to "CONTROL-_A".

{ESC}0 CONTROL-L

To move point and the current line to the top of the screen, use the command {ESC}0 CNTRL-L. This command is documented under CONTROL-L on page 4-6, but should have its own entry.

The following correction applies to the EMACS Extension Writing Guide (IDR5025).

Explore Mode

The wallpaper listing on page 8-1 indicates that the B command is bound to prev_line_command and the space character is bound to next_line_command. This is incorrect. These commands are bound to the prev_line and next_line functions, respectively. Because EMACS functions do not have associated documentation strings, a wallpaper listing does not reveal what action occurs when the key is pressed.

SOFTWARE PROBLEMS FIXED

Fixed at Rev. 19.0 and Rev. 19.1

Corrupted Text: The bug that caused EMACS to corrupt text with control characters and ILLEGAL SEGNO$ messages has been fixed (POLER #31723).

First Fixed at Rev. 19.1

Terminal Type: If the user specifies NONE as the terminal type, EMACS now immediately exits to Primos with ER! (POLER #43673).

Comment Delimiters: The install files EMACS.INSTALL.COMI and EMACS.INITINSTALL.COMI have been fixed. They now use the correct comment delimiters (POLER #45963).

EXPAND_MACRO: EXPAND_MACRO now converts user-supplied macro names to lowercase before performing the expansion. This is because all EMACS commands and functions must be lowercase unless they are to be accessed strictly from the PEEL extension language (POLERS 52243).

EXPLORE: The EXPLORE package now works with passworded directories.

DESCRIBE: The DESCRIBE command no longer fails when asked to describe & or other single-character nonalphanumeric printing characters.

Compile-related Commands: The compile-related commands set_lm_list_path$, set_lm_bin_path$, set_ftn_compile$, set_f77_compile$, set_rpg_compile$, and set_vrpg_compile$ have been fixed.

The compile-related commands set_cobol_compile$ and set_cbl_compile$ have been added. These commands are described in the EMACS Reference Guide (IDR5026), but were not shipped in Revisions 18.3 through 19.0.

FORTRAN Mode: FORTRAN mode no longer takes a stack overflow if there is a comment on the first line of the file.

Recursion: Infinite PEEL recursion no longer causes stack overflow. The stack size has been increased to 3 segments and recursion is limited to 100 levels.

SOFTWARE PROBLEMS OUTSTANDING

Wildcards

At PRIMOS Rev. 18, there was no wildcarding on the PRIMOS command line. Thus users were able to initialize EMACS and have it load all files meeting a wildcard specification given on the command line. For example, a request to initialize with file MYFILE@@EM would start up EMACS with buffers (myfile.000.em) and (myfile.lll.em) if those two files matched. However, at PRIMOS Rev. 19, wildcards should not be used on a command line to EMACS because they will cause reentry for each file match. The above example would initialize once with the single buffer (myfile.000.em) and again, after the user quits, with the second buffer (myfile.lll.em).

There are two simple workarounds for this problem. First, the user can quote the wildcards on the command line; second, the System Administrator can make a copy of CMDNC0>EMACS.SAVE as NX$EMACS.SAVE and have users invoke EMACS as NX$EMACS.SAVE. (SAVE files in CMDNC0 whose names begin with NX$ do not have their wildcards expanded.)

Type-ahead

Users of the PST 100 must not type ahead while waiting for EMACS to initialize. Type-ahead may interfere with the string EMACS gets from the terminal describing its configuration (i.e. reverse video, soft-scroll, etc.) EMACS uses this string to return the terminal to its initial state during EMACS exit.

PST 100

Users of the PST 100 terminal must exit from the primos_internal_screen command by giving a single character command such as control-L (refresh) or control-G (abort). (Single-letter characters also work, but they are inserted into the working buffer upon reentry.) The use

of PST 100 function keys that emit more than one character interferes with the process of resetting the terminal configuration string upon reentry to the normal EMACS display. (The primos_internal_screen command is used in the SUI commands stat, attach, and listf, and is available by name.)

Upon exit to PRIMOS, some aspects of the PST 100 configuration are not properly restored. The most notable of these is 2-page mode. If the terminal was in 2-page mode before EMACS was used, it will be in 1-page mode upon exit.

-XOFF vs. -NOXOFF

In previous versions, -NOXOFF was the default. At Rev. 19.1, -XOFF has been made the default for the PT45 and the PST 100. Consequently, the {CONTROL-S} and {CONTROL-Q} keys may not be used. The functions affected are:

{CONTROL-S}	^s_forward_search_command
{CONTROL-Q}	^q_quote_command
{CONTROL-X}{CONTROL-S}	save_file

In previous versions of EMACS, these functions could be invoked using other keystrokes. Specifically:

{ESCAPE}S	forward_search_command
{CONTROL-X}Q	quote_command
{CONTROL-X}S	save_file

-XOFF mode is the default mode for the FOX, PT45, and PST 100 terminals. People who like to use the ^S command to initiate forward_search should switch to the ESC-s command. The save_file command, formerly bound to ^X-^S, is also available bound to ^X-s. The quote function is also available as ^X-q. Users of SUI or SUIX will be less affected by this change because there are SAVE FILE and FORWARD SEARCH function keys already. Using -XOFF mode does expose them to the ^S key's effect of freezing the display of the PT45. Only ^Q will unfreeze it. -XOFF mode is extremely desirable when the line speed approaches 9600 baud.

It is not advisable to use the PST 100 terminal in -NOXOFF mode.

Terminal Echo

The -ECHO_CPL command line option produces "double echo" while EMACS is reading from a CPL or COMINPUT file. The first echo is the character as it is being removed from the file. The second echo is EMACS's redisplay updating the screen.

FIX_DISK

SOFTWARE PROBLEMS FIXED

First Fixed at Rev. 19.1

-COMDEV Option: When FIX_DISK is run with the -COMDEV option, event logging is reenabled if necessary.

When FIX_DISK is run with the -COMDEV option on a disk with priority ACLs, the priority ACLs are restored on the disk after the disk is added.

-INTERACTIVE Option: The -INTERACTIVE option now allows the user to rebuild the DSKRAT if the disk version number is an unknown or illegal value. FIX_DISK issues the prompt:

DO YOU WISH TO CHANGE THE DISK REVISION?

A response of NO returns the user to PRIMOS. A response of YES elicits a further prompt:

DOES THIS PARTITION SUPPORT ACLS OR QUOTAS?

The user's response determines the new version number of the disk (1 if YES, 0 if NO).

FIX_DISK now includes a description of the -INTERACTIVE option when displaying the available options.

Writing the DSKRAT: A problem in writing out the DSKRAT has been corrected.

FIX_DISK now corrects the current record address of the DSKRAT file if it is wrong.

FIX_DISK now writes the number of cylinders into the DSKRAT header for use by -CONVERT_19.

FIX_DISK now correctly initializes the partition type when recreating the DSKRAT header.

When rebuilding the DSKRAT file, FIX_DISK now asks:

STORAGE MODULE OR CMD?

instead of:

80 OR 300 MB?

Reserved Words: FIX_DISK now ensures that the three reserved words in a file entry are in fact 0, as they should be.

Nested Directories: FIX_DISK now detects directory nesting that exceeds the legal limit.

Badspots: Badspots are now correctly added on disk errors.

If a badspot is detected and there is no BADSPT file defined on the current partition, FIX_DISK no longer prints a spurious ADDED TO BADSPT FILE message.

Valid Entries: FIX_DISK now checks the validity of filenames in directory entries.

FIX_DISK no longer hangs on certain bad directory entries.

Version Check: FIX_DISK now disallows partitions newer than the current FIX_DISK version.

Device Numbers: Physical device numbers with the high-order bit set are now handled correctly.

Error Messages: FIX_DISK now prints names of files and access categories with its ACL error messages.

FIX_DISK no longer produces erroneous DIRECTORY/TREE USED INCORRECT messages.

LOAD

SOFTWARE PROBLEMS FIXED

Fixed at Rev. 19.0 and Rev. 19.1

Return Code: LOAD will now set the warning flag and return code appropriately.

LOGPRT

NEW FEATURE

At Rev. 19.1, the event PACL has been added to the set of events that are reported in the system event log. The event PACL occurs whenever a priority ACL is added to the system. This event is accompanied by a message to the supervisor terminal. (For more information, see Logging of Priority ACLs in the section called PRIMOS, earlier in this chapter.) No event is recorded when a priority ACL is removed.

Note

At Rev. 19.1, the LOGPRT command has been moved from the TOOLS directory to the CMDNCO directory on the Master Disk.

SOFTWARE PROBLEMS FIXED

Fixed at Rev. 19.0 and Rev. 19.1

LFERNEXT: The LFERNEXT bit, which is bit 22 of DSWPARITY, is now correctly interpreted with respect to its meaning.

MAGNET

DOCUMENTATION CORRECTION

The following information should be added to the description of the LABELS option on pages 7-17 and 7-19 of the Rev. 18.4 update to the Magnetic Tape User's Guide (UPD5028-184):

> MAGNET supports only unlabelled tapes. If the LABELS option is used to specify either ANSI or IBM labelled tapes, MAGNET will print a warning message and ignore the specification.

SOFTWARE PROBLEMS FIXED

Fixed at Rev. 18.4 and Rev. 19.1

Writing Binary: Some MAGNET binary writes did not write anything to tape. This problem has been corrected (POLER #40325).

The binary write operation failed on restricted instruction 4000(3)/3211. This problem has been corrected (POLER #47479).

Error Recovery: MAGNET's error recovery has been improved to prevent certain errors from being written to output files (POLER #40765).

&DATA Groups: MAGNET did not handle indentation inside &DATA groups. This problem has been corrected (POLER #43974).

Disk File Truncation: When overwrite options are used in tape-to-disk operations, and if the old file is longer than the new file, MAGNET now correctly truncates the old file (POLER #45411).

Controller Boards: A problem concerning the 2269 and 2270 tape controller boards has been corrected (POLER #45421).

Reading IBM, DEC ASCII: MAGNET can now correctly read IBM and DEC ASCII characters (POLER #824565).

Access Violation: MAGNET encountered an access violation when executed as the first command after login or DELSEG ALL. This problem has been corrected (POLERs #43660, 57974).

Fixed at Rev. 19.0 and Rev. 19.1

Zero LRECL: Using objects where LRECL is set to zero no longer causes an abnormal return to PRIMOS command level.

First Fixed at Rev. 19.1

Translation: The BCD translation tables have been corrected (POLER #45975).

Tapes written to 7-track drives using ASCII-BCD translation are now read correctly.

Log Record Length: MAGNET now accepts a log record length (LRECL) of 1 (POLER #46164).

MAKE

SOFTWARE PROBLEMS FIXED

Fixed at Rev. 19.0 and Rev. 19.1

Badspots: MAKE now continues processing correctly even if the user types NO in response to the ADD TO BADSPOT FILE? prompt.

Clarified Question: The question 80 OR 300MB STORAGE MOD? has been changed to read STORAGE MODULE OR CMD? for purposes of clarity.

PHYSAV/PHYRST

NEW FEATURES

At Rev. 19.1, both PHYSAV and PHYRST have been enhanced to support the Prime 2250. This section describes the changes that are visible to users.

Calculating, Recording, and Checking Disk Size

If PHYSAV cannot calculate disk size (cylinder limits) from the number of records on a partition, it now asks the user what type of disk is being saved. This situation may arise, for example, if the user tries to save more than one partition.

PHYSAV records the cylinder limits of each partition being saved in words 261 through 270 of the tape header, where PHYRST can read them. PHYRST therefore does not have to restore the DSKRAT onto disk before deciding whether the rest of a restore can be accomplished.

Once PHYRST decides to perform a restore, it compares the disk size in the DSKRAT with the disk size recorded on tape. If there is a discrepancy, the restore is not performed and an error message is produced. For example:

```
UNEQUAL SIZE DISKS
THE DISK IS A 68MB DISK NUMBER 461
THE DISK ON TAPE IS A 80MB DISK NUMBER 1061
```

After producing this message, PHYRST reprompts for the disks that are to be restored.

New PHYRST Message

PHYRST now informs the user what types of disks are on a tape. For example:

```
PARTITIONS SAVED
010460  BACK1                   80 OR 300 MB
030462  BACK2                   600 MB
```

Note

PHYRST cannot distinguish between an 80 MB disk and a 300 MB disk, since both have the same cylinder limit.

SOFTWARE PROBLEMS FIXED

Fixed at Rev. 19.0 and Rev. 19.1

End of Tape During GAP: PHYSAV will now recover correctly if it encounters an End of Tape when performing a GAP operation during error recovery; it will continue onto the next tape with the same logical tape number.

REN Command: PHYSAV now works if the user hits the BREAK key and types REN after the END OF REEL, MOUNT REEL and UNIT NO: prompts.

FIX_DISK: PHYRST will no longer tell the user to run FIX_DISK while in verify mode.

RUNOFF

DOCUMENTATION CORRECTIONS

The following corrections apply to The New User's Guide to EDITOR and RUNOFF (FDR3104).

.FILE Command

The .FILE command cannot be abbreviated. On page 10-5, the entry for .FILE indicates that the abbreviation .FIL may be used for .FILE. However, RUNOFF treats .FIL as an abbreviation for .FILL.

Tabbing

The following information should be added to the bottom of page 10-13:

> Text that follows a tab character on a RUNOFF input line is inserted into the output file without fill or adjustment. If the input line is longer than the output line, the extra characters take up a second output line.

SOFTWARE PROBLEMS FIXED

Fixed at Rev. 18.4 and Rev. 19.1

Table of Contents: Long headings in a table of contents are now underlined correctly (POLER #23242).

.DD: .DD statements without headings now work correctly (POLER #29153).

File Units: RUNOFF no longer uses file unit 2 instead of prompting for an output filename (POLERs #29971, 42651).

Underlining: Underlining was not always working correctly when certain combinations of .INDENT and .PARAGRAPH commands were in effect. This problem has been solved (POLERs #31245, 34264, 35560).

Underlining was not occurring when RUNOFF found one or more blanks before the end underline delimiter. This problem has been corrected (POLER #34257).

Underlining now works correctly at the end of an insert file that comes at the end of the entire input file (POLER #34588).

Adjustment: A single word on a line is no longer split and "adjusted" (POLER #40187).

.COLUMNS Command: On pages with two columns, when the number of lines to be output matched the page length (as set by .LENGTH), RUNOFF was omitting the final page. This problem has been corrected (POLER #81527).

.DRESET Command: The .DRESET command now undents text correctly (POLER #82095).

SEG

ENHANCEMENT

At Rev. 19.1, several internal changes have been made to SEG for increased efficiency. Program restore time has been improved, especially for large programs and heavy paging environments.

DOCUMENTATION CORRECTION

In the table on page 4-15 of the LOAD and SEG Reference Guide (PDR3524), in the second entry (addr), change '170000 to '150000.

SOFTWARE PROBLEMS FIXED

Fixed at Rev. 18.4 and Rev. 19.1

Start_address: The start_address is now correctly initialized to the null pointer ('177777/0) rather than to 0/0 (POLER #29718).

File Units: SEG no longer dedicates any file units except unit 13, which is used for the maps (POLERs #20795, 37978).

Redefining Common Blocks: SEG now issues a warning if the user redefines a common block that is shorter than a segment to one that is longer than a segment (POLER #32068).

Stack Overflow and Extension Stacks: Stack overflow and extension stacks now work correctly (POLER #27021).

MODIFY Command: All reported problems with the MODIFY command have been corrected (POLERs #32187, 30102).

Old Runfiles: SEG now runs Revs. 14, 15, and 16 runfiles correctly (POLERs #41718, 40634).

The following three paragraphs are responses to reported problems.

DELETE Subcommand: If the DELETE subcommand is issued from a command file at the beginning of a SEG session, and the file to be deleted does not exist, then SEG aborts the command file. However, the DELETE command is not needed at the beginning of a SEG session, since SEG truncates an existing file before writing to it. Thus, the DELETE command should be removed from the beginnings of command files (POLER #34484).

Start_address: When the SPLIT command is used, the start_address value indicated in the map has no meaning and should be ignored (POLER #32730).

ECB Location: ECBs should be placed in the link frame rather than the procedure frame (POLER #36423).

Fixed at Rev. 19.0 and Rev. 19.1

CMDSEG Rewritten: CMDSEG has been rewritten at Rev. 19.0. It has an improved user interface. For more information, refer to the System Administrator's Guide (DOC5037-190).

SOFTWARE PROBLEM OUTSTANDING

SEG uses segment 4035 for its symbol table (POLER #36475).

SPOOL

SOFTWARE PROBLEMS FIXED

Fixed at Rev. 19.0 and Rev. 19.1

Implied -OPEN: The SPOOL program no longer treats a SPOOL command with no arguments as an implicit SPOOL -OPEN. Now, one of four things must be specified on a SPOOL command line:

- A pathname
- The -LIST option
- The -CANCEL option
- The -OPEN option

If none of these is specified, a helpful message is printed, and the error message "Missing argument to command. Filename required (SPOOL)" is generated.

Large Files: If a file that is larger than 32,767 records is spooled, its size will be recorded as 32,767 records regardless of how large it actually is. The size will no longer wrap around to negative numbers.

Slash Character: The slash (/) character, which is a legal filename character, is now printed in the banner section of the header pages rather than being replaced by a blank.

First Fixed at Rev. 19.1

PROP Command: The PROP -MODIFY command now correctly accepts the options NOW, FINISH, and IDLE on the command line.

Subroutines

DOCUMENTATION CORRECTIONS AND ADDITIONS

The following corrections apply to the Subroutines Reference Guide (DOC3621-190).

Sample FORTRAN calls

On page 5-20, delete lines from THE FILE POINTER ASSOCIATED WITH SGUNIT through FILE CANNOT BE PRESENT (lines 9-17 of the page).

On page 5-23, change GO TO 1000 to GO TO 2000 (lines 33 and 47).

On page 5-24, add the line:

```
INTEGER*2 BUFLNG /*BUFFER SIZE
```

to the program. Line 20 of the program should be:

```
PARAMETER (FUNIT=2, SGUNIT=1, SAMSEG=2, DAMSEG=3, BUFLNG=1)
```

On page 5-26, insert the following after line 17:

```
C POSITION TO REQUIRED WORD, AND READ IN BUFFER
  CALL PRWF$$(K$READ+K$PREA, FUNIT, LOC(BUFF), BUFLNG,
X INTL(WRDNUM), NMREAD, CODE)
  IF (CODE .NE. 0) GO TO 100
```

ATCH$$

In calls to ATCH$$, the key K$ALLD searches all started-up local devices in logical device order and then all remote devices in logical order. This information should be added to page 9-8.

SRSFX$

The suffix-list for SRSFX$ should be declared as (*)CHAR(32)VAR, as it can be a structure of variable strings. This correction should be made on page 9-56.

PAR$RV

Please insert the following entry before the entry for PHANT$ on page 10-34:

▶ PAR$RV

Purpose

This function returns the revision number of a disk partition, given the name of the partition.

Usage

```
DCL PAR$RV ENTRY (CHAR (32) VAR, FIXED BIN) RETURNS (FIXED BIN);

PAR_REV = $PAR$RV (part_name, code);
```

part_name — 32-character varying string containing the partition name

code — error return code

E$FNTF	Partition name not found in disk tables
E$BNAM	Illegal disk partition name

par_rev — partition revision number

0	Pre-ACLs and quotas
1	Converted to allow ACLs and quotas
-1	Error -- see error return code (above)

PRI$RV

Please insert the following entry before the entry for PWCHK$ on page 10-36:

▶ PRI$RV

Purpose

This subroutine returns the revision number of the currently running PRIMOS operating system.

Usage

```
DCL PRI$RV ENTRY (CHAR(32) VAR);

CALL PRI$RV (primos_rev);
```

primos_rev — 32-character varying string containing the PRIMOS revision number

SETRC$

The following text should be inserted before the write-up on TEXTO$ on page 10-44:

▶ SETRC$

Purpose

Sets Return Code

Usage

```
DCL SETRC$ ENTRY (FIXED BIN)

CALL SETRC$(errcode)
```

errcode — Error code value to be returned to the command processor (input). Zero indicates no error.

Discussion

This subroutine permits static mode programs to return an error code value to the command processor that invoked them.

LON$CN

On page 10-31, delete the text on LON$CN.

Physical Devices

Delete the last two lines of the table on page 14-8.

Terminal and Paper Tape Routines

The subroutines described in Chapter 18 expect data to be word-aligned. F77 calls with a substring argument do not always pass the argument left-justified in a word.

TNOU, TNOUA

The count expected by TNOU and TNOUA must be a 16-bit integer (INTEGER*2). This information should be added to page 18-12.

Named Semaphores

In line 4 of page 21-18, replace "Rev. 19" with "Rev. 18.3".

Message Subroutines

Please replace the descriptions of the message subroutines in Appendix B with the following information.

▶ MGSET$

Purpose

MGSET$ is used to set the message receive state of the calling process. The receive state determines the willingness of the process to accept messages sent to it. There are three possible states that a process may have: accept all messages, accept only deferred messages, and reject all messages. Messages that are deferred are not necessarily delivered immediately when sent, but rather are buffered by the system and delivered later. Deferring messages allows the receiver to accept messages at times that are convenient for him or her, rather than at times convenient to the sender. Users may explicitly request waiting deferred messages via the RMSGD$ call, or they may allow the system to deliver deferred messages automatically after PRIMOS commands complete their execution.

Usage

```
DCL MGSET$ ENTRY (FIXED, FIXED);

CALL MGSET$ (key, code);
```

key — Provided by the user. A standard system key that specifies the receive state to be set.

K$ACPT	Accept all messages
K$DEFR	Accept only deferred messages
K$RJCT	Reject all messages

code — A standard system error code returned by the subroutine.

E$BKEY	Bad key
0	No error

▶ MSG$ST

Purpose

MSG$ST allows the caller to determine the receive state of processes. If the caller supplies a specific user number, the receive state and user name of that process are returned. If the caller supplies a user name, the user number and receive state of the most permissive user with the specified name are returned.

Usage

```
DCL MSG$ST ENTRY (FIXED, FIXED, CHAR(*), FIXED, CHAR(*),
                  FIXED, FIXED);

CALL MSG$ST (key, user_num, system_name, system_name_len,
             user_name, user_name_len, receive_state);
```

key — Provided by the user. Can be either of the following:

K$READ	Return the user's name and state for user user_num on system system_name.
2	Return the user's number and state for user user_name on system system_name.

user_num — The user number of the process. If key = K$READ, user_num is provided by the user. If key = 2, user_num is returned by the subroutine.

system_name	The name of the system on which the desired process is found. Provided by the user.
system_name_len	The length of system_name in characters. If system_name_len = 0, the local system is assumed. Provided by the user.
user_name	The user name of the process. If key = K$READ, this parameter is returned by the subroutine. If key = 2, this parameter is provided by the user.
user_name_len	The length of user_name in characters. Provided by the user.
receive_state	The receive state of the process. This parameter can be any of the following:

K$ACPT	Accepting all messages
K$DEFR	Accepting deferred messages only
K$RJCT	Rejecting all messages
K$NONE	User does not exist
K$BKEY	Invalid state, bad key in call
K$BREM	Invalid state, bad system_name

▶ RMSGD$

Purpose

RMSGD$ returns waiting deferred messages to the caller. This routine does not return immediate messages. Users wishing to obtain all messages via this routine must inhibit immediate messages by setting their receive state to receive only deferred messages (see MGSET$ with a key of K$DEFR).

Usage

```
DCL RMSGD$ ENTRY (CHAR(*), FIXED, FIXED, CHAR(*), FIXED,
                  FIXED, CHAR(*), FIXED);

CALL RMSGD$ (from_name, from_name_len, from_num, system_name,
             system_name_len, time_sent, text, text_len);
```

from_name	The user name of the sender. Returned by the subroutine.
from_name_len	The length of from_name in characters. Provided by the user.
from_num	The sender's user number. Returned by the subroutine.
system_name	The name of the system from which the message was sent. Returned by the subroutine.
system_name_len	The length of system_name in characters. Provided by the user.
time_sent	The time, in minutes past midnight, at which the message was sent. If no message is returned, time_sent is set to -1. Returned by the subroutine.
text	The text of the message. Returned by the subroutine.
text_len	The length of text. Provided by the user.

▶ SMSG$

Purpose

SMSG$ sends a message. Messages may either be sent immediately or deferred. Immediate messages are delivered to the recipient at the time the message is sent. Deferred messages are held in a system buffer until the receiver requests them. (Deferred messages are also delivered to a user automatically after each PRIMOS command completes execution.) Messages may be sent to other processes by addressing them to either their user numbers or their user names. If user name is used, all interactive users with that name will receive the message.

Usage

```
DCL SMSG$ ENTRY (FIXED, CHAR(*), FIXED, FIXED, CHAR(*), FIXED,
                 CHAR(*), FIXED, (131) FIXED);

CALL SMSG$ (key, to_name, to_name_len, to_user_num,
            to_system_name, to_system_len, text, text_len,
            error_vector);
```

All parameters except error_vector are provided by the user.

key — Specifies the type of message, immediate or deferred.

- 0 — Deferred message. Messages are buffered and delivered at the receiver's convenience.
- 1 — Immediate message. Messages are delivered immediately when sent.

to_name — The user name of the user to whom the message is to be sent. If to_name is nonblank, the message is sent to all interactive users logged in under that name. If to_name is blank, the message is sent by to_num, and to_name is ignored.

to_name_len — The length of to_name in characters.

to_user_num — The user number of the user to whom the message is sent. If to_num is positive, to_name is ignored. If to_num is zero and to_name is blank, the message is sent to the operator.

to_system_name — The name of the node to which the message to be sent.

to_system_len — The length of to_system_name in characters. If to_system_len is zero, the local system is assumed.

text — The text of the message. Messages may be up to 80 characters in length, and either blank-padded or terminated with a newline. Only printable characters and the bell character are printed by the operating system.

text_len — The length of text in characters.

error_vector — An array that reports the success or failure of the call. Its size can range from 4 through 131. Its elements have the following meanings:

error_vector (1) — An overall status code returned by the subroutine.

E$NRCV	Operation aborted because sender does not have receive enabled.
E$UADR	Unknown addressee.
E$UDEF	Receiver not receiving.
E$PRTL	Operation partially blocked.
E$NSUC	Operation failed.
0	Operation succeeded.

error_vector (2) — Three less than the total number of elements in error_vector. Normally set to the number of configured users (128). Provided by the user.

error_vector (3) — An overall network error code returned by the subroutine.

XS$CLR	Connect cleared
XS$BPM	Unknown node address
XS$DWN	Node not responding

error_vector (4-131) — An optional status vector whose length is the value of error_vector (2). If supplied, each element is a status code returned by the subroutine, indicating success or failure to send a message to user number n-3, where n is the index into error_vector. For example, error_vector (10) is the status for user number 7.

E$UBSY	User busy, please wait
E$UNRV	User not receiving now

RTRN$S

At the top of page 13-23, the phrase:

... calls to RLSE$S to return ...

should read:

... calls to RTRN$S to return ...

CLNU$S

At the top of page 13-24, insert another variable into the example, as follows:

& NIL /* Dummy variable for ignored return values

On the first line of page 13-25, replace the argument (0,0) with (NIL,NIL). A constant should never be used as an argument for a returned value, lest the value of the constant be accidentally changed.

DIR$LS

On page A-24, the following information should be added to the entries for entr-returned and type-counts: entr-returned is the number of entries returned by DIR$LS in the current call. However, type-counts is a cumulative value, recording the total number of entries of each type returned by DIR$LS within the current application.

PRWF$$

In the third paragraph of page 9-22, the first sentence should begin, "On a DISK FULL or QUOTA EXCEEDED error, the file pointer..."

BAD_PASSWORD$

On page 22-27, in the entry for BAD_PASSWORD$, the first sentence should begin, "This condition is raised by the ATCH$$ primitive and its replacements when..."

FINISH

On page 22-30, the entry for FINISH should read as follows:

▶ FINISH

{software, returnable)

This condition is signalled before process termination, usually after files are closed. It closes any open files and returns to the point at which the condition was signalled. This condition is not signalled if the process is prematurely exhausted or destroyed. Available through PL1G.

The default on-unit simply returns.

SIZE

On page 22-39, in the entry for SIZE, change "either a floating-point number or a decimal integer" to "a floating-point number, a decimal integer, or a character string".

STORAGE

On page 22-40, the entry for STORAGE should be corrected to state that this condition is available through PL1G.

SUBSCRIPTRANGE

On page 22-40, the entry for SUBSCRIPTRANGE should state that it is a software, returnable condition.

TRANSMIT

On page 22-41, the entry for TRANSMIT should be corrected to state that this condition is available through PL1G.

SOFTWARE PROBLEMS FIXED

Fixed at Rev. 18.4 and Rev. 19.1

PHNIM$: The PHNIM$ routine now always closes the input file by unit rather than by name, in case the user has the file open on another unit.

CHAPTER 3

LANGUAGES

BASIC/VM

SOFTWARE PROBLEMS FIXED

Fixed at Rev. 18.4 and Rev. 19.1

OLD Command: If the OLD command was used with a nonexistent filename as argument, the foreground file was deleted. This problem has been corrected (POLER #23481).

Abnormal Program Halts: BASICV was causing abnormal and inconsistent program halts. This problem has been corrected (POLER #28791).

SPA Modifier: Using SPA with negative arguments caused program errors. Negative arguments are now treated as zero (POLER #28793).

LIST Following RESEQUENCE: The LIST command following a RESEQUENCE command now works correctly (POLER #28914).

RESEQUENCE Command: The RESEQUENCE command was deleting lines, producing lines with duplicate line numbers, and using incorrect increments. These problems have been corrected (POLER #28792).

ATTACH Command: The ATTACH command of BASICV now accepts pathnames with passwords (POLERs #24783, 33441, 23937, 82477).

CVT$$ Function: Mask 1 (parity bit off) of the CVT$$ function now works correctly (POLER # 33526).

The FNEND Statement: Extraneous FNEND statements in a program were causing a halt with error ACCESS_VIOLATION$. The compiler now flags extraneous FNENDs (POLERs #34821, 41405).

Breakpoints: Setting a breakpoint to a nonexistent line caused unpredictable results. This problem has been corrected (POLER #35153).

MIDAS: Sequential reads now work correctly after deletion of the current record (POLER #35889).

TAB Function: The TAB function was giving erratic results. This problem has been corrected (POLER #41408).

File Writing Errors: Under certain circumstances, writing a file from a BASICV program caused unpredictable results (for example, failure of the BREAK key). This problem has been corrected (POLER #41409).

Spurious blank records were being written to ASCDA files. This problem has been corrected (POLER #36059).

Problems With Large Programs: Very large programs sometimes do not compile because the compiler's constant table is exhausted. Patches are now available upon request (POLERs #42933, 81488).

Spurious error messages were being generated during the editing of large BASICV programs. This problem has been corrected (POLER #35435).

Erroneous Error Messages: The FILE command was producing the error message "File in use. Tree search error" when there was no error. This problem has been corrected (POLERs #82214, 27836).

BASICV was producing the error message "File open on delete" when there was no error. This problem has been corrected (POLERs #33438, 33503, 33558).

First Fixed at Rev. 19.1

Multiuser Record Access: BASICV now correctly handles multiuser access of locked MIDAS records (POLER #45881).

COBOL

NEW FEATURES

COBOL/MIDAS Record Size Compatibility

When you use MIDAS with COBOL, it is strongly recommended that the data record size you give CREATK during template creation match the data record size in the corresponding FD of the COBOL program. (Refer to Page 7-2 of the MIDAS User's Guide.) For example, if an FD for a file in a COBOL program defines an 80-character record, the compiler allocates space (40 words) for this record. When the MIDAS template is created by CREATK, the DATA SIZE should be 40. At Rev. 19.1, when the OPEN statement is executed on a MIDAS file from a COBOL program, these record sizes are checked for agreement. If the data size of the record according to MIDAS is greater than the record as defined in the COBOL program, a fatal error is generated by the COBOL library and a diagnostic message is displayed. The diagnostic message is as follows:

```
ERROR DETECTED WHILE ATTEMPTING TO OPEN A MIDAS FILE.
THE DATA RECORD SIZE OF THIS MIDAS FILE IS BIGGER THAN
THE COBOL PROGRAM'S DATA RECORD AREA FOR THIS FILE.
THE DATA RECORD SIZE ACCORDING TO MIDAS (IN WORDS) IS:  ...
BUT ACCORDING TO THE COBOL PROGRAM IT IS (IN WORDS):    ...

CHECK THE CREATK DIALOG AND THE FD IN THE COBOL PROGRAM FOR THIS FILE.
FILE-ID: ...          OWNER: ...  DEVICE: ...
FATAL RUN-TIME I/O ERROR.
```

This runtime check is done to prevent serious runtime problems. If a program is aborted because of record size discrepancy, one of the following actions should remedy the situation:

- Change the FD of the offending file in the COBOL program to reflect a record size that is the same as (or greater than) the size specified during CREATK. Recompile the program.

- Use the CREATK utility to change the data record size to agree with the COBOL program (or, if the file is just being created, make sure the correct data size is given).

COBOL programs that have record sizes equal to or greater than the MIDAS data record size are not affected. Furthermore, if a MIDAS file was created with variable record sizes (DATA SIZE = 0), the COBOL library does not test for record size compatibility. The only case that causes a runtime abort of the COBOL program is if the MIDAS data record size is greater than the COBOL data record size.

Library Enhancement

The COBOL library has been changed to increase the speed of secondary duplicate key processing in MIDASPLUS. Increased speed is the only visible result of the changes. MIDAS users are not affected.

DOCUMENTATION CORRECTIONS

The following corrections apply to the COBOL Reference Guide (FDR3056).

VALUE Clause

On page 7-9, Format Three should read as follows:

```
88 condition-name; VALUE IS literal-1 [ { THROUGH } literal-2]
                                       { THRU    }

                  [, literal-3 [ { THROUGH } literal-4]]... .
                                 { THRU    }
```

Format Four should be:

```
88 condition-name; VALUE IS literal-1 [, literal-2]... .
```

On page 7-27, add Syntax Rules 7 and 8:

7. A maximum of 35 values per each 88 level is allowed. The total number of characters in the 88-level values cannot exceed 1024 words. An odd number of characters, n, requires n+1/2 words.

8. In value lists, a comma must be followed by a space.

TALLYING Clause

On page 8-20, under General Rule 3, the first bulleted item should read as follows:

- The user may initialize data-name-2 prior to the operation, but this is not required. Data-name-2 is not initialized by the INSPECT statement.

SOFTWARE PROBLEMS FIXED

Fixed at Rev. 18.4 and Rev. 19.1

Error Messages: Two new error messages have been added to the COBOL compiler:

- The compiler no longer aborts with INTERNAL ERRORs when more than 35 values are used on a level-88 data name. Instead, the error message "NUMBER OF VALUES IN LEVEL 88 EXCEEDS 35" is generated (POLERs #29465, 37613).

- Since no conversion is done when an elementary numeric item is moved to a group level item, the warning message "MOVE DONE WITHOUT CONVERSION" is now issued. (For example, this warning is issued if a PIC 9(5) COMP-3 field is moved into a group level item that is made up of numeric DISPLAY fields.)

PROCEDURE DIVISION: Compilation no longer aborts if a legal paragraph name does not immediately follow the PROCEDURE DIVISION header (POLERs #14651, 25058, 25773, 29983, 33284, 35961, 36466, 40110, 40869, 41604, 46852, 82062, 25768).

Minor errors preceding the PROCEDURE DIVISION header no longer cause the compiler to abort (POLERs #21255, 40070).

Condition Names: Subscripted numeric level-88 condition names were not producing correct results. For example, IF TEST(I)... would not always evaluate properly if the data type was numeric. This problem has been corrected for decimal tests (POLERs #20475, 32159, 41857).

BLANK WHEN ZERO: The BLANK WHEN ZERO clause no longer forces the suppression of leading zeros (POLER #21793).

PIC: Moving a PIC S9(5) field to a group item defined as PIC X or PIC 9(5) no longer causes a character to be lost (POLER #22077).

If a blank was the last character in a signed PICTURE item, the NUMERIC class test would evaluate to TRUE. This problem has been corrected (POLER #30265).

Paragraph Names: The compiler no longer aborts if a paragraph has the same name as a variable in the program (POLERs #27069, 44216).

The compiler no longer aborts if a paragraph name is a reserved COBOL word (POLER #40544).

INTERNAL ERROR Messages: Setting an indexed data name to an index (or vice versa) no longer generates an INTERNAL ERROR 116 or 106 (POLERs #28617, 27362, 36548).

An illegal move to a subscripted identifier no longer causes INTERNAL ERROR 117 (POLERs #29336, 32701).

Paragraphs containing 330 statements no longer generate INTERNAL ERROR 106 (POLER #31848).

Displaying a field with more than 131 characters no longer causes INTERNAL ERROR 116 (POLERs #34804, 30147, 27685).

INTERNAL ERROR 116 is no longer generated by a COMPUTE statement with a subscripted result and an ON SIZE ERROR clause (POLERs #36021, 82846, 40719).

A MOVE CORRESPONDING with more than 11 subscripted correspondences no longer causes an INTERNAL ERROR 117 (POLER #41909).

The compiler no longer generates INTERNAL ERROR 101 if no paragraph name is found after the PROCEDURE DIVISION header and the program uses one file (POLER #48079).

I-O Statements Within IF Statements: Prime's current COBOL compiler does not support conditional I-O clauses on I-O statements that are contained in IF statements. This means that an I-O statement inside an IF statement may not contain an INVALID KEY or an AT END clause. Any I-O statement having one of these clauses inside an IF statement is now flagged as an error (POLERS #28812, 40604, 54757, 27390). For example, the statement:

```
IF A = B READ FILE-NAME AT END GO TO END-PARA.
```

is now flagged as an error. However, the statement:

```
IF A = B READ FILE-NAME.
```

is not flagged. It is suggested that a PERFORM statement be used to accomplish this kind of logic; for example,

```
IF A = B PERFORM MY-READ-PARA.
```

File Description: Placing an 01 where FD should appear no longer causes serious compilation problems (POLERs #28815, 36282).

MIDAS: When a MIDAS file was read sequentially by primary key and then a DELETE was issued, subsequent READ statements would read by primary key. This problem has been corrected (POLER #29211).

A file-status of 22 was returned when the user performed a re-write of a MIDAS file whose primary key took up more than half the data record. This problem has been corrected (POLERs #28774, 40202, 42822, 45819).

COMPUTE: The compiler can now handle COMPUTE statements with more than 12 subscripted variables (POLER #29474).

Source Line Counter: The compiler's internal source line counter now works correctly for up to 9999 lines (POLER #29488).

MOVE: A MOVE statement between two ACCEPT...FROM DATE statements no longer generates incorrect object code and incorrect run-time results (POLERs #30266, 35441).

Numeric Arrays: Reading a file into a numeric array now works correctly (POLERs #32157, 48146).

READ Syntax: The compiler now flags a "READ <filename> NEXT RECORD KEY IS <data name>" statement as an error, since the NEXT RECORD and KEY IS clauses conflict in this statement (POLER #32354).

STRING: The STRING statement no longer automatically takes the ON OVERFLOW clause (POLERs #32618, 35212, 40353).

RELATIVE Files: The compiler no longer generates a fatal error when a RELATIVE file is assigned but never opened (POLER #32678).

File Assignments: Repeating file assignments at run-time no longer causes two files to be assigned to the same file-id (POLER #36368).

ADD and SUBTRACT: All ADD statements following a bad SUBTRACT statement were being flagged as errors until a good SUBTRACT statement was found. This problem has been corrected (POLER #36658).

COPY: A COPY statement without a period is now flagged as a syntax error (POLER #82847).

Notes

Users of COBOL should note that the "DEPENDING ON <data name>" clause is not supported in the DATA DIVISION. If used there, this clause may cause unpredictable results at compile-time.

The COBOL compiler no longer allows nonnumeric figurative constants to be moved to numeric target fields, as in the statement "MOVE SPACES TO PIC-9-FIELD". Such statements are now flagged as errors by the compiler. This change has been made to conform to ANSI standards.

First Fixed at Rev. 19.1

SORT Verb: Sort work files were being left open when the SORT verb was used with the combination "USING filename ... OUTPUT PROCEDURE IS ..." This problem has been corrected (POLER #14161).

Program control now correctly returns from an output procedure when the SORT verb is used with the combination "USING filename ... OUTPUT PROCEDURE ..." (POLER #33489).

VALUE ALL Clause: The VALUE ALL clause was not initializing storage properly. For example,

```
01 DATA-NAME-1 VALUE ALL 'Z'
```

would cause DATA-NAME-1 to be initialized to Z Z Z Z Z instead of ZZZZZZZZZZ. This problem has been corrected (POLER #27025).

DBG (Source Level Debugger)

SOFTWARE PROBLEMS FIXED

Fixed at Rev. 18.4 and Rev. 19.1

New SOURCE Subcommand: A new DBG SOURCE subcommand has been added to reset the default source filename for a given program block. The syntax of this command is:

SOURCE RENAME filename [-BLOCK program-block-name]

(abbreviation: SRC RN)

This command replaces the default (compiler-supplied) source filename of the indicated program block with filename. If program-block-name is not specified, the program block corresponding to the evaluation environment is assumed. If the indicated program block is the same as the current block, the current source file is changed to filename (POLER #26947).

Value Tracing: Value tracing performance has been improved by a factor of 2 to 3, depending on program size (POLER #81498).

Access Violation: Initialization of more than one external module in a single file no longer causes an access violation. Use of the FULL_INIT option is no longer required in such cases (POLER #25450).

A problem that was causing occasional access violations during evaluation of arrays has been corrected.

External Variables: Variables declared outside the scope of a procedure are now visible to DBG (POLERs #25450, 35852).

PL/I Replaced Symbols: Replaced symbols in PL/I, Subset G now apply to the entire file rather than only the first procedure (POLERs #25450, 35852).

Negative Constants: It is now possible to evaluate or assign large negative constants (POLER #28711).

File Units: File units were not being handled correctly by DBG on systems configured for fewer than 62 file units. This problem has been corrected (POLER #34352).

RESUBMIT Command: The RESUBMIT command now works correctly with separators inside action lists (POLER #34956).

Following a command error, the previous command line is no longer truncated, and is still available for RESUBMIT.

FORTRAN: Substring references are now permitted on the left-hand sides of FORTRAN assignment statements (POLER #45443).

FORTRAN 32-bit octal constants are now evaluated correctly.

CALL Command: The CALL command now correctly passes alternate return labels to FORTRAN modules.

Strings: A problem that was causing a stringrange error on some literals has been corrected.

Alternate Entry Blocks: Full initialization of alternate entry blocks is now done correctly.

Special Characters: Special characters are now handled correctly in action lists.

Error Message: An appropriate error message is now supplied for a command line that is too long.

Fixed at Rev. 19.0 and Rev. 19.1

Array Evaluation: A problem that was causing occasional access violations during evaluation of arrays has been corrected.

Parameters With Bit Offsets: Parameters with bit offsets are now evaluated correctly in : commands, IF commands, etc. (POLER #45236).

First Fixed at Rev. 19.1

The ABS Intrinsic: The ABS intrinsic now accepts integer*4 arguments (POLER #34535).

Exponentiation in PL/I, Subset G: The exponential operator in PL/I, Subset G now works.

Note

According to the rules of PL/I, Subset G, -a**b is equivalent to -(a**b).

Pascal: Members of Pascal pointer-addressed records are now evaluated correctly (POLER #40061).

Floating Point Numbers: DBG can now display floating point values with four-digit exponents (POLER #40066).

WATCH Function: Attempting to WATCH an array element with a subscript that is out of range now produces a warning rather than an error (POLER #43651).

FORTRAN Arrays: FORTRAN variable extent arrays are now evaluated correctly (POLERs #43822, 45123).

Pascal IN Operator: The PASCAL IN operator now checks for a scalar value that is not within the range of the set.

Value Tracing: Value tracing now detects changes in values that are members of a based structure.

Command Line Overflow: Command line overflow is now detected when an action list is input.

Pascal Pointer Functions: Pascal functions of type pointer can now be evaluated.

STATUS Command: The STATUS command now displays the type of value tracing selected (entry_exit or full).

SOFTWARE PROBLEM OUTSTANDING

FORTRAN Exponents

In FORTRAN evaluation mode, two-digit exponents containing a digit larger than 7 are not accepted by the evaluator (POLER #32260).

FORTRAN (FTN)

NEW FEATURE

At Rev. 19.1, if FTN finds an error (other than a warning) in a source file, the binary (object) file that the compiler has produced is deleted. Deletion occurs only if FTN itself has produced the binary file. If the binary file was connected before FTN was invoked, FTN writes a BAD OBJECT FILE flag into the binary file as a signal to Prime's loaders, so that an erroneous object file will not be loaded into a program. The binary file is then left open.

Two compiler options have been added to control this new feature: -DELBIN and -NODELBIN. -DELBIN tells FTN to delete the binary file if errors are encountered. -NODELBIN causes FTN to behave as it did prior to Rev. 19.1 — that is, binary files are not deleted. -DELBIN is now the default.

SOFTWARE PROBLEMS FIXED

Fixed at Rev. 18.4 and Rev. 19.1

Error Messages: The "Parameter is better" warning message was occasionally inaccurate. This problem has been corrected (POLER #12484).

A program size overflow message has been clarified.

ENCODE: The modification of an array by an ENCODE statement is now reflected in the cross-reference listing and in the "Never given a value" warning of debug mode (POLER #12490).

Format Specifiers: The compiler now accepts a variable as the format specifier in an I/O statement (POLER #21112).

Continuation Lines: The compiler now uses two source lines, if necessary, to determine the type of a FTN statement. This enhancement corrects some problems that were caused by continuation lines (POLERs #27520, 82273, 36472, 32652, 29567, 45138, 37287, 81485).

Floating Point: A problem with floating point comparison in 64R mode has been corrected (POLER #33631).

DMIN1 Function: The DMIN1 function in 64V mode no longer rejects argument lists with more than four arguments (POLER #34908).

Cross-reference Listings: The line numbers where equivalenced variables were specified are no longer included in cross-reference listings (POLER #21197).

External names with $ as the second character are no longer omitted from the cross-reference listings (POLER #37636).

The cross-reference listings for certain intrinsic functions have been corrected (POLER #82611).

Rounding: A problem with the rounding of real and double-precision literals has been corrected, improving arithmetic precision.

SOFTWARE PROBLEMS OUTSTANDING

FTN vs. F77

A program that performs a great deal of single-precision arithmetic may produce different results, depending upon whether it is compiled by FTN or by F77 (POLER #28703).

XREFS Option

When compiling some large programs with the XREFS option, FTN produces a "program overflow" error message and eventually halts abnormally (POLER #31241).

XREFL Option

When a program is compiled with the XREFL and UNCOPT options, the XREFL listing incorrectly places some statement labels at the end of a do loop instead of within the loop (POLER #45192).

Arithmetic

A particular arithmetic expression with mixed integer and real operands evaluates incorrectly (POLER #43819).

Subtractions of double precision variables that have exponents greater than 17 are sometimes evaluated inexactly (POLER #43820).

FORTRAN 77 (F77)

NEW FEATURES

Prime Standards

F77's initial banner has been changed to be more consistent with most of Prime's other major compilers (for example, Pascal and PL/I, Subset G).

External Names

External names of up to 32 characters in length are now supported by F77. Warning 214 no longer appears. Warning 29 is now given for external names that are longer than 32 characters.

At Rev. 19.1, SEG does not support external names that are longer than 8 characters; it truncates such names. Therefore, it is recommended that F77 external names be unique within the first eight characters.

New Compiler Options

The STORE_OWNER_FIELD (SOF) and NO_STORE_OWNER_FIELD (NSOF) options have been added to the F77 compiler at Rev. 19.1. SOF enables generation of module names into the executable code following each module's ECB. This option is especially useful in debugging F77 programs, since utilities such as DMSTK will have access to module names. However, use of this option increases the size of the generated code and linkage and also slightly degrades execution speed.

NSOF is the default option. Thus, compatibility with earlier F77 compilers is preserved. This option suppresses the EAL and JSXB instructions at the beginning of every F77 routine. It also suppresses the generation of module names into the executable code. With this option, users should notice no difference in the execution speed of their programs.

SOFTWARE PROBLEMS FIXED

Fixed at Rev. 18.4 and Rev. 19.1

Runtime Library Changes: Changes in the runtime library have increased the following three limits:

- The number of arguments that can be passed to and from subroutines (now up to 254 in most cases).
- The number of entries that may be present in a NAMELIST block (now up to 247 in most cases).
- The number of separate target labels permitted in a computed GOTO statement (now up to 254).

There are exceptions to the first two limits indicated above. The exceptions occur particularly when CHARACTER data is involved. For example, if all arguments to a subroutine were of type CHARACTER, only 127 arguments would be allowed. However, if arguments were of mixed data types (CHARACTER and other types), the maximum would be between 127 and 254, and would depend upon the positions occupied by the CHARACTER data in the argument list. Similar restrictions hold for NAMELIST blocks (POLERs #25369, 33487).

Data Type Conversion: The INT, INTS, and CHAR intrinsic functions now function correctly (POLERs 27719, 31868).

Output Format: The unimplemented output formats Z (hexadecimal) and O (octal) now result in the compiler error message "Unexpected character in format" (POLER #36544).

The non-ANSI FORTRAN omission of a delimiter (such as a comma) after an X format descriptor now results in compiler error 449 (POLER #36500).

Arrays: The source level debugger (DBG) now correctly evaluates the type of an assumed-size array. Previously, an access violation occurred when the assumed-size array type was requested. Note, however, that information on the correct dimensions of such arrays is unavailable to DBG. (POLER #29734).

A new error message (#447) prohibits the use of assumed-size arrays in I/O lists (POLER #35846).

Adjustable-dimensioned arrays are now passed correctly to subroutines (POLER #40064).

Register Values: All levels of optimization now preserve register values. For example, the contents of the X register are preserved across all calls to LOG, EXP, SIN, etc. (POLERs #32927, 36765, 41471, 35769).

First Fixed at Rev. 19.1

Intrinsic Functions: Negative constant arguments to intrinsic functions were evaluated incorrectly at Rev. 18.4. This problem has been corrected.

SOFTWARE PROBLEMS OUTSTANDING

Exponents

The compiler cannot yet handle double-precision constants where the exponent has more than 2 digits, as in 1.0D200 (POLERs #29335, 32927).

Floating Point Precision

Floating point constants evaluated within F77 should be more precise (POLER #36539).

Pascal

NEW FEATURES

The new features discussed in this section are also described in the new edition of the Pascal Reference Guide (DOC4303-191). For more information on the new Pascal book, refer to the section called NEW TITLES in Chapter 1 of this book.

LONGINTEGER Data Type

Rev. 19.1 introduces a new Pascal data type called LONGINTEGER. LONGINTEGER allows you to use 32-bit integers without declaring a subrange. For example:

```
VAR
  I : LONGINTEGER;
```

This declaration means that the variable I can have a value anywhere within the subrange -2147483648..+2147483647.

On Prime machines, an integer is a 16-bit (one-word) number within the subrange -32768..+32767.

The arithmetic and relational operators and standard Pascal functions can be used with LONGINTEGER as well as with INTEGER. LONGINTEGER can also be passed as a parameter in procedures and functions. However, longintegers cannot be used as array indexes.

It is recommended that you do not mix INTEGER types with LONGINTEGER types. You can assign an integer to a longinteger, but when you try to assign a longinteger to an integer, a warning message will be given at compile time.

LONGINTEGER constants are allowed. The compiler decides whether the constant is an integer or a longinteger. (These rules also apply to REAL and LONGREAL. The LONGREAL data type is discussed below.)

Consider the following program example:

```
PROGRAM Longint;
CONST
  X = 55000;
VAR
  A : INTEGER;
  B : LONGINTEGER;
  C : INTEGER;
```

```
BEGIN
  A := B;
  C := X;
  C := A + B
END.
```

Each executable statement in the example above assigns a longinteger (B or X) to an integer (A or C). Each statement, therefore, would receive the following Severity 1 warning:

```
ERROR 121 SEVERITY 1 BEGINNING ON LINE zzz
A type conversion must be made in this statement and may cause the
program to fail if the conversion is not possible.
```

LONGREAL Data Type

Rev. 19.1 also introduces the new Pascal data type LONGREAL. Longreals are 64-bit real numbers, as opposed to reals, which are 32-bit numbers. Variables are simply declared:

```
VAR
  X : LONGREAL;
```

The arithmetic and relational operators and standard Pascal functions can be used with LONGREAL as well as with REAL. LONGREAL can be passed as a parameter in procedures and functions.

It is recommended that you do not mix REAL and LONGREAL types for the same reasons given in the LONGINTEGER discussion earlier in this section.

Reading Arrays

At Rev. 19.1, the ARRAY OF CHAR data type has been enhanced to make the reading of character arrays much easier. On Rev. 19.1 systems, you can read an array of characters as one unit, instead of reading one character at a time. For example, Prime's ARRAY OF CHAR function allows you to declare and read character arrays like this:

```
PROGRAM Primearray;
VAR
  A : ARRAY[1..10] OF CHAR;
  B : ARRAY[1..60] OF CHAR;
BEGIN
  READLN(A);
  READLN(B)
END.
```

Previously, reading was done one character at a time within a loop:

```
BEGIN
  FOR I := 1 TO 10 DO
    READ(A[I]);
  FOR I := 1 TO 60 DO
    READ(B[I])
  END.
```

You can still use loops to read an ARRAY OF CHAR. However, it is easier and more efficient to use the Prime extension.

The length of a Prime character array can be two to 256 characters.

Consider the following:

```
VAR
  A : ARRAY[1..30] OF CHAR;
BEGIN
  READLN(A);
```

With READLN, if less than 30 characters are typed in, the remaining characters will be blank-filled. If more than 30 characters are typed in, only the first 30 characters will be assigned. You will not be warned that you have typed in extra characters.

If you are using a READ, all 30 characters must be typed in. The remainder of the array will not be padded with blanks, according to the standard Pascal definition of READ.

Multidimensional arrays can be read and written more easily with Prime's ARRAY OF CHAR:

```
VAR
  A : ARRAY[1..10, 1..200] OF CHAR;
BEGIN
  READLN(A[1]);
```

This will read the first row of 200 characters into A[1]. A READ(A) statement would generate an error because you would be trying to read an array of strings, and not a single string.

Using PRIMOS Erase and Kill Characters

Rev. 19.1 Pascal incorporates a new switch, -INTERACTIVE, which allows you to use the PRIMOS erase and kill characters during input at the terminal.

The PRIMOS erase character, the double question mark (??), erases the immediately preceding character. The kill character, the single question mark (?), deletes the entire current line. Prior to Rev. 19.1, the erase and kill functions of the question marks could not be used in Pascal input at the terminal. At Rev. 19.1, you can activate the erase and kill characters by setting the -INTERACTIVE switch in the RESET statement. For example:

```
VAR
  I,J: INTEGER;
BEGIN
  RESET (INPUT, '-INTERACTIVE');
  READLN(I);
  READLN(J)
END.
```

The word -INTERACTIVE must be enclosed in single quotes.

> Caution
>
> You can only use READLNs with the -INTERACTIVE switch. Do not use READs. An attempt to use READs in a program with -INTERACTIVE generates an error message.

You can turn the -INTERACTIVE switch on or off within a program at your discretion. To turn the switch off, use the -TTY feature in another RESET statement. For example:

```
VAR
  A,B,C,D: INTEGER
BEGIN
  RESET (INPUT, '-INTERACTIVE');
  READLN(A);
  READLN(B);
  RESET(INPUT, '-TTY');
  READ(C);
  READ(D);
END.
```

The RESET statement with -TTY deactivates the erase and kill characters for subsequent terminal input.

> Note
>
> Use of the PRIMOS erase and kill characters, as well as READs and READLNs, is always allowed in input that is typed into PRIMOS input files.

Optimization

At Rev. 19.1, two new levels of optimization, OPT1 and OPT3, provide a wider spectrum of optimization than was previously available in Prime Pascal.

The default optimization option, -OPTIMIZE, causes the object code to be optimized. Optimized code runs more efficiently than nonoptimized code, but takes somewhat longer to compile.

The -OPT1 option optimizes less code and is less efficient than -OPTIMIZE, but compilation time is faster than -OPTIMIZE.

The -OPT3 option optimizes more code and is more efficient than -OPTIMIZE, but compilation time is slower than -OPTIMIZE.

When -NOOPTIMIZE is invoked, optimization does not occur.

Passing Procedures and Functions as Parameters

In Prime Pascal, as in standard Pascal, you can declare and pass a procedure or function as a parameter. Any procedure or function can pass any other procedure or function.

Declaring Procedures and Functions as Parameters: A procedure or function declaration must list another procedure or function as a formal parameter. For example:

```
PROCEDURE A (PROCEDURE X);
```

If the procedure or function that is being passed has parameters of its own, the number and types of parameters must also be listed. For example:

```
PROCEDURE A (PROCEDURE X(X1, X2 :  INTEGER));
```

If you are passing a function, the type that the function returns must also be given:

```
PROCEDURE A (FUNCTION Y(Y1, Y2 : INTEGER) : INTEGER;)
```

A procedure or function parameter can even have other procedures and/or functions as parameters:

```
PROCEDURE A (FUNCTION Y(PROCEDURE Y1;
                        FUNCTION Y2 : CHAR) : INTEGER);
```

The procedure or function that is declared must match, parameter for parameter, in number and type, the declaration for the procedure or function that is passed.

Passing Procedures and Functions as Parameters: The name of a procedure or function is passed the same way any variable is passed. For example:

```
ADD(SORT);
```

The procedure called SORT is passed to a procedure called ADD. The name SORT is passed without parameters, but the number and type of parameters declared in the SORT procedure must match those in the ADD declaration, parameter for parameter.

Here are some examples:

Example 1:

```
VAR
  I : INTEGER;
PROCEDURE ADD1;
  BEGIN {procedure ADD1}
    I := I + 1;
  END;
PROCEDURE CALLPROC(PROCEDURE X);
  BEGIN {procedure CALLPROC}
    X;
  END
BEGIN {main program}
  I := 0;
  CALLPROC(ADD1);
END. {I = 1}
```

Example 2:

```
VAR
  I : INTEGER;
FUNCTION ADD10 : INTEGER;
  BEGIN {function ADD10}
    ADD10 := 10;
  END;
FUNCTION CALLF(FUNCTION X : INTEGER) : INTEGER;
  BEGIN {function CALLF}
    CALLF := X + X;
  END;
BEGIN {main program}
  I := CALLF(ADD10);
END. {I = 20}
```

Example 3:

```
VAR
  I, J : INTEGER;
FUNCTION SQUARE(X : INTEGER) : INTEGER;
  BEGIN {function SQUARE}
    SQUARE := X * X;
  END;
FUNCTION FCALLFUNC(FUNCTION X(R : INTEGER) : INTEGER) : INTEGER;
  BEGIN {function FCALLFUNC}
    FCALLFUNC := X(5);
  END;
PROCEDURE PCALLFUNC(FUNCTION Y(Q : INTEGER) : INTEGER);
  BEGIN {procedure PCALLFUNC}
    I := Y(5);
  END;
BEGIN [main program}
  J := FCALLFUNC(SQUARE);
  PCALLFUNC(SQUARE);
END. {I = 25 and J = 25}
```

Example 4:

```
VAR
  I, J, K : INTEGER;
PROCEDURE ADD(X : INTEGER);
  BEGIN {procedure ADD}
    I := I + X;
  END;
FUNCTION FCALLPROC(PROCEDURE X(R : INTEGER)) : INTEGER;
  VAR
    R : REAL;
  BEGIN {function FCALLPROC}
    R := 2.19;
    X(ROUND(R)); {result of function call is passed}
    FCALLPROC := 10;
  END;
PROCEDURE PCALLPROC(PROCEDURE Y(Q : INTEGER));
  BEGIN {procedure PCALLPROC}
    Y(8);
    J := 10;
  END;
BEGIN {main program}
  I := 0;
  K := FCALLPROC(ADD);
  PCALLPROC(ADD);
END. {I, J, and K each = 10}
```

Example 5:

```
VAR
  I, J : INTEGER;
PROCEDURE ADD2(PROCEDURE A1);
  BEGIN {procedure ADD2}
    A1;
    A1;
  END;
PROCEDURE ADD1;
  BEGIN {procedure ADD1}
    I := I + 1;
  END;
PROCEDURE CALLPROC(PROCEDURE X(PROCEDURE Y); PROCEDURE Z);
  BEGIN {procedure CALLPROC}
    Z;
    X(Z);
  END;
BEGIN {main program}
  I := 0;
  CALLPROC(ADD2, ADD1);
END. {I = 3}
```

External Identifiers

External identifiers may now be a maximum of 32 characters in length. Previous releases of Pascal limited external names to 8 characters.

New Error and Informational Messages

Error Severity: A line of information now appears after the ERRORS line in the compiler listing. This new line indicates the maximum severity of the errors found in the program.

Warning Messages: New messages have been added to warn the user when:

- Labels, constants, types, variables, and procedures/functions are declared in nonstandard order. (Arbitrary order is legal in Prime Pascal.)
- Label, constant, type, or var statements are used more than once in a block.

Set Operators: The compiler now issues an error message when the set operator IN is used on a non-SET type.

Miscellaneous Improvements

= Corrections: Pascal now corrects a = to be a := when appropriate. A Severity 2 error message is generated instead of a Severity 3 message.

Identifier Lengths: If an identifier is more than the allowed maximum of 32 characters long, a warning message is printed and the identifier is truncated.

Command Line Options: All command line options are now printed to the listing file.

%INCLUDE Files: Errors in %INCLUDE files no longer cause Pascal to leave the %INCLUDE files open.

Better Error Messages: Each Pascal error message now includes the error number, the severity, and the line where the error occurred. In addition, if the error occurred in an %INCLUDE file, the name of the file is reported.

SOFTWARE PROBLEMS FIXED

Fixed at Rev. 18.4 and Rev. 19.1

DBG: The source level debugger (DBG) now recognizes variables defined in external procedures as external references (POLER #35852).

DBG also now recognizes the standard identifiers INPUT and OUTPUT (POLER #35847).

Compiler Listing: In the listing file that is generated at compile time, the attribute for P$AINP is now INPUT EXTERNAL, and the attribute for P$AOUT is now OUTPUT EXTERNAL. (Previously, FILE EXTERNAL was printed for both.)

Optimization: Compiling a program with the DEBUG option now automatically turns off optimization (POLER #35851).

A problem that was causing the compiler to abort during optimization of a valid program has been corrected (POLER #35858).

Fixed at Rev. 19.0 and Rev. 19.1

Read of a Record: A read of a record no longer causes a compiler error.

First Fixed at Rev. 19.1

Writing Real Numbers: An attempt to write a real number with a specified field-width of 13 and no fractional digit specification resulted in a null field being written. This problem has been corrected (POLER #41849).

An attempt to use WRITELN to write a real number that had no fractional digits and whose total width was an expression evaluating to 13 resulted in a null field being written. This problem has been corrected (POLER #43889).

When printing real numbers in a specified field width, Pascal was truncating instead of rounding. This has been corrected to conform with the latest standard.

Pascal now detects an attempt to write a real number with a specified field width of less than 1 or a specified number of decimal digits of less than 1. These conditions now cause runtime errors.

Reading From the Terminal: Errors resulted when an attempt was made to read more than 256 characters from the terminal using a READ statement in a loop without a READLN. This problem has been corrected.

Writing Strings: When the user attempted to write a string, and specified a field width shorter than the string itself, Pascal ignored the specified field width and wrote the entire string. According to the new Pascal standard, only the specified number of characters should be printed. The Pascal compiler has been changed to conform to this standard.

The -NOERRTTY Option: The -NOERRTTY option now works correctly.

Negatives of Negatives: The negative of a negative constant is now evaluated properly in a constant declaration.

PL/I, Subset G

NEW FEATURE

Floating Point Numbers

Prime's PL/I, Subset G compiler can now reference floating point values with exponents between 1 and 4 digits long.

DOCUMENTATION ADDITIONS AND CORRECTIONS

The following corrections apply to the PL/I, Subset G Reference Guide (IDR4031).

PL/I and Other Languages

On page 1-6, the following item should be added to the bulleted list in the section called INTERFACE TO OTHER LANGUAGES:

- Variable (*) extents cannot be used with other languages, as they are not defined there. For example, if a Pascal program calls a PL/I (Subset G) procedure, passing it an array defined as:

```
var f:array[1...5] of char;
```

  then the PL/I routine must declare:

```
declare f(5) char(1);
```

  rather than

```
declare f(*) char(1);
```

This information should also be added to page 4-6, where * extents are discussed.

Testing for End of File

The following information should be added to page 9-25, before the section on ENDPAGE condition (POLER #47447).

When a loop that reads records from a file is to be executed several times, the

```
ON ENDFILE ... GOTO
```

statement should not be used to break out of the loop. The GOTO prevents internal stack space from being released. Instead, set a bit on endfile using a statement such as

```
ON ENDFILE ... EOF_BIT='1'B;
```

Then test the bit inside the read loop, and use that test to leave the loop.

SOFTWARE PROBLEMS FIXED

Fixed at Rev. 18.4 and Rev. 19.1

New Command Line Options: A new pair of command line options is now available for PL/I, Subset G. The -OVERFLOW option allows for detection of zero-divide and fixed binary overflow conditions. The -NO_OVERFLOW option (which is the default) suppresses checking for these conditions. It should be noted that specifying the -OVERFLOW option significantly increases code generation time for programs that use arrays. When -OVERFLOW is specified, the built-in function DIVIDE now returns an error message if a program attempts to divide a binary value by zero. When -OVERFLOW is specified, fixed binary values greater than 32767 now create an overflow and generate an approriate error message (POLERs #24308, 30110, 29320, 32674, 24308, 40320).

First Fixed at Rev. 19.1

Error Messages: The compiler generated error 32 if BIT was used but no error if BIT(1) was used. This problem has been corrected (POLER #20845).

The compiler now signals as an error an attempt to use duplicate primary keys in MIDAS (POLER #28091).

The compiler now signals as an error an attempt to define an array with a subscript range exceeding the legal maximum (POLER #29319).

DIMENSION Function: The DIMENSION function now works with a based and allocated character array (POLER #27473).

Error Recovery: The compiler now recovers properly from an error 82 (POLER #32707).

DBG: Inconsistencies between values from runtime and values from the debugger have been corrected (POLER #37315).

DO...TO...BY...Sequence: The DO...TO...BY... sequence did not work correctly with expressions. This problem has been corrected (POLER #40983).

Procedure Calls: A call to a procedure with a constant raised a fixed overflow error. This problem has been corrected (POLER #40995).

DAM Files: A problem concerning the use of large keys in accessing a DAM file has been corrected (POLER #43827).

Multiple Modules: A debug-mode compilation of more than one module no longer fails.

RPG

SOFTWARE PROBLEMS FIXED

Fixed at Rev. 18.4 and Rev. 19.1

Spacing 0: Spacing 0 before and after on output now both work correctly (POLER #33310).

Fetch Overflow: Fetch overflow is now properly coordinated with page eject (POLER #33311).

DIVIDE Operation: DIVIDE now works correctly with a subscripted array as result field (POLER #33481).

Matching Fields: Sequence checking now works with packed decimal matching fields (POLER #35157).

VFINLIB

SOFTWARE PROBLEMS FIXED

Fixed at Rev. 18.4 and Rev. 19.1

Error Codes: System error codes are now appropriate for the types of errors generated, assuming that VFINLIB is built using PRIMOS Rev. 18.4, Rev. 19.1, or later.

Fixed at Rev. 19.0 and Rev. 19.1

F$IOFIN: F$IOFIN now accepts B-format statements, which may be surrounded with blanks (POLER #35387).

P$ATOA: P$ATOA now traps numbers with multiple decimal points and delivers an error message (POLER #30110).

P$ATOA now correctly handles the rounding of floating point numbers (POLER #32365).

First Fixed at Rev. 19.1

F$IO77: F$IO77 B-formats no longer produce unwanted characters at the end (POLERs #23929, 29743, 31162).

F$IO77 now allows lowercase formatters in uncompiled format statements (POLER #33786).

F$IO77 error returns now operate only on statements within their jurisdiction (POLER #46976).

F$IOFIN: F$IOFIN now accepts a user-supplied ERR= return statement on a bad format specifier (POLER #40721).

P$ATOA: P$ATOA now correctly handles exponents with more then two digits (POLER #29335).

F$STRANS2: F$STRANS2 now signals error conditions correctly.

Zero Exponents: FORTRAN evaluations of numbers to the zeroth power now return an appropriate form of 1 (POLER #35139).

Error Messages: PRIMOS error messages have been corrected (POLERs #37051, 40608, 48230).

VM$__: VM$__ now attempts a more intelligent magnetic tape recovery (POLER #37913).

F$CLOS: F$CLOS now closes the correct file unit (POLERs #47450, 31191, 48003).

VRPG

SOFTWARE PROBLEMS FIXED

Fixed at Rev 19.0 and Rev. 19.1

Maximum Table and Array Size: The maximum allowable size of compile-time tables or arrays has been expanded from 80 character records to 256 character records (POLER #37191).

BLANK AFTER Works: The BLANK AFTER specification now works properly for tables, arrays, and array elements (POLER #45957).

Packed and Binary Fields: When a decimal position length was greater than the length of the field, a severity 3 error would occur. Now this error only occurs when the decimal position length is greater than the unpacked decimal length of the field (POLER #45942).

Lookup Operation: For the lookup operation that used binary or packed data, the check made to test that the index field length was equal to the table or array length now correctly compares the unpacked decimal length of both factors.

Duplicate Output Lines: OR on the output form no longer causes duplicate output lines (POLERs #45961, 37191).

Comma as Decimal Point: When I or J is placed in column 21 of the header statement, VRPG now correctly allows use of the comma as decimal point (POLER #51802).

SOFTWARE PROBLEMS OUTSTANDING

Passing Arrays

There exists a problem with passing an array as a parameter to an external subroutine (POLER #45959).

Access Violations

An access violation may occur if the user requests continuation after a record with match fields is reported to be out of sequence.

/* Delimiter

The RPG delimiter for end of input is /*. This causes some confusion within a CPL program that considers /* as a comment delimiter. A conflict arises if an RPG program is executed within an &DATA - &END construct that also contains input data for a console file. If /* is one of the data entries, CPL treats it as a comment and does not pass that line of data to the RPG program.

Note

The VRPG driver program, RPGDF.SAVE, which sets the default options of the VRPG compiler, now resides in VRPG>TOOLS. The System Administrator can change the default options by entering the command:

```
RESUME RPGDF SYSOVL>RPGDATA [options]
```

Refer to Chapter 16 of the System Administrator's Guide (DOC5037-190) for more information.

CHAPTER 4

DATA MANAGEMENT SYSTEMS

DBMS

NEW FEATURES

DMLCP

While the size of the run unit table remains at 32,000 words, this total no longer includes the alphabetic names of areas, sets, records, and items, nor the four-words-per-name hash table entry. This change allows a larger subschema to be compiled and accessed through DBMS. Note, however, that the total number of of area, set, and CALC files accessed cannot exceed the system limit of 128 for any one user invocation of the subschema (POLERs #44460, 37380, 37381, 41294).

Several fundamental DMLCP routines have been rewritten in a PL/I-like language to take advantage of inherent efficiencies in that language. This enhancement results in a performance improvement of up to 23 % in some cases.

DBUTL

.DBMS_EDIT ACL Group: In order to update a schema with DBUTL's EDITOR or REWIND command, a user must be a member of a special ACL group called .DBMS_EDIT. The System Administrator should add this ACL group to the user profile of each user who will be authorized to do schema updates with DBUTL. A user who is not so authorized will be reminded upon invocation of one of the Editor commands. He or she will be allowed to look at the files with the Editor, but not to change them in any way.

ED SC: The ED SC command allows editing of the schema table in the same manner that the other ED commands allow editing of area, set, and CALC files. This facility has been available since Rev 18.3, but was never before documented (POLER #48573).

WARNING

Proper use of the ED SC command and other DBUTL ED commands requires a thorough knowledge of the internals of DBMS. Errors and inconsistencies should not occur in DBMS files. Any that do occur should be reported to a Prime field engineer. Incorrect use of the editor commands may introduce additional, perhaps worse, problems in the data base files, and extra costs may be incurred in order to fix these problems.

REW: A new REW command is available for rolling back changes made to a CALC file or schema table file with ED. The format of the command is as follows:

```
REW { R [rec-id] }
    { SC        }
```

The record id is optional if current of type exists. This command prompts for the revision number of file at which to start rewind and for the number of edit sessions to rewind. This command works in a manner similar to that of the REW Area and REW Set commands.

SC schema-name -H: The -H, or History, option displays the user name and number of the last person to modify the schema table using DBUTL commands EDIT or REWIND.

DUMP CALC rec-id -H: If the -H option is specified in this command, rec-id is required. After displaying the value of magic keys, the command now displays the user name and number of the last person to modify the file using DBUTL commands EDIT or REWIND.

Rewind Files: All rewind files are now stored in a central directory, DBMSLB>EDITS, rather than in the directory from which the edit session was run.

Log File: All DBUTL edits and rewinds are logged in the file DBMSLB>EDIT.LOG. This log contains:

- Date and time of edit
- Type (ED or REW)
- File type (set, area, record, schema table)
- Rewind file name
- User number and name

All Products

A mechanism has been added to all products to facilitate identification of the version of DBMS running at a particular site: a task previously made difficult when one or more patches had been applied to a module. All products will continue to greet the user with a revision stamp as before, but the stamp may now include a patch number. For example, Rev. 19.1.2 would indicate the second patch applied to Rev 19.1.

The libraries upon which some or all products are built (e.g., ILIB, CLIB, ULIB, RLIB, ASI, ASG, TEXTED) may contain patches, but do not themselves produce a greeting message. Their revision stamps may be viewed with the aid of SEG and VPSD as follows:

1. Look at a SEG map (created by SEG from the existing seg files or, in the case of DMLCP, already located in DBMSEX>BINARY>LIBRARY.MAP) for entries called "library-nameREV" (for example, ILIBREV, ULIBREV). Note the segment number and address.

2. For DMLCP, invoke VPSD. For all other products, proceed as follows:

```
SEG
# RESTORE DBMSLB>product.SEG
# PSD
```

3. Then issue the following commands:

```
$ :A
$ SN segment-number
$ A address
/
$ Q
# Q   */  (for SEG users)
```

   The revision and patch number are contained in ASCII in the four words beginning at address.

Note

DBMS runtime (DMLCP) requires the use of shared segments 2001, 2002, 2003, and 2012 as well as private segments 4030 through 4034.

DOCUMENTATION CORRECTIONS AND ADDITIONS

Data-base-data-name

On page 2-22 of the DBMS COBOL Reference Guide (PDR3046), Rule 3 should read as follows (POLER #45738):

3. Must Be Unique

   Data-base-data-names must be unique within the subschema per current implementation.

PRIVACY KEY

In the example on page 3-71 of the DBMS FORTRAN Reference Guide (PDR3045), replace the word LOCK with the word MYLOCK wherever it occurs. LOCK is a reserved word (POLER #51660).

Schema Editor

On page 11-5 of the DBMS Administrator's Guide (PDR3276), under SPECIAL CONSIDERATIONS, the portion of text beginning "The Schema Editor allows ..." should read:

The Schema Editor allows the user to change the member subentry as follows (POLER #52747):

From	To
Mandatory	Optional
Optional	Mandatory
Automatic	Manual
Manual	Automatic

SOFTWARE PROBLEMS FIXED

Fixed at Rev. 18.4 and Rev. 19.1

DMLCP: DMLCP now establishes correct set currency to access NEXT/PRIOR in a set subsequent to the deletion of a record occurrence in that set. Thus, a DELETE can be followed by a FIND NEXT or a FIND PRIOR (POLERs #15456, 41436, 45089, 24259, 37971, 34678, 82606, 82630, 44228).

The DMLCP command line -VALIDATE option (formerly the VERIFY option) is now able to find member records sorted by data base key (POLER #48084).

It is no longer necessary to open a file on unit 45 before using the DMLCP -VALIDATE option. If -VALIDATE is specified, DBMS asks for the name of a file to which the debug information will be written. This file is opened on a dynamically allocated unit.

DMLCP now checks that the run-unit table does not exceed a maximum size of 32,000 words (POLERs #37380, 41640, 41639, 40566).

As of Rev. 18.3, retrieval transactions are no longer written to the after-image file. The screening process noted above is still valid, however, in order to allow for after-image recovery using after-image files created prior to Rev. 18.3.

Error message (30F) is now generated when too many arguments appear in the DMLCP call. The limit on the number of arguments has been increased to 64 but requires preprocessor support. Therefore the effective limit is still 20 (POLERs #41442, 41617, 51667).

An error code of 26 (NO RECORD SATISFIES FIND) is now returned when a FIND is performed on an empty set occurrence whose owner was stored using a subschema not including the target set (POLERs #40529, 40537).

Within multimember sets, successive calls to FIND NEXT or FIND PRIOR specifying different record types now either find the record or result in the appropriate nonfatal error (POLER #37463).

Mapping of bit strings to INTEGER*2 now moves all bits (POLER #34469).

DMLCP now correctly marks the status of deleted record partitions (POLER #47650).

As a schema is being closed, a check is now made to see that the user control block has not already been deleted.

An #EXIT DBMS following an Internal Fatal Error will now validate that the user control blocks and their links are correct.

FTRACE dumps performed on the same day are now appended to the PDBMS>FTRACE file.

The minimum amount of space required for a bucket to remain on the available space chain has been increased. For more information, refer to the section on DBACP under First Fixed at Rev. 19.1, below.

A FIND...USING ITEM1, ITEM2, ... ITEMN now retrieves records containing the entire concatenated key (POLER #45976).

A duplicate CALC key consisting of concatenated character strings will now hash to the correct location (POLERs #45971, 55533, 55594).

Subordinate items in a repeating or naming group now have all privacy locks of any group names above them. They may additionally have privacy locks of their own.

Before attempting to INSERT a record occurrence into a set, DMLCP now checks all entries in a set node, whether or not all record types are included in the subschema. Thus, an INSERT into an ORDER IS LAST set positions the record at the very end of a set, instead of at the end of those records included in the subschema (POLER #44385).

Before deleting a record occurrence from the data base, DMLCP now checks all record occurrences that the record owns, whether or not all record types are included in the subschema. An attempt to delete a record that owns records not included in the subschema generates an 0208 error: "Referenced record or set not in subschema" (POLER #47114).

DBUTL: DBUTL can now operate on a schema table of less than or equal to 32,767 words in size (POLER #41952).

DBUTL may now open up to 128 files.

When the VERIFY command encounters an inconsistency in a set file, it now displays the error and continues processing. It also validates the owner data base key (DBK) in the owner directory entries of a set file. A bad owner DBK is indicated by a -1 in the entry-number of the display.

A record dump now checks the status of record fragments and ignores those that have been deleted.

A record dump now correctly outputs the record's offset in the bucket as Rec FWA (POLER #44651).

DBACP: DBACP EXPAND and PACK now reflect the change (from 8 to 100) in the number of words required to remain on the available space chain (POLERs #47652, 40183, 40182).

DBACP now rechains a bucket in which some records have been deleted but none of whose remaining partitioned records can be packed.

DBACP now checks the validity of directory space pointers and record space pointers in VERIFY, EXPAND, and PACK AREA. All bucket pointers are checked for internal consistency.

A schema restore now updates all schema table information correctly when one or more subschemas reside on one volume and other data base files reside on another volume. A multivolume restore proceeds as follows:

> After restoring the schema table but before restoring any other data base files, DBACP scans the names of all volumes used by the saved schema and checks that all are available. For each volume that is not available, DBACP asks for a substitute. Only when all substitute volumes have been supplied and verified as available does the restore of the rest of the data base continue. If the user enters NO instead of a substitute volume name, the restore is halted. (POLERs #48007, 34245, 32190).

If the before-image file is on Volume A while the schema table is on Volume B, then VERIFY SCHEMA correctly reports the volume on which the schema table resides. In addition, a saved version of this schema will now also indicate the correct volume (POLER #31024).

If a subschema resides on one volume and all nonsubschema files reside on a different volume, the subschema may now be deleted using DBACP's DELETE SUBSCHEMA command (POLERs #28882, 32706, 44171).

An area file may be expanded onto another volume, even when another data base file already exists on that volume (POLERs #36992, 32869).

If a data base resides on volume A with the exception of some CALC files on volume B, and the data base is restored on a system where volume B is not available, DBACP is now able to restore all CALC files to a different specified volume (POLERs #12002, 47633).

SCHEMA: SCHEMA now generates a fatal error when the schema table size exceeds the maximum of 32,767 words (POLER #37380).

SCHEMA now flags as an error an attempt to declare a record more than once as a member record of the same set (POLER #34475).

SCHEMA now flags a bit string that exceeds the maximum length of 1056 bits (POLER #34475).

SCHED: The descriptions of data items output by SCHED now correspond to those in the schema. For example, S9(8)V99 in the schema is now described by SCHED as 99999999V99 (POLER #37614).

The size of an internal table has been increased so that adding a record with a large number of items is very unlikely to cause an overflow (POLER #34480).

CSUBS: If a virtual record section follows a set section in which only certain sets are copied (i.e. not COPY ALL SETS), that virtual record section is now compiled (POLER #41616).

CSUBS now generates a fatal error if it detects that a subschema is too large for DBMS runtime to handle (POLER #37380).

FSUBS: FSUBS now generates a fatal error if it detects that a subschema is too large for DBMS runtime to handle (POLER #37380).

Logical arrays are now flagged if they exceed the maximum size, currently 66 (POLER #34470).

A data item of type DATE is now chunked if the item is declared in the subschema as INTEGER*4 (POLER #33510).

ASG: DBMS now supports the VISTA SELECT command where the underlying base record is located in up to 15 areas (POLER #36385).

Fixed at Rev. 19.0 and Rev. 19.1

DMLCP: DMLCP is now able to read or write addresses greater than 32K bytes in the user work area. Only Rev. 18.2 FTN dml programs were affected by this problem.

DBMS no longer gets into an infinite error loop when a fatal error is encountered.

Set sort orders are now correctly maintained. The new search algorithm may yield up to 15% improvement in searches of large sets (POLER #20941).

Concurrency errors will no longer occur due to data being retained across transactions (POLER #40402).

DBUTL: Switching from an area to a set or from one list to another no longer displays useless information (POLERs #27966, 44172).

After the DBUTL editor is used to patch a set, the modified data set is now available, so the issuance of another set command is not required.

DBUTL no longer gives incorrect statistics during a DUMP AFTER.

DBACP: After imaging is now correctly turned off following a PACK or EXPAND of a data base (POLERS #32193, 32192).

CDML/FDML: All keywords are now treated as reserved words in DML, whether they are required or otherwise. For example, the word OTHER is a keyword in the ON ERROR clause and is consequently reserved. Therefore, use of OTHER as a field name will correctly generate a fatal error (POLER #37464).

CDML now generates the correct error messages when an area name cannot be located via a specific OPEN statement (POLER #41434).

CDML will now report the maximum number of data items allowed, as well as how many were found, in a FETCH in the DML statement (POLER #32202).

CDML now accepts ON ERROR clause paragraph names that begin with digits (POLER #12613).

An FDML compilation no longer produces an L_ file; therefore such a file is not left open if it encounters errors (POLER #20712).

Both CDML and FDML will now close all files under normal and error conditions (POLER #32346).

First Fixed at Rev. 19.1

CSUBS: CSUBS now accepts a period after each section — AREA SECTION, RECORD SECTION, SET SECTION, VIRTUAL RECORD SECTION (POLERs #48256, #82153).

CSUBS now lists compile-time errors both on the screen and in the output listing file (POLERs #41449, #51139).

The size of the RENAMING SECTION has been increased from 32,000 to 32,767 words. Various other internal tables have been similarly increased (POLER #44173).

DBACP: The minimum amount of free space required for a bucket to remain on the available space chain is now set by DBACP to be ten percent of the size of the largest record in the area, or 9 words, whichever is larger. This will decrease fragmentation of records, particularly large ones, hence requiring fewer disk accesses and improving performance. Rev. 18.4 had increased the amount to a flat 100 words per bucket; Rev. 19.1 further refines the algorithm to adjust to each schema definition (POLERs #40183, 40182, 47656, 31413).

A VERIFY VOLUMES now reattaches to the home directory before returning to DBACP command level (POLERs #47652, 47096).

The number of FCBs allocated to a user at DBACP invocation has been increased from 64 to 128. This change allows operations to be performed on a schema with a greater number of files than were previously allowed.

DBUIL: DBUIL now dumps records fragmented across up to 20 buckets (POLER #28865).

DMLCP: DMLCP now removes a bucket from the available space chain if it contains less than the minimum space set by DBACP. If the data base is pre-Rev. 19.1, and a DBACP EXPAND has not been done under Rev. 19.1, the value used will be 100, as in Rev. 18.4 (POLERs #40182, 40183).

Subordinate items in a repeating or naming group now have the same privacy locks as any group names above them.

A record may not be stored if its search, sort, or CALC keys exceed the maximum size of 30 words.

A FIND/FETCH using an invalid DBK will now give the following error message with a minor code of 12: DATABASE KEY NOT AVAILABLE (POLER #45820).

After a fatal DBMS error, incomplete and/or incorrect deallocation of shared data structures has been known to make DBMS and CLUP hang. This problem has been fixed. Additional checking is now made as files are closed (POLER #32637).

SCHED: SCHED now removes a bucket from the available space chain if it contains less than the minimum space set by DBACP. If the data base is pre-Rev 19.1, and a DBACP EXPAND has not been done under Rev 19.1, the value used will be 100, as in Rev 18.4 (POLERs #40182, 40183).

SCHED now disallows the addition of items to a record initially one word in length (POLER #48051).

SCHED now supports record expansion across 20 buckets.

SCHED now disallows the addition of items with field length greater than the DBMS limit of 132 characters (POLER #28859).

SOFTWARE PROBLEMS OUTSTANDING

DBACP

DBACP does not consistently close all opened files (POLERs #34772, 36602, 37900, 37752, 81962, 34052).

The DELETE SCHEMA command sometimes causes DBMS errors because the SD# entry was left around (POLERs #37900, 46653, 81975).

DBACP gives nondescriptive error messages when it fails to open DB# file (POLER #33542).

DBACP does not always restore DB properly if DB was saved without the after image file (POLERs #34679, 25408, 48007).

DBACP lacks the facility to retry a save to or restore from tape that has hit a bad spot on the tape (POLERs #34481, 35201, 40760).

DBACP does not allow a user to verify an after image tape and still turn after-image on after the verify (POLER #20531).

Entering incorrect treename during after-image restore may cause a pointer fault (POLER #32211).

PACK and EXPAND of areas may cause high I/O usage (POLER #32867).

DBACP does not report the size of the data base with complete accurracy (POLERs #40934, 41637).

Allocation of an empty area gives a 33 EOF error (POLER #46238).

DMLCP

DMLCP cannot reference subschemas larger than 64KB (POLERs #37381, 37380).

START TRANSACTION UPDATE is not flagged when the areas are opened in retrieval mode (POLER #34499).

CSUBS/FSUBS

FSUBS does not indicate line number of a duplicate element name (POLER #36876).

Chunking of packed fields by CSUBS may be inconsistent (POLER #46660).

RLIB

The transaction bit map may overflow if there is a hung or very slow user (POLER #29298).

SCHDEC

SCHDEC does not accept single quotes around a pathname with passwords (POLER #33119).

SCHDEC truncates V99X to X on output source (POLER #33849).

SCHDEC truncates output source pathnames to 35 characters (POLER #36005).

An item that is signed (PIC S99V99) before a schema decompile will become unsigned (PIC 99V99) afterwards (POLER #42934).

SCHEMA

The schema compiler does not flag as an error an attempt to define a search, sort, or CALC key with a length greater than the maximum of 30 words. This causes DMLCP to give an internal fatal error when such records are accessed (POLER #43242).

SCHED

SCHED fails to stop after-imaging. Any after-images made after the edit session will not correctly roll forward the data base (POLER #48540).

DBMS/QUERY

NEW FEATURES

Four syntax changes to DBMS/QUERY have been made for Rev. 19.1. These changes are summarized as follows:

1. The syntax for certain commands that operate on tables, formats, and procedures now include optional words that increase the consistency of the syntax. The function of each of the commands remains the same.

2. The SET and PRINT/SPOOL commands now allow reporting directly to an output file in addition to the terminal and printer. This gives you more flexibility to alter the QUERY output before printing it. The PRINT/SPOOL command also includes additional printer options.

3. The virtual record count that is displayed during a SELECT command can now be turned off.

4. The table operations now include the new command PRODUCE. This command allows you to combine two tables that have been separately selected into one table for reporting purposes.

Additionally, public and private startup procedures can now be used. This enables standard site-specific or user-specific commands (or both) to be entered automatically at the start of a session. The function of the startup procedure is to avoid repetitive typing and ensure a consistent environment for each QUERY session.

For more information on these changes, refer to the following section.

DOCUMENTATION CORRECTIONS AND ADDITIONS

This section contains documentation changes for the DBMS/QUERY Reference Guide (IDR4607).

Syntax Changes

The following syntax changes are listed in the order in which they appear in the DBMS/QUERY Reference Guide.

SELECT Command: The optional words TABLE and FORMAT have been added to the FOR and USING options, respectively. Replace the SELECT syntax on page 4-2 with the following:

```
SELECT [ AND DISPLAY
       | target-list
       | FOR TABLE table
       | FROM virtual-record
       | WHERE condition ]
       [ USING [ PUBLIC  ] FORMAT format ]
               [ PRIVATE ]
```

DELETE Command: The optional word TABLE or TABLES can now be specified in either option of the DELETE command. Replace the DELETE syntax on page 7-2 with the following:

```
DELETE [ [TABLES] table-list ]
       [ ALL [TABLES]        ]
```

DISPLAY Command: The optional words TABLE and FORMAT have been added to this command. Replace the DISPLAY syntax on page 7-2 with the following:

```
DISPLAY [ TABLE [table-1]  [ USING [ PRIVATE ] FORMAT format ] ]
                                   [ PUBLIC  ]
        [ ONLY item-list [OF TABLE table-2]                    ]
```

REMOVE DUPLICATES Command: The optional word TABLE has been added to both the FROM and GIVING clauses of this command. Replace the REMOVE DUPLICATES syntax on page 7-3 with the following:

```
REMOVE DUPLICATES

       [FROM TABLE table-1]

       [GIVING TABLE table-2]
```

RENAME Command: The optional word TABLE can now be specified with both the old and new table names. Replace the RENAME syntax on page 7-4 with the following:

```
RENAME [ TABLE table-1] TO TABLE table-2
```

SAVE Command: The optional word TABLE or TABLES can now be specified in either option of this command. Replace the SAVE syntax on page 7-4 with the following:

```
SAVE [ [ TABLE  table] [TO filename]  ]
     [                                ]
     [ ALL [TABLES] [TO pathname]     ]
```

SORT Command: The optional word TABLE can now be specified with the old and new table names. Replace the SORT syntax on page 7-5 with the following:

```
SORT [ TABLE table-1] [GIVING TABLE table-2]

                [ ASCENDING                    ]
                [            ON item-list-1    ]
                [ DESCENDING                   ]

                [ , ASCENDING                  ]
                [           ON item-list-2 ... ]
                [   DESCENDING                 ]
```

RENAME PROCEDURE Command: The optional word PROCEDURE can now be specified with the new procedure name. Replace the RENAME PROCEDURE syntax on page 8-7 with the following:

```
RENAME [ PRIVATE ] PROCEDURE procedure-1 TO [PROCEDURE] procedure-2
       [         ]
       [ PUBLIC  ]
```

RENAME FORMAT Command: The optional word FORMAT can now be specified with the new format name. Replace the RENAME FORMAT syntax on page 10-5 with the following:

```
RENAME [ PRIVATE ] FORMAT format-1 TO FORMAT format-2
       [         ]
       [ PUBLIC  ]
```

New Functionality in Existing Commands

You can now write QUERY output to a file, as well as to a terminal or printer. The FILE option, like the TERMINAL and PRINTER options, concerns the formatted report derived from a retrieved table.

Note

The additional functionality offered by the FILE option should not be confused with that provided by the SAVE command. SAVE saves only a retrieved table in its internal form, while FILE results in a formatted ASCII report.

The SET, PRINT, and SPOOL commands now include the new FILE option. The following changes should be made to the existing documentation:

SET Command: This command defines the page dimensions used for the terminal, line printer, or file. Replace the SET syntax and the information on default values on page 6-4 with the following:

```
SET { TERMINAL } || LENGTH [ TO n LINES     ] ||
    { PRINTER  }
    { FILE     }  || WIDTH  [ TO m POSITIONS ] ||
```

n and m must be positive integers. The administrator can define installation-wide defaults when DBMS/QUERY is installed if the standard defaults do not meet the installation requirements. The system default values are as follows:

Device	Length	Width
Terminal	23 lines	79 positions
Printer	66 lines	132 positions
File	66 lines	132 positions

If the TO clause of the SET command is not specified, the length and/or width values for the device are reset to the default values in effect (either system or installation-wide).

When scrolling is enabled, the continuation prompt appears every n lines as defined by the SET command. If n is not specified, the continuation prompt appears every default-value line. Similarly, if a format declaration includes page numbering, the page numbers appear every n or default-value line.

If a format used with a DISPLAY or PRINT/SPOOL command includes page size declarations, those declarations override the page sizes established by the SET command for the duration of the DISPLAY or PRINT/SPOOL command.

PRINT/SPOOL Command: This command now sends table information to a line printer or a disk file. It includes the following options:

Device	Option
Printer	Number of copies Form type Location of printer Name of report
Disk file	Name of file

Replace the PRINT/SPOOL syntax on page 7-2 and add the accompanying information to that already present on page 7-3. The DISPLAY syntax remains unchanged.

```
{ PRINT }
{       } [n COPIES OF ]
{ SPOOL }

    [ TABLE [table-1] [out-option-list] [USING format-ref] ]
    [ ONLY item-list [OF TABLE table-2] [out-option-list]  ]
```

where format-ref is

```
[ PRIVATE ]
[         ] FORMAT format
[ PUBLIC  ]
```

and out-option-list is

```
[ [ { ON   form-1 } ]                ]
[ [ { FORM form-2 } ]                ]
[ [ AT destination ]            ...  ]
[ [ AS output-name ]                 ]
[ TO FILE [filename]                 ]
```

The COPIES clause can be specified only for the printer. That is, the FILE option and the COPIES clause cannot be used in the same command. n must be a positive integer. Omit the COPIES clause when you require only one copy of a report.

The ON/FORM, AT, and AS clauses refer to the printer only. One PRINT/SPOOL command can contain one clause for each of these options. (ON and FORM are synonymous; use only one per PRINT/SPOOL command.)

ON/FORM specifies the form type. AT specifies printer location. AS specifies the report name. The specified name will appear as a banner on the output report. If the specified filename exists, you will be asked for permission to overwrite the file. Alternatively, you will be asked for another filename if you do not wish to overwrite the existing file. If you do not specify the AS clause, the banner appears as follows:

named table: tablename.RPT

unnamed table: username.RPT

If you specify a filename with the FILE option, that name will be applied to the file written to disk. If you specify the FILE option without a filename, DBMS will name the file as follows:

reporting from a named table: tablename.n.RPT

reporting from an unnamed table: username.n.RPT

The n in the generated filename is a positive integer that ensures its uniqueness. This number starts at 1 and is incremented by 1 until a unique filename is found. The command is aborted if n reaches 10,000.

New Commands

ENABLE/DISABLE VIRTUAL RECORD COUNT Command: Whenever a SELECT command without the DISPLAY option is specified, DBMS/QUERY displays a running count and a total count of retrieved records. This new command allows this display to be turned off and to be restarted, as circumstances dictate. This feature is especially useful for hard-copy terminals or slow terminals from which selection of data base records is I/O bound to the printing of the virtual record count. Add this discussion and the following syntax to Section 6.

{ ENABLE / DISABLE } [VIRTUAL] [RECORD] COUNT

The virtual record count is always ENABLED, unless specifically disabled. This command can be included in a public or private startup procedure if disabling the count is a site or user standard.

PRODUCE Command: This command combines two previously selected, identically structured tables into a new table according to specified conditions. The input tables must contain the same item names and data types. This feature can eliminate unnecessary SELECTs if the information desired has already been retrieved but is located in more than one table. It can also be used to provide more meaningful information by combining tables derived from different record types that contain the same item names, attributes, and meanings.

Add this information and the following syntax discussion to Section 7.

PRODUCE TABLE [table-1] FROM TABLE table-2
pro-op TABLE table-3

where pro-op is

{ IN COMMON WITH
| IN UNION WITH
| BUT NOT IN }

The resulting table becomes the current table. It will not have any duplicate rows even if the two input tables contain duplicate rows.

The output table will be unnamed unless you specify table-1. The SELECT commands for table-2 and table-3 must specify the same number of items, the same item names, and the same data types. (Item names can be either fully qualified with the record name or not.) If the items selected in table-2 and table-3 are in different order, the resulting table will be in the same order as table-2.

The rows of data contained in the output table vary according to the operation specified in the PRODUCE command, as follows:

Operation	Output Table Contents
IN COMMON WITH	Rows that are the same in both table-2 and table-3
IN UNION WITH	Rows that are in table-2 and rows that are in table-3
BUT NOT IN	Rows that are in table-2 and not in table-3

One use of the PRODUCE command could be to combine output selected from two separate sets that make up a bill of materials relationship in a data base. Another example is the diamond structure. In that instance, you could create two separate subschemas that are identical except for the virtual records. In one subschema the virtual record path would travel up the left side of the diamond. In the other, the path would travel up the right side. You can then use the PRODUCE command to combine the tables selected separately from each of these virtual records.

The following examples show two tables, TAB1 and TAB2, that were selected from different virtual records in the same subschema but include the same fields. Three new tables are then constructed from those two tables using the PRODUCE command.

Example 1:

TAB1 is created by selecting the customer and order numbers from the item-record for all those records that have ordered a quantity less than 5. The table has 16 rows. TAB2 is created by selecting customer and order numbers from the order-record for all those customers not in the state of Illinois. The table has 10 rows. Note that TAB1 has duplicate rows.

```
> SE DI CUSTOMER-ID, ORDER-NUMBER FOR TAB1 FROM ITEM-RECORD ~
> WHERE ITEM-QUANTITY LT 5
CUSTOMER-ID  ORDER-NUMBER

       1005        100006
       1006        100007
       1006        100007
       1010        100008
       1010        100008
       1010        100008
       1010        100008
       1010        100008
       1007        100009
       1011        100010
       1011        100010
       1011        100010
       1011        100010
       1011        100010
       1011        100010
       1004        100005

> SE DI ORDER-NUMBER, CUSTOMER-ID FOR TAB2 FROM ORDER-RECORD ~
> WHERE CUSTOMER-STATE-COUNTY NE 'IL'
ORDER-NUMBER  CUSTOMER-ID

      100001         1001
      100004         1003
      100006         1005
      100007         1006
      100008         1010
      100009         1007
      100010         1011
      100005         1004
```

Example 2:

TAB3 is created by combining all rows that are the same in TAB2 and TAB1. That is, the new table will contain all current order numbers for customers that are not in the state of Illinois and have placed orders for less than five items. Note that the new TAB3 has no duplicate rows. Note also that the items in TAB3 are in the sequence of those in TAB2 since TAB2 is mentioned first in the PRODUCE command.

```
> PRODUCE TAB3 FROM TAB2 IN COMMON WITH TAB1
The new table contains 6 rows.
> DI
ORDER-NUMBER  CUSTOMER-ID

      100006         1005
      100007         1006
      100008         1010
      100009         1007
      100010         1011
      100005         1004
```

Example 3:

TAB4 is created by combining all the data in TAB1 and TAB2, but without the duplicate rows. That is, TAB4 contains all the order numbers that contain orders for less than five items and all the orders that were placed by customers not having an office in Illinois.

```
> PRODUCE TAB4 FROM TAB2 IN UNION WITH TAB1
The new table contains 8 rows.
> di
ORDER-NUMBER  CUSTOMER-ID

      100001         1001
      100004         1003
      100006         1005
      100007         1006
      100008         1010
      100009         1007
      100010         1011
      100005         1004
```

Example 4:

TAB5 is created by extracting those rows that are only in TAB2. Any rows that appear in both tables are not used in TAB5. That is, TAB5 consists of those customers that are not in Illinois but have placed orders of five or more items.

```
> PRODUCE TAB5 FROM TAB2 BUT NOT IN TAB1
The new table contains 2 rows.
> di
ORDER-NUMBER  CUSTOMER-ID

      100001         1001
      100004         1003
```

Example 5:

Another table is requested for all rows in TAB1 that do not appear in TAB2, but none exist.

```
> PRODUCE TAB6 FROM TAB1 BUT NOT IN TAB2
The new table is empty.
```

Startup Procedures

Add the following information to page 8-3, prior to the section on GENERAL RULES.

A procedure designated as a startup procedure is executed automatically when you sign on to DBMS/QUERY. The startup procedure enables users to configure their environment to their own tastes, either on a single user or an installation-wide basis. It can also include specific commands that are always executed upon entering a QUERY session.

Startup procedures can be public or private, and are created in the same manner as all other procedures. DBMS/QUERY determines that a procedure is a startup procedure by its name. The default name is STARTUP. However, the administrator can specify a different name by changing the configuration file and resharing DBMS/QUERY.

Startup procedures are not required. Either the public or private startup can execute without the other. However, if a public startup exists, it will be executed before the private startup. If an error occurs during execution of the public startup procedure, the private procedure will not execute automatically. DBMS/QUERY informs the user that a startup procedure is executing.

Reserved Words

Add the following reserved words to page B-1:

AS
BUT
COMMON
FILE (FIL)
PRODUCE (PRO)
UNION

SOFTWARE PROBLEMS FIXED

Fixed at Rev. 18.4 and Rev. 19.1

File System Errors: DBMS/QUERY no longer displays the complete pathname (including passwords) of a file in the PDBMS UFD when a file system error occurs during file opening (POLER #37621).

File system errors during SAVE commands no longer close all DBMS/QUERY file units, causing INTERNAL ERRORs.

Footer Position: In list and block detail, page ejects no longer cause footers to be misplaced.

Footer position is now properly checked when margins are specified.

PRINT Command: The PRINT command sometimes issued the error message "An error occurred while trying to open the spool file" when a complicated data base was being accessed on a system with fewer than the default number of user segments configured. This problem has been corrected (POLER # 48075).

Justification: Zero replacement strings (strings following the ZERO option) are now right-justified instead of left-justified.

Any justification specifications that occur in list or block detail now correctly override default justification. Thus, to left-justify all items in a list (including numeric items, which are right-justified by default), the user must specify LEFT LIST detail.

Spurious Blanks: A problem that caused spurious blanks between the page heading and the first line of block detail has been corrected (POLER #44283).

Fixed at Rev. 19.0 and Rev. 19.1

SELECT Command: When using the SELECT command, where the underlying base record is in more than 9 areas, DBMS error 2710F is no longer produced. The new limit is now 15 areas (POLER #36385).

Cover Item on Line 0: The Report Generator no longer accepts items on the cover specified to be on line 0. Previously, it was accepting such specifications and placing the items on line 1.

First Fixed at Rev. 19.1

Long Login Names: If a user with a login name more than six characters long created a private procedure or format, DBMS/QUERY aborted the command with the file system error "File not found." (POLER #00069).

Terminal Mode: When DBMS/QUERY was exited, terminal mode was set to full-duplex even if it was set to half-duplex when the user entered the subsystem (POLER #43044).

SAVE TO: If the user typed SAVE TO at command level without a pathname, all subsequent commands were ignored until the user hit the BREAK key (CTRL-P).

Large Subschemas: If a subschema with a total size greater than 32,767 words was used, the LIST RECORDS command caused DBMS/QUERY to go into an infinite loop that could only be exited by force-logging the user out (POLER #31175).

Fast Index: Because of the need to increase the size of the DBMS/QUERY fast index, and the already massive working set size, the DBMS/QUERY fast index (which is read-only) has been modified to be shared by all users. This also eliminates the "One moment please..." message that occurred while a user tried to retrieve information from the DBMS/QUERY HELP data base.

STATUS Command: To decrease the amount of printing required by the STATUS command after the user has changed the size of the terminal, printer, or file from the defaults, the STATUS command output has been modified.

Report Generator: Double precision (REAL*8) numbers are now printed with 4-digit exponents. DBMS/QUERY has been modified to handle this properly.

SOFTWARE PROBLEM OUTSTANDING

SELECT

DBMS/QUERY will not allow a SELECT of the entire base record (i.e. SELECT FROM base-record-name) when the base record contains over 229 items. Instead, the command is aborted with the DBMS error message TOO MANY TARGET LIST ITEMS (POLER #45927).

MIDAS

NEW FILE DUMP UTILITY (MDUMP)

Rev. 19.1 MIDAS includes a new file dump utility called MDUMP. MDUMP replaces the KDUMP utility, which is no longer included on the Master Disk. This section describes:

- MDUMP's uses
- MDUMP's options
- The dump file that MDUMP creates
- The MDUMP dialog
- Status and descriptive messages that MDUMP produces
- Error messages that MDUMP produces

In addition, this section includes several sample MDUMP sessions.

MDUMP's Uses

MDUMP dumps a specified MIDAS file into a sequential disk file. By examining this sequential file, you can see at a glance all of the data records (and optionally, the key values) in a MIDAS file. MDUMP is therefore useful for debugging MIDAS applications.

MDUMP can also be used to rebuild existing MIDAS files. You can dump a MIDAS file, edit the resulting sequential file, and feed the edited file to the KBUILD utility. KBUILD uses the sequential file to build a new MIDAS file. KBUILD is described in the MIDAS User's Guide (IDR4558).

MDUMP's Options

When you run MDUMP, a series of prompts ask you about the order, contents, and format of the dump you wish to perform. The prompts also ask you how and where MDUMP should record status and error information during the dump. The choices you can make are as follows:

- Order of dump

  MDUMP can order the output records by any of the index keys or according to the sequence of the data subfile records. (Data subfile records are stored in the order in which they are added

to the file, and thus are not necessarily ordered by key value.) If you dump by a secondary key, only the records that contain that secondary key will be dumped.

- Contents of dump

  You can specify that the output file should contain data records (that is, data subfile entries) only, or index entries only, or both data records and index entries. For example, to check pairs of primary and secondary key values, you might dump only the primary index and one secondary index.

- Format of dump

  MDUMP can produce output in any of the formats that KBUILD accepts: BINARY, COBOL, FTNBIN, RPG, or TEXT. (Refer to the MIDAS User's Guide for an explanation of these formats.)

- Log/error file

  Whenever you perform a dump, MDUMP displays information about the layout of the output file. It also displays a status report of the dump and any appropriate error messages. If you provide the name of a log/error file, MDUMP writes this information to the file as well as to the screen.

- Milestone recording

  The status report that MDUMP displays consists of a series of entries made periodically during the dump. Each entry includes information such as the current time, the amount of CPU and disk time used so far in the dump, and the number of records processed so far. The milestone count determines how many of these entries are generated.

  For a milestone count of 0, MDUMP produces two entries: one when the dump begins and one when it is complete. For a milestone count of N>0, an entry is produced for every N records processed. Any errors that MDUMP finds while performing the dump (for example, a record with an invalid primary key) are reported along with the milestone statistics.

The Sequential Dump File

Each record of MDUMP's sequential dump file corresponds to one record in the MIDAS file. The following are the fields that a record of MDUMP's output can contain:

- The data subfile entry of the MIDAS record
- The length of the data subfile entry (if the MIDAS file has variable-length records)

- The direct access record number (if the MIDAS file is a direct access file)
- The primary key
- The secondary key (if the dump is by secondary key)

An MDUMP file would not actually contain all of these fields. For example, direct access files must have fixed-length records, so a record number and a data size would never appear in the same MDUMP file. However, the fields that do appear are in the order of the list above. Note that this order makes the dump file acceptable as input to KBUILD. Sample dump files are shown later in this section. The following points should be noted:

- If you dump both the data subfile and an index whose keys are embedded in the data, then the keys will appear twice in the MDUMP file.
- There are certain cases in which you must dump the primary key. For example, suppose you want to feed the sequential file to KBUILD to build a new MIDAS file. If the primary key is not embedded in the data record, the only way for you to provide KBUILD with the key is to dump it along with the data.
- Secondary data cannot be dumped.
- If you suspect that your index subfiles have been damaged, and if the keys are embedded in the data, you can use MDUMP and KBUILD to rebuild the file. Dump the data records in the order of physical storage without dumping any indexes. This will prevent MDUMP from referring at all to the damaged index subfiles.
- MDUMP recognizes and reports on any damage that it finds in the MIDAS file being dumped. Error messages are included in the milestone reports. MDUMP's error messages are described later in this section.

The MDUMP Dialog

To invoke MDUMP, enter the command MDUMP. The utility announces itself and begins the dialog. MDUMP's prompts are listed below. Each prompt is followed by a summary of the possible responses to it.

1. ENTER TREENAME OF MIDAS FILE TO DUMP:

   Enter the treename of the MIDAS file to be dumped.

2. ENTER DUMP METHOD ('DATA' OR AN INDEX #):

   Enter DATA to dump records in the order of the data subfile

records. Enter an index number (0 to 17) to dump records in ascending order by that index.

3. DO YOU WANT THE DATA RECORD DUMPED?

   Enter YES if the data portion of the MIDAS record is to be dumped. Enter NO if only index keys are to be dumped.

4. DO YOU WANT THE PRIMARY INDEX KEY DUMPED?

   Enter YES if the primary key is to be dumped, and NO if not.

5. DO YOU WANT THE INDEX <#> KEY DUMPED?

   This prompt appears only if a secondary index was specified in response to prompt 2. In that case, # designates the number of that secondary index. Enter YES if you want the index dumped, and NO if not.

6. ENTER OUTPUT FILE TREENAME:

   Enter a filename for the output file. If a file by that name already exists, MDUMP will supply an error message and ask for another name.

7. ENTER OUTPUT FILE FORMAT:

   Enter BINARY, COBOL, FINBIN, RPG, or TEXT. You can also enter a carriage return or the word HELP to obtain a list of these options. For an explanation of the formats, refer to the MIDAS User's Guide.

8. ENTER LOG/ERROR TREENAME:

   Enter the name of the file to be opened for recording errors and statistics. If the file already exists, it will be overwritten. Enter a carriage return if no log/error file is to be opened. In the latter case, statistics and error messages are displayed on the screen as MDUMP runs.

9. ENTER MILESTONE COUNT:

   Enter appropriate number of records for milestone report intervals. Enter 0 for the briefest version of the status report.

Status and Descriptive Messages

MDUMP produces a series of messages that describe the status of the dump and the format of the output file. All descriptive and status messages are displayed at the terminal. If you specified a log/error file, all messages are also written to that file. This section

describes the dump process and the messages that normally are produced.

When the dialog is complete, MDUMP uses your responses to plan the dump file's format. If you are dumping data from a MIDAS file with variable-length records, the first set of messages you see looks like this:

```
SCANNING MIDAS FILE TO DETERMINE LARGEST RECORD
COUNT      DATE      TIME      CPU MIN    DISK MIN    TOTAL TM    DIFF
    0     1-18-82  23:59:50      0.000       0.000       0.000   0.000
    3     1-18-82  23:59:50      0.001       0.001       0.002   0.002
    6     1-18-82  23:39:51      0.002       0.002       0.004   0.002
```

This set of messages indicates that MDUMP is reading through the MIDAS file to find the longest record. MDUMP does this to determine how long the dump records must be. (The dump file always has fixed-length records.) In the example above, the MILESTONE COUNT was 3, so a report was generated for each three records scanned. Each milestone report (one row) shows how many records have been scanned so far (COUNT); the current date and time; the CPU, disk, and total (CPU + disk) time (in minutes) elapsed since the beginning of the scan; and the increment of total time elapsed since the last milestone.

Next, MDUMP tells you exactly what you will find in each word of an output record. One message is produced for each field appearing in the output file. A list of the possible messages follows. Messages 1 through 3 always appear.

1. FORMAT OF MDUMP DUMP FILE: treename_of_dump_file

2. DUMP FROM MIDAS FILE: treename_of_MIDAS_file

3. RECORDS ARE record_length WORDS LONG
   WRITTEN IN format_name FORMAT

   record_length is the length (in words) of the entire MDUMP output record. format_name is BINARY, COBOL, FTNBIN, RPG, or TEXT.

4a. THE DATA PORTION OCCUPIES WORDS 1 THRU x

4b. THE DATA PORTION IS VARIABLE AND OCCUPIES WORDS 1 THRU x

   These messages appear only if you are dumping data records. Message 4a appears if the dump file is a text file, and message 4b appears for all other types of dump files. x is the last word occupied by the data.

5a. THE DATA LENGTH IS SPECIFIED AS A ASCII STRING
   IN BYTES x THRU y

5b. THE DATA LENGTH IS SPECIFIED AS A BINARY STRING
   IN WORD z

These messages appear only if you are dumping variable-length data records. Message 5a appears if the dump file is a text file. x and y are the starting and ending bytes, respectively, of the data length. Message 5b appears for nontext dump files. BINARY STRING refers to a single-word INTEGER*2 integer. z is the word that contains the data length.

The data length is the length of the data record in the MIDAS file — that is, before any padding occurred in the dump. You can use this information to distinguish between blanks that are part of the data and blanks that merely pad short records.

6a. THE DIRECT ACCESS RECORD NUMBER
IS SPECIFIED AS A ASCII STRING IN BYTES x THRU y

6b. THE DIRECT ACCESS RECORD NUMBER
IS SPECIFIED AS A BINARY STRING IN WORDS z THRU w

These messages appear only if you are dumping direct access data records in order of DATA or primary key.
Message 6a appears if the dump file is a text file. x and y are the starting and ending bytes, respectively, of the direct access record number. Message 6b appears for nontext dump files. BINARY STRING refers to a two-word REAL*4 floating point number. z and w are the starting and ending words, respectively, of the direct access record number.

If you dump a direct access file by a secondary key, or do not dump the data records, then the record numbers will not appear in the output file.

7. THE PRIMARY KEY (INDEX 0) IS A key_type KEY
IN BYTES x THRU y

This message appears only if you are dumping the primary key. key_type is the data type of the key as defined in the CREATK session that created the MIDAS file. x and y are the starting and ending bytes, respectively, of the primary key.

8. THE INDEX w KEY IS A key_type KEY IN BYTES x THRU y

This message appears only if you are dumping a secondary key. Again, key_type is the data type of the key as defined in the CREATK session. x and y are the starting and ending bytes, respectively, of the secondary key.

After these messages are produced, the dump begins. MDUMP traverses the MIDAS file in the specified order, reading in records, constructing and writing output records, and producing milestone reports as requested in the dialog. The following is an example of a short milestone report (MILESTONE COUNT = 0).

```
BEGINNING DUMP
 COUNT      DATE       TIME      CPU MIN    DISK MIN    TOTAL TM      DIFF
     0  12-07-81   16:03:51        0.000       0.000       0.000       0.000
DUMP COMPLETE,       7 RECORDS DUMPED
     8  12-07-81   16:03:51        0.001       0.001       0.002       0.002
```

The column headings have meanings similar to those described above. This time, COUNT indicates the number of records processed so far in the dump. CPU and disk time are measured from the beginning of the dump. Note that the number of records dumped may not be the same as the COUNT of the last milestone. This is because records that have been marked for deletion will be counted in COUNT, but not dumped.

Error Messages

When MDUMP dumps a file, errors that it encounters are reported along with the milestone statistics. The following are MDUMP's error messages and their meanings:

- BAD DATA RECORD POINTER - IGNORED

MDUMP found a bad data record pointer in the MIDAS file. The dump continues.

- BAD INDEX BLOCK OR INDEX BLOCK POINTER

MDUMP found a bad index block or index block pointer in the MIDAS file. The dump halts.

- INDEX BLOCK SIZE GREATER THAN MAXIMUM DEFAULT

MDUMP found an index block larger than the maximum default size. The dump halts.

- UNABLE TO REACH BOTTOM INDEX LEVEL

MDUMP found a bad index block or index block pointer before dumping any records. The dump does not occur.

Sample MDUMP Sessions

This section presents the dialog and output from six sample MDUMP sessions. The first four examples show how the data and keys of one file can be dumped in different combinations, orders, and formats. The fifth example shows a dump of a file with variable-length records. In the sixth example, a direct access file is dumped.

The MIDAS file DUMP.EX in Examples 1 through 4 has fixed-length records (nine words long), a one-word ASCII primary key, and a one-word ASCII secondary key (with duplicates).

Example 1:

In this example, the data and primary key of the file DUMP.EX are dumped in DATA order and in RPG format. Note that the primary keys are not sorted.

```
OK, mdump
[MDUMP rev 19.1]

ENTER TREENAME OF MIDAS FILE TO DUMP: dump.ex
ENTER DUMP METHOD ('DATA' OR AN INDEX #): data
DO YOU WANT THE DATA RECORD DUMPED? y
DO YOU WANT THE PRIMARY INDEX KEY DUMPED? y
ENTER OUTPUT FILE TREENAME: ex.1.dump
ENTER OUTPUT FILE FORMAT: rpg
ENTER LOG/ERROR FILE NAME: log
ENTER MILESTONE COUNT: 2


FORMAT OF MDUMP DUMP FILE:  <TPUBS>EMILY>EX.1.DUMP
DUMP FROM MIDAS FILE: <TPUBS>EMILY>DUMP.EX

RECORDS ARE     10 WORDS LONG WRITTEN IN 'RPG    ' FORMAT
THE DATA PORTION OCCUPIES WORDS 1 THRU     9
THE PRIMARY KEY (INDEX 0) IS A ASCII STRING KEY IN BYTES    19 THRU 20


BEGINNING DUMP
 COUNT      DATE      TIME      CPU MIN    DISK MIN    TOTAL TM      DIFF
       0  01-19-82  12:42:32       0.000       0.000       0.000     0.000
       2  01-19-82  12:42:32       0.001       0.002       0.003     0.003
       4  01-19-82  12:42:32       0.001       0.002       0.004     0.000
       6  01-19-82  12:42:32       0.002       0.002       0.004     0.000
       8  01-19-82  12:42:32       0.002       0.002       0.005     0.000
DUMP COMPLETE,        8 RECORDS DUMPED
       8  01-19-82  12:42:32       0.003       0.002       0.005     0.000


OK, slist ex.1.dump
01hhfirstrecord   01
02eesecondrecord  02
08hheigthrecord   08
04iifourthrecord  04
05bbfifthrecord   05
07ggseventhrecord 07
09iininthrecord   09
10jjtenthrecord   10
```

Example 2:

This time, the data and primary key of DUMP.EX are dumped in primary key order and in TEXT format.

```
OK, mdump
[MDUMP rev 19.1]

ENTER TREENAME OF MIDAS FILE TO DUMP: dump.ex
ENTER DUMP METHOD ('DATA' OR AN INDEX #): 0
DO YOU WANT THE DATA RECORD DUMPED? y
DO YOU WANT THE PRIMARY INDEX KEY DUMPED? y
ENTER OUTPUT FILE TREENAME: ex.2.dump
ENTER OUTPUT FILE FORMAT: text
ENTER LOG/ERROR FILE NAME: log2
ENTER MILESTONE COUNT: 0


FORMAT OF MDUMP DUMP FILE:  <TPUBS>EMILY>EX.2.DUMP
DUMP FROM MIDAS FILE: <TPUBS>EMILY>DUMP.EX

RECORDS ARE     10 WORDS LONG WRITTEN IN 'TEXT  ' FORMAT
THE DATA PORTION OCCUPIES WORDS 1 THRU     9
THE PRIMARY KEY (INDEX 0) IS A ASCII STRING KEY IN BYTES    19 THRU 20


BEGINNING DUMP
 COUNT      DATE      TIME      CPU MIN   DISK MIN   TOTAL TM     DIFF
       0  01-19-82  11:02:34      0.000      0.000      0.000    0.000
DUMP COMPLETE,       8 RECORDS DUMPED
       8  01-19-82  11:02:34      0.001      0.000      0.001    0.001


OK, slist ex.2.dump
01hhfirstrecord   01
02eesecondrecord  02
04iifourthrecord  04
05bbfifthrecord   05
07ggseventhrecord 07
08hheigthrecord   08
09iininthrecord   09
10jjtenthrecord   10
```

Example 3:

In this example, the data and both keys are dumped in order of the secondary key in TEXT format.

```
OK, mdump
[MDUMP rev 19.1]

ENTER TREENAME OF MIDAS FILE TO DUMP: dump.ex
ENTER DUMP METHOD ('DATA' OR AN INDEX #): 1
DO YOU WANT THE DATA RECORD DUMPED? y
DO YOU WANT THE PRIMARY INDEX KEY DUMPED? y
DO YOU WANT THE INDEX  1 KEY DUMPED? y
ENTER OUTPUT FILE TREENAME: ex.3.dump
ENTER OUTPUT FILE FORMAT: text
ENTER LOG/ERROR FILE NAME: log3
ENTER MILESTONE COUNT: 0


FORMAT OF MDUMP DUMP FILE:  <TPUBS>EMILY>EX.3.DUMP
DUMP FROM MIDAS FILE: <TPUBS>EMILY>DUMP.EX

RECORDS ARE     11 WORDS LONG WRITTEN IN 'TEXT  ' FORMAT
THE DATA PORTION OCCUPIES WORDS 1 THRU     9
THE PRIMARY KEY (INDEX 0) IS A ASCII STRING KEY IN BYTES    19 THRU 20
THE INDEX  1 KEY IS A ASCII STRING KEY IN BYTES    21 THRU    22


BEGINNING DUMP
 COUNT      DATE      TIME     CPU MIN   DISK MIN   TOTAL TM     DIFF
       0  01-19-82  11:02:38      0.000      0.000      0.000    0.000
DUMP COMPLETE,       8 RECORDS DUMPED
       8  01-19-82  11:02:38      0.001      0.000      0.002    0.002


OK, slist ex.3.dump
05bbfifthrecord   05bb
02eesecondrecord  02ee
07ggseventhrecord 07gg
01hhfirstrecord   01hh
08hheigthrecord   08hh
04iifourthrecord  04ii
09iininthrecord   09ii
10jjtenthrecord   10jj
```

Example 4:

Finally, only the two keys of DUMP.EX are dumped. They are sorted by secondary key.

```
OK, mdump
[MDUMP rev 19.1]

ENTER TREENAME OF MIDAS FILE TO DUMP: dump.ex
ENTER DUMP METHOD ('DATA' OR AN INDEX #): 1
DO YOU WANT THE DATA RECORD DUMPED? n
DO YOU WANT THE PRIMARY INDEX KEY DUMPED? y
DO YOU WANT THE INDEX  1 KEY DUMPED? y
ENTER OUTPUT FILE TREENAME: ex.4.dump
ENTER OUTPUT FILE FORMAT: cobol
ENTER LOG/ERROR FILE NAME: log4
ENTER MILESTONE COUNT: 0


FORMAT OF MDUMP DUMP FILE:  <TPUBS>EMILY>EX.4.DUMP
DUMP FROM MIDAS FILE: <TPUBS>EMILY>DUMP.EX

RECORDS ARE      2 WORDS LONG WRITTEN IN 'COBOL ' FORMAT
THE PRIMARY KEY (INDEX 0) IS A ASCII STRING KEY IN BYTES     1 THRU 2
THE INDEX  1 KEY IS A ASCII STRING KEY IN BYTES     3 THRU     4


BEGINNING DUMP
 COUNT      DATE      TIME      CPU MIN   DISK MIN   TOTAL TM      DIFF
       0  01-19-82  11:58:30      0.000      0.000      0.000     0.000
DUMP COMPLETE,       8 RECORDS DUMPED
       8  01-19-82  11:58:30      0.001      0.001      0.002     0.002


OK, slist ex.4.dump
05bb
02ee
07gg
01hh
08hh
04ii
09ii
10jj
```

Example 5:

In this example, the file VAR.EX has variable-length records and a one-word ASCII primary key. The data and primary key are dumped in DATA order. The data length (in words) follows the data and precedes the primary key in each output record.

```
OK, mdump
[MDUMP rev 19.1]

ENTER TREENAME OF MIDAS FILE TO DUMP: var.ex
ENTER DUMP METHOD ('DATA' OR AN INDEX #): data
DO YOU WANT THE DATA RECORD DUMPED? y
DO YOU WANT THE PRIMARY INDEX KEY DUMPED? y
ENTER OUTPUT FILE TREENAME: ex.5.dump
ENTER OUTPUT FILE FORMAT: text
ENTER LOG/ERROR FILE NAME: log5
ENTER MILESTONE COUNT: 0

SCANNING MIDAS FILE TO DETERMINE LARGEST RECORD
 COUNT      DATE      TIME     CPU MIN   DISK MIN   TOTAL TM     DIFF
      0  01-19-82  11:41:57      0.000      0.000      0.000    0.000
SCAN COMPLETE
      4  01-19-82  11:41:57      0.001      0.002      0.003    0.003


FORMAT OF MDUMP DUMP FILE:  <TPUBS>EMILY>EX.5.DUMP
DUMP FROM MIDAS FILE: <TPUBS>EMILY>VAR.EX

RECORDS ARE      9 WORDS LONG WRITTEN IN 'TEXT  ' FORMAT
THE DATA PORTION IS VARIABLE AND OCCUPIES WORDS 1 THRU     5
THE DATA LENGTH IS SPECIFIED AS A ASCII STRING IN BYTES    11 THRU  16
THE PRIMARY KEY (INDEX 0) IS A ASCII STRING KEY IN BYTES    17 THRU  18


BEGINNING DUMP
 COUNT      DATE      TIME     CPU MIN   DISK MIN   TOTAL TM     DIFF
      0  01-19-82  11:41:57      0.000      0.000      0.000    0.000
DUMP COMPLETE,       4 RECORDS DUMPED
      4  01-19-82  11:41:57      0.001      0.000      0.001    0.001


OK, slist ex.5.dump
01first        401
02second       402
03next         303
04another      504
```

Example 6:

In this last example, the file DIR.EX is a direct access file with five-word data records and an ASCII record number that also serves as the primary key. The data and primary key of the direct access file DIR.EX are dumped in TEXT format. The three-word record number/primary key appears three times in each record: once in the data, once as record number, and once as primary key.

```
OK, mdump
[MDUMP rev 19.1]

ENTER TREENAME OF MIDAS FILE TO DUMP: dir.ex
ENTER DUMP METHOD ('DATA' OR AN INDEX #): 0
DO YOU WANT THE DATA RECORD DUMPED? y
DO YOU WANT THE PRIMARY INDEX KEY DUMPED? y
ENTER OUTPUT FILE TREENAME: ex.6.dump
ENTER OUTPUT FILE FORMAT: text
ENTER LOG/ERROR FILE NAME: log6
ENTER MILESTONE COUNT: 0


FORMAT OF MDUMP DUMP FILE:  <TPUBS>EMILY>ERX.6.DUMP
DUMP FROM MIDAS FILE: <TPUBS>EMILY>dir.ex

RECORDS ARE     15 WORDS LONG WRITTEN IN 'TEXT  ' FORMAT
THE DATA PORTION OCCUPIES WORDS 1 THRU     5
THE DIRECT ACCESS RECORD NUMBER
  IS SPECIFIED AS A ASCII STRING IN BYTES    11 THRU    24
THE PRIMARY KEY (INDEX 0) IS A ASCII STRING KEY IN BYTES    25 THRU 30


BEGINNING DUMP
 COUNT      DATE      TIME     CPU MIN   DISK MIN   TOTAL TM     DIFF
      0  12-07-81  16:03:53      0.000      0.000      0.000    0.000
DUMP COMPLETE,       3 RECORDS DUMPED
      3  12-07-81  16:03:53      0.001      0.001      0.002    0.002


OK, slist ex.6.dump
000001data 0.1000000E 01000001
000002data 0.2000000E 01000002
000003data 0.3000000E 01000003
```

OTHER NEW FEATURES

Several changes have been made to MIDAS at Rev. 19.1 in order to support compatibility with MIDASPLUS. (MIDASPLUS is described in the next section of this chapter.)

A file that is being accessed by the online MIDAS routines must not be accessed at the same time by MIDASPLUS, by an unshared version of MIDAS, or by the MIDAS offline utilities. Controls have been implemented to ensure that a MIDAS/MIDASPLUS file will be accessed exclusively by:

- n MIDASPLUS users or
- n unshared MIDAS users or
- 1 MIDAS utility user

In addiion, PRIBLD, SECBLD, and BILD$R only allow one user per file.

The effects of these controls are outlined below.

MIDAS File Opening and Closing

MIDAS now requires that all files be:

- Opened by means of the OPENM$ routine or the NTFYM$ routine. You can either use OPENM$ by itself or use NTFYM$ along with the SRCH$$ or TSRC$$ routine. (SRCH$$ and TSRC$$ are the standard PRIMOS subroutines for opening and closing files.)
- Closed by means of the CLOSM$ routine or the NTFYM$ routine. You can either use CLOSM$ by itself or use NTFYM$ along with the SRCH$$ or TSRC$$ routine.

Error Codes

The OPENM$ routine has three new error codes indicating that a requested file is in use by a conflicting program:

```
10006 -- File in use by unshared MIDAS (from MIDASPLUS only)
10007 -- File in use by MIDASPLUS (from MIDAS only)
10008 -- File in use by a MIDAS utility or user calls
                 to PRIBLD, SECBLD, or BILD$R
```

In addiiton, if you make a call to MIDAS for a file that was not correctly opened, MIDAS error 23 is returned in the first word of the communication array.

CREATK

The CREATK options ADD, DATA, EXTEND, and MODIFY require exclusive file use. If CREATK cannot get exclusive use, the following message appears:

THIS FILE IS IN USE. AVAILABLE OPTIONS ARE:

CREATK then lists the remaining options.

KBUILD, MPACK, and KIDDEL

If a file is not available for use by KBUILD, MPACK, or KIDDEL, the message:

FILE IN USE

is displayed, and the user is returned to PRIMOS.

MPACK

An abnormal exit from a program will leave the file access controls in an inconsistent state. For example, should a user break out of KBUILD once utility access has been established, no user will be able to access the file through MIDASPLUS, even though no one is actually using the file. Similarly, should a user break out of a program that accessed a file via the MIDAS online routines, no user will be able to access the file through MIDASPLUS or a utility. These situations can be corrected by a new MPACK function called ACCESS.

After the user types MPACK and enters the name of the file to be cleaned up, the prompt:

'MPACK', 'UNLOCK' or 'ACCESS':

appears. The MPACK and UNLOCK options check that no one else is using the file and then continue as they always did. (MPACK is documented in the MIDAS User's Guide (IDR4558). The ACCESS option displays the message:

OK TO RELEASE FILE ACCESS CONTROL?

A response of YES, OK, or AYE causes MPACK to release all access controls on the file and then return to PRIMOS. A response of NO or NAY causes no action; the user is returned to the previous prompt. Any other response elicits the prompt:

PLEASE ANSWER YES OR NO

It is recommended that all use of MPACK be restricted to the System Administrator.

Notes

Breaking out of a MIDAS or MIDASPLUS program will probably leave a file with erroneous access control information. This could be a problem for some applications, such as POWER, in which the user has been told to use BREAK or CTRL-P in certain cases.

Programs using the offline routines PRIBLD, SECBLD, and BILD$R should include alternate returns in the calls. This will ensure that the routines will reset the file access control words in case of errors with sequence flags 0 or 2. (If an error occurs when the sequence flag is 1, the routines will not reset the controls, because it is expected that the user will again call the routine with a sequence flag set equal to 2 to signal the end of the building process.)

If the internal MIDAS locking is disabled (if SHDSEG = .FALSE. in KPARAM.INS.FTN), then MIDAS calls to ADD1$, DELET$, NEXT$, etc. are unable to verify that a MIDAS file has been opened correctly with OPENM$ or NTFYM$. If the user has opened the file correctly, there is no problem. However, if the user has not used OPENM$ or NTFYM$, no file access controls can be checked. Thus, there will be no protection from concurrent use by MIDASPLUS or a utility. (Installations that have modified SHDSEG have usually done so in order to use MIDAS over the network. MIDASPLUS is particularly designed for use on remote files. In most cases, MIDASPLUS (rather than an unshared version of MIDAS) should be used for network applications.

DOCUMENTATION CORRECTIONS

The following corrections apply to the MIDAS User's Guide (IDR4558).

START and Locked Records

On page 7-18, the two paragraphs in the section called START and Locked Records should be replaced by the following text:

If you attempt a START operation on a record that is already locked by another user, MIDAS returns a status code of 0 (successful START). However, if you then attempt to READ that record, you receive a status code of 90, indicating that the record is locked. Your START and READ do not unlock the record for the other user.

CLOSM$

In the example at the bottom of page 6-7, the line:

```
CALL CLOSM$(UNIT,STATUS)
```

should read:

```
CALL CLOSM$(FUNIT,STATUS)
```

for consistency with the rest of the section.

ADD1$

On page 6-21, insert the following entry between the entries for index and bufsiz:

file-no	obsolete, set to 0

FIND$

On page 6-31, insert the following entry between the entries for index and bufsiz:

file-no	obsolete, set to 0

UPDAT$

On page 6-52, in the calling sequence for UPDAT$, the parameter key should be inserted between buffer and array (POLER #47934).

DELET$

On page 6-58, in the calling sequence for DELET$, the parameter key should be inserted between buffer and array.

GDATA$

On page 6-47, in the argument list for GDATA$, the entry for bufsiz should read: "Size of buffer in bytes."

FORTRAN Examples

In the FORTRAN examples throughout Chapter 6, wherever the format (xAy) is used to read a key, it should be replaced by the format (zAy), where z is half of x.

OPENM$

On page 6-6, the following Note should be inserted just after the entry for namlen:

Note

FORTRAN 77 programs will not run if the namlen parameter is supplied as a constant.

SOFTWARE PROBLEMS FIXED

Fixed at Rev. 18.4 and Rev. 19.1

CREATK's MODIFY Option: The MODIFY option of CREATK now correctly modifies synonym (duplicate) support as specified by the user. ASCII key sizes are also changed correctly (POLER #32195).

First Fixed at Rev. 19.1

Extended MIDAS Files: MIDAS is now able to add data records to an empty MIDAS file whose maximum segment size has been extended through CREATK (POLER #27410).

BILD$R: BILD$R now correctly adds duplicate secondary keys (POLER #41858).

KBUILD: KBUILD now correctly adds duplicate keys to secondary indexes that are already populated (POLER #45082).

CREATK Dialog: The CREATK prompt for data record size has been expanded from DATA SIZE = : to DATA SIZE IN WORDS = :. Similarly, the prompt for secondary data size has been expanded from SECONDARY DATA SIZE = : to SECONDARY DATA SIZE IN WORDS = :.

User Breaks: MIDAS no longer hangs all users if the user holding the MIDAS lock breaks out of MIDAS (POLER #37186).

SIZE: The CREATK SIZE option now calculates the amount of space required for a given number of data records without using an overhead of a record per segment (POLER #45492).

KBUILD: When KBUILD was run to add entries to an index that was currently empty but had once been populated, the utility aborted after the IS FILE SORTED? prompt and issued the message "Operation illegal on a directory". This problem has been corrected (POLER #47473).

REMAKE: CREATK and MPACK no longer produce the messages NOT A VALID MIDAS FILE and STOP REMAKE THIS FILE when a valid MIDAS file has been updated through REMAKE.

SOFTWARE PROBLEMS OUTSTANDING

File System Errors

MIDAS leaves a file in an inconsistent state when a disk full error arises on a call to the file system. MIDAS aborts the process.

Upon receiving an ALL FILE UNITS IN USE error on calls to the file system, MIDAS is unable to multiplex its file unit usage. The process is aborted.

FIND$, NEXT$

MIDAS ignores the request to return data shorter than the actual record length in FIND$ and NEXT$ calls. The full record is returned instead.

Direct Access ADD1$

When adding records to a direct access file, ADD1$ sometimes erroneously extends the file allocation.

MIDASPLUS

INTRODUCTION

MIDASPLUS is a new data management product that is fully compatible with MIDAS but offers substantially improved performance.

For full information on MIDASPLUS, refer to Prime Technical Update 98 (PTU98). That document is described in the section called NEW BOOK TITLES in Chapter 1 of this book.

Note

MIDASPLUS users should read the section called OTHER NEW FEATURES under MIDAS, earlier in this chapter.

DOCUMENTATION CORRECTION

On page 98-15 of Prime Technical Update 98 (PTU98), under MIDASPLUS Linkage, '170000 should be changed to '150000.

POWERPLUS

NEW FEATURES

POWER Passwords

At Rev. 19.1, the POWER system passwords have been removed from the PASSWD file in POWER*. This will not affect the PASSWD SYSTEM command. A utility to convert to the new password format is included in POWERPLUS>TOOLS. This utility will be run by the POWERPLUS install file.

PRIVATE PROCEDURE Files

POWER now supports PUBLIC and PRIVATE PROCEDURE files. PRIVATE PROCEDURES are stored in the directory specified during the PROCEDURE CREATE command and are only accessible by the user that created the procedure.

The PROC CREATE Command: The format of the PROC CREATE command has been changed to allow the user to specify a public or private procedure. The new format is:

```
PROC CREATE PUBLIC
```

or

```
PROC CREATE PRIVATE
```

PRIVATE is the default CREATE mode. CREATE PRIVATE asks for the following information:

```
ENTER DIRECTORY (UFD) NAME :
ENTER DIRECTORY PASSWORD   :
ENTER PROCEDURE NAME       :
ENTER PROCEDURE DESCRIPTION:
```

* is acceptable as a valid directory name. CREATE PUBLIC creates procedure files in the POWRCM UFD.

The PROC LIST and PROC LIST ALL Commands: The PROC LIST command lists all procedure files created by the current user. The PROC LIST ALL command lists all public procedure files.

The PROC DELETE Command: The PROC DELETE command allows the user to delete any private procedure file created under his or her login id, or any public procedure file. Procedure files created prior to Rev. 19.1 do not have associated login ids. They are public files and can be deleted by any user.

PROCS Created by Other Commands: The CREATE, FORM, REPORT, and SCREEN commands allow the user to save responses in a procedure file. When the user requests that responses be saved, the following prompts are issued:

```
DO YOU WANT TO CREATE A PRIVATE PROC? :
ENTER DIRECTORY(UFD) NAME            :
ENTER DIRECTORY PASSWORD             :
ENTER PROCEDURE NAME                 :
ENTER PROCEDURE DESCRIPTION          :
```

If the user chooses to create a public procedure, the command omits the prompts for directory name and directory password.

The DISPLAY USING SCREEN Command

The DISPLAY USING SCREEN command displays the first record in a specified set on a specified screen. The format of the command is as follows:

```
DISPLAY [setN] USING screenxx
```

where setN is an optional set name or set number and screenxx is a valid screen name.

When you issue this command, the bottom line of the screen appears as follows:

```
ENTER COMMAND ("H" FOR HELP): RECORD: 1
```

The current record number is displayed after the word RECORD, and the cursor is positioned after the ENTER COMMAND ... prompt. The commands you can choose from at this point are as follows:

Command	Action
H	Display the available commands
END	Clear the screen and return to POWER command level
REDRAW	Redisplay the screen and the current record
NEXT	Display the next record in the set

PREVIOUS	Display the previous record in the set
TOP	Display the first record in the set
BOTTOM	Display the last record in the set
<N>	Display the Nth record in the set
+N or -N	Display the Nth relative record in the set

All commands can be abbreviated to one letter.

The DESTROY Command's -DELETE Option

A -DELETE option has been added to the DESTROY command. The command:

DEST <filename> <password> -DELETE

removes the specified file from the POWER dictionary and also deletes the disk file. If the file was created with * as the directory name, the user must be attached to the directory containing the file in order to delete it.

The Table Skip

When records are added to a file with an interactive add command, all or part of a table field may be blank-filled. To accomplish this, enter $N as the next value in the table. $N causes the add to skip to the next field in the record.

The LIST SYSTEM Command

The LIST SYS or LIST SYSTEM command allows the System Administrator to list all POWER files, both public and private.

Null Commands

POWER no longer displays INVALID COMMAND when the user enters a blank command.

DISPLAY FILLER Fields

The DESCRIPTORS command now displays FILLER fields.

Numeric Field Names

The CREATE and CREATE CHANGE commands now accept the following field type names:

REAL*8	or NUM
REAL	or NUM2
INT	or NUM3
INT*4	or NUM4
DEC	or NUM5
COMP-3	or NUM6

The LIST VALIDATION Command

The LIST VALIDATION command lists all of the validated descriptors and associated validation files for the currently selected file. The format of the command is as follows:

```
LIST VAL
```

or

```
LIST VALIDATION
```

The descriptors and files appear under headers as follows:

```
DESCRIPTOR                              FILE NAME
**********                              *********
```

where DESCRIPTOR is the name of the descriptor being validated and FILE NAME is the name of the file that contains its validation table. If FILE NAME is blank, the descriptor is validated through a range check.

The RANGE Function

The RANGE function displays the following values for a specified descriptor in the current or specified set:

LOWEST VALUE

LOWER QUARTILE

MEDIAN

MEAN

UPPER QUARTILE

HIGHEST VALUE

NUMBER OF ENTRIES

The MEDIAN is defined as the value at position (N+1)/2.
The LOWER QUARTILE is the value at position (N+2)/4.
The UPPER QUARTILE is the value at position (3N+2)/4.
The MEAN value is the average of the N set elements.
The number of entries is N, the number of elements in the set.

RANGE values can be obtained in two ways: by invoking the RANGE function for a particular descriptor, and through REPORTS.

The Interactive RANGE Function: To use this command, you must select a file and create a set from it. The format of the command is as follows:

RANGE [set name] descriptor

This command displays the following for the specified descriptor:

RANGE OF <descriptor> IS
****

MIN	LQ	MEDIAN	MEAN	UQ	MAX	NO

All values except number of entries are printed to four decimal places. RANGE can only be used on numeric descriptors.

Invoking RANGE From a Report: The REPORT CREATE dialogue asks "DO YOU WANT AVERAGE OR RANGE" for any numeric descriptor. You may respond:

YES	(AVERAGE ONLY)
BOTH	(AVERAGE AND RANGE)
RANGE	(RANGE ONLY)
NO	(NO AVERAGE OR RANGE)

If you select RANGE, the RANGE values are displayed at the end of the REPORT.

PST 100 Terminal Support

POWER now supports the PST 100 terminal. The PST 100 must be index 9 in the POWER terminal table; therefore, TERM** and TERM## should be copied from POWERPLUS>TOOLS.

DOCUMENTATION CORRECTIONS

The following corrections apply to the Prime/POWER Guide (PDR3709).

CREATE Dialog

The CREATE prompt ENTER LENGTH IN CHARACTERS has been changed to ENTER LENGTH IN CHARACTERS OR FIELD TYPE. This change should be noted on pages 4-11 and 17-10.

TABLE Creation

The extended options dialog for creating a TABLE has been changed. The following example, patterned after the example on pages 11-4 and 11-5, illustrates the new dialog.

```
ENTER DESCRIPTOR NAME: SURFACE
ENTER LENGTH IN CHARACTERS OR TYPE: TABLE
*** ENTER TABLE INFORMATION ***
ENTER TABLE SIZE IN CHARACTERS:  35
ENTER TABLE STARTING POSITION:   25
*** ENTER DESCRIPTOR INFORMATION ***
ENTER DESCRIPTOR POSITION IN TABLE: 1
ENTER MAXIMUM NUMBER OF ENTRIES:  3
ENTER LENGTH IN CHARACTERS OR TYPE:  10
ARE DUPLICATE VALUES PERMITTED:  YES
ENTER SECURITY LEVEL: 0
```

Overlaying Files

When overlaying a file with keywords, you should always define indexes 15-17 as BIT strings. Page B-2 currently states that indexes 15-17 should be defined as ASCII (POLER #41859).

Adding Keywords

It is not possible to add keywords once text has been initially created. Later additions of keywords must be made by editing the keywords (as opposed to editing the text). All references to adding keywords on page 5-4 should be removed (POLER #45149).

SOFTWARE PROBLEMS FIXED

Fixed at Rev. 18.4 and Rev. 19.1

Procedures: Variable names in procedures are now checked for valid characters. (The same restrictions apply as in the case of descriptor names.) Procedures with variables for UFD, PASSWORD, and FILENAME for CREATE commands now substitute variables correctly (POLER #27517).

* is now accepted in procedures to indicate the current UFD for the CREATE command (POLERs #37794, 35908, 32653).

CREATE: Field names defined by CREATE ADD sequences as synonyms for indexed fields now function properly in FINDs (POLERs #45074, 43565).

Head of Form: POWER was failing to generate a head of form on the first page of a printout. This problem has been corrected (POLERs #41486, 31134, 40125).

Linked Fields: Data in linked fields is now displayed correctly (POLER #31542).

When a report was run on linked fields, blank lines were inserted for every owner record, whether or not member records were present. This problem has been corrected (POLER #34642).

A DISPLAY with six linked fields and left side headers now works correctly (POLER #35655).

Computed Fields: Computed fields (N1...Nx) now compare correctly to decimal file values (POLERs #29583, 46850).

Reports: The report writer now prints descriptors that are two links away from the selected file correctly and in entirety (POLERs #36575, 31342).

The existence of a mixture of security levels in a file no longer prevents the calculation of variables in a report (POLERs #27491, 32049).

Errors in display condition are now recognized during the report create dialog (POLER #34630).

Linked file descriptors are now used correctly during report creation (POLERs #40130, 48548).

DISPLAY: DISPLAY USING FORM no longer aborts with an illegal access violation (POLER #24267).

Fixed at Rev. 19.0 and Rev. 19.1

Combining Sets: The "Warning - line truncated" message is no longer displayed when combining sets (POLER #35382).

The correct number of entries is now generated on a combine (POLER #35383).

Compute With No Sets: A Fortran I/O error no longer occurs as a result of a compute with no sets available (POLER #11983).

Find on Long Integer: A FIND on a long integer no longer gives incorrect results (POLERs #11982, 20786).

Change Descriptor Name: A change descriptor name followed by a change description no longer corrupts the filename (POLER #82294).

Batch Add File: When a character descriptor is missing from a Batch Add file, POWER no longer inserts the value from the previously added record (POLER #36763).

Change Descriptor Error: Occurance of an error in change descriptor no longer prevents previous changes from being made (POLER #82285).

Comment Line: After processing a comment line in a procedure, POWER will now issue a prompt (POLER #25397).

Reports: In reports suppressing duplicate numeric variables, the output of variables is now correct (POLER #32173).

Zero Title Lines: Specifying no title lines no longer ruins the first line of the heading (POLER #32818).

List Display: Member linked fields are no longer offset in a list display (POLER #82023).

Validate: Validate now works properly when using more than 51 numeric fields (POLER #36521).

Modifying Reports: Reports with no title lines and/or numeric descriptors can now be modified (POLER #37513).

Find on Character Field: An inequal FIND on a character field no longer returns "equal" (POLER #82202).

$FILL: When $FILL is entered after an erroneous entry in validation mode, an incorrect field is no longer added (POLER #81600).

Blank Date: The blank date indicator of 9999 can now be suppressed in a report, although 9999 will still appear in a DISPLAY (POLER #36242).

Compute in Date Field: The Compute verb no longer inserts a month in a date field containing only a year (POLER #34346).

Spool Files: POWER no longer creates null or incorrect spool names for spooled files (POLER #82128).

Report Create: The user no longer has to know how many characters a variable used in the heading will have at Report Create time (POLER #35804).

First Fixed at Rev. 19.1

Concurrency: When one user creates a template, other POWER users are no longer locked out of SELECTs (POLER #41488).

Page Numbers: When the page number at bottom option was requested in a report, POWER was not printing the page number on the last page of the report. This problem has been corrected (POLER #42800).

Date Fields: With an indexed data field using < OR >, a partial date produced erroneous results depending on the last full value used for an indexed date find. This problem has been corrected (POLERs #45070, 45148).

An erroneous date field was displayed when REPORT FORMAT was used (POLER #47247).

Table Descriptor Prompts: Report Create now correctly prompts for all table descriptors after a NO in answer to the prompt DO YOU WANT ALL TABLE ITEMS DISPLAYED (POLER #37327).

CHANGE Command: The word TO is now optional in the CHANGE command (POLER #42799).

Audit File: A problem concerning overlays of the audit file in POWER* has been corrected. This problem caused programs to abort with a BEGINNING OF FILE error on logical unit 14 (POLER #42797).

REPORT CHANGE: REPORT CHANGE was printing old title lines incorrectly after the number of title lines was increased. This problem has been corrected (POLER #41912).

Descriptor Names: POWER did not allow descriptors to have the following names: E, E1, E2, ... D, D1, D2 ... (The names EE and DD were allowed.) This problem has been corrected (POLER #48172).

FIND KEY: FIND KEY = 'A' no longer results in a record being selected twice (POLER #48171).

Spurious $: The sums of fields of picture 9(8)V99 or greater no longer are displayed with spurious $s (POLER #41609).

Long Keywords: When a keyword of more than 20 characters is entered, POWER no longer makes one keyword from the first 20 characters and another keyword from the remaining characters (POLER #41850).

Combining Large Sets: The combination of two large sets (one with more than 3000 entries) no longer produces incorrect results (POLER #48162).

The combination of certain large sets produced a COMBINE FORMAT ERROR message, but created a new current set that may not have been complete. This problem has been corrected (POLER #44688).

SORT: A SORT on a named set of 256 entries made from FINDs on sets of 20,000 entries produced a RECORD LENGTH INDICATED IS < 1 error message. This problem has been corrected (POLERs #44689, 41910).

Passwords: When a password was changed for a directory containing POWER files, POWER produced the message DISK NOT ON SYSTEM when trying to access those files. This problem has been corrected (POLER #40097).

Column 1 Missing: POWER no longer omits the first character of a title that begins in column 1 (POLER #45507).

FIND GT: FIND GT now correctly returns all of the expected records (POLER #47221).

Long Descriptors: When dumped or displayed, a descriptor more than 60 characters long was truncated. This problem has been corrected (POLER #42801).

Vertical Display: Vertical displays with more than one option now work correctly (POLER #47470).

Vertical display now correctly uses formats set during heading create.

Title Changes: Report titles of more than 68 characters can now be changed correctly (POLER #48037).

FIND Inconsistencies: Equivalent FINDs no longer produce different results (POLER #27497).

Table Item Display: All table items are now correctly displayed by the DISPLAY and PRINT commands (POLER #44336).

MIDAS: An overlaid MIDAS file with bit string and ASCII descriptors and keywords failed to perform correctly on a FIND KEYWORD that followed a FIND of an ASCII descriptor. This problem has been corrected (POLER #41852).

Saved Sets: When a user had a full complement of sets, POWER failed to find entries and became confused about the number of sets saved. This problem has been corrected (POLER #47477).

File Validation: If a file to be validated was linked to two validation files, and if the same descriptor names were used in the validation files and in the file to be validated, then the second validation failed. This problem has been corrected.

Command Priority: Individual command priority did not always function if the user name contained a period. This problem has been corrected.

Error Messages: Error messages are now correctly cleared during a CHANGE USING SCREEN function.

An F$IO error occurred during display of a NUM5 descriptor with a format of more than 24 characters. This problem has been corrected.

TERM USER: TERM USER did not allow a user number greater than 60. The limit has been changed to 128.

CHAPTER 5

FORMS AND FED

FED

NEW FEATURES

PST 100 Terminal Support

FED can now be run on the PST 100 terminal as well as the PT45. You can design forms for use on any block mode terminal supported by FORMS by running FED on a PST 100. For more information on the PST 100, refer to the section called FORMS, later in this chapter.

Special Symbols

On screens that use special FED symbols, the relevant symbols are now listed off-screen, adjacent to the list of available function keys.

The SKELETON Option

The SKELETON option has been moved from inside CREATE to a separate function at the same level as CREATE and MODIFY. Moreover, skeletons can now be created from FDL source as well as from scratch or other skeletons.

The MODIFY Option

The MODIFY option now contains five new commands:

Command	Action
DISPLAY	Displays the form as invoked.
IDENTIFY	Highlights and displays the name of a field.
COLLECT	Collects fields together temporarily.
SDETAILS	Identifies and renames substreams.
RETYPE	Amends the picture of a field.

In addition, the ADD command has been enhanced to allow easier addition of fields, as well as the addition of more than one field at once.

The DETAILS Command

The DETAILS command has been extended to allow the detailing of a set of fields together. CREATE and MODIFY allow fields to be selected for detailing on their current types and attributes. MODIFY also allows the temporary collection to be detailed as one entity.

Field Verification

The verification of fields in the workline now includes the fields' names and types.

New Function Key

A new function key has been added to FED. By pressing the LOG key, you can spool a simplistic copy of the FED screen to the line printer.

FORMS

NEW FEATURES

PST 100 Terminal Support

Support for the new PST 100 terminal has been added to Rev. 19.1 FORMS. Form definitions should include a "device PST100" ... "end device" section if they are to run on a PST 100. The TCB command required in FAP is TCB * PST100.

The PST 100's screen dimensions are 24 lines by 80 columns (1920 characters). However, the terminal has a display memory of 48 lines by 80 columns (3840 characters), allowing the definition of 2-page forms. All character positions are available for use by the form definition, as FORMS prompts are displayed at the right-hand end of the terminal status line. There are no device limits on the number of fields per line or per form. Fields may be adjacent, and there are no limitations on where they may start and end.

Thirty-two function keys are available for use on the PST 100:

F1-8 (1 - 8)
SHIFT F1-8 (9 - 16)
CTRL F1-8 (17-24)
CTRL SHIFT F1-8 (25 - 32)

The numeric keypad is not used as a programmable function key pad.

The PST 100's HELP key can be used as a "pseudo" function key. It is treated like any other function key by FORMS, and it returns the value 99 as its key number. SHIFT HELP, CTRL HELP, and CTRL-SHIFT HELP all return the same number.

When a form is initially invoked, the screen remains blank until the form has been completely output.

All FDL field attribute parameters apply to the PST 100. For completeness, a description of their set states is provided below:

- ENABLE Parameter

  The ENABLE parameter causes the field to be write-enabled on the terminal.

- BLINK Parameter

  The BLINK parameter causes the field to blink on the terminal. Note that it is the text displayed in the field that blinks.

- REVERSE VIDEO Parameter

  The REVERSE VIDEO parameter causes the field to be displayed in reverse video on the terminal.

- NODISPLAY Parameter

  The NODISPLAY parameter prevents the field's text from being displayed on the terminal. Note that NODISPLAY when used in conjunction with the REVERSE VIDEO parameter has a different effect depending on whether or not the field is enabled. An enabled field appears in reverse video; a protected field appears in normal video.

- UNDERSCORE Parameter

  The UNDERSCORE parameter causes the field to be underscored when displayed on the terminal.

- LOW-INTENSITY Parameter

  The LOW-INTENSITY parameter causes a field to be displayed in low intensity on the terminal.

- HIGH-INTENSITY Paramter

  The HIGH-INTENSITY parameter causes a field to be displayed in high (or full) intensity on the terminal. If both high and low intensity are omitted, the default of low intensity for protected fields and high intensity for enabled fields is applied.

- TOTAL-FILL Parameter

  The TOTAL-FILL parameter causes the terminal to require that the field be totally entered into by the user. That is, every character position must be modified and must contain nonblank data. This requirement is checked when the terminal user presses the ENTER key. Should the test fail, the terminal places the cursor at the first field that failed a fill test, displays a warning message in the status line, rings the bell, and "soft locks" the keyboard. The user must then press the RESET key to continue. There are no restrictions on tabbing past the field if the fill condition has not been met. Note that fill tests are not carried out when a function key is pressed.

- PARTIAL-FILL Parameter

  The PARTIAL-FILL parameter causes the terminal to require that at least one character position be entered into by the user. See the description on TOTAL-FILL for an explanation of how the terminal checks this.

- GRAPHICS Parameter

  The GRAPHICS parameter causes the terminal to treat the field's data as line-drawing graphic entities. It should only be used on protected fields.

- NUMERIC-ONLY Parameter

  The NUMERIC-ONLY parameter causes the field to accept only numeric data (0-9, +, -, comma, . , space) from the keyboard. An attempt to enter nonnumeric data causes the terminal to display a warning message on the status line, ring the bell, and soft lock the terminal.

- ALPHA-ONLY Parameter

  The ALPHA-ONLY parameter causes the field to accept only alphabetic data (space, +, -, comma, ', all uppercase and lowercase letters) from the keyboard. An attempt to enter nonalphabetic data will cause the terminal to display a warning message on the status line, ring the bell, and soft lock the terminal.

- ALPHANUMERIC Parameter

  This parameter is the default. Despite its name, it allows entry of any character into the field.

The following messages may occur. They are displayed at the right-hand end of the status line.

- ENTER

  Operator input is required.

- DATA ERROR

  The data in the field to which the cursor is positioned does not conform to any of the validation criteria specified in the form definition. In addition to displaying this message, FORMS rings the terminal bell and soft locks the keyboard.

- SIZE ERROR

  The data transmitted from terminal to CPU does not match what was required. This usually indicates that a transmission error has occurred. In addition to displaying this message, FORMS rings the terminal bell and soft locks the keyboard.

- KEY DISABLED

  The terminal operator either pressed a key that had been deliberately disabled by FORMS, or pressed a function key when function keys were disabled. In addition to displaying this message, FORMS rings the terminal bell and soft locks the keyboard.

- *HELP

  Indicates that the help key has been pressed.

- *F#

  Indicates that function key # has been pressed.

- PRINTING...

  Indicates that a ##PRINT LOCAL command is taking place.

No special switch settings are required on the PST 100 for FORMS applications.

TCB Entries

Terminals that have no TCB entry are now treated as if they have an entry of INQUIRE.

Users whose terminals have a TCB entry of INQUIRE or have no TCB entry at all can now query the valid list of terminal names by responding with INQUIRE when asked to enter their terminal type.

FORMS no longer looks for the TCB* directory at runtime by default. If you require this feature, then the FORMS package should be rebuilt from the supervisor terminal as follows:

```
> resume formssrc>forms.build -rebuild -tcb
```

It will then be necessary to reshare the VFORMS library.

Reading and Writing Extra Data

At Rev. 19.1, data can be written to a field that has a longer stream section length than that specified in the format section. Although the extra data does not appear on the screen, FORMS stores it and returns it unchanged when the field is read. Prior to Rev. 19.1, the extra data was converted to spaces (and therefore lost).

SOFTWARE PROBLEMS FIXED

Fixed at Rev. 18.4 and Rev. 19.1

Field Attributes: Changing a field from protected to unprotected at runtime now works correctly.

NODISPLAY INPUT fields may now be used.

Function Keys: Function keys 11 and 12 now work correctly. The CLEAR key now returns number 14 when function keys are enabled. The ##FKEYS ON directive now works correctly.

3271 Support: FORMS can now be run on plug-compatible 3271 units.

Note

The default build of FORMS now uses the _DD versions of FM$TCB. These versions cause FORMS to ignore the TCB* directory and go directly to FORMS* instead. To make use of TCB*, you must change the files FAP>C_SUBS, RUN>C_FMV, and RUN>C_FMR to use the version of FM$TCB without the _DD suffix.

Fixed at Rev. 19.0 and Rev. 19.1

Delete Character Function: The D-CHAR key on a Prime PT45 now works correctly (POLER #33295).

Insert Character Function: The I-CHAR is now correctly switched off so that the screen is not corrupted when output is sent to the terminal (POLER #36359).

First Fixed at Rev. 19.1

Column 80: Enabled fields can now end in column 79, with the logical attribute in column 80. Visual attributes are still output in column 1 of the next line.

##NODISPLAY: ##NODISPLAY now works correctly (POLER #21621).

SOFTWARE PROBLEMS OUTSTANDING

Vistar 3

FORMS does not work on a Vistar 3 (POLER #27842).

Rebuilds

FORMS cannot be rebuilt (POLER #32416).

CHAPTER 6

COMMUNICATIONS

DPTX

SOFTWARE PROBLEMS FIXED

Fixed at Rev. 18.4 and Rev. 19.1

PA Keys: DPTX/TCF would not accept any PA key as the exit indicator. This problem has been corrected (POLER #42967).

OWLDSC Command: The OWLDSC command no longer causes an "Invalid Screen Address" error on a call to BD$OUT (POLER #46862).

Fixed at Rev. 19.0 and Rev. 19.1

Bad Dim Data Error: The receiver is now reenabled on a "Bad Dim Data" error.

No Host Response: If the host does not respond with an ACK or a NAK to text messages sent by the emulator, ENQ messages are now sent.

Remote Timeout: When the remote end times out, receives will no longer be retried, so that the protocol handler will immediately recognize the timeout condition.

Logged-out Lines: Logged-out lines are now correctly recognized as being logged out.

SYNC in Text Field: The emulator will now handle a data stream that contains a SYNC character embedded in a text field (POLER #43856).

Short Read AID: The emulator will now send a completed read data stream when a short read AID is outstanding and a read command is issued (POLER #32431).

Force-logout on Queue Full: The emulator will now allow a force-logout to occur when a queue full condition exists.

Keyboard Unlocked: The keyboard is now unlocked when a NAK is received while the emulator is in Response Outstanding mode (POLER #23272).

Powered Off Device: The Traffic Manager will no longer continue selecting a device that is powered off. It will now wait for a DE status to be received before selecting it again.

Nonconfigured Device: If the Traffic Manager receives an AID for a nonconfigured device, it will now discard it rather than marking the control unit down.

Attribute Array Overflow: The attribute array can no longer overflow. Previously, an Illegal Segment Number error occurred when it overflowed (POLER #28601).

FTS

SOFTWARE PROBLEMS FIXED

First Fixed at Rev. 19.1

Uppercase Display: The FTR -LOG pathname is now uppercased, so passwords quoted in lowercase on the command line will match the directory password.

The FTR -NAME request name is now uppercased to be consistent with the default request name, which is always in uppercase.

Recording Passwords: Log file entries no longer record directory passwords.

The FTR -DISPLAY command now displays passwords in source, destination, and log file pathnames only when the request owner invokes FTR.

The FTSQ*>YTSMAN.COMO file no longer records server passwords that have been entered correctly. Incorrect server passwords are still recorded. For example:

```
User name is YTSMAN                          . User number is  40
Port 256 assigned. - Operation complete
Port 252 assigned. - Operation complete

Call received
Called address is ftsx
circuit passed off ok

Call received
Called address is ftsx
Circuit   4 cleared by YTSMAN - Incorrect server password - passw
Circuit   4 cleared by network.
```

FTS Rights: The FTOP and FTGEN commands can now be invoked only by user SYSTEM. In particular, these commands can no longer be invoked by users FTP, FTS, or YTSMAN. Any user, other than SYSTEM, who attempts to run FTOP or FTGEN receives the following messages:

```
OK, ftop -lsvs
[FTOP rev 1.1]
You do not have operator privileges.(ftop)
ER! ftgen
[FTGEN rev 1.1]
You do not have operator privileges.(ftgen)
ER!
```

Furthermore, users FTP, FTS, and YTSMAN now have the same rights as ordinary users with regard to the FTR command. SYSTEM is now the only user who may control other users' requests by means of FTR.

YTSMAN Logout: YTSMAN now has an on-unit for the LOGOUT$ condition. This on-unit clears any current incoming remote requests and then causes YTSMAN to log out.

Prior to Rev. 19.1, the FTSQ*>YTSMAN.COMO file recorded a forced logout as follows:

```
***From PRIMOS: forced logout.

YTSMAN (user 40) logged out Thursday, 16 Sep 82 19:59:32.
Time used: 11h 14m connect, 00m 10s CPU, 02m 30s I/O.
```

At Rev. 19.1, the forced logout is recorded as follows:

```
LOGOUT$ condition detected.

YTSMAN (user 40) logged out Wednesday, 03 Nov 82 12:00:56.
Time used: 25h 43m connect, 00m 05s CPU, 00m 04s I/O.
```

In addition, the following message is displayed at the supervisor terminal:

```
*** YTSMAN (user 40 on SITE5) at 12:00
YTSMAN closing down - LOGOUT$ condition detected.
```

FTP Logout: FTP now has an on-unit for the LOGOUT$ condition. This on-unit aborts active transfers and cleans up before logging a server out. FTS will automatically retry the aborted transfers.

Note

Forced logout is not the recommended method of closing down an FTS server. The FTOP -STOP_SRVR or FTOP -ABND_SRVR command should be used instead.

The following is a sample extract from a server log file that records a forced logout. (TRACE level logging is used.)

```
11.59.09: [1.1] Request TEST_FILE (7182963) started Wednesday,
     November 3, 1982
11.59.10: [1.1] Submitting user is TEST
11.59.10: [1.2] Request submitted on 82-11-03.11:58:36
11.59.10: [1.1] Local file is <TSTDSK>TEST>TEST_FILE
11.59.10: [1.2] Remote file is <TSTCPY>TEST>TEMP>TEST_FILE
11.59.10: [1.2] Local address is SITE5
```

```
11.59.10: [1.2] Remote address is SITE6
11.59.10: [1.2] File has been requested to be sent.
11.59.10: [1.3] Size of file is      118106 bytes.
11.59.10: [1.4] Opening: #7182963.T  Operation ok.
11.59.10: [1.4] Connect message sent, vcid =  3 - Operation complete
11.59.12: [1.4] Accept message received.
11.59.13: [1.4] SFT command sent.
11.59.13: [1.4] RPOS command received.
11.59.13: [1.4] Number of parameters on RPOS command is  0
11.59.13: [1.4] GO command sent.
11.59.13: [1.4] Sending SS command.
12.00.40: [1.4] X.25 clear sent, vcid =  3 - Node going out of service.
12.00.40: [1.1] RESULT: Transfer aborted : Server FTS1 closing down -
     LOGOUT$ condition detected.
12.00.41: [1.1] The transfer will be retried later.
12.00.41: [1.1] Request TEST_FILE (7182963) finished.
12.00.41: FTS Server - FTS1 closed down on Wednesday, November 3, 1982
12.00.41: LOGOUT$ condition detected.
```

In addition, the following message would be displayed at the supervisor terminal:

```
*** FTP (user 50 on SITE5) at 12:00
12.00.41: FTS Server - FTS1 closed down on Wednesday, November 3, 1982

*** FTP (user 50 on SITE5) at 12:00
12.00.41: LOGOUT$ condition detected.
```

Default Port: The default PRIMENET listen port for an FTS server is now Port 1. The previous default was Port 0. This change is demonstrated in the following example:

```
OK, ftgen
[FTGEN rev 1.1]
FTS STATUS
Server directory is ftsq*.
System issue number is  17.
Number of queues configured is  1.
Number of servers configured is  2.
Number of sites configured is  6.
ftgen> add_server new_server
server: 1
Server          : new_server
Server status   : Running.
Password        :
Queue           :
Log             : -OFF
Message_level   : normal
Program         : ftp.seg
Port            :     1
Priority        :     1
```

```
Timeslice       :     20
server: queue fts$2
server: log new_server.log
server: msgl trace
server: password safety
server: l
Server          : new_server
Server status   : Running.
Password        : safety
Queue           : fts$2
Log             : new_server.log
Message_level   : trace
Program         : ftp.seg
Port            :      1
Priority        :      1
Timeslice       :     20
server: file
Server added.
ftgen> q
OK,
```

Rev. Stamp: The Rev. stamp for all FTS programs has been changed from 1.0 to 1.1. For example:

```
[FTR rev 1.1]
[FTOP rev 1.1]
[FTGEN rev 1.1]
[FTP rev 1.1]
[YTSMAN rev 1.1]
```

Protocol Implementation Level: In the Protocol Implementation Level message, which is logged in the server log file at TRACE message level, the Protocol Implementation Level has been changed from 00 to 01. For example:

```
15.38.16: [1.4] Protocol Implementation Level is 01
```

PRIMENET

SOFTWARE PROBLEMS FIXED

Fixed at Rev. 18.4 and Rev. 19.1

Node-node Password: The Node-node password is now tested correctly (POLER #45343).

A BAD PASSWORD error was sometimes produced when a process attached to a remote disk and then ran an external command. This problem has been corrected (POLERs #43290, 36726).

Remote File Access: Opening a remote como file and logging out without closing it resulted in an error that could "hang" the user terminal. This problem has been corrected (POLER #45343).

A problem in the initialization of network slaves has been corrected (POLER #45392).

Phantom jobs doing I/O over the network were not always correctly logged out by a forced logout. This problem has been corrected (POLER #44296).

Slaves are no longer able to log out other slaves (or any other processes). A slave can only either log itself out or be logged out from the supervisor terminal (POLER #45340).

Fixed at Rev. 19.0 and Rev. 19.1

Protocol Errors: Several problems in NETLINK that were caused by protocol errors have been fixed (POLER #32437).

32K-byte Files: NETLINK can now handle files longer than 32K bytes.

FAM SGDR$$: All keys to SGDR$$ now work through FAM II.

Slaves Run After Setting Time: Slaves are no longer allowed to run until the first SETIME command is issued from the supervisor terminal.

ADDISK -RENAME: The ADDISK -RENAME command, issued from the supervisor terminal, no longer crashes the system.

Error Codes: Error codes are now more reliably reported (via FAM II) by file system subroutine calls that occur on a different node.

Remote Command Output Files: Remote command output files no longer cause a user to hang if logout happens before a COMO -END is issued.

New Network Logging Events: When the catchall error E$NETE is returned by FAM II as a result of accessing a resource on a remote system, additional information may be recorded in the network event logging file. Network event logging files now reside in PRIMENET*. There are five new events that can be recorded in the network event log file as a result of a software-detected network error. An E$NETE error could also have caused one of these new events to be recorded. The LOGPRT command may be used to peruse event log files. For a complete listing of network event messages, refer to the System Operator's Guide (DOC5038-190).

Type-ahead in NETLINK: When a call is pending as a result of a C or NC command to NETLINK, and the "sysnam Connected" message has not yet been output, type-ahead is now accepted. To escape to command mode, use the break or control-P key.

Reverse Charge Flag: The reverse charge flag for open virtual circuits is now correct (POLER #37575).

First Fixed at Rev. 19.1

NETLINK: NETLINK was displaying spurious characters as part of the "disconnected" message. This problem has been corrected.

RJE

DOCUMENTATION CORRECTION

On page A-4 of the RJE User's Guide (IDR4036), the paragraph beginning "IF YOU ARE RUNNING..." should read as follows:

> IF YOU ARE RUNNING FROM THE SEND UTILITY AND THIS ERROR OCCURRED AS THE RESULT OF A 'TOSITE' COMMAND THEN IT IS PROBABLE THAT THIS SITE HAS NOT YET BEEN DEFINED BY THE WORKSTATION. TYPE 'STATUS' (WITHIN THE WORKSTATION) AND IF THE SITE REQUIRED IS NOT PRESENT THEN ISSUE THE 'TOSITE' COMMAND FROM THE WORKSTATION.

SOFTWARE PROBLEMS FIXED

Fixed at Rev. 19.0 and Rev. 19.1

Cardspool: CARDSPOOL can now handle lowercase characters correctly.

1004: Decompression of print and punch files is now handled correctly (POLER #32662).

GRTS: Mixed media receive files are now processed correctly (POLER #23814).

HASP: HASP can now be run at full 512 byte block size (POLER #21681).

The HASP symbiont no longer aborts with an error message specifying error code 26 (POLER #23223).

One-block files will no longer be incorrectly deleted (POLER #29181).

The error message "E005: file in use" will no longer appear when recovering the line (POLER #31744).

Receiving a punch file with the punch binary option is now handled correctly (POLER #31855).

The switching of VFC files is now correctly tracked between untranslated files (POLER #37438).

Operator messages are no longer being ignored when T files are in the QHASP UFD (POLER #37937).

The error message "Bad T File" will no longer be seen on concat files, as the logic that calculated the length of a concat file now works (POLER #40046).

A retry count has been added. This count will cause a message to be displayed if the count is exceeded without getting a reply from the host. This will allow the user to know if the host has disconnected (POLER #37595).

EMX80: Problems involving occasionally losing blocks have been fixed (POLER #14804).

Component selection in ASCII files is now correctly handled (POLERs #23735, 45553).

The emulator no longer aborts transmission on receiving EOT (POLER #14206).

Data is no longer lost if a line is disabled in midfile (POLER #22404).

SOFTWARE PROBLEMS OUTSTANDING

EM1004

On delsite, untranslated receive files cause a receive file pointer error (POLER #25127).

Timeouts occur when running at 1200 baud (POLER #81776).

Note

Users who run RJE on a Prime 2250 may find that RJE sometimes unexpectedly disconnects a line (but continues running). To continue using the disconnected line, warm start the system.

Users of RJE on a Prime 2250 may also receive intermittent error messages reporting access violations or illegal segment numbers. Should these errors occur, reinitialize RJE.

CHAPTER 7

SUPPORT OF NEW HARDWARE

Cartridge Tape Drive Support

INTRODUCTION

At Rev. 19.1, PRIMOS supports the cartridge tape drive of the Prime 2250. Except for one new option, you can assign a cartridge tape drive as you would any other Prime magnetic tape drive. For information on the ASSIGN command, refer to the System Operator's Guide (DOC5038-190). The ASSIGN command's new -RETENSION option is available for use with cartridge tape drives only.

THE -RETENSION OPTION

The -RETENSION option is used to ensure even tension in the tape cartridge. When you specify this option, ASSIGN fast-forwards the tape to end-of-tape and then rewinds to beginning-of-tape. In the process, tape-to-head pressure is stabilized and the tape is evenly stacked on the reels. This operation takes approximately two to three minutes with a 4800-foot tape cartridge.

You should specify the -RETENSION option in any of the following cases:

- The tape cartridge is new.
- An excessive number of errors occur when the cartridge is used. (In this case, you may also need to clean the tape head and transport.)
- You suspect that the cartridge has been subject to a physical shock, or has been exposed to temperatures outside the recommended range. The recommended range of temperatures is 41 - 113 degrees F (5 - 45 degrees C).
- You suspect that the cartridge has been exposed to humidity outside the recommended range of 20% to 90% noncondensing.
- The cartridge has been improperly stored. (Cartridges should be stored on edge, that is, with the metal base vertical. Otherwise, the tape may begin to drop off the open reel tape hubs.)

If any of the above is true, load the cartridge in the drive and assign the drive using the -RETENSION option before using the cartridge in any other way.

THE T$MT SUBROUTINE

The T$MT subroutine, which is described in the Subroutines Reference Guide (DOC3621-190), is used in manipulating magnetic tapes. One of the arguments in a call to T$MT is a numerical instruction that specifies what the subroutine should do. The following instructions are available for use with the cartridge tape drive:

Octal	Hexadecimal	Meaning
000040	0020	Rewind to BOT.
022100	2440	Backspace one file mark.
062100	6440	Backspace one record.
022220	2490	Write file mark.
062200	6480	Forward one record.
022200	2480	Forward one file mark.
100000	8000	Select drive and return status.
140000	C000	Return device id: Cartridge = 000113 octal (version 5 controller).
042220	4490	Write record, one character per record.
042620	4590	Write record, two characters per record.
042200	4480	Read record, one character per record.
042600	4580	Read record, two characters per record.
100220	8090	Retension tape (Cartridge -- version 5 controller -- only).
140000	C000	Return controller id.

ICS1 Controller Support

INTRODUCTION

The ICS1 is a new intelligent communications controller that has eight asynchronous lines and one synchronous line. At Rev. 19.1, ICS1 support is limited to:

- The asynchronous lines
- The synchronous line, for use by RJE and the T$SLC0 subroutine only. (See the section called SYNCHRONOUS LINE USAGE, below.)

Note

For character-synchronous protocols at Rev. 19.1, ICS1 synchronous line speeds cannot exceed 4800 baud.

PRIMOS COLD START

Line Speed Selection

Lines that connect to an ICS1 controller can run at the following speeds: 50, 75, 110, 150, 200, 300, 600, 1200, 1800, 2400, 3600, 4800, 7200, 9600, and 19200 bps. On non-ICS1 lines, some of these speeds can only be assigned by hardware jumpers. However, for ICS1 lines, any of these 15 speeds can now be selected by means of software alone. To accomplish this, a new configuration directive, ICS JUMPER, has been introduced at Rev. 19.1.

The format of the ICS JUMPER directive is:

```
ICS JUMPER a b c
```

where a, b, and c are line speeds in octal. These three speeds should be chosen from the list of 15 speeds above. They should be speeds that you will want to assign to your ICS1 lines.

Once these speeds are listed in the CONFIG file, you can refer to them in subsequent AMLC commands that start up ICS1 lines. To start up an ICS1 line at speed a, issue the AMLC command and specify a configuration of 2513. Use a configuration of 2613 to select speed b, and a configuration of 2713 to select speed c. For general information on the AMLC command, refer to the System Administrator's Guide (DOC5037-190).

If you do not include the ICS JUMPER directive in your CONFIG file, then AMLC configurations of 2513, 2613, and 2713 will start up ICS1 lines at speeds of 75 bps, 150 bps, and 1800 bps, respectively.

AMLCLK Restrictions

If you set up a line by means of the AMLC command with a configuration of 2413, the line takes on the speed of the system's internal AMLC clock. This clock rate, which is set by the AMLCLK configuration directive, may be anywhere in the range from '35 to '45400 (decimal 29 to 19200) bps. However, ICS1 lines can only run at the 15 speeds listed in the previous section. Thus, you should only start up an ICS1 line with a configuration of 2413 if the clock rate is one of those 15 speeds. If the clock rate is not one of the ICS1 speeds, an error message results and line speed is not changed.

Note

If the AMLCLK directive was not included in the CONFIG file, then a configuration of 2413 sets line speed to 9600 bps.

For more information on AMLCLK, refer to the System Administrator's Guide (DOC5037-190).

Downline Load File

If your system uses an ICS1 controller, you should be sure that the file ICS1.DL is located in the top-level directory DOWN_LINE_LOAD* on the command partition. This file contains code and data that are loaded into the ICS1 at cold start.

PRIMOS WARM START

If your system uses ICS1 controllers, warm starting PRIMOS will take significantly longer than without the ICS1. The extra time is required for the downline load of software to the controller. The message:

```
SYSTEM WARM STARTING, PLEASE WAIT
```

is displated at the supervisor terminal when warm start begins. When warm start is complete, the message:

```
*** WARM START ***
```

is displayed, as usual.

ASYNCHRONOUS LOGICAL LINE NUMBER ASSIGNMENT

In a system that includes both AMLC and ICS1 controllers, logical line numbers are assigned beginning with the AMLCs. When all AMLC lines have been allocated, ICS1 line numbers are assigned, starting at the next 16 ('20) line boundary. For example, if a system had two 16-line AMLCs and three ICS1s, line numbers would be assigned as follows:

AMLC	lines	0 - 17
AMLC		20 - 37
ICS1		40 - 47
ICS1		50 - 57
ICS1		60 - 67

Note that if the NTUSR configuration directive is set small enough in a given hardware configuration, logging in on an ICS1 line may be impossible.

> Caution
>
> The baud rate of the last AMLC line on the last AMLC board is the AMLC interrupt rate for the system. The standard rate is 110 (one interrupt every tenth of a second). Changing the baud rate of this line can have adverse effects on the rest of the system. For this reason, it is recommended that users not be allowed to assign this line (and perhaps change its baud rate). Instead, this last line should be assigned to SYSTEM or made nonassignable.
>
> The above holds true even if ICS1 boards are present in the backplane. However, ICS1 lines have their interrupt rates hard-wired. Thus, the baud rate of the last AMLC line does not affect the ICS1 lines.

SYNCHRONOUS LINE USAGE

At Rev. 19.1, ICS1 synchronous support is limited to:

- The RJE X80 and HASP products.
- Bisynchronous T$SLC0 use.

RJE is described in the Remote Job Entry Phase II Guide (DOC6053-190). The T$SLC0 subroutine is described in the Subroutines Reference Guide (DOC3621-190).

Changes in T$SLC0

The following changes and additions to the T$SLC0 subroutine should be noted:

- When T$SLC0 is called with a key of 7, the value returned for an ICS1 controller must be interpreted as a decimal value to obtain the model id number for the controller.

- T$SLC0 has several new error messages that can occur when an ICS1 synchronous line is in use. Refer to the subsection called T$SLC0 Error Messages of the section called ICS1 CONTROLLER ERROR MESSAGES, below.

ICS1 CONTROLLER ERROR MESSAGES

This section describes the error messages that may arise when the ICS1 Communications Controller is used.

Error Messages for ASSIGN and AMLC Commands

The following errors may be reported by the ASSIGN or AMLC commands when an ICS1 line is involved:

- BAD PARAMETER

An invalid line speed was specified.

- DEVICE NOT AVAILABLE

Although a particular ICS1 controller is present and configured, it has had a failure and is no longer available.

- OPERATION UNSUCCESSFUL

An attempt to change the line's configuration was temporarily unsuccessful. A subsequent attempt should be successful. If this error persists, a failure in the ICS1 controller is indicated.

Cold Start Directive-related Warnings and Errors

The following messages may appear during cold start if you use the configuration directives incorrectly:

- ILLEGAL AMLCLK ARG SPECIFIED FOR ICS CONTROLLERS

This message is a warning. An attempt was made to specify via AMLCLK a line speed that the ICS1 does not support. If this message appears, the system will still continue cold start procedures, but the "programmable clock" line speed may not be selected for ICS1 lines. This speed may still be selected for non-ICS1 AMLC lines, however.

- BAD ICS DIRECTIVE: xxxxxx

The specified ICS directive either does not exist or has been specified incorrectly. The system will still continue cold start procedure, but this invalid directive will be ignored.

Errors Indicating Possible Hardware Failure

The following errors can occur during cold start or warm start. These errors indicate possible hardware problems. The system will continue to run, but the particular device mentioned in the error message will no longer be available for use. To eliminate the error message, physically remove the device from the system backplane next time the system is shut down.

In the following messages, ZZ indicates a device address, and XXXXXX and YYYYYY are error codes used in diagnosing the particular failure.

- IPQCS ERROR XXXXXX YYYYYY STOPPING DEVICE ZZ

- IPQCS - IF$CDF FAILED FOR DEVICE ZZ: CODES ARE XXXXXX YYYYYY

- ERROR WHILE BOOTING DEVICE ZZ

- CONTROLLER RETURNED Y WORDS OF (HEX) STATUS: XXXX

- CONTROLLER HAS NOT RESPONDED

- ERROR WHILE LOADING DEVICE ZZ

- CONTROLLER CODES: 8001, XXXX

- ERROR WHILE LOADING DEVICE ZZ

Errors Indicating Internal Software Problems

The following error messages can occur during cold start or warm start. These errors indicate internal problems with Primos and/or ICS1 support software.

In some cases, these errors may be serious enough to abort a cold start. Some errors may be transient in nature and may not occur on subsequent cold starts. If an error persists, physically removing all ICS1 controllers from the system backplane should allow the system to run normally.

In some cases, these errors will not abort the system startup. However, some or all of the MDLC, ICS1, and AMLC controllers may not be available for use after cold start has completed. If all of these controllers are required, the system should be shut down and another attempt should be made at cold starting.

- NO TIMED NOTIFY AVAILABLE FOR PCC$HT: SYSTEM MAY HANG

- ERROR WHILE STARTING AMINIT: AMLNO MISMATCH = XXX, AMINIT ERROR = YYYYYY

- MISMATCH OF CCPAT AND AMLCOM. UNUSED CONTROLLER = X, REMAINING AMLCS = Y

- [other error text] ALLOCATING ICS1 FREE POOL

- NO SEG. 0 BUFFER SPACE

- NO PHANTOM INTERRUPT HANDLER

- ERROR WHILE LOADING DEVICE ZZ. TIMED NOTIFY NOT AVAILABLE. SYSTEM WILL HANG IF CONTROLER FAILS TO RESPOND.

- IPQN CODES XXXXXX YYYYYY. ERROR WHILE LOADING DEVICE ZZ.

ICS Downline Load Error Messages

The file ICS1.DL must be contained in top-level directory DOWN_LINE_LOAD*. This file is used during cold and warm starts. It contains software that is loaded into the ICS1 controller. The following error messages are generated if this file is not correct or not present. For example, these messages may be generated if FIX_DISK has truncated ICS1.DL, if the file has been incorrectly rebuilt, or if DOWN_LINE_LOAD* cannot be accessed.

These error messages may be printed during cold start or warm start. If they occur, restore the file ICS1.DL from backup media or rebuild it from source.

- ERROR WHILE LOADING DEVICE ZZ.
- DL FILE NOT FORMATTED FOR DMT. ERROR WHILE LOADING DEVICE ZZ.
- DL FILE PACKETS ARE TOO LARGE. ERROR WHILE LOADING DEVICE ZZ.
- NO DATA PACKETS IN DL FILE. ERROR WHILE LOADING DEVICE ZZ.
- PROGRAM TOO LARGE FOR CONTROLLER. ERROR WHILE LOADING DEVICE ZZ.

If the message:

[other error text]. ERROR WHILE BOOTING DEVICE ZZ.

occurs, the file ICS1.DL could not be accessed. This may be because the file does not exist, the directory DOWN_LINE_LOAD* does not exist, or access was not permitted.

Warning Messages

The following messages may occur during cold start or warm start. These messages are warnings only. They indicate hardware that cannot be identified or that is usable but not operating perfectly. After a warning is issued, the system continues to cold start, but the indicated controller will not be available.

- BROKEN ASYNC LINES

- UNKNOWN CONTROLLER WITH COMMUNICATION DEVICE ADDRESS. CONTROLLER DEVICE ADDRESS IS ZZ. IT RETURNES AN I.D. CODE OF YYYYYY.

- UNKNOWN MDLC-STYLE DEVICE I.D. CONTROLLER DEVICE ADDRESS IS ZZ. IT RETURNED AN I.D. CODE OF YYYYYY.

T$SLC0 Error Messages

The following two error messages can occur when the T$SLC0 subroutine is used with an ICS1 synchronous line:

- SMLCzz ICS CONFIGURATION FAILURE

- SMLCzz ICS CONTROL FAILURE

These error messages indicate internal software problems. If one of these messages occurs, unassign and reassign the synchronous line indicated by zz, and then restart the application that uses T$SLC. If the error persists, synchronous service on the controller in question will not be available.

PST 100 Terminal Support

Rev. 19.1 introduces the PST 100, the new Prime microprocessor-based terminal. Products that can be used on the PST 100 include POWERPLUS, FORMS, FED, and EMACS.

When POWERPLUS is run on a PST 100, the terminal must be index 9 in the POWER terminal table. TERM** and TERM## should be copied from POWERPLUS>TOOLS.

In FORMS, a form definition should include a "device PST100 ... end device" section in order to run on a PST 100. A form can be designed on a PST 100 for use on any block mode terminal supported by FORMS. The TCB command required in FAP is TCB * PST100. For more information on FORMS and FED, including PST 100 support, refer to Chapter 5.

To run EMACS on a PST 100, you must include the command line option:

```
-TTP PST100
```

Refer to the section called EMACS in Chapter 2 of this book.

For more information on the PST 100, refer to the books described in the section called NEW BOOK TITLES in Chapter 1 of this book.