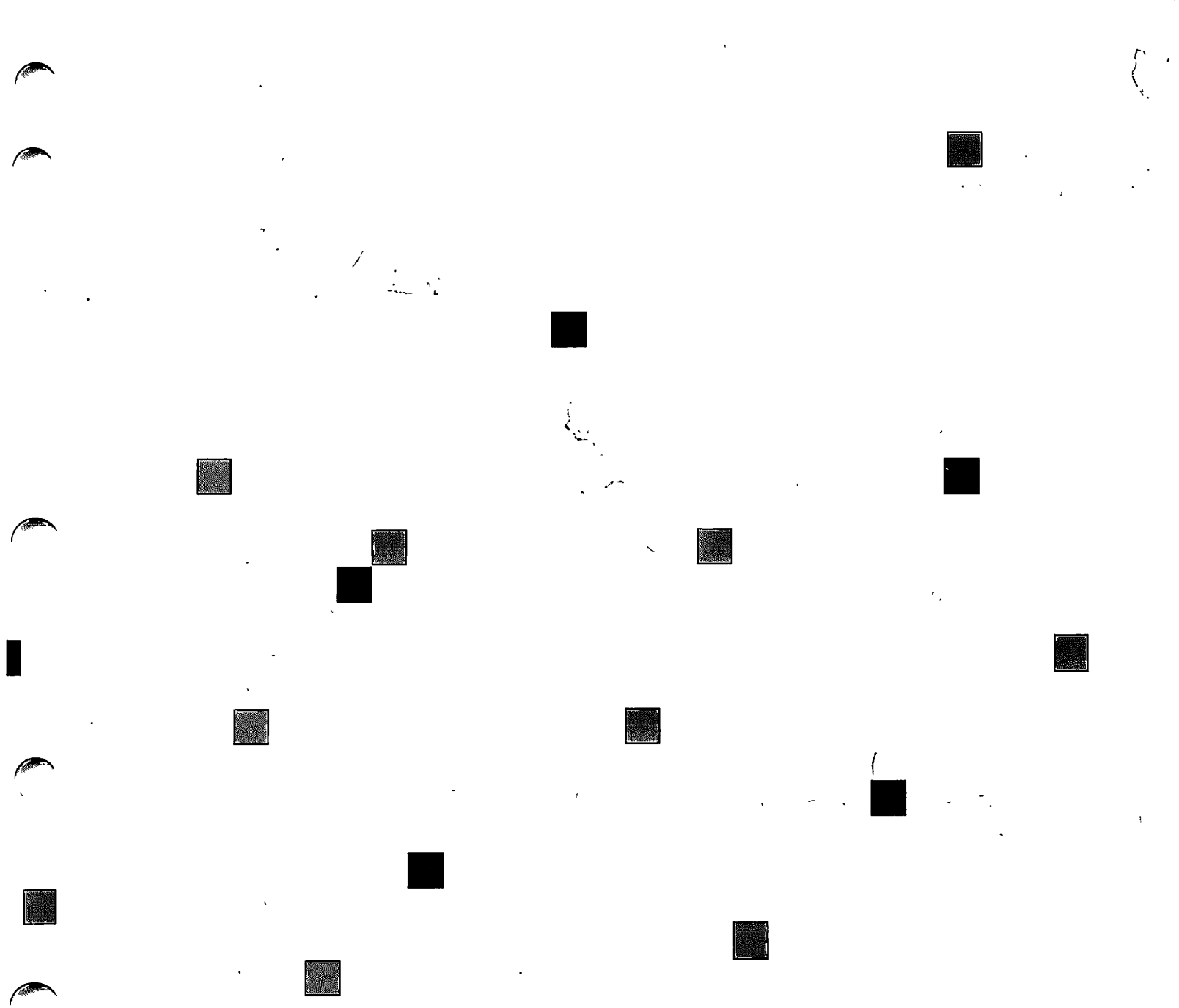

Prime®

Software Release Document

Revision 21.0

DOC10001-4PA

Software Release Document

Fourth Edition

by

William Carbonneau

Joan Karp

Kim Seward

Daniel E. Laukaitis

This guide documents the software operation of the Prime Computer and its supporting systems and utilities as implemented at Master Disk Revision Level 21.0 (Rev. 21.0).

Prime Computer, Inc.
Prime Park
Natick, Massachusetts 01760

COPYRIGHT INFORMATION

PRINTING HISTORY — Software Release Document

Edition	Date	Number	Software Release
First	April 1985	DOC10001-3PA	19.4
Second	January 1986	DOC10001-2PA	20.0
Third	August 1986	DOC10001-3PA	20.2
Fourth	July 1987	DOC10001-4PA	21.0

HOW TO ORDER TECHNICAL DOCUMENTS

To order copies of documents, or to obtain a catalog and price list:

United States Customers	International
Call Prime Telemarketing, toll free, at 1-800-343-2533, Monday through Friday, 8:30 a.m. to 5:00 p.m. (EST).	Contact your local Prime subsidiary or distributor.

CUSTOMER SUPPORT

Prime provides the following toll-free numbers for customers in the United States needing service:

1-800-322-2838 (within Massachusetts)
1-800-343-2320 (within other states)
1-800-541-8888 (within Alaska)
1-800-651-1313 (within Hawaii)

For other locations, contact your Prime representative.

CONTENTS

ABOUT THIS BOOK

Software Release Document Rev. 21.0 provides a summary of both new and enhanced functionality to Prime's user software at Rev. 21.0 This book is divided topically into five chapters and three appendixes, as follows:

- Introduction (Chapter 1)
- PRIMOS (Chapter 2)
- Translator and Environment Products (Chapter 3)
- Data Management Products (Chapter 4)
- Communications Products (Chapter 5)
- Rev. 21.0 Publications (Appendix A)

This book should be used only as an overview of Rev. 21.0, and should be used in conjunction with both Prime user documentation and the online INFO files. This book is intended to be used by those familiar with the Prime operating system and Prime software products.

CHAPTER 1

Introduction

This chapter summarizes the functionality changes to Prime's operating system at Rev. 21.0. Specifically, this chapter lists features and enhancements for PRIMOS, Translator and Environment products, Data Management products, Communication products, and the new Command Line Editor. In addition, this chapter details Special Considerations for the user at Rev. 21.0. (Note that Special Considerations pertaining to LAN300 and LTS300 appear in Chapter 5.)

PRIMOS ENHANCEMENTS

Functionality changes and enhancements to PRIMOS at Rev. 21.0 include the following:

- New CONFIG directive — A new requirement to use the SYSNAM directive to enter a system name during coldstart.
- Forced Shutdown — A mechanism to prevent need for FIX_DISK after some system crashes. Includes discussion of the TPDUMP directive and Error Check messages.
- Paging partitions — A new directive, PAGING, and a new Operator command, PRATIO.
- Security enhancements — Increased system security. Includes the ability to set access on devices, a new Owner access right, and C2-level security.

- Dynamic Badspot Handling and Mirroring — A new mechanism to increase system availability, made available with the introduction of intelligent disk controllers. Includes discussion of conversion to Rev. 21.0 partitions and configuration directives needed for mirroring, disk allocation record schemes including reverse sectoring (made available with the introduction of intelligent controllers, and new options to MAKE and FIX_DISK that implement these features.

- Distributed Systems Management (DSM) — A collection of commands and services that assist in system management. Includes new event logging mechanism for all system logging and several commands for system monitoring.

- Timer Process — A new continuously-running phantom and a mechanism for establishing local time on your system, including Daylight Saving Time.

- Changes to ADDISK — An extension to the number of entries allowed.

- Physical Device Numbers — A redefinition of the binary specification of pdevs to allow an increase in the number of disk controllers and disk drive units that can be put on a system.

- Tape Dumps — New commands that allow partial tape dumps in the event of system crash.

- COMM_CONTROLLER Command — A new command to shut down, initialize, or reboot intelligent controllers interactively or at cold or warm starts.

- ASSIGN and UNASSIGN commands — Enhancements to allow user control of disks or peripheral devices. Includes discussion of enhancements to SET_ASYNC command.

- Batch Subsystem — Changes to files required and security. Includes a discussion on compatibility.

- COPY_DISK command — Enhancements to allow use of multiple partitions, and the RAT, and to display new reports.

- Spooler subsystem — Enhancements to both SPOOL and PROP commands, and addition of new directories. Includes discussion of security, ACL groups and rights, spool queues.

- Search Rules — Extension to the use of search lists and ability for the user to generate these.

- Prime ECS — An expanded character set, ASCII-7 with the 8th bit turned on.

- EMACS screen editor — Enhancements to support LISP.

- Software Serialization — A subroutine and three functions for access to serialization information.

- Documentation corrections — Corrections to New User's Guide to EDITOR and RUNOFF.

TRANSLATOR AND ENVIRONMENT PRODUCT ENHANCEMENTS

Following is a list of functionality changes to Translator and Environment products at Rev. 21.0:

- Prime ECS support.

- Library bundling

- CBL support of INCLUDE$ search rule, relative file enhancements for MIDASPLUS and PRISAM, new options, and a new form of the CALL statement.

- CC support of new options, the UNIX/ANSI restriction on files opened with FOPEN, and a new meaning of the returned value of OPEN().

- F77 support of INCLUDE$, SHORTCALL functionality in I-mode, longer string constants, and a format edit descriptor enhancement.

- FTN support of default code generation.

- Prime Common LISP support of EMACS interface.

- PMA support of the MIP pseudo-op, mode determination of variables and expressions, assembler listing, general register relative format, and IX-mode instructions.

- VRPG support of $INCLUDE search rule.

- BIND support of COMPRESS and INITIALIZE_DATA.

- Common Envelope support of improved stack allocation, and 1020 static character constant strings.

- EMACS interface with Prime Common LISP.

- Changes to SEG LOAD and RPG II documentation.

DATA MANAGEMENT PRODUCT ENHANCEMENTS

Following is a list of functionality changes to Data Management products at Rev. 21.0:

- Data Management support of Prime ECS.
- ROAM supports multiple disk save/restore, increased buffers, 256 default buffers, PRISAM remote transactional access, runtime notification to save After Image file.
- DISCOVER supports item-to-item comparison for PRISAM files, QUIT from "Enter CR" prompt, and increased number of abbreviations.
- MIDASPLUS provides new SYSCOM files and eliminates R-mode dependency for offline utilities and the runtime library.
- POWERPLUS has eliminated R-mode dependency.
- PRISAM provides remote transactional access, new Z$OPEN name, dynamic transaction allocation, new configuration variables, new SYSCOM files, and new error codes.

COMMUNICATIONS PRODUCT ENHANCEMENTS

Following is a list of functionality changes to Communications products at Rev. 21.0:

- DPTX supports BIND loading of DSC modules, and Prime ECS support of PTDSC on PT200 terminals with newer firmware.
- The new product DSM supports improved systems management facilities from any convenient point on the network.
- ICS2 supports PRIMENET X.25 half-duplex functionality.
- ICS3 supports PRIMENET X.25 half-duplex functionality.
- The new product NTS provides a mechanism to connect asynchronous terminals, printers, and other devices to Prime computers through a Local Area Network, the LAN300.
- PRIMENET supports 255 VCs, X.25-1984, and the LAN300.
- X.25 supports 255 VCs, X.25-1984, and the LAN300.
- FTS supports temporary destination files, transfer relative priorities, CONVERT_DATABASE user requirements, changed logging output destination, new priority default, new defer transfer time, more configurable system parameters, and remote revision checking.

COMMAND LINE EDITOR

The new PRIMOS Command Line Editor, EDIT_CMD_LINE (abbreviated ECL), allows you to edit, save, and redisplay PRIMOS command lines, edit and insert text, and expand partial pathnames.

SPECIAL CONSIDERATIONS FOR REV. 21.0

This section replaces the SPECIAL CONSIDERATIONS listed in the document READ ME FIRST for Beta 1 of Rev. 21.

The following items are incompatible with previous revisions. These changes are grouped by function.

Configuration, Coldstart, and PRIMOS.COMI Commands

- Coldstart procedures now require you to supply a system name with the new configuration directive SYSNAM. Coldstart is not able to continue without SYSNAM; the operator will be prompted for a system name if it is omitted.

- The PRATIO config directive is now ignored if present in the configuration file. It has been replaced by the PRATIO operator command, which can be used to change the paging ratio.

  If the PRATIO command is not used, the default paging ratio for a paging partition is based on the size of the paging partition in relation to the total number of paging records on all paging partitions. For example, if a paging partition contained 12% of all the paging records on the system, then the paging ratio for that partition would be set to 12 (out of 100) by default and the partition would be used 12% of the time.

- Remove the command SHARE 40 600 from your PRIMOS.COMI file.

- Remove the following three commands (which shared spool libraries) from PRIMOS.COMI:

```
SHARE SYSTEM>S$2167 2167
R SYSTEM>S$4000 1/12
SHARE 2020 700
```

Add the following line after your ADDISK commands:

```
CO SYSTEM>SPOOL.SHARE.COMI 7
```

Failure to remove old spooler library SHARE commands will cause products that use static mode libraries (e.g., MAGSAV) to fail when invoked.

Doing Backups on a Mirrored Partition

When doing a backup on a mirrored partition, it is important to bring up each partition with the MIRROR_ON command rather than the ADDISK command. This is to keep identical date stamps on the primary and secondary partitions. Specify the command as follows:

```
MIRROR_ON primary_partition secondary_partition
```

Using the ADDISK command causes the date/time of the primary partition to be different from the date/time of the secondary partition. As a result the mirrored software considers the partitions out of date and proceeds to completely copy the entire primary partition to the secondary using a system phantom called the COPY_SERVER. This significantly degrades performance.

Networks

- You must install Rev. 21.0 START_NET if Remote Login is to function. START_NET now sets up the Remote Login port assignments.

- Rev. 21.0 START_NET will invalidate the cache files for existing configurations, causing a slow network startup on the first boot of 21.0. CONFIG_NET will read old configurations but save them as Revision 21.0. An additional question about non-Prime nodes will be asked at validation time.

- Files created with a Rev. 21 version of CONFIG_NET cannot be read with a pre-Rev. 21 version of CONFIG_NET or START_NET. However, Rev. 21 CONFIG_NET and START_NET can read files created with a pre-Rev. 21 CONFIG_NET.

- The DATPAC80 PSDN definition that was removed during beta test has been put back in at FCS. This will only affect customers who had to make adjustments during beta test.

Logging

- Event logging requires the installation of DSM. See the discussion of DSM in Chapter 2, and the DSM Users's Guide.

- Rev. 21.0 system and network event log files (DSM) cannot be read by PRINT_SYSLOG and PRINT_NETLOG, but only by the Rev. 21.0 command DISPLAY_LOG. DISPLAY_LOG cannot read pre-Rev. 21.0 log files. PRINT_SYSLOG and PRINT_NETLOG can read only pre-Rev. 21.0 log files. LOGPRT and NETPRT no longer operate.

- The NETREC and LOGREC configuration directives and the EVENT_LOG operator command no longer operate. Remove NETREC and LOGREC from the CONFIG file to avoid error prompts concerning them at cold start.

FIX_DISK with DSM

- Make sure to run STOP_DSM before running FIX_DISK on a partition that includes the DSM* directory. (This will usually be COMDEV.) STOP_DSM logs out all DSM phantoms for the user, which can take up to two minutes.

  This is necessary because DSM log files are always open. While FIX_DISK will execute properly without this procedure, if DSM is left running, the quota and count value for DSM* will be invalid.

Spooler

- If you are on a pre-Rev. 21 system, you cannot use the -DISK option to spool a file onto a Rev. 21 system. Pre-Rev. 21 systems cannot read Rev. 21 spool queues.

  If you are on a Rev. 21 system, then you can use -DISK to spool the file on a pre-Rev. 21 system. Rev. 21 systems can read pre-Rev. 21 spool queues.

- Any existing printer environment files cannot be used by Rev. 21 spooler. Therefore you must create new printer environment files. Use the conversion program called CONVERT_ENVIRONMENT.RUN in the SPOOL* directory for automatic conversion.

Prime ECS

- PT200 Microcode must be updated to Rev. E to use Prime ECS.

- Printing extended characters: The Rev. 21 Spooler handles each character's top (eighth) bit. Because of this, you should not use the TTY8 protocol to drive printers (and flip the bit); instead specify the TTY protocol. Note that printers attached to a parallel port do not support Prime ECS and will print only seven bits.

  If you have an 8-bit printer and have previously modified the Spooler to flip the eighth bit, and are continuing to use the Rev. 20.2 version of the Spooler, continue to use TTY protocol. When you upgrade to Rev. 21 Spooler with this 8-bit printer, the Spooler will flip the eighth bit; do not incorporate your previous modification into the Rev. 21 Spooler.

ACLs

- Objects that had ACL rights of ALL prior to Rev. 21 will now display as PDALURWX. An ALL designation now includes O (Owner) rights; add O or change to ALL to have ALL display once again.

- If a printer requires a device ACL, ensure that the despooler has access. Use the EDIT_DEVICE_ACCESS command on the corresponding directory in DEVICE*.

Batch

- If you are presently running a Batch subsystem which pre-dates Rev. 20.0, installing Rev. 21.0 Batch will completely destroy your queue definitions. Before you begin installing Rev. 21.0 Batch, be sure to print out your current queue definitions with BATGEN -DISPLAY. You can then use this list to re-enter your Batch queues.

  If you are presently running Batch 20.0 or later, queue definition files will automatically be converted to Rev. 21.0 format upon installation of Rev. 21.0 Batch.

Subroutines (PRIMENET)

- Error returns on X$ routines are more likely. In particular, some calls can now return XS$MEM.

- A new bit, XK$FCW, is used in the flags halfword. Because this bit was previously ignored, this can be accidentally set by some X$ routines in user applications that establish a VC. This can cause unintentional enabling of flow control.

Installing Libraries

- Any pre-Rev. 21 LIB>VSRTLI.BIN file should be deleted prior to restoring the Rev. 21 tape. Explanation follows:

  At Rev. 21, the file LIB>VSRTLI.BIN is a DAM type file. Previous versions of this library were SAM files. This means that, when restoring a new version from tape, MAGRST will not overwrite the old version because the file types are different. MAGRST issues a non-fatal error message, which could be overlooked, leaving an old version of the library on the system.

  Alternatively, restore Rev. 21 files to a holding place on disk, and use COPY to copy files to the appropriate directories.

Compilers

- Some abbreviations for compiler command line options are no longer accepted, for purposes of consistency. An appropriate message indicates that the specified abbreviation is not valid.

- By default, FTN now produces V-mode code. If you want to generate R-mode code, you must add the -32R option to the FTN command line.

CHAPTER 2

PRIMOS ENHANCEMENTS

This chapter contains new features of PRIMOS that concern the System Administrator, the Operator, and the general user. The following topics are covered:

- New CONFIG directive — A new requirement to use the SYSNAM directive to enter a system name during coldstart.
- Forced Shutdown — A mechanism to prevent need for FIX_DISK after some system crashes. Includes discussion of the TPDUMP directive and Error Check messages.
- Paging partitions — A new directive, PAGING, and a new Operator command, PRATIO.
- Security enhancements — Increased system security. Includes the ability to set access on devices, a new Owner access right, and C2-level security.
- Dynamic Badspot Handling and Mirroring — A new mechanism to increase system availability, made available with the introduction of intelligent disk controllers. Includes discussion of conversion to Rev. 21.0 partitions and configuration directives needed for mirroring, disk allocation record schemes including reverse sectoring (made available with the introduction of intelligent controllers, and new options to MAKE and FIX_DISK that implement these features.

- Distributed Systems Management (DSM) — A collection of commands and services that assist in system management. Includes new event logging mechanism for all system logging and several commands for system monitoring.

- Timer Process — A new continuously-running phantom and a mechanism for establishing local time on your system, including Daylight Saving Time.

- Changes to ADDISK — An extension to the number of entries allowed.

- Physical Device Numbers — A redefinition of the binary specification of pdevs to allow an increase in the number of disk controllers and disk drive units that can be put on a system.

- Tape Dumps -- New commands that allow partial tape dumps in the event of system crash.

- COMM_CONTROLLER Command — A new command to shut down, initialize, or reboot intelligent controllers interactively or at cold or warm starts.

- ASSIGN and UNASSIGN commands — Enhancements to allow user control of disks or peripheral devices. Includes discussion of enhancements to SET_ASYNC command.

- Batch Subsystem — Changes to files required and security. Includes a discussion on compatibility.

- COPY_DISK command — Enhancements to allow use of multiple partitions, and the RAT, and to display new reports.

- Spooler subsystem — Enhancements to both SPOOL and PROP commands, and addition of new directories. Includes discussion of security, ACL groups and rights, spool queues.

- Search Rules — Extension to the use of search lists and ability for the user to generate these.

- Prime ECS — An expanded character set, ASCII-7 with the 8th bit turned on.

- EMACS screen editor — Enhancements to support LISP.

- Software Serialization — A subroutine and three functions for access to serialization information. Also includes a section on rebuilding serialized software.

- Documentation corrections — Corrections to New User's Guide to EDITOR and RUNOFF.

NEW SYSTEM NAME CONFIG DIRECTIVE

PRIMOS now requires you to enter a system name during coldstart, through a new CONFIG directive, SYSNAM. A system name is similar to a PRIMENET node name, but a system will have a system name whenever it is running PRIMOS and not just when PRIMENET is running. System names use the same syntax as PRIMENET node names: 1 to 6 characters, contains any of the characters A to Z, 0 to 9, and the seven special characters "&-$._/#"; the first character must be a letter. When PRIMENET is started on a system, it will always adopt the system name as its local node name.

Once the system name has been set it cannot be changed except by a cold start. STATUS SYSTEM will now display the current system name, for example, "System MYNODE is running Primos rev ....".

Use of a system name is mandatory. If the SYSNAM directive is missing or incorrect, PRIMOS will prompt the operator to enter one at the system console. Cold start will not complete until the system name has been set. There is no default system name; the system administrator and/or operator must choose a name for each system. If PRIMENET will be run on the system, the chosen name should obviously be the PRIMENET node name.

Here are examples of how the SYSNAM directive works.

1. The CONFIG directive file contains a SYSNAM directive, for example, "SYSNAM SYS47". The system name will be automatically set to SYS47.

2. The CONFIG directive file contains a SYSNAM directive with no name, "SYSNAM". When PRIMOS encounters this directive it will prompt for a name. When you see the "Enter system name: " prompt, type in your system name.

3. The CONFIG directive file contains an invalid SYSNAM directive, for example, "SYSNAM SYSA+B". PRIMOS will print out an error message, in this case "Invalid system name SYSA+B: contains illegal character '+'.", followed by the "Enter system name: " prompt.

4. The CONFIG directive file contains no SYSNAM directive. PRIMOS will print out a "Missing directive" message and then the "Enter system name" prompt.

FORCED SHUTDOWN

The Forced Shutdown mechanism helps to prevent the administrator from having to run FIX_DISK after a system crash. When PRIMOS encounters a fatal system error (formerly known as a call to BOOT/BOOTI), it attempts to flush the system of all disk information, thereby not making it necessary for the administrator or operator to run FIX_DISK

after a cold start. Please note that this does not mean that running FIX_DISK will no longer be necessary, but that it will be needed less often after a system crash. If Forced Shutdown is successful, the message:

**** From PRIMOS: Forced Shutdown!

prints on each connected user terminal and the CPU halts at the BOOT0_ label. Forced shutdown has a possibility of hanging in the same fashion that shutting a disk down (with the SHUTDN command) can.

Note

Shutdown does not guarantee disk integrity; it only claims that the disks are no worse off than if they had been shutdown normally with the SHUTDN command (e.g. "SHUTDN ALL"). Normal FIX_DISK maintenance is still required and highly recommended on a regular basis.

Since Forced Shutdown must destroy the Ring 0 stacks in order to perform its work, then crash dumps after a Forced Shutdown will not have any stack information available in them for the currently running process. To assist the system programmer, a new CONFIG directive, TPDUMP, has been added.

The TPDUMP CONFIG Directive

The TPDUMP Config directive tells PRIMOS to stop for a crash dump before performing the Forced Shutdown work. The format of the TPDUMP directive is:

```
TPDUMP [YES]
       [NO ]
```

If you do not specify TDUMP in the CONFIG file, the default is not to stop for a crash dump. If you do specify the directive, but without specifying YES or NO, then YES is assumed.

The CPU halts at the label THALT_ when this directive is specified. The administrator can then perform a crash dump following the usual procedures. You should always warmstart after the system stops at this halt location (even if no crash dump is taken). Otherwise, the Forced Shutdown work will not be performed and disk integrity cannot be guaranteed. For information about starting up ROAM-based systems after a Forced Shutdown, see Chapter 4.

Error Check

An Error Check occurs when PRIMOS encounters a fatal situation that results in PRIMOS doing a Forced Shutdown (described above). The format for an error check is:

Error Check: <text of problem encountered>

At Rev. 21.0 four Error Checks are possible. Previously, the following four situations usually resulted in the system halting.

Error TExt	Labeled Halt/Condition
Wired Ring 0 Stack Overflow	R0OVR_
Unwired Ring 0 Stack Overflow	PAGES_
Paging Disk Error During Unwired Ring 0 Page Fault	PAGES_
Paging Disk Full during Ring 0 Page Fault	BOOT0/PAGING_DEVICE_FULL$

The labeled halts have been left in PRIMOS as NOPs for the system programmer.

The Error Check mechanism is available to the system programmer for critical situations where printing a message to the console may not be possible due to page fault considerations.

PAGING PARTITIONS

Use of paging partitions has been enhanced by the addition of the PAGING directive and the PRATIO command.

The new PAGING directive in the CONFIG file allows the operator to allocate up to eight paging partitions, thus enabling the fine-tuning of paging activity among the partitions.

The format of the PAGING directive is:

```
PAGING pdev1 [...pdev8]
```

Unlike the old PAGDEV and ALTDEV directives, you cannot specify the number of paging records on the partitions. You cannot have both the new PAGING directive and the old PAGDEV/ALTDEV directives.

The PRATIO command allows the operator to interactively change the percentages of paging taking place on the paging partitions currently installed on the system.

The PRATIO command replaces the PRATIO config directive.

The format of the PRATIO command follows.

```
PRATIO percent1 [...percent8]
       [-DISPLAY]
```

percent1...percent8 refer to the percentage of paging activity. The total of percentages for all paging must equal 100.

The -DISPLAY option provides information on the percentage of paging activity currently specified for each paging partition.

If the total of the PRATIO percentages does not add up to 100, the system generates an error message. An error message also displays if you specify more PRATIO values than there are paging partitions.

Note that the old PRATIO configuration directive is now ignored.

SECURITY ENHANCEMENTS

The DEVICE_ACLS Command

The System Administrator can use the command DEVICE_ACLS to enable or disable device ACLs.

```
DEVICE_ACLS {-ON }
            {-OFF}
```

At coldstart, the command defaults to DEVICE_ACLS -OFF. Some preparatory work is necessary before the System Administrator can successfully use the command DEVICE_ACLS -ON, either as a command line or as a line in the PRIMOS.COMI file:

1. The System Administrator must verify that the new local directory DEVICE* contains the proper subdirectories, each for a given peripheral device on the system. (DEVICE* is created by running DEVICE_ACLS.BUILD.CPL, or by running the DEVICE_ACLS.INSTALL.CPL program found in the DEVICE_ACLS directory.)

2. After verifying the existence of the subdirectories, the System Administrator sets ACLs on the subdirectories, giving U rights to those allowed access to a device, and NONE to $REST.

If DEVICE_ACLS -ON is used before the subdirectories to DEVICE* are set with ACLs, access is denied to any device not properly prepared. If DEVICE_ACLS -ON is used and no DEVICE* exists, the following message appears:

```
WARNING: DEVICE ACLS ARE ENABLED, BUT DEVICE* COULD NOT BE FOUND.
```

The DEVICE_ACLS command can be included in the PRIMOS.COMI file.

For more information, see the System Administrators Guide, Vol. III: System Access and Security.

New File System Access Right: O

At Rev. 21.0, there is a new access right, Owner (O). This right applies to both files and directories. If you have Owner rights on an object, you can set the access rights (except for Protect (P) rights) on that object. If the object is a file or a segment directory, you can also set its read/write lock.

At Rev. 21.0, the LIST_ACCESS (LAC) and LD commands used on objects for which a user previously had ALL rights will display as PDALURWX for that user. ALL now refers only to those users that have OPDALURWX on a specific object.

C2 Security

A separately-priced Security Auditing Facility is available at Rev. 21.0. This facility brings PRIMOS to the C2 level defined by the U.S. Department of Defense Trusted Systems Evaluation Criteria. This facility creates audit trails, provides reports on the status of the auditing facility, provides reports of closed audit files, and provides a transfer utility to backup audit files on disk to tape or to recover audit files that have already been backed up. The facility includes three new commands:

- SECURITY_MONITOR -- Used to start or stop the Audit Collection Facility
- SECURITY_STATUS -- Provides information on the status of the Audit Collection Facility
- PRINT_SECURITY_LOG -- Provides a report of a closed audit file

In addition, there are three utilities provided with this facility:

- TRANSFER_LOG -- A utility that provides backup and restore functions for systems using the Audit Trail Subsystem.
- CRASH_AUDIT -- A utility that retrieves audit records that are in memory buffers at the time of a system halt.
- CONVERT_TO_ACLS -- A utility that converts all password directories on a partition into ACL directories.

See the System Administrator's Guide, Volume III: System Access and Security for complete details on how to set up and use the Security Auditing Facility.

DYNAMIC BADSPOT HANDLING AND MIRRORING

Dynamic badspot handling and mirroring are available only on Rev. 21.0 partitions.

Conversion to Rev. 21 Partitions

To convert Rev. 20.0 partitions directly to Rev. 21.0 partitions, use FIX_DISK. To convert pre-Rev. 20.0 partitions to Rev. 21.0, it is necessary to use MAKE, because hashed directories are available starting at Rev. 20.0. The two file attributes, Date/Time Created (DTC) and Date/Time last Accessed (DTA), and Contiguous Access Method (CAM) files used with data base management products are also available starting at Rev. 20.0. Thus, if possible, use Rev. 21.0 MAKE to convert pre-Rev. 20.0 partitions. Back up any data on partitions before using MAKE on the partition.

It is necessary to convert to Rev. 21.0 in order to enable dynamic badspot handling and disk mirroring. Prior to Rev. 21.0, when badspots were marked in use in the DSKRAT, two partitions could not appear identical to PRIMOS unless they each had no badspots. At Rev. 21.0, with dynamic badspot handling, intelligent disk controllers make entire physical disks appear error-free to PRIMOS. The intelligent disk controller does this by redirecting all attempted accesses to a badspot to a remapped record. Nonintelligent disk controllers cannot remap records. FIX_DISK marks all badspots as being in use in the DSKRAT thus preventing them from being accessed by a nonintelligent disk controller.

FIX_DISK can make a Rev. 21.0 partition, created in Dynamic Badspot Handling (-IC) mode, compatible with nonintelligent disk controllers. FIX_DISK can also restore Rev. 21.0 partitions, which are compatible with nonintelligent controllers, to Dynamic Badspot Handling (-IC) mode.

Requirements for Dynamic Badspot Handling: The only requirements to activate dynamic badspot handling are to have an appropriate revision intelligent disk controller in the system and to be using Rev. 21.0. Dynamic badspot handling is not supported with a nonintelligent disk controller or with an intelligent disk controller in Nondynamic Badspot Handling (-AC) mode. The conversion to and from Dynamic Badspot Handling (-IC) mode and Nondynamic Badspot Handling (-AC) mode can be done only with an intelligent disk controller. Conversions from Dynamic Badspot Handling (-IC) mode to Nondynamic Badspot Handling (-AC) mode can be done with any disk controller.

All pre-Rev. 21.0 disks can be used on a Rev. 21.0 system. All disks partitioned by Rev. 21.0 MAKE or converted to Rev. 21.0 by FIX_DISK have the appropriate files on them for dynamic badspot handling. At Rev. 21.0, an intelligent disk controller in Dynamic Badspot Handling (-IC) mode can handle all badspots by using the dynamic badspot (DBS) files.

The DBS files are inactive on Rev. 21.0 disks associated with nonintelligent disk controllers or with intelligent disk controllers in Nondynamic Badspot Handling (-AC) mode. FIX_DISK marks the badspots as being in use so that PRIMOS does not attempt to write to them.

Use FIX_DISK to convert between the two modes, Dynamic Badspot Handling (-IC) and Nondynamic Badspot Handling (-AC), of Rev. 21.0 disks. Although -IC mode disks cannot be handled on a nonintelligent disk controller, an -AC mode (nonintelligent disk controller compatible) disk can be used normally on the intelligent disk controller. An -AC mode disk cannot be mirrored, nor does it have dynamic badspot handling occurring on it.

Disk Mirroring

The purpose of disk mirroring is to increase system availability by making it possible to process with pairs of logical disks. These logical disks are equivalent such that if one fails, the other is an exact duplicate and is available for use. The transition to the use of the duplicate disk is automatic.

Disk mirroring allows PRIMOS to:

- Mirror partitions on different disk drive units (which thus have different disk drive unit numbers) of the same disk controller
- Mirror partitions on the same disk drive units (which have the same disk drive unit numbers) of different disk controllers
- Mirror partitions on different disk drive units of different disk controllers
- Continue processing on one partition if the other fails

- Copy a partition as a background process while the partition pair is being mirrored

Mirroring means that all records written to a partition, called the primary partition, are also written to another partition, called the secondary partition. This means that all write operations are duplicated.

Reading of records is not duplicated. Reading is split so that records are alternately read from one mirrored partition and then from the other mirrored partition. This serves to reduce the average time it takes to read a record.

Configuration Directives for Mirroring: You may activate mirroring or initiate mirroring of the command device (COMDEV) or the paging devices at system startup by the use of directives in the configuration file. If you want to mirror any partitions, at least one of the following directives must be in the configuration file:

```
MIRROR
COMDVM pdev
PAGINM pdev1 [...pdev8]
```

Use MIRROR if neither the command device (COMDEV) nor any of the paging devices is to be mirrored at system startup but you want to be able to mirror partitions after startup. If the MIRROR directive is in the configuration file, you can then mirror any partitions after system startup. If either or both of the directives COMDVM and PAGINM are in the configuration file, mirroring is turned on at coldstart for the command device (COMDEV) in the case of the COMDVM directive and for the paging devices in the case of the PAGINM directive. In addition, these directives have the same effect as the MIRROR directive and file system partitions can be mirrored after system startup.

Use COMDVM to initiate mirroring of the command device (COMDEV) at system startup. If you use the COMDVM directive, it must follow the COMDEV directive in the configuration file.

Use the PAGINM directive to specify the pdevs (pdev1...pdev8) on which to mirror corresponding pdevs issued with the PAGING directive. If a pdev used in the PAGING directive does not have a mirror, a 0 (zero) must be used in the corresponding position in the PAGINM directive.

Mirroring Commands: The format of the mirroring commands is as follows. To start the mirroring process, use this command format:

```
{MIRROR_ON}   pdev1 pdev2    [-NO_QUERY]
{MON       }                 [-HELP    ]
```

To stop the mirroring process, use this command format:

```
                                [-SHUT_PRIMARY      ]
{MIRROR_OFF}   pdev1 pdev2      [-SHUT_BOTH [-FORCE]]
{MOFF       }                   [-SHUT_SECONDARY    ]
                                [-HELP              ]
```

Reverse Sectoring

The order in which disk records are allocated for a file is structured to optimize retrieval of records by intelligent disk controllers, that is, disk controllers that are microprocessor-based and are capable of buffering data. Records are allocated to maximize the number of sequential records that are in the disk controller's memory. When you want to read a record, the controller reads the records that logically follow the one you want before getting to your record. These other records in the controller's memory are then available to be used and do not have to be read again from disk.

Thus, the way in which disk records are laid out on disk takes advantage of disk controllers that can read ahead and buffer, or cache, records. Disk I/O performance is improved on disks using intelligent disk controllers by taking full advantage of the controller's buffering capabilities. However, disks using nonintelligent disk controllers do do not suffer any performance degradations brought about by optimizing the allocation of disk records for intelligent controllers.

Forward sectoring records on disk is suited to nonintelligent disk controllers. The nonintelligent disk controller is not capable of buffering records. For intelligent disk controllers, records are allocated to a file so as to maximize the number of records of the file lying on the same physical track that passes under the read head before the desired record. These file records are logically contiguous to the desired record.

For more specific information on reverse sectoring and for recommendations on when to use reverse sectoring, see the Operator's Guide to File System Maintenance.

MAKE and FIX_DISK

New options have been added to MAKE and FIX_DISK as a result of implementation of the features discussed above. The new MAKE options are

-ALL_CONTROLLER (-AC)
-INTELLIGENT_CONTROLLER (-IC)
-OVERRIDE_DEFAULT_INTERLEAVE (-ODI)
-RESTORE_DEFAULT_INTERLEAVE (-RDI)

The argument 21 has been added to the -DISK_REVISION option.

The new FIX_DISK options are

-ALL_CONTROLLER (-AC)
-INTELLIGENT_CONTROLLER (-IC)
-CONVERT_21
-DISK_TYPE (-DT)
-OVERRIDE_DEFAULT_INTERLEAVE (-ODI)
-RESTORE_DEFAULT_INTERLEAVE (-RDI)
-DUMP_DBS (-DDBS)

The pdev argument has been added to the -COMDEV option to allow FIX_DISK to find the DBS file. Please see the Operator's Guide to File System Maintenance for details on all of the MAKE and FIX_DISK options.

DISTRIBUTED SYSTEM MANAGEMENT (DSM)

DSM is a new product at Rev. 21.0. DSM enables administrators, operators and support analysts to perform system management functions from any point in a network. DSM provides the information necessary to do this, and gives these privileged users the capability of defining policy for the network of machines, rather than doing so one node at a time. DSM makes it possible to partition the network in order to define spheres of control within the global network, and to restrict the services available to particular users.

Installing DSM

Build procedures are standard. Once installed, it is necessary to set required ACLs on the DSM* directory and its contents. The following command runs the CPL program that sets these ACLs:

```
R SYSTEM>DSM.INSTALL_ACL.CPL
```

This CPL program can also be used to reset ACLs if corruption is suspected.

Note

The format of DSM*>LOGS>DSM_LG_PURGE_LIST$ is different from the format contained in an earlier version of Rev. 21.0. If you had DSM on a pre-FCS version of Rev. 21.0, you must delete this file. It will be recreated automatically.

Also, the contents of DSM*>CONFIG_FILES>DSM_UMH.CONFIG are different from the contents of DSM*>CONFIG_FILES>DSM_UMH.CONFIG in versions of pre-FCS Rev. 21.0. Users who are reinstalling over an existing Rev. 21.0 should use the new version to ensure all System log messages are correctly collected. Contact your Customer Service Representative for more details.

New Event Logging Mechanism

At Rev. 21.0 Prime system and network event logging is implemented through DSM. This REPLACES the previous mechanism that was performed by User 1, and offers new possibilities for event log management and display.

Notes

In order to make use of event logging, you must have installed DSM.

The commands EVENT_LOG, NETPRT, and LOGPRT no longer operate.

The config directives NETREC and LOGREC no longer operate.

DSM event logging provides a number of new messages, which are detailed in the DSM User's Guide.

The new mechanism uses a privileged process, the System Manager (user SYSTEM_MANAGER), to transfer PRIMOS and PRIMENET event messages to DSM for logging. The System Manager is started and controlled by DSM. Events are passed through the DSM Unsolicited Message Handling (UMH) facility, and recorded in DSM logs. This provides the flexibility of administering and displaying event logs through DSM commands, and the ability to tailor the routing of event messages for specific needs.

The system default is set up to log to the local node. Using CONFIG_UM, you can change where logging goes, either to a different logfile on the local node or to a remote node. Using this capability, you can direct all logging to the same file.

For details of the changes, see the DSM User's Guide.

Functionality

DSM provides the following facilities :

- Collection and collation of event messages, including PRIMOS and network events, through DSM's Unsolicited Message Handling (UMH) and logging services

- Logging of DSM messages in private and system log files throughout the network
- Routing of event messages to any log file or user
- Flexible and powerful log administration and display commands
- Utilities for defining access to DSM on the network, separate from ACLs

A list of DSM commands follows:

CONFIG_UM	Configures destination for unsolicited messages, including PRIMOS and PRIMENET messages
ADMIN_LOG	Creates and administers DSM log files
DISPLAY_LOG	Displays the contents of log files
CONFIG_DSM	DSM configuration file editor
DISTRIBUTE_DSM	Distributes DSM configuration file
STATUS_DSM	Displays currently active configuration file
START_DSM	Starts DSM (supervisor terminal command)
STOP_DSM	Stops DSM (supervisor terminal command)

Additionally, there are a group of commands that make use of DSM. These provide the following facilities:

- System status and resource monitoring of local and remote systems from any point within the network — the System Information/Metering (SIM) commands. The SIM commands follow.

  LIST_ASSIGNED_DEVICES
  LIST_ASYNC
  LIST_COMM_CONTROLLERS
  LIST_CONFIG
  LIST_DISKS
  LIST_LAN_NODES
  LIST_MEMORY
  LIST_PRIMENET_LINKS
  LIST_PRIMENET_NODES
  LIST_PRIMENET_PORTS
  LIST_PROCESS
  LIST_SEMAPHORES
  LIST_SYNC
  LIST_UNITS
  LIST_VCS

- Control of remote Prime systems from any terminal — the Remote System User (RESUS) facility.

For more information, see the DSM User's Guide.

TIMER PROCESS AND NEW OPERATOR COMMAND

Starting at Rev. 21.0, a new TIMER_PROCESS phantom runs continuously. This phantom starts up automatically at cold start when system time is established (by the SETIME command if there is no battery clock).

The SET_TIME_INFO command establishes time information for the local time, for example, the time zone, whether or not daylight saving time goes into effect, and when. This information is used to calculate the Universal Time (formerly Greenwich Mean Time).

This command should be issued before SETIME at coldstart. Ideally, this command should be part of the PRIMOS.COMI file so that the system can calculate Universal Time at startup. If this command is not issued, the time zone is defaulted to 0 and daylight saving time is not used.

Usage

```
{SET_TIME_INFO} { -TIMEZONE/-TZ timezone_offset                    }
{STI          } { -DLST YES [start-date end-date dlst_offset]      }
                { -DLST NO                                         }
```

Options

The -TIMEZONE (-TZ) option specifies the local time of the local system, as offset from Universal Time. The value for timezone_offset is expressed as HHMM or -HHMM. Starting at Universal Time, the absolute values of the negative time zone offsets increase from east to west; the values of the positive time zone offsets increase from west to east. For example, the timezone_offset for Framingham, Massachusetts, is -0500. A time zone can range from 12 hours ahead to 12 hours behind Universal Time.

-DLST YES indicates that daylight saving time is to be considered when calculating universal time. If this option is not specified, then Standard Local Time is not adjusted during the year. Specify the start-date, the end-date, and the dlst_offset value. dlst_offset is expressed as HHMM or -HHMM from Standard Local Time. start-date and end-date are expressed as MMDDYY HHMM. If start-date, end-date, and dlst_offset are not specified, then the default is the U.S. standard.

-DLST NO indicates that daylight saving time is not to be considered when calculating universal time.

CHANGES TO ADDISK

The added disk table (ADDISK) has been extended from 62 entries to 238 entries. This allows any additional local and/or remote partitions to be visible from a given system.

The maximum LDEV of 237 (the range is from decimal 0...237 or octal 0...355) is due to the fact that an LDEV is eight bits (or decimal 0...255), minus room for 10 assignable partitions and 8 paging partitions.

PHYSICAL DEVICE NUMBERS

The following information applies only if you are using more than four controllers.

The binary breakdown of physical device numbers (pdevs) has changed at Rev. 21.0. In addition, the number of disk controllers which may be on one system has increased. Depending on what type of controllers you have, a maximum of eight disk drives may be connected to each disk controller, and one system may have a maximum of eight disk controllers for a total of 64 disk drives per system.

The Operator's Guide to File System Maintenance contains the table (Table 3-2) of pdevs. Notice that the numbers in the table for SMDs and FMDs appear different than previous versions of this table because the basic pdev is computed for controller 0 (controller address '24) whereas previous versions used controller 1 (controller address '26). Any basic pdevs you may already be using based on controller 1 are still valid. You do not have to change basic pdevs.

See Chapter 3 of the Operator's Guide to File System Maintenance for a discussion of pdevs and for instructions on how to construct them.

TAPE DUMPS

In the event the system crashes, it is important that data in memory is collected on crash dump tapes for analysis.

Because physical memory is getting larger, tape dumps can take a long time. Thus, it is now possible for you to get a partial tape dump by using the VCP command RUN 777. Four operator commands allow you to specify how much memory to dump. These commands, which can be issued only from the supervisor terminal, are:

DUMP_SEGMENT	LIST_DUMP
DUMP_USER	RESET_DUMP

DUMP_SEGMENT specifies which segments for all users are written to tape during a partial tape dump. The segments must be specified according to their octal numbers.

DUMP_USER specifies which users will have their segments written to tape during a partial tape dump. The segment numbers must be in the range from '4000 to '7777 octal.

LIST_DUMP displays the current parameters for a partial tape dump. These parameters are either the default parameters or those set by the DUMP_SEGMENT and/or DUMP_USER commands.

RESET_DUMP resets the parameters for a partial tape dump to the default values.

By default, octal segments '0 through '1777 and '6000 through '6003 for all users are to be written to tape during a tape dump. You can change the default values with the DUMP_SEGMENT and DUMP_USER commands.

All four of these commands have a -HELP option to display their syntax. Also, you can type HELP followed by one of the above commands to get a brief discussion of the command's operation. These commands and their options are described in the Operator's Guide to System Commands.

COMM_CONTROLLER COMMAND

The COMM_CONTROLLER command permits the System Administrator or Network Operator to shut down, initialize, or reboot ICS controllers interactively, as well as at cold or warm starts. Use this command to reboot or write the memory contents of a LAN Terminal Server (LTS) or a LAN Host Controller (LHC) to the host for analysis.

This command can only be issued from the Supervisor terminal. Prompt and error messages are displayed to the operator at the terminal. More detailed messages are logged to a DSM Log File that can be retrieved by the operator any time after command completion.

The command is loaded directly into the CMDNC0 directory by the build CPL program. The COMM_CONTROLLER command also uses the DOWN_LINE_LOAD* (downline load file support) and NETWORK_MGT* (server logic run files) directories.

The format of this command is

```
COMM_CONTROLLER -subcommand [-option argument....]
```

There are 5 subcommands:

Subcommand (Abbreviation)	Enabled For

HELP	ICS1/2/3 and LHC/LTS
INIT	ICS1/2/3
LOAD	ICS1/2/3 and LHC/LTS
SHUTDOWN	ICS1/2/3
UPLINE_DUMP (ULD)	LHC/LTS

HELP and LOAD subcommands are common to all controllers supported. Choose only one of the subcommands on a command line. If you do not specify one of these subcommands, HELP is assumed and the help text is displayed.

INIT — Initiates an automatic shutdown followed by an integrity check on the designated ICS controller(s). The controller(s) is reset and a PROM SELF-VERIFY is done resulting in memory size being returned.

LOAD — Initiates a full downline load of a specified file or protocol combination to the designated controller(s). An automatic shutdown is executed followed by a series of steps that bring the controller(s) to a state of readiness.

SHUTDOWN — Freezes the designated ICS controller(s) and breaks all logical connections to that controller(s).

UPLINE DUMP — Initiates a request for the designated LHC or LTS to dump pre-determined sections of its memory to a specified file on the host machine.

System or Network Administrators can include this command in abbreviations, CPL or comi files.

Options can be specified in any order on the command line. However, the type and combination of options for a given request associated with various controllers is critical. The complete list of options follows.

Option (Abbreviation)	Enabled For
ALL	ICS1/2/3 and LHC/LTS
PROTOCOL (PR)	ICS2/3 and LHC
DEVICE_ADDRESS (DA)	ICS1/2/3 and LHC/LTS
DEVICE (DEV)	ICS1/2/3 and LHC/LTS
LAN_NAME (LN)	LHC/LTS
DEST_NODE_ADDRESS (DNA)	LHC/LTS
DEST_NODE_NAME (DNN)	LHC/LTS
PATHNAME (PN)	ICS1/2/3 and LHC/LTS
NO_QUERY (NQ)	ICS1/2/3 and LHC/LTS

Example of using COMM_CONTROLLER command

```
COMM_CONTROLLER -LOAD -DEV ICS3 -DA 10
[COMM_CONTROLLER Rev. 21.0 Copyright (c) 1986, Prime Computer, Inc.]

ICS3 controller(s) 10  currently active.
Continue <Y or N>? y

ICS3 controller shutdown in progress...

ICS3 prom self-verify diagnostics in progress...

ICS3 downline load in progress...

ICS3 run-time self-verify diagnostics in progress...

ICS3 controller startup in progress...

ICS3 downline load operation results:
     ICS3 at address 10: SUCCESSFUL
```

Restrictions

At Rev. 21.0 the following restrictions apply to COMM_CONTROLLER.

- If an ICS controller is to be shutdown and that particular controller supports SNA, RJE and/or PRIMENET/X.25 lines then one or all of the SNA, RJE, PRIMENET/X.25 lines must be stopped before executing the COMM_CONTROLLER command.

- The COMM_CONTROLLER command can not allocate line numbers. When you replace, remove, or add a LAC, you must change your ICS CARDS directive to agree with the current configuration and coldstart the system before the new lines can be used.

- Resources allocated to ICS controllers are drawn from a Free Pool. As controllers are shutdown their resources are returned to the Free Pool until such time as a reboot takes place. Since the Free Pool space is a limited and valuable system resource, frequent shutdowns and reboots of ICS controllers, particularly with different load files or protocol combinations, may result in the available space in the Free Pool becoming fragmented and unusable. When this occurs an error indicating 'insufficient seg-0 space' will be issued and logged. While the logic does a certain amount of garbage collection within the Free Pool it is inevitable that frequent stops and starts may result in a Free Pool that is unusable.

To resolve this without having to cold start the system, it is suggested that the System or Network Operator shut down all ICS controllers on the backplane, taking the first restriction above into consideration. When they are all shutdown the Free Pool will be resolved and controllers can then be rebooted.

For more information, see the Rev. 21.0 update to the ICS User's Guide or the System Administrator's Guide, Volume II: Communication Lines and Controllers.

ENHANCEMENTS TO THE ASSIGN AND UNASSIGN COMMANDS

The ASSIGN command gives the user complete control of a disk or a peripheral device. At Rev. 21, the format

```
ASSIGN ASYNC -LINE n [-TO m]
```

(where n and m are DECIMAL line numbers) replaces the octal format

```
ASSIGN AMLC protocol amlc-line config_word.
```

The -TO option conveniently assigns a range of lines from n to m inclusive in one command line.

The new ASSIGN ASYNC command does not include the protocol and config word options that were available with the ASSIGN AMLC command. If you would like to change the protocol or any other options, issue the SET_ASYNC command to do this. For example, the two command lines

```
ASSIGN ASYNC -LINE 8
SET_ASYNC -LINE 8 -SPEED 9600 -PROTOCOL TTY
```

replace the old command

```
ASSIGN AMLC TTY 10 2413
```

Although command lines issued in the old format are supported at this revision, the new decimal format is recommended.

A new format for the UNASSIGN Command is available as a companion to ASSIGN ASYNC -LINE n. Exclusive use of an asynchronous line is relinquished by issuing the command

```
UNASSIGN ASYNC -LINE n [-TO m]
```

where n and m are DECIMAL line numbers).

Do not follow a decimal ASSIGN ASYNC command with an octal UNASSIGN AMLC command. In limited situations this works, but it is generally better not to combine octal and decimal formats.

Enhancements to the SET_ASYNC Command

SET_ASYNC defines terminal line characteristics for an individual line or a range of consecutively numbered lines. Use SET_ASYNC at coldstart to set the parameters for a line, or use it interactively to change the characteristics of a particular line.

Prior to PRIMOS Rev. 21.0 the SET_ASYNC command was restricted for the System Administrator's use. At PRIMOS Rev. 21.0, local and NTS users can temporarily change the characteristics of their own line or any lines they have assigned. These changes will remain in effect until the user logs out or issues another SET_ASYNC command. Certain line characteristics, such as ASSIGNABLE, DISLOG, SPEED_DETECT, LOOP_LINE, and their converses, may effect buffer allocation and/or depend on hardware. These options are reserved for the System Administrator.

```
          {-DISPLAY                    }
SET_ASYNC {-LINE n [-TO m] [options]}
          {-HELP                       }
          {-SYSTEM                     }
```

-DISPLAY, -DP Displays the current characteristics for the line.

-HELP, -H Provides the command line format and a list of all available options. Some options are restricted for the System Administrator's use.

-LINE Configures a line (or lines) with the selected options.

-SYSTEM, -SYS This new option sets the line characteristics to the system defaults.

Several new Administrator options have been added to SET_ASYNC:

-DISLOG Automatically logs out the line if disconnected or the carrier signal is logically low.

-NO_DISLOG Disables DISLOG.

-SPEED_DETECT Enables Auto Speed Detect to determine the speed of incoming data on the line.

-NO_SPEED_DETECT Disables Auto Speed Detect.

All input to SET_ASYNC must be decimal based. Octal input is not supported. The new protocols, TTY8 and TTY8HS, for use with Prime ECS, are described in Chapter 5, COMMUNICATIONS PRODUCTS ENHANCEMENTS.

ENHANCEMENTS TO BATCH SUBSYSTEM

Several files previously required by Batch are no longer in existence. This includes the directory OTHER.

BATCHQ must reside in an ACL partition. This requires at least one partition in the system to have ACLs.

The CIFILE and Q.CTRL subdirectories are now ACL format and do not have passwords. This affects any program that was depending on password access to these directories.

The BATCHQ>INIT program no longer supports the -ADMIN option and no longer sets the user running it as a Batch Administrator. In this release, to do builds and installations, you will need to be a member of the .BATCH_ADMIN$ group. Batch Administrators are identified using EDIT_PROFILE to set them as members of the ACL group .BATCH_ADMIN$. For complete information on Rev. 21.0 Batch, see the Operator's Guide to the Batch Subsystem.

Compatibility

The BATDEF file, which contains queue descriptions, is retained with only a change of version code, so it is not necessary to redefine the queues as was required by Rev. 20.0. All other files have been revised and the ACL protections set to be much more restrictive.

The new BATCHQ>INIT program converts the files and directories automatically when run on a Rev. 21.0 system. The new INIT program will not operate on a prior PRIMOS revision, and the old Batch Subsystem will not operate with the new queue files after the new INIT program is run.

ENHANCEMENTS TO COPY_DISK

At Rev. 21.0, COPY_DISK is enhanced in the following areas:

- Uses multiple partitions
- Uses the Record Availability Table (RAT)
- Displays useful partition statistics
- Displays a report of the copy operation in progress

Multiple Partitions

COPY_DISK now allows a total of 17 partitions for both input and output. Each input partition must have a corresponding output partition, and all partitions in the copy process must reside on the same physical pack. Partition pairs may be specified in any order.

Note

The new COPY_DISK multiple partition functionality applies to all disk drives except CMD disk drives.

Using the Record Availability Table

At Rev. 21.0, COPY_DISK initially checks the Record Availability Table (RAT) for tracks that have currently active records so that COPY_DISK copies only those tracks to the correct positions on the output partition.

User Display

After you enter all input/output partition pairs, COPY_DISK displays the following information:

- The names of all the partitions that are being copied
- The physical device numbers of these partitions
- The number of tracks that contain active records
- The percentage of tracks currently in use

Copy in Progress Report

At Rev. 21.0, COPY_DISK displays the progress of the copy operation as it is happening. An asterisk (*) appears for every 5 percent of tracks that have been successfully copied: the report totals 20 asterisks (100 percent). Above the row of asterisks is a tabulation line that indicates the percentage of successfully copied tracks. Also, COPY_DISK displays a message that indicates when the copy operation has been completed.

For more information on COPY_DISK, see the Data Backup and Recovery Guide.

ENHANCEMENTS TO THE SPOOLER SUBSYSTEM

This section gives a brief overview of the changes that have been made to the operation and structure of the Spooler subsystem at Rev. 21.0, and of the new responsibilities of the person who sets up the Spooler. Changes to the PROP and SPOOL commands to accommodate the revised functionality, and more detailed descriptions of the functionality appear in the Rev. 21.0 edition of the PRIMOS Commands Reference Guide and the Operator's Guide to the Spooler Subsystem.

Coldstart Procedure for the Spooler Subsystem

At coldstart, the Spooler subsystem must be started after the date and time are set, but before the MAXUSR command and before allowing remote systems access to your system. To share the Spooler subsystem at coldstart, do not use previous SHARE commands (such as SHARE SYSTEM>S$2167 2167). Instead, insert in the PRIMOS.COMI file the following command that runs the Prime-supplied SPOOL.SHARE.COMI file:

```
CO SYSTEM>SPOOL.SHARE.COMI
```

Note that the SPOOL.SHARE.COMI file contains the PROP -COLDSTART command. Do not remove this command from the file. Make sure that PROP -START commands in PRIMOS.COMI occur after the CO SYSTEM>SPOOL.SHARE.COMI command.

Spooler System Directories

At Rev. 21.0, the Spooler subsystem uses three system directories: SPOOL*, SPOOL_QUEUE*, and SPOOL_DATA*. The pre-Rev. 21.0 SPOOLQ still exists, but it is used only by the Prime OAS software. The functions of the Spooler directories are as follows:

- SPOOL* holds administrative and definition files and subdirectories. Such files include printer environment files, spool queue definition files, and device handler programs. There can be only one SPOOL* directory on the system.

- SPOOL_QUEUE* contains the file that controls all print requests (that is, all spooled files). The file is an index to the actual spooled files that are stored in the SPOOL_DATA* directory. SPOOL_QUEUE* can also contain two optional files: the FULL_LIST_USERS file that has the names of users who can see the entire spool queue with the SPOOL -LIST command; and the DATA_PARTITIONS file that lists the names of local partitions that contain SPOOL_DATA* directories. There can be only one SPOOL_QUEUE* directory on the system.

- SPOOL_DATA* is the directory in which the spooled files are actually stored. There can be several SPOOL_DATA* directories on the system, but only one on any partition.

Printer Environments

Rev. 21.0 printer environments are no longer binary files created by the PROP -CREATE command. Instead, they are now ASCII files that the operator creates with a standard text editor (such as ED or EMACS). No pre-Rev. 21.0 printer environment files can by used by the Rev. 21.0 version of spooler; thus you must create new printer environment files. These files are stored in the SPOOL* directory.

Printer environment files are also modified with a text editor and are deleted with the DELETE command.

Changes to the PROP Command

Because printer environment files are created and modified with a text editor, the PROP command no longer supports the -CREATE, -MODIFY, and -DELETE options.

The PROP command has two new options. The -VERIFY option allows you to check environment files before you put them into service. The -RESET option allows you to stop an environment file and start another in its place, all in a single operation.

The output of the PROP -STATUS command is different. Instead of giving either a "stopped" or "started" status for each printer, the Rev. 21.0 PROP can give any one of eleven statuses for started printer environments, depending on what the printer is doing: Aborting, Backing Up, Dropping, Hanging, Idle, Lineup, Printing, Reset, Restarting, Starting Up, and Stopping. To check the status of all defined environments, use the PROP -STATUS -ALL command.

Note also that the spelling of the IDLE, FINISH, and NOW arguments now begin with hyphens (-IDLE, -FINISH, and -NOW).

Access, Security, and Privileged Users

At Rev. 21.0, all Spooler access is controlled by setting ACLs on the relevant files and directories, and through two special ACL groups, .SPOOL$$ and .SPOOL_ADMINISTRATOR$.

The ACLs for both SPOOL_QUEUE* and SPOOL_DATA* must be as follows:

```
.SPOOL$$:ALL
$REST:NONE.
```

The ACL for SPOOL* should be:

```
.SPOOL_ADMINISTRATOR$:ALL
SYSTEM:ALL
$REST:LUR
```

The System Administrator should be a member of .SPOOL_ADMINISTRATOR$; if not, then he or she should have ALL rights.

A supplied utility named SYSTEM>SPOOL.INSTALL_ACL.CPL will set proper ACLs automatically on the local SPOOL_DATA* and SPOOL_QUEUE* directories. If you have created a SPOOL*>DATA_PARTITIONS files, the program sets ACLs on all the SPOOL_DATA* partitions listed in that file. If no such file exists, the program sets ACLs only on the first SPOOL_DATA* partition it finds.

The .SPOOL$$ ACL group should not be assigned to any user, or else that user will have complete access to the spooled files.

Membership in the .SPOOL_ADMINISTRATOR$ ACL group gives users special privileges, such as the right to modify the printing attributes of other users' spooled files.

Spool Queues

There can be only one spool queue (named SPOOL_QUEUE*) on each Rev. 21.0 system. The only size limit to the number of entries in the queue is the size of the partition.

Rev. 21.0 spool queues cannot be accessed by despooler phantoms of earlier revisions of the Spooler. However, Rev. 21.0 despooler phantoms can access spool queues of earlier revisions of the Spooler subsystem.

EVFU File Formats

A new, easier to use EVFU file format is provided from Rev. 21.0 onwards. The pre-Rev. 21.0 type of EVFU file continues to function so existing EVFU files can be used.

Accounting Routine

An interface is supplied that allows users to supply their own accounting routines.

Functionality Changes and Additions for the General User

Printer attributes: Attributes such as form type and printer designation can now be specified with the one option -ATT. Support for the -FORM and -AT options are maintained. Attributes are required, but may be set up as defaults. Attributes are stored in a special file in the SPOOL* directory.

The -LIST option: At Rev. 21.0, for added security, this command displays only files spooled by the issuing user, not by all users. The System Administrator, however can change this for any user so that the user can see the entire spool queue.

Despooler: The Rev. 21.0 despooler lets the administrator choose between defining very detailed queue locations or letting the despooler take default actions. Also, the Rev. 21.0 despooler allows support of non-standard printers by supplying device drivers as EPFs. This means users can write their own device drivers for their printers. The Rev. 21.0 despooler allocates one semaphore for each despooler phantom so that different printers don't compete for the same resources.

SEARCH RULES

A search rule is a predetermined pathname of a file system object. When a user requests a file system object, but does not give the fully-qualified name of that object, PRIMOS uses search rules to attempt to locate the object. Search rules are contained in lists called search lists. The order in which the rules appear in the search list determines the order in which PRIMOS looks for the object.

Previously, PRIMOS had exclusive control of these search rules. At Rev. 21.0, the PRIMOS search rules facility has been extended so that users can now set their own search rules and build search lists with rules in any order they require. User-created search rules override PRIMOS search rules.

PRIMOS maintains five separate default search lists in the SEARCH_RULES* directory:

ENTRY$
COMMAND$
ATTACH$
BINARY$
INCLUDE$

As mentioned above, users can override any of these default lists or create additional private lists.

INCLUDE$ and BINARY$ searh lists are used with compilers.

See the second edition (Rev. 21.0) of Advanced Programmer's Guide, Volume II for a complete discussion of Search Rules.

A new command EXPAND_SEARCH_RULES (ESR) permits users to expand a filename to its fully-qualified pathname using the search rules facility. ESR can be invoked as a PRIMOS command or a CPL function. The LIST_SEARCH_RULES (LSR) and SET_SEARCH_RULES (SSR) commands support the five standard PRIMOS search list types as well as any private search lists. For details on the commands, see the Rev. 21.0 edition of the PRIMOS Commands Reference Guide.

Search Rules can also be accessed by programs through subroutines. These are documented in the Rev. 21.0 update to Volume II of Subroutines Reference Guide.

PRIME EXTENDED CHARACTER SET

As of Rev. 21.0, Prime has expanded its character set. The basic character set remains the same as it was before Rev. 21.0: it is the ASCII 7-bit set (called ASCII-7), with the 8th bit turned on. However, the 8th bit is now significant; when it is turned off, it signifies a different character. Thus, the size of the character set has doubled, from 128 to 256 characters. This expanded character set is called the Prime Extended Character Set (Prime ECS). All non-chargeable software supports Prime ECS except MAGNET. To use ASCII-8 tapes, use a User Table that flips the 8th bit.

The pre-Rev. 21.0 character set is a proper subset of Prime ECS. These characters have not changed. Software written before Rev. 21.0 will continue to run exactly as it did before. Software written at Rev. 21.0 that does not use the new characters needs no special coding to use the old ones.

Prime ECS support is automatic at Rev. 21.0. You may begin to use characters that have the 8th bit turned off. However, the extra characters are not available on most printers and terminals. Check with your System Administrator to find out whether you can take advantage of the new characters in Prime ECS.

The pre-Rev. 21.0 character set consists of the characters with decimal values 128 through 255 (octal values 200 through 377). The characters added at Rev. 21.0 all have decimal values less than 128 (octal values less than 200).

Specifying PRIME ECS Characters

Direct Entry: On terminals that support Prime ECS, you can enter the printing characters directly; the characters appear on the screen as you type them. For information on how to do this, see the appropriate manual for your terminal.

A terminal supports Prime ECS if

1. It uses ASCII-8 as its internal character set, and

2. The TTY8 protocol is configured on your asynchronous line.

If you do not know whether your terminal supports Prime ECS, ask your System Administrator.

On terminals that do not support Prime ECS, you can enter any of the ASCII-7 printing characters (characters with a decimal value of 160 or higher) directly by just typing them.

Octal Notation: If you use the Editor (ED), you can enter any Prime ECS character by typing

```
^octal-value
```

where octal-value is the three-digit octal number representing the character. You must type all three digits, including leading zeroes.

Before you use this method to enter any of the ECS characters that have decimal values between 32 and 127, first specify the following ED command:

```
MODE CKPAR
```

This command permits ED to print as ^nnn any characters that have a first bit of 0.

See the Rev. 21.0 update to Prime User's Guide for a table of Prime ECS characters.

EMACS SCREEN EDITOR

Allocating Segments

Unless your user process has been allocated an adequate number of dynamic segments, you will be plagued by EMACS telling you "NOT ENOUGH SEGMENTS" and having many of your commands (e.g., find_file, insert_buffer, etc.) aborted. The fix for this is very simple.

To run EMACS without getting such errors, you should have your System Administrator set your allowed number of dynamic segments to at least 100. This should only be done by the System Administrator, using EDIT_PROFILE.

Note: EMACS uses segments in the 2000 range for its libraries. At Rev. 19.4 and later revisions, it allocates storage in the dynamic segments.

EMACS Support for LISP

EMACS support for LISP is based on the existing LISP language mode that is documented in the EMACS Reference Guide. Two new modes are added. LISP mode has several additions/enhancements:

esc-q (new) Pretty print the current s-expression starting at the current position. This works for both PEEL (the PRIME EMACS Extension Language) and PRIME Common LISP.

esc^A (new) begin_defun
Go to the beginning of the current defun or whatever.

esc^E (new) end_defun
Go to the end of the current defun or whatever.

esc^R (new) move_defun_to_screen_top
Move defun to screen top.

balbak (enhanced)
Much faster than before.

esc-h (new) Mark the current defxx.

Common_lisp mode adds the following functionality:

CL_on Turns on the shadow mode for Common_lisp. This will also turn on LISP mode.

CL_off Turns off the mode.

CL_send_region (esc-e)
Sends the current region to PRIME Common LISP for interpretation.

CL_macroexpand_region (esc-m)
Sends the current region to PRIME Common LISP for macroexpansion.

CL_compile_region (esc-z)
Sends the current region to PRIME Common LISP for compilation.

CL_describe_current_symbol (esc-=)
Describe the current symbol (send the current symbol wrapped in (describe ...).).

CL_send_file (esc-t)
Tell PRIME Common LISP to load file (after saving).

CL_help (^x-?) Online summary of this information.

CL_listener Puts the user in a LISP listener window (buffer = [listener]). While in the listener buffer, the user is — in effect — typing at LISP top-level. More information on the Listener below.

CL_balbak ( )) Common LISP paren balancer. Slightly modified from the paren balancer from PEEL (LISP mode).

CL_cr (cr) Typing indenter. Pretty prints program as it is.

The Listener is a buffer that 'talks' to PRIME Common LISP. The specific commands and keybindings for the Listener are:

cl_send_lisp_line (cr)
Send current line to Common LISP (end of buffer only!).

(^x-?) Online help in the Listener.

The other PRIME Common LISP 'common_lisp' mode extensions (esc-=, esc-e, esc-m, and esc-z) also work in the Listener.

Note

LISP comes with an EMACS interface for those sites which have EMACS. The additional support for LISP is installed as part of the install file. EMACS should be reshared after installing LISP if EMACS support is desired.

SOFTWARE SERIALIZATION

Rev. 21.0 provides a subroutine and three active functions to allow Prime and user-written software to access serialization information for any serialized product on the system.

- KLM$IF - Information routine
- KLMT - Active function to test field values
- KLMD - Active function to return groups of serialization data
- KLMF - Active function to return a serialization data field

See the Rev. 21.0 update to Subroutines Reference Guide, Volume III for a complete discussion of KLM$IF. See the Rev. 21.0 edition of the CPL User's Guide for a complete discussion of KLMT, KLMD, and KLMF.

REBUILDING SERIALIZED SOFTWARE

Rebuilding Prime software necessitates transferring serialization information from the original product into the rebuilt product. Additional steps are required before commencing rebuilds:

1. The top-level UFD TEMPLATE* must be restored from the release tapes. This is normally the last logical tape of the sequence and contains serialization data files for each product on the tape.

2. Execute the procedure KLM.INSTALL.CPL provided in the TEMPLATE* UFD. This ensures that the initialization utility for rebuilding is set up in both CMDNC0 and KLM*>CMDNC0 directories.

The customer will then perform modifications to the source code of the product and rebuild the product using the Prime-supplied build files. The build file will automatically restore the serialization information into the rebuilt product.

INITKLM (Rebuild) - Restore serialization template utility

Command: INITKLM [output_pathname] [options]

This utility is invoked from the original Prime-supplied build files and normally does not require any changes to be made to these build files. The rebuilt software will have a 'v' appended to the revision stamp. The build file may be modified to make use of the version option of the INITKLM utility which will allow customers to track rebuilds. Addition of

-VeRSion <string>

to the INITKLM command line will append 'v<string>' to the revision stamp in the rebuilt software.

Example:

```
initklm primos.save -vrs A5 {existing options}
```

gives login messages:

```
XXXXXX (user 66) logged in Tuesday, 09 Sep 86 10:59:52.
Welcome to PRIMOS version 21.0vA5
Copyright (c) 1986, Prime Computer, Inc.
Serial #XXXXXXXXXXXXXXXX (A. Customer)
Last login Tuesday, 09 Sep 86 07:56:12.
```

DOCUMENTATION CORRECTIONS

Following are corrections to the New User's Guide to EDITOR and RUNOFF (FDR3104-101A/B).

ED

1. If you wish to use ED to edit files containing Prime Extended Character Set (Prime ECS) characters, first specify the following ED command:

   ```
   MODE CKPAR
   ```

   This command permits ED to print as ^nnn any characters that have a first bit of 0.

2. In the GMODIFY command example on page 8-13, the g command is not correct. The correct command is

   ```
   g d:e2fsi/:  /c:;n;*
   ```

RUNOFF

1. A .SKIP command at the top of the first page of a document is ignored.

2. The argument to the .LENGTH command is not optional; you must specify a value.

Corrections to Operator Information

Page 2-15 in Operator's System Overview, DOC9298-2LA erroneously documents three non-existent commands: SET_DEVICE_ACCESS, EDIT_DEVICE_ACCESS, and LIST_DEVICE_ACCESS. Please see System Administrator's Guide, Volume III for information on setting Device ACLs.

CHAPTER 3

TRANSLATOR AND ENVIRONMENT PRODUCT ENHANCEMENTS

This chapter covers the enhancements to the Translator (Language) and Environment product areas. Among the topics discussed are:

- Libraries
- Compilers
- Utilities
- Changes to Documentation

LIBRARIES

This section documents the changes to libraries at Rev. 21.0. These include:

- Support of Prime ECS
- Bundling of Runtime Libraries
- CC Library
- PL/I G Library

SUPPORT FOR PRIME ECS

At Rev. 21.0, the Pascal library, APPLIB, the PL/I G library, and the Fortran Matrix library (MATHLIB) support the Prime Extended Character Set (Prime ECS). For more details on ECS, see Chapter 2.

BUNDLING OF RUNTIME LIBRARIES

As of Rev. 21.0, the following libraries are bundled: CBL, PL/I G, and SYSTEM.

CC LIBRARY

This section describes the Rev. 21.0 changes for the CC library.

Note

The size of CCLIB.BIN (the static library that is linked with programs using the SEG linking-loader) has increased over the size of its Rev. 20.2 version. This does not affect use of the shared, dynamically-linked EPF library.

UNIX/ANSI Restriction on the Use of FOPEN

When a file is opened using update mode ('+' as the second or third character in the mode argument), both input and output may be performed on the associated stream. However, output may not be directly followed by input without using an intervening call to the fflush function or to a file positioning function (fseek or rewind), and input may not be directly followed by output without using an intervening call to the fflush function or to a file positioning function, unless the input operation encounters End-Of-File.

The Meaning of the Return Value of OPEN()

The integer file ID returned by OPEN() is no longer the PRIMOS file unit on which a disk file is really open. It is now a unique file handle used to identify the file to the library's low-level I/O routines. There may be cases, however, that a user must determine the PRIMOS file unit that is being used to access a disk file: the routine BIO$PrimosFileUnit() has been added for this purpose. Given a file ID returned from OPEN() or FILENO() of a "FILE *" variable, BIO$PrimosFileUnit() will return the corresponding PRIMOS file unit. For example:

```
#include <stdio.h>
int primosUnit1, primosUnit2;

int fileID = open("FileName", 2);
FILE *handle = fopen("AnotherFileName", "w");

primosUnit1 = BIO$PrimosFileUnit(fileID);
primosUnit2 = BIO$PrimosFileUnit(fileno(handle));
```

The library routine FILENO(), returns the file handle associated with a given file pointer returned by the FOPEN() routine. When a given C library routine is documented as having an argument which is the return value of OPEN(), the given routine no longer expects a PRIMOS file unit as that argument but instead expects the unique file handle returned by OPEN() or FILENO(). Any attempt to pass a PRIMOS file unit to a C library routine that expects the new file handle, will likely result in a file system error (for instance, "Unit not open.")

Permanent Restrictions

Calling Non-C Language Routines: 64V and 32IX Compatibility: Interfacing to non-C language routines is done by declaring the non-C routine with the FORTRAN keyword in the calling C program. Past versions of CC (64V-Mode) have contained support for the options -OLDFORTRAN and -NEWFORTRAN, where -OLDFORTRAN was the default. These options effect the interpretation of a call to a FORTRAN-declared routine from a C program.

-NEWFORTRAN is the default. -OLDFORTRAN corresponds to the old method of calling non-C routines. If neither -OLDFORTRAN or -NEWFORTRAN are specified on the commandline, the compiler will produce a warning urging the usage of -NEWFORTRAN and its corresponding style. Otherwise, this warning will not be produced.

When using the -32IX option to compile C programs, only the -NEWFORTRAN semantics are supported. Therefore, existing 64V-Mode C programs that contain calls to FORTRAN-declared routines must be updated to the -NEWFORTRAN calling style before re-compiling them using -32IX.

See the Rev. 21.0 C User's Guide documentation for a full explanation of interfacing C routines to routines written in other languages and usage of the -NEWFORTRAN option.

Miscellaneous

All character arguments are promoted to INTs when they are received as a parameter to a function. This conforms to generally accepted C language standards. However, taking the address of the CHAR parameter will yield the address of the INT in which it is stored. If the user desires to use the address of the actual character, then the CHAR parameter should be assigned to a locally declared CHAR variable. Then, the address of the locally declared CHAR variable can be used for whatever the user desires.

If a constant value is to be passed to a function which expects that parameter to be of type pointer, then the constant value should be explicitly cast to type pointer in the function call. The particular case where this is important is in 64V mode where the constant NULL (as defined in stdio.h) is used to represent a NULL pointer value. This is because the constant will cause only TWO 16-bit half-words to be reserved on the stack for that value, rather than the necessary THREE.

PL/I G LIBRARY

The PL/I G library at Rev. 21.0 supports translation of strings in both Prime-7 and Prime ECS (via P$TRNS).

COMPILERS

This section documents changes to language compilers for Rev. 21.0. These compilers include:

- CBL
- CC
- F77
- FTN
- Prime Common LISP
- Pascal
- PMA
- VRPG

CBL COMPILER

The CBL compiler offers the following new features and compiler options:

- Prime ECS support
- INCLUDE$ search rule support
- Enhanced magnetic tape support
- Relative file enhancements for MIDASPLUS and PRISAM
- -SPACE/-TIME compiler options

Prime ECS Support

Chapter 2 of this manual describes the Prime Extended Character Set in detail. This section describes the implementations specific to CBL. Refer to the Rev. 21.0 update to the COBOL 74 Reference Guide for more information.

For verbs that allow multiple literals in their formats and contain a mnemonic, that mnemonic will cause an implied concatenation with a surrounding literal unless a separator "," or ";" is used. For example,

```
A:  DISPLAY 'hello' \(BEL).

B:  DISPLAY 'hello', \(BEL).
```

In both statements the same characters will be displayed on the terminal. However, the internal processing is different in that Statement B is processed as two separate entities, while Statement A causes a single 6-character literal to be created.

Conversely, for the statement: MOVE 'hello' \(BEL) TO X the concatenation of \(BEL) to 'hello' is performed internally. A separator after 'hello' would cause two separate entities to be recognized and would therefore be illegal because the format for a MOVE statement does not allow two entities.

For the most part, this difference in internal processing will be invisible. However, of particular interest is the STRING statement and the VALUES clause. Consider the following example:

```
MOVE 'line1' \(CR) 'line2' TO data-name.

   STRING data-name  DELIMITED BY \(CR)
          'xyz'      DELIMITED BY SIZE
          INTO x.
```

When the compiler encounters \(CR), in the STRING statement, it will continue processing 'xyz' as part of the first DELIMITED BY literal. The syntax rules for mnemonics permit a literal to follow a mnemonic. However, as this example illustrates, the intended delimiter is a \(CR), not \(CR)xyz. The compiler will report this as an error because the STRING statement will subsequently fail when the second DELIMITED clause is found without a preceding object. By inserting a comma or semicolon after the \(CR) to distinguish the two literals as two separate entities, the statement will work as intended.

Also, a VALUE clause specified for a condition-name can cause a similar ambiguity.

```
   88 condition-name1   VALUES 'A'  \(SP)  'B'.
   88 condition-name2   VALUES 'A', \(SP), 'B'.
   88 condition-name3   VALUES \(SP)  'A'  'B'.
   88 condition-name4   VALUES \(SP), 'A', 'B'.
```

Condition-name1 will be interpreted as a 3-character literal 'A B' because the mnemonic \(SP) is treated as a concatenator. However, Condition-name2 will be treated as one of 'A' or ' ' or 'B'. Condition-name3 demonstrates a partial problem. The \(SP) and the 'A' will be concatenated as one VALUE for the condition and the 'B' will be the second value. Using commas as shown in condition-name4 clarifies the intent. (The comma after the 'A' is not required as two adjacent literals will never be concatenated.)

Mnemonics follow the same coding rules as any other literal in source code. They will not be converted to their character equivalent in comments, nor acted upon inside of literals. In COPY REPLACING statements they will be acted upon according to their contextual use.

For example, with pseudo-text delimiters:

```
COPY copyfile REPLACING ==\(DOLR)== BY ==\(NUMB)==.
```

The above statement will cause all instances of \(DOLR) in the target source to be changed to \(NUMB) in the destination source. However, as a stand-alone literal:

```
COPY copyfile REPLACING \(DOLR) BY \(NUMB).
```

This statement will cause all instances of The Prime ECS character '$' in the target source to be changed to the Prime ECS character '#' in the destination source.

INCLUDE$ Search Rule Support

Files referenced in COPY statements can now be located by the CBL compiler based upon the INCLUDE$ search rules in effect for a user's environment. Files that are referenced as pathnames will be located by those pathnames; while files referenced as simple filenames will be located using INCLUDE$ search rules. For further information on the INCLUDE$ search rules refer to the Rev. 21.0 update to the COBOL 74 Reference Guide.

Enhanced Magnetic Tape Support

For all tape operations, CBL now interfaces with MAGLIB, Prime's magnetic tape subroutine library. Full compatibility with previous CBL tape operations is assured and the user does not notice any variance from previous operational procedures with the exception of improved error recovery and more explicit error messages. Some MAGLIB routines have had their argument lists extended to include an optional second error code, but the pre-Rev. 21.0 calls to these routines still work in the same way.

With this new interface, increased tape functionality is available as described below:

Variable Length Record Support for Magnetic Tape: Variable Length Record functionality offers an opportunity to save a significant amount of tape space for those applications in which records of varying sizes are being processed.

Variable Length Records will be written to tape according to the format specified in the ANSI Magtape Standard (ANSI X3.27-1978). A variable length record consists of the record control word (RCW) followed by the actual data record. The record control word is four characters long and contains a character value of the record length in bytes. The record length is considered to be the length of the data record plus the four characters of the record control word. Therefore, the maximum

size of a variable length record is restricted to 9995 characters (9999 minus the 4 character ROW). To ease in tape portability, the ROW will be written in standard ANSI ASCII. The actual data records will continue to be written using Prime's native character set.

This functionality will be available with CBL under Normal I-O and can be specified by using those options which were introduced with disk-oriented Variable Length Record functionality at Rev 20.2. As in Rev 20.2, programs compiled using the -CLDIO compiler option will not be able to utilize variable length record functionality.

Improved Error Recovery and Reporting: Error recovery in the areas of internal recovery of tape errors such as bad spots on tape will be greatly improved. Any I/O operation which fails during tape processing will be retried up to 20 times for successful execution. Only at that point, if the operation was not successfully completed, will a fatal error message be reported. Additional recovery will entail marking off any bad spots encountered and positioning the tape around the so-designated bad spot. These recovery operations will occur as part of the internal I/O operation and therefore is not visible.

In addition, more extensive error messages are available. Those error messages which are deemed recoverable will also provide an interlude whereby the user may correct the problem reported and re-start processing from where the error occurred.

Standard ANSI ASCII Level 3 Tape Label Support: Tape labels will be written in standard ANSI ASCII in the format specified in the ANSI Magtape Standard (ANSI X3.27-1978). This feature eases in the porting of tapes between Prime and other vendors systems.

Due to the new complexity and higher validation level of tape labelling being supported at Rev. 21.0, older revs of the CBL library are not capable of interpreting the tape labels now being generated. Therefore, any user who wishes to transport a labeled tape that was created on a Rev. 21.0 system back to a pre-Rev. 21.0 system will need to have a specific fix release of the CBL library installed on that (pre-Rev. 21.0) system which can successfully handle the new tape labels. Starting with the revisions 19.4.11, 20.0.5, and 20.2.1, and all subsequent fix releases, code will have been added to properly handle the new tape label format introduced at Rev. 21.0.

Relative File Rewrite Enhancement

For MIDASPLUS or PRISAM relative files in dynamic or random access modes, a READ statement is no longer required to be executed prior to the REWRITE statement. The value of the relative key will be used to determine which record is to be updated. If the file does not contain the record specified by the key, the invalid key condition exists.

The REWRITE statement does not affect the current file position when the file is being referenced sequentially.

Relative File Key Size

The relative key data item is no longer restricted to a maximum of six digits. It must reference an unsigned integer data item.

-SPACE/-TIME Compiler Options

Two new compiler options have been added in this release. These options control the preference of the kinds of optimizations that the compiler is to perform. If you use the -TIME option, then the compiler will generate object code to cause your program to execute as quickly as possible, even when the resulting object code takes more space than it might otherwise. The -SPACE option optimizes in the other direction, giving preference to making the generated code as small as possible, even if the result takes more execution time. The default is -TIME.

CC COMPILER

The CC compiler offers the following new functionality at Rev. 21.0:

- Prime Extended Character Set (ECS) Support
- INCLUDE$ Search Rules Support
- UNIX/ANSI Restriction On the Use of FOPEN

Note

The size of CCLIB.BIN (the static library that is linked with programs using the SEG linking-loader) has increased over the size of its Rev. 20.2 version. This does not affect use of the shared, dynamically-linked EPF library.

Prime ECS Support

Prime ECS support has been added to C through the use of the file PRIME_ECS_CHARS.H.INS.CC (found in each system's SYSTEM directory). The contents of this file are a number of #define statements of the form:

```
#define BEL_CHAR     '\207'
#define BEL_STR      "\207"
```

The above example shows how the ECS representation for Prime-ASCII 207 (octal) is defined. The entire Prime-ASCII character set is defined this way.

One would use the above definitions as follows:

```
#include <prime_ecs_chars.h>
  .
  .
  .
message = strcat("ATTENTION!", BEL_STR);
puts(message);
  .
  .
  .
ch = char_string[8];
if (ch == BEL_CHAR) puts("Eighth character of string is BEL.");
  .
  .
  .
```

The PRIME_ECS_CHARS.H.INS.CC file contains the full set of mnemonics available for referencing the Prime Extended Character Set. Note that all of the supplied mnemonics are in uppercase.

INCLUDE$ Search Rules Support

INCLUDE$ search rules are supported by CC at Rev. 21.0. Include file searching always proceeds in this order:

1. If the file name is delimited by "..." (double quotes), search absolutely for the specified file or pathname.

2. Search the UFDs specified in "-include" options (if any).

3. Search the UFDs specified in the user's current INCLUDE$ search rules.

4. Search the top-level directory SYSCOM.

An error is reported if the include file cannot be found by use of the above algorithm.

UNIX/ANSI Restriction on the Use of FOPEN

When a file is opened using update mode ('+' as the second or third character in the mode argument), both input and output may be performed on the associated stream. However, output may not be directly followed by input without using an intervening call to the fflush function or to a file positioning function (fseek or rewind), and input may not be directly followed by output without using an intervening call to the fflush function or to a file positioning function, unless the input operation encounters End-Of-File.

The Meaning of the Return Value of OPEN()

The integer file ID returned by OPEN() is no longer the PRIMOS file unit on which a disk file is really open. It is now a unique file handle used to identify the file to the library's low-level I/O routines. There may be cases, however, when a user must determine the PRIMOS file unit that is being used to access a disk file: the routine BIO$PrimosFileUnit() has been added for this purpose. Given a file ID returned from OPEN() or FILENO() of a "FILE *" variable, BIO$PrimosFileUnit() will return the corresponding PRIMOS file unit. For example:

```
#include <stdio.h>
int primosUnit1, primosUnit2;

int fileID = open("FileName", 2);
FILE *handle = fopen("AnotherFileName", "w");

primosUnit1 = BIO$PrimosFileUnit(fileID);
primosUnit2 = BIO$PrimosFileUnit(fileno(handle));
```

The library routine FILENO(), returns the file handle associated with a given file pointer returned by the FOPEN() routine. When a given C library routine is documented as having an argument which is the return value of OPEN(), the given routine no longer expects a PRIMOS file unit as that argument but instead expects the unique file handle returned by OPEN() or FILENO(). Any attempt to pass a PRIMOS file unit to a C library routine that expects the new file handle, will likely result in a file system error (for instance, "Unit not open.")

Permanent Restrictions

Calling Non-C Language Routines — 64V and 32IX Compatibility: As usual, interfacing to non-C language routines is done by declaring the non-C routine with the FORTRAN keyword in the calling C program. Past versions of CC (64V-Mode) have contained support for the options -OLDFORTRAN and -NEWFORTRAN, where -OLDFORTRAN was the default. These options effect the interpretation of a call to a FORTRAN-declared routine from a C program.

-NEWFORTRAN is the default. -OLDFORTRAN corresponds to the old method of calling non-C routines. If neither -OLDFORTRAN or -NEWFORTRAN are specified on the commandline, the compiler will produce a warning urging the usage of -NEWFORTRAN and its corresponding style. Otherwise, this warning will not be produced.

Note

When using the -32IX option to compile C programs, only the -NEWFORTRAN semantics are supported. Therefore, existing 64V-Mode C programs that contain calls to FORTRAN-declared routines must be updated to the -NEWFORTRAN calling style before re-compiling them using -32IX.

See the C User's Guide and update for a full explanation of interfacing C routines to routines written in other languages and usage of the -NEWFORTRAN option.

Interfacing 64V and 32IX Mode C Programs — Restrictions: Routines compiled in one mode will not interface to routines compiled in another without the use of certain command line options and/or program syntaxes. See the C User's Guide and update for a full explanation of these limitations.

Preconnection: When using the -64V option, the C compiler does not support listing file preconnection. In addition, when using the -32IX option, the compiler does not support preconnection of the source, binary or listing files.

Miscellaneous: All character arguments are promoted to INTs when they are received as a parameter to a function. This conforms to generally accepted C language standards. However, taking the address of the CHAR parameter will yield the address of the INT in which it is stored. If the user desires to use the address of the actual character, then the CHAR parameter should be assigned to a locally declared CHAR variable. Then, the address of the locally declared CHAR variable can be used for whatever the user desires.

If a constant value is to be passed to a function which expects that parameter to be of type pointer, then the constant value should be explicitly cast to type pointer in the function call. The particular case where this is important is in 64V mode where the constant NULL (as defined in stdio.h) is used to represent a NULL pointer value. This is because the constant will cause only TWO 16-bit halfwords to be reserved on the stack for that value, rather than the necessary THREE.

Installation and Build Procedures

CC is supplied in the directory CC. To install CC, run CC>CC.INSTALL.COMI, followed by SYSTEM>CC.SHARE.COMI.

F77 COMPILER

The F77 compiler includes the following features and options as of Rev. 21.0:

- Prime ECS support, including the -ECS option
- INCLUDE$ search rule support
- SHORTCALL functionality in I-mode
- Longer string constants
- SHORTCALL statement stack size
- Format edit descriptor enhancement

Prime ECS Support

Chapter 2 of this manual describes the Prime Extended Character Set in detail. This section explains implementations specific to F77. Refer to the Rev. 21.0 update to the FORTRAN 77 Reference Guide for more information.

For the F77 Intrinsic Function, CHAR, current behavior always causes the high bit (bit1) to be turned on. Thus, 40(octal) and 240(octal) arguments both return the SPACE character. With the support of a larger character set, each of those arguments can produce a unique character. Current behavior will remain as the default. It will be required of users that they request suppression of such behavior via a new command line option. The option, -ECS, will cause all unique arguments to the CHAR function to return unique characters.

INCLUDE$ Search Rule Support

Files referenced in $INSERT and INCLUDE statements can now be located by the F77 compiler based upon the INCLUDE$ search rules in effect for a user's environment. Files that are referenced as pathnames will be located by those pathnames; while files referenced as simple filenames will be located using INCLUDE$ search rules. For further information on the INCLUDE$ search rules refer to the Rev. 21.0 update to the FORTRAN 77 Reference Guide.

F77 does not support INCLUDE$ location of files with default (i.e., non-explicitly stated) suffixes. Full filenames must still be used.

SHORTCALL Functionality in I-Mode

F77 now supports the use of the SHORTCALL statement when compiling for I-mode architectures. The availability of general registers leads to a more complex argument-passing algorithm. For details, please consult the Rev. 21.0 Assembly Language Programmer's Guide documentation.

Longer String Constants

F77 now allows a character literal string to contain up to 1019 characters.

-MAX_GROWTH_PERCENT [decimal_integer]

This option allows the user to specify suggested limits to the growth of the program due to optimization. The decimal number is the desired limit to the growth (increase) in code size in units of percent of the original code size. The default is 100, and no operand implies 0. The optimizer uses this limit in a number of ways. For example, when expanding subprograms inline, it keeps track of the percent increase of its internal representation of the program and when that percent increase reaches this limit, further inline expansion is inhibited. Thus it can be seen that this limit is only a suggested limit, and there can be no guarantee that the growth in program size will not actually exceed the specified limit.

-MAX_SUB_STATEMENTS_INLINE [decimal_integer]

This new option provides another way to control inline procedure expansion. The decimal integer is the maximum number of executable statements that a subprogram may have while still being eligible for inline expansion. Some source statements, such as a logical IF, count as two executable statements. The precise meaning of "statement" is

the same as the places where breakpoints could be placed using DBG. The default value for this parameter, if the option is not supplied, is determined by the System Administrator, but is 20 as supplied by Prime. If the option is specified without a decimal value then a value of 0 is used. This effectively disables inline expansion.

Format Edit Descriptor Enhancement

The ability to use counts greater than 255 with format edit descriptors has been added to FORMAT statements to match the functionality available within runtime formats.

FTN COMPILER

The FTN compiler supports the following new features and options:

- Prime ECS support
- Default code generation
- Format edit descriptor enhancement

Prime ECS Support

Chapter 2 of this manual describes the Prime Extended Character Set in detail. Refer to the Rev. 21.0 update to the FORTRAN Reference Guide for more information about the specifics of ECS for FTN.

Default Code Generation

The FTN compiler now generates 64V-mode code by default. Users wishing to produce R-Mode binaries must now use an explicit command line option (-32R).

Format Edit Descriptor Enhancements

The ability to use counts greater than 255 with format edit descriptors has been added to FORMAT statements to match the functionality available within runtime formats.

PRIME COMMON LISP

Prime Common LISP is a new product that requires PRIMOS Rev. 20.0 or greater. This description of Rev. 1.0 contains the following sections:

- Configuration information
- Memory requirements
- ANSI Common LISP
- Standard Common LISP
- Environment
- Installation and Build Procedures

Configuration Information

Prime Common LISP users should be configured to a minimum of 256 dynamic segments. Machines running Prime Common LISP must have adequate paging configured. Paging for each average user should be approximately 10 Mb. Users who run larger heaps than the default will require more paging area.

Prime Common LISP comes with an EMACS interface for those sites that have EMACS. Chapter 2 in this manual contains more information on the EMACS interface.

At revisions earlier than 21.0, the EMACS interface is installed as part of the install file, and EMACS should be reshared after Prime Common LISP is installed. At Rev. 21.0, the EMACS interface is part of EMACS, and it is no longer necessary to reshare EMACS after installing Prime Common LISP.

> WARNING
>
> If the system has insufficient paging space a number of adverse things may occur, including having the system halt. If a Prime Common LISP user gets a stack overflow (condition STACK_OVF$) or fatal error in crawlout message then a paging device full error must be suspected. You will not see the PAGING_DEVICE_FULL$ signal printed on the Prime Common LISP user's terminal. This is due to circumstances beyond the control of the user's process. This underscores the importance of configuring sufficient paging room.

Memory Requirements

Approximate memory requirements for Prime Common LISP have been determined. The system needs a minimum of 4 Mb of physical memory for a single Prime Common LISP user. Each additional user will require approximately 2 Mb of additional physical memory. These numbers refer to users actively running Prime Common LISP on the system.

The search rule mechanism in PRIMOS is used for several things. There are some implications for resources used when the system has Prime Common LISP on it. The default installation, which puts the Prime Common LISP library in the ENTRY$ search rules, may not be the best way to run Prime Common LISP on your system (this will use significant dynamic segments for each user, even those who don't run Prime Common LISP). The suggested alternative is to have each user who wishes to run Prime Common LISP add the following lines (or equivalent if you have changed the installation) to a personal ENTRY$.SR file:

```
LIBRARIES*>LISP.RUN
LIBRARIES*>LISP_SHARED.RUN
LIBRARIES*>LISP_LM.RUN
```

This is activated by the SET_SEARCH_RULE command (SSR). To display the ENTRY$ search rules, use the LIST_SEARCH_RULE command (LSR ENTRY$).

ANSI Common LISP

Prime is participating in the recently formed ANSI committee for Common LISP (X3J13) to determine any potential incompatibility between ANSI Common LISP and Prime Common LISP.

Side effects or results unspecified by Steele in Common LISP the Language will probably be specified in the future standard, and these results may not agree with any particular current implementation. Users of Prime Common LISP should therefore not rely on unspecified results or side effects when designing portable code.

Users of Prime Common LISP should also avoid the practice of mixing the function and value name spaces. The future ANSI Common LISP standard may not allow the function LIST, for example, to be assigned a value.

Standard Common LISP

To ensure conformity with standard Common LISP for creating portable code, take the following steps:

- Create a separate package for the application.
- Do not use the system package.

All the functions and variables defined in Common LISP the Language by Guy Steele are accessible from the LISP package. All new packages (as created by make-package and in-package) use only the LISP package. See the PRIME Common LISP Language Reference Manual for further discussion of packages.

Environment

Prime Common LISP requires PRIMOS Revision 20.0 or greater; EMACS users must have EMACS revision 20.0.3 or greater. PRIME Common LISP runs on machines with 32-IX mode/CGRR support. This includes all 2xxx series machines except the 2250 and all 9xxx series machines. No 3-digit machines include this support (such as 250 ,450, etc.).

Installation and Build Procedures

Prime Common LISP asks several questions during installation, shown below.

Full site name: enter your location, such as Prime Computer Inc

Short site name: an abbreviated site name, such as Prime

Machine version: the model number, such as 9950

Machine instance: your network node name, or your short site name repeated

Time zone: minutes before GMT (because the world has time zones with fractional hours). The installation lists all time zones including those outside the continental United States.

Because Prime Common LISP is run during its installation, the system console or user who installs this product must have sufficient dynamic segments.

To install this product from a user terminal, take the following steps before the installation:

- Use the SET_PRIORITY_ACCESS (SPAC) command to give the user ID ALL access to the MFDs containing the directories SYSTEM, SEARCH_RULES* and LIBRARIES*, and optionally to EMACS* if the site has EMACS.
- Make sure the partition containing the directory LIBRARIES* has at least 4000 free records.
- Make sure that the user has sufficient dynamic segments. A minimum of 256 dynamic segments is recommended for running Prime Common LISP.

The rest of the installation is automatic.

PASCAL COMPILER

The changes to the Pascal compiler at Rev. 21.0 are detailed below.

Extended String Search Rule Support

Files referenced in $INSERT and INCLUDE statements can now be located by the PASCAL compiler based upon the INCLUDE$ Search Rules in effect for a user's environment. Files that are referenced as pathnames will be located by those pathnames; while files referenced as simple filenames will be located using INCLUDE$ Search Rules. For further information on the INCLUDE$ Search Rule Mechanism, see the PRIMOS Commands Reference Guide available at Rev 21.0.

Changes for Validation

A number of changes have been made at Rev. 21.0 to continue to bring Pascal into compliance with ANS standards. The following are those features which may have some user-visible consequences:

1. Program statements are necessary to declare the INPUT and OUTPUT files. If the standard files INPUT and OUTPUT are not used, the program statement may still be omitted.

2. When the program statement is present, it must conform to the syntax and semantics of the standard. Files occurring in the heading are now considered "external." This means they can be opened with a 1 parameter reset/rewrite. "Internal" files (not in heading) may still be connected to external (PRIMOS) files by use of nonstandard 2 parameter reset/rewrite. "Internal" files not connected to PRIMOS files by a 2 parameter reset/rewrite are temporary and will be deleted when they are no longer in use.

3. CLOSE is now optional and need not (and probably ought not) be used. Files are closed automatically.

4. Leading zeroes in labels and constants are disregarded.

5. "(." and ".)" are now interpreted as equivalent to "[" and "]".

PMA ASSEMBLER

The PMA assembler contains enhancements in the following areas as of Rev. 21.0:

- The MIP pseudo-op
- Determining modes of variables and expressions
- Symbol table
- Assembler listing
- Prime ECS support
- General register relative format
- IX-mode instructions

The MIP Pseudo-op

Some programs include IPs that are subsequently modified during program execution. Such pointers should not be placed in the linkage(Z) frame. To allow the distinction to be made, a new pseudo-op, MIP, has been introduced. MIP stands for Modifiable Indirect Pointer. It is used exactly like the IP pseudo-op. However, any pointer defined using MIP is placed into the link frame, whether or not the SPLIT(_LINKAGE) operand is present.

Determining the Mode of a Variable or Expression

A variable or expression preceded by the character '[' now represents the mode of that variable or expression. This makes it possible to determine within a macro whether an argument of the macro has absolute mode or is link or procedure relative, for example. The possible modes are:

0 - ABSOLUTE
1 - RELATIVE
2 - COMMON
3 - EXTERNAL
4 - STACK RELATIVE
5 - PB-ABSOLUTE
6 - LB-RELATIVE
7 - XB-RELATIVE

Note that a [-expression may not be part of another expression. Thus, to determine if the variable XYZ is stack-relative, for example, would require the following instructions:

```
MODE    XSET   [XYZ
        IF MODE.EQ.4,.....
```

The Symbol Table

The assembler now places the symbol table in segments 4001 and up. This makes it possible to assemble very large programs that previously could not be assembled using PMA.

The Assembler Listing

The following changes have now been made in the assembler listing:

- IF statements occurring within macros are now printed together with the code they generate when LSMD is in effect.

- The current value of the program counter is now printed alongside each macro call when NLSM is in effect. This can be helpful in debugging large programs consisting of a sequence of macro calls.

- When NLSM is in effect, the printing of local variables in the concordance is suppressed. It is possible to assemble a program with NLSM and still get the local variables printed in the concordance. This can be done by putting an LSMD statement directly before the END statement. Similarly, in a program

assembled with LSMD or LSTM, an NLSM instruction directly before the END statement will have the effect of suppressing the printing of local variables in the concordance. In this case, it is important to include LSMD or LSTM explicitly at the beginning of the program. Otherwise, the NLSM instruction encountered by the assembler at the end of Pass 1 of the assembly will still be in effect during Pass 2.

Prime ECS Support

Chapter 2 of this manual describes the Prime Extended Character Set in detail. This section explains implementations specific to PMA. Refer to the Rev. 21.0 update to the Assembly Language Programmer's Guide for more information.

Special characters can only be used in quoted strings in literals or in DATA and BCI and BCZ statements.

One exception are instructions of the form

```
BCI    n,.....
```

where the character string is not enclosed in delimiters. Such a character string may not contain special characters.

Because of its use in designating special characters, the backslash may no longer be used as a character string delimiter in a BCI or BCZ instruction.

General Register Relative Format

The assembler now recognizes the general register relative (GRR) format in I-mode memory reference instructions. (GRR format can only be used on 2550 CPU's and up).

In an instruction of the form

```
ST 3,R5%+6
```

the effective address of the second operand is determined by adding 6 to the contents of general register 5. In assembling this type of instruction, the B-field in the op-code is set to 3, the S-field contains the number of the general register being referenced (5 in the case of our example), and the T-field is set to zero.

The general register relative format cannot be combined with indexing or indirect addressing.

It is possible to define general register relative address constants, and then to use them in memory reference instructions, as in the following:

```
X       EQU     R6%+7
        L       3,X
        ST      3,X+2
```

In the listing file, general register relative addresses are indicated by use of an 'R' suffix.

IX-mode Instructions

The assembler now supports the following IX-mode instructions: AIP, LIP, ACP, CCP, DCP, ICP, LCC, SCC, TCNP. (See the Instruction Sets Guide for Rev. 21.0 for detailed information on these instructions.)

VRPG COMPILER

The VRPG compiler contains the following new features as of Rev. 21.0:

- Prime ECS support
- INCLUDE$ search rule support

Prime ECS Support

Chapter 2 of this manual describes the Prime Extended Character Set in detail. This section explains the implementations specific to VRPG. Refer to the Rev. 21.0 update to the RPG II V-Mode Compiler Reference Guide for more information.

Two of the specification formats in VRPG will allow the new extended character syntax. These two statements are the Calculation Specification and the Output Specification.

Any operations in a Calculation Specification which can involve an alphanumeric literal will allow the \(<mnemonic>) format to be used. Factors 1 and 2 are the fields affected. The operations and the fields of these operations allowing the new syntax are:

DSPLY, DEBUG, LOKUP, SETLL:
Factor 1 may contain the new syntax.

MOVE, MOVEA, MOVEL, MHHZO, MHLZO, MLHZO, MLLZO:
Factor 2 may contain the new syntax.

COMP:
Factor 1 and Factor 2 both may contain the new syntax.

Columns 45-70 of an Output Specification may contain one or more of the "extended string" syntax constants. The \(<mnemonic>) format constants may be juxtaposed and, therefore, concatenated with each other and/or with quoted string constants.

Following is an example of the new extended character syntax used in a VRPG program:

```
H                                                    T
FINPUT   IP AF  80  80           DISK
FOUTPUT  O   F  40  40           PRINTER
IINPUT       01
I                                      1   6 TOTAL
I                                      1   1 AAA
I                                      2   5 AMOUNT
C                    SETON                      02
C           AAA      COMP \(DOLR)               030405
C  N05               MOVEL\(DOLR)   TOTAL
OOUTPUT  H       1P
O                                   10 \(LBKT)'Debits'\(RBKT)
O                                   21 \(LBKT) 'Credits' \(RBKT)
OOUTPUT  D       03
O                        TOTAL      10
OOUTPUT  D       04
O                        TOTAL      10
OOUTPUT  D       05
O                        TOTAL      20
```

This program reads in values and prints them in two columns: under the Credits column if the first character of the value is a dollar sign and under the Debits column otherwise. With the following input:

```
$100
-500
-300
$600
```

the output would be:

```
[Debits]  [Credits]
            $100
  $500
  $300
            $600
```

INCLUDE$ Search Rule Support

INCLUDE$ search rules can be used to retrieve include files when compiling VRPG programs. If a user has INCLUDE$ search rules in effect during compilation of a VRPG program which uses the %INCLUDE or /COPY syntax, the VRPG compiler will use this pathname search list to locate the source to be inserted. Files referenced by complete pathnames in %INCLUDE or /COPY lines will be located by that pathname, as in the past. For further information on the INCLUDE$ search rules refer to the Rev. 21.0 update to the RPG II V-Mode Compiler Reference Guide.

UTILITIES

This section describes changes to BIND, DBG, and the Common Envelope as of Rev. 21.0.

BIND

As of Rev. 21.0, the BIND command has two new subcommands:

- COMPRESS
- INITIALIZE_DATA (IDATA)

COMPRESS

COMPRESS compresses the run file by expunging information not used in execution. This includes BIND's symbol table and DBG's information for debugging the program. As a result, the program may not be reloaded and maps may not be generated from existing compressed EPFs. Also, programs may be executed in DBG, but all source information is lost; variable evaluation, listing of source lines, and other functions that DBG performs may not be done. This option is intended for debugged production copies of programs. The result is a smaller file. COMPRESS has no options.

INITIALIZE_DATA

INITIALIZE_DATA (abbreviation: IDATA) permits a developer to initialize all uninitialized areas for debugging purposes. The command format is:

```
INITIALIZE_DATA [-OCTAL] <initial-value>

OR

IDATA [-OCT] <initial-value>
```

where <initial-value> is a decimal value in the range -32768 to 32767 or, if the -OCTAL option (abbreviation: -OCT) is given, an octal number from 0 to 177777 that the uninitalized areas are to be set. Since this will generate a larger runfile which takes longer to start, it is not recommended that this be used for production copies.

COMMON ENVELOPE

The Common Envelope has the following new functionality as of Rev. 21.0:

- Prime ECS support
- Improved temporary stack allocation for character varying strings
- Increased implementation limit of initializing static character constant strings, from 512 to 1020

Note

The following files are no longer needed and should be deleted if they exist:

```
LIB>BACKEND(V,I).BIN

LIBRARIES*>(CODEGENV,CODEGENI,OPTIMIZER,CODEGEN_COMMON).RUN
```

DBG SUPPORT FOR PRIME ECS

Rev. 21.0 DBG supports the Prime Extended Character Set (ECS) syntax for all languages. As part of this support, evaluation of character strings constants using implicit concatenation is now allowed. This enables you to group two or more character constants together to form a resulting concatenation of the two constants. For example, in PL/1

```
'now' || 'together'
```

can be expressed as

```
'now''together'
```

yielding the character string 'nowtogether'.

CHANGES TO DOCUMENTATION

This section describes corrections to the following two manuals:

- The SEG and LOAD Reference Guide
- The RPG II V-Mode Compiler Reference Guide

Changes to RPG II V-Mode Compiler Reference Guide

In addition to the options described in the reference guide, VPRG also supports the following two options:

- -MAPWIDE
- -MAXERRORS

-MAPWIDE Compiler Option: The format of the -MAPWIDE option is

```
-MAPWIDE [decimal_integer]
```

-MAPWIDE specifies the width in number of characters of the cross-reference map that appears in the listing file, as well as the width of the options list that appears at the beginning of the listing file. The valid range of values for decimal_integer is from 80 to 160 inclusive. The default width of the cross-reference map, if -MAPWIDE is not specified is 80, provided that a listing is being produced. The default width if -MAPWIDE is specified without an argument is 108.

The abbreviation for the -MAPWIDE option is -MAPW.

-MAXERRORS Compiler Option: The format of the -MAXERRORS option is

```
-MAXERRORS [decimal_integer]
```

-MAXERRORS specifies the maximum number of compilation errors to be reported. If in a given compilation the specified maximum is reached, then an error message is issued and the compilation is aborted. The valid range for the number of errors to be reported, specified by decimal_integer, is 1 to 32767.

The default maximum number of errors, if -MAXERRORS is not specified, remains 100; if -MAXERRORS is specified without a decimal argument, the maximum number of errors that can be reported is 32767.

The abbreviation for the -MAXERRORS option is -MAXE.

Changes to SEG and LOAD Reference Guide

This subsection describes changes to the SPLIT, MIX, and STACK commands of the SEG LOAD subprocessor.

SPLIT Command: SPLIT commands to SEG's LOAD subprocessor must specify a stack start address of at least 4. (The stack root uses the first 4 words of any stack to find the next location.) Thus, the following SPLIT command is not correct:

```
SPLIT 170000 4002 1
```

and results in a runfile that does not execute. (This SPLIT command sets the start address of the stack at word 1.)

MIX Command: The MIX command of SEG's LOAD subprocessor does not always reduce program size, especially if it is used with the SPLIT command.

STACK Command: The STACK command of SEG's LOAD subprocessor does not work with the SPLIT command. The stack must be explicitly allocated using a SPLIT command.

CHAPTER 4

DATA MANAGEMENT PRODUCT ENHANCEMENTS

Prime's Data Management systems have a number of enhancements at Rev. 21.0. This chapter outlines the enhancements to the following:

- ROAM
- DBMS
- DISCOVER
- MIDASPLUS
- POWERPLUS
- PRISAM

ROAM ENHANCEMENTS

- Multiple disk Save/Restore
- Increased ROAM buffers
- Default value for the number of ROAM buffers
- Support for PRISAM Remote Transactional Access
- Improved runtime notification to save After Image file

Multiple Disk Save/Restore

ROAM now allows you to have a single backup disk online at a time when using SAVE_RBF or RESTORE_RBF on a database or PRISAM file that exists on two or more disks. You issue the SAVE_RBF or RESTORE_RBF command from a user terminal, so that the supervisor terminal is free for you to use the ADDISK command on the backup disks. The command syntax is identical to the syntax used when all backup disks are simultaneously online, but some additional prompts appear.

This facility requires no changes to the command line format of SAVE_RBF and RESTORE_RBF. There are some differences, however, in the dialog that saves or restores the entire database.

For further details and sample output, see the Rev 21.0 edition of the ROAM Administrators Guide.

Increased ROAM Buffers

The ROAM Buffer Pool may now contain up to 1024 buffers. For busy systems with a large number of users, this reduces competition for buffers and increases throughput. The use of more ROAM buffers is recommended for those who have a relatively large amount of memory available on their system.

The number of ROAM buffers may be increased by invoking ROSAU and executing the BUFFERS command. Acceptable values are now between 32 and 1024, inclusive.

Default Value for the Number of ROAM Buffers

If ROAM's configuration file (ROAM*>ROAM.CONFIG) does not contain a value for NUM_BUFFERS (the constant which specifies the number of ROAM buffers to configure) at initialization, IROAM initializes it with a default value of 256. This situation may occur if you have upgraded from 19.4 PRISAM or DBMS and have not executed the ROSAU BUFFERS command.

At previous revisions, IROAM displayed the message "Bad NUM_BUFFERS in config file, defaulting to maximum number" each time it was run. The message has been changed to "Initializing NUM_BUFFERS to 256" and is displayed only the first time IROAM is run. The default value may be changed with the ROSAU BUFFERS command.

Support for PRISAM Remote Transactional Access

ROAM now supports PRISAM remote transactional access. The Distribution Manager component of ROAM provides PRISAM users with the ability to access PRISAM files at a single remote site within a transaction. For more details on PRISAM remote transactional access, please see the revision 21.0 updates to the PRISAM User's Guide.

Most of this support in ROAM is invisible to PRISAM users, although several new files have been added to the ROAM install directory: ROAM*>DDM.ERR, SYSTEM>DDM2600, SYSTEM>DDM4000, and LIB>DDMLIB.BIN. In addition, ROAM creates the file DDM_ERROR_LOG in ROAM*>USER when ROAM detects the first remote transaction. All remote transaction errors detected by ROAM are logged in this file. See the section Components of the ROAM System below for a description of these files.

Improved Runtime Notification to Save After Image File

ROAM now periodically reminds PRISAM and DBMS users to save the After Image file by sending a message to the supervisor terminal: "The After Image file needs to be saved." Previous versions of ROAM would display this message only once. ROAM now displays this message when the After Image file becomes 20% full (as before) and again displays the message thereafter as the After Image file becomes another 10% full, and at each additional 10% increase.

Support for Prime Extended Character Set (ECS)

ROAM now supports the Prime Extended Character Set (ECS) at Rev. 21.0.

TPDUMP Directive and ROAM

Ordinarily, warmstarting the system does not recover ROAM files. The only exception is when a forced shutdown has caused the halt and the TPDUMP directive is enabled in the PRIMOS configuration file. If this is the case, PRIMOS displays the following at the supervisor terminal:

```
*** From PRIMOS:  Please take a crash dump, then warmstart.
```

Warmstarting after this message has been displayed completes the disk I/O that was unfinished at the time of the forced shutdown. However, you must still perform a coldstart once the warmstart is complete. For more details, see the CPU handbook for your machine.

Restrictions

Because all linkage areas for static mode libraries are statically allocated, you can have only one program EPF using the same library at any given time. Therefore, you cannot access ROAM from more than one EPF level at the same time. This is an EPF restriction on static mode libraries in general.

The following tape recovery functions are not available to users of the 4585F cartridge tape drive or the 5 and 1/4 inch tape drive used on the Prime 2350 and 2450 systems:

1) automatic saving of after-images to tape (enabled through the AUTOSAVE command of ROSAU)

2) manual save of after-images to tape (SAVE AIFILE command of ROSAU) and roll-forward recovery from tape (ROLLF command of ROSAU).

Manual save and roll-forward recovery remain available for on-disk after-image recovery. An attempt to use one of these restricted functions with the 4585F cartridge tape drive or the 5 and 1/4 inch tape drive used on the Prime 2350 and 2450 systems results in the following error message: "Requested function is invalid for this tape controller. Please use another tape device."

Environment: Rev. 21.0 ROAM requires PRIMOS Rev. 21.0 and associated PRIMOS Utilities. ROAM requires the use of shared segments 2200 through 2203, 2217, 2220, 2223, and 2224. ROAM also requires the use of private segments 6007 (words 0 through '47777), 6006 (words '100000 through '177777), and 6011. The Distribution Manager component of ROAM requires the use of shared segments 2600 and 2601.

Components Of The ROAM System

The ROAM top-level directory contains the following files:

- ROAM.INITINSTALL.COMI -- used to install ROAM for the first time
- ROAM.INSTALL.COMI -- used to install ROAM after its initial installation
- ROAM.INSTALL.CPL -- used to install ROAM after its initial installation
- ROAM.SHARE.COMI -- used to share the ROAM shared segments

ROAM>CMDNC0: This directory holds the CPL files to those V-mode segment directories found in SEGRUN* which become external commands to PRIMOS. These files are moved to CMDNC0 by ROAM.INSTALL.COMI:

CLUP.SAVE	CN_RBF.CPL	COPY_RBF.CPL	DELETE_RBF.CPL
IROAM.CPL	LIST_RBF.CPL	REST_RBF.CPL	ROSAU.CPL
SAVE_RBF.CPL	SET_RBF.CPL	RESTORE_RBF.CPL	LEM.RUN

ROAM>HELP*: This directory contains Help files for ROSAU commands and the Help file for the LEM command.

ROAM>INFO: This directory contains the documentation for the current release of ROAM, the files ROAM.RUNI and ROAM.RUNO.

ROAM>LIB: This directory contains the ROAM binary object libraries: ROOFFLIB.BIN (the ROAM offline library), RORUNLIB.BIN (the ROAM runtime library), and DDMLIB.BIN (the Distribution Manager component of ROAM).

ROAM>ROAM*: This directory contains the following file system objects:

5 Files: REV_NUM, ROAM.ERR, ROLLB.CPL, TAPE_PH.CPL, DDM.ERR
2 Segment Directories: ROLLB.SEG, TAPE_PH.SEG
4 Directories: HELP, TOOLS, USER, TRACE

REV_NUM is a text file provided for the internal use of PRIMEWAY. ROAM.ERR is the ROAM error message file. ROLLB.CPL and ROLLB.SEG are the CPL file and the V-mode segment directory, respectively, for the ROLLB (Before-Image Recovery) phantom. TAPE_PH.CPL and TAPE_PH.SEG are the CPL file and the V-mode segment directory, respectively, for the TAPE_PH phantom (used to automatically save the After Image file). DDM.ERR is the error message file for the Distribution Manager component of ROAM.

ROAM>ROAM*>HELP: This directory contains Help files for ROAM commands. This currently consists of RO_TRACE_EVENTS, RO_DUMP_EVENTS and ROSAU. The directory ROSAU.UFD contains help files for all ROSAU subcommands.

ROAM>ROAM*>HELP>ROSAU.UFD: Each file in this directory contains information corresponding to a ROSAU subcommand of the same name, for example, the file ALLOCATE contains information on the ROSAU ALLOCATE subcommand.

ALLOCATE	AUTOSAVE	BUFFERS	CLEAR
CREATE	FIX	HELP	INSTALL
LAR	LIST_ACTIVE_RBFS	LOCK	MEND
MOVE	QUIT	RECON	RESET
ROLLF	SAVE	STATUS	UNLOCK
WAIT			

ROAM>ROAM*>TOOLS: The TOOLS directory contains ROAM tools for system analysts.

ROUTL.CPL	RO_TRACE_EVENTS.SEG	RORUNLIB.MAP	ROOFFLIB.MAP
ROUTL.SEG	RO_DUMP_EVENTS.SEG	CNV_BIFILE.SEG	
SET_IROAM_NOT_INIT.SEG			

The ROUTL.CPL and ROUTL.SEG are the CPL file and V-mode segment directory, respectively, for the ROUTL tool. The RO_TRACE_EVENTS.SEG and RO_DUMP_EVENTS.SEG are V-mode segment directories for the ROAM Monitor trace tool. Both tools are used to trace and dump runtime ROAM. THE RORUNLIB.MAP and ROOFFLIB.MAP are ROAM runtime and offline shared library maps. The CNV_BIFILE.SEG is V-mode segment directory for before image file conversion and reversion.

ROAM>ROAM*>USER: The USER directory contains the ROAM system logs and also contains any user system-error logs that might be created.

The ROAM system log is a record of ROAM events that have occurred. Each log is kept for seven days, with the log name in the format ROAM.mmddyy.LOG, where mmddyy is the date the log was created.

The system-error user logs are named in the format USER ##-time-date.LOG and contain valuable information about the state of ROAM when a system error occurs. If you should get a system error, spool both the user log and the system log (ROAM.mmddyy.LOG) for your Prime analyst.

The TRACE directory contains text files describing trace options and contains trace output.

ROAM>SEGRUN*: This directory contains the following V-mode segment directories corresponding to commands in ROAM>CMDNC0:

CLUP.SEG	CN_RBF.SEG	COPY_RBF.SEG	DELETE_RBF.SEG
IROAM.SEG	LIST_RBF.SEG	REST_RBF.SEG	ROSAU.SEG
SAVE_RBF.SEG	SET_RBF.SEG		

ROAM>SYSTEM: This directory contains the ROAM shared segment run files. The RRxxxx files correspond to the ROAM Runtime shared segments, the ROxxxx files correspond to the ROAM Offline shared segments and the DDMxxxx files correspond to the Distribution Manager component of ROAM shared segments.

RO2203	RO4000	RR2200	RR2201
RR2220	RR4000	DDM2600	DDM2601
DDM4000			

INSTALLING ROAM

The following two procedures describe how to install ROAM for the first time and how to reinstall it.

Note

Although you can install Rev. 21.0 ROAM on partitions formatted for either Rev. 19, Rev. 20, or Rev. 21, you must use a partition formatted for Rev. 20 or a higher revision, if you wish to create Contiguous Access Method (CAM) files. CAM files were introduced at Rev. 20 as an alternative file type for certain ROAM files that previously could only be DAM files. See the ROAM Administrator's Guide for information about choosing file types.

Initial Installation

To install ROAM for the first time, do the following:

1. Set a priority ACL on all partitions that you will be saving.
2. Use the BACKUP or MAGSAV command to save to tape all partitions that you want to reformat for the current revision of PRIMOS. For extra security, you may wish to save all partitions on the system.
3. Install PRIMOS. See the Rev. 21.0 Software Installation Guide.
4. Restore the tape containing ROAM.
5. Execute the ROAM initial install program, by entering:

```
CO ROAM>ROAM.INITINSTALL.COMI
```

   This program creates the top-level ROAM* directory on the command device and copies the ROAM system subdirectories to the appropriate top level PRIMOS directories, such as CMDNCO.

6. Restore all partitions you backed up.
7. Edit the user profiles to add the .ROAM_ADMIN group and, if DBMS will be installed, the .DBMS_ADMIN group.
8. If you do not want the ROAM* directory in the system partition, move it with the PRIMOS COPY command. Once ROAM is shared, moving this directory requires a more elaborate procedure.

9. Unless the system date and time are already set, set them so that the command procedure in the next step can execute properly.

10. Share ROAM from the supervisor terminal by entering the following:

```
CO SYSTEM>ROAM.SHARE.COMI
```

Note

You can disregard certain error messages that appear when this share procedure invokes IROAM. The messages indicate you have not created the ROAM system files. Using the INSTALL command in step 12 corrects this situation. When you use the share procedure again in step 13, initializing ROAM should not generate any error messages. See the ROAM Administrator's Guide for further information on initializing ROAM.

11. Before you use the INSTALL command in step 12, you must determine how you will respond to various prompts this command issues. These prompts ask you about creating the ROAM system files and configuring the ROAM buffer pool. At this point, you must consider the following five questions:

    a. Do you want to create an After-image File (AIFILE)? Assuming you will routinely make interactive updates, you should create an AIFILE. The AIFILE tracks transactions so that you can reapply them if they are lost through a media failure.

    b. Where will you place the Recovery Table (RCVTAB), the Before-image File (BIFILE), and the after-image file (AIFILE)? Careful placement of these files optimizes system performance and recovery from a storage media failure. The following rules apply to file placement:

        - To optimize recovery, you should keep the BIFILE in one partition and the AIFILE and the RCVTAB either in a second partition or in second and third partitions.

        - To optimize recovery, the partition containing the AIFILE should be different from the partition you plan to use for your DBMS and PRISAM data files. Ideally, the AIFILE should be on a different disk drive from the one containing the data files.

- To optimize I/O performance, the BIFILE's partition should be on a different disk drive than the drive(s) used by the AIFILE, RCVTAB, and your DBMS and PRISAM data files.

If an ideal file arrangement is not currently feasible, you can use the ROSAU utility later to move any of the system files after you have completed this procedure.

c. If you are placing the before-image file in a partition formatted for Rev. 20 or a higher revision, do you want to create it as a CAM file? Using CAM rather than DAM as the BIFILE file type may result in performance improvement. However, you should experiment on a test system to determine which file type is best for your system.

d. How many pages (blocks or physical records) do you want to initially allocate to the BIFILE and the AIFILE? At installation, allocate to each file the number of blocks required to contain 10 per cent of all data in all PRISAM and DBMS files on your system. In other words, if your system will have three PRISAM files and one DBMS file that together require 2500 data blocks, you would allocate 250 pages to the BIFILE and 250 pages to the AIFILE. Remember that each data block contains 2048 bytes, 2030 of which are available for data.

You can later increase or decrease the size of either file with the ROSAU ALLOCATE command. Allocating additional space for the BIFILE is necessary if the system displays BIFILE overflow messages at runtime. Allocating additional space for the AIFILE allows you to save the AIFILE less often.

e. How many buffers do you want in the ROAM buffer pool? This number can be from 32 to 1024. Choosing an appropriate number of buffers for your system optimizes data access speed. The ROAM Administrator's Guide provides guidelines for determining an appropriate number of buffers and describes how to set or reset this number.

12. After making the decisions outlined in step 11, you can use the INSTALL command, which is part of the ROAM System Administrator Utility (ROSAU). To use ROSAU's INSTALL command, you must:

a. Invoke ROSAU by entering ROSAU.

b. Enter INSTALL and respond to the prompts based on your decisions in step 10. When prompted for a treename, enter only a fully qualified pathname; that is, a pathname that begins with an MFD.

An example of the INSTALL command follows. User input is shaded. In this example, the disk specified for the BIFILE is a Rev. 21 disk. Therefore, the system asks whether the BIFILE should be a CAM or DAM file. (If a Rev. 19 disk had been used, this prompt would not appear and ROAM would automatically create a DAM BIFILE.) When done, the INSTALL command tells the user to rerun IROAM; IROAM is one of the commands in the command input file executed in step 12.

```
> install
This command will attempt to lock the system.
The ROAM file system is locked.
Creating the Recovery Table.
Enter new fully qualified UFD treename.
File will be named RCVTAB in this UFD.
> <part1>roam*
Creating the Before Image File.
Enter new fully qualified UFD treename.
File will be named BIFILE in this UFD.
> <part2>north
Please enter the file type for the BIFILE.
(CAM or DAM)
> cam
Please enter the size of the Before Image File in
pages.
> 50
Do you want an After Image file to be created? yes
Enter new fully qualified UFD treename.
File will be named AIFILE in this UFD.
> <part3>south
Please enter the size of the After Image file in
pages.
> 50
Please enter the number of buffers to configure in
the buffer pool:
> 1024
All the specified system files have been
successfully created.  Please rerun IROAM to enable
access to the new system files.
```

c. Exit ROSAU by entering Q or QUIT.

13. Share ROAM again from the supervisor terminal by entering:

```
CO SYSTEM>ROAM.SHARE.COMI
```

No error messages should result from sharing ROAM this time.

Reinstallation

To upgrade ROAM from a previous revision, perform the following steps:

1. Determine which partitions you want to reformat to the current revision or PRIMOS. Remember, you can only have CAM files on partitions formatted for Rev. 20 or a higher revision.

2. Set a priority ACL on all partitions that you will be saving. Since RBF slaves may be on a partition different from the RBF master, you will need access to both of them at the time the master is being saved.

3. To allow proper use of the ROAM system files, you must back up the partitions you are reformatting in a specific order. If you are not saving any partitions containing system files, you can omit this step and instead perform step 4. To save ROAM system files to tape, you can only use MAGSAV.

    a. Use ROSAU STATUS to locate the RCVTAB and the AIFILE.

    b. Create three logical tapes by using MAGSAV with full pathnames on the following objects in the order shown below. You reduce the chance of creating duplicate system files by using full pathnames.

        - The RCVTAB and the file ROAM*>ROAM.ERR
        - All other partitions you want to reformat
        - The AIFILE

        Saving files and partitions in the preceding order allows you to restore them in an order that accommodates ROAM requirements. Details on these requirements appear later in this procedure.

4. If you did not back up any partitions containing system files, and therefore omitted step 3, use the BACKUP or MAGSAV command to save to tape all partitions that you want to reformat for the current revision of PRIMOS. For extra security, you may wish to save all partitions on the system.

5. Install PRIMOS by following the installation instructions in the current edition of the System Administrator's Guide, Volume I: System Configuration.

6. Restore the tape containing the new revision of ROAM. This software overwrites the existing software.

7. Execute the ROAM reinstallation program, by entering:

```
CO ROAM>ROAM.INSTALL.COMI
```

8. If you saved the RCVTAB and the AIFILE, you must restore these files in the specific order outlined below, after recreating the directories that contained them. If you did not save these files, skip down to step 9. Use the following procedure to correctly recreate these directories and restore these files:

   a. Recreate the directories that contained the RCVTAB and the AIFILE, and recreate the ROAM* directory.

   b. Restore the RCVTAB and ROAM*>ROAM.ERR.

      You restore the RCVTAB and ROAM.ERR first, so that the RBFs can be restored properly. If you instead restore RBFs first, you need to use the ROSAU MEND on each RBF, once the RCVTAB is restored.

   c. Restore the third logical tape: the last created copy of the AIFILE.

      You specifically need this copy, since it contains saved-file records created when RBFs were backed up to the second tape. The copy of the AIFILE on the disk you restore in the next step does not overlay the copy you restore in this step, since tape utilities can not delete an on-disk copy of a ROAM system file.

9. Restore all the partitions you have not yet restored. Use the -CAM_RBF option if you want to restore the subfiles as CAM files. With both BACKUP_RESTORE and MAGRST, this option causes all subfiles of RBFs to be restored as CAM files, regardless of the file type saved on the tape. Omitting the -CAM_RBF option causes each subfile to be restored with the file type it had when saved, either CAM or DAM.

10. Unless the system date and time is already set, set it so that the command procedure in the next step can execute properly.

11. Share ROAM from the supervisor terminal by entering:

```
CO SYSTEM>ROAM.SHARE.COMI
```

If you have never configured the ROAM buffer pool, a feature introduced at Rev. 20, the share procedure configures it to the the default size of 256 buffers, when IROAM is invoked, and ROAM generates the following message:

```
Initializing NUM_BUFFERS to 256
```

As of Rev. 21.0, you can reconfigure the buffer pool to any number between 32 and 1024. Choosing the right number of buffers for your system optimizes data access speed. The ROAM Administrator's Guide provides guidelines for determining an appropriate number of buffers for your system and describes how to change the current number by using the ROSAU BUFFERS command.

12. Share ROAM again by entering:

```
CO SYSTEM>ROAM.SHARE.COMI
```

13. Roll forward all the RBFs that have after-image recovery enabled to remove the write access lock on these RBFs.

14. If you are upgrading from a pre-Rev. 20 system and need to convert your before-image file to a CAM file, follow the procedure in Appendix G of the ROAM Administrator's Guide.

DBMS ENHANCEMENTS

DBMS has enhancements at Rev. 21.0 in the following areas:

- DMLCP
- IDBMS
- DML

DMLCP

The portion of the DBMS runtime system which handles access to area files was revised at Rev. 21.0 to improve maintainability and extendability. The only functional change is that space within area buckets is better utilized.

IDBMS

The minimum number of buffers to allocate at start transaction has been changed from 2 to 3. The default number of buffers is changed from 3 to 4. The DBA should make sure that the number of buffers is set to at least 3. DBMS will not work correctly if less than 3 buffers are allocated at start transaction.

DML

A new command line option, -DYNAMIC (abbreviation: -DY), has been added to the DML compiler at Rev. 21.0. Intended for PRIMEWAY users, this option causes the preprocessor to generate a minor code of 2, which invokes a schema by name rather than a schema number obtained at compile time. This Dynamic Invoke feature does not maintain any specific correspondence to a schema name and its structure. It is your responsibility to ensure the proper correspondence of the schema structure assumed by the DML program and the actual structure of the schema at run time.

Terminology: The term Dynamic Invoke provides a means to invoke a schema by name rather than a schema number obtained at compile time. The schema number is dynamically resolved at run time based on the schema name.

Example: All DML commands syntactically use the -DYNAMIC option in the same manner. Following is an example of the CBLDML command with the -DYNAMIC option:

```
OK, CBLDML FORMAT-78.CBLDML -DY
[CBLDML Rev. 21 Copyright (c) Prime Computer, Inc. 1987]
processing input file - FORMAT-78.CBLDML
processing output file - FORMAT-78.COBOL
processing error file - FORMAT-78.ERROR
Pre-processing successfully completed
     0 ERRORS
     0 WARNINGS
OK,
```

DBMS Installation and Build Procedures

Rev. 21.0 DBMS requires PRIMOS Rev. 21.0, ROAM Rev. 21.0 or (PRIMOS Rev. 20.2 and ROAM Rev. 20.2) and SEG Rev. 19.0 or greater. DBMS runtime (DMLCP) requires the use of shared segments 2001, 2002, 2003 and 2012 as well as private segments 4030, 4031, 4032, 4033 and 4034.

Please note that when you restore this revision from tape, a file type mismatch occurs if these existing disk files are of type DAM:

```
DBMSCOB>CMDNC0>CDML.CPL
DBMSCOB>JOBS>CSUBS_LOAD.CPL
DBMSEX>JOBS>VERSIO.FTN
```

To prevent the mismatch error from occurring, delete the existing DAM files before restoring. If these files are already of type SAM, then no action is necessary.

Files On System Tape

The following sections outline the file system objects supplied with the install tape.

DBMSEX: This top-level directory is the primary packaged directory supplied on the system tape. It is the prerequisite for all the others. By itself, DBMSEX is adequate to allow for the execution only of DBMS. Its directory structure is followed by each of the other packaged directories. (Exceptions are noted where appropriate.)

The three files at the top level are the install procedures DBMS.INSTALL.CPL and DBMS.INSTALL.COMI, plus the share procedure DBMS.SHARE.COMI. The install file, DBMS.INSTALL.COMI, can now be invoked as a COMINPUT file. It executes the old install file DBMS.INSTALL.CPL, which is still a CPL routine. The install files for each of the individual DBMS packaged directories remain as CPL routines.

DBMSEX>DBMSLB: The shareable run files, segment directories, and support utilities are listed below. They are moved to the top-level directory DBMSLB by DBMS.INSTALL.

```
DB2001 DB2002 DB2003 DB2012 DB4000
DB2070 VFYPRT.SAVE DUMP.SEG IDBMS.SEG
DBACP.SEG DBUTL.SEG SUMMARY.SAVE DAEERS
DBMSE
```

DBMSEX>CMDNC0: This directory holds the CPL interludes, which become commands external to PRIMOS, to the V-mode products found in DBMSLB. These interludes are moved to the top level DBMSLB by DBMS.INSTALL. and are listed below.

```
DBACP.CPL DBUTL.CPL
```

DBMSEX>BINARY: Each sub-product in a package has a binary file in this directory in the format prod.BIN. In addition, the following files are produced by LOAD_LIB:

DBDATA.BIN VERSIO.BIN LIBRARY.MAP

DBMSEX>JOBS: This directory contains the procedures (in the form prod_LOAD.CPL) which produce the run files or segment directories in DBMSEX>DBMSLB for this package. The DMLCP routine is called LOAD_LIB.CPL and it requires the following additional files:

DBDATA.FTN VERSIO.FTN HTAB.INS.FTN DYNT.PMA

JOBS also contains the following two utilities for setting up special configuration options at the time of the install and automating reloads of subproducts.

SETUP.CPL RESET_UFD.CPL

DBMSEX>INFO: This directory contains the documentation for the current release of DBMS.

DBMSDEF: This directory has a similar structure to DBMSEX. It is packaged with DBMSEX for systems on which schemas are developed; the sub-products contained herein are SCHEMA and SCHDEC. The install file DBMSDEF.INSTALL.CPL is also contained here (top level).

DBMSDEF>DBMSLB: This directory contains the following file system objects:

SCHEMA.SEG SCHDEC.SEG

DBMSDEF>CMDNC0: This directory contains the following file system objects:

SCHEMA.CPL SCHDEC.CPL EXEC.SAVE

DBMSDEF>BINARY: This directory contains the following file system objects:

SCHEMA.BIN SCHDEC.BIN EXEC.BIN

DBMSDEF>JOBS: This directory contains the following file system objects:

SCHEMA_LOAD.CPL SCHDEC_LOAD.CPL EXEC_LOAD.CPL

DBMSFTN: This directory is packaged with DBMSDEF and DBMSEX for those systems on which FTN applications are developed. It contains the sub-products FSUBS and DML as well as the LIB entry point file DMLLIB. The only file at the top level is DBMSFTN.INSTALL.CPL.

DBMSFTN>DBMSLB: This directory contains the following file system objects:

FSUBS.SEG DML.SEG

DBMSFTN>CMDNC0: This directory contains the following file system objects:

FSUBS.CPL FDML.CPL

DBMSFTN>LIB: This directory contains the following file system objects:

DMLLIB.BIN

DBMSFTN>BINARY: This directory contains the following file system objects:

FSUBS.BIN DML.BIN

DBMSFTN>JOBS: This directory contains the programs FSUBS_LOAD.CPL and FDML_LOAD.CPL. Also, it contains the two utilities FDML.CPL and FLOAD.CPL for use in precompiling and loading applications programs. (See the section Creation Of A DML Application Program below.) They are moved to the top level directory DBMSLB by the install procedure.

DBMSF77: This directory is packaged with DBMSDEF and DBMSEX for F77 application development systems. Its sub-products include F77SUBS and DML as well as the LIB entry point file DMLLIB. The only file at the top level is DBMSF77.INSTALL.CPL.

DBMSF77>DBMSLB: This directory contains the following file system objects:

F77SUBS.SEG DML.SEG

DBMSF77>CMDNC0: This directory contains the following file system objects:

F77SUBS.CPL F77DML.CPL

DBMSF77>LIB: This directory contains the following file system objects:

DMLLIB.BIN

DBMSF77>BINARY: This directory contains the following file system objects:

F77SUBS.BIN DML.BIN

DBMSF77>JOBS: Here in addition to F77SUBS_LOAD.CPL and F77DML_LOAD.CPL we have the two utilities F77DML.CPL and F77LOAD.CPL for use in precompiling and loading applications programs. (See following section CREATION OF A DML APPLICATION PROGRAM.) They are moved to the top level directory DBMSLB by the INSTALL procedure.

DBMSCOB: This directory is packaged with DBMSDEF and DBMSEX for COBOL application development systems. The sub-products include CSUBS and DML as well as the LIB entry point file DMLLIB. The only file at the top level is DBMSCOB.INSTALL.CPL.

DBMSCOB>DBMSLB: This directory contains the following file system objects:

CSUBS.SEG DML.SEG

DBMSCOB>CMDNC0: This directory contains the following file system objects:

CSUBS.CPL CDML.CPL

DBMSCOB>LIB: This directory contains the following file system objects:

DMLLIB.BIN

DBMSCOB>BINARY: This directory contains the following file system objects:

CSUBS.BIN DML.BIN

DBMSCOB>JOBS: This directory contains the CSUBS_LOAD.CPL and CDML_LOAD.CPL programs. It also contains the two utilities CDML.CPL and CLOAD.CPL for use in precompiling and loading applications programs. (See the section Creation Of A DML Application Program below.) They are moved to the top level directory DBMSLB by the install procedure.

DBMSCBL: This directory is packaged with DBMSDEF and DBMSEX for those systems on which the new CBL (ANSI 1974 COBOL) compiler is installed and will be used to develop applications. Thus we include the sub-products CBLSUBS and DML as well as the LIB entry point file DMLLIB. The only file at the top level is DBMSCBL.INSTALL.CPL.

DBMSCBL>DBMSLB: This directory contains the following file system objects:

CBLSUBS.SEG DML.SEG

DBMSCBL>CMDNC0: This directory contains the following file system objects:

CBLSUBS.CPL CBLDML.CPL

DBMSCBL>LIB: This directory contains the following file system objects:

DMLLIB.BIN

DBMSCBL>BINARY: This directory contains the following file system objects:

CBLSUBS.BIN DML.BIN

DBMSCBL>JOBS: This directory contains the programs CBLSUBS_LOAD.CPL and CBLDML_LOAD.CPL. Also, it contains the utilities CBLDML.CPL and CBLLOAD.CPL for use in precompiling and loading applications programs. (See the section Creation Of A DML Application Program below.) They are moved to the top level directory DBMSLB by the install procedure.

DBMSLGCL: This directory is packaged with DBMSDEF and DBMSEX for use on systems with the schema editor. The install file SCHED.INSTALL.CPL is here at the top level.

DBMSLGCL>DBMSLB: This directory contains the following file system objects:

SCHED.SEG

DBMSLGCL>CMDNC0: This directory contains the following file system objects:

SCHED.CPL

DBMSLGCL>BINARY: This directory contains the following file system objects:

SCHED.BIN TEXTED.BIN

DBMSLGCL>JOBS: This directory contains the following file system objects:

SCHED_LOAD.CPL

INSTALLING DBMS

If your site uses DBMS, use one of the following procedures, depending on whether you are performing an initial installation or a reinstallation.

Initial Installation

After installing the current version of ROAM using the preceding procedure, you can install DBMS for the first time by doing the following:

1. Restore the DBMS directories supplied on the system tape. These may be one or more of the following:

```
DBMSEX    DBMSDEF   DBMSFTN   DBMSF77
DBMSCOB   DBMSCBL   DBMSLGCL
```

2. Attach to the MFD where you want DBMS to reside and enter the following:

```
CREATE DBMSLB
```

3. If you want to inhibit an introductory message when you run a DBMS program, run SETUP.CPL in DBMSEX>JOBS. This utility asks you if you want the message, makes the appropriate change to VERSIO.FTN, recreates the DMLCP shared segments, and tells you to reshare DBMS. You will reshare DBMS in step 5 below.

4. To install DBMS, enter the following from the supervisor terminal:

```
CO DBMSEX>DBMS.INSTALL.COMI
```

5. Share DBMS from the supervisor terminal by entering:

```
CO SYSTEM>DBMS.SHARE.COMI
```

6. Delete the directories you restored from the system tape in step 1. DO NOT delete any other DBMS directories.

7. Edit the ACL for the MFD containing ROAM* to give U access to all DBMS users.

8. Edit the user profiles to add the .ROAM_ADMIN and .DBMS_ADMIN groups.

Reinstallation

After installing the current version of ROAM using the preceding procedure, you can upgrade DBMS from a previous revision by doing the following:

1. If you are upgrading from Rev 18.2 or later, delete the DBMSxxxxBIN packaged binary directories. These directories are no longer on the system tape and thus will remain with outdated binaries on your system until manually deleted.

2. Restore the DBMS directories supplied on the system tape. These may be one or more of the following:

   DBMSEX DBMSDEF DBMSFTN DBMSF77
   DBMSCOB DBMSCBL DBMSLGCL

3. If you want to inhibit an introductory message when you run a DBMS program, run SETUP.CPL in DBMSEX>JOBS. This utility asks you if you want the message, makes the appropriate change to VERSIO.FTN, recreates the DMLCP shared segments, and directs you to reshare DBMS. You will reshare DBMS in step 5.

4. To install DBMS, enter the following from the supervisor terminal:

   ```
   CO DBMSEX>DBMS.INSTALL.COMI
   ```

   As of Rev 19.0, the error message files DAERRS and DBMSE are copied by the INSTALL procedure from DBMSEX>DBMSLB to the directory DBMSLB.

5. Share DBMS from the supervisor terminal by entering:

   ```
   CO SYSTEM>DBMS.SHARE.COMI
   ```

6. Delete the directories you restored from the system tape in step 2. DO NOT delete any other DBMS directories.

Data Administrator Authorization

Every DBMS site must have the following User Profile Groups defined:

.ROAM_ADMIN ROAM data administrators
.DBMS_ADMIN DBMS data base administrators

Persons belonging to the group .DBMS_ADMIN are authorized as valid data administrators. Without such membership, a user may not use any of the DBACP commands which alter a database or display sensitive information (such as privacy keys). Persons who also belong to the .ROAM_ADMIN group are classified privileged administrators. These are data administrators who may bypass the various schema privacy locks when using DBACP. A privileged data administrator is responsible for the management and integrity of DBMS.

Reloading Products

There are times when a specific sub-product of DBMS needs to be reloaded, for example, a segment directory might need to be recreated. To do this, use the same job streams which were used in the original building of the components of DBMS, and start with the packaged directories delivered on your system tape.

These job streams are in the JOBS directories of the corresponding packages and the names are in the format prod_LOAD.CPL.

Note

If you have replaced any of the libraries (ULIB, CLIB, ILIB, TEXTED, DMLCP, ASI, or ASG), you should reload all the sub-products which use them in their loading procedures. If you would like to automate the remaining reloads, you can use RESET_UFD.CPL (see the section Files on System Tape).

The load procedures for sub-products produce segment directories in the directory DBMSLB within their respective packages. To install a reloaded segment directory, copy it up to the top level directory DBMSLB. The shareable segments produced by LOAD_LIB.CPL should also be copied to the top level DBMSLB: these segments do not come into use until the next time the DBMS.SHARE.COMI command stream is run.

DMLCP Installation

DMLCP requires the exclusive use of shared segments 2001, 2002, 2003, and 2012 and private segments 4030, 4031, 4032, 4033 and 4034. To install the shared library version of the DML command processor, the following command must be executed from the supervisor terminal:

```
CO SYSTEM>DBMS.SHARE.COMI
```

This command stream installs the DBMS shared library, shares and initializes the DBMS segments, and initializes the Ring 3 semaphores. This command should be incorporated into PRIMOS.COMI (or C_PRMO), the command file which is always run after a cold start.

Creation Of A DML Application Program

Once a schema and subschema have been written and compiled and the database files have been allocated with DBACP, you can write application programs for the database in either COBOL or FORTRAN. Here is the sequence used to transform the source code into executable code:

1. Preprocess the source code with the host language preprocessor (FDML, CDML, F77DML or CBLDML).

2. Compile the output of the preprocessor (xxxxx.FTN, xxxxx.F77, xxxxx.COBOL, or xxxxx.CBL) with the host language compiler.

3. Link the binary output of the compiler to the DML command processor with the segmented loader SEG.

Command procedures that perform these operations with either a COBOL, CBL, FTN or F77 program are found in the directory DBMSLB and are called CDML.CPL, CLOAD.CPL, CBLDML.CPL, CBLLOAD.CPL, FDML.CPL, FLOAD.CPL, F77DML.CPL, and F77LOAD.CPL. (The installation utility selects these out of the JOBS directory of the appropriate package if supplied on the system tape and copies them to the top level DBMSLB.)

For example, to compile and load a COBOL program called "PROG", execute the following command:

```
R DBMSLB>CDML PROG
```

To compile and load a CBL program called "PROG", execute the following command:

```
R DBMSLB>CBLDML PROG
```

To compile and load a FORTRAN program called "PROG":

```
R DBMSLB>FDML PROG
```

To compile and load a F77 program called "PROG":

```
R DBMSLB>F77DML PROG
```

The output files created when using CDML.CPL, CBLDML.CPL, FDML.CPL, or F77DML.CPL on the source file "PROG" are:

```
PROG.LIST - The preprocessor and compiler listings.
PROG.BIN  - The binary file output by the compiler.
```

The output files from using CLOAD.CPL, CBLLOAD.CPL, or FLOAD.CPL with program "PROG" are:

```
PROG.MAP  - SEG program map.
PROG.SEG  - The segmented run file.
```

The resulting user program is executed with the command:

```
SEG PROG
```

DISCOVER ENHANCEMENTS

The DISCOVER query language contains the following enhancements:

- Item-to-item comparison for PRISAM files
- QUIT from "ENTER CR" prompt
- Prime Extended Character Set (ECS) Support
- Number of Abbreviations in DISCOVER Increased

Item-to-Item Comparison For PRISAM files

Previously, DISCOVER only supported the comparison between an item and a literal, for example:

```
SELECT ALL from Emp-relation WHERE Emp.salary >= '45000'
```

At Rev. 21.0, the WHERE clause of the SELECT, MODIFY, or ERASE command supports the comparison of two data items when operating on PRISAM records and relations, for example:

```
SELECT ALL FROM Emp-relation WHERE Emp.salary >= Manager.salary
```

The data items being compared must be of the same basic type (character or numeric). This feature is not available for DBMS records.

QUIT from "Enter CR to continue" prompt

DISCOVER displays the "Enter CR to continue" prompt when a command returns more then one screenful of information. At Rev. 21.0, you may type "Q" or "QUIT" to abort the execution of the command. Previously, you needed to type CONTROL-P to abort the command.

Prime Extended Character Set (ECS) Support

DISCOVER now supports the Prime-ECS character set. You can:

1. Display data items extracted from both DBMS and PRISAM databases that contain new characters supported by Prime-ECS, for example, a capital A character with a circumflex (^) accent.

2. Store and modify data items that contain the new characters in PRISAM databases.

3. Define and display reports with header text that contains the new characters.

4. Include the new characters in strings used as selection criteria within SELECT commands.

Number of Abbreviations in DISCOVER Increased

At Rev. 21.0, the number of abbreviations allowed has increased from 100 to 256.

Environment

DISCOVER requires Rev. 21.0 DBMS, PRISAM, ROAM, and PRIMOS.

Installation and Build Procedures

DISCOVER installation and build procedures are standard, except for the items outlined in the following sections.

Upgrading from VISTA to DISCOVER: Users upgrading from VISTA to DISCOVER must use the DISCOVER_UPGRADE program in the directory DISCOVER_TOOLS to make their VISTA formats, procedures, and abbreviations available to DISCOVER. The instructions in the file DISCOVER_UPGRADE.RUNO in the directory DISCOVER_TOOLS must be followed to insure a successful upgrade from VISTA to DISCOVER. If you are not upgrading from VISTA, you may ignore this section.

Recompiling VISTA Formats: Users with formats compiled by a revision of VISTA prior to Rev. 19.2.3 must recompile all of these formats before they will work with DISCOVER. The program RECOMPILE, contained in the directory DISCOVER_TOOLS, recompiles all public and private formats in the DISCOVER catalog and eliminates this problem. Instructions for using this program are in the file RECOMPILE.RUNO in the directory DISCOVER_TOOLS.

DISCOVER Configuration File: DISCOVER uses a configuration file that defines several parameters that define the way DISCOVER interacts with users, as well as the directories that contain the DISCOVER catalogs and Help database. This file is DISCOVER.CONFIG in the directory DISCOVER_DBMS>SYSTEM, and it is copied to the directory SYSTEM by the installation procedure. If you have made any changes to this file for prior releases of DISCOVER, enter them into DISCOVER_DBMS>SYSTEM>DISCOVER.CONFIG before performing the installation, or your changes will be lost.

The DISCOVER configuration file, SYSTEM>DISCOVER.CONFIG, consists of 21 lines plus a 4 line header. Each line must be exactly as described in these instructions or DISCOVER will not run predictably. The file format consists of the information required by DISCOVER followed by an optional comment on each line. A comment begins with "/*" and ends at the end of the line. The information on each line is as follows:

LINES 1-4: Configuration file header. These lines are ignored by DISCINIT.SAVE; they help document the file. Do not delete these lines: if they are deleted, DISCINIT will still ignore the first 4 lines of the file and will lose necessary information.

LINE 5: The number of characters per line on the type of terminal you are using to run DISCOVER. This number should be 1 character less than the actual screen width to avoid unwanted automatic linefeeds. Note: This number should be greater or equal to 71 for optimum performance of DISCOVER. (Default = 79)

LINE 6: The number of lines per screen on the type of terminal you are using to run DISCOVER. This number should be 1 less than the actual screen length to allow for the scrolling prompt.
(Default = 23)

LINE 7: The number of characters per logical line on the printer you are using with DISCOVER. (The number of characters on the line after the printer has inserted its side margins.)
(default = 132)

LINE 8: The number of lines per logical page on the printer you are using with DISCOVER. (The number of lines on the page after the printer has inserted its top and bottom margins.)
(default = 66)

LINE 9: The maximum number of characters per line written to a file when you use the FILE option of the PRINT command.
(default = 132)

LINE 10: The maximum number of lines per page written to a file when the FILE option of the PRINT command is used.
(default = 66)

LINE 11: This line is reserved for future expansion, and should be left blank.

LINE 12: This line is reserved for future expansion, and should be left blank.

LINE 13: The name of the master DISCOVER directory.
(default = DISCOVER*)

LINE 14: The owner password of the master DISCOVER directory.
(default = '')

LINE 15: The owner password of the DISCOVER CATALOG directory (where the procedures, formats and abbreviations are stored).
(default = '')

LINE 16: The master directory of the DISCOVER HELP subsystem files.
(default = DISCOVER*)

LINE 17: The owner password of the master HELP directory. (Note: If the default master DISCOVER directory name (DISCOVER*) is used and the default master DISCOVER HELP Subsystem directory name (DISCOVER*) is also used, then the passwords on this line and line 14 must be the same, since the directories they apply to are so named.)
(default = '')

LINE 18: The DISCOVER HELP subdirectory (the actual data files of the HELP subsystem reside here).
(default = HELP)

LINE 19: The DISCOVER HELP subsystem directory owner password.
(default = '')

LINE 20: The DISCOVER HELP subsystem top-level prefix. Since the HELP subsystem prints the actual directory name where it is currently located, it deletes the topmost (passworded) directory names and their passwords from the HELP subsystem header. This prefix replaces the deleted portion.
(default = HELP DISCOVER)

LINE 21: The scrolling default: Set to "1" for SCROLL ENABLED, set to "0" for SCROLL DISABLED.
(default = 1)

LINE 22: The number of records retrieved between printing the record count. Note: If you use DISCOVER with hardcopy terminals, it is suggested that you set the default number as "1". To disable by default the printing of the record count, the System Administrator can create a public DISCOVER "STARTUP" procedure containing the command "DISABLE RECORD COUNT". This procedure enables you to avoid the constant overwriting of the record count.
(default = 1)

LINE 23: The number of "expected" tokens to print after a syntax error. When DISCOVER finds a token it didn't expect, it prints what it did expect, up to the maximum number of tokens set by this variable. Users with slow or hard-copy terminals may want to set this to lower than the default value to speed up DISCOVER after syntax errors.
(default = 48)

LINE 24: The name of the public and/or private DISCOVER procedures automatically executed when entering DISCOVER.
(default = STARTUP)

LINE 25: The number associated with the level of update security you want this DISCOVER installation to have. Any or all of the update commands can be restricted by changing the value of this parameter. It must be set to the sum of the keys for the functions that you wish to allow, from this list: ERASE = 1, MODIFY = 2, STORE = 4.
(default = 7, all UPDATE commands enabled)

PRISAM SELECT Performance Issues

This section describes the behavior of the SELECT command when used on PRISAM files, and what steps can be taken to create more efficient SELECT statements.

The performance of a query statement depends on a number of factors. The best way to predict performance is to follow some general guidelines, discussed below. Be aware that some of the guidelines are release-dependent and specifics may change as the product develops, but the intent of each guideline remains the same.

Relation Selection Optimizations

The FROM clause in the SELECT, MODIFY, and ERASE commands specifies the relation which will be processed. Choosing an "appropriate" relation for a query is important for best performance.

Single-record relations are established by the USE command. The command "USE FILE CUSTOMER", assuming that file CUSTOMER is a PRISAM file with a single record of the same name, would establish one single-record relation named CUSTOMER. The commands "USE FILE PRODUCT" and "USE FILE ORDER", given similar assumptions, would establish two additional single-record relations, PRODUCT and ORDER. You should note that single-record relations are usually referred to as records in most of the DISCOVER documentation.

Multi-record relations are built by executing one or more JOIN commands. The first JOIN command that is executed on a given relation establishes the base record for the relation, names the relation, and sets up a default processing sequence for the records in the relation:

```
JOIN RECORD CUSTOMER TO RECORD ORDER ~
     ON CUSTOMER-ID = ORDER-CUSTOMER-ID ~
     GIVING RELATION CUSTOMER-ORDER
```

The record named in the "TO RECORD" clause tells DISCOVER that ORDER is the base record for this relation. The relation is named by the "GIVING RELATION" clause. The default processing sequence is base record ORDER followed by first joined record CUSTOMER. All other records that are part of this relation will be joined directly to the base record or indirectly to the base record through one or more intermediate records.

JOIN commands which use a previously-named relation add records to the named relation:

```
JOIN RECORD PRODUCT TO RELATION CUSTOMER-ORDER ~
     ON PRODUCT-CODE = ORDER-PRODUCT-CODE
```

The above command adds another record PRODUCT to an existing relation CUSTOMER-ORDER. The PRODUCT record is appended to the end of the default processing sequence.

The two JOIN commands above establish a relation named CUSTOMER-ORDER with a base record of ORDER and a default process sequence of ORDER record, CUSTOMER record, PRODUCT record.

Now consider the following commands:

```
JOIN RECORD PRODUCT TO RECORD ORDER ~
     ON PRODUCT-CODE = ORDER-PRODUCT-CODE ~
     GIVING RELATION PRODUCT-ORDER


JOIN RECORD CUSTOMER TO RELATION PRODUCT-ORDER ~
     ON CUSTOMER-ID = ORDER-CUSTOMER-ID
```

These two JOIN commands establish a relation named PRODUCT-ORDER which is similar to the relation CUSTOMER-ORDER, but PRODUCT-ORDER has a default process sequence of ORDER record, PRODUCT record, CUSTOMER record.

If you followed the command sequence above, there would be three single-record relations (CUSTOMER, ORDER, and PRODUCT) and two multi-record relations (CUSTOMER-ORDER and PRODUCT-ORDER) defined to DISCOVER which could be used in the FROM clause. When choosing which of the above is the most appropriate for a given query, you should apply the following rules:

1. Chose the relation that has the smallest number of records and the smallest PRISAM record descriptions that contain all the data you want and all of the qualifying conditions that you are placing on the data.

2. If more than one relation satisfies Rule 1, look at the default processing sequence of the relations that satisfies Rule 1 and, if there is a WHERE clause, choose the relation that will reject occurrences of the relation as early as possible along the default processing path. If there is no WHERE clause, choose the relation that will accept occurrences of the relation as early as possible along the default processing path.

3. Always be aware of implied conditions when using a multirecord relation. "SELECT CUSTOMER FROM CUSTOMER" tells DISCOVER to find each record occurrence in CUSTOMER and to copy it into the current table, while "SELECT CUSTOMER FROM CUSTOMER-ORDER" tells DISCOVER to find each record occurrence in ORDER, then to find the CUSTOMER occurrence that is joined to that ORDER occurrence, and finally to copy that occurrence of CUSTOMER into the current table.

   The first SELECT copies one occurrence of each CUSTOMER into the current table. No CUSTOMER occurrence is skipped and none are duplicated, assuming that none are duplicated in the CUSTOMER file. The second SELECT copies one occurrence of the joined CUSTOMER into the current table for each occurrence of ORDER in the ORDER file. CUSTOMERS are skipped if there is no joined ORDER and, CUSTOMERS that have more than one joined ORDER are duplicated.

Examples: The following examples are based upon the relations defined above.

1. Find all of the data about all of the CUSTOMERS.

```
SELECT ALL FROM CUSTOMER
```

   The multi-record relations CUSTOMER-ORDER or PRODUCT-ORDER could have also been used, but they do not satisfy Rule 1.

2. Find all of the data about PRODUCTS which are out of stock.

```
SELECT ALL FROM PRODUCT WHERE PRODUCT-ONHAND = 0
```

   The multi-record relations CUSTOMER-ORDER or PRODUCT-ORDER could have also been used but they do not satisfy Rule 1.

3. Find the CUSTOMER-ID and PRODUCT-CODE for all ORDERS.

```
SELECT CUSTOMER-ID, PRODUCT-CODE FROM CUSTOMER-ORDER
```

   or

```
SELECT CUSTOMER-ID, PRODUCT-CODE FROM PRODUCT-ORDER
```

   Both are equally appropriate because all three records must be processed to satisfy the query and there is no WHERE clause to aid in selecting the best relation.

4. Find the CUSTOMER-ID for all ORDERS.

```
SELECT CUSTOMER-ID FROM CUSTOMER-ORDER
```

   CUSTOMER-ORDER is appropriate because of the implied condition. PRODUCT-ORDER could also be used but the PRODUCT would have to be processed unnecessarily (see Rule 2). Note that there are duplicate CUSTOMER-IDs if an occurrence of CUSTOMER has more than one ORDER joined to it. The REMOVE DUPLICATES command can be used to eliminate any duplicates.

5. Find the CUSTOMER-ID for all customers ordering PRODUCT AB123.

```
SELECT CUSTOMER-ID FROM PRODUCT-ORDER ~
       WHERE PRODUCT-CODE = 'AB123'
```

   PRODUCT-ORDER is appropriate because of the implied condition, and DISCOVER can reject occurrences as soon as it has processed ORDER (second on the default process list for PRODUCT-ORDER) without processing the CUSTOMER (third on the list). CUSTOMER-ORDER could also be used but DISCOVER would always process the CUSTOMER (second on the default process list for CUSTOMER-ORDER) before it processed PRODUCT. CUSTOMER occurrences that had not ordered the product would be processed. The same duplicate condition may occur as above.

Data Ordering Optimizations

DISCOVER takes advantage of the processing sequence of data whenever it is possible to do so. Consider the last example above when DISCOVER was used to find the CUSTOMER-ID of all CUSTOMERS who had ordered PRODUCT AB123.

If ORDER is sorted in the sequence ORDER-CUSTOMER-ID ORDER-PRODUCT-CODE, DISCOVER takes advantage of having already processed CUSTOMER and/or PRODUCT for previous occurrence of ORDER (if they have the same ORDER-CUSTOMER-ID and/or ORDER-PRODUCT-CODE as the current occurrence of ORDER). Given the command "SELECT CUSTOMER-ID FROM PRODUCT-ORDER WHERE PRODUCT-CODE = 'AB123'", DISCOVER processes an ORDER occurrence, then the PRODUCT occurrence joined to the ORDER and, assuming that it is indeed product AB123, DISCOVER then processes the CUSTOMER joined to ORDER.

Having completed processing of that occurrence of ORDER, DISCOVER saves the PRODUCT and CUSTOMER that it has just processed, and proceeds to the next occurrence of ORDER. If the next occurrence of ORDER has the same CUSTOMER and/or PRODUCT joined to it as the previous ORDER had, DISCOVER performs a substantially shorter processing cycle on CUSTOMER and/or PRODUCT because of the saved data.

PRISAM Key Optimizations

When executing the SELECT, MODIFY or ERASE commands, DISCOVER first finds an occurrence of the base record in the relation, then finds the additional record occurrences which are joined (directly or indirectly) to that occurrence of the base record. To find occurrences of the base record, DISCOVER either scans the base record sequentially, or performs keyed reads on the base record. The sequential read always works, but can be quite inefficient when the total occurrences of the base record far outnumber the occurrences actually requested.

DISCOVER attempts to use a keyed read on the base record if a WHERE clause has been specified. This succeeds only if at least one key applies to the entire WHERE clause, and if the conditions on such a key are useful (for example, the conditions "PRICE < 10.00 OR PRICE > 9.00" together cover all possible values but they are not useful for reducing the number of occurrences read). To use a keyed read:

1. Manipulate the WHERE clause so that it has the general form

   ```
   (condition AND condition AND ... AND condition) OR
   (condition AND ...) OR (condition ...) OR ...
   ```

   Call each AND term a "group".

2. For each group, list the names that are defined as keys in the base record, and do not use the ^= (NOT EQUAL) comparison operator.

3. Mark the names that appear in all the lists from step 2. These are candidates for performing a keyed read. If there are no names that appear in all the lists from step 2, then a sequential read must be performed. Otherwise, a keyed read is possible (subject to the notion of "usefulness" mentioned above).

Examples: Assume there is a PRISAM file OFFICE that contains the keys OFFICE-ID and ZIP, and a non-key data item STATE.

1. SELECT ALL FROM OFFICE WHERE OFFICE-ID = 'MILLINOCKET'

   The single group specifies the key OFFICE-ID. This query is a candidate for a keyed read.

2. SELECT ALL FROM OFFICE WHERE STATE = 'NM'

   No key is specified in this query; a sequential read will be used.

3. SELECT ALL FROM OFFICE WHERE STATE = 'MA' AND ZIP < 02100

   The single group specifies the key ZIP. This query is a candidate for a keyed read.

4. SELECT ALL FROM OFFICE WHERE ~
   OFFICE-ID = 'SOUTHWEST' OR ~
   OFFICE-ID = 'HARTFORD'

   Both groups specify the key OFFICE-ID; a keyed read is possible.

5. SELECT ALL FROM OFFICE WHERE ~
   OFFICE-ID = 'PITTSFIELD' OR ZIP > 69999

   The first group specifies the key OFFICE-ID, the second specifies the key ZIP. Neither key appears in both groups, so a sequential read must be used.

6 SELECT ALL FR OFFICE WHERE STATE = 'CT' OR ZIP <= 02299

   The second group specifies the key ZIP, but the first group does not specify any key. A sequential read must be used.

7. SELECT ALL FR OFFICE WHERE ~
   (OFFICE-ID = 'HARTFORD' AND STATE = 'CT') OR ~
   (OFFICE-ID = BOSTON AND ZIP < 02299)

   The first group specifies the key OFFICE-ID; the second group specifies the keys OFFICE-ID and ZIP. The OFFICE-ID key appears in both groups, so a keyed read is possible.

Command Optimizations

A sequence of simple DISCOVER commands normally provides better overall system performance than one compound command, especially in a multiuser environment involving large PRISAM files. The command sequence "SELECT ALL FROM CUSTOMER", followed by a DISPLAY command, normally takes less elapsed time and makes better use of system resources than the command "SELECT AND DISPLAY ALL FROM CUSTOMER". This is especially apparent if there are several other users on the system competing with DISCOVER for system resources and the DISCOVER user takes time to peruse each screen of data before scrolling to the next screen.

DBMS SELECT Performance Issues

The purpose of this section is to describe the behavior of the DBMS Access Strategy Generator (ASG) in producing strategies for answering queries. It applies to Rev 20.0 DBMS and following revisions.

Because the ASG is a significant piece of software, this section contains a considerable amount of densely-packed information about its behavior. A good understanding of the behavior of the ASG can be achieved through careful study of this section. The audience for this section is assumed to be fairly familiar with the facilities provided by DISCOVER/DBMS, especially the SELECT command. In particular, readers are assumed to be familiar with its syntax and purpose, and with DBMS features used in schema and subschema design, including the concept of the virtual record.

Many terms, however, are used in this section and not defined elsewhere in DBMS or DISCOVER documentation. Please refer to the glossary at the end of this section for the definitions of unfamiliar terms.

All examples in this section utilize a single schema and subschema, contained at the end of this section.

The behavior of the ASG can be described by outlining the processing and stating the rules which the ASG follows in constructing an access strategy. A rule is composed of two parts: conditions and results. The general format of a rule is:

> IF a set of conditions about the query and/or the data base is true
>
> THEN <a set of results occurs>.

Conditions

The conditions in each ASG rule are composed of simple conditions about either the query (SELECT command) being processed, or about the data base (subschema or schema) being accessed. These simple conditions all

have one thing in common: each simple condition potentially affects the access strategy chosen by the ASG.

The set of conditions which forms the IF part of an ASG rule is made up of simple conditions about either the SELECT command or about the data base.

SELECT Command Conditions: These conditions fall into two categories: characteristics of target lists and characteristics of WHERE clause predicates.

Target List Characteristics: The characteristics of the target list which are important to the ASG refer to how many and which records of the virtual record structure are chosen to be part of the target list. Each subschema record which contains an item that is referenced in the target list must be retrieved by the access strategy when materializing the portion of the virtual record being selected.

In general, the fewer different subschema records referenced in the target list, and the closer these records are to the base record, the more efficient the access strategy and the performance of the query.

WHERE Clause Predicate Characteristics: The WHERE clause of a SELECT command specifies the restrictions placed on the virtual records to be retrieved by the command. Thus, the contents of the WHERE clause affects the access strategy chosen. Among the important factors are: whether the items referenced are access keys of the data base, what kind of keys they are, the logical operators connecting the WHERE clause predicates and their precedence, and the number of different subschema records containing the items referenced in the WHERE clause.

Data Base Conditions: These conditions relate to the structure of the schema and subschema which define the data base being accessed. Many factors are involved here. Among them are: the number of different record types which comprise the virtual record, the position of each record type in the subschema structure, the presence and number of access keys useful as entry points to the structure, (these access keys to entry point record types are called record keys), the locations of these keys within the subschema structure, and the presence, number, and types of sets relating record types within the subschema structure.

Virtual Record Characteristics: In a complex subschema, a given record type may be the base record of several virtual records, of increasing complexity as one includes record types related up through the connecting sets. However, the SELECT command may refer to only a few of these record types related to a given base record. The ASG attempts to deal with the simplest possible virtual record structure by eliminating from consideration those record types which do not participate in the query.

```
SELECT ORDER-RECORD ~
   FROM ITEM-RECORD ~
   WHERE ORDER-DATE = 830401
```

The base record in the above example could include each record type in the subschema, but it does not. The base record is always part of the virtual record, so it is not eliminated. In fact, the base record is crucial to this query since if an ORDER-RECORD occurrence satisfies the WHERE clause, it can still fail to satisfy the query if it owns no ITEM-RECORD occurrences. ORDER-RECORD is obviously part of the virtual record since it participates both in the target list and in the WHERE clause. However, neither CUSTOMER-RECORD nor PRODUCT-RECORD are part of the simplified virtual record since neither participates in any part of the query and neither one is a necessary part of the virtual record structure which connects ORDER-RECORD to ITEM-RECORD.

Results: The output of an execution of the ASG on a SELECT command is a generated access strategy. This strategy is then executed against the data base by the Access Strategy Interpreter (ASI). The result of an ASG rule specifies characteristics useful in generating an access strategy.

Types of Strategies

There are two basic types of access strategies.

Bottom-Up: The bottom-up access strategy scans each bucket of each area containing occurrences of the base record type of the virtual record. If a record occurrence satisfies the predicates specified on it in the WHERE clause, then all owners that are required to satisfy additional predicates or materialize target list items are found. These find-owner operations are done for all necessary record types, utilizing the sets and structures specified in the subschema.

Top-Down: The top-down access strategy utilizes the subschema structure and the WHERE clause predicate to limit the number of record occurrences accessed in each record type of the virtual record. Records accessed must include those containing items used in the WHERE clause predicates or in the target list.

In a top-down strategy, the data base structure is navigated from a record type at the top of the schema structure, called an entry point, downward via the appropriate sets, until the base record is reached, with record occurrences qualified or materialized along the way. Record types that are not along the access path from the entry point to the base record are qualified and materialized as needed by finding the

owner via sets which relate these records to records which are along the main access path.

The ASG frequently uses several entry points to retrieve the desired virtual records, utilizing different top-down substrategies.

Expected Access Strategy Performance: In general, top-down access strategies perform better than bottom-up access strategies. The main component of this performance is the actual number of disk accesses which are made in reading area buckets, sets, and calc files. For the bottom-up strategies, reading every bucket of an area will take a fairly long time, especially with large areas.

However, areas that are small may make the bottom-up strategy more efficient in some cases. Top-down strategies have a much better chance of performing better since efforts are made to restrict the number of records actually accessed by using sets and calc files.

Since bottom-up strategies are likely to take a long time to execute, the ASG gives the user the option of whether to proceed with a query if a bottom-up strategy is chosen. If the VERIFY feature in DISCOVER is ENABLED, then the ASG prompts as follows:

> "DISCOVER/DBMS is unable to optimize the submitted query for efficient access. Execution of this query may be lengthy. Do you wish this query to proceed?"

If the user answers YES, then the query will be executed normally. A NO answer will cause the query processing to terminate and the following message to be displayed to the user:

> "Query terminated at user's request."

followed by the DISCOVER prompt indicating that DISCOVER is ready for the next command. YES is the default response.

Statistics Used by the ASG

If the ASG determines that a top-down strategy is possible, it often must choose among several different methods of executing the strategy. In these cases, the ASG attempts to estimate the cost of executing all potential top-down access strategies. The ASG utilizes three kinds of statistics in determining these cost estimates.

Cardinality Statistics: The first type, cardinality statistics, provides counts of objects in the schema, such as the number of buckets in an area, the number of record occurrences in an area, and the number of members per owner a set contains. These are actual statistics, gathered and stored by the EXAMINE AREA and EXAMINE SET commands of DBACP, and can provide accurate information to the ASG in making its

cost estimates. When these statistics are not present, hypothetical values for these statistics are used in access strategy generation.

Selectivity Statistics: The second kind, selectivities, are probabilities that a record will satisfy a record predicate in a given query. Actual selectivities are unknown to the ASG, except for the case where the record predicate specifies equalities on all items of a key which allows no duplicates. In all other cases, hypothetical values for these selectivities are used in access strategy generation.

Operation Cost Statistics: The third kind, operation costs, measures the expected cost of executing an operation, such as find next of set or find owner, against a schema. These costs are likewise unknown to the ASG, which assumes that all operations are of equal cost.

Note

Since actual values for some of these statistics are not known to the ASG, some access strategies which are deemed most efficient by the ASG may in fact not be the most efficient available.

ASG Behavior

The purpose of the ASG is to determine and generate the most efficient access strategy available to satisfy the SELECT command. The inputs to the ASG are: descriptions of the virtual record, the target list, and the WHERE clause, all of which are specified or implied in the SELECT command. The output of the ASG is the set of instructions used by DBMS to execute the access strategy and produce individual virtual records which match the WHERE clause predicate. These instructions are interpreted and executed by a portion of DBMS called the Access Strategy Interpreter (ASI).

The processing that the ASG performs in generating a strategy is organized into two major sections: the first section analyzes the input provided and the structure of the virtual record and data base to determine the best access strategy available; the second section generates the instructions which implement the access strategy selected.

As stated above, conditions and results are combined to form the rules that the ASG follows in producing access strategies. The actual rules used by the ASG are contained in the following subsections.

Determining the Access Strategy

Determining the access strategy to be executed to satisfy a query is the heart of the ASG - where the decisions are made regarding the most efficient strategy to be chosen. This processing is decomposed into several steps, which follow.

Gather Schema Information: Scan the data base schema to gather relevant information about the items specified in the WHERE clause and target list items.

Rule

IF no WHERE clause is specified on the SELECT command, THEN a bottom-up strategy is generated.

Rule

IF no simple predicate in the WHERE clause on the SELECT command references the initial item of any key, THEN a bottom-up strategy is generated.

Normalize the WHERE Clause Predicate: The WHERE clause of a SELECT command can specify the same predicate in many different ways. In order to determine whether a SELECT command can be processed using a top-down strategy, the ASG converts the WHERE clause into a standard or normal form. The smallest component of a normalized predicate is known as a simple predicate. It has the form "operand operator operand" (i.e., CUSTOMER-ID = 3968). The next level of aggregation within a normalized predicate is called the record predicate. Simple predicates referring to items within the same record type can be collected together into a unit, the record predicate. For example:

```
CUSTOMER-ID = 3968 OR CUSTOMER-NAME = 'SMITH'
```

Record predicates are next grouped together with the AND operator. These collections of ANDed record predicates are called conjunctive groups (CGs). For example:

```
(CUSTOMER-ID = 3968 OR ~
    CUSTOMER-NAME = 'SMITH') AND ~
ORDER-DATE > 830401 AND PRODUCT-ONHAND > 100
```

Finally, the entire WHERE clause is organized into disjunctive normal form (DNF). DNF has the general form:

```
(record-pred AND..AND record-pred) OR..OR
(record-pred AND..AND record-pred).
```

in which all CGs are ORed together to form the entire WHERE clause predicate. For example:

```
((CUSTOMER-ID = 3968 OR ~
       CUSTOMER-NAME = 'SMITH') AND ~
    ORDER-DATE > 830401 AND ~
    PRODUCT-ONHAND > 100) OR ~
 (ITEM-QUANTITY > 1000 AND ~
    ORDER-SALESPERSON = 'SCHMIDT')
```

If utilizing a top-down strategy is possible, each conjunctive group in the normalized WHERE clause predicate generates its own top-down substrategy.

1) The first step of normalization is to use DeMorgan's Laws to distribute all negation operators inward to the simple predicates. For example, if ^ (CUSTOMER-ID < 3968) is specified in the WHERE clause, the normalized version is CUSTOMER-ID >= 3968.

2) The second step in the normalization process is to convert the whole WHERE clause predicate to DNF. Note that collections of simple predicates that form record predicates are not split up during the conversion to DNF. For example, the normal form of:

```
CUSTOMER-ID = 3968 AND
(ORDER-DATE > 830401 OR ITEM-QUANTITY > 1000)
```

is

```
(CUSTOMER-ID = 3968 AND ORDER-DATE > 830401) OR
(CUSTOMER-ID = 3968 AND ITEM-QUANTITY > 1000)
```

whereas

```
(CUSTOMER-ID = 3968 OR CUSTOMER-NAME = 'SMITH') AND ORDER-DATE
> 830401
```

is already in normal form.

3) The third step is to convert each record predicate to DNF as well.

Examine the Normalized Record Predicates: Each normalized record predicate must exhibit certain characteristics for a top-down access strategy to be considered. The rules in this section describe those characteristics and apply to each record predicate.

Rule

IF at least one simple predicate in a conjunctive group of a record predicate specifies an access key,

AND EITHER the access key is a full or partial sort/search key,

OR the access key is a full calc key and the predicate operator is equality,

THEN that conjunctive group is called amenable.

For example:

```
SELECT ORDER-RECORD ~
   FROM ORDER-RECORD ~
   WHERE ORDER-DATE = 830401 AND ~
         ORDER-SHIP-BY > 830908 OR ~
         ORDER-NUMBER > 12345
```

In this example, the entire WHERE clause is a single record predicate which contains two conjunctive groups. In the first group, both items of ORDER-RECORD participate in a search list of the SYSTEM-SRTD-ORDER-2 set. They thus constitute an access key which is a full search key. In the second group, ORDER-NUMBER is the sort key of the SYSTEM-SRTD-ORDER-1 set. Both conjunctive groups are therefore amenable.

Rule

IF more than one simple predicate in a conjunctive group of a record predicate qualifies that conjunctive group as amenable,

THEN only one record key is chosen to make that conjunctive group amenable. The record key selected is the one predicted to be the most selective over all the record occurrences of that record type. The following guidelines apply in choosing the most selective record key:

- Predicates containing the equality operator are more selective than predicates containing inequality operators, such as GE or NE.
- Two inequality predicates on the same item specifying boundaries on the desired values are more selective than just one.

- Equality predicates containing fully specified keys are more selective than those containing partially specified keys.
- For equality predicates which fully specify keys, keys with no duplicates are more selective than keys which allow duplicates.
- For choices between equality predicates which fully specify keys, predicates on calc keys are chosen over predicates on system-owned sets.

For example:

```
SELECT ORDER-RECORD ~
   FROM ORDER-RECORD ~
   WHERE ORDER-DATE = 830401 AND ~
         ORDER-NUMBER > 12345 AND ~
         ORDER-SHIP-BY = 830908
```

This example contains just one record predicate with one conjunctive group. There are three possible record keys which make this conjunctive group amenable: through the ascending key of SYSTEM-SRTD-ORDER-1, through the ascending key of SYSTEM-SRTD-ORDER-2, and through the second search key of SYSTEM-SRTD-ORDER-2. In the first two cases, the items involve inequality operators. In the third case, the two items specify equality operators. Thus, the second search key is chosen as the most selective record key to make this conjunctive group amenable.

For Example:

```
SELECT ORDER-RECORD ~
  FROM ORDER-RECORD ~
  WHERE ORDER-NUMBER <= 10000 AND ~
     ORDER-NUMBER > 12345 AND ~
     ORDER-SALESPERSON >= 'S'
```

This example again contains just one record predicate with one conjunctive group. There are three possible record keys which make this conjunctive group amenable: through the ascending key of SYSTEM-SRTD-ORDER-1, through the ascending key of SYSTEM-SRTD-ORDER-2, and through the first search key of SYSTEM-SRTD-ORDER-2.

In the first two cases the items involve two inequality predicates, forming both a lower and an upper bound on ORDER-NUMBER; in the third case the inequality predicate specifies only one bound on ORDER-SALESPERSON. Since only the first of the two items of the ascending key of SYSTEM-SRTD-ORDER-2 is specified, the fully specified SYSTEM-SRTD-ORDER-1 sort key is chosen as the most selective record key to make this conjunctive group amenable.

```
SELECT ORDER-RECORD ~
  FROM ORDER-RECORD ~
  WHERE ORDER-NUMBER = 10000 AND ~
     ORDER-SALESPERSON = ~
      'STEIGER'
```

This example again contains just one record predicate with one conjunctive group. There are four possible record keys which make this conjunctive group amenable: through the ORDER-RECORD calc key, through the ascending key of SYSTEM-SRTD-ORDER-1, through the ascending key of SYSTEM-SRTD-ORDER-2, and through the first search key of SYSTEM-SRTD-ORDER-2.

Since only the first of the two items of the ascending key of SYSTEM-SRTD-ORDER-2 is specified, it is not chosen. All the others are fully specified. The first search key of SYSTEM-SRTD-ORDER-2 is not chosen since it allows duplicates and the other two do not. Finally, the ORDER-RECORD calc key is chosen over the SYSTEM-SRTD-ORDER-1 sort key as the most selective record key to make this conjunctive group amenable.

Rule

IF all conjunctive groups of a normalized record predicate are amenable,

THEN that normalized record predicate is also amenable.

Examine the Normalized WHERE Clause Predicate: The normalized WHERE clause predicate must also exhibit certain characteristics for a top-down access strategy to be considered. The rules in this section describe those characteristics.

Rule

IF any conjunctive group contains no amenable record predicates,

THEN a bottom-up access strategy will be generated for this SELECT command.

For Example:

```
SELECT ORDER-RECORD ~
   FROM ORDER-RECORD ~
   WHERE CUSTOMER-NAME = ~
            'DVORAK VIOLA CO.' AND ~
         ORDER-SHIP-BY > 830908 OR ~
         ORDER-NUMBER > 12345
```

In this example, there are two conjunctive groups. In the second one, after the OR, ORDER-NUMBER is the record key through set SYSTEM-SRTD-ORDER-1. In the first conjunctive group, however, neither item is an access key or a calc key, and therefore this conjunctive group contains no amenable record predicates. A bottom-up access strategy is generated.

Rule

IF a record predicate of a conjunctive group is amenable,

AND the access key that made this conjunctive group amenable is a record key,

THEN that record type is a candidate entry point for a top-down substrategy to be generated,

AND that conjunctive group becomes a candidate for a top-down substrategy. For example:

```
SELECT ORDER-RECORD ~
   FROM ORDER-RECORD ~
   WHERE CUSTOMER-NAME =
            'DVORAK VIOLA CO.' AND ~
         ORDER-NUMBER > 12345
```

The WHERE clause in this example contains two record predicates, only one of which is amenable, the second. Since ORDER-NUMBER is the record key through set SYSTEM-SRTD-ORDER-1, ORDER-RECORD is a candidate entry point record for a top-down access strategy. Therefore, this conjunctive group (the only one in the query) is a candidate for a top-down substrategy.

Rule

IF a record predicate of a conjunctive group is not amenable,

AND the record type is a MANDATORY AUTOMATIC member of a system owned set,

THEN that record type is a candidate entry point for a top-down substrategy to be generated,

AND that conjunctive group becomes a candidate for a top-down substrategy.

For example:

```
SELECT ORDER-RECORD ~
   FROM ITEM-RECORD ~
   WHERE ITEM-BACKORDERED > 100 ~
     AND ORDER-SHIP-BY > '010184'
```

The WHERE clause in this example contains two record predicates, only one of which is amenable, the first. Since ORDER-RECORD is a MANDATORY AUTOMATIC member of set SYSTEM-SRTD- ORDER-1, even though the record predicate ORDER-SHIP-BY is not amenable, ORDER-RECORD is a candidate entry point record for a top-down access strategy. Therefore, this conjunctive group (the only one in the query) is a candidate for a top-down substrategy.

Rule

IF a record type in the virtual record structure has no record predicate in the conjunctive group,

AND the record type is a MANDATORY AUTOMATIC member of a system owned set,

AND a direct or indirect owner record of this record type is qualified by a record predicate of this conjunctive group,

THEN that record type is a candidate entry point for a top-down substrategy to be generated,

AND that conjunctive group becomes a candidate for a top-down substrategy.

For example:

```
SELECT ORDER-RECORD ~
   FROM ITEM-RECORD ~
   WHERE ITEM-BACKORDERED > 100 ~
     AND CUSTOMER-CITY = 'BOSTON'
```

The WHERE clause in this example contains two record predicates, only one of which is amenable, the first. CUSTOMER-CITY is not an amenable record predicate for CUSTOMER-RECORD, but since CUSTOMER-RECORD is a MANDATORY AUTOMATIC member of SYSTEM-SRTD-CUSTOMER-1, by the previous rule, it is a candidate entry point record.

In addition, since ORDER-RECORD is a MANDATORY AUTOMATIC member of set SYSTEM-SRTD-ORDER-1 (even though there is no record predicate on this record), by this rule, ORDER-RECORD is a candidate entry point record for a top-down access strategy because CUSTOMER-RECORD owns ORDER-RECORD and has a record predicate in the WHERE clause. Therefore, this conjunctive group (the only one in the query) is a candidate for a top-down substrategy.

Rule

IF a record type in the virtual record structure has no record predicate in the conjunctive group,

AND the record type is a MANDATORY AUTOMATIC member of a system owned set,

AND every member record along the access path from this record type to the first record type qualified by a record predicate has AUTOMATIC and MANDATORY membership in the relating set,

THEN that record type is a candidate entry point for a top-down substrategy to be generated,

AND that conjunctive group becomes a candidate for a top-down substrategy.

For example:

```
SELECT CUSTOMER-RECORD ~
   FROM ITEM-RECORD ~
   WHERE ITEM-BACKORDERED > 100
```

The WHERE clause in this example contains only one amenable record predicate. This prevents the query from being rejected outright and forced to use a bottom-up access strategy.

The virtual record ITEM-RECORD consists of the records CUSTOMER-RECORD, ORDER-RECORD, and ITEM-RECORD. ORDER-RECORD is a candidate entry point since it is a MANDATORY AUTOMATIC member of set SYSTEM-SRTD-ORDER-1, and since ITEM-RECORD is a MANDATORY AUTOMATIC member of set ORDER-ITEM. CUSTOMER-RECORD is a candidate entry point since it is a MANDATORY AUTOMATIC member of set SYSTEM-SRTD-CUSTOMER-1, since ORDER-RECORD is a MANDATORY AUTOMATIC member of set CUSTOMER-ORDER, and because ORDER-RECORD is a candidate entry point. Therefore, this conjunctive group (the only one in the query) is twice a candidate for a top-down substrategy.

Rule

IF all conjunctive groups of the normalized WHERE clause predicate are candidates for a top-down substrategy,

THEN a top-down access strategy will be generated for this SELECT command.

OTHERWISE a bottom-up access strategy will be generated.

For Example:

```
SELECT ORDER-RECORD ~
   FROM ORDER-RECORD ~
   WHERE ORDER-DATE > 830908 OR ~
         CUSTOMER-NAME = ~
            'DVORAK VIOLA CO.' AND ~
         ORDER-NUMBER > 12345
```

In this example, the second conjunctive group is already known to be a candidate for a top-down substrategy. The first conjunctive group is also since ORDER-DATE is a partial search key of the SYSTEM-SRTD-ORDER-2 set, thus making it amenable and a record key. Thus, two top-down access substrategies will be generated for this query. Select The Best Substrategies: Under the assumption that a top-down access strategy will be employed, the ASG must now choose the best one. One top-down access substrategy is necessary for each conjunctive group, and these substrategies are independent of each other, although they might select duplicate virtual record occurrences. In fact, the ASG may need to choose between several possible substrategies for each conjunctive group, considered separately.

Rule

IF a record predicate of a member record type specifies a (partial) sort/search key of a set which lies along the access path from the entry point record to the base record, (called a set key)

THEN that (partial) sort/search key is utilized to access occurrences of that member record type during a top-down access substrategy in order to reduce the number of the record occurrences accessed from the owner record type.

Rule

IF there is only one candidate entry point for a top-down substrategy,

THEN that entry point will be chosen as the entry point for that top-down substrategy.

For Example:

```
SELECT PRODUCT-ONHAND, ~
       ITEM-QUANTITY ~
   FROM ITEM-RECORD ~
   WHERE PRODUCT-CODE = 'AJ104580' ~
         AND ITEM-QUANTITY > 100
```

The WHERE clause contains only one conjunctive group since there are no OR operators. There are two record predicates since PRODUCT-CODE is in PRODUCT-RECORD and ITEM-QUANTITY is in ITEM-RECORD.

In fact, these are the only two record types utilized in this query since the target list items are contained in these two records also. The PRODUCT-RECORD type utilizes PRODUCT-CODE both as a calc key and as an ascending key through the SYSTEM-SRTD-PRODUCT-1 set. Since equality is the predicate operator in the WHERE clause, the CALC key will be used to access this entry record type. The ITEM-RECORD type has no key items which can be used to access each ITEM-RECORD record occurrence.

Thus, PRODUCT-RECORD is the only candidate entry point for a top-down substrategy. Note that since there is only one conjunctive group, there is only one top-down substrategy to be generated. ITEM-QUANTITY will be used to restrict access to record occurrences in ITEM-RECORD that are owned by the occurrences of PRODUCT-RECORD which satisfy the record predicate. The PRODUCT-ITEM set is used for this purpose.

Rule

IF there is more than one candidate entry point for a top-down substrategy, over a given conjunctive group,

THEN only one entry point is chosen to begin the substrategy for this conjunctive group.

The entry point selected is the one predicted to have the least execution cost of all entry point candidates. The cost estimations for each entry point are done by complex calculations, based on the actual cardinality statistics, if available, and on the remaining hypothetical statistics relating to the data base and the normalized WHERE clause predicate.

In general, the choice exhibits these characteristics: the entry point with the most selective record predicate is chosen, the entry point with the most selective predicates on the sets encountered along the access path to the base record is chosen, the entry point containing the least record occurrences, combined with the smallest number of record occurrences for all the records along the path to the base record, is chosen. There is no obviously discernible priority given to these guidelines. For example:

```
SELECT PRODUCT-DESCRIPTION ~
   FROM ITEM-RECORD ~
   WHERE ORDER-NUMBER > 10000 AND ~
         PRODUCT-CODE = 'AJ104523'
```

There are three record types involved in this query: ORDER-RECORD, ITEM-RECORD, and PRODUCT-RECORD. ORDER-NUMBER is the sort list for the

set SYSTEM-SRTD-ORDER-1, thereby making the ORDER-RECORD a candidate entry point. PRODUCT-CODE is both the calc key for PRODUCT-RECORD and the sort list for the SYSTEM-SRTD-PRODUCT-1 set, thus making PRODUCT-RECORD also a candidate entry point. The CALC key is used since equality is the predicate operator. The ASG chooses the PRODUCT-RECORD as the entry point for this query since the equality operator on a fully specified key without duplicates is more selective than the greater-than operator. For example:

```
SELECT PRODUCT-DESCRIPTION, ~
       PRODUCT-ONHAND ~
   FROM ITEM-RECORD ~
   WHERE ORDER-NUMBER = 10000 AND ~
         PRODUCT-CODE = 'AJ104523' ~
         AND ITEM-QUANTITY > 100
```

Again there are three record types involved in this query: ORDER-RECORD, ITEM-RECORD, and PRODUCT-RECORD. ORDER-NUMBER is the sort list for the set SYSTEM-SRTD-ORDER-1 and the CALC key for ORDER-RECORD, thereby making the ORDER-RECORD a candidate entry point using the CALC key. PRODUCT-CODE is both the CALC key for PRODUCT-RECORD and the sort list for the SYSTEM-SRTD-PRODUCT-1 set, thus making PRODUCT-RECORD also a candidate entry point using the CALC key.

In addition, ITEM-QUANTITY is the sort key of the PRODUCT-ITEM set. Therefore, since the predicates on the two entry record types are equally selective, the ASG usually chooses the PRODUCT-RECORD as the entry point for this query since the ITEM-QUANTITY predicate is probably a more selective access path than using the ORDER-ITEM set, whose sort key is not used in this query. Actual cardinality statistics may show that the other strategy is more efficient. For example:

```
SELECT CUSTOMER-NAME ~
   FROM ITEM-RECORD ~
   WHERE ORDER-NUMBER > 10000 AND ~
         PRODUCT-CODE > 'AJ104523' ~
         AND CUSTOMER-ID > 1000
```

Again there are three record types involved in this query: CUSTOMER-RECORD, ORDER-RECORD, and PRODUCT-RECORD. ITEM-RECORD is also involved since it is the base record for the virtual record. CUSTOMER-ID is the sort list for the set SYSTEM-SRTD-CUSTOMER-1, thereby making CUSTOMER-RECORD a candidate entry point using that set. ORDER-NUMBER is the sort list for the set SYSTEM-SRTD-ORDER-1, thereby making ORDER-RECORD a candidate entry point using that set.

PRODUCT-CODE is the sort list for the SYSTEM-SRTD-PRODUCT-1 set, thus making PRODUCT-RECORD also a candidate entry point using that set.

Depending on the actual cardinality statistics, the ASG could choose any of these record types as the entry point for this query since each record predicate is equally selective, and no additional selectivity is present along any of the connecting sets.

Generating the Access Strategy

The navigation that the two basic access strategies utilize has been described above. The bottom-up strategy makes one pass over the data base, qualifying and materializing virtual records. The top-down strategy is actually several serial strategies, each with its own pass over the data base. Duplicate qualified virtual record occurrences are prevented.

Glossary

The reader is assumed to be familiar with the concepts and terminology of CODASYL databases, virtual record structures, and SELECT commands. This section defines some additional terms used in this section.

Access Key: a record key or a set key.

Access Path: a portion of a schema structure (sets, records, and calc files) which provides the means to navigate from an entry point to the base record of a virtual record and to all record types required for materialization or qualification.

Access Strategy: a dynamically constructed plan or program to navigate a DBMS schema in order to efficiently access the record occurrences which form virtual records satisfying a DISCOVER query (SELECT command).

Amenable: a (portion of a) predicate in a SELECT command is amenable if it specifies access keys which permit a top-down strategy to utilize those keys to limit the number of record occurrences accessed for qualification.

ASG: Access Strategy Generator: the portion of DBMS intended for DISCOVER/DBMS support which determines the most effective means of navigating the subschema in order to satisfy a query against a virtual record.

ASI: Access Strategy Interpreter: the portion of DBMS intended for DISCOVER/DBMS support which executes the access strategy generated by the the ASG.

Cardinality Statistics: statistics that provide counts of objects in a schema, such as the number of buckets in an area, the number of record occurrences in an area, and the number of members per owner a set contains.

Conjunction: a predicate consisting of the logical AND of one or more sub-predicates, e.g. A AND B AND C.

Conjunctive Group: a conjunction of record predicates forming the main components of the disjunctive normal form of a predicate.

Disjunction: a predicate consisting of the logical OR of one or more sub-predicates, e.g. A OR B OR C.

Disjunctive Normal Form (DNF): a canonical form for predicates. A predicate is in DNF if it is the disjunction of one or more conjunctions. Every predicate consisting of AND's, OR's, and NOT's of simple predicates has an equivalent DNF predicate.

Entry Point: Given a predicate against a virtual record structure, an entry point for that predicate is a record type in the virtual record structure that is used as the starting point for navigation of the structure to identify those virtual records satisfying the predicate.

Materialization: the goal of an access strategy that retrieves a set of record occurrences forming a virtual record in order to satisfy the target list of a SELECT command.

Operation Costs: statistics that measure the expected cost of executing an operation (such as "find owner") against a schema.

Partial Key (Value): for access keys which are composed of several data items in a record, a partial key is one where not every data item is specified in the query. At least the first item is specified, and possibly others in increasing sequence, as long as there is at least one data item which is not specified.

Predicate: a statement of conditions which, in a SELECT command, restricts the virtual records selected for retrieval. The statement is expressed using logical operators.

Qualification: the goal of an access strategy which retrieves record occurrences and checks simple predicate conditions on items contained in those record occurrences for the purpose of deciding whether a candidate virtual record satisfies the WHERE clause conditions of a SELECT command.

Qualified Record Type: a record type against which there is a record predicate.

Record Key: a sort or search key of a system-owned set whose related record type has AUTOMATIC and MANDATORY membership in that system-owned set, or a calc key of a record.

Record Predicate: a predicate referencing a single record type.

Rule: in this section, a precise description of some aspect of the behavior of the ASG.

Selectivities: statistics that measure the probability that a record will satisfy a record predicate in a given query.

Set Key: a sort or search key of a set which relates two record types that lie along the access path from the entry point record and the base record.

Simple Condition: a statement in this section about characteristics of the SELECT command or the data base which contains neither conjunctions nor disjunctions of other conditions.

Simple Predicate: a predicate that does not involve the logical operators AND, OR, and NOT, e.g. LASTNAME='SMITH'.

Strategy: see Access Strategy

1 Example Schema

```
SCHEMA NAME IS AW2-SCHEMA.
AREA NAME IS ORDER-AREA.
RECORD NAME IS CONTROL-RECORD;
    LOCATION MODE IS CALC USING CONTROL-KEY
    DUPLICATES ARE NOT ALLOWED.
/*      01  CONTROL-RECORD.*/
        01  CONTROL-KEY; INTEGER*2.
        01  CONTROL-LAST-CUSTOMER-ID; PICTURE '9(4)'.
        01  CONTROL-LAST-ORDER-NUMBER; PICTURE '9(6)'.
RECORD NAME IS CUSTOMER-RECORD;
    LOCATION MODE IS CALC USING CUSTOMER-ID
    DUPLICATES ARE NOT ALLOWED.
/*      01  CUSTOMER-RECORD.*/
        01  CUSTOMER-ID; INTEGER*2.
        01  CUSTOMER-NAME; PICTURE 'X(20)'.
        01  CUSTOMER-ADDRESS; PICTURE 'X(20)'.
        01  CUSTOMER-CITY; PICTURE 'X(12)'.
        01  CUSTOMER-STATE-COUNTY; PICTURE 'X(7)'.
        01  CUSTOMER-ZIP-POST-CODE; PICTURE 'X(7)'.
        01  CUSTOMER-COUNTRY; PICTURE 'X(12)'.
RECORD NAME IS ORDER-RECORD;
    LOCATION MODE IS CALC USING ORDER-NUMBER
    DUPLICATES ARE NOT ALLOWED.
/*      01  ORDER-RECORD.*/
        01  ORDER-NUMBER; INTEGER*4.
        01  ORDER-DATE; PICTURE '9(6)'.
        01  ORDER-SALESPERSON; PICTURE 'X(20)'.
        01  ORDER-SHIP-BY; PICTURE 'X(20)'.
RECORD NAME IS ITEM-RECORD;
    LOCATION MODE IS VIA ORDER-ITEM SET.
/*      01  ITEM-RECORD.*/
        01  ITEM-QUANTITY; PICTURE '9(3)'.
        01  ITEM-BACKORDERED; PICTURE '9(3)'.
RECORD NAME IS PRODUCT-RECORD;
    LOCATION MODE IS CALC USING PRODUCT-CODE
    DUPLICATES ARE NOT ALLOWED.
/*      01  PRODUCT-RECORD.*/
        01  PRODUCT-CODE; PICTURE 'X(8)'.
        01  PRODUCT-DESCRIPTION; PICTURE 'X(20)'.
        01  PRODUCT-PRICE; PICTURE '9(2)V99'.
        01  PRODUCT-ONHAND; PICTURE '9(4)'.
        01  PRODUCT-LOCATION.
            02  PRODUCT-ROW; PICTURE 'X(2)'.
            02  PRODUCT-BIN; PICTURE '9(2)'.
```

```
SET NAME IS CUSTOMER-ORDER;
    ORDER IS SORTED;
    OWNER IS CUSTOMER-RECORD.
    MEMBER IS ORDER-RECORD MANDATORY AUTOMATIC;
    ASCENDING KEY IS ORDER-DATE DUPLICATES ALLOWED.
    SET OCCURRENCE SELECTION IS THRU CURRENT OF SET.
SET NAME IS ORDER-ITEM;
    ORDER IS SORTED;
    OWNER IS ORDER-RECORD.
    MEMBER IS ITEM-RECORD MANDATORY AUTOMATIC;
    ASCENDING KEY IS ITEM-BACKORDERED DUPLICATES ALLOWED;
    SET OCCURRENCE SELECTION IS THRU CURRENT OF SET.
SET NAME IS PRODUCT-ITEM;
    ORDER IS SORTED;
    OWNER IS PRODUCT-RECORD.
    MEMBER IS ITEM-RECORD OPTIONAL AUTOMATIC;
    DESCENDING KEY IS ITEM-QUANTITY DUPLICATES ALLOWED;
    SET OCCURRENCE SELECTION IS THRU CURRENT OF SET.
SET NAME IS SYSTEM-SRTD-CUSTOMER-1;
    ORDER IS SORTED;
    OWNER IS SYSTEM.
    MEMBER IS CUSTOMER-RECORD MANDATORY AUTOMATIC;
    ASCENDING KEY IS CUSTOMER-ID DUPLICATES NOT ALLOWED.
SET NAME IS SYSTEM-SRTD-ORDER-1;
    ORDER IS SORTED;
    OWNER IS SYSTEM.
    MEMBER IS ORDER-RECORD MANDATORY AUTOMATIC;
    ASCENDING KEY IS ORDER-NUMBER DUPLICATES NOT ALLOWED.
SET NAME IS SYSTEM-SRTD-ORDER-2;
    ORDER IS SORTED;
    OWNER IS SYSTEM.
    MEMBER IS ORDER-RECORD MANDATORY AUTOMATIC;
    ASCENDING KEY IS ORDER-NUMBER,
        ORDER-DATE DUPLICATES NOT ALLOWED;
    SEARCH KEY IS ORDER-SALESPERSON DUPLICATES ALLOWED;
    SEARCH KEY IS ORDER-DATE,
        ORDER-SHIP-BY DUPLICATES ALLOWED.
SET NAME IS SYSTEM-ORDER-LAST;
    ORDER IS LAST;
    OWNER IS SYSTEM.
    MEMBER IS ORDER-RECORD MANDATORY AUTOMATIC.
SET NAME IS SYSTEM-SRTD-PRODUCT-1;
    ORDER IS SORTED;
    OWNER IS SYSTEM.
    MEMBER IS PRODUCT-RECORD MANDATORY AUTOMATIC;
    ASCENDING KEY IS PRODUCT-CODE DUPLICATES NOT ALLOWED;
    SEARCH KEY IS PRODUCT-ROW,
        PRODUCT-BIN DUPLICATES ALLOWED.
END SCHEMA.
```

Example COBOL Subschema

```
SUBSCHEMA NAME IS SUB-1 OF SCHEMA AW2-SCHEMA.
AREA SECTION
COPY ALL AREAS.
RECORD SECTION
01  CUSTOMER-RECORD.
    02  CUSTOMER-ID; PICTURE 9(4).
    02  CUSTOMER-NAME; PICTURE X(20).
    02  CUSTOMER-ADDRESS; PICTURE X(20).
    02  CUSTOMER-CITY; PICTURE X(12).
    02  CUSTOMER-STATE-COUNTY; PICTURE X(7).
    02  CUSTOMER-ZIP-POST-CODE; PICTURE X(7).
    02  CUSTOMER-COUNTRY; PICTURE X(12).
01  ORDER-RECORD.
    02  ORDER-NUMBER; PICTURE 9(6).
    02  ORDER-DATE; PICTURE 9(6).
    02  ORDER-SALESPERSON; PICTURE X(20).
    02  ORDER-SHIP-BY; PICTURE X(20).
01  ITEM-RECORD.
    02  ITEM-QUANTITY; PICTURE 9(3).
    02  ITEM-BACKORDERED; PICTURE 9(3).
01  PRODUCT-RECORD.
    02  PRODUCT-CODE; PICTURE X(8).
    02  PRODUCT-DESCRIPTION; PICTURE X(20).
    02  PRODUCT-PRICE; PICTURE 9(2)V99.
    02  PRODUCT-ONHAND; PICTURE 9(4).
    02  PRODUCT-LOCATION.
        03  PRODUCT-ROW; PICTURE X(2).
        03  PRODUCT-BIN; PICTURE 9(2).
SET SECTION
    COPY ALL SETS.
```

PRISAM SELECT Performance Issues

This section describes the behavior of the SELECT command when used on PRISAM files, and what steps can be taken to formulate more efficient SELECT statements.

Questions concerning the performance of DISCOVER generally take the form: "How should a given query be set up for best performance?". The performance of a query depends on a number of factors. The best way to address this question is with some general guidelines, outlined in the sections following. Note that some of the guidelines are release dependent and specifics of each guideline may change as development on DISCOVER continues; however, the overall intent of each guideline will not change.

Relation Selection Optimizations: The FROM clause in the SELECT, MODIFY, and ERASE commands specifies the relation which will be

processed. Choosing an appropriate relation for a query is important for best performance.

Single-record relations are established by the USE command. The command "USE FILE CUSTOMER", assuming that file CUSTOMER is a PRISAM file with a single record of the same name, would establish one single-record relation named CUSTOMER. The commands "USE FILE PRODUCT" and "USE FILE ORDER", given similar assumptions, would establish two additional single-record relations, PRODUCT and ORDER. Note that single-record relations are usually referred to as records in most of the DISCOVER documentation.

Multi-record relations are built by executing one or more JOIN commands. The first JOIN command that is executed on a given relation establishes the base record for the relation, names the relation, and sets up a default processing sequence for the records in the relation. Consider the command:

```
JOIN RECORD CUSTOMER TO RECORD ORDER ~
     ON CUSTOMER-ID = ORDER-CUSTOMER-ID ~
     GIVING RELATION CUSTOMER-ORDER
```

The record named in the "TO RECORD" clause tells DISCOVER that ORDER is the base record for this relation. The relation is named by the "GIVING RELATION" clause. The default processing sequence is base record ORDER followed by first joined record CUSTOMER. All other records that are part of this relation will be joined directly to the base record or indirectly to the base record through one or more intermediate records.

JOIN commands which use a previously named relation will add records to the named relation. Consider the command:

```
JOIN RECORD PRODUCT TO RELATION CUSTOMER-ORDER ~
     ON PRODUCT-CODE = ORDER-PRODUCT-CODE
```

This command will add another record PRODUCT to an existing relation CUSTOMER-ORDER. The PRODUCT record will be appended to the end of the default processing sequence.

The two JOIN commands above will establish a relation named CUSTOMER-ORDER with a base record of ORDER and a default process sequence of ORDER record, CUSTOMER record, PRODUCT record.

Now consider the commands:

```
JOIN RECORD PRODUCT TO RECORD ORDER ~
     ON PRODUCT-CODE = ORDER-PRODUCT-CODE ~
     GIVING RELATION PRODUCT-ORDER
```

and

```
JOIN RECORD CUSTOMER TO RELATION PRODUCT-ORDER ~
     ON CUSTOMER-ID = ORDER-CUSTOMER-ID
```

These two JOIN commands will establish a relation named PRODUCT-ORDER which is similar to the relation CUSTOMER-ORDER but it will have a default process sequence of ORDER record, PRODUCT record, CUSTOMER record.

If a user followed the command sequence above, there would be three single-record relations (CUSTOMER, ORDER, and PRODUCT) and two multi-record relations (CUSTOMER-ORDER and PRODUCT-ORDER) defined to DISCOVER which could be used in the FROM clause. The question is "which is the most appropriate for a given query?" The following rules should be applied to answer this question.

1. Chose the relation that has the smallest number of records and the smallest PRISAM record descriptions that contain all the data the user is looking for and all of the qualifying conditions that the user is placing on the data.

2. If more than one relation satisfies rule 1, look at the default processing sequence of the relations that satisfy rule 1 and choose the relation that will reject occurrences of the relation as early as possible along the default processing path if there is a WHERE clause. If there is no WHERE clause, choose the relation that will accept occurrences of the relation as early as possible along the default processing path.

3. Always be aware of implied conditions when using a multi-record relation. "SELECT CUSTOMER FROM CUSTOMER" tells DISCOVER to find each record occurrence in CUSTOMER and copy it into the current table; "SELECT CUSTOMER FROM CUSTOMER-ORDER" tells DISCOVER to find each record occurrence in ORDER, then find the CUSTOMER occurrence that is joined to that ORDER occurrence, and copy that occurrence of CUSTOMER into the current table.

   The first SELECT will copy one occurrence of each CUSTOMER into the current table. No CUSTOMER occurrence will be skipped and none will be duplicated, assuming none are duplicated in the CUSTOMER file. The second SELECT will copy one occurrence of the joined CUSTOMER into the current table for each occurrence of ORDER in the ORDER file. CUSTOMERS will be skipped if there is no joined ORDER and CUSTOMERS that have more than one joined ORDER will be duplicated.

EXAMPLES: Given the relations defined above.

1. Find all of the data about all of the CUSTOMERS.

```
SELECT ALL FROM CUSTOMER
```

   The multi-record relations CUSTOMER-ORDER or PRODUCT-ORDER could have also been used, but they do not satisfy rule 1.

2. Find all of the data about PRODUCTS which are out of stock.

```
SELECT ALL FROM PRODUCT WHERE PRODUCT-ONHAND = 0
```

The multi-record relations CUSTOMER-ORDER or PRODUCT-ORDER could have also been used but they do not satisfy rule 1.

3. Find the CUSTOMER-ID and PRODUCT-CODE for all ORDERS.

```
SELECT CUSTOMER-ID, PRODUCT-CODE FROM CUSTOMER-ORDER

   or

SELECT CUSTOMER-ID, PRODUCT-CODE FROM PRODUCT-ORDER
```

Both are equally appropriate because all three records must be processed to satisfy the query and there is no WHERE clause to aid in selecting the best relation.

4. Find the CUSTOMER-ID for all ORDERS.

```
SELECT CUSTOMER-ID FROM CUSTOMER-ORDER
```

CUSTOMER-ORDER is appropriate because of the implied condition. PRODUCT-ORDER could also be used but the PRODUCT would have to be processed when there is no need to do so (see Rule 2). Note that there will be duplicate CUSTOMER-IDs if an occurrence of CUSTOMER has more than one ORDER joined to it. The REMOVE DUPLICATES command can be used to eliminate any duplicates.

5. Find the CUSTOMER-ID for all CUSTOMERS who have ordered PRODUCT AB123.

```
SELECT CUSTOMER-ID FROM PRODUCT-ORDER ~
       WHERE PRODUCT-CODE = 'AB123'
```

PRODUCT-ORDER is appropriate because of the implied condition. DISCOVER can reject occurrences as soon as it has processed ORDER (second on the default process list for PRODUCT-ORDER) without processing the CUSTOMER (third on the list). CUSTOMER-ORDER could also be used but DISCOVER would always process the CUSTOMER (second on the default process list for CUSTOMER-ORDER) before it processed PRODUCT. CUSTOMER occurrences would be processed that had not ordered the product. The same duplicate condition may occur as described in the example above.

Data Ordering Optimizations: DISCOVER will take advantage of the processing sequence of data whenever it can. Consider the last example above where DISCOVER is asked to find the CUSTOMER-ID of all CUSTOMERS who have ordered PRODUCT AB123.

If ORDER is sorted in ORDER-CUSTOMER-ID then ORDER-PRODUCT-CODE sequence, DISCOVER will be able to take advantage of having already processed CUSTOMER and/or PRODUCT for previous occurrence of ORDER if they have the same ORDER-CUSTOMER-ID and/or ORDER-PRODUCT-CODE as the current occurrence of ORDER. Given the command "SELECT CUSTOMER-ID FROM PRODUCT-ORDER WHERE PRODUCT-CODE = 'AB123'", DISCOVER will process an ORDER occurrence, then the PRODUCT occurrence joined to the ORDER, and assuming that it is indeed product AB123, DISCOVER would process the CUSTOMER joined to ORDER.

Having completed processing of that occurrence of ORDER, DISCOVER will save the PRODUCT and CUSTOMER that it has just processed and proceed to the next occurrence of ORDER. If the next occurrence of ORDER has the same CUSTOMER and/or PRODUCT joined to it as the previous ORDER had, DISCOVER will perform a substantially shorter processing cycle on CUSTOMER and/or PRODUCT because of the saved data.

PRISAM key optimizations: When performing the SELECT, MODIFY, and ERASE commands, DISCOVER first finds an occurrence of the base record in the relation, and then finds the additional record occurrences which are joined (directly or indirectly) to that occurrence of the base record.

To find occurrences of the base record, DISCOVER can scan the base record sequentially, or perform keyed reads on the base record. The sequential read will always work, but can be quite inefficient when the total occurrences of the base record far outnumber the occurrences actually requested.

DISCOVER will attempt to use a keyed read on the base record if a WHERE clause has been specified. This will succeed only if at least one key applies to the entire WHERE clause, and the conditions on such a key are "useful" (for example, the conditions PRICE<10.00 OR PRICE>9.00 together cover all possible values. They are not useful for reducing the number of occurrences read). To determine whether a keyed read can be used:

1. Manipulate the WHERE clause so that it has the general form

   (condition AND condition AND ... AND condition) OR
   (condition AND ...) OR (condition ...) OR ...

   Call each ANDed term a "group".

2. For each group, list the names which are defined as keys in the base record and which do not use the ^= (NOT EQUAL) comparison operator.

3. Mark which names appear in all the lists from step 2. These are candidates for performing a keyed read. If there are no names which appear in all the lists from step 2, then a sequential read must be performed. Otherwise, a keyed read is possible (subject to the notion of "usefulness" mentioned above).

EXAMPLES: Assume we have a PRISAM file OFFICE which contains the keys OFFICE-ID and ZIP, and a non-key data item STATE.

1. SELECT ALL FROM OFFICE WHERE OFFICE-ID = 'MILLINOCKET'

   The single group specifies the key OFFICE-ID. This query is a candidate for a keyed read.

2. SELECT ALL FROM OFFICE WHERE STATE = 'NM'

   No key is specified in this query; a sequential read will be used.

3. SELECT ALL FROM OFFICE WHERE STATE = 'MA' AND ZIP < 02100

   The single group specifies the key ZIP. This query is a candidate for a keyed read.

4. SELECT ALL FROM OFFICE WHERE ~
   OFFICE-ID = 'SOUTHWEST' OR ~
   OFFICE-ID = 'HARTFORD'

   Both groups specify the key OFFICE-ID; a keyed read is possible.

5. SELECT ALL FROM OFFICE WHERE ~
   OFFICE-ID = 'PITTSFIELD' OR ZIP > 69999

   The first group specifies the key OFFICE-ID, the second specifies the key ZIP. Neither key appears in both groups, so a sequential read must be used.

6. SELECT ALL FR OFFICE WHERE STATE = 'CT' OR ZIP <= 02299

   The second group specifies the key ZIP, but the first group does not specify any key. A sequential read must be used.

7. SELECT ALL FR OFFICE WHERE ~
   (OFFICE-ID = 'HARTFORD' AND STATE = 'CT') OR ~
   (OFFICE-ID = BOSTON AND ZIP < 02299)

   The first group specifies the key OFFICE-ID; the second group specifies the keys OFFICE-ID and ZIP. The OFFICE-ID key appears in both groups, so a keyed read is possible.

Command Optimizations: A sequence of simple DISCOVER commands normally provides better overall system performance than one compound command, especially in a multi-user environment involving large PRISAM files. The command sequence of "SELECT ALL FROM CUSTOMER" followed by a DISPLAY command normally takes less time and makes better use of system resources than the command "SELECT AND DISPLAY ALL FROM CUSTOMER". This will be especially apparent if there are several other users on the system competing with DISCOVER for system resources and the DISCOVER user takes time to peruse each screen of data before scrolling to the next screen.

MIDASPLUS ENHANCEMENTS

MIDASPLUS enhancements at Rev. 21.0 are discussed below.

R-mode Dependency Eliminated

MIDASPLUS has eliminated R-mode dependency in offline utilities and the runtime library.

ECS

MIDASPLUS now supports the Prime Extended Character Set (ECS).

New SYSCOM Files

At Rev. 21.0, MIDASPLUS extends the support of the C and Pascal languages by providing new SYSCOM insert files. These insert files allow programmers to refer to error codes and key values by mnemonic names rather than absolute values.

To use the new insert files, C programmers should include the following lines at the top of each MIDASPLUS calling module:

```
#include "syscom>parm.k.ins.cc"
#include "syscom>keys.ins.cc"
```

Similarly, Pascal programmers should include these lines at the top of each MIDASPLUS calling module:

```
%include 'syscom>parm.k.ins.pascal'
%include 'syscom>keys.ins.pascal'
```

MIDASPLUS Requirements

MIDASPLUS requires PRIMOS Rev. 21.0. MIDASPLUS reserves shared segments 2122, 2123, 2124, 2125, 2320, and 2321, and of per-user static storage segment 6006 addresses 40000 to 70000.

Note

A child process that is a MIDASPLUS application should not be forked from a MIDASPLUS application. The results of such an operation are unpredictable.

Installing MIDASPLUS

The MIDASPLUS installation procedures are standard using the three command files described below:

1) MIDASPLUS.INITINSTALL.COMI is a COMINPUT file that should be run the first time MIDASPLUS is installed at Rev 20.2 or higher. This file creates the MIDASPLUS* directory which is used for error logging and cominputs MIDASPLUS.INSTALL.COMI.

2) MIDASPLUS.INSTALL.COMI is a COMINPUT file that places all the MIDASPLUS files in the appropriate system directories. This file is similar to the standard Master Disk install files.

3) MIDASPLUS.SHARE.COMI is a COMINPUT file that properly shares and initializes MIDASPLUS on the system. This file is similar to the standard Master Disk share files. It is recommended that MIDASPLUS.SHARE.COMI be placed in the system startup file (either PRIMOS.COMI or C_PRMO) so that MIDASPLUS is shared and initialized at each system cold start.

MIDASPLUS includes all the online and offline callable routines, plus the MIDASPLUS offline utilities CREATK, KBUILD, KIDDEL, MDUMP, MPACK, MPLUSCLUP, and SPY. In addition, both an R-mode library (KIDALB) and a synonym library (VKDALB) are built for compatibility with previous MIDAS/MIDASPLUS releases.

If you desire to use any of the MIDASPLUS CONFIG directives (see below), you should create a file called MPLUS.CONFIG in SYSTEM containing the CONFIG directives, which are optional.

Password Directories

Beginning at Rev. 19.3, a change in the scheme for preventing concurrent file access by a MIDASPLUS utility user and a MIDASPLUS runtime library user requires that any MIDASPLUS file on which a utility will be used be in an ACL directory and that its parent directory be an ACL directory also. The utilities only work in a passworded directory if there is no owner password on the current directory or on the parent directory.

This new scheme reads and then changes the RWLOCK on the MIDASPLUS file to EXCLUSIVE before a utility is allowed access to the file. Any user who wishes to use the MIDASPLUS utilites on an existing file must have the following ACL rights:

LUP rights on the current directory

LU rights on the parent directory

LURW rights on the MIDASPLUS file

Configuration Parameters

MIDASPLUS has available configuration parameters that may be set upon system initialization. MIDASPLUS is initialized by the IMIDASPLUS command contained in the MIDASPLUS.SHARE.COMI COMINPUT file.

The IMIDASPLUS command sets some configuration parameters as directed in a configuration file (if specified) and makes the appropriate initialization call to the MIDASPLUS library. Below is a description of the configuration file and how to utilize it.

IMIDASPLUS allows a configuration file treename to be given on the command line, or optionally displays a list of the available configuration parameters if "-HELP" is supplied on the command line. If a treename is not specified, IMIDASPLUS assumes that the configuration file to be utilized is the file MPLUS.CONFIG in the current directory. If the MPLUS.CONFIG file does not exist, then IMIDASPLUS assumes that all system configuration defaults should be followed and continues the initialization.

The configuration file may contain a number of directives that specify MIDASPLUS configuration parameters. These parameters are contained in standard text, one-per-line, within the file. If an error is encountered, IMIDASPLUS prints an error message, assumes the default, and continues the initialization. The available configuration directives are described in the paragraphs that follow. Any parameter not specified by a directive assumes its default.

SEMAPHORE [semaphore_number]

Specifies the PRIMOS semaphore given by semaphore_number utilized for user synchronization. The default for semaphore_number is -14.

DEBUG [ON]
[OFF]

Controls MIDASPLUS debug execution and print options for developer debug. The default for this parameter is OFF. This parameter sets debug control on a system wide basis. It may be set on a per user basis by calls to MSGCTL.

FUNITS [max_number_file_units]

Specifies the maximum number of file units per user that MIDASPLUS will use for MIDASPLUS subfiles. (This number does not include file units that are used for main MIDASPLUS segment directories.) The default is 256, the max is 512. The value specified should not be less than the maximun number of MIDASPLUS segment directories a user is likely to have open at a time. In most cases, the value should be at least four times that number. This will allow for four subfiles per MIDASPLUS file to be open.

TIMEOUT [seconds]

Specifies the number of seconds that any user will wait for some internal resource (e.g., locks, buffers) before assuming that the system is hung and aborting the current operation. The argument seconds should be a non-negative integer expressing the maximum number of seconds to wait for any resource. If seconds is zero, then no timeout will occur and the user will wait indefinitely. The default value is 300 seconds or 5 minutes.

PRINT_ERROR [ON]
[OFF]

Specifies whether MIDASPLUS should print error messages when fatal errors occur. If ON is given, MIDASPLUS will print the MIDASPLUS error code for any fatal type of error encountered. If a PRIMOS system call error was encountered, it will also print the system error message. Specifying OFF will cause MIDASPLUS to not print any error messages. The default for this parameter is ON. This parameter sets the error print control on a system wide basis. It may additionally be controlled on a per user basis by calls to MSGCTL.

BUFFERS [buffer_count]

Specifies the number of internal file buffers that MIDASPLUS is to utilize. The value of buffer_count must be between 2 and 64, inclusive. Giving a low value will utilize less working-set, but increase the number of system I/Os and user wait time for a buffer. Giving a large value will decrease the user wait times, but utilize more working-set. The default value for buffer_count is 64. It is recommended that this parameter not be changed except on systems with few MIDASPLUS users and limited memory.

REMOTE_TRANSMIT [ON]
[OFF]

When set ON, allows outgoing MIDASPLUS access to remote files. When set OFF disallows outgoing remote calls. The default for this parameter is ON.

REMOTE_RECEIVE [ON]
[OFF]

When set ON, allows incoming requests from other network nodes for access to MIDASPLUS files on this system to be processed. When set OFF, incoming requests are denied. The default for this parameter is ON.

REPORT_DUPS [ON]
[OFF]

When set ON, the reporting of the existence of duplicate entries by returning a value of one in the returned status code (word one of the ARRAY) is enabled. When set OFF, a status code of one is never returned. This facility is useful since MIDAS was inconsistent in returning the duplicate indicator and many applications which consider a non-zero returned status to be an error might no longer operate in the same fashion with MIDASPLUS.

The default value for this parameter is ON. The parameter is setting a system wide control on the returning of duplicate indicators. It may be controlled on a per user basis by calls to MSGCTL.

REPORT_LOCKED [ON]
[OFF]

When set ON, the reporting of locked records when performing read operations is enabled (i.e. FIND$, NEXT$ operations). The fact that the record which has been read is locked is indicated by the value of Bit 5 in array word 13 being set to 1 (as in MIDAS). This should not be confused with Bit 1 of array word 10, which is set to 1 to indicate successful operation of a lock record call.

When set to OFF (the default) Bit 5 of array word 13 is always set to zero, regardless of the record's locked status. This directive sets the system-wide state for locked record reporting. It may be set on a per-user basis by MSGCTL.

SPY_FNAMES [OFF]
[ON]

When set ON will save the filenames of open MIDASPLUS files for SPY to use when displaying which files have record locks. When set OFF will not save the filenames. The default is OFF.

INIT_USER_COMMON [OFF]
[ON]

When set ON (default), per user common information is re-initialized at each program re-entry. When set OFF per user common information is not re-initialized at each program re-entry. However, the use of setting this switch OFF is not recommended because it allows MIDASPLUS files to be closed without updating the MIDASPLUS per user and system shared information.

SYSLOG [ON]
[OFF]

When SYSLOG is set on, error logging is enabled. When it is set off (default), no errors will be logged to the system error log. For more information, see the description in the section Detailed Error Logging.

SYSTRACE

Each SYSTRACE directive will cause one or more MIDASPLUS data structures to be traced to the system error log file. If no SYSTRACE directive is given (default), no traces occur. Error logging (SYSLOG) must be enabled in order to use SYSTRACE. Each SYSTRACE directive must be on the same line and separated by spaces. The currently recognized directives, and the data structures they control, are:

FMS	- System Error Variable
USRCOM	- Miscellaneous USRCOM variables
UCA	- User Communication Array (USRCOM)
BUFFERS	- Per user buffer info
GDUCA	- USRCOM GDATA$ user communication array
BCB	- Buffer control block
DRLCOM	- Data record locking common
FILCOM	- File common
FS	- File table structure
STK	- MIDASPLUS stack
CUR	- Current operation variables
ALL	- All of the above

For example, to trace the User Communication Array and the MIDASPLUS stack, use the directive:

```
SYSTRACE USRCOM STK
```

POWERPLUS

- POWERPLUS now supports the PRIME Extended Character Set (ECS).
- POWERPLUS has eliminated R-mode dependency. All the modules are compiled in V-mode. POWERPLUS>TOOLS>TERM.INFO provides information for adding a new terminal type.
- Rev. 21 POWERPLUS must be installed with Rev 21 PRIMOS, FORTRAN and MIDASPLUS.
- You must use the install command file supplied with POWERPLUS in order to install Rev. 21.0 POWERPLUS.

PRISAM

PRISAM at Rev. 21.0 has enhancements in the following areas:

- Remote Transactional Access
- Z$OPEN
- Dynamic Transaction Allocation
- New error codes
- New SYSCOM files (for C and Pascal)

Remote Transactional Access

At Rev. 21.0, PRISAM provides runtime access to remote transactional PRISAM files. This functionality allows you to access PRISAM files located on different systems within the PRIMENET network.

Note that PRISAM is not providing distributed functionality and that each transaction is limited to accessing a single system. If attempts are made to access files on different systems within the same transaction, those attempts will fail with the error code ER$IDT (Invalid Distributed Transaction). Since this is a ROAM restriction, mixed mode transactions will behave in a similar manner.

Existing user applications do not need to be changed as a result of this new remote functionality. However, in order to use the new functionality, the changes described below must be made.

Z$OPEN

All PRISAM documentation now refers to the tranmode argument as the verifymode argument. This change has been made to reflect the expanded functionality associated with this argument. The change affects only the name; Z$OPEN's declaration and argument list remains the same. Associated with this documentation change, the error mnemonic, ER$ITM (Invalid Tranmode Argument), is supplemented with the new error mnemonic, ER$IVM (Invalid Verifymode Argument). To remain compatible with earlier revisions of PRISAM, the ER$ITM mnemonic will continue to exist in the PRISAM SYSCOM files. Also, the values associated with both mnemonics are identical.

The verifymode argument now recognizes a new additive key, O$LOC, defined in the PRISAM SYSCOM files. This key is optional. When O$LOC is specified, Z$OPEN will only open a local file. If the file found by Z$OPEN (with O$LOC specified) is remote (that is, exists on a different system), the error code ER$FNL (File Not Local) will be returned to the user. The syntax for the verifymode argument is:

```
verifymode = <O$NCK> + <O$TRM> + <O$LOC>
```

where at least one of the three keys must be specified.

Dynamic Transaction Allocation

The internal resource allocation associated with each transaction has been moved to the first transactional file access within each transaction. Therefore, Z$STRT will no longer timeout while starting a transaction and will no longer return the ER$TIM error code.

New Configuration Variables

PRISAM will recognize two new configuration variables, REMOTE_TRANSMIT and REMOTE_RECEIVE. These variables expect boolean values. Setting REMOTE_TRANSMIT to TRUE allows local users to access remote PRISAM files. Setting REMOTE_RECEIVE to TRUE allows remote users to access local PRISAM files.

CPRISAM

The PRISAM configuration utility, CPRISAM, contains additional prompts to set the new configuration variables.

IPRISAM

The PRISAM initialization utility, IPRISAM, recognizes the two new variables. If these variables are not present in the PRISAM configuration file, IPRISAM will default to initialize these system variables to FALSE.

Display Shared Memory

The values for these system variables can be viewed by using FAU to display PRISAM's shared memory.

New Error Codes

Following are error codes added to PRISAM at Rev. 21.0:

ER$IDT (Invalid Distributed Transaction) - An attempt was made to access a different system from the system where the transaction is defined. Returned by Z$READ, Z$FIND, Z$INSR, Z$UPDT, Z$KDEL, and Z$DELE.

ER$FNL (File Not Local) - This error is returned by Z$OPEN when attempting to open a remote file and, either the local system is not configured to transmit remotely, or the verifymode argument has specified the O$LOC key. Returned by Z$OPEN.

ER$NRR (No Remote Receive) - The remote system accessed is not configured to receive remote access. Returned by Z$OPEN.

ER$DDM (Unspecified DDM error) - 'DDM' stands for Distributed Data Management and is a part of the ROAM system. It provides the underlying communications primitives utilized by PRISAM to provide remote functionality. ER$DDM indicates that the DDM system has returned an unexpected error. Please refer to the PRISAM error log and the DDM error log in DDM* for more information. This error is specific to remote access only. Returned by Z$OPEN, Z$CLOS, Z$READ, Z$FIND, Z$INSR, Z$UPDT, Z$KDEL, Z$DELE, and Z$KYST.

ER$RMT (ROAM Error on Remote System) - One or more ROAM errors were encountered on a remote node while ending or aborting a transaction. Refer to local PRISAM error log for more information.

ER$RSU (Remote System is Unavailable) - The remote system has become unavailable. This can result from a variety of reasons, including network trouble and problems with the remote system. If a transaction was active, it has been aborted. All files opened on the particular system, have been closed. Refer to the PRISAM error logs on the remote system and on the local system for more information. This error is specific to remote access only. Returned by Z$OPEN, Z$CLOS, Z$READ, Z$FIND, Z$INSR, Z$UPDT, Z$KDEL, Z$DELE, and Z$KYST.

New SYSCOM files The following new insert files have been added in the SYSCOM directory for PASCAL and C language users:

PRISAM_USER_DCLS.INS.CC
PRISAM_USER_ERRS.INS.CC
PRISAM_USER_ERRS.INS.PASCAL
PRISAM_USER_KEYS.INS.CC
PRISAM_USER_KEYS.INS.PASCAL

Security

With the addition of remote functionality, new security measures should be considered to protect your database from unauthorized users. Users can protect their database(s), as they protect any other PRIMOS file, by appropriate use of ACLs and by forcing network validation.

In addition, PRISAM provides two configuration variables which will allow users to protect their local PRISAM files from remote users (REMOTE_RECEIVE) or to restrict local users to accessing only local PRISAM files (REMOTE_TRANSMIT).

Compatibility

Rev. 21.0 PRISAM is fully compatible with earlier versions of PRISAM. However, in order to use the remote functionality, the following configuration must be present:

- The local system and all remote systems to be accessed must be running compatible versions of Rev. 21.0 or later PRISAM and ROAM. If not, the error ER$RSU will be returned.
- If the application uses mixed mode functionality, the DBMS system must also be at Rev 21.0 or later. If not, DBMS will abort the process.
- The local system must have configured REMOTE_TRANSMIT to TRUE. If not, the error ER$FNL will be returned.
- All remote systems to be accessed must have configured REMOTE_RECEIVE to TRUE. If not, the error ER$NRR will be returned.
- The local user must have the appropriate access rights to the remote files. If network validation is enforced, the user must have used ARID to add a valid remote ID. If not, the user will receive an invalid access rights error.
- All remote files to be accessed must reside on a partition that is visible from the local system's disk table. If not, the file cannot be found.
- When you open a PRIMOS file without specifying the partition name, PRIMOS will open the first file found in the disk table that matches the pathname specified. To remove the possibility of this type of mistake occurring with your database, the use of fully-qualified pathnames is recommended.

Note

Ordinarily, warmstarting the system does not recover ROAM files. The only exception is when a forced shutdown has caused the halt and the TPDUMP directive is enabled in the PRIMOS configuration file. If this is the case, PRIMOS displays the following at the supervisor terminal:

*** From PRIMOS: Please take a crash dump, then warmstart.

Warmstarting after this message has been displayed completes the disk I/O that was unfinished at the time of the forced shutdown. However, you must still perform a coldstart once the warmstart is complete. For more details, see the CPU handbook for your machine.

Environment

PRISAM Rev. 21.0 requires PRIMOS Rev. 21.0 and ROAM Rev. 21.0. No previous revs of PRIMOS or ROAM will work correctly with this rev of PRISAM. PRISAM requires the use of shared segments 2204, 2205, 2207, and 2227, and of per-user static storage segment 6007, addresses from 50000 to 122777.

FILES ON SYSTEM TAPE

The PRISAM Directory

The top level directory contains the following files:

- PRISAM.INITINSTALL.COMI - used to install PRISAM for the first time
- PRISAM.INSTALL.COMI - used to install PRISAM at any other time
- PRISAM.SHARE.COMI used to share and initialize the PRISAM shared segments.

PRISAM>CMDNC0: This directory holds the CPL interludes to those V-mode segment directories found in PRISAM>SEGRUN* which will become external commands to PRIMOS. These interludes are moved to CMDNC0 on the command partition by PRISAM.INSTALL.COMI.

DIAG.SAVE FAU.SAVE

PRISAM>INFO: This directory contains the documentation for the current release of PRISAM, PRISAM.RUNI and PRISAM.RUNO.

PRISAM>LIB: This directory contains the PRISAM binary object libraries: PRISAMLIB.BIN (the shared PRISAM runtime library) and FAULIB.BIN (the unshared PRISAM utilities library).

PRISAM>PRISAM*: This directory contains the PRISAM error message file, ERROR_TEXTS.ERR, the PRISAM rev stamp file, REV_NUM, segment directories used for configuring and initializing PRISAM, the Rev 1 to Rev 2 File Conversion Utility, and the HELP directory for the PRISAM File Administrator Utility.

ERROR_TEXTS.ERR REV_NUM

CPRISAM.SEG IPRISAM.SEG REV2_CONVERT.SEG SPRISAM.SEG

HELP

PRISAM>SEGRUN*: This directory contains the segment directories for the commands in PRISAM>CMDNC0:

DIAG.SEG FAU.SEG

PRISAM>SYSCOM: This directory contains language source files which may be included into user programs. These files define the entry points, user keys, and error codes for PRISAM. The installation procedure copies these files to SYSCOM on the command partition.

PRISAM_USER_DCLS.INS.PL1
PRISAM_USER_DCLS.INS.CC
PRISAM_USER_ERRS.INS.FTN
PRISAM_USER_ERRS.INS.PL1
PRISAM_USER_ERRS.INS.PMA
PRISAM_USER_ERRS.INS.CBL
PRISAM_USER_ERRS.INS.CC
PRISAM_USER_ERRS.INS.PASCAL
PRISAM_USER_KEYS.INS.FTN
PRISAM_USER_KEYS.INS.PL1
PRISAM_USER_KEYS.INS.PMA
PRISAM_USER_KEYS.INS.CBL
PRISAM_USER_KEYS.INS.CC
PRISAM_USER_KEYS.INS.PASCAL

PRISAM>SYSTEM: This directory contains the PRISAM shared segment run files:

PR2204 PR2205 PR4000

INSTALLING PRISAM

If your site uses PRISAM, follow the instructions below or the duplicate instructions in the installation appendix in the PRISAM User's Guide.

Initial Installation

After installing the current version of ROAM using the preceding procedure, you can install PRISAM for the first time by doing the following:

1. Attach to the MFD where you wish to place the PRISAM system.

2. Restore the directories supplied on the tape.

3. Run the command file that creates the PRISAM* directory and installs PRISAM. To accomplish this, enter the following at the supervisor terminal:

```
CO PRISAM>PRISAM.INITINSTALL.COMI
```

4. If the PRISAM* directory is to have ACL protection, set the specific ACLs, SYSTEM:ALL and $REST:ALURW. Alternatively, for $REST, you can specify the names of all individuals and groups that use PRISAM and give each ALURW rights. These rights are the minimum required for PRISAM users. Then set specific ACLs for each PRISAM system file installed on PRISAM* to SYSTEM:ALL and $REST:LUR. These six files are mentioned in Chapter 12 of the PRISAM User's Guide, PRISAM SYSTEM FILES AND ACLS, and must not be modified by users.

   If the PRISAM* directory is to be a password-protected directory (not recommended), it is safest to allow all users to be owners, having at least RW rights. If PRISAM users must be non-owners, they must have at least RW rights.

5. Run the command input file that shares PRISAM. To accomplish this, enter the following at the supervisor terminal:

```
CO SYSTEM>PRISAM.SHARE.COMI
```

6. If you want to modify any of the preset configuration parameters, you may do so by following the procedures in the relevant section of the installation appendix in the PRISAM User's Guide.

Reinstallation

After installing the current version of ROAM using the preceding procedure, you can reinstall PRISAM by doing the following:

1. Attach to the MFD where the PRISAM system files reside.

2. Restore the directories supplied on the tape.

3. Run the command file that installs PRISAM. To accomplish this, enter the following at the supervisor terminal:

```
CO PRISAM>PRISAM.INSTALL.COMI
```

4. If the PRISAM* directory is to have ACL protection, set the specific ACLs, SYSTEM:ALL and $REST:ALURW. Alternatively, for $REST, you can specify the names of all individuals and groups that use PRISAM and give each ALURW rights. These rights are the minimum required for PRISAM users. Then set specific ACLs for each PRISAM system file installed on PRISAM* to SYSTEM:ALL and $REST:LUR. These six files are mentioned in Chapter 12 of the PRISAM User's Guide, PRISAM SYSTEM FILES AND ACLS, and must not be modified by users. If the PRISAM* directory is to be a password-protected directory (not recommended), it is safest to allow all users to be owners, having at least RW rights. If PRISAM users must be non-owners, they must have at least RW rights.

5. Run the command input file that shares PRISAM. To accomplish this, enter the following at the supervisor terminal:

```
CO SYSTEM>PRISAM.SHARE.COMI
```

6. If you want to modify any of the preset configuration parameters, you may do so by following the procedures in the relevant section of the installation appendix of the PRISAM User's Guide.

CHAPTER 5

COMMUNICATIONS PRODUCT ENHANCEMENTS

The following Communications products are new or have enhanced functionality at Rev. 21.0:

- DPTX
- Distributed System Management (DSM)
- Intelligent Communications Subsystems, Model 2 (ICS2)
- Intelligent Communications Subsystems, Model 3 (ICS3)
- Network Terminal Service (NTS)
- Local Area Network (LAN300) Management Facility, including Special Considerations
- PRIMENET
- X.25

DPTX

The following modules are now EPFs loaded via BIND instead of SEG: PTDSC, PT45DSC, PT46DSC, DPTXMTR, PRTDSC, CWLDSC.

Prime ECS supports PTDSC with PT200 terminals that have newer revisions of firmware.

INTELLIGENT COMMUNICATIONS SUBSYSTEMS, MODEL 2

PRIMENET X.25 Half-Duplex functionality has been added to the ICS2 at Rev. 21.0.

INTELLIGENT COMMUNICATIONS SUBSYSTEMS, MODEL 3

PRIMENET X.25 Half-Duplex functionality has been added to the ICS3 at Rev. 21.0.

NETWORK TERMINAL SERVICE (NTS)

NTS is a new product at Rev. 21.0. It provides a mechanism to connect asynchronous terminals, printers, and other devices to Prime computers through a Local Area Network. It provides an alternative to directly connecting terminals through the RS232/asynchronous interface provided by the AMLC or ICS controllers.

There are seven Ring 3 NTS commands. They are CONFIG_NTS, START_NTS, STOP_NTS, NTS_ASSOCIATE, NTS_UNASSOCIATE, NTS_LIST_ASSOCIATE, and NTS_LINE. These commands are briefly discussed below.

Before you attempt to use NTS, make sure the PRIMOS CONFIG file contains certain directives that initialize NTS on your system. The directives include: LHC, NTSUSR, NTSASL, NTSBUF and NTSABF. The system must be coldstarted after adding any appropriate directives. For more information on these directives, refer to the System Administrator's Guide, Volume II: Communications Lines and Controllers.

To use NTS, you must describe the system configuration using the CONFIG_NTS program. The configuration components are:

1. the Local Area Network, or LAN300 —
   the physical medium that interconnects terminals and Prime computers. At Rev. 21.0, the IEEE 802.3-based LAN300 is supported.

2. the LAN Terminal Server, or LTS —
   a Prime supplied device that interfaces asynchronous terminals with a LAN.

3. the Prime host —
   a Prime computer connected to one or more LANs.

4. the LAN Host Controller, or LHC —
   a board that plugs into the Prime computer backplane and connects the Prime host to the LAN. A single host may have up to 4 such controllers for NTS, each being connected to the same or different LAN300s.

For more information on CONFIG_NTS and planning an NTS network, refer to the NTS Planning and Configuration Guide.

Once an NTS configuration file has been created, the START_NTS command can be used on a host, which allows users to login from an LTS (provided NTSUSR is not zero). It also allows for assigned lines to an LTS (provided NTSASL is not zero). The command STOP_NTS terminates such activity.

The network administrator can map and unmap LTS lines using the commands NTS_ASSOCIATE and NTS_UNASSOCIATE. This is a precursor to assigning an LTS line. Any user can display the active associations using the NTS_LIST_ASSOCIATE command.

The NTS_LINE command allows an LTS terminal user to escape back to LTS command mode.

For more information on these commands, refer to the NTS User's Guide.

Local Area Network (LAN300) Management Facility

The LAN300 Management Facility is new at Rev. 21.0. It is a tool for the real-time operation, control and analysis of a Prime LAN300 (IEEE 802.3-based) installation.

The LAN300 Management Facility is implemented via a collection of software modules that run on multiple stations throughout the network. This group of programs works in unison to allow both automatic and manual monitoring of the network's operation, and to provide real-time services to users.

The LAN300 Management Facility supports the operation of PRIMENET and Network Terminal Service (NTS) on a LAN300 installation. The Management Facility is automatically initiated by starting up either one of the above communication services.

LAN300 Management Facility software allows the operator to control, query and test network components from central and remote locations. This software also allows the operator to collect event and performance information required to maintain the daily operation of the network, as well as to provide future network planning capability.

Special Considerations For LAN300

There are several things that you should be aware of when using the LAN300 product. Most of these are procedural items that merit care when configuring and starting the system. These items are listed below under the major product function which they affect.

PRIMENET:

- Make sure that the LHC directive has been added to CONFIG before attempting to start PRIMENET on an IEEE 802.3 LAN.

- Make sure that the number of phantom users configured in CONFIG (NPUSR) is sufficient to support the various Rev 21 servers. The minimum to run PRIMENET and NTS concurrently is 11 (NPUSR 13). This will allow all system phantoms to run. Additional user phantoms, if they are desired, must be added to this number.

- Make sure that the appropriate version of DSM has been installed and started prior to starting PRIMENET on an IEEE 802.3 LAN.

- Make sure the the LHC has been downline loaded using COMM_CONTROLLER prior to starting PRIMENET on an IEEE 802.3 LAN.

- Care should be taken when using the Rev 21 versions of CONFIG_NET and START_NET. The Rev 21 version of CONFIG_NET can be used to edit a configuration file created by the pre-Rev 21 version of CONFIG_NET, however the opposite is not true. Also, a configuration file created with the Rev 21 version of CONFIG_NET must be started up using the Rev 21 version of START_NET.

- When using PRIMENET/IEEE 802.3 and NTS, be sure that LAN names and host names are used consistently between the CONFIG_NET and CONFIG_NTS configuration files. Failure to do this can result in undefined network operation.

NTS:

- Make sure that the LHC directive has been added to CONFIG before attempting to start NTS.

- Make sure that the NTSUSR and/or NTSASL directives are properly configured in CONFIG.

- Make sure that the number of phantom users configured in CONFIG (NPUSR) is sufficient to support the various Rev 21 servers. The minimum to run PRIMENET and NTS concurrently is 11 (NPUSR 13). This will allow all system phantoms to run. Additional user phantoms, if they are desired, must be added to this number.

- Make sure that the appropriate 3 version of DSM has been installed and started prior to starting NTS.

- Make sure the the LHC has been downline loaded using COMM_CONTROLLER prior to starting NTS.

- When using NTS and LAN, the LAN names and host names must be used consistently between the CONFIG_NTS and CONFIG_NET configuration files or undefined network operation may result.

LAN300 Network Management:

- Be sure to configure the DSM Unsolicited Message facility to handle LAN300 Network Management events. Using CONFIG_UM, setup three private logs (NMSR.LOG, DLL.LOG & ULD.LOG) in the directory NETWORK_MGT*. Also, use EDIT_ACCESS to give DSMASR "ALL" access rights on the NETWORK_MGT* log directory.

- Make sure that the NTS services provided by LAN300 Network Management are configured on a per LAN300 network basis. Using CONFIG_NTS, configure the primary and secondary host systems that will support the LTS300 downline load and event logging services per LAN300. Note that although it is listed in CONFIG_NTS, LTS300 up-line dump services are not supported in the beta release of the LAN300 Network Management product.

- Note that only USER 1 and members of the .NETWORK_MGT$ ACL group are allowed to invoke the LAN300 Network Management commands (COMM_CONTROLLER, LIST_LHC_STATUS, LIST_LTS_STATUS and LOOPBACK). Using EDIT_PROFILE, configure the network operators and administrators for the .NETWORK_MGT$ ACL group.

- If the DSM System Manager should abnormally logout, it is important to stop and start DSM in order to resume proper logging of LAN300 Network Management event messages.

Components

The following are provided by the LAN300 Management Facility:

- LHC300 downline load and upline dump services
- LHC300 polling and recovery function
- LTS300 downline load services
- LTS300 user services
- LHC300/LTS300 event logging services
- LIST_LHC_STATUS command
- LIST_LTS_STATUS command
- LOOPBACK command

The following sections provide further information on these components.

LHC300 Downline Load and Upline Dump Services

LAN Host Controller (LHC300) downline load and upline dump services provide boot and dump capabilities for local LHC300 controllers.

Downline load, or booting, is the process of providing an intelligent device (with no permanent storage capability of its own) with its software execution image. LHC300 controllers must be booted by their local host systems before they can operate. Prior to booting, a self-verify of the LHC300 controller is performed to insure its ability to operate properly.

Upline dump, or dumping, is the process of retrieving the current memory image from a failed intelligent device for the purpose of reviewing the image to determine why the controller failed. Reviewing a controller's memory image is a difficult task and requires detailed knowledge of the controller's software image.

The LHC300 downline load and upline dump services are only supported for local LHC300 controllers and are not supported over the network. The boot and dump functions are performed by transient programs that execute on host systems. These "transient servers" are initiated by either the COMM_CONTROLLER command or the LHC300 Polling and Recovery mechanism.

LHC300 downline load and upline dump services will provide the following functions:

LHC300 prom self-verify analysis

LHC300 downline load

Verification of LHC300 operating system operation

LHC300 upline dump

Event logging to DSM

LHC300 downline load and upline dump threshold monitoring

LHC300 Polling and Recovery Function

The LAN300 Server monitors local LHC300s, detects any failures, and recovers from those failures without manual intervention. The goal of the mechanism is to increase controller up-time by shortening the LHC300 failure detection and recovery time through the use of an automatic detection/recovery mechanism.

Manual detection requires that a network user notice impaired network performance and then report the problem to a network operator. The network operator must then isolate the problem and initiate manual

intervention. Both manual detection and recovery may take users a significant amount of time. The automatic detection/recovery mechanism will shorten the detection/recovery time without the need of human intervention.

LTS300 Downline Load Services

LAN300 Terminal Server (LTS300) downline load services provide boot capabilities for LTS300 servers on the network. Downline load, or booting, is the process of providing an intelligent device (with no permanent storage capability of its own) with its software execution image. LTS300 servers must be booted by designated host systems on the network before they can operate. Prior to booting, a self-verify of the LTS300 server is performed to insure its ability to operate properly.

The LTS300 downline load functions are performed by a transient program that executes on host systems. This "transient server" is initiated by either the COMM_CONTROLLER command or the LTS300 powerup/restart logic.

LTS300 downline load services will provide the following functions:

LTS300 prom self-verify analysis

LTS300 downline load

Verification of LTS300 operating system operation

Event logging to DSM

LTS300 downline load threshold monitoring

LTS300 User Services

LAN300 provides several user services which support the NTS operation on the LTS300. The services currently provided are a "Host Locator" service and a "Host Listing" service. These services are described in the following sections.

"Host Locator" Service: The "Host Locator" service on the LTS300 supports the LTS300 Packet Assembler/Disassembler (PAD) software, a component of NTS, in providing a network-wide naming service that allows users to employ symbolic names to access hosts throughout the network. One of the motivating factors in the design of the Host Locator service is the implementation of a user-friendly interface to the network. Another less obvious factor is the provision of locational transparency; that is, the physical location of a host need not be known in order to access it.

"Host Listing" Service: The "Host Listing" service on the LTS300 supports the LTS300 PAD software in providing a network-wide listing service that allows users to obtain a current status of available hosts that may be access through the network. The LTS300 PAD application utilizes the Host Listing service to supply network users with status displays.

LAN300 Event Logging Services

The LAN300 Event Logging services provides the mechanism through which event conditions that arise within the LAN300 software may be reported to the DSM event logging facility. Event conditions are defined as informational, warning or alarm states which are judged to warrant reporting to a centralized log. The centralized log may be used by a network operator to analyze network operation.

Event reports which originate within the LAN300 server are logged directly to DSM. Event reports which originate within the LHC300 or LTS300 software cannot be logged directly to DSM. Consequently, these reports are sent to a LAN300 server. The server is responsible for logging the events to DSM.

The LAN300 event logging services has five components:

Administering log files (provided by DSM)

Configuring where to log events (provided by DSM)

Logging events (provided by LAN300 and DSM)

Reporting events (provided by DSM)

Message internationalization (provided by LAN300 and DSM)

LIST_LHC_STATUS Command

The LIST_LHC_STATUS command is a PRIMOS command that provides network users the ability to manually solicit and examine status information pertaining to an LAN300 host controller. The online operational status information retrieved from an LHC300 can be used to determine up-to-date utilization and general status of the LHC300 located either local to the host system or remotely on the network. This function relys upon a request/response handshake between the command software in PRIMOS and the local network management agent on the LHC300.

The information presented by the LIST_LHC_STATUS command is categorized into three screens.

- An initial or overview screen provides identification and performance data which reports the general health of the server (with the -PERFORMANCE option).

- A second screen displays summary and itemized connection data (with the -CONNECTION option).

- A third screen details operating system and management status information (with the -MANAGEMENT option).

All available screens may be retrieved via a single LIST_LHC_STATUS command issued with the -ALL option.

All information retrieved by this command is obtained directly from the specified controller, not internal PRIMOS data structures. This fact is important because it guarantees that the data displayed will always properly reflect the information as currently known by the controller.

Examples: The following examples show the use of various options and the display of the -PERFORMANCE, -CONNECTION and -MANAGEMENT screens for the LIST_LHC_STATUS command. Notice how the LHC is identified in each example.

-PEFORMANCE Example

```
OK, LIST_LHC_STATUS -DEST_NODE_NAME HOST1 -DEST_LHC_NUMBER 2 -PERFORMANCE
[LIST_LHC_STATUS Rev. 21.0 Copyright (c) 1987, Prime Computer, Inc.]
Host Name: HOST1          LHC Number:  02    Address: 08-00-2F-00-01-02
Hw/Fw Rev: 08.00/02.14     Slot Number: 13    Device Address: 32

State     : OPERATIONAL          Active Protocols: PRIMENET/NIS
LAN Name  : LAN1
Load File: <LNMPAK>DOWN_LINE_LOAD*>LHC.DL
Load File Rev: 01.01
Performance Statistics                                        Count
------------------------------------------------------------------
Active connections                                      :     5
Frames transmitted                                      :     8014
Frames transmitted with error                           :     6
Frames retransmitted                                    :     6
Frames received                                         :     11118
Frames received with network (CRC/Alignment) error      :     0
Frames lost due to internal (resource) error            :     1
Percent CPU idle time                                   :     89.3%
Current percent data buffer availability                :     96.4%
Lowest  percent data buffer availability                :     95.8%
Alarms reported                                         :     0
OK,
```

-CONNECTIONS Example

```
OK, LIST_LHC_STATUS -DEST_NODE_ADDRESS 00-01-02  -CONNECTION
[LIST_LHC_STATUS Rev. 21.0 Copyright (c) 1987, Prime Computer, Inc.]
Host Name: HOST1          LHC Number: 02     Address: 08-00-2F-00-01-02

Current active connections   :  5     Total connection count      : 25
Current active multicast addr:  3     Greatest connection count   :  7

Primenet active connections  :  2     NTS active connections      :  3
Primenet inactive connections:  4     NTS transitional connections:  0

ACTIVE CONNECTIONS
Type       State    Line no    Remote Station          Data Characters Tx/Rx
-------------------------------------------------------------------------
PRIMENET   Active     —         HOST2/LHC00               82674/109363
PRIMENET   Active     —         HOST3/LHC07                 871/1982
NTS        Active    1024       LTS1/PORT1                 197/276
NTS        Active    1025       LTS1/PORT5                  87/285
NTS        Connect   1026       LTS1/PORT6                  15/34
```

-MANAGEMENT Example

```
OK, LIST_LHC_STATUS -DNN HOST1 -LAN_NAME LAN1 -MANAGEMENT
[LIST_LHC_STATUS Rev. 21.0 Copyright (c) 1987, Prime Computer, Inc.]
Host Name: HOST1          LHC Number: 02     Address: 08-00-2F-00-01-02

OPERATING SYSTEM STATISTICS
  Running Time: 00:05:41:16
  Percent CPU Usage  —  Idle: 89.3%
    LLC: 06.2%     Primenet: 01.6%     NTS: 02.4%     Net Mgmt: 00.5%
  Error Counts  —  Total: 0
    Correctable memory errors: 0          Watchdog timer expired: 0

  Active I/O Bus connections: 7
    I/O Bus buffers Tx/Rx    : 5866/5824
    I/O Bus commands Tx/Rx   : 324/12
    I/O Bus bytes Tx/Rx      : 1205441/179696

NETWORK MANAGEMENT STATISTICS
  Alarms reported  —  Total: 0
    Logged: 0    Suppressed: 0    Dropped: 0
    OS: 0    LLC: 0    Primenet: 0    NTS: 0    Net Mgmt: 0

  Primary management support: MANAGEMENT      NTS              PRIMENET
  Primary NTS support       : LOG             DUMP             BOOT
OK,
```

LIST_LTS_STATUS Command

The LIST_LTS_STATUS command is a PRIMOS command that provides network users the ability to manually solicit and examine status information pertaining to an LAN300 terminal server. The online operational status information retrieved from an LTS300 can be used to determine up-to-date utilization and general status of the LTS300 located on the network. This function relys upon a request/response handshake between the command software in PRIMOS and the local network management agent on the LTS300.

The information presented by the LIST_LTS_STATUS command is categorized into three screens.

- An initial or overview screen provides identification and performance data which reports the general health of the server (with the -PERFORMANCE option).
- A second screen displays summary and itemized connection data (with the -CONNECTION option).
- A third screen details operating system and management status information (with the -MANAGEMENT option).

All available screens may be retrieved via a single LIST_LTS_STATUS command issued with the -ALL option.

All information retrieved by this command is obtained directly from the specified server, not internal PRIMOS data structures. This fact is important because it guarantees that the data displayed will always properly reflect the information as currently known by the server.

Examples: The following examples show the use of various options and the display of the -PERFORMANCE, -CONNECTION and -MANAGEMENT screens for the LIST_LTS_STATUS command. Notice how the LTS is identified in each example.

-PERFORMANCE Example

```
OK, LIST_LTS_STATUS -DEST_NODE_NAME LTS1 -PERFORMANCE
[LIST_LTS_STATUS Rev. 21.0 Copyright (c) 1986, Prime Computer, Inc.]
LTS Name:  LTS1                  Address: 08-00-2F-F8-00-11
                                 Active Ports: 3
Hw/Diag/Dll Rev: 01.00/01.00/01.00
State       : OPERATIONAL        Active Protocols: NTS
Booting Host: HOST1              LAN Name : LAN1
Load File   : <LNMPAK>DOWN_LINE_LOAD*>LTS.DL
Load File Rev: 01.00
Performance Statistics                                          Count
-------------------------------------------------------------------------
Active connections                                        :     3
Frames transmitted                                        :     7372
Frames transmitted with error                             :     0
Frames retransmitted                                      :     0
Frames received                                           :     32464
Frames received with network (CRC/Alignment) error        :     0
Frames lost due to internal (resource) error              :     0
Percent CPU idle time                                     :     86.5%
Current percent data buffer availability                  :     94.4%
Lowest  percent data buffer availability                  :     89.6%
Alarms reported                                           :     0
OK,
```

-CONNECTION Example

```
OK, LIST_LTS_STATUS -DEST_NODE_ADDRESS F8-00-11 -CONNECTION
[LIST_LTS_STATUS Rev. 21.0 Copyright (c) 1986, Prime Computer, Inc.]
LTS Name: LTS1                          Address: 08-00-2F-F8-00-11

Current active connections   : 3     Total connection count   : 8
Current active multicast addr: 2     Greatest connection count: 3

ACTIVE CONNECTIONS
State    Port   Remote Station    Data Characters Tx/Rx    Duration
-------------------------------------------------------------------
Active   01     HOST1/LHC02              357/204          00:02:06:45
Active   05     HOST1/LHC02              314/94           00:00:56:12
Active   06     HOST1/LHC02               85/58           00:00:00:34
OK,
```

-MANAGEMENT Example

```
OK, LIST_LTS_STATUS -DEST_NODE_NAME LTS1 -MANAGEMENT
[LIST_LTS_STATUS Rev. 21.0 Copyright (c) 1986, Prime Computer, Inc.]
LTS Name: LTS1                          Address: 08-00-2F-F8-00-11

OPERATING SYSTEM STATISTICS
  Running Time: 01:23:51:49
  Percent CPU Usage  —  Idle: 86.5%
    LLC: 02.7     Net Mgmt: 00.8%
  Error Counts  —  Total: 0
    Correctable memory errors: 0        Watchdog timer expired: 0

NETWORK MANAGEMENT STATISTICS
  Alarms reported  —  Total: 0
  Logged: 0    Suppressed: 0    Dropped: 0
  OS: 0      LLC: 0    NTS: 0    Net Mgmt: 0
OK,
```

LOOPBACK Command

LOOPBACK is a PRIMOS command that allows network users to loopback packets between target components on the network and have the results reported back to the user. This test will be done during network operation (online) and is used to verify the software operation in two entities and the network path between them. The hardware components (i.e., cable, taps) in the network should be tested by the use of diagnostic equipment and monitors.

The basic scenario for a loopback test is an "echo" operation where a source entity sends a test packet to a destination entity which in turn echoes the packet back to the source. At the source, the sent and received packets are compared and the results reported to the user. A more common scenario adds a third entity which is the initiator of the loopback test. In this case, the initiator sends a loopback request to a source entity which then sends a test packet to a destination entity. The test packet is then echoed back to the source entity and the test packets will be compared. The results will then be sent back to the initiator. This scenario will commonly have the initiating entity being a network operator station from which validation is wanted on a communications path between two other end points (eg, an LTS300 and a host).

A predefined test message will always be used. As each packet is returned, its contents are checked against what was transmitted. Any alterations, other than addressing, are displayed to the user.

The LOOPBACK command can perform the following test scenarios:

1. Source host to remote LHC300. This is a test between a host and a remote network controller. This tests checks a hosts ability to transmit and receive properly over the network to a remote host controller.

2. Source host to self. This test causes a test message to be generated at the host, transmitted onto the network over one LHC300, and received by the same host over a different LHC300. The test checks a host's ability to transmit and receive properly over the network. The test must involve two active LHC300s on the same host because an LHC300 is unable to transmit and receive the entire test message at the same time.

3. Source host to remote LTS300. This test is similar to the previous test except it loops back at a remote LTS300.

4. Source host to remote host. This is a test between two host Network Management Servers on the network. This tests the complete path of going from one system in the network to another.

5. Source LHC300 to remote LHC300. This is a test between two host controllers in the network. This checks the end-to-end integrity up to the controller level on both sides.

6. Source LTS300 to LHC300. This is a test between a terminal concentrator and a host controller on the network.

7. Source LTS300 to host. This is a test between a terminal concentrator and a host Network Manager Server on the network. This test can be used to check a major portion of the path an LTS300 user will take when connecting to a host.

The current implementation provides a command/response exchange. The command uses a standard TTY/user interface.

Examples: The following examples show the use of various options for the LOOPBACK command. Notice how the destination and source points are identified in each example.

```
(Network user issues the following from HOST1:)

OK, LOOPBACK -DEST_NODE_NAME LTS1
[LOOPBACK Rev. 21.0 Copyright (c) 1986, Prime Computer, Inc.]

Source Node Name:            HOST1
Source LHC:                  02
Source Address:              08-00-2F-00-01-02
Source Loopback Layer:       NMSR

Destination Node Name:       LTS1
Destination Address:         08-00-2F-F8-00-11
Destination Loopback Layer: NME

Lan Name:                    LAN1

Test Results: COMPARISON ERROR!      Bytes Sent:   72  Bytes Received:   72
S: AAAA5555AAAA555501010202040408081010202040408080FEFEFDFDFBFBF7F7EFEFDFDF
R: AAAA5555AAAA555501010202040400001010202040408080FEFEFDFDFBFBF7F7EFEFDFDF
                                  ^ ^

(..continued)
S: BFBF7F7F00000101030307070F0F1F1F3F3F7F7FFFFFFEFEFCFCF8F8F0F0E0E0C0C08080
R: BFBF7F7F00000101030307070F0F1F1F3F3F7F7FFFFFFEFEFCFCF8F8F0F0E0E0C0C08080
```

(Network user issues the following from HOST1:)

```
OK, LOOPBACK -SRC_NODE_NAME LTS1 -DEST_NODE_NAME LTS2
[LOOPBACK Rev. 21.0 Copyright (c) 1986, Prime Computer, Inc.]

Source Node Name:           LTS1
Source Address:             08-00-2F-F8-00-11
Source Loopback Layer:      NME

Destination Node Name:      LTS2
Destination Address:        08-00-2F-F8-00-14
Destination Loopback Layer: NME

Lan Name:                   LAN1

Test Results: LENGTH ERROR!          Bytes Sent:   72  Bytes Received:  224
S: AAAA5555AAAA555501010202040408081010202040408080FEFEFDFDFBFBF7F7EFEFDFDF
R: AAAA5555AAAA555501010202040400001010202040408080FEFEFDFDFBFBF7F7EFEFDFDF


(..continued)
S: BFBF7F7F0000010103030707070F0F1F1F3F3F7F7FFFFFFEFEFCFCF8F8F0F0E0E0C0C08080
R: BFBF7F7F0000010103030707070F0F1F1F3F3F7F7FFFFFFEFEFCFCF8F8F0F0E0E0C0C08080


(..continued)
S:
R: 0A67F0000000287610000000000000000000003DBC1012C0000000000000000000000...
```

(Network user issues the following from HOST1:)

```
OK, LOOPBACK -SRC_NODE_NAME HOST1 -DEST_NODE_ADDRESS F8-0-11
[LOOPBACK Rev. 21.0 Copyright (c) 1986, Prime Computer, Inc.]

Source Node Name:           HOST1
Source Address:             08-00-2F-00-01-02
Source Loopback Layer:      NMSR

Destination Node Name:      LTS1
Destination Address:        08-00-2F-F8-00-11
Destination Loopback Layer: NME

Lan Name:                   LAN1

Test Results: PASS                    Bytes Sent:   72  Bytes Received:   72
S: AAAA5555AAAA555501010202040408081010202040408080FEFEFDFDFBFBF7F7EFEFDFDF
R: AAAA5555AAAA555501010202040408081010202040408080FEFEFDFDFBFBF7F7EFEFDFDF


(..continued)
S: BFBF7F7F00000101030307070F0F1F1F3F3F7F7FFFFFFEFEFCFCF8F8F0F0E0E0C0C08080
R: BFBF7F7F00000101030307070F0F1F1F3F3F7F7FFFFFFEFEFCFCF8F8F0F0E0E0C0C08080
```

Installation and Build Procedures

- Install the NETWORK_MGT directory on the pack.
- Create INFO top level directory on the pack.
- Run the NETWORK_MGT.INSTALL file on the pack.

PRIMENET

At Rev. 21.0, PRIMENET supports 255 VCs, X.25-1984, and the LAN300 network. This support includes configuring LAN300s (CONFIG_NET), starting and stopping networks containing LAN300s (START_NET and STOP_NET), monitoring traffic over them (MONITOR_NET), and using NETLINK over them.

X.25

At Rev. 21.0, X.25 supports 255 VCs, X.25-1984, and the LAN300 network. This support includes configuring LAN300s (CONFIG_NET), starting and stopping networks containing LAN300s (START_NET and STOP_NET), monitoring traffic over them (MONITOR_NET), and using NETLINK over them.

FILE TRANSFER SERVICE (FTS)

At Rev. 21.0, FTS has enhanced functionality in the following areas:

- Temporary destination files
- Transfer relative priorities
- CONVERT_DATABASE user requirements
- Logging mechanism
- Priority default
- Transfer defer time
- More configurable system parameters
- Remote revision checking

Temporary Destination Files

Destination files are now created as temporary files and renamed when the transfer has completed. In-use EPFs are mapped out at this time, as appropriate.

Transfer Relative Priorities

It is possible to specify a relative priority on the FTR command line, which allows a certain amount of scheduling. The default priority for each queue can be configured using FTGEN.

CONVERT_DATABASE user requirements

At Rev. 21.0, access requirements have changed for CONVERT_DATABASE from user SYSTEM to User 1 or the System Administrator.

Logging Mechanism

The log messages have changed for remote rev stamp compatilility to output to the master log, not the operator log.

Priority Default

The default priority has changed from 5 to 0 (off).

Transfer Defer Time

The time at which a transfer request is made available for processing by the server can be delayed by specifying a "defer time" on the FTR command line.

Configurable System Parameters

Several new subcommands have been added to the FTGEN server subsystem. This allows the Network Administrator to change the number of concurrent transfers and various retry limits.

Remote Revision Checking

Not all revisions of FTS support all facilities. Therefore, it is essential that the ISSUE field for each site configured with FTGEN is kept up-to-date. From Revision 21 Issue 27 onwards, FTS servers will communicate their respective issue numbers with each other, and a warning message will be output to the supervisor terminal if a discrepancy is found.

Refer to the Network Planning and Configuration Guide for further details.

Database Conversion Tool

The FTS.INSTALL.COMI file now contains a call to CONVERT_DATABASE which will convert the existing FTSQ* database to the new format, if required. This tool is described below.

Permanent Restrictions.

Versions 2.1 and later cannot transfer SEG files to version 2.0 systems. Version 2.0 systems can transfer SEG files to other version 2.0 systems.

Environment

- FTS Rev. 5.0 will run on all Rev. 20.2 and later PRIMOS revisions.
- FTS Rev. 5.0 can only be installed on 20.2 and later because it contains calls to KLM code.
- To build Rev. 5.0 FTS requires a Rev. 20.2 or later system.
- FTS does not use any segments beyond 4007 in the 4000 range.
- FTS uses the following allocated segments:

  2026 2027 2126 2127 6006

  Locations 0 to 37777.

- FTS uses the following PRIMENET port numbers:

  Ports 252 and 256 are utilised by the FTS manager phantom, YTSMAN, when processing incoming requests.

For every configured FTS server, a port number is required in the range 1 to 99.

Each server requires its own unique port number, which must not be used by any other process on the machine.

FTS Build Procedures

The command to build the complete FTS subsystem is:

```
RESUME FTSSRC>FTS.BUILD.CPL [-COMO]
```

The -COMO option will generate a COMOUTPUT file called FTS.BUILD.COMO.

Individual FTS modules may be built by the appropriate CPL files in the FTSSRC>CPL directory.

If there are requirements to do anything more complicated, then it is suggested that you refer to the CPL build file listings, because a number of other options exist (normally only be used in a development environment) that may be helpful.

FTS Installation Procedures

Having built FTS as indicated above, it must now be installed by following the instructions detailed below.

Note

FTSQ* is the runtime directory for FTS and not only contains the FTS database, but also acts as the spool directory for the FTS transfer requests.

Initial Installation

Use this section if FTS has never been installed on your system.

At this point, no FTSQ* directory exists at the MFD level. Execute the following command file:

```
COMINPUT FTS>FTS.INITINSTALL.COMI
```

This will create the FTSQ* directory at the MFD level and copy up the contents of the FTS>FTSQ* directory. The following also happens:

- The FTS utilities FTR, FTOP, FIGEN are copied to CMDNC0
- The shared segments FQ2026, FQ4000, FQ2126A, FQ2126B, SV2127, FR2126A, FR2126B, and the command file, FTS.SHARE.COMI, are copied to the SYSTEM directory.
- The files FT$SUB.INS.PL1 and FT$SUB.INS.FTN are copied to the SYSCOM directory.
- The customer-loadable library VFTSLB is copied to the LIB directory.

Note

FTS.INITINSTALL.COMI may need modifying to specify explicitly the particular command partition where FTSQ* is to reside.

If FTSQ* is an ACL protected directory, then you must ensure that YTSMAN and the File Transfer server(s) (e.g. FTS1) are granted ALL ACL access rights to the FTSQ* directory. Otherwise, the YTSMAN and File Transfer server phantoms will fail to start up.

In addition, the user SYSTEM needs ALL access rights to FTSQ* and the $REST access rights should be set to DALURW.

After running the install file, edit the file CMDNC0>PRIMOS.COMI (or C_PRMO) to include the following:

1) A command to share FTS

   CO SYSTEM>FTS.SHARE.COMI 10

   This command should be inserted after initialisation of the SPL run-time library, and before the following two commands.

2) A command to phantom the File Transfer Service manager

   FTOP -START_MNGR

3) A command to phantom the File Transfer Server

   FTOP -START_SRVR FTS1

Note

The FTOP commands should be inserted in PRIMOS.COMI (or C_PRMO) after the command that sets the system date and time.

From the supervisor terminal, run the following command file:

```
COMINPUT SYSTEM>FTS.SHARE.COMI
```

Now invoke FTGEN and configure the local FTS subsystem. Note that you must be logged in as SYSTEM in order to use FTGEN, and that a FTGEN INITIALIZE_FTS command must be performed before configuring the system.

After FTS has been installed on sites that have X.MAIL, you should create the file 'MAIL_ON.FTS' in the FTSQ* directory. This will enable users to send mail via FTS and to recieve source/destination notify messages via X.MAIL.

Following configuration, perform a cold start to test the edited PRIMOS.COMI (or C_PRMO) and to start up the YTSMAN and File Transfer server phantom(s).

A simple example configuration sequence is shown below:

```
OK, FTGEN
[FTGEN Rev. 5.0 Copyright (c) Prime Computer, Inc. 1985]
FTS STATUS
----------
Server directory is ftsq*.
System issue number is  27.
The FTS data base is invalid.  (status)
ftgen> initialize_fts          /* Initialize the new system.
FTS STATUS
----------
Server directory is ftsq*.
System issue number is  27.
Number of queues configured is  0.
Number of servers configured is  0.
Number of sites configured is  0.
ftgen> add_queue fts$1             /* Add a single queue.
queue: log -off
queue: maximum_requests 200
queue: file
Queue added.
ftgen> add_server fts1             /* Add a single server.
server: queue fts$1
server: log fts1.log
server: message_level detailed
server: port 2
server: file
```

```
Server added.
ftgen> add_site locl              /* Add the local site.
site: address locl+fts1
site: issue 27
site: queue fts$1
site: log -off
site: file
Site added.
ftgen> add_site rem1        /* Add a remote Rev. 5.0 (Issue 27) site.
site: address rem1+fts1
site: issue 27
site: queue fts$1
site: log rem1.log
site: message_level detailed
site: file
Site added.
ftgen> add_site rem2       /* Add a remote Rev. 2.0 (Issue 20) site.
site: address rem2+fts1
site: issue 20
site: queue fts$1
site: log rem2.log
site: message_level detailed
site: file
Site added.
ftgen> add_site rem3     /* Add a remote Rev. 1.1(Issue 17) site
site: address rem3+fts1
site: issue 17
site: queue fts$1
site: log rem3.log
site: message_level detailed
site: file
Site added.
ftgen> add_site rem4     /* Add a remote Rev. 1.0 (Issue 14) site.
site: address rem4+fts1
site: issue 14
site: queue fts$1
site: log rem4.log
site: message_level detailed
site: file
Site added.
ftgen> quit              /* OK that's it, let's quit.
OK,
```

Reinstalling Rev. 5.0 On Existing Rev. 2.x and Later Systems

If Issue 27 is installed onto an existing Issue 19 system, then a complete reconfiguration of FTS site servers, sites, and queues using FTGEN will be necessary (see below).

The recommended procedure is to allow any outstanding queued or transferring File Transfer requests to be completed before attempting to reinstall. The FTP server(s) should be closed down (FTOP -STOP_SRVR server_name). The FTS manager, YTSMAN, should be closed down (FTOP -STOP_MNGR).

It is a good idea to take a backup copy of the current FTSQ* directory and the FTS utilities in CMDNC0, SYSTEM, SYSCOM, and LIB. The previous version can then be easily reinstalled if necessary.

First, execute the following command:

```
CO FTS>FTS.INSTALL.COMI
```

This command file copies the contents of the FTS>FTSQ* directory into the FTSQ* directory, replacing matching files that already exist in the FTSQ* directory. Also,

- The FTS utilities FTR, FTOP, FTGEN are copied to CMDNC0 .
- The shared segments FQ2026, FQ4000, FQ2126A, FQ2126B, SV2127, FR2126A, FR2126B, and the command file, FTS.SHARE.COMI, are copied to the SYSTEM directory.
- The files FT$SUB.INS.PL1 and FT$SUB.INS.FTN are copied to the SYSCOM directory.
- The customer-loadable library VFTSLB is copied to the LIB directory.

  Any required database conversions are carried out by the CONVERT_DATABASE tool: see the section CONVERT_DATABASE, below.

If FTSQ* is an ACL protected directory, ensure that YTSMAN and the File Transfer server(s) (for example, FTS1) are granted ALL access rights to the FTSQ* directory.

In addition, the user SYSTEM needs ALL access rights to FTSQ* and the $REST access rights should be set to DALURW.

Ensure the file CMDNC0>PRIMOS.COMI (or C_PRMO) includes the following:

1) A command to share FTS

   ```
   CO SYSTEM>FTS.SHARE.COMI 10
   ```

   This command should be inserted after initialization of the SPL run-time library, and before the following two commands.

2) A command to phantom the File Transfer Service manager

   FTOP -START_MNGR

3) A command to phantom the File Transfer Server

   FTOP -START_SRVR FTS1

Note

The FTOP commands should be inserted in PRIMOS.COMI (or C_PRMO) after the command that sets the system date and time.

From the supervisor terminal, issue the following:

```
COMINPUT SYSTEM>FTS.SHARE.COMI
```

Now invoke FTGEN and issue the INITIALIZE_FTS command (abbreviation IFTS), and check the configuration.

Note

If Issue 27 is installed on an existing Issue 19 system, then a complete reconfiguration of FTS site servers, sites, and queues that use FTGEN will be necessary.

At this point, either perform a cold start to check out any PRIMOS.COMI (or C_PRMO) changes and start up both the new YTSMAN and File Transfer server phantoms(s), or start up the manager and server(s) using the above FTOP commands from the supervisor terminal.

Installation of Rev. 5.0 On Existing Rev. 1.x Sites.

The Rev. 5.0 system cannot be simply installed on top of a Rev. 1.x FTS system for the following reasons:

- Rev. 1.x requests cannot be processed by Rev. 5.0 FTS, and vice versa.
- Rev. 1.x configuration formats are incompatible with Rev. 5.0 and vice-versa.

Therefore, it is important to perform this install in the manner outlined below.

1) Use the FTGEN BLOCK_QUEUE command to prevent users from submitting further transfer requests while the install is in progress.

   Since Rev. 1.x requests will not be processed by Rev. 5.0, it is advisable to ensure all outstanding requests have been transferred and removed from the queue(s) before continuing with the install.

   Alternatively, make a COMOUTPUT record of the outstanding requests using a FTR -DISPLAY command while logged in as SYSTEM. This record can then be used to resubmit the requests following installation.

2) The FTP server(s) should be closed down (with FTOP -STOP_SRVR server_name).

3) The FTS manager, YTSMAN, should be closed down (FTOP -STOP_MNGR).

4) Make a COMOUTPUT record of the Rev. 1.x configuration using the FTGEN commands LS -ALL, LSITE -ALL, and LQ -ALL. This is essential, because you will need to re-configure the FTS configuration once the install is complete. This COMO file can also be edited to form a COMINPUT file to reduce the amount of retyping the configuration information and reduce the risk of errors.

5) Preserve the current Rev. 1.x software just in case of any problems. This includes the following:

   Copy the SYSTEM files FTS.SHARE.COMI, Q4000, and QP2026.

   Copy the CMDNC0 files FTR.SAVE, FTOP.SAVE, FTGEN.SAVE.

   CNAME the FTSQ* to FTSQ*_SAVE, for example.

6) Execute the following command:

   CO FTS>FTS.INITINSTALL.COMI

   This command file creates the FTSQ* directory at the MFD level and copies up the contents of the FTS>FTSQ* directory. Also, it

   - copies the FTS utilities FTR, FTOP, FTGEN to CMDNC0
   - copies the shared segments FQ2026, FQ4000, FQ2126A, FQ2126B, SV2127, FR2126A, FR2126B, and the command file, FTS.SHARE.COMI, to the SYSTEM directory
   - copies FT$SUB.INS.PL1 and FT$SUB.INS.FTN to SYSCOM
   - copies the library VFTSLB to the LIB directory

Note

FTS.INITINSTALL.COMI may need modifying to specify explicitly the particular command partition where FTSQ* is to reside.

7) If FTSQ* is an ACL protected directory, ensure that YTSMAN and the file transfer server(s) (for example, FTS1) are granted ALL ACL access rights to the FTSQ* directory. Otherwise, the YTSMAN and file transfer phantoms will fail to start up. In addition, the user SYSTEM needs ALL access rights to FTSQ* and the $REST access group should have DALURW rights.

   Ensure the file CMDNC0>PRIMOS.COMI (or C_PRMO) includes the following:

   - A command to share FTS for use

     COMINPUT SYSTEM>FTS.SHARE.COMI 10

     This command should be inserted after initialisation of the SPL run-time library, and before the following two commands.

   - A command to phantom the transport service manager

     FTOP -START_MNGR

   - A command to phantom the File Transfer server

     FTOP -START_SRVR FTS1

Note

The FTOP commands should be inserted in PRIMOS.COMI (or C_PRMO) after the command to set the system date and time.

8) From the supervisor terminal, execute the following command:

   COMINPUT SYSTEM>FTS.SHARE.COMI

9) Now invoke FTGEN and reconfigure the FTS configuration.

Note

Pay particular attention to the setting of the new site configuration parameter, as specified by the ISSUE site subcommand. You need to specify what issue (version) of FTS is running on the remote sites. The default value is ISSUE 27 (FTS Rev. 5.0). It is important that Rev. 1 sites are configured with the correct ISSUE number: otherwise, transfers to these sites may fail.

10) At this point, either perform a cold start to check out any PRIMOS.COMI changes and start up both the new YTSMAN and File Transfer server phantoms(s), or start up the manager and server(s) using the above FTOP commands from the supervisor terminal.

11) Finally, resubmit any outstanding Rev. 1.x requests using the COMOUTPUT record made prior to installation.

Recovering From A Corrupt FTS Database

The FTSQ* directory is the runtime directory for FTS. It contains the FTS configuration database, also acts as the spool directory for the FTS transfer requests. One of the functions performed by FTS is validation of the FTS database. While the database is valid, the file VALID_SUBSYTEM.FTS exists in the FTSQ* directory. Should invalid data be detected, the VALID_SUBSYTEM.FTS file is deleted, and the database is invalidated.

The likelihood of this event occurring is very small, but the following steps outline a recover strategy. Follow these steps in order. Only proceed to the next stage if the preceeding one fails.

- Invoke the FTGEN utility and try an INITIALIZE_FTS. Make sure the configuration is complete by using the FTGEN commands LSITE -ALL, LS -ALL, and LQ -ALL.

- Close down the file transfer server by using the FTOP utility, and then log out the YTSMAN phantom.

- Perform an initialization of the FTS subsystem with the following:

  RESUME FTSQ*>INIT -WORLD_RESET

  Followed by:

  RESUME FTSQ*>HINIT

Do an INITIALIZE_FTS in FTGEN to see if the database is now valid.

- The ultimate solution is to attach to the FTSQ* directory and delete all the files with a .FTS suffix.

  At the supervisor terminal, reshare FTS by typing the following commands:

  ```
  COMI SYSTEM>FTS.SHARE.COMI

  RESUME FTSQ*>INIT -RESET -SUBSYSTEM FTSQ*

  RESUME FTSQ*>HINIT
  ```

  Do an INITIALIZE_FTS.

  Re-configure the FTS configuration of queues, sites and servers using FTGEN.

CONVERT_DATABASE

CONVERT_DATABASE is an installation tool supplied at Rev. 21.0 and later. It performs any required manipulation of the FTS database in FTSQ*. CONVERT_DATABASE is designed to be self-recovering in the event of error. The -RESET option resets the database to its original state. Use -RESET when a machine failure occurs during conversion, or when reverting to the previous version of FTS.

CONVERT_DATABASE performs the following actions:

- The database is checked. If no conversion is required then CONVERT_DATABASE terminates. This means that the call to CONVERT_DATABASE can remain in the INSTALL.COMI file permanently, though usually it will not do anything.

- If conversion is required, the existing database is archived as follows:

  ```
  SITE_CONFIG.FTS renamed to SITE_CONFIG.SAVED

  QUEUE_CONFIG.FTS renamed to QUEUE_CONFIG.SAVED

  PROCESS_TABLE.FTS renamed to PROCESS_TABLE.SAVED

  queue_name renamed to queue_name.SAVED
  ```

  When archiving is complete, each file is converted in turn, producing a new set of files. If, at any stage, a failure occurs, all new files are deleted and the original ones are given back their original names. You may force this occurrence

by specifying -RESET on the command line. During the archiving and conversion process, the database status is set to INVALID. This ensures that if the conversion does not complete properly for any reason, then FTS will be unusable until the database is reset.

Note

Once the .SAVED files have been deleted, it is not possible to reset the database. Do not delete the .SAVED files until you are satisfied that the FTS installation is successful and will not need to be reverted.

CONVERT_DATABASE Error Conditions

The next sections document the error messages that may be output by CONVERT_DATABASE.

Messages output alone: The following messages are the sole output of the appropriate error conditions.

- Database converted OK

  The conversion has been successful

- You do not have operator privileges

  CONVERT_DATABASE must be run from the supervisor terminal.

- Database reset finished

  A forced reset (-RESET) has completed successfully.

- Database conversion failed

  The conversion attempt has been unsuccessful; the database will have been reset.

- You cannot quit during reset

  It is not possible to interrupt the reset process. This is a safety measure: if a reset is done accidentally, it is safer to let it finish and subsequently retry the conversion.

- QUIT! - Database will be reset

  You have interrupted the conversion process. The database will be reset to avoid possible discrepancies.

Messages Preceded By PRIMOS File System Messages: The following FTS messages are preceded by PRIMOS file system messages. (The symbol <*> indicates a filename or pathname.)

Opening <*>
Reading <*>
Closing <*>
Writing <*>
Attaching to <*>
Attaching to home
Renaming <*> to <*>
Unable to reset <*>

These messages indicate what was being attempted when the Primos error occurred. Likely causes are Insufficient Access Rights, Disk Full, Quota Exceeded, File Open, and Disk Offline. In all cases, the database will be reset unless a fatal PRIMOS error or hardware error occurs: in this case, the -RESET option must be used (see above).

Changes Made To Database Structure Since Rev. 21.0: The following message informs you of changes made to the database:

Rev21 - New fields added to Queue, Server and Request databases

Structure of FTS Directories

```
FTSSRC      /* The top level FILE TRANSFER SERVICE source directory.
               This directory contains all the source of the FTS plus
               the command files to Build and List the complete
               FTS system. */
 FTR        /* The FTR Interactive submittal program directory. */
  SOURCE    /* The directory that contains the FTR source. */
  BINARY    /* The directory that contains the FTR binary files. */
  LISTING   /* The directory that contains the FTR listing files. */
  INSERT    /* The directory that contains the FTR insert files. */
 FTGEN      /* The FTGEN utility directory. */
  SOURCE    /* The directory that contains the FTGEN source. */
  BINARY    /* The directory that contains the FTGEN binary files. */
  LISTING   /* The directory that contains the FTGEN listing files. */
  INSERT    /* The directory that contains the FTGEN insert files. */
 FTOP       /* The FTOP utility directory. */
  SOURCE    /* The directory that contains the FTOP source. */
  BINARY    /* The directory that contains the FTOP binary files. */
  LISTING   /* the directory that contains the FTOP listing files. */
  INSERT    /* The directory that contains the FTOP insert files. */
 FTP        /* The FTP server program directory. */
  SOURCE    /* The directory that contains the FTP source. */
  BINARY    /* The directory that contains the FTP binary files. */
  LISTING   /* the directory that contains the FTP listing files. */
  INSERT    /* The directory that contains the FTP insert files. */
 LIBRARY    /* The FTS LIBRARY directory (FTS common subroutines). */
  SOURCE    /* The directory that contains the LIBRARY source. */
  BINARY    /* The directory that contains
               the LIBRARY binary files. */
  LISTING   /* The directory that contains
               the LIBRARY listing files. */
  INSERT    /* The directory that contains
               the LIBRARY insert files. */
 SYSCOM     /* The directory containing the source of files which
               will be placed in SYSCOM top-level directory. */
 YTS        /* The directory containing the source of the
               Yellow book Transport Service. */
  SOURCE    /* The directory that contains the YTS source. */
  BINARY    /* The directory that contains the YTS binary files. */
  LISTING   /* The directory that contains the YTS listing files. */
  INSERT    /* The directory that contains the YTS insert files. */
 YTSMAN     /* The directory containing the source of the
               FTS Manger process, YTSMAN. */
  SOURCE    /* The directory that contains the YTSMAN source. */
  BINARY    /* The directory that contains the YTSMAN binary files. */
  LISTING   /* The directory that contains the YTSMAN listing files.*/
  INSERT    /* The directory that contains the YTSMAN insert files. */
 QPAKS      /* The directory that contains the source for the
               QPAKS queuing utility. */
  SOURCE    /* The directory that contains the QPAKS source. */
  INSERT    /* The directory that contains the QPAKS insert files. */
  UTILS     /* The directory that contains utility files for
               building QPAKS. */
```

```
  LIB         /* The directory that contains the QPAKS library
                 runfile. */
 HINIT        /* The directory containing the source of the
                 FTS Histogram initialisation program. */
  SOURCE      /* The directory that contains the HINIT source. */
  BINARY      /* The directory that contains the HINIT binary files. */
  LISTING     /* The directory that contains the HINIT listing files. */
  INSERT      /* The directory that contains the HINIT insert files. */
TOOLS         /* The directory containing the source of sundry tools */
  SOURCE      /* The directory that contains the TOOLS source. */
  BINARY      /* The directory that contains the TOOLS binary files. */
  LISTING     /* The directory that contains the TOOLS listing files. */
  INSERT      /* The directory that contains the TOOLS insert files. */

FTS           /* This directory contains all the FTS run files,
                 installation command files, a share command file,
                 and a skeleton of the FTSQ* directory. */
 CMDNC0       /* Contains the run files for the FTR,FTOP,
                 and FTGEN utilities. */
 FTSQ*        /* Contains the skeleton of the FTSQ* directory. */
 INFO         /* Contains FTS build and installation instructions
                 and details relating to the current release. */
 LIB          /* Contains the VFTSLB library which is copied up
                 to the LIB directory. */
 SYSCOM       /* The directory containing a copy of files which will
                 be placed in SYSCOM top-level directory. */
 SYSTEM       /* Contains the FTS shared segments which are copied
                 to the SYSTEM directory. */

FTSQ*         /* This directory contains the FTS configuration files
                 and all the FTS run-time files. */
 HELP.FTS     /* Contains all the HELP information
                 for the FTS system. */
  FTR         /* Contains the HELP information files
                 for the FTR utility. */
  FTOP        /* Contains the HELP information files
                 for the FTOP utility. */
  FTGEN       /* Contains the HELP information files
                 for the FTGEN utility. */

 COMO.FTS     /* Contains the Comoutput files associated
                 with File Transfer Servers. */
 LOCK.FTS     /* Contains the Lock files associated with
                 File Transfer Servers. */
 START.FTS    /* Contains files associated with the start up of
                 File Transfer Servers. */
```

CHAPTER 6

COMMAND LINE EDITOR

The PRIMOS command line editor, EDIT_CMD_LINE (abbreviated ECL), lets you edit, save, and redisplay PRIMOS command lines. You do not need to retype entire command lines if you enter wrong characters by mistake. With ECL you can

- Move the cursor to the beginning or end of the line, or move it a specified number of characters or words in either direction.
- Delete a specified number of characters or words, or delete everything to the right of the cursor.
- Edit and insert command line text.
- Locate and redisplay a previously executed command line or section of a command line, or all commands executed during the current login session (up to a maximum of 200).
- Save commands in a file for subsequent reexecution.
- Redisplay deleted command line text.
- Expand partially entered pathnames.
- Perform all of the above functions on the local system or on a remote system across a network.

You can perform most of these functions with just one or two keystrokes. The keystrokes required are very similar to those used within the EMACS editor, so you might already be familiar with many of

them. However, you do not have to know EMACS to use ECL.

The first part of this chapter shows you how to get started with ECL. The second part explains more advanced ECL features. A summary of ECL commands and descriptions of options you can include when invoking ECL appear at the end of the chapter.

GETTING STARTED

At the PRIMOS prompt, enter

```
ECL -ON
```

to begin using any of the ECL commands. (To disable ECL, enter ECL -OFF.) For convenience, you might want to add the ECL -ON command, with any other frequently used options, to your login file so that ECL is in effect as soon as you log in.

By default, ECL uses the PRIMOS ready and error prompts OK, and ER!. The Customizing Your ECL Prompts section explains how to specify different ECL prompts.

This section covers the basics you need to use ECL. The headings included are listed in order below.

- ECL Command Conventions
- Aborting ECL Commands
- Moving the Cursor on the Current Line
- Deleting Characters and Lines
- Refreshing the Current Command Line
- Redisplaying the Previous Command Line
- Searching for Previously Entered Text

ECL Command Conventions

All ECL commands begin with either the ESCAPE key or the CONTROL key. Therefore, whenever you press the ESCAPE key and another character or the CONTROL key and another character, ECL knows that you are giving it a command.

For instance, to move the cursor to the beginning of the command line, simultaneously press the CONTROL key and the A key. Such ECL commands

are shown as CONTROL-A. Do not type the hyphen between the keys.

With ESCAPE key commands, you must press and release the ESCAPE key before you press the character key. Because you do not hold the ESCAPE key down, these sequences are shown as ESCAPE character (such as ESCAPE R for reverse searching).

You can precede ECL commands by a numeric count that represents the number of times to repeat that command. To repeat an ECL command, press ESCAPE and then press the digit for the number of times you want to repeat the next command (ESCAPE 3 tells ECL to perform the next ECL command entered three times). Refer to the Repeating ECL Commands section for more information.

An easier way to repeat CONTROL key commands is to hold down both the CONTROL key and character key for several seconds, instead of pressing both quickly. This is similar to the difference between pressing the space bar and holding it down for several seconds; if you do the latter, the cursor moves to the right until you release the space bar. Since the ESCAPE key commands are separate keystrokes, you must use the methods described in the Repeating ECL Commands section to repeat those commands.

The position of the cursor determines which part of the line is affected by the ECL command entered. Some ECL commands affect text to the left of the cursor, some affect text to the right of the cursor, and some affect only the character over which the cursor is positioned. In sample command lines, the underscore character (_) represents the cursor position.

Aborting ECL Commands

Enter CONTROL-G to abort an ECL command not yet executed.

Moving the Cursor on the Current Line

The most basic ECL cursor commands move the cursor to the beginning or end of a line, and forward or backward by one or more characters. Table 6-1 shows the commands to perform these functions. Individual sections describing each command follow the table.

Table 6-1
Cursor-moving Commands

Position	ECL Command
Beginning of line	CONTROL-A
End of line	CONTROL-E
Back one character	CONTROL-B
Forward one character	CONTROL-F

Moving the Cursor to the Beginning of the Line: To move the cursor to the beginning of the line, press CONTROL-A. For example, suppose that you have just entered the command line below. The underscore character represents the cursor, shown at the end of the command line.

```
OK, *>STATUS_
```

What you really wanted to do was attach to the STATUS directory. Instead of using the PRIMOS erase or kill characters and then retyping the entire command line, use the ECL CONTROL-A command.

```
OK, *>STATUS_  [press CONTROL-A]
OK, _*>STATUS
```

CONTROL-A positions the cursor on the first character of the line, in this case, the asterisk. Now enter the PRIMOS ATTACH command and press the RETURN key to execute the command line.

```
OK, ATTACH _*>STATUS
```

Notice that the cursor remains positioned on the asterisk. When you enter characters on a line, any existing characters to the right of the cursor are shifted to the right. ECL automatically wraps as many as 158 characters of command line input.

Also note that when you press the RETURN key, PRIMOS receives and processes all characters on the command line, regardless of the cursor's position.

Moving the Cursor to the End of the Line: To move the cursor to the end of the line, press CONTROL-E. Suppose that in the previous example, you wanted to edit a file in the STATUS directory. The cursor is currently in the middle of the command line, as shown below.

```
OK, ATTACH *>STATUS  [press CONTROL-E]
OK, ATTACH *>STATUS_
```

Now you can enter an additional command, using a semicolon (;) to separate the two commands.

```
OK, ATTACH *>STATUS; ED JUN.30_
```

Unlike CONTROL-A, which positions the cursor on the first character of the command line, CONTROL-E positions the cursor after the last character on the line. Positioning the cursor on a blank space, rather than on the last character of a line, simplifies adding text to the end of the line.

Moving the Cursor Backward: To move the cursor backward on the command line without deleting text, use CONTROL-B. If you press CONTROL-B once, the cursor moves back one character. If you hold the two keys down instead of pressing them briefly, the cursor continues to move backward until you release the keys.

Take a look at the following command line.

```
OK, SLIT MANNERS_
```

To change SLIT to SLIST, press CONTROL-B until the cursor is positioned on the T. Then enter S, as shown below.

```
OK, SLIT MANNERS_  [hold down CONTROL-B]
OK, SLIT MANNERS  [enter S]
OK, SLIST MANNERS
```

Notice that the character entered appears to the left of the current cursor position.

Moving the Cursor Forward: Use CONTROL-F to move the cursor forward one or more characters. As with all CONTROL commands, if you hold the two keys down for several seconds, the cursor continues to move until you release them.

Deleting Characters and Lines

ECL observes erase and kill characters specified with the PRIMOS TERM command unless you indicate otherwise with the ECL -NO_OBEY_ERKL option. ECL also has its own commands that delete characters and lines, shown in Table 6-2.

Table 6-2
Delete Commands for Characters and Lines

Deletion	ECL Command
Character that cursor is on	CONTROL-D
Previous character	CONTROL-H (see below)
Rest of line (from cursor to end)	CONTROL-K
Entire line	CONTROL-U CONTROL-W (see below)

CONTROL-H works the same as the PRIMOS erase character (defined with the TERM -ERASE command) and the BACKSPACE key.

CONTROL-U CONTROL-W works the same as the PRIMOS kill character. See the Deleting Regions section for more information on CONTROL-W.

With ECL you also can delete words in either direction, as well as regions of text. These other delete commands are described in the Advanced ECL Commands section later in this chapter. The Redisplaying the Previous Command Line section includes an example of CONTROL-D.

Refreshing the Current Command Line

Pressing CONTROL-L redisplays, or refreshes, the current ECL command line. CONTROL-L is especially useful if you receive a message from a phantom or another user before you have completed an ECL function. When ECL is interrupted by a message, it performs the action that you requested but does not display that action unless you press CONTROL-L.

Redisplaying the Previous Command Line

Use CONTROL-Z to redisplay the most recently executed command line. CONTROL-Z is particularly useful if you misspell something on a command line just executed. The following example illustrates this point.

```
OK, COPY JAKE>CARD>ENGINE.PROB CARS>COMPLAINTS>JAKE.ENGINE
  [press RETURN]
```

The pathname of the file being copied is incorrect; the CARD directory should be the CARS directory. Consequently, PRIMOS displays an error message. The following steps show how to correct this.

1. Press CONTROL-Z to redisplay the command line, with the cursor positioned after the last character on the line.

   ```
   OK, COPY JAKE>CARD>ENGINE.PROB CARS>COMPLAINTS>JAKE.ENGINE_
   ```

2. Press CONTROL-B to move the cursor back to the D in CARD.

   ```
   OK, COPY JAKE>CARD>ENGINE.PROB CARS>COMPLAINTS>JAKE.ENGINE
   ```

3. Press CONTROL-D to delete the D.

   ```
   OK, COPY JAKE>CAR>ENGINE.PROB CARS>COMPLAINTS>JAKE.ENGINE
   ```

4. Enter S.

   ```
   OK, COPY JAKE>CARS>ENGINE.PROB CARS>COMPLAINTS>JAKE.ENGINE
   ```

5. Press the RETURN key.

ECL remembers the last 200 command lines executed. This list of commands is called the command history, and pressing CONTROL-Z in succession lets you individually review its contents. The Command History section describes other ways to use the list of commands that ECL remembers.

Searching for Previously Entered Text

You also can redisplay a command line by specifying a unique search string. ECL has two commands, ESCAPE S (forward search) and ESCAPE R or CONTROL-R (reverse search), that prompt for a character string. ESCAPE R searches backward through your command lines for the string entered, beginning with the most recently entered command line. ESCAPE S begins the search with the first command line in the command history.

After you enter the string, ECL performs a case-insensitive search in the specified direction for the most recently executed command line

containing that string. ECL searches the last 200 PRIMOS command lines entered until it finds the string specified. If the string is not there, the terminal beeps.

ECL maintains a search buffer containing the 10 most recently requested search strings. If you press the RETURN key at the search prompt instead of entering a string, ECL looks for the most recently entered search string.

The following example shows how to use the search command to find a previously entered pathname. When you press ESCAPE R on a command line, ECL displays its search prompt, shown below. Then enter the string that you want to find.

```
R-search: DINOS
```

ECL displays the last command line entered that contained the string DINOS, and you can reexecute the command, editing it first if necessary.

```
OK, EMACS GIFTS>FRIENDS>DINOS
```

Table 6-3 shows the ECL commands for searching through command line entries.

Table 6-3
Search Commands

Direction	ECL Command
Backward	ESCAPE R or CONTROL-R
Forward	ESCAPE S

ADVANCED ECL COMMANDS

This section describes more advanced functions that you can perform with ECL. The headings included in this section are listed in order below.

- Moving and Deleting Words
- Changing Case and Character Position
- Using Regions of Text

- Restoring Copied and Deleted Text
- Repeating ECL Commands
- The Command History
- Expanding Pathnames
- Referencing Other Directories
- Customizing Your ECL Prompts
- Using ECL Across a Network
- ECL Command Summary
- ECL Command Options

Moving and Deleting Words

ECL defines a word as a string of alphanumeric and underscore characters only; spaces and any other characters are word delimiters. Therefore, ECL considers the string

```
DATE; SLIST SQUID_RECIPE
```

to be three words long, while

```
DATE; SLIST SQUID.RECIPE
```

is four words long.

You can instruct ECL to consider all legal PRIMOS filename characters as valid characters rather than as word delimiters with the ECL -ENTRY option. Refer to the ECL Command Options section at the end of this description.

Table 6-4 lists the commands for cursor movement by words and for deletion of words.

Table 6-4
Moving the Cursor and Deleting by Words

Action	ECL Command
Move backward one word	ESCAPE B
Move forward one word	ESCAPE F
Delete next word	ESCAPE D
Delete previous word	ESCAPE CONTROL-H (see below)

You can also use ESCAPE BACKSPACE to delete the previous word.

Changing Case and Character Position

Some basic character modifications that ECL permits are

- Changing characters and words to uppercase
- Changing characters and words to lowercase
- Transposing two characters to the left of the cursor

ECL defines words as described in the previous section, Moving and Deleting Words.

Table 6-5 shows the ECL commands to perform uppercase and lowercase conversion, as well as character transposition.

Table 6-5
Case and Character Position Commands

Action	Command Sequence
Change lowercase character to uppercase or vice versa	CONTROL-^
Change word to all uppercase	ESCAPE U
Change word to all lowercase	ESCAPE L
Transpose two characters to left of cursor	CONTROL-T

Using Regions of Text

With ECL you can define a region of text that you can copy, delete, and reinsert into one or more command lines. A region is a block of text that you define to ECL by specifying its boundaries. This section describes defining, copying, and deleting regions. The next section describes how to restore copied and deleted regions.

Defining Regions of Text: Use CONTROL-@ to define one of the region's boundaries, called the mark. To define the other boundary, move the cursor to the place where you want the region to end. The region is the area of text between the mark and the new cursor position. Within ECL, the maximum size for a region is one PRIMOS command line (a maximum of 158 characters). By default, the mark is located at the first position to the right of the prompt.

The mark does not appear on your terminal screen; your command line looks identical regardless of whether or not you have defined a mark position. To check a region's current boundaries, use CONTROL-X CONTROL-X to swap the mark and cursor positions. CONTROL-X CONTROL-X does not change the boundaries of the region.

Copying Regions of Text: ESCAPE W copies a defined region and stores it in a buffer for subsequent retrieval. With ESCAPE W your command line does not change; the region that you define and copy is still displayed. However, you can redisplay and reexecute the commands in the copied region by using the yank command, explained in the Restoring Copied and Deleted Text section.

Deleting Regions: Use CONTROL-W to delete a defined region. If you use CONTROL-W with a repeat command of 2 or more (such as ESCAPE 2), the command deletes the entire line just like the PRIMOS kill character. The repeat command is described later in this section.

Table 6-6 below summarizes the ECL commands to define regions, check their boundaries, and copy and delete them.

Table 6-6
Region Commands

Action	ECL Command
Define one end of a region	CONTROL-@
Check region boundaries	CONTROL-X CONTROL-X
Copy a region into a buffer	ESCAPE W
Delete a region	CONTROL-W

Restoring Copied and Deleted Text

ECL maintains a 10-item buffer containing the most recent

- Word deletions
- Line deletions
- Region deletions
- Copied regions

You can redisplay the most recently deleted or copied text in this buffer with the ECL yank command, CONTROL-Y. CONTROL-Y inserts the most recent buffer entry to the right of the cursor. If you press CONTROL-Y several times in succession, the same entry is displayed — the last deletion or copied region. Specifying the repeat command (ESCAPE number) before CONTROL-Y redisplays the nth most recently deleted or copied text.

Note

You cannot restore character deletions performed with the PRIMOS erase character or with CONTROL-D because ECL does not add these deletions to the buffer.

CONTROL-Y displays the most recent buffer entry. To display the second most recent and previous entries, use ESCAPE Y. Unlike CONTROL-Y, each time you press ESCAPE Y ECL changes the pointer within the buffer to the previous entry and then displays that entry. With ESCAPE Y you can view from the second most recent to the tenth most recent buffer entries. If you continue pressing ESCAPE Y after reaching the last item in the buffer, you will review the same entries again in the same order because the buffer is circular.

Table 6-7 summarizes the two ECL commands that redisplay deleted or copied text.

Table 6-7
Commands to Redisplay Deleted and Copied Text

Action	ECL Command
Redisplay most recent buffer entry	CONTROL-Y
Redisplay from second to tenth most recent buffer entry	ESCAPE Y

Repeating ECL Commands

As mentioned earlier, the ECL repeat command lets you specify a number of times to repeat the next ECL command you enter. To use the repeat command

1. Press ESCAPE and the digit for the number of times you want the command executed. For example, ESCAPE 3.

2. Enter the ECL command you want repeated.

You must specify the number of times before entering the ECL command to be repeated. For example, if the cursor is in the middle of a command line and you want to delete the two characters to the right of the cursor, press ESCAPE 2 CONTROL-D. To delete 12 characters to the right of the cursor (including spaces), press ESCAPE 1 2 CONTROL-D.

The multiplier command, CONTROL-U, works similarly to the repeat command, ESCAPE number. To repeat an ECL command 12 times you can use either CONTROL-U 1 2 or ESCAPE 1 2. Unlike the repeat command, however, CONTROL-U used alone repeats the next command four times. For instance, pressing CONTROL-U CONTROL-D deletes the next four characters on the command line. When preceded with a repeat count, CONTROL-U multiplies that count by four. For example, if you press

CONTROL-U CONTROL-U CONTROL-F

ECL moves the cursor forward by 16 (4 x 4) characters. You can also use the repeat and multiplier commands together. For instance,

ESCAPE 1 2 CONTROL-U CONTROL-F

moves the cursor ahead 48 (12 x 4) characters. If you use these two commands together, you must enter the repeat command first.

You can also repeat the previous ECL command with CONTROL-C. To perform the previous ECL command more than once, enter the repeat or multiplier command followed by CONTROL-C. If the previous command also had a repeat count associated with it, ECL multiplies the two counts and performs the previous command that number of times.

ECL never considers CONTROL-C as the previous command. Therefore if you enter CONTROL-C twice, ECL reexecutes the command entered before the first CONTROL-C command.

Table 6-8 summarizes the commands to repeat ECL commands.

Table 6-8
Repeat and Multiplier Commands

Action	ECL Command
Repeat next ECL command [n] times	ESCAPE [n] or CONTROL-U [n]
Repeat next ECL command four times	CONTROL-U
Repeat next ECL command a multiple [n] of four times	ESCAPE [n] CONTROL-U
Repeat next ECL command 16 times	CONTROL-U CONTROL-U
Repeat previous ECL command	CONTROL-C

The Command History

In addition to a buffer of deleted and copied command line text, ECL also maintains a list of command lines executed. Every PRIMOS command line that you execute becomes part of your ECL command history. The command history is a list of up to 200 of the most recently executed command lines. You can save and restore command histories. You also can use them to display

- All command lines in the list
- A specified number of the most recently executed command lines
- A command line meeting specified search criteria
- A command line of the indicated line number
- The previous command line
- The next command line (if you are not positioned at the end of the command history)

Displaying the Command History: To display the current command history, use the repeat and refresh commands. Press ESCAPE 2 0 0 CONTROL-L. Specifying a repeat of 200 ensures that you will see up to the 200 most recently entered PRIMOS command lines. If your current history is smaller than that, ECL does not display the unused entries.

ECL automatically numbers each command line in the display. A sample command history is shown below.

```
001) DATE
002) A *>NEW.PRODUCTS; LD
003) EMACS TENNIS.BALL.RETRIEVER
```

You also can instruct ECL to display this number to the left of each command line prompt. Refer to the Customizing Your ECL Prompts section later in this chapter.

The command history does not contain deletions; ECL maintains a separate buffer for them. As with the buffer for deletions and copied regions, the command history is circular. ECL overwrites the oldest (least recent) commands when you have entered more than 200 commands as part of the same command history. In this case, command number 201 becomes command 001, 202 becomes 002, and so on, and your command history would contain the sequence shown below.

```
199)
200)
001)
002)
```

Moving Around in the Command History: Several ECL commands let you redisplay commands within the command history. CONTROL-Z, described eariler, lets you move back through the history one command line at a time. Another command, CONTROL-N, lets you move forward through the history again. You can also redisplay a specific command line with the goto line command, ESCAPE line-number ESCAPE G. For example, pressing

```
ESCAPE 2 ESCAPE G
```

redisplays the second command in the history, shown below.

```
002) A *>NEW.PRODUCTS;  LD
```

To move from this position to the end of the history, you could use CONTROL-N until you got there. An easier way to get to the end of the history is to use CONTROL-G (the ECL abort command) or the goto line command with a line number of zero (ESCAPE 0 ESCAPE G).

Table 6-9 lists ECL commands for displaying specific command lines of the command history, as well as for displaying the entire list.

Table 6-9
Command Line Display Commands

Command Line Display	ECL Command
All (entire list)	ESCAPE 2 0 0 CONTROL-L
Current (redisplay)	CONTROL-L
Line number specified	ESCAPE [line number] ESCAPE G
Next line	CONTROL-N
Previous line	CONTROL-Z
Previous number of lines	ESCAPE [number of lines] CONTROL-L

Staying Positioned Within the Command History: A pointer indicates your position within the command history. By default, after executing a previously entered command, the pointer is positioned at the end of the command history.

In the partial command history below, the pointer is at line 27, which is currently empty.

```
023) EMACS RECENT.SIGHTINGS
024) DATE
025) EMACS IFO.LOG
026) SPOOL IFO.LOG
```

To edit the RECENT.SIGHTINGS file again, you could issue the previous line command four times (CONTROL-U CONTROL-Z or ESCAPE 4 CONTROL-Z) to redisplay and reexecute line 23. After finishing the editing session, the command line below is displayed.

```
027) OK,
```

If you have a series of command lines that you want to reexecute, it is

more convenient if the pointer moves with you in the history, instead of going to the end of the list. The ECL -STICK option lets you do this. Using the above example, if you specify the -STICK option and reexecute line 23, the next line displayed is

```
024) OK,
```

Changing Commands Within the Command History: With the -STICK option, if you edit and reexecute a previously entered command, the edited version replaces the original version in the command history. To prevent this, use the ECL -STACK option. If you specify the -STACK option, ECL adds edited command lines to the end of the command history after you execute them, instead of overwriting the original command lines.

The Hidden Command: You must be careful when using the -STICK option because it does not let you see text on the current command line. For instance in the above example, line 24 actually contains the PRIMOS DATE command, although ECL does not automatically display it. A command line containing text that you cannot see is called a hidden command line. If you enter something else on a hidden command line, ECL overwrites the invisible, previously entered text with your new entry.

To prevent inadvertent overwriting of command lines, use the ECL -SHOW_HIDDEN option with the -STICK option. With the -SHOW_HIDDEN option, ECL displays the next command line, complete with text. Hidden commands occur only when you use the -STICK option; therefore, the -SHOW_HIDDEN option is useful only if you have also specified the -STICK option.

If the -SHOW_HIDDEN option were in force in the previous example, line 24 would look like

```
024) OK, DATE
```

After executing line 24, the command line below would appear.

```
025) OK, EMACS IFO.LOG
```

Note that when a hidden command is revealed in this way, the cursor appears at the beginning of the command line instead of at the end, where it appears for all other commands. A revealed hidden command stays on the command line only if you issue an ECL editing command, such as CONTROL-D to delete a character. If instead you begin entering text, the revealed command automatically disappears to make inserting new text easier.

By specifying both the -SHOW_HIDDEN and -STICK options, you can sequentially redisplay and reexecute a series of commands. The -NO_SHOW_HIDDEN and -NO_STICK options reset the default state, so that the pointer automatically moves to the end of the command history after each command.

Saving and Restoring the Command History: If you have a series of commonly used command lines, you can store them in a command history file. To file a command history, use the command

```
OK, ECL -SAVE_HISTORY filename
```

after executing the commands you want to store. The filename entered will contain the list of up to the last 200 command lines executed, as well as the 10 most recent deletions and copied regions, and the 10 most recently entered search strings.

Note

Do not enter the ECL -SAVE_HISTORY command at the beginning of a login session. If you do, the only commands in the file specified will be those that you entered prior to the ECL -SAVE_HISTORY command; commands entered after the ECL -SAVE_HISTORY command will not be placed in the file.

To reactivate a command history file, use

```
OK, ECL -RESTORE_HISTORY filename
```

The above command makes the command history stored in filename your current command history. The commands in the file replace any active command history you might have had. Therefore, consider restoring a command history file as soon as you login; that way new command lines are appended to the restored command history, and become part of your current list.

Note

You cannot display command history files with the PRIMOS SLIST command, nor can you print them. To restore and reexecute the command lines in files, you must use ECL commands. To view and print a command history without using ECL commands, use a command output file and specify the ECL -NO_CLEAN_COMO option.

Expanding Pathnames

ECL can perform automatic pathname completion for files and directories on your system. When you use the ECL expand wild command by pressing CONTROL-I or the TAB key, ECL tries to complete what you've typed with a file or directory name that matches your entry. The expand wild command works similarly to entering the PRIMOS LD command with a partial pathname, followed by wildcard characters (@@).

When you use the expand wild command, ECL displays one of the following.

- If two or more file system objects match the partial pathname entered, ECL completes what you've typed with the portion of the pathname that is common to all of them. Press CONTROL-I again to see the list of all files and directories containing the partial pathname.

- If only one file system object matches the partial pathname entered, ECL replaces the string entered with the name of that object. If this is a file, ECL redisplays the filename when you press CONTROL-I a second time. If the matching pathname expands to a directory name, ECL automatically adds a greater-than sign (>) to the end of it. In this case when you press CONTROL-I a second time, ECL tries to expand within this directory, which usually results in the display of the directory's contents.

- If no file system objects match the partial pathname entered, the terminal beeps and the command line remains the same.

When listing possible completions, ECL arranges the display in alphabetical order horizontally, as with the PRIMOS LD command; however, instead of listing directories separately, the expand wild command designates directories by placing a greater-than sign (>) at the end of directory names.

Note

If you are using pathname expansion to specify a directory name with PRIMOS commands such as ATTACH or COPY, you must remove the final > before executing the command. Otherwise you will receive an error message.

Here's an example of how the expand wild command works. In the command lines below, the underscore character (_) represents the cursor.

```
OK, SLIST REV_      [press CONTROL-I]
```

The previous command line changes to

```
OK, SLIST REVIEW._        [press CONTROL-I again]
```

The following is displayed.

```
  REVIEW.1      REVIEW.2      REVIEW.DRAFT      REVIEW.OLD>

OK, SLIST REVIEW._
```

Notice in the above example that ECL completes pathnames that start with the characters entered (this is the default), and produces a final listing similar to the display shown by

```
OK, LD REV@@
```

The ECL -WILD_TAIL option, in conjunction with the expand wild command, lets you complete pathnames that end with a particular character string. Its use is similar to entering the PRIMOS LD command with wildcards placed before a string of pathname characters (such as LD @@CPL) or in the middle of a string (such as LD REV@@.RUN). To use the -WILD_TAIL option, position the cursor exactly where you want the wildcarding to occur and then press CONTROL-I. The expand wild command with the -WILD_TAIL option in force produces the same display format as shown previously.

The following example illustrates use of the -WILD_TAIL option.

```
OK, LD <DISK1>USER1>DIRECTORY1>TXT [press CONTROL-I]
```

ECL then displays the files within DIRECTORY1 that end in TXT, shown below.

```
  FILE1.TXT      FILE2.TXT

OK, LD <DISK1>USER1>DIRECTORY1>TXT
```

You can use both the repeat and multiplier options with either version of the expand wild command; doing so skips the intermediate step of partial expansion. The previous example is altered to demonstrate this.

```
OK, REV_   [press ESCAPE 2 or CONTROL-U; then press CONTROL-I]

  REVIEW.1      REVIEW.2      REVIEW.DRAFT      REVIEW.OLD>

OK, REV_
```

Note that the final result is different only in that the PRIMOS command line at the bottom is the same as the one originally entered, not the partially expanded version produced by pressing CONTROL-I alone (without preceding it with the repeat or multiplier commands).

Referencing Other Directories

The previous section describes use of the expand wild command within your current attach point. The expand wild command is also useful in conjunction with ECL commands to refer to other directories. These referencing commands enable pathname completion in directories above and including your current directory; these commands do not change your current attach point.

The format of displays produced by referencing different directories is very similar to those produced for pathname completion within the current directory. The example below shows how to refer to a parent directory.

```
OK, EMACS *<P_   [press CONTROL-I]
```

The following display appears.

```
  PALS                PICNIC              PLANES>
  PLANS.BIG>          PLANS.SMALL>        POINTS
  PROMISES>           PROPERTY>           PROSPECTS

OK, EMACS *<P_
```

A very similar display would have appeared if you had attached to the directory above the current directory and entered

```
OK, LD P@@
```

The difference is that with CONTROL-I, you don't have to attach to the directory or enter its pathname to view its contents. Also, ECL redisplays the original command line at the bottom of the listing (in this case, EMACS *<P), so you can complete the rest of the filename that you want to edit.

In the previous example, ECL listed all file system objects beginning with the letter P that you could view from the parent directory. However, if you had entered

```
OK, EMACS *<PI_    [press CONTROL-I]
```

ECL would have changed the command line to

```
OK, EMACS *<PICNIC_
```

because the PICNIC file is the only possible expansion for a string of PI. Had you entered

```
OK, EMACS *<PE_ [press CONTROL-I]
```

the terminal would have beeped, because the parent directory does not contain any file system objects that begin with those letters.

To display the contents of a higher-level directory, enter *< CONTROL-I for the parent directory or *<< CONTROL-I for the grandparent directory. You can refer to even higher-level directories by including additional greater-than signs to the right of the asterisk.

Table 6-10 lists the referencing commands.

Table 6-10
Reference Commands

Reference Position	Reference Command
Parent directory	*<[string] CONTROL-I
Grandparent directory	*<<[string] CONTROL-I
Current directory	*> CONTROL-I
Absolute directory	<MFD>directory>[string] CONTROL-I or directory>[string] CONTROL-I

If you reference a directory without entering a partial pathname for completion within that directory, such as a pathname ending with >, this usually results in display of that directory's contents.

Customizing Your ECL Prompts

Similar to the PRIMOS RDY command, you can customize your ECL ready, error, and warning prompts with the ECL -READY_BRIEF, -ERROR_BRIEF, and -WARNING_BRIEF options. You also can include ECL line numbers to the left of the prompt specified by entering a pound sign (#) before your prompt. In addition, you might want to enter a colon, right parenthesis, or comma before your prompt. As when you customize PRIMOS prompts, you should put the entire string in single quotation marks. For example,

```
ECL -READY_BRIEF '#) YOU RANG? '
```

sets your ECL ready prompt to

```
001) YOU RANG?
```

Using ECL Across a Network

To use ECL across a network, enter the NETLINK -MODE REMOTE_ECHO command. You must open any command output (COMO) files after you are on the remote system for the ECL -CLEAN_COMO option (the default) to work. If you open a COMO file on the local system and then attach to a remote system, the file will contain all ECL dialogue. Refer to the ECL Command Options section for a description of all options.

ECL COMMAND SUMMARY

This section lists all of the ECL commands previously discussed, grouped alphabetically according to function. In addition to entering command sequences, you can issue most ECL commands by entering their command names. To use the names, press ESCAPE X, enter the command name at the ECL command prompt, and press the RETURN key. You can enter command names in either upper- or lowercase.

An asterisk (*) indicates functions that accept the repeat (ESCAPE number) and multiplier (CONTROL-U) commands.

Action	Command Sequence	Command Name (precede with ESCAPE X)
Abort Command		
Abort the most recent ECL command	CONTROL-G	abort_cmd
Case and Character Position Commands		
Change lowercase character to uppercase and vice versa*	CONTROL-^	toggle_case
Change word to all uppercase*	ESCAPE U	upcase_word
Change word to all lowercase*	ESCAPE L	downcase_word
Transpose two characters to left of cursor	CONTROL-T	twiddle
Command Line Display Commands		
Display command history	ESCAPE 2 0 0 CONTROL-L	
Display current line (redisplay)	CONTROL-L	refresh
Display line number specified	ESCAPE [line number] ESCAPE G	goto_line [line number]

Display next line*	CONTROL-N	next_line
Display previous line*	CONTROL-Z	prev_line
Display previous number of lines	ESCAPE [number of lines] CONTROL-L	
Explicity display hidden command	ESCAPE 0 CONTROL-N	

Copied and Deleted Text Redisplay Commands

Redisplay most recent buffer entry*	CONTROL-Y	yank
Redisplay from the second to the tenth most recent buffer entries*	ESCAPE Y	yank_replace

Cursor-moving Commands

Move back one character*	CONTROL-B	back_char
Move back one word*	ESCAPE B	back_word
Move to beginning of line	CONTROL-A	begin_line
Move to end of line	CONTROL-E	end_line
Move forward one character*	CONTROL-F	forward_char
Move forward one word*	ESCAPE F	forward_word

Delete Commands

Delete next character*	CONTROL-D	delete_char
Delete next* word	ESCAPE D	delete_word

Delete previous character*	CONTROL-H (see below)	rubout_char
Delete previous* word	ESCAPE CONTROL-H (see below)	rubout_word
Delete region* specified with mark command	CONTROL-W	kill_region
Delete rest of line (from cursor to end)	CONTROL-K	kill_line
Delete entire line	CONTROL-U CONTROL-W	

You can also use the BACKSPACE key to delete the previous character, and ESCAPE BACKSPACE to delete the previous word.

Pathname Expansion Commands

Complete rest of common pathname	CONTROL-I	expand_wild
Refer to parent directory	*<[string] CONTROL-I	
Refer to grandparent directory	*<<[string] CONTROL-I	
Refer to absolute directory	<MFD>directory>[string] CONTROL-I or directory>[string] CONTROL-I	

Region Commands

Check region boundaries	CONTROL-X CONTROL-X	exchange_mark
Copy region into a buffer	ESCAPE W	copy_region
Define one end of a region	CONTROL-@	mark

Repeat and Multiplier Commands

Repeat next ECL command [n] times	ESCAPE [n] or CONTROL-U [n]	
Repeat next ECL command four times	CONTROL-U	multiplier
Repeat next ECL command a multiple [n] of four times	ESCAPE [n] CONTROL-U	
Repeat next ECL command 16 times	CONTROL-U CONTROL-U	
Repeat previous ECL command*	CONTROL-C	reexecute

Search Commands

Search backward*	ESCAPE R	reverse_search
Search forward*	ESCAPE S	forward_search

ECL COMMAND OPTIONS

This section describes the options that you can issue with the PRIMOS ECL command. The description is arranged alphabetically. Some options you must specify, while others are default options. Default options and their counterparts are paired, with the default appearing first. Valid abbreviations are shown in braces.

Option	Meaning
-CLEAN_COMO {-CCOMO} -NO_CLEAN_COMO {-NCCOMO}	Controls whether ECL terminal output is placed in command output files. With the default, -CLEAN_COMO, all ECL commands and listings are ignored. If you are working across a network and are using a command output file, you must open the file on the remote system for the -CLEAN_COMO option to work properly.
-COMPONENT {-COMP} -ENTRY	Defines characters that ECL considers valid for words. The default, -COMPONENT, indicates that words can consist only of alphanumeric and underscore characters. -ENTRY adds the characters # $ & * / - . @ + ^ = (those valid for PRIMOS file system objects). The definition of word affects ECL commands such as ESCAPE F (forward word), ESCAPE B (back word), and ESCAPE D (delete word).
-ERROR_BRIEF [prompt] {-EB}	Sets the error prompt displayed within ECL to [prompt]. The Customizing Your ECL Prompts section shows an example of how to reset your prompts.
-HELP {-H}	Displays the valid ECL options.
-INITIALIZE {-INIT}	Reinitializes ECL to its default options and clears the command history as well as the search and the copy and delete buffers.
-NO_CASE_SEARCH {-NCASE} -CASE_SEARCH {-CASE}	Controls whether searches for strings entered will be case sensitive.
-NO_EDIT_COMI {-NECOMI} -EDIT_COMI {-ECOMI}	Controls whether ECL interprets and includes the contents of command input files in the command history.
-NO_SHOW_HIDDEN {-NSHOW} -SHOW_HIDDEN {-SHOW}	Controls whether or not ECL displays the hidden command. Refer to The Hidden Command section for more information.

-NO_STACK {-NSTACK} -STACK	Controls whether ECL considers the command history a ring (-NO_STACK) or a stack (-STACK). With a ring, modifications to command lines already executed replace the original commands in the command history. With a stack, each modification becomes a new command in the command history. Stacks minimize lost commands, but make executing a sequence of previous commands more difficult.
-NO_STICK {-NSTICK} -STICK	Controls whether redisplaying a previous command repositions the command history pointer to the end of the history or leaves the pointer at its current position, at the line number following the command just executed. With -STICK, you can execute a sequence of previous commands. Refer to the Changing the Command History Pointer section for more information.
-NO_WILD_TAIL {-NWT} -WILD_TAIL {-WT}	Controls whether automatic pathname completion performs as if wildcards are appended at the end of a string (-NO_WILD_TAIL) or before or in the middle of a string (-WILD_TAIL).
-OBEY_ERKL {-OBEK} -NO_OBEY_ERKL {-NOBEK}	Controls whether or not ECL observes the characters defined as your PRIMOS erase and kill characters (using the PRIMOS TERM command).
-OFF	Turns off the ECL environment and reverts to PRIMOS command line processing.
-ON	Invokes the ECL editor.
-READY_BRIEF [prompt] {-RB}	Sets the ready prompt within ECL to [prompt]. Refer to the Customizing Your ECL Prompts section for an example of how to do this.
-RESTORE_HISTORY filename {-RHIST}	Makes the command history contained in filename the current command history. Refer to the Saving and Restoring the Command History section for more information.
-ROW_MAJOR {-ROW} -COL_MAJOR {-COL}	Indicates whether the expand wild command displays alphabetical lists horizontally (-ROW_MAJOR) or vertically (-COL_MAJOR).
-SAVE_HISTORY filename {-SHIST}	Stores the current command history in filename. Refer to the Saving and Restoring the Command History section for more information.

-SILENT {-SI}	Tells ECL to display fatal and command line error messages only.
-WARNING_BRIEF [prompt] {-WB}	Sets the warning prompt within ECL to [prompt]. The Customizing Your Prompts section includes an example of how to reset your prompts.

APPENDIX A

REV. 21.0 PUBLICATIONS

This appendix lists all books that are integral to Master Disk Revision 21.0. For all non-Revision related books, see the Guide to Prime User Documents (DOC6138-5PA) or type HELP DOCUMENTS.

** PRIMOS ADMINISTRATION AND OPERATION **

	RELEASE		
BOOK	20.0	20.2	21.0
Software Release Doc.	DOC10001-2PA	DOC10001-3PA	DOC10001-4PA
System Administrator's Guide, Volume I: System Configuration *			DOC10131-1LA
System Administrator's Guide, Volume II: Communication Lines and Controllers *			DOC10132-1LA
System Administrator's Guide, Volume III: Access and Security *			DOC10133-1LA

* Replaces DOC5037-4LA, System Administrator's Guide.

** PRIMOS ADMINISTRATION AND OPERATION **

	RELEASE		
BOOK	20.0	20.2	21.0
System Administrator's Programmer's Companion	FDR3622-192		
Operator's Guide to System Commands			DOC9304-3LA
Operator's Guide to System Monitoring			DOC9299-3LA
Operator's Guide to File System Maint.			DOC9300-3LA
Operator's System Overview			DOC9298-2LA
Operator's Guide to the Batch Subsystem			DOC9302-3LA
Operator's Guide to the Spooler Subsystem			DOC9303-2LA
Operator's Master Index			DOC10110-3LA
Data Backup and Recovery Guide ***			DOC10129-1LA
Magnet User's Guide			DOC10156-1LA
System Operator's Programmer Companion	FDR7812-192		
Using Your 2755			DOC8552-3LA
PRIME 9955 Handbook	DOC8887-2LA	UPD8887-21A	UPD8887-22A
2455 Handbook (includes 2450)			DOC10086-2LA
PRIME 750/850 Handbook	DOC10063-1LA		UPD10063-11A UPD10063-12A

*** Replaces DOC9301-1LA, Operator's Guide to System Backups, and DOC5027-2LA, Mag Tape User's Guide.

** PRIMOS ADMINISTRATION AND OPERATION **

	RELEASE		
BOOK	20.0	20.2	21.0
PRIME 400 Handbook	DOC8799-1LA		UPD8799-11A UPD8799-12A
2755 Handbook			DOC8638-3LA
2550 Handbook	DOC8638-192L UPD8638-11A		
6350 Handbook		DOC10161-LA	UPD10161-11A
Using Your 2455			DOC10085-2LA
2250 Handbook	DOC10073-1LA		UPD10073-11A
Using PRIME 2250	DOC6516-191L		UPD6516-11A
PRIME 50 Series Technical Summary	DOC6904-191P		
DSM User's Guide			DOC10061-1LA

** PRIMOS ARCHITECTURE AND ASSEMBLY **

	RELEASE		
BOOK	20.0	20.2	21.0
System Architecture Reference Guide			DOC9473-2LA
Instruction Sets Guide			DOC9474-2LA
Assembly Language Programmers Guide			DOC3059-2LA

** PRIMOS USER AND PROGRAMMER **

	RELEASE		
BOOK	20.0	20.2	21.0
PRIMOS Commands Reference Guide			DOC3108-6LA
PRIMOS Commands Programmer's Companion	FDR3250-192		
PRIME User's Guide	DOC4130-4LA UPD4130-41A		UPD4130-42A
Introduction to PRIMOS		DOC10111-1XA	
CPL User's Guide			DOC4302-3LA
CPL Programmers Companion	FDR7811-193		
Security Features User's Guide **			DOC10130-1LA

** For C2 systems only.

Advanced Programmer's Guide, Volume 0: Intro. and Error Codes	DOC10066-1LA		
Advanced Programmer's Guide, Volume 1: BIND and EPF's	DOC10055-1LA		
Advanced Programmer's Guide, Volume 2: File System			DOC10056-2LA
Advanced Programmer's Guide, Volume 3: Command Environment	DOC10057-1LA		
Subroutines Reference Guide, Volume I			DOC10080-2LA
Subroutines Reference Guide, Volume II		DOC10081-1LA	UPD10081-11A

** PRIMOS USER AND PROGRAMMER **

	RELEASE		
BOOK	20.0	20.2	21.0
Subroutines Reference Guide, Volume III		DOC10082-1LA	UPD10082-11A
Subroutines Reference Guide, Volume IV		DOC10083-1LA	UPD10082-11A

** COMMUNICATIONS **

	RELEASE		
BOOK	20.0	20.2	21.0
DPTX Guide		DOC4035-193	UPD4035-3LA
PRIMENET Guide	DOC3710-193P UPD3710-32A	UPD3710-33A	
Network Planning and Administration Guide	DOC7532-2PA		
PRIMENET Planning and Configuration Guide			DOC7532-1LA
User's Guide to PRIME Network Services			DOC10115-1LA
Programmer's Guide to PRIME Networks			DOC10113-1LA
Operator's Guide to PRIME Networks			DOC10114-1LA
LTS300 Installation Guide			DOC11034-1LA
NTS User's Guide			DOC10117-1LA
NTS Guide to Planning and Configuration			DOC10159-1LA
Remote Job Entry Phase II Guide			DOC6053-4LA
PRIME / SNA Administrator's Guide			DOC8908-3LA

** COMMUNICATIONS **

	RELEASE		
BOOK	20.0	20.2	21.0
DPTX Guide		DOC4035-193	UPD4035-3LA
PRIME / SNA Operator's Guide			DOC8909-3LA
PRIME / SNA Interactive Terminal User's Guide	DOC8910-2LA		
PRIME / SNA Interactive Terminal User's Reference Card			IDR8910-2RA
WSI300 User's Guide			DOC10155-1LA

** DATA MANAGEMENT **

	RELEASE		
BOOK	20.0	20.2	21.0
DBMS Manipulation Lang. Reference Guide	DOC5308-190P UPD5308-11A		UPD5308-12A
DBMS Data Desc. Lang. Reference Guide	DOC5717-181P UPD5717-11A		
DBMS Administrator's Guide	DOC6292-192P UPD6292-11A UPD6292-12A	UPD6292-13A	
DBMS User's Guide	DOC6291-192P	UPD6291-11A	
DBMS Programmer's Comp.	DOC8645-1XA		
DBMS Master Index			DOC10164-2PA
DBMS/QUERY Report Generator Casebook	IDR5650-182P		
DISCOVER Reference Guide	DOC7798-192L UPD7798-11A	UPD7798-12A	UPD7798-13A
DISCOVER User's Guide	DOC7799-1PA UPD7799-11A		

** DATA MANAGEMENT **

	RELEASE		
BOOK	20.0	20.2	21.0
FORMS Prog. Guide	PDR3040-163P PTU2600-110		
FED User's Guide	DOC4940-191P		
MIDAS User's Guide	IDR4558-176P PTU2600-089		
MIDASPLUS Concepts	DOC9243-1PA		
MIDASPLUS User's Guide	DOC9244-1PA UPD9244-11A	UPD9244-12A	UPD9244-13A
MIDASPLUS Prog. Compan.	DOC10045-1XA		
PRIME ORACLE Administrator's Manual	SDP10226-001		
PRIME ORACLE User and Reference Manuals	SDP10226-001		
PRIME/POWER Guide	PDR3709-173P PTU2600-072 PTU2600-090		
PRIME/POWER Companion	FDR4034-000		
PRISAM User's Guide		DOC7999-3LA	UPD7999-31A
PRISAM Prog. Comp.			DOC10088-1XA
ROAM Administrator's Guide			DOC7345-3LA

** LANGUAGES **

	RELEASE		
BOOK	20.0	20.2	21.0
Interpretive BASIC Programmer's Guide	IDR1813-140P		
BASIC/VM Programmer's Guide	COR3058-002 UPD3058-33A	UPD3058-34A	

** LANGUAGES **

	RELEASE		
Basic Programmer's Companion	FDR3341-000		
C User's Guide			DOC7534-3LA
C Programmer's Comp.	FDR8419-193		
COBOL 74 Ref. Guide	DOC5039-2LA	UPD5039-21A	UPD5039-22A
COBOL to CBL Conversion Guide	MAN10002-1LA		
FORTRAN Reference Guide	COR3057-002 UPD3057-33A FDR3057-101		UPD3057-34A
FORTRAN Prog. Comp	FDR3338-000		
FORTRAN 77 Reference Guide	DOC4029-4LA	UPD4029-41A	UPD4029-42A
FORTRAN 77 Programmers Companion	FDR4030-000		
Pascal Ref. Guide		DOC4303-3LA	UPD4303-31A
Pascal Prog. Comp.	FDR7095-192		
PL/I Reference Guide	DOC5041-1LA		UPD5041-11A
PL/I Subset G Ref.Guide	PTU2600-084 IRD4031 UPD4031-13A		
BP99 User's Guide	MAN10042-1LA MAN10042-2LA MAN10042-3LA		
RPG II V-mode Compiler Reference Guide	DOC5040-2LA		UPD5040-21A
PRIME RPG II Debugging Template	IDR11001-2XA		

** UTILITIES **

	RELEASE		
BOOK	20.0	20.2	21.0
Source Level Debugger User's Guide	DOC4033-192L UPD4033-21A	UPD4033-22A	
Loading and Debugging Programmer's Companion	FDR5094-000		
Source Level Debugging Programmer's Companion	DOC8916-1PA		
EMACS Primer	IDR6107-183P		
EMACS Reference Guide			DOC5026-2LA
EMACS Extension Writing Guide	DOC5025-2LA		
EMACS Standard User Interface Guide	DOC7446-2LA		
EMACS Reference Card			IDR5026-1RA
Loading and Debugging Programmer's Companion	FDR5094-000		
User's Guide to Editor/ Runoff	DOC3104-4LA		
Programmer's Guide to Bind/EPF's	DOC8691-1LA		
Load/Seg Reference Guide	DOC3524-192L		

** UTILITIES **

	RELEASE		
BOOK	20.0	20.2	21.0
Source Level Debugger User's Guide	DOC4033-192L UPD4033-21A	UPD4033-22A	
Loading and Debugging Programmer's Companion	FDR5094-000		
Source Level Debugging Programmer's Companion	DOC8916-1PA		
EMACS Primer	IDR6107-183P		
EMACS Reference Guide			DOC5026-2LA
EMACS Extension Writing Guide	DOC5025-2LA		
EMACS Standard User Interface Guide	DOC7446-2LA		
EMACS Reference Card			IDR5026-1RA
Loading and Debugging Programmer's Companion	FDR5094-000		
User's Guide to Editor/ Runoff	DOC3104-4LA		
Programmer's Guide to Bind/EPF's	DOC8691-1LA		
Load/Seg Reference Guide	DOC3524-192L		

DOC10001-4PA