900 Series HP 3000 Computer Systems

Migration Toolset Guide

HEWLETT PACKARD

Edition 1 E0787

30367-90008
Printed in U.S.A. 7/87

Printing History

New editions are complete revisions of the manual. Update packages, which are issued between editions, contain additional and replacement pages to be merged into the manual by the customer. The dates on the title page change only when a new edition or a new update is published. No information is incorporated into a reprinting unless it appears as a prior update; the edition does not change when an update is incorporated.

The software code printed alongside the date indicates the version level of the software product at the time the manual edition or update was issued. Many product updates and fixes do not require manual changes, and conversely, manual corrections may be done without accompanying product changes. Therefore, do not expect a one-to-one correspondence between product updates and manual updates.

First Edition . . . . . .July 1987

HP Computer Museum
www.hpmuseum.net

For research and education purposes only.

List of Effective Pages

The List of Effective Pages gives the date of the most recent version of each page of the manual. To verify that your manual contains the most current information, check the dates printed at the bottom of each page with those listed below. The date on the bottom of each page reflects the edition or subsequent update in which that page was printed.

Effective Page	Date
All	July 1987

Preface

The *Migration Toolset Guide* is written for programmers and system administrators that are using the Migration Toolset in preliminary planning for migrating applications or operating environment from an MPE V/E-based HP 3000 to an MPE XL-based HP 3000. This manual describes the activities of the migration process arranged in five stages. While you will only be able to accomplish tasks in the early stages of migration, it is recommended that you read the entire manual to get an overview of the migration process and effectively plan your migration effort.

Organization of this Manual

CHAPTER 1 contains a migration overview, listing the six stages of migration.

CHAPTER 2 explains the second stage of migration, Planning and Analysis.

CHAPTER 3 explains the third stage of migration, Preparation.

CHAPTER 4 explains the fourth stage of migration, Installation.

CHAPTER 5 explains the fifth stage of migration, Compatibility Mode.

CHAPTER 6 explains the sixth stage of migration, Native Mode.

The appendices contain step-by-step procedures for performing migration, additional information to assist the migration team in completing the migration stages, and details on how to use the Migration Toolset (the Object Code Analyzer and Run Time Monitor).

APPENDIX A describes how to maintain backward compatibility for programs and files, so that programs developed on an MPE XL-based system may continue to run on an MPE V/E-based system.

APPENDIX B describes how to use the Object Code Analyzer.

APPENDIX C describes how to use the Run Time Monitor.

APPENDIX D lists all known incompatibilities between MPE V/E and MPE XL.

APPENDIX E contains notes on migrating VPLUS.

APPENDIX F lists application tools for the 900 Series HP 3000 computer system.

How to Use this Manual

If this is your first introduction to the migration process, you should read the entire manual. If you are using the Migration Toolset before you have received your 900 Series HP 3000 computer system, you should read the entire manual, but only do activities listed in Chapters 1, 2, 3, and Appendices A, B, C, and D. If you are interested only in information on migrating the operating environment, you should only do activities listed in Chapters 3, 4 and Appendix F.

CAUTION

This manual contains only a preliminary set of incompatibilities. At this time, the content of this manual should be used only as an introduction to the migration process. To complete the migration from MPE V/E to MPE XL you will need other manuals that are not available at this time. HP has provided this manual and the Migration Toolset so that you can get a start on the migration process, and get a feel for the compatibility of your programs and applications that will be migrated to the MPE XL-based systems. HP recommends that you not finalize your migration plans and decisions based on the content of this manual as incompatibilities and strategies presented are subject to change and modification. HP has also developed FastLane 3000, this is a combination of training and consulting to ensure a smooth and successful migration.

If you have questions about the use of the Migration tools described in this manual, you should contact your support representative. If you want further assistance on migration planning and with potential incompatibilities, Hewlett-Packard recommends that you purchase FastLane 3000.

Table of Contents

Chapter 1
Introduction and Migration View

Chapter 2
Analysis and Planning

Table of Contents (Continued)

Table of Contents (Continued)

Table of Contents (Continued)

Table of Contents (Continued)

Appendix E
Notes on Migrating VPLUS

Appendix F
Tools for Application Development

Figures

Tables

Conventions

NOTATION	DESCRIPTION
UPPERCASE	Within syntax statements, characters in uppercase must be entered in exactly the order shown, though you can enter them in either uppercase or lowercase. For example: `SHOWJOB` Valid entries: `showjob` `ShowJob` `SHOWJOB` Invalid entries: `shojwob` `Shojob` `SHOW_JOB`
italics	Within syntax statements, a word in italics represents a formal parameter or argument that you must replace with an actual value. In the following example, you must replace *filename* with the name of the file you want to release: `RELEASE` *filename*
punctuation	Within syntax statements, punctuation characters (other than brackets, braces, vertical parallel lines, and ellipses) must be entered exactly as shown.

{ }	Within syntax statements, braces enclose required elements. When several elements within braces are stacked, you must select one. In the following example, you must select ON or OFF:
	
	`SETMSG {ON }`
	`{OFF}`

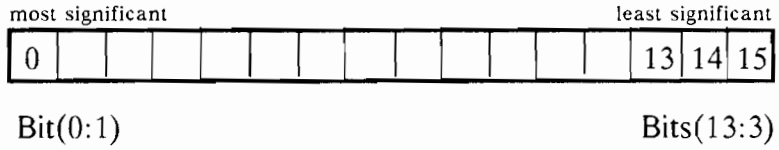
[ ]	Within syntax statements, brackets enclose optional elements. In the following example, brackets around ,TEMP indicate that the parameter and its delimiter are optional:
	
	`PURGE` *filename*[,TEMP]
	
	When several elements within brackets are stacked, you can select any one of the elements or none. In the following example, you can select *devicename* or *deviceclass* or neither:
	
	`SHOWDEV [`*devicename*`]`
	`[ `*deviceclass*`]`

Conventions (Continued)

NOTATION	DESCRIPTION
[...]	Within syntax statements, a horizontal ellipsis enclosed in brackets indicates that you can repeatedly select elements that appear within the immediately preceding pair of brackets or braces. In the following example, you can select *itemname* and its delimiter zero or more times. Each instance of *itemname* must be preceded by a comma: [,*itemname*][...] If a punctuation character precedes the ellipsis, you must use that character as a delimiter to separate repeated elements. However, if you select only one element, the delimiter is not required. In the following example, the comma cannot precede the first instance of *itemname*: [*itemname*][,...]
|...|	Within syntax statements, a horizontal ellipsis enclosed in parallel vertical lines indicates that you can select more than one element that appears within the immediately preceding pair of brackets or braces. However, each element can be selected only one time. In the following example, you must select ,A or ,B or ,A,B or ,B,A : {,A } {,B }|...| If a punctuation character precedes the ellipsis, you must use that character as a delimiter to separate repeated elements. However, if you select only one element, the delimiter is not required. In the following example, you must select A or B or A,B or B,A . The first element cannot be preceded by a comma: {A } {B }|,...|
... ⋮	Within examples, horizontal or vertical ellipses indicate where portions of the example are omitted.
△	Within syntax statements, the space symbol △ shows a required blank. In the following example, you must separate *modifier* and *variable* with a blank: SET[(*modifier*)]△(*variable*);

Conventions (Continued)

NOTATION	DESCRIPTION
[ ]	The symbol [ ] indicates a key on the terminal's keyboard. For example, [CTRL] indicates the Control key.
[CTRL] *char*	[CTRL] *char* indicates a control character. For example, [CTRL] Y means you have to simultaneously press the Control key and the Y key on the keyboard.
base prefixes	The prefixes %, #, and $ specify the numerical base of the value that follows: *%num* specifies an octal number *#num* specifies a decimal number $*num* specifies a hexadecimal number When no base is specified, decimal is assumed.
Bit (*bit:length*)	When a parameter contains more than one piece of data within its bit field, the different data fields are described in the format Bit (*bit:length*), where *bit* is the first bit in the field and *length* is the number of consecutive bits in the field. For example, Bits (13:3) indicates bits 13, 14, and 15:



1

Introduction and Migration Overview

Manual Structure and Purpose

The 900 Series HP 3000 computer family is based on a new architecture and features a new operating system (MPE XL). While MPE XL-based systems are highly compatible with MPE V/E-based systems, some important differences exist between the two. Therefore, applications software and the operating environment from an MPE V/E-based system must be "migrated" to an MPE XL-based system. This manual defines a process for completing this migration. Details are described from an operating system perspective, with references to documents describing migration concerns specific to databases, languages, networks, other products, and subsystems. Included are appendices on maintaining backward compatibility, the Migration Toolset (Object Code Analyzer and Run Time Monitor), Directory Migration Tool, FOS incompatibilities, and VPLUS migration specifics.

This chapter introduces the migration process. It includes:

- "System Overview", an introduction to the 900 Series Hewlett-Packard 3000 hardware and software.
- "Migration Options and Solutions", a discussion of the migration options and solutions developed by Hewlett-Packard to make your migration easier.
- "Migration Stages", a summary of the stages of migration defined by Hewlett-Packard to organize migration.
- "Migration Tools", an overview of the Hewlett-Packard tools available to assist in migration.
- "Documentation and Training", a list of Hewlett-Packard documentation and training that offer information on migration.
- "Migration Project Team", a definition of the project team you should form to complete migration.

System Overview

Before describing the options and issues involved in migrating to a 900 Series HP 3000, it is important to understand the hardware and software components of the system. The 900 Series HP 3000 family is based on HP Precision Architecture, an

architecture founded on the concepts of RISC (Reduced Instruction Set Computer). The operating system for the 900 Series family is MPE XL (MPE with extended large addressing).

HP Precision Architecture

HP Precision Architecture combines RISC principles with additional features to provide a flexible, expandable architecture and to further increase performance. These features include extended addressing (up to 64-bit virtual addressing), support for co-processors and multi-processors, and a memory-mapped I/O system.

RISC instructions are implemented directly in hardware to eliminate the system overhead associated with the microcode of conventional systems. Pipelining is enhanced by the uniformity of the HP Precision Architecture instructions. Improved performance also results from the memory hierarchy design of the new architecture. For a more detailed discussion of HP Precision Architecture, refer to the *General Information Manual* (5954-7418).

MPE XL

The improvements in hardware allow improvements to the operating system. MPE XL, the operating system which runs on the 900 Series HP 3000 is a superset of MPE V/E, the operating system which runs on the Series 37 through Series 70 HP 3000 computers. Along with enhanced functionality and ease of use, it provides greater performance and capacity through a virtually addressed file system and higher availability of data by featuring on-line concurrent backup and terminal configuration. Although MPE XL offers users a superset of MPE V/E features, some incompatibilities exist due to fundamental differences between MPE V/E and MPE XL systems software and hardware.

MPE XL provides two environments or modes for program execution: Compatibility Mode (CM) and Native Mode (NM). An application or program can execute in Compatibility Mode or Native Mode with no noticeable differences to the user. Additionally, program execution is not restricted to operating in Compatibility Mode or Native Mode only; a program operating in one mode can call procedures that operate in the other. This combination of Compatibility Mode and Native Mode execution is referred to as mixed-mode operation.

Compatibility Mode

Compatibility Mode (CM) is the emulation of MPE V/E-based machine instructions on a 900 Series HP 3000. This operating mode provides object code compatibility between MPE V/E-based systems and MPE XL-based HP Precision Architecture machines, allowing programs running on MPE V/E-based systems to run on 900 Series HP 3000s.

Compatibility Mode is implemented as part of the MPE XL operating system using an emulator which functions as an object code interpreter. The Emulator translates at run-time each MPE V/E machine instruction into a functionally equivalent sequence of HP Precision Architecture machine instructions. MPE V/E programs restored onto MPE XL-based systems will run transparently under the Emulator.

The MPE XL environment also supports the translation of MPE V/E machine instructions through the Object Code Translator (OCT). The Object Code Translator is an MPE XL system utility which accepts as input an MPE V/E program, performs an optimized translation of the MPE V/E compatible object code into HP Precision Architecture machine instructions, and appends the output of the translation to the end of the MPE V/E program. Appending the translated object code eliminates the need for fetching and decoding instructions each time the program is run, and the resulting program file can be executed on both MPE V/E- and MPE XL-based systems. Object Code Translation results in better performance since overhead during execution is significantly reduced. It is important to note that the translated program will require more disc space than the original program file.

For either emulated or translated execution, MPE XL preserves MPE V/E program and data structures and allows access to most MPE V/E callable intrinsics. The application is still subject to the same addressing limitations found on the MPE V/E-based HP 3000 computers.

Native Mode

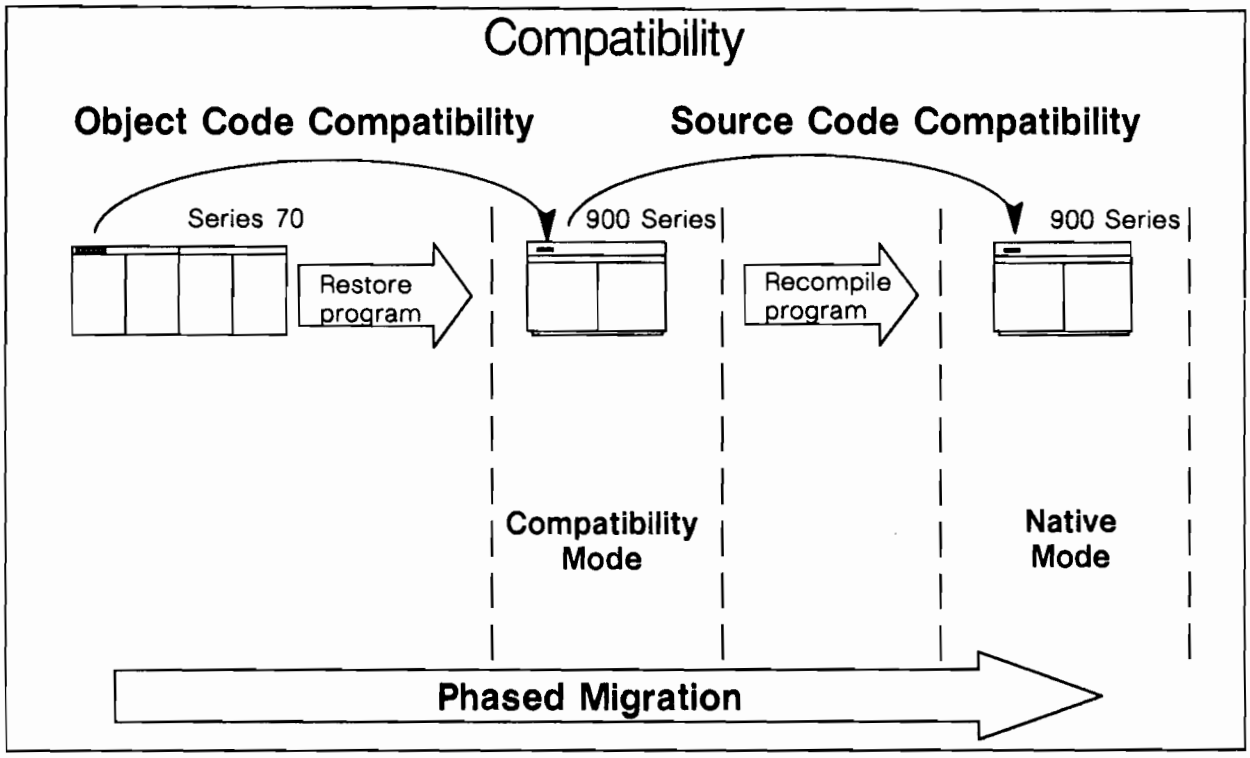
Native Mode (NM) is the natural operating mode of the 900 Series HP 3000. Native Mode allows users to utilize the features the 900 Series HP 3000 offers, including higher performance and the expanded addressing capabilities of HP Precision Architecture. Programs developed and compiled with MPE XL Native Mode compilers automatically run in Native Mode. Programs developed on MPE V/E-based systems need to be recompiled with the MPE XL version of compilers in order to run in Native Mode. The ability to recompile MPE V/E applications with MPE XL compilers provides source code compatibility between the two systems.

Mixed Mode

Applications that operate in both Compatibility Mode and Native Mode are called mixed-mode applications. The ability to mix modes of execution is provided by MPE XL through the Switch subsystem. The MPE XL Switch Subsystem provides the ability for programs executing in Native Mode to call procedures which reside in Compatibility Mode Segmented Libraries (SLs). The Switch Subsystem also provides the ability for Compatibility Mode programs to call procedures located in Native Mode Executable Libraries (XLs). The Switch subsystem resolves the differences between Native Mode and Compatibility Mode execution through a set of intrinsics that provide mixed-mode execution access. The ability provided by MPE XL to mix modes of execution allows applications to be migrated to Native Mode in phases.

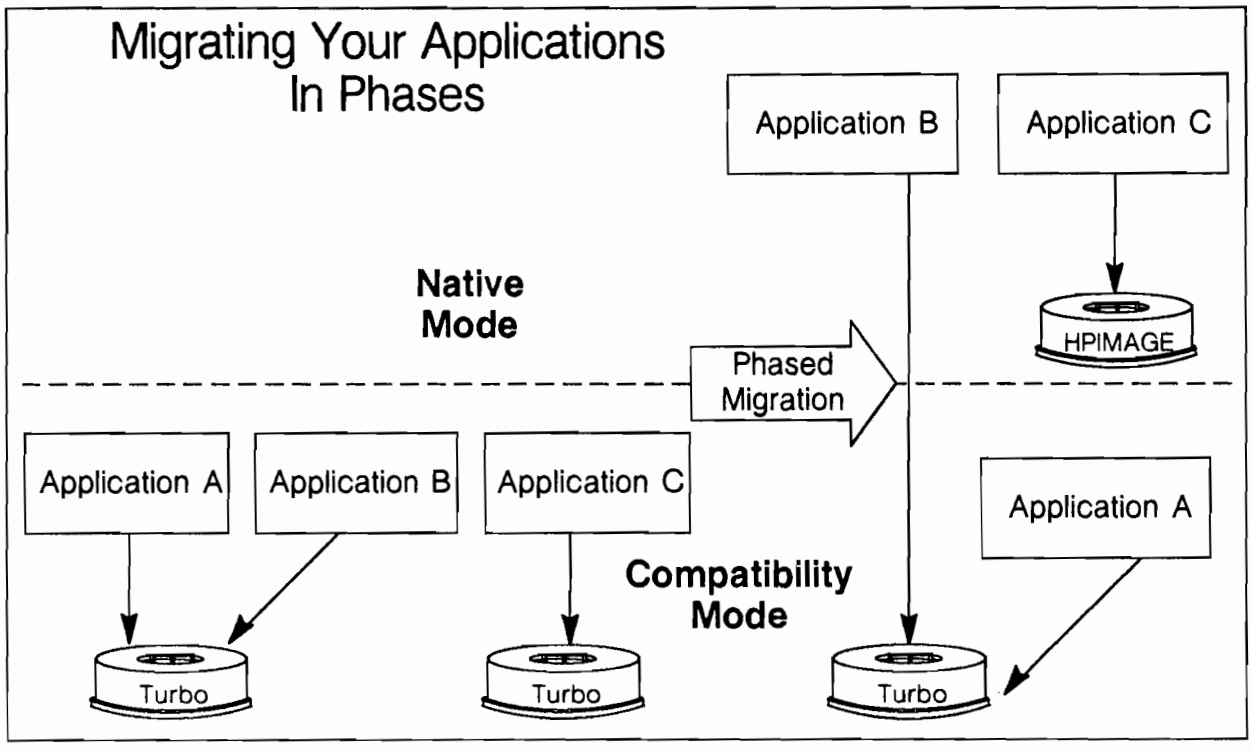
Phased Migration

Phased migration is the ability to move programs and data to Native Mode, over a period of time. This results in immediate productivity on the 900 Series HP 3000 family. The object code and source code compatibility and mixed mode operations offered by MPE XL allow you to migrate code and data independently as shown in Figure 1-1 and Figure 1-2.



LG200045-001

Figure 1-1. Phased Migration



LG200045-002

Figure 1-2. Migrating in Phases

Factors such as application size, availability of source code, and language implementation may prevent full recompilation to Native Mode. Partial recompilation may lead to mixing modes of execution (CM and NM) until the entire application is migrated to Native Mode. Thus, the ability to mix modes of execution provided by the Switch subsystem is an important part of phased migration.

Migration Options and Solutions

Since an application can consist of programs (composed of object code and possible library procedures), data, job streams, and UDCs, Hewlett-Packard has defined various migration options and solutions to help you solve potential application migration problems. Migration options address database applications. Migration solutions address other aspects which may affect application migration.

Migration Options

MPE XL supports two database management systems: TurboIMAGE/XL in Native Mode and ALLBASE/XL in Native Mode. ALLBASE/XL is a database management system that provides a relational database model called HP SQL.

The migration options for database applications allow a migration project team great flexibility in planning and executing an application migration. The following two application migration options were developed for database applications:

- Option 1. A Compatibility Mode program accessing a TurboIMAGE database.
- Option 2. A Native Mode program accessing a TurboIMAGE database.

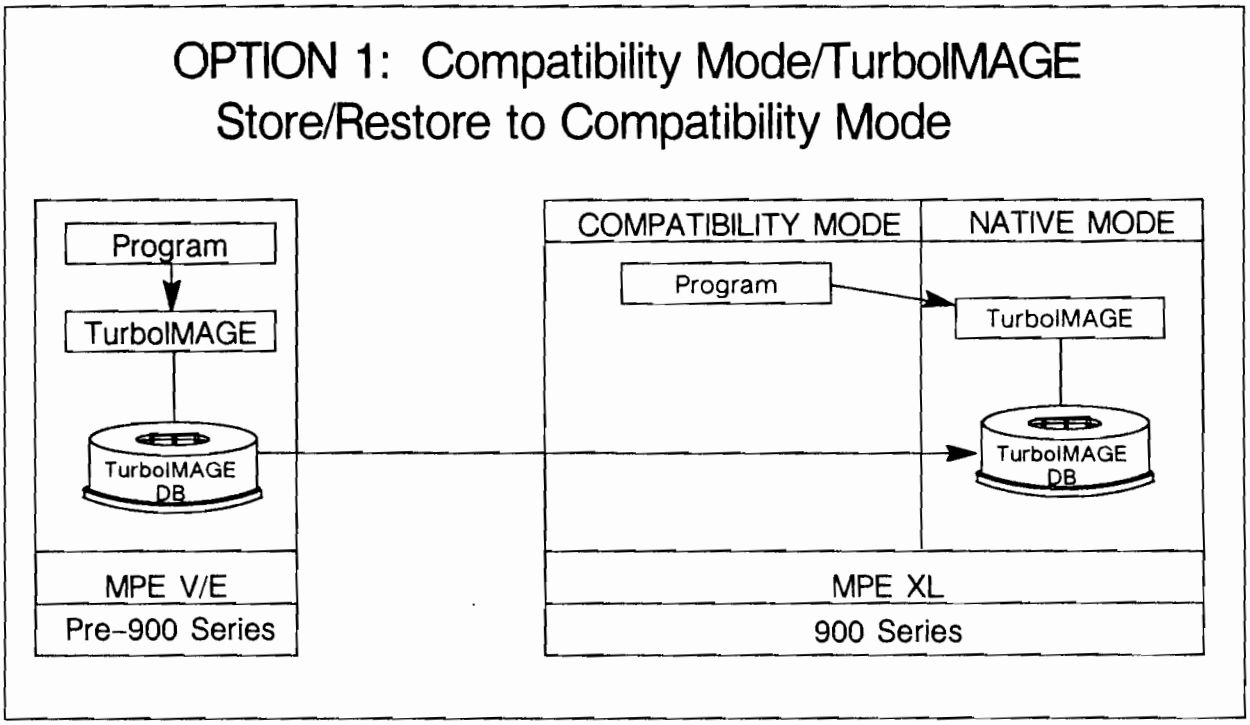
Option 1

Migration Option 1: Compatibility Mode programs accessing TurboIMAGE databases. Option 1 relies on MPE V/E object code compatibility. In most cases, the programs and databases are restored from an MPE V/E STORE tape. Option 1 has several advantages:

- MPE V/E program files run with few or no modifications.
- The TurboIMAGE database can be immediately accessed by the program.
- Users are immediately productive.
- No overhead is incurred for migration software.

Option 1 is considered the first step in application migration to Native Mode. While applications execute in Compatibility Mode, a migration project team can perform program migration to Native Mode without impacting productivity. Note that DBSTORE may be used to store the database from the MPE V/E system and that DBRESTOR may be used to restore the database onto the MPE XL system.

Since TurboIMAGE/XL runs in Native Mode, the TurboIMAGE/XL intrinsics are invoked via the Switch subsystem when a Compatibility Mode program calls a TurboIMAGE/XL intrinsic. The intrinsic calls do not need modification by the user. Access to Native Mode TurboIMAGE/XL intrinsics is transparent to the application because TurboIMAGE/XL intrinsic Switch stubs exist in the MPE XL Compatibility Mode System Library. Figure 1-3 illustrates Option 1.



LG200045-003

Figure 1-3. Option 1

Option 2

Migration Option 2: Native Mode programs accessing TurboIMAGE databases. Migrating the program from Compatibility Mode to Native Mode is one migration solution. Option 2 allows the program to retain the TurboIMAGE/XL intrinsic calls and still be compiled with a Native Mode compiler. The database remains in TurboIMAGE format and is accessed in Native Mode by existing TurboIMAGE/XL intrinsics. Figure 1-4 and Figure 1-5 illustrate Option 2. Option 2 offers these advantages:

- The application's performance may improve over Compatibility Mode execution.
- The TurboIMAGE/XL intrinsic calls remain intact.
- The database is compatible with MPE V/E-based systems. Therefore, programs that access the same database can be systematically migrated to Native Mode without impacting productivity of other Compatibility Mode programs accessing the database.
- Remote TurboIMAGE access is preserved.

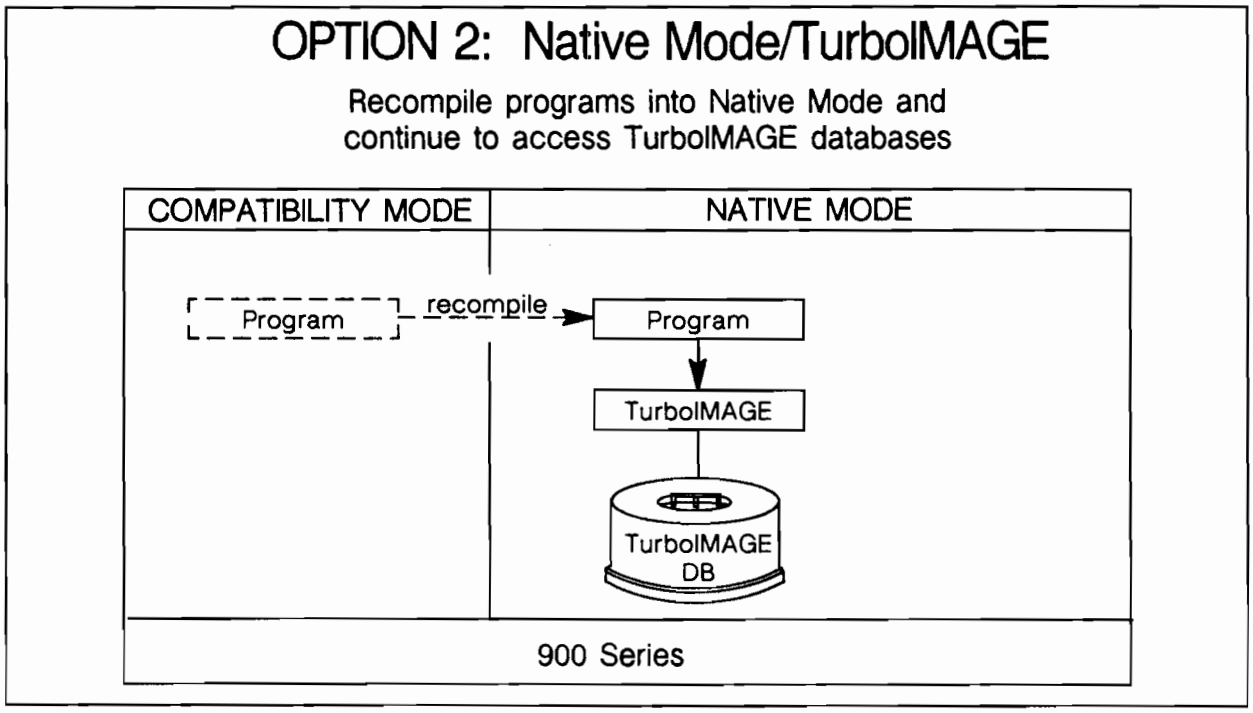


Figure 1-4. Option 2

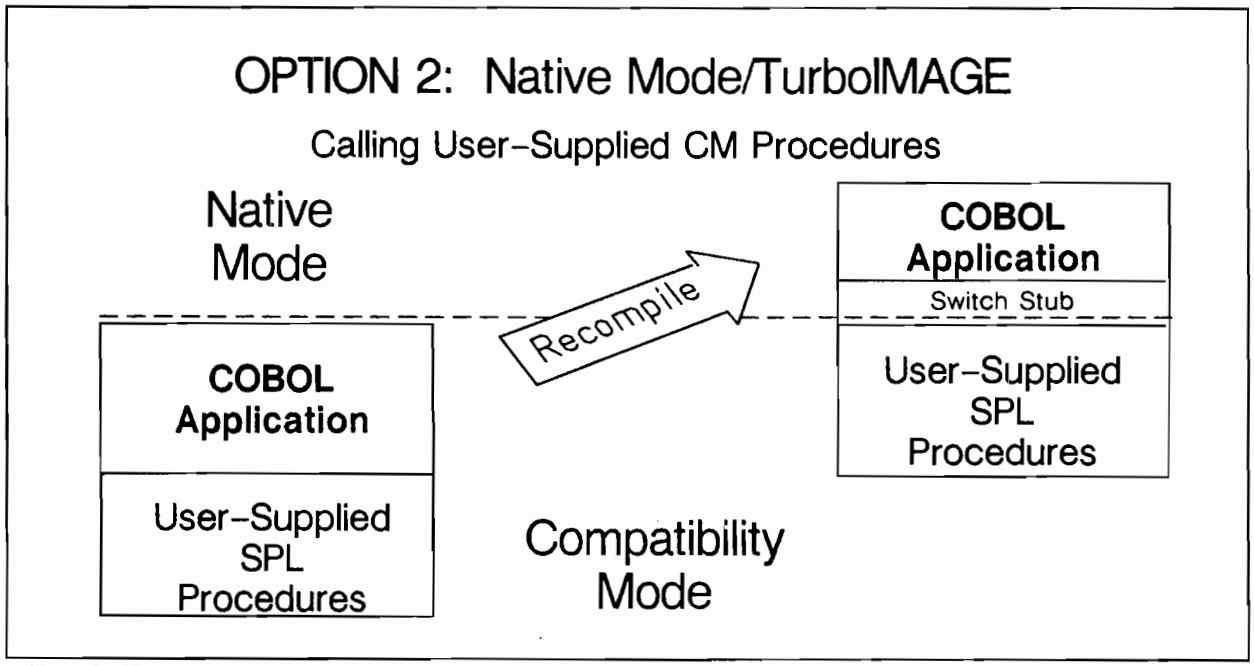


Figure 1-5. Option 2 with Switch

Program Migration Solutions

The phased approach in migrating MPE V/E-based programs to Native Mode over time allows a migration project team several options in achieving that goal. Hewlett-Packard has developed migration solutions for various system components: programs, operating environment, data files, and data communications. These solutions include:

- Migration to Compatibility Mode
- Migration to Native Mode
- Mixed mode applications
- Data alignment options
- Floating-point conversion routines

Migration to Compatibility Mode

Compatibility Mode is available to simplify the migration to the MPE XL system. This stage may not be necessary for certain applications. Most non-Privileged Mode MPE V/E programs and data can be restored to an MPE XL system and run in Compatibility Mode without the need for recompilation. This is extremely important in the cases where source code is not available for application programs. Programs may need to be modified before they can execute on MPE XL if they run in Privileged Mode or if incompatibilities have been identified. The Object Code Translator may be used to increase performance with this solution.

Compatibility Mode offers an immediate migration path to MPE XL-based systems and immediate productivity on the new system. Compatibility Mode program development is available in MPE XL, but migration to Compatibility Mode should be considered the first stepping stone on the path to Native Mode execution.

Compatibility Mode offers a variety of MPE V/E compatible compilers and the MPE V Segmenter for building and maintaining program and library files for execution in Compatibility Mode on the 900 Series. After recompiling the application, it is suggested that you thoroughly test the application to ensure valid results.

Migration to Native Mode

To fully utilize the features of MPE XL and HP Precision Architecture, Compatibility Mode programs need to be migrated to Native Mode. This is achieved through source code recompilation with MPE XL compilers. Before attempting recompilation, the migration project team should analyze the program for possible incompatibilities between Compatibility Mode and Native Mode. Any incompatibilities should be resolved before recompilation. After recompiling the application, it is suggested that you thoroughly test the application to ensure valid results.

COBOL II, HP FORTRAN 77, and Pascal programs can be compiled into Native Mode using the appropriate MPE XL optimizing compilers: COBOL II/XL, HP FORTRAN 77/XL, and HP Pascal/XL, respectively. In the future, optimizing compilers will be available. Native Mode also employs LinkEditor to build and maintain program files, relocatable libraries, and executable libraries.

The Native Mode compilers offer three levels of optimization, levels 0 through 2. Each incremental level provides a higher degree of execution performance. Optimization level 0 is "no optimization", while level 2 is "full optimization". The optimization levels are invoked through compiler options and directives. Hewlett-Packard recommends that programs be recompiled with level 0 optimization, tested, and subsequently recompiled with other optimization levels. Refer to the appropriate language reference manual for specific details on compiler optimization.

Migrating SPL programs to Native Mode is more complex. Due to its dependence on the architecture of MPE V/E-based systems, a Native Mode SPL compiler is not supported by Hewlett-Packard. Several solutions are available for SPL programs:

- Leave SPL programs in Compatibility Mode and maintain them with the SPL/V CM compiler available on MPE XL.
- Translate SPL programs into HP Precision Architecture machine instructions using the HP Object Code Translator (OCT). Note that while the newly translated program will not incur the overhead of the Emulator, the program still adheres to the restrictions imposed on any MPE V/E object file running in Compatibility Mode.
- Rewrite SPL programs using a high-level language that can be compiled into Native Mode; HP Pascal/XL or C, for example.
- Leave the SPL libraries in Compatibility Mode; migrate the program to Native Mode and access the libraries via the Switch subsystem.

Mixed Mode Programs

Recompilation to Native Mode may not be a viable alternative for all programs because source code may be missing, the program is written in SPL, or the program is written in another language for which a Native Mode optimizing compiler is not available. The same holds true for segmented libraries (SLs). If programs are executed (loaded) with group or account libraries, and the libraries cannot be migrated to Native Mode, then creating a mixed-mode program is an alternative. It is possible to migrate the program to Native Mode while leaving the libraries in Compatibility Mode. The Native Mode program can access the library procedures via the MPE XL Switch subsystem.

Switching between modes usually does not require source code modifications. In most cases, a program is recompiled to Native Mode while the libraries accessed by the program remain in Compatibility Mode. A Switch stub is required for every procedure that the program accesses from the CM library. These Switch stubs are placed in a Native Mode library which is linked with the program. A Switch stub is used to inform the Switch subsystem of the specifics of a procedure in a CM library. You can create Switch stubs either using the Switch Assist Tool (SWAT) or on your own.

Note that mixed mode program execution can also define a Compatibility Mode program accessing a routine in a Native Mode library. Once again, neither the program making the call, nor the procedure being accessed, need to be modified.

Some overhead is incurred when using the Switch subsystem. The user needs to understand the trade-off between the performance implications of using the Switch subsystem and the effort necessary to recompile routines to Native Mode. A mixed mode program can either be an intermediate step to full Native Mode or an alternative to rewriting (or in some cases locating) source code.

Mixed mode offers the highest degree of flexibility for the phased migration of programs. Even users who choose to recompile existing source code with the MPE XL version of compilers may need Switch if the source contains calls to user-written procedures which still remain in CM SLs. Switch is also needed by developers of new Native Mode applications that require the services of CM SL procedures. If application size is a consideration, performance-sensitive portions of the CM application may be rewritten or recompiled in Native Mode first, allowing CM programs to continue to call the procedures. Migrating critical portions of an application to Native Mode while the remainder of the application executes in Compatibility Mode results in performance gains and allows for a continual, incremental upgrade path.

Data Alignment

The migration team needs to be concerned about program data structure alignment and data files when programs are migrated to Native Mode. On MPE V/E-based systems and in Compatibility Mode, the natural alignment supported by the compilers is 16 bits. Data is aligned on 16-bit boundaries. In Native Mode the natural alignment supported by the compilers and the architecture is 32 bits. The data is aligned on 32-bit boundaries.

Data, or "flat", files generally contain byte streams of data. The data, when read, is interpreted as defined by the program's internal data structures. The data structures are aligned by compilers, as defined by the program, using the system's natural alignment. When MPE V/E programs are compiled to Native Mode, the natural alignment changes from 16 bits to 32 bits. But, the data in the files does not change. What changes is the alignment of variables when Compatibility Mode programs are compiled to Native Mode. Data accessed by the Native Mode programs may be interpreted inaccurately by the same program running in Compatibility Mode.

For example, an MPE V/E-based program that depended upon a 16-bit alignment for its record structure may find that same record structure aligned on a 32-bit boundary once the program is compiled to Native Mode. The migration project team needs to identify all such possibilities before migrating to Native Mode. Once identified, source code can be modified to retain data compatibility.

For example, an MPE V/E-based program that depended upon a 16-bit alignment for its record structure may find that same record structure aligned on a 32-bit boundary once the program is compiled to Native Mode. The migration project team needs to identify all such possibilities before migrating to Native Mode. Once identified, source code can be modified to retain data compatibility.

Each Native Mode compiler supports directives that select data compatibility with MPE V/E. For example, COBOL II/XL supports directives that align synchronized data either on 16-bit or 32-bit boundaries. HP Pascal XL provides a compiler option that directs HP Pascal/XL to size and align data the same way as MPE V/E. HP FORTRAN 77/XL provides a similar compiler option. Through the use of these compiler options, source modifications for data compatibility are minimized. For optimal performance, data files should be converted to the natural alignment for the 900 Series (32-bit).

Floating Point Representation

MPE XL supports two floating-point representations: HP 3000 Floating Point and IEEE Floating Point.

MPE V/E programs running in Compatibility Mode are not affected since the HP 3000 floating-point representation is supported on MPE XL. The Native Mode compiler options and directives used to select data alignments can be used to select HP 3000 floating-point representation or IEEE floating-point representation.

If full Native Mode performance and precision is necessary, then the customer will need to convert floating-point data to IEEE format. Hewlett-Packard has provided an intrinsic, HPFPCONVERT, that will convert a value represented by HP 3000 floating-point format to a value represented by the IEEE floating-point format and vice-versa. For complete information on data representation and floating-point conversion, consult the appropriate language migration manual. Conversion of data files will affect program conversions. Thus, migration is an iterative process; once all programs are in Native Mode, then all data should be converted to Native Mode.

Data Communication Migration Solutions

Data communication migration solutions apply to Hewlett-Packard manufactured networking subsystem. DS/3000 is not supported on MPE XL. NS 3000/XL is the supported data communication software on MPE XL. NS 3000/XL provides most of the features supported by NS 3000/V.

Hewlett-Packard recommends two solutions in migrating software dependent upon data communications:

- Update the MPE V/E system from DS/3000 to NS 3000/V; and convert all applications, job files, and User Defined Commands to use the NS 3000/V subsystem as the NS3000/V features are supported by NS 3000/XL.
- Migrate all applications, job files, and User Defined Commands directly to MPE XL and convert them to use the NS 3000/XL subsystem.

Operating Environment Migration Solution

An operating environment contains:

- Directory accounting structure
- UserDefined Command (UDC) structure
- Private volume configurations (volume sets and media)
- Global Resource Identification Number (RIN) configurations
- User logging identifiers
- User files

This is the user's operating environment. Hewlett-Packard recommends the use of the Hewlett-Packard supplied directory migration tool, DIRMIG, to migrate your MPE V/E operating environment to MPE XL. Available on the MPE XL installation tapes, DIRMIG is a flexible, one-step migration tool that migrates the MPE V/E operating environment from an MPE V/E SYSDUMP tape.

Migration Stages

Phased migration requires analysis of the existing application environment, consideration of MPE XL functionality requirements and dependencies, and careful planning based on migration options available, time, resources required, and desired performance. In order to help organize the activities involved in the migration process, a six-stage migration process is defined. These stages include:

- Education
- Analysis and planning
- Preparation
- Installation
- Compatibility Mode operation
- Native Mode operation

These stages should be used as a guideline for migration. Stages can, and often will, overlap or be skipped altogether. Figure 1-6 lists the migration project stages.

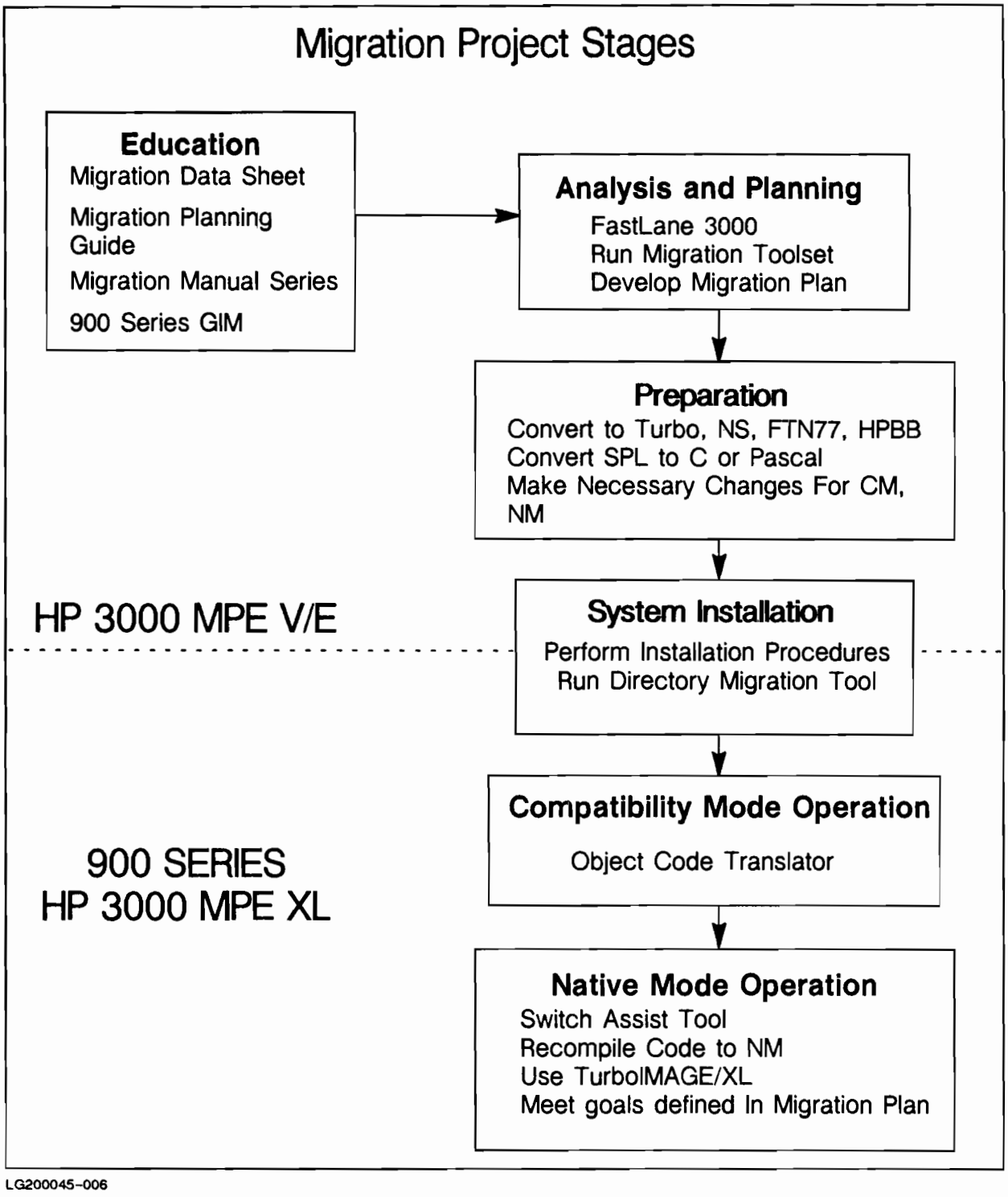


Figure 1-6. Migration Project Stages

Education

The first stage of any migration process is the education of the customer. During this stage, the customer is presented with an overview of migration, and high level customer expectations are set. Finally, the customer should become familiar with the generic migration process and should understand the availability of documents, tools, training, and services.

Analysis and Planning

This is the second stage of the migration process. During this stage you will develop your migration plan. Thorough analysis and planning is crucial to the success of migration. MPE XL has been designed as a superset of MPE V/E to provide compatibility between the two systems. However, you may find some incompatibilities which are a result of the enhanced features offered in MPE XL. There are some tools which will be helpful during this stage of the migration process. These tools are known as the Migration Toolset.

The process should begin by taking a system-level view of the components to be migrated, and the functionality provided by the new MPE XL system. The Migration Project Team should take inventory of all applications and databases to be migrated. Applications to migrate should include those which would benefit from the increased performance of the MPE XL system.

The effort involved in migrating each application should also be considered. Any incompatibilities which may exist in each user-developed application should be identified by running the migration tools. Incompatibility issues may be identified in any of the following areas: (1) obsolete MPE V/E commands, (2) intrinsics, (3) subsystems, (4) languages (SPL, in particular), (5) Privileged-Mode applications, and (6) data communications. The number of incompatibilities found and the scope of the changes required to work around them for both Compatibility Mode operation and Native Mode operation will provide an estimate of the migration effort needed. The analysis also assists in formulating a migration strategy for each application and data base.

The migration options and solutions should be used in defining a migration strategy for an application. A migration strategy for a particular application could specify which portions of the application will execute in Compatibility Mode, Native Mode, or mixed mode. It should describe the migration phases, such as, move all code and data to Compatibility Mode, test and verify that they meet or exceed performance expectations; recompile selected portions in Native Mode with Compatibility Mode data alignment options; and modify applications to access Native Mode 32-bit data. Hewlett-Packard assistance is available for developing a migration plan through FastLane 3000.

Preparation

This is the third stage of the migration process. Activities of this stage consist of updating software to make migration easier. This stage may span the entire migration process. It can start before the 900 Series arrives.

Preparation involves updating to more recent versions of MPE software and HP products such as databases, data communications, and languages. For example, the MPE V/E-based system must be updated to U-MIT or later software prior to migrating to the 900 Series HP 3000. Databases must be converted from IMAGE/3000 to TurboIMAGE using the utility DBCONV. Data communications software should be upgraded from DS/3000 to NS 3000. All applications should be updated to the appropriate language version if Native Mode performance is required. For example, FORTRAN/V should be updated to HP FORTRAN 77 and BASIC/V should be updated to HP Business BASIC. Any recoding necessary to remove incompatibilities found in the Analysis and Planning Stage can begin in this stage. Preparation should also include recompiling all source on MPE V/E to verify that the correct versions of source are available and used.

Installation

This is the fourth stage of the migration process. During this stage your 900 Series HP 3000 is installed and configured, and all programs and files are moved to the new machine.

The objective of the fourth stage is to have a functional MPE XL-based system in place in order to begin the migration. The result of this stage is a 900 Series HP 3000 installed with the MPE XL operating system, configured with appropriate peripherals, and prepared for Native Mode development. During this stage, the MPE V/E operating environment may be migrated using DIRMIG, and MPE XL subsystems are installed.

Compatibility Mode Operation

This is the fifth stage of the migration process. During this stage, object code files that ran on an MPE V/E-based system are run in Compatibility Mode (CM) on the MPE XL system, and their results are validated with the originals. It is suggested that you thoroughly test the application to ensure valid results.

This stage allows for immediate productive use of the 900 Series HP 3000. However, some migration plans may not include Compatibility Mode operation, opting instead, to move directly to Native Mode. This stage is recommended for users so that program integrity is verified before moving to NM. It may involve completing source code changes to remove CM incompatibilities found during analysis, thus easing the later Native Mode conversion. The Object Code Translator may be used in this stage to gain additional performance for CM applications.

Native Mode Operation

This is the sixth, and final, stage of the migration process. Programs developed on MPE V/E-based systems need to be recompiled with the MPE XL version of compilers in order to run in Native Mode. Certain programs may require some source code changes to operate efficiently on the 900 Series HP 3000.

The objective and result of this stage is the implementation of the migration plan so that it meets the goals and performance expectations of the end-user. The length of this stage depends on the scope of the migration plan and the strategy chosen and the extent of changes that must be made to the application to meet the performance goals and user expectations. It may even overlap with the Compatibility Mode Operation Stage, depending on the migration plan.

Activities included in this stage are recompiling source files using MPE XL compilers, determining what mixed-mode calls may be necessary for an application, and completing changes for Native Mode incompatibilities found in the Analysis and Planning Stage. The results may include development of mixed-mode applications, Native Mode programs manipulating MPE V/E-compatible data, and ultimately, Native Mode programs manipulating MPE XL-compatible data. This stage will most likely occur in phases where each phase is followed by a test period to verify the correctness of the application and an evaluation period to verify performance.

After a particular application has been validated under Compatibility Mode operation, the next phase may involve partial recompilation to Native Mode using the Switch subsystem to access procedures in Compatibility Mode SLs. At the end of this phase, if performance of an application which uses Switch has not met the goals defined in the migration plan, the plan may be modified to include an additional phase. The additional phase would involve also recompiling the Compatibility Mode procedures to Native Mode or rewriting the Compatibility Mode procedures if they are implemented in a language not supported in Native Mode.

Migration Tools

HP has developed the following tools to help with migration:

- Migration Toolset which includes two tools that identify possible incompatibilities between MPE V/E and MPE XL. These tools, Run-Time Monitor and Object Code Analyzer, are discussed in appendices to this manual. The Object Code Analyzer (OCA) uses MPE V/E object code files as input and generates a list of possible incompatibilities between MPE V/E and MPE XL as output. The Run Time Monitor (RTM) provides customers with a list of run-time incompatibilities by monitoring applications as they execute and logging incompatibilities as they occur.
- Directory Migration Tool (DIRMIG), migrates the operating environment. Operation of DIRMIG is discussed in an appendix to this manual.
- Switch Assist Tool (SWAT), facilitates phased application migration from Compatibility Mode to Native Mode and builds Switch Stubs to allow mixed mode execution of applications.
- The Object Code Translator (OCT) performs an optimized translation of MPE V/E-based object code to MPE XL-based object code.
- Language conversion utilities are available to change applications from FORTRAN 66/V to HP FORTRAN 77/V and from BASIC/V to HP Business BASIC/V. The appropriate conversion utility is supplied with HP FORTRAN 77/V, HP FORTRAN 77/XL, and HP Business BASIC/V.

- A data alignment compiler option is available with all Native Mode compilers, so that data can be aligned on 16-bit or 32-bit boundaries for access from Compatibility Mode or Native Mode programs, respectively.
- An intrinsic, HPFPCONVERT, converts between HP 3000 and IEEE floating-point representations.
- DBCONV is a utility that will convert your IMAGE/3000 database to a TurboIMAGE database. It is available on MPE V/E systems.
- SDCONV is a program that converts and loads data from Dictionary/V to System Dictionary/XL.

Documentation and Training

In addition to this manual, Hewlett-Packard has developed many other manuals and training classes to assist with various aspects of migration.

Documents

- *General Information Manual* (5954-7418) provides a general overview of the 900 Series HP 3000 family.
- *HP 3000 Computer Systems System Configuration Guide* (5954-9354) provides current information on supported peripherals.
- *BASIC/3000 to HP Business BASIC Conversion Guide* (32115-90004) provides information on converting applications written in BASIC/V to HP Business BASIC/V.
- *COBOL II/3000 Conversion Guide* (32233-900005) provides information on converting applications written in COBOL 68 to COBOL II.

Migration Training

FastLane 3000 is an Hewlett-Packard consulting service that helps you achieve more efficient use of time and resources as you migrate your application to the 900 Series HP 3000 system. It helps you understand how the migration process could impact your particular data processing environment, and what actions you can take to streamline that process.

FastLane 3000 is delivered to your migration project team by a specially trained Hewlett-Packard engineer and consists of two component services:

- System Planning builds the foundation of knowledge essential to migration analysis and planning.
- Application Planning guides your project team through the steps of developing a complete migration plan for one of your applications.

Migration Project Team

The most important item to remember is that careful planning and a working knowledge of the existing installations needs and applications are key to a successful migration effort. The migration team must fully understand all the migration options and the ramifications of migrating some applications and not others. Before actually beginning the migration process, you should define the team of people assigned to complete the various tasks of migration. A well balanced migration project team should consist of a project leader and representatives from:

- Operations staff
- Programming staff
- Hewlett-Packard management staff
- Users of major applications to be migrated
- Hewlett-Packard representative

Depending on the extent of your migration, one or more representatives from each area need to be included in your migration project team.

Project Leader Responsibilities

The project leader coordinates activities of the project team. To do this the project leader must:

- Serve as a key contact person for Hewlett-Packard for FastLane 3000
- Assign tasks and responsibilities to individual team members
- Assess documentation and training needs and ensure that they are met
- Schedule and conduct team meetings
- Monitor overall project
- Maintain Migration Project Notebook
- Report progress to users and upper management
- Resolve conflicts

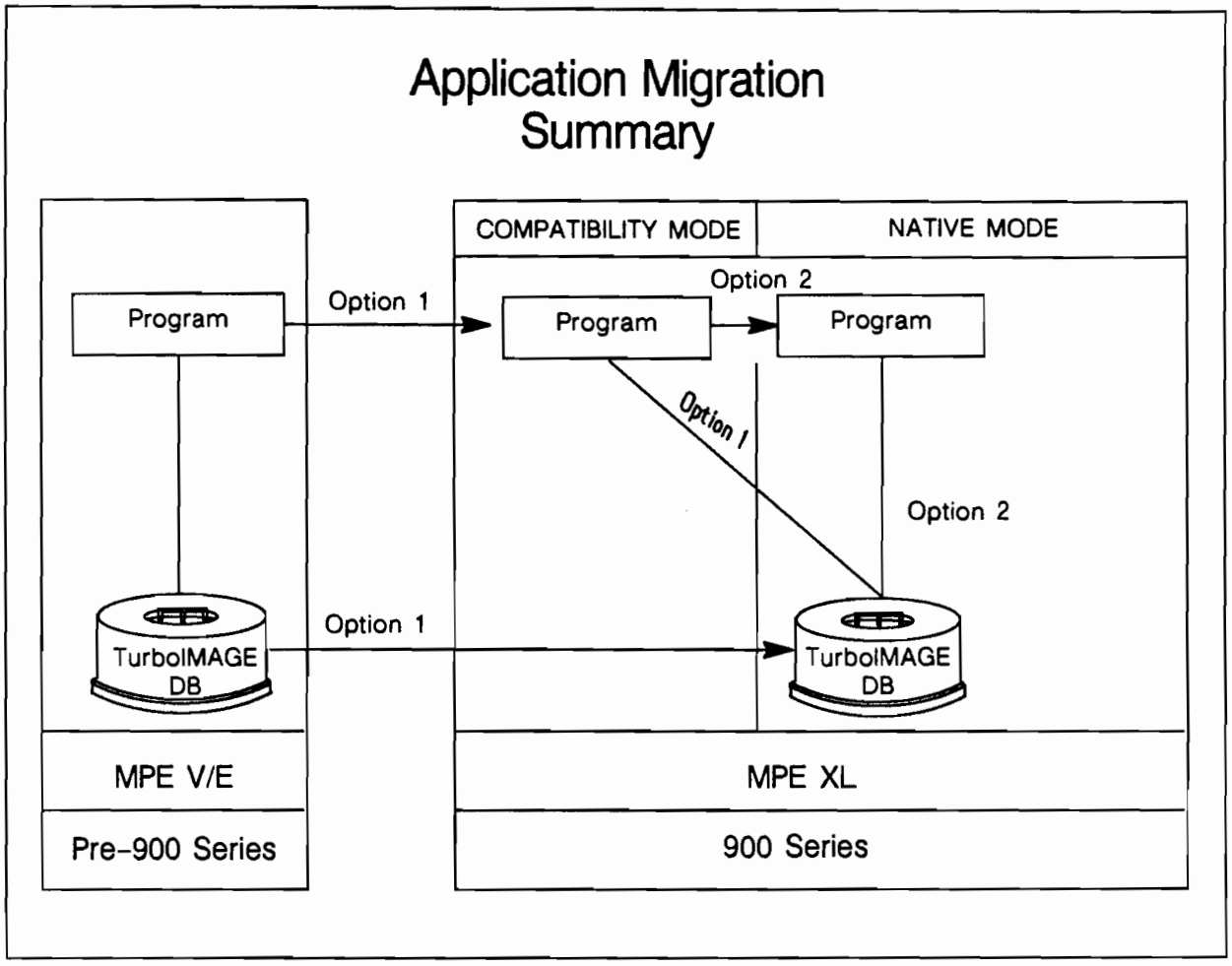
Project Team Responsibilities

The project team is responsible for project management. The duties of the project team include:

- Set goal and scope of the project
- Identify key activities and milestones
- Develop detailed project schedule
- Monitor progress of assigned tasks

Conclusion

HP has provided solutions to ease the migration process at each stage: the Migration Toolset (Object Code Analyzer and Run Time Monitor) for use during the analysis/planning stage, DIRMIG for operating environment migration during the installation stage, and SWAT for creating mixed-mode applications during the migration to Native Mode. However, before these tools can be effectively used, it is important to understand the migration process and the key to a successful migration. A successful migration depends on three main concepts: careful analysis and planning to formulate a realistic migration plan, understanding the benefits and trade-offs of the migration options, and taking advantage of the ability to phase the migration process. Figure 1-7 summarizes the migration options for programs and databases.



LG200045-007

Figure 1-7. Summary of Migration Options

2

Analysis and Planning

Introduction

Analysis and planning is the second stage of migration. The objective of this stage is to gather and analyze data in order to assess the requirements and time frame for migration. The result of this stage is a migration plan based on a thorough analysis of the applications to be migrated. Activities of this phase are divided into the following groups:

- Analyzing the application
- Planning the migration

Analyzing the Application

Careful and thorough analysis will ensure a successful migration plan. When analyzing an application to be migrated, you should include the following considerations:

- Which applications will migrate to an MPE XL-based system?
- Does the application have any incompatibilities?
- What are the characteristics of the application?
- How much time do you have to complete the migration?
- What resources are needed to complete the migration?
- What are the migration goals for the application?

Detecting Incompatibilities

Object Code Analyzer (OCA) and Run Time Monitor (RTM) are migration tools that can detect most incompatibilities. These tools allow you to detect problems in object code and at run-time before migrating to the 900 Series HP 3000. Some incompatibilities cannot be detected with these tools and can only be determined by having working knowledge of what the application does and how it does it. Appendix D lists all currently known incompatibilities between MPE V/E and MPE XL.

Analyzing Object Code Problems

The Object Code Analyzer is a utility to aid in migrating application programs from an MPE V/E-based HP 3000 (running U-MIT or later) to an MPE XL-based HP 3000. OCA reports definite problem areas and warns of potential problems that may be encountered when migrating an application to an MPE XL-based system. The Object Code Analyzer can scan program and SL files on MPE V/E-based HP 3000 systems in either interactive or batch mode. OCA will be able to locate any potential incompatibilities with only one pass. Appendix B describes how to use the Object Code Analyzer.

Object Code Analyzer can:

- Locate program and SL files to be analyzed.
- Identify incompatibilities independent of the source language.
- Find all occurrences of a potential incompatibility. OCA does not depend on run-time execution.
- Identify the location of incompatibilities in code by reporting the code segment and the offset within the segment.
- Determine which SL files are required to migrate an application.
- Look for calls to uncallable entry points to the MPE V/E operating system.
- Look for calls to any user-selected external procedure.

OCA can be used to provide code segment and offset information when an incompatibility is found. You can then determine the procedure and the offset within the procedure by using a PMAP to trace an incompatibility back to a location in the source code. This information simplifies the task of identifying the exact location of the incompatibility. By examining each of these locations you can determine if a true incompatibility exists.

Monitoring Run-Time Events

Run Time Monitor is a data capture facility that allows you to identify incompatible areas at run-time within MPE V/E-based applications. The events logged consist of calls to MPE V/E intrinsics and programmatically executable commands which are not supported, or have modified parameters or return values in MPE XL. When a user program or library procedure makes a call to system software to invoke an incompatible feature, the event and location of the call is logged. You can select from predefined classes of events to detect with the program RTMSYS. A report program, RTMREP, provides summaries according to program and event, in addition to a record-level detailed report.

You should run RTM long enough to have a complete activity cycle monitored. For instance, if you want to determine if an accounting application has any run time incompatibilities, you will need to have RTM enabled long enough to monitor not only end-of-week or end-of-month activities, but end-of-quarter or end-of-year activities to ensure that all portions of the application program paths are properly exercised and monitored. Appendix C describes how to use the Run Time Monitor.

Run Time Monitor can:

- Identify incompatibilities independent of the source language.
- Trap all events from all library and program paths executed.
- Trace an event to the code segment and offset in code. Use this information, with a PMAP and source listing to locate the incompatibility.
- Flag a potential problem in cases where return codes or values have been modified for MPE XL. However, no incompatibility exists if your program does not depend on the modified return value.

Object Code Analyzer and Run Time Monitor: A Comparison

There are some inherent differences between OCA and RTM. While one tool may be better suited for detecting a particular type of event, it is recommended that you use both tools to get an accurate picture of your potential incompatibilities.

OCA is best suited for:

- Complete coverage of all programs and SL files to be migrated to a 900 series HP 3000.
- Detecting incompatibilities in code paths that may not have been executed while Run Time Monitor was enabled.
- Locating calls to user selected external procedures.
- Quickly determine potential incompatibilities.
- Detecting calls to unsupported routines.

RTM is best suited for:

- Investigating a particular class of events based on an OCA report.
- Detecting incompatibilities that can only be determined at run-time.
- Long-term analysis of the application.
- Obtaining information on the frequency of particular incompatibilities.

Analyzing the Reports

Appendix D lists the incompatibilities detected by the Object Code Analyzer and Run Time Monitor. While this list of possible incompatibilities appears long, most programs only contain a few, if any, of these. Also included is the cause of the incompatibility and any work around for the problem. Use this information plus the information in "Undetected Incompatibilities" to plan your migration strategy and to identify any changes that you need to make in your applications.

Potential incompatibilities reported by OCA can fall into one of four categories as illustrated by Figure 2-1. These categories relate to whether the event was executed while RTM was in effect and if the event detected by OCA is a real incompatibility. For example, OCA detects a COMMAND intrinsic call, event 200, but is unable to determine if the command executed in the call is incompatible. RTM only reports on the call if it is executed with an incompatible command. Since a different command may be executed at different times by the same COMMAND intrinsic call, what is compatible at one time may not be compatible at another. Thus, the fact that OCA detected a possible incompatibility with the COMMAND intrinsic, event 200, and that RTM did not detect a problem with the COMMAND intrinsic, events 201 through 218, does not necessarily mean that the COMMAND call detected by OCA is not problematic with respect to events 201 through 218. Careful inspection of the code in question may be necessary to resolve all potential incompatibilities. When analyzing the reports generated by OCA and RTM, take into account the limitations of each tool on detecting incompatibilities.

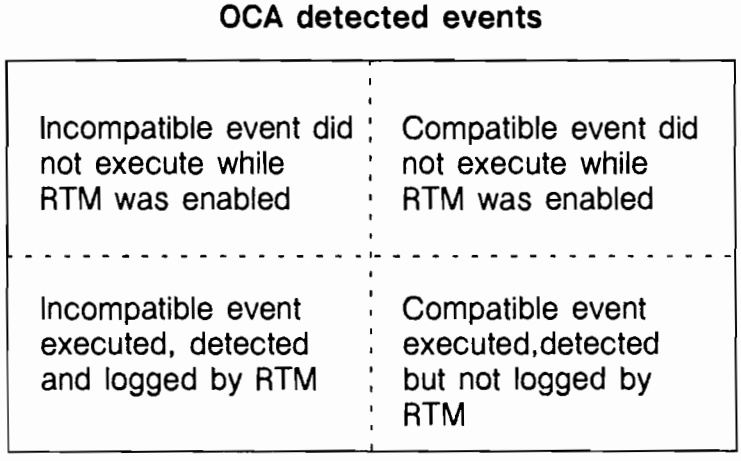


Figure 2-1. Compatibility Matrix.

Your migration strategy should be influenced by the number and complexity of the changes you need to make to your application to have it function properly on MPE XL-based systems. Also, you should note the changes which are necessary to run in Compatibility Mode or Native Mode, as this too influences your migration strategy. A thorough understanding of the application may be necessary to assess the impact of the incompatibility. Once all necessary changes are identified, and the migration plan is developed, you can estimate the resources needed to complete your migration.

Severity of Incompatibilities

Once you have determined what potential incompatibilities exist in your application, you need to determine how your application depends on the item in question. In some cases, what is flagged as an incompatibility may not cause your application to run incorrectly but it is something that you should investigate to determine the impact on your application.

To determine accurately what, if any, modifications your application will require to run on the 900 Series HP 3000, you have to decide whether the application will be run in Compatibility Mode or Native Mode. Additionally, you need to know whether it needs to call Native Mode or Compatibility Mode procedures.

The most severe incompatibilities include the use of Privileged Mode, uncallable routines, or undocumented entry points. Resolution of these potential incompatibilities will require a detailed study of the application in question.

Application Characteristics

Your application may consist of more than just a program file. A complete application may consist of programs, SLs, data, job streams, and UDCs. The application may have other characteristics that will affect your migration strategy such as:

- The Hewlett-Packard languages used by the application.
- The number of lines of code in each program in the application.
- If the source code is available for all the programs in the application.
- Whether or not the application relies on Third Party software.
- The MPE V/E features that the application uses.
- How the application uses MPE intrinsics.
- How the application accesses data.
- Whether or not the application encounters storage or performance limits.
- Whether or not the application interfaces with another application.
- What related applications are being migrated.

Migration Goals

After analyzing the incompatibilities and characteristics of the application, you need to establish migration goals. These goals should define the intended mode of operation for the application, and what data alignment is expected by the application. You may also want to identify interim goals. For instance, if you have an application that you ultimately want to migrate to Native Mode, but it shares data with another application which will not be migrated for some time, an interim migration goal might be Compatibility Mode or mixed mode, until the second application can be migrated to Native Mode. To accurately determine the goals for migrating the application you will need to consider the following:

- What languages are available in the desired mode?
- Is the source code available for making changes and recompiling?
- What incompatibilities need to be resolved?
- Does going to Native Mode increase the number of incompatibilities?
- What mode do dependencies work in?
- What data format is required?
- Do you need to share files with MPE V/E systems in a network?
- Will source and program files be sharable across both MPE V/E and MPE XL based machines?

Time and Resources

Once you have established the migration goals, you will need to determine what time and resources you need to commit to completing those goals. Time and resource requirements needed may be influenced by the following factors:

- The number of lines of code in the application.
- Availability of source code.
- Tools used in software development.
- Activities needed to properly exercise all code paths for RTM.
- The needs of the application user group.
- The length of time both machines will be available to complete parallel testing.

Planning the Migration

Now that the analysis is complete, and you have developed a strategy for migrating your application, it is time to develop the migration plan. Your complete migration plan should include:

- Purpose and scope
- Project identification
- Migration strategy
- Migration schedule
- Resource requirements
- Documentation and training requirements
- Detailed application analysis
- Test plan

Purpose and Scope

This is the first section of your migration plan. The scope should include all areas addressed by the plan. The purpose is a statement of how you will use the migration plan.

Project Identification

This is the second section of your migration plan. Project identification includes both customer and application identification. Application identification is a brief description of the application covered by this migration plan.

Migration Strategy

This is the third section of your migration plan. The migration strategy includes migration goals, target state, major milestones, project completion criteria, project team members, support staff, and your Hewlett-Packard support team. If the application has no incompatibilities you may be able to move the application directly to Native Mode. If however, your application has some incompatibilities, then your migration strategy will be slightly more complex. There are different methods of defining the migration strategy for a particular application. The flexibility offered by Hewlett-Packard in migrating applications implies decisions must be made about which migration options to choose.

A migration goal is a statement describing the functionality and performance expectations, as well as a timeframe for completion. Target state describes the operating mode and options for all programs and databases that make up the application. The target state should also specify which portions of the application will be in Compatibility Mode, Native Mode, or mixed mode with Switch Stubs. Major milestones list each major step involved in reaching the target state. These milestones will be the basis for the migration schedule. Project completion criteria lists the specific criteria for project completion.

The migration strategy should also list the project team members, support staff, and Hewlett-Packard personnel involved.

Migration Schedule

This is the fourth section of your migration plan. The migration schedule should be a graphic representation of milestones and durations. For example, the vertical axis lists milestones and the horizontal axis lists dates pertaining to duration and completion. Milestones include: detailed analysis, migration plan, software prepared, installation, Compatibility Mode test, recompiling to Native Mode, database conversions, Native Mode test, project completion, and all intermediate milestones.

Resource Requirements

This is the fifth section of your migration plan. Resource requirements should list the hardware and software requirements for completing migration. Hardware requirements list the 900 Series, disc drives, tape drives, printers, terminals, and any other hardware needed for migration. Software requirements list MPE XL with version number, compilers, subsystems, tools, DBMS, and any other software needed for this project.

Documentation and Training Requirements

This is the sixth section of your migration plan. Documentation and training requirements should list the training and documentation needs for all personnel involved in the migration project. Training needs should list all training courses, dates, and who will attend. Documentation needs should list migration guides and reference manuals. Indicate how many copies of each are needed and when they are needed. Be sure to plan time for individuals to read and study the documentation. Other documentation considerations include updating in-house documentation that needs to be changed due to migration.

Detailed Application Analysis

This is the seventh section of your migration plan. A detailed analysis of every application being migrated includes: application organization, languages used, number of lines of code, system dependencies, user interfaces, networking, database access, Third Party software, contributed library dependencies, known incompatibilities, potential problem areas, list of intrinsics, and external references. A procedure flowchart or call map can be very useful in describing the application. Reports from the Migration Toolset (Object Code Analyzer and Run Time Monitor) should be included here.

Test Plan

This is the eighth and last section of your migration plan. The test plan should include the test data and procedures that will be used at each stage of the migration. A complete test plan will ensure that most problems are avoided before the application is in full use and should ensure that any change made to the application is thoroughly tested.

3

Preparation

Introduction

Preparation readies the existing MPE V/E system for migration. Software updates can be done before the 900 Series hardware arrives. This involves updating to U-MIT or later software, including databases, data communications, and languages.

Compiler Conversions

Applications written MPE V/E-based languages will run in Compatibility Mode, but moving these applications to the new compilers provides increased performance and features offered by Precision Architecture.

The following changes are recommended for migration to Native Mode:

- Avoid the use of SPL for future development. Translate SPL code into HP Pascal. SPL will run in Compatibility Mode and an SPL compiler will be available for developing and maintaining SPL code in Compatibility Mode on the 900 Series HP 3000.
- Convert applications written in FORTRAN 66/V to HP FORTRAN 77/V.
- Convert applications written in BASIC/V to HP Business BASIC/V. For more information, refer to *BASIC/3000 to HP Business BASIC Conversion Guide* (32115-90004).
- Convert applications written in COBOL 68 to COBOL II. For more information, refer to *COBOL II/3000 Conversion Guide* (32233-90005).

Subsystem Changes

Convert all databases to TurboIMAGE before migrating to the 900 Series HP 3000.

Convert from DS/3000 to NS 3000. NS 3000/XL is the data communications system for the 900 Series HP 3000. To convert DS/3000 to NS 3000, examine job streams and UDCs for DS/3000 control commands that may not work on NS 3000, and make the appropriate changes.

Application Changes

Decide whether to move the application to Compatibility Mode (CM). Examine RTM and OCA output for CM incompatibilities. Determine when to convert the application to Native Mode (NM). Then, examine RTM and OCA output for CM/NM incompatibilities.

The use of undocumented intrinsics, nonintrinsic SPL procedures, and other procedures that remain in Compatibility Mode (CM) that are called from high-level languages, require Switch stubs. Examine cross-mode calls for performance trade-offs.

Preparing the HP 3000 MPE V/E System for Migration

The Directory Migration Tool (DIRMIG) allows the System Manager to duplicate the MPE V/E operating environment on the MPE XL system. The operating environment consists of account structure information, including the UDC environment, private volume information, and system tables information. The Directory Migration Tool runs on your MPE XL-based system and takes as input your MPE V/E SYSDUMP tape. Before using an MPE V/E SYSDUMP tape and DIRMIG to migrate your operating environment, please review these checklists:

Accounting Structure

- Certain MPE XL accounts and groups are reserved for system use. Check for accounts and groups on the MPE V/E system that are reserved on the MPE XL system (`DIAG.SYS`, `CONFIG.SYS`, `MPE XL.SYS`). DIRMIG will not migrate these groups and accounts.
- Use `:LISTF` to check for files that exist on both the MPE V/E system and the MPE XL system. DIRMIG will not restore user files that already exist on MPE XL. Purge duplicate files, or create groups and accounts on the MPE V/E system to accommodate duplicate files, if needed. Move duplicate files into new groups and accounts on the MPE V/E system.

User Logging ID Table

On the MPE V/E system:

- Use `:LISTLOG` to determine which User Logging IDs to migrate.
- Use `:RELLOG` to release logging identifiers that should not be migrated or which are no longer in use.
- Define user logging parameters via SYSDUMP. This includes the number of user logging processes, and the number of users per logging process.

UDCs and User Files

Disable any system-level, account-level, and user-level UDCs that the users do not want to migrate.

- A tool will be provided on the MPE V/E system to create a file containing the names of the Command File COMMAND.PUB.SYS and all active MPE V/E UDC files. Use this file when creating the SYSDUMP tape.
- Edit the file created with this tool containing:
  - COMMAND.PUB.SYS
  - UDC environment file names
  - Add the names of any other file sets the System Manager wants to restore on the MPE XL system.

Create a SYSDUMP Tape

NOTE

A non-carriage-return full backup tape is recommended.

Use SYSDUMP to:

- Delete global RINs that will not be migrated. Configure RIN parameters.
- Define user logging parameters. Configure appropriate user logging ID parameters, including the number of user logging processes and the number of users per logging process.
- Enter the name of the indirect file created by this tool in response to the *fileset(s)* prompt.

Preparing to Perform Migration

Ensure that all necessary components are available before continuing.

An Hewlett Packard representative should have installed and configured the 900 Series HP 3000. Check to be sure that the system is up and booted.

NOTE

Please refer to the *HP 3000 Computer Systems System Configuration Guide* (5954-9354) for current information on supported peripherals.

Software Requirements

An MPE XL load tape, and a SYSDUMP tape from an MPE V/E-based system are required to complete system migration.

4

System Installation

Introduction

During Stage Four, System Installation, the migration team completes the installation, initial system test, and configuration of the 900 Series HP 3000.

The team uses the Directory Migration Tool (DIRMIG.PUB.SYS) to migrate most of the MPE V/E operating environment to MPE XL.

Tasks performed during the installation stage include:

- Installing MPE XL
- Configuring the MPE XL system
- Completing System Migration
- Installing HP subsystems with AUTOINST XL

Directory Migration Tool (DIRMIG)

The Directory Migration Tool (DIRMIG) migrates the MPE V/E operating environment to MPE XL. The operating environment includes:

- System tables information
- Global Resource Identification Numbers (RINs)
- User logging identifiers
- Directory structure/user files
- UDC environment
- Private volume information

DIRMIG uses an MPE V/E SYSDUMP tape as input.

The Directory Migration Tool is menu-driven and provides the user with an interactive means of selecting any combination of MPE V/E components to migrate:

- Global RINs
- User logging identifiers
- The directory, including UDC/user files
- Private volume information

DIRMIG features partial directory migration, and the choice of overriding or not overriding existing directory information on the MPE XL system with MPE V/E directory information, if accounts conflict. This permits directory and UDC information from multiple MPE V/E systems to be merged on one MPE XL system.

The Directory Migration Tool also has the ability to detect MPE V/E directory corruption.

DIRMIG restores user files with the KEEP option specified. This prevents the user from restoring over files installed from the MPE XL installation tape. The user has the option of choosing which file sets to restore. A log file, DIRLOGnn.PUB.SYS, is created each time DIRMIG is run, to track the progress of migration. The logging facility records the migration status of each component and provides detailed information in the event that any errors occur.

To assist with private volume migration, DIRMIG creates three other files: PVASSIST.PUB.SYS, PVSUMARY.PUB.SYS, and VOLUTIL Command Files as specified by the user. PVASSIST contains private volume information for the accounts DIRMIG migrates. PVSUMARY is a VOLUTIL Command File generated from all the information contained in PVASSIST to initialize media with MPE V/E directory information. The Volume Utility, VOLUTIL.PUB.SYS, allows the user to create, modify, and delete volumes in MPE XL.

PVSUMARY must be modified to make it usable by VOLUTIL and the VOLUTIL command files. The user must:

- Provide logical device information for media initialization.
- Modify member and class information for volume sets, if any changes are desired.
- Delete/modify spanning information.

These modifications can be made using any HP 3000 standard text editor, such as Editor/3000 or TDP.

To use DIRMIG, you must logon as MANAGER.SYS,PUB, mount the MPE V/E SYSDUMP tape, then run the DIRMIG program.

The DIRMIG main menu offers several options:

- Complete migration with no dialog
- Complete migration with dialog
- Migration of global RINs
- Migration of user logging IDs
- Directory migration
- Private volume environment migration

The dialog option of complete migration displays more detailed menus at each step of the process. These detailed menus allow the user to continue without migrating a particular component, to display information from the MPE V/E SYSDUMP tape, and to use the HELP feature.

5

Compatibility Mode Operation

Introduction

This is the fifth stage of migration. The objective of this stage is to ensure that programs that run successfully on MPE V/E-based HP 3000 systems also run successfully on MPE XL-based HP 3000 systems in Compatibility Mode. The result of this stage is a set of Compatibility Mode applications whose results have been validated with the original versions. Activities of this stage include:

- Completing the changes identified in the analysis and planning stage necessary for operation in Compatibility Mode.
- Verifying program integrity after changes have been made.
- Using the Object Code Translator.

Compatibility Mode

Compatibility Mode provides most of the functions available to programmers in MPE V/E-based systems. Additionally, Compatibility Mode programs can access Native Mode data files or procedures. All MPE V/E object code is supported in Compatibility Mode. In addition SPL/V, RPG/V, and HP Business BASIC/V compilers are available in Compatibility Mode at first release. Other subsystems available for Compatibility Mode use include: KSAM/V, Transact/V, Report/V, Business Report Writer/V, VPLUS/V, Dictionary/V, Inform/V, and HP Access Central/V.

By default, Compatibility Mode programs will access data that is compatible with MPE V/E-based systems.

Native Mode procedures can be called through the Switch subsystem.

If recompiling is necessary for Compatibility Mode operation, you should use one of the MPE V/E compilers available in Compatibility Mode on MPE XL and the MPE V/E Segmenter if necessary.

Program Validation

A complete Compatibility Mode test plan should be part of your migration plan. Once all necessary changes have been made to get your application running in Compatibility Mode, it is time to complete the test procedures. When you first run your program in Compatibility Mode, you may want to continue to run the original version on the MPE V/E-based system to ensure that the migrated version gives the same results as the original version of the program. Testing should include monthly, quarterly, or yearly activities that are critical to your operation.

Improving Performance

The Object Code Translator (OCT) converts Compatibility Mode object code to the 900 Series Instruction Set for increased performance. The effort involved in using the Object Code Translator is equivalent to recompiling, and, like recompiling, the application should be retested to verify proper execution. You can invoke the Object Code Translator through the MPE XL :OCTCOMP command.

OCT translates most Compatibility Mode instructions into HP Precision Architecture instructions and appends them to the end of the destination file. The resulting file can be executed on either MPE V/E-based or MPE XL-based systems.

By using the MPE XL command :OCTCOMP, you can create a new file with translated object code, translate only selected segments of the object code, or add translated segments to another file. Object code translation will increase the size of the file.

6

Migration to Native Mode

Introduction

This is the sixth and final stage of migration. The objective of this stage is the completion of the migration plan. The result of this stage is an application that meets the goals and performance expectations of the migration plan and end-user. The length of this stage depends on the scope of the migration plan and the strategy chosen. This stage may overlap with the Compatibility Mode operation stage. Activities of this stage include:

- Completing any changes to Native Mode incompatibilities.
- Recompiling with Native Mode compilers.
- Verifying program integrity.
- Converting data files to Native Mode (32-bit) alignment.

This stage may have several distinct phases. The results of these phases may include mixed-mode applications, Native Mode programs manipulating MPE V/E compatible data, and, ultimately, Native Mode programs manipulating MPE XL compatible data. Each phase should be followed by a test period to verify that the application functions properly and meets performance expectations. Also during this stage you will need to create stubs for any calls to Compatibility Mode procedures. The Switch Assist Tool (SWAT) is available on MPE XL to help you create Switch Stubs.

The Native Mode program environment offers a variety of compilers and products. Native Mode compilers available at first release include: HP FORTRAN 77/XL, COBOL II/XL, and HP Pascal/XL. Products that function in Native Mode are: NS3000/XL, TurboIMAGE/XL, ALLBASE/XL, Toolset/XL, VPLUS/XL, and System Dictionary/XL.

After a particular application has been validated under Compatibility Mode operation, the next phase may involve partial recompilation to Native Mode using the Switch subsystem to access procedures that still reside in Compatibility Mode SLs. At the end of this phase, if performance of the application does not meet the goals defined in the migration plan, the plan may be modified to include a third phase. The third phase would involve recompiling the Compatibility Mode procedures to Native Mode or rewriting the Compatibility Mode procedures, if they are written in a language that is not supported in Native Mode.

Using Compatibility Mode Data

While your application is running in Native Mode, it may need to manipulate data that is compatible with MPE V/E data files. This can be accomplished by using the appropriate compiler directive during a Native Mode compile. For more specifics on compiler options, refer to the appropriate language conversion guide.

The MPE XL operating system also recognizes two formats for storing floating-point numbers, HP 3000 and IEEE. The HP3000_16 and HP3000_32 compiler directives will also determine the storage format for floating-point numbers. The HPFPCONVERT intrinsic converts between these different formats. Compatibility Mode code uses data structures with 16-bit alignment by default, while Native Mode code uses 32-bit data alignment by default.

NOTE

The HP3000_16 compiler directives do more than maintain data alignment and format compatibility with MPE V/E; they also impact your ability to use Native Mode data structures which will impact program performance. Unless specified otherwise, all data elements will be in the mode of the program or as specified by the compiler directive in effect. For example, when you need to maintain Compatibility Mode data alignment and format, but you also need to create Native Mode data structures, the individual structures that must be native mode format should be defined explicitly as HP3000_32 while operating under the HP3000_16 compiler directive. The other alternative is to create a program that will read data in one mode and write it in another.

Using Native Mode Data

Programs running in Native Mode by default use Native Mode data alignment. Data files that need to be moved between MPE V/E and MPE XL must maintain 16-bit alignment.

Recompiling the Application

Migration to Native Mode involves using the Native Mode optimizing compilers. It may also involve interacting with the LinkEditor.

NOTE

The MPE V/E Segmenter cannot be used with Native Mode programs. The LinkEditor should be used for building and maintaining Native Mode program and library files.

The Native Mode compilers offer various levels of optimization. While COBOL II/XL only offers Level 0 and 1, all other Native Mode compilers offer Levels 0, 1, and 2. These levels are defined as follows:

- Level 0, no optimization.
- Level 1, partial optimization.
- Level 2, full optimization.

Each incremental level provides a higher degree of execution performance over the previous level. The optimization levels are invoked through the compiler options and directives. When using optimizing compilers, you should first compile programs with level 0 optimization and test the program before recompiling with level 1 or level 2 optimization. Refer to the appropriate MPE XL language reference manual for specific details on compiler optimization.

Program Validation

When you first run your program in Native Mode, you may want to continue to run the original version on the MPE V/E based system, or the Compatibility Mode version, to ensure that the migrated version gives the same results as the original version of the program. After recompiling from one level of optimization to another, you should retest the program to validate results.

Mechanisms to Gain Performance

While merely eliminating incompatibilities and recompiling the application using Native Mode compilers should increase the performance of an application, the final phase of your migration may include rewriting some portions of your application to make the most efficient use of MPE XL and HP Precision Architecture. The following MPE XL features may provide greater performance over MPE V/E systems:

- User mapped files.
- Larger memory.

A

Maintaining Backward Compatibility

Introduction

This appendix describes how to maintain backward compatibility for programs and files. It is especially useful for developing programs on an MPE XL-based system that may be transported to an MPE V/E-based system.

How to Ensure Compatibility

Programs can be developed in Compatibility Mode on MPE XL for use on MPE V/E-based systems. It is important to understand that you are basically working in the MPE V/E environment when you are in Compatibility Mode. Consequently, most of the features and limitations of MPE V/E are in effect. Therefore, to maintain backward compatibility, your programs should adhere to the following guidelines:

- Use only the standard language features.
- Use the intrinsic mechanism, except for calling VPLUS from COBOL programs.
- Use only User Mode programming techniques.
- Avoid all migration issues listed in Appendix D.
- Use only compatible compiler options.
- Maintain MPE V/E data alignment schemes.
- Stay within MPE V/E limitations.
- Use only MPE V/E compatible subsystems and products.
- Use compatible database products.

When changes are made to an MPE XL Compatibility Mode program, the same changes should be made to the MPE V/E program, and both programs should be validated for consistency and accuracy.

Unavoidable Problems

There are certain differences between MPE V/E and MPE XL that are unavoidable. These include:

- Different job streams and UDCs must be used for compiles.
- System usage of the stack is different. This can be a problem for very large programs on MPE V/E.

B

Using the Object Code Analyzer

Introduction

This appendix describes how to use the Object Code Analyzer (OCA). OCA is intended to be used in conjunction with its companion Migration Toolset utility, the Run-Time Monitor (RTM) during the analysis and planning stage of the migration effort. OCA runs on both MPE V/E- and MPE XL-based systems. Included in this appendix are discussions on the following subjects:

- "OCA Operation" a general overview of how the tool operates
- "Using OCA" a detailed description of the user interface
- "OCA Report Formats" sample report formats created by OCA

OCA Operation

OCA is a tool designed to aid in migrating applications from an MPE V/E-based system to an MPE XL-based system. OCA is a dialog-driven program that scans specified MPE V/E program and SL files in either batch or interactive mode and reports both definite and potential problem areas that may be encountered when migrating an application to an MPE XL-based system. OCA allows you to specify the following options prior to entering the scanning portion of the program:

- The "`Scan User Externals`" option enables you to add your own list of incompatible procedures to OCAs set of predefined system incompatibilities. This list can include either system entry points or user-written procedures which are called as externals of a program or SL being scanned.
- The "`Build Indirect File`" option enables you to construct indirect files (files that contain file specifications, including other indirect file names) from generic file sets.

These two options can be used only one time per execution of OCA. Thus, one set of options remains in effect for all program or SL files scanned. If only default values are desired, the first part can be skipped, and OCA proceeds to the scanning part of the program.

During the scanning portion of the program, you direct OCA to scan MPE V/E program and SL files and report both definite and potential problem areas that may be encountered when migrating an application to an MPE XL-based system. The scanning portion of the program can be repeated until all specified program and/or SL files are scanned.

OCA outputs a report of incompatibilities in either a brief report format or a detailed report format. The brief report format (the default) includes names of externally referenced procedures that have been identified as incompatibilities for the purposes of migration. Use the brief report format if you are interested primarily in a migration report.

When you specify a detailed report format, OCA reports on all of the program file's externally referenced procedures, as well as general information about the program's structure. Thus, the detailed report creation may take much longer to produce than the brief report, and the output volume is much greater.

During the scanning portion of the program, OCA opens each file you specify and determines if it is an SL file or a program file. If the file is a program file, OCA satisfies the program's external references before proceeding. If the file is an SL file, OCA does not satisfy external references. For either type of file, OCA compares the procedure names against OCAs predefined set of incompatible system procedures. If a detailed report is specified, then OCA reads every record of the file and determines the locations of the PCAL instructions to incompatible procedures.

NOTE

In extremely rare cases, OCA cannot distinguish between a valid PCAL and a piece of data that resembles a PCAL. Therefore, some PCALs detected and reported as incompatibles may be erroneous.

Using OCA

The Object Code Analyzer is initiated by running the program file OCA.PUB.SYS either interactively or in batch mode. You can get an explanation of the input expected for any particular prompt by entering a question mark (?). You may also issue any MPE command that can be executed programmatically by prefixing it with a colon (:). After the HELP text is displayed, or the MPE command is executed, the original prompt is redisplayed. A double slash (//) at any prompt causes OCA to terminate.

Figure B–1 charts the flow of control for OCA prompts. Additional information on the prompts is provided in subsequent headings. Defaults for all prompts are shown in upper case.

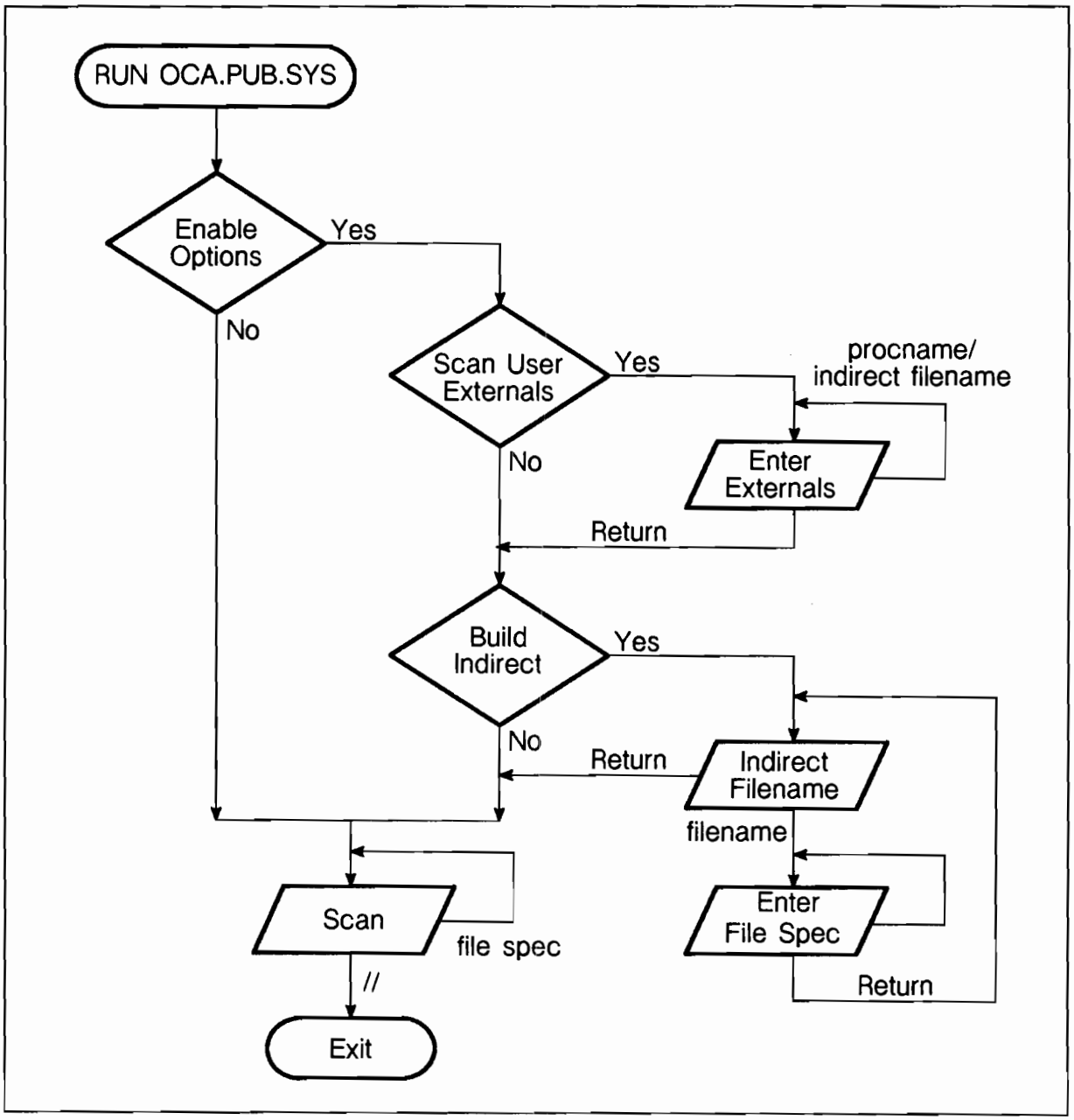


Figure B–1. Object Code Analyzer.

Enable Options Prompt

```
Do you want to enable any options (yes/NO)?
```

This is the first prompt from OCA. You are being asked if you wish to access the two optional features of OCA prior to entering the scanning portion of the program.

- A positive response takes you to the User Externals prompt.
- A negative response (the default) or a Return takes you directly to the SCAN prompt with default values used for all intervening prompts.
- A "//" terminates OCA.

Respond YES only if you need to add user-selected, externally referenced procedures or build an indirect file of file specifications identifying files you want scanned.

Scan User Externals Prompt

```
Scan for user selected external procedure references (yes/NO)?
```

OCA scans files for a set of predefined system incompatibilities. You are being asked if you have additional external procedures that you want temporarily added to the set of predefined system incompatibilities detected by OCA:

- A positive response indicates that you have additional external procedures that you want OCA to look for and takes you to the Enter External prompt.
- A negative response (the default) or a Return indicates that you want OCA only to look for its set of predefined system incompatibilities, and OCA takes you to the next option, the Build Indirect File prompt.
- A "//" terminates OCA.

When OCA scans program files, user-selected external procedure references are reported if they resolve to any SL. Predefined system incompatibilities are reported only when the procedure resolves to the system library SL.PUB.SYS. If a user-selected procedure name is identical to a predefined incompatible system procedure, then two different messages may be reported. OCA prints a message if you add an external procedure name already in the list.

Enter External Prompt

```
Enter external procedure name or indirect file name:
```

You are being prompted for the external procedures that you want added to the events detected by OCA. You can specify system entry points and user-written procedures. The object code scanner looks for references to these external procedures in addition to the set of predefined system incompatibilities that OCA scans.

Press Return when you are finished entering procedure names and wish to proceed to the next prompt.

The syntax for this prompt is as follows:

```
[ external procedure name  ]
[ ! indirect file name     ]
[ % indirect file name     ]
```

All of the external procedure names you enter at this prompt, either directly or via an indirect file, are written to a Temporary file named OCAPNAME. If you wish, you can save this file as a permanent file using the MPE V/E :SAVE command and use it as an indirect file on subsequent invocations of OCA.

External Procedure Names

OCA accepts a very general, language independent syntax for procedure names. Names are required to start with an alpha character and then can contain any combination of alpha and numeric characters plus the special characters " ' " and " _ ".

OCA returns an error message if the procedure name is syntactically incorrect, and returns the prompt again.

Because of restrictions in the MPE V/E Segmenter and the program/SL file format, procedure names are truncated to 15 characters. OCA follows this convention and issues a warning message when a name is over 15 characters in length.

External procedure names can be quoted (with either single or double quotes) or unquoted:

- Unquoted procedure names are upshifted as is the usual convention on an MPE V/E based system.
- Quoted procedure names are treated as literals and are not upshifted or downshifted.(Use double-quotes around procedure names containing imbedded single-quotes.)

Indirect File Names

As an alternative to entering the names of all your external procedures one at a time, you may specify an indirect file that contains the names of external procedures you wish added to the set of predefined system incompatibilities OCA searches for during the scanning portion of the program..

As is the case with all OCA indirect files, the file name must be preceded by either a ! or a %. The indirect file must be an EDIT/3000 compatible file, numbered or unnumbered, with the following physical characteristics:

- Standard file type (the default)
- ASCII

- Permanent, existing in the system file domain
- File code must be 0 (the default)

If you respond with an incorrect or nonexistent file name, OCA produces an error message and issues the prompt again.

The general format of this file is similar to the format of indirect files described later in this appendix, except that nested indirect files are not allowed at this prompt. The file should contain one procedure name per line (leading and trailing blanks are ignored). The conventions for specifying external procedures are described in " External Procedure Names" above. You can use the string "--" to delineate the rest of the line as a comment that is ignored by OCA. The end of line ends the comment. Blank lines are likewise ignored. In addition, only the first 72 bytes of each record are read by OCA. Everything past column 72 is ignored.

Following is an example of the file of procedure names created by OCA:

```
5     READ'TAB'1      -- the rest of this line is a comment
6     PROC1
7
8     KEEPTIME
9     UPDATE_TAB      --this is where a comment could go
10    "Abort'IO"      --a quoted procedure name
11    'AbortProg'     --another quoted procedure name
```

Build Indirect File Prompt

```
Build an indirect file of file names (yes/NO)?
```

You are being asked if you want OCA to build a file that contains the MPE file designators of the files you later want OCA to scan during the scan portion of the program.

- A positive response to this prompt causes OCA to build an indirect file you can later specify at the SCAN prompt and takes you to the Indirect File Name prompt.
- A negative response (the default) or a Return takes you to the SCAN prompt.
- A "//" terminates OCA.

This "Build Indirect File" feature can be useful if you have a large number of files to scan that are centralized in a few key groups and accounts on the system, and you want to save the set of file names for future use. A second use of this feature is generating a list of the names of programs and SL files to be considered as candidates for migration.

Indirect File Name Prompt

```
Enter name for indirect file:
```

You are being prompted for the name of the file you want to associate with the indirect file you are about to create. This file will contain the names of files that you later want OCA to scan during the scanning portion of the program. Valid responses to this prompt are:

- A file name following MPE V/E file naming conventions.
- A [Return] takes you to the Scan prompt.
- A "//" terminates OCA .

If you specify a valid MPE file name and press [Return], OCA creates and opens a new file with the name you specified at the prompt, then proceeds to the "Enter File Specification" prompt (described below).

Enter File Specification Prompt

```
Enter file specification:
```

You are being prompted for the information to include in the indirect file you just created. The indirect file is used to specify the files to be scanned in response to the SCAN prompt, and is helpful in identifying program and SL files from generic file sets. Each line of the indirect file will be treated by the SCAN prompt as if you had entered it interactively (in other words, as a command image). Thus, the indirect file can contain any valid response to the SCAN prompt.

After OCA has opened the indirect file, it repeatedly prompts you for files or file sets to be scanned. The file set specifications may include wild cards. OCA automatically selects all program and SL files residing in the file set. The fully qualified names of individual program and SL files are written to the indirect file.

Press [Return] when you are finished entering file specifications and wish to proceed to the next prompt. Once you have entered the contents of the indirect file, OCA attempts to close the indirect file as a permanent file.

If a file of the same name already exists in the permanent file directory, OCA asks you if the old file should be purged. If you respond YES, OCA purges the old file before it closes the indirect file.

If you respond NO to the purge request (or press [Return]), OCA asks if you want to save the file under a different name. If you respond NO to this prompt (or press [Return]), the indirect file you just built is lost. If you respond YES, you are prompted for a different name for the indirect file name. If you press [Return] in response to the prompt for a different name, the indirect file you just built is lost.

In both cases where the file is lost, OCA returns you to the Indirect File Name prompt and allows you to create a new indirect file. When you succeed at saving the indirect file, OCA takes you to the SCAN prompt.

The syntax for the "Enter File Specification" prompt is:

file specification |*library specification*| |*report level*| |*report destination*|

The syntax for the required *file specification* is:

```
{ filename            }
{ fileset             }
{ !indirect filename  }
{ #indirect filename  }
```

The syntax for the optional *library specification* is:

```
            { GROUP  }
[; LIB = { PUBLIC } ]
            { SYSTEM }
```

The syntax for the optional *report level* is:

```
[; DETAILED ]
[; BRIEF    ]
```

The syntax for the optional *report destination* is:

```
[; OFFLINE ]
```

As a convention OCA keywords are shown in upper case, although you do not have to enter them in upper case on the command line. In addition, only the minimum number of characters necessary to recognize the keyword must be specified. For example, ;DETAILED can be entered as ;DET or as ;D.

NOTE

Each line of the indirect file is treated as a command image at the SCAN prompt. Therefore, the syntax and meaning of the options listed above are identical to the optional parameters available through the SCAN prompt.

Details of the *file specification*, *library specification*, *report level*, and *report destination* parameters are discussed below.

File Specifications

OCA scans only MPE V/E format SL or program files. You can specify the files you want placed in the indirect file in the following ways:

- By file name
- By file set
- By indirect file name

By File Name. You can enter the name of a single program or SL file. OCA displays an error message if the file is not of the correct type. Program and SL files are treated slightly differently, the differences are reflected only in the detailed report.

By File Set. You can enter a file set. You can select groups of program and/or SL files using MPE wild card characters (@, #, ?) in conjunction with file designators. OCA search the specified file set and include in the indirect file the fully qualified file designators of only the program and SL files located in the specified file set. For example, if you specify `@.mygroup.myacct`, OCA places in the indirect file all program and SL files located in group `mygroup`, account `myacct`.

NOTE

The *file set* specification can contain MPE wild card characters (@, #, ?) except that OCA interprets the single question mark followed by a space or Return as a request for the OCA HELP facility instead of as a wild card character. Therefore, if you want to specify all single- character file names, you must use a more qualified file specification, for example:

```
?.mygroup
?.mygroup.myacct
```

By Indirect File Name. You can enter the name of an indirect file that you have previously created either by the `Build Indirect File` option of OCA or from an editor.

As is the case with all OCA indirect files, the file name must be preceded by either a ! or a %. The indirect file must be an EDIT/3000 compatible file, numbered or unnumbered, with the following physical characteristics:

- Standard file type (the default)
- ASCII
- Permanent, existing in the system file domain
- File code must be 0 (the default)

If you respond with an incorrect or nonexistent file name, OCA produces an error message and issues the prompt again.

The general format of this file is similar to the format of indirect files described earlier in this appendix, except that indirect files of file specifications can be nested up to five levels deep. The file should contain one command per line (following the syntax described above). Leading and trailing blanks are ignored. You can use the string "--" to delineate the rest of the line as a comment that is ignored by OCA. The end of line ends the comment. Blank lines are likewise ignored. In addition, only the first 72 bytes of each record are read by OCA (everything past column 72 is ignored).

Following is an example of the contents of an indirect file of file specifications:

```
1      EOMRPT;LIB=G
2      SCRNDVR.PUB.DRIVERS  --comment to end of line
3      @.PUB.OURSYS;DETA
4      ZEDIT;BRIEF;OFFLINE
5      !INDIRECT.PUB.MYACCT;BRIEF  --an indirect file
```

NOTE

The additional options you specify with the *indirect filename* option will override whatever options that may have been specified within that indirect file. In the example above, specification of the ;BRIEF option for the indirect file !INDIRECT.PUB.MYACCT overrides any specifications for a detailed report format that may be made within !INDIRECT.PUB.MYACCT.

LIB= Option

The LIB= option tells OCA what system libraries the loader would search if the specified program file were to be loaded. It is interpreted in exactly the same way as the LIB= option on the MPE V/E :RUN command. The default library indicator is "S". Specifying LIB=S is equivalent to omitting the LIB= parameter entirely.

The LIB= option should only be specified with program files. If you specify LIB with an SL file, the LIB= parameter is ignored.

When the LIB= option is used with a file set or with an indirect file, the specified library is applied to those files in the file set (or in the indirect file) that are program files and is ignored for the SL files.

If you request that a file is to be scanned LIB=G or LIB=P and a group or account SL does not exist, OCA does not print an error message. This is consistent with the behavior of the MPE V/E :RUN command. However, the OCA detailed report notes when an SL is not required.

OCA always tells you which SL file the external procedure will actually be resolved to when the program is loaded. It also tells you if you specified a group or account SL and none of the externals resolved to the group or account SL specified. This information is useful for determining which SLs are needed (and not needed) for migration.

NOTE

If you request that an SL file is to be scanned, OCA scans the entire SL file. If you request that a program file be scanned with the LIB= option, OCA scans only the segments that contain external procedures directly referenced by the program, or indirectly referenced by externals of SL segments.

Brief and Detailed Options

By default, OCA produces a report on *file specification* in a brief format that provides only information directly related to migrating the program file successfully, including names of external procedures that have been identified as incompatibilities for the purposes of migration. Specifying ;BRIEF with a *file name* or *file set* is the same as not specifying ;BRIEF.

You can optionally specify ;DETAILED if you want much greater detail about all of *file specification's* externally referenced procedures. In addition, the ;DETAILED option produces general information about the program structure of *file specification*.

Specifying either the ;BRIEF or the ;DETAILED option with an indirect file overrides all uses of this option that may be found within that indirect file.

Offline Option

By default, OCA output is directed to the job/session list device $STDLIST. If you use the ;OFFLINE option, you direct OCA output to the OCA list file OCALIST. By default this file is assigned to the system line printer (DEV=LP). If you want to redirect output to a different device, you can create an MPE file equation using the :FILE command to redefine the characteristics for OCALIST. This file equation is invoked for *file specification* only if you specify ;OFFLINE. In the following example, the :FILE command is invoked at the SCAN prompt to create a file equation for OCALILST, then a file name is specified with the ;OFFLINE option:

```
SCAN>:FILE OCALIST;DEV=EPOC
SCAN>@.TOOLS.SWAT;OFFLINE
SCAN>
```

SCAN Prompt

```
SCAN>
```

You are being prompted for the name of the file(s) you want OCA to scan for incompatibilities.

Valid responses to this prompt are:

- A file specification (with optional keyword parameters), following the syntax description described below
- A [Return] returns you to the SCAN prompt
- A "//" terminates OCA

A report is generated for each file scanned, and you are prompted for the name of the next file to scan. If a file set has been specified, OCA displays progress messages indicating the start of the directory search and the number of program and SL files found. OCA always displays the name of the file being scanned.

Normally, OCA automatically checks the files against a set of predefined system incompatibilities located in the file OCAINCOM.PUB.SYS and, optionally, a set of user-selected incompatibilities created by the Build Indirect File option of OCA. If you want OCA to scan only for user-selected incompatibilities, you can temporarily disable access to OCAINCOM.PUB.SYS by using a file equation to equate OCAINCOM.PUB.SYS to $NULL prior to executing OCA. You can later invoke the :RESET command to enable OCAINCOM.PUB.SYS. Following is a sequence of SCAN prompt entries where this is accomplished:

```
:FILE OCAINCOM.PUB.SYS=$NULL
:RUN OCA.PUB.SYS
      .
      .
      .
SCAN>//
:RESET OCAINCOM.PUB.SYS
```

The syntax for the SCAN prompt is:

file specification [*library specification*] [*report level*] [*report destination*]

The syntax for the required *file specification* is:

```
{ filename          }
{ fileset           }
{ !indirect filename}
{ #indirect filename}
```

The syntax for the optional *library specification* is:

```
          { GROUP  }
[; LIB = { PUBLIC } ]
          { SYSTEM }
```

The syntax for the optional *report level* is:

```
[; DETAILED ]
[; BRIEF    ]
```

The syntax for the optional *report destination* is:

```
[; OFFLINE ]
```

As a convention OCA keywords are shown in upper case, although you do not have to enter them in upper case on the command line. In addition, only the minimum number of characters necessary to recognize the keyword must be specified. For example, ;DETAILED can be entered as ;DET or as ;D.

If you specify the file name EXIT (or any of it's substrings EXI, EX, E), and a file by that name does not exist in the current logon group, OCA terminates. To be safe, qualify EXIT (or its substring) with the group name (for example, EXI.MYGROUP).

Details on the *file specification*, *library specification*, *report level*, and *report destination* parameters are discussed below.

File Specifications

OCA scans only MPE V/E format SL or program files. You can specify the files you want scanned in the following ways:

- By file name
- By file set
- By indirect file name

By Filename. You can enter the name of a single program or SL file. OCA displays an error message if the file is not of the correct type. Program and SL files are treated slightly differently, the differences are reflected only in the detailed report.

By File Set. You can enter a file set, selecting groups of program and/or SL files using MPE wild card characters (@, #, ?) in conjunction with file designators. OCA searches the specified file set and scans only the program and SL files located in the specified file set. For example, if you specify @.mygroup.myacct, OCA scans all program and SL files located in group mygroup, account myacct.

NOTE

The *fileset* specification can contain MPE wild card characters (@, #, ?) except that OCA interprets the single question mark followed by a space or [Return] as a request for the OCA HELP facility instead of as a wild card character. Therefore, if you want to scan single-character file names you must use a more qualified file specification, for example:

```
?.mygroup
?.mygroup.myacct
```

By Indirect File Name. You can enter the name of an indirect file that you have previously created either by the `Build Indirect File` option of OCA or from an editor. As is the case with all OCA indirect files, the file name must be preceded by either a ! or a %. The indirect file must be an EDIT/3000 compatible file, numbered or unnumbered, with the following physical characteristics:

- Standard file type (the default)
- ASCII
- Permanent, existing in the system file domain
- File code must be 0 (the default)

If you respond with an incorrect or nonexistent file name, OCA produces an error message and issues the prompt again.

The general format of this file is similar to the format of indirect files described earlier in this appendix; indirect files of file specifications can be nested up to five levels deep. The file should contain one file specification per line (following the syntax described above). Leading and trailing blanks are ignored. You can use the string "--" to delineate the rest of the line as a comment that is ignored by OCA. The end of line ends the comment. Blank lines are likewise ignored. In addition, only the first 72 bytes of each record are read by OCA (everything past column 72 is ignored).

Following is an example of the contents of an indirect file of file specifications:

```
1     EOMRPT;LIB=G
2     SCRNDVR.PUB.DRIVERS  --comment to end of line
3     SCRAMBLE.PUB.OURSYS
4     ZEDIT;BRIEF;OFFLINE
5     !FILESPECS.PUB.MYACCT;DETAILED  --an indirect file
```

NOTE

The additional options you specify with the indirect file name option at the SCAN prompt will override whatever options that may have been specified within that indirect file. For example, if you specify at the SCAN prompt the indirect file described above with the ;BRIEF option, the contents of the file set @.PUB.OURSYS and the contents of the indirect file !FILESPECS.PUB.MYACCT will be reported in the brief report format.

LIB= Option

The LIB= option tells OCA what system libraries would be searched if the program file were to be loaded. It is interpreted in exactly the same way as the LIB parameter on the MPE V/E :RUN command. The default library indicator is "S". Specifying LIB=S is equivalent to omitting the LIB= parameter.

The LIB= option should only be used if scanning a program file. If you specify LIB= with an SL file, the LIB= parameter is ignored.

When the LIB= option is used with either a file set or an indirect file, the specified library is applied to those files in the file set (or in the indirect file) that are program files, and ignored for the SL files.

If you request that a file is to be scanned LIB=G or LIB=P and a group or account SL does not exist, OCA does not print an error message. This is consistent with the behavior of the MPE :RUN command. However, the OCA detailed report notes when an SL is not required.

OCA always tells you which SL file the external procedure will actually resolve when the program is loaded. It also tells you if you specified a group or account SL and none of the externals would resolve to the group or account SL specified. This information is useful for determining which SLs are needed (and not needed) for migration.

NOTE

If you request that an SL file is to be scanned, OCA scans the entire SL file. If you request that a program file be scanned with the LIB parameter, OCA scans only the segments that contain external procedures directly referenced by the program, or indirectly referenced by externals of SL segments.

Brief and Detailed Options

By default, OCA produces a report on *file specification* in a brief format that provides only information directly related to the ability of the program file to migrate successfully, including names of external procedures that have been identified as incompatibilities for the purposes of migration. Specifying ;BRIEF with a file name or file set is the same as not specifying ;BRIEF.

You can optionally specify ;DETAILED if you want much greater detail about all of *file specification's* externally referenced procedures. In addition, the ;DETAILED option produces general information about the program structure of *file specification.*

Specifying either the ;BRIEF or the ;DETAILED option with an indirect file overrides all uses of this option that may be found within that indirect file.

Offline Option

By default, OCA output is directed to the job/session list device $STDLIST. If you use the ;OFFLINE option, you direct OCA output to the OCA list file OCALIST. By default this file is assigned to the system line printer (DEV=LP). If you want to redirect output to a different device, you can create an MPE file equation using the :FILE command to redefine the characteristics for OCALIST. This file equation is invoked for *file specification* only if you specify ;OFFLINE. In the following example, the :FILE command is invoked at the SCAN prompt to create a file equation for OCALILST, then a file name is specified with the ;OFFLINE option:

```
SCAN>:FILE OCALIST;DEV=EPOC
SCAN>@.TOOLS.SWAT;OFFLINE
SCAN>
```

Interactive Pagination

When using OCA interactively and the report fills one page on your screen, OCA displays the More prompt. OCA does not display the More prompt when run from batch mode.

```
More (YES/no/all)?
```

The following responses are valid to the More prompt:

- YES (the default) tells OCA that you want to continue listing the text. OCA will display the More prompt each time it fills one page on your screen.
- NO tells OCA that you do not want to see the rest of the text. OCA stops scanning any other files specified, and returns you to the "Scan" prompt.
- ALL tells OCA that you want to see all of the report without intervening More prompts.

OCA uses two session level job control words (JCWs) to control page length for output (help text or reports). If OCA determines that the output destination is offline then OCA will use the JCW OCAPAGESIZE to determine the number of lines per page. This would be the case if you specified ;OFFLINE on the command line, or OCA was being run from a :JOB. If the output destination is your terminal screen, OCA will use the OCASCREENSIZE JCW. If these JCWs do not exist or are set to values out of range, OCA will use default values for the page size. If the appropriate JCW is set to zero, then the paging mechanism is disabled.

Below are the default values and legal ranges for each of the JCWs described above:

- For JCW OCASCREENSIZE, default value is 23, legal range is 0, 1..100
- For JCW OCAPAGESIZE, default value is 60, legal range is 0, 5..100

Exiting OCA

Entering a double slash (//) at any prompt terminates OCA. Additionally, if you specify EXIT (or any of it's substrings EXI, EX, E) at the SCAN prompt, or if EXIT is encountered in an indirect file, and a file by that name does not exist in the current logon group, OCA terminates.

Security Considerations

There are no special restrictions on who can run OCA. However, OCA places restrictions on which files may be scanned. You must have READ access to the file to scan it. In addition, you must provide lockwords for files requiring them. Users possessing Account Manager capability may scan any file residing in their accounts without specifying the lockword. Similarly, System Manager capability allows all files on the system to be scanned without specifying the lockword. Access restrictions, however, may prevent you from scanning a file despite your capabilities. In most cases, if you can run the program, you can scan it.

Running OCA in Batch Mode

Figure B-2 illustrates how you can use a job stream file to invoke OCA in batch mode. Figure B-2 shows all the major operations available through OCA.PUB.SYS. Note how MPE V/E commands are executed and then control is returned to the same OCA prompt. In addition, the string "--" indicates to OCA that the remainder of the input line is a comment. Because OCA deletes these comments from the input line before parsing occurs, the comments can be included on MPE V/E command execution lines (see OCA invocation of :FILE command).

```
:job ocajob, manager.sys
:run oca.pub.sys
y                       -- want to enable some options
y                       -- want to add user selected externals
procedure1              -- enter the names of some procedures you
procedure2              -- want to add to predefined list of system
procedure3              -- incompatibilities
procedure4
procedure5
                        -- blank line indicates no more procedure names
:comment save the procedure names entered above for later use
:save ocapname
y                       -- want to build an indirect file
indirect                -- call the file indirect
@.@.finance             -- get all programs/SLs in finance account
@.tools.@;lib=g         -- get all tools groups in all accounts ;LIB=G
                        -- blank line to indicate no more filesets
                        -- blank line to indicate no more indirect files

-- We are at the Scan> prompt now, blank lines or comment have no effect.
-- We are going to do a :showtime before and after we generate the
-- report just so we can tell how long it took.
:showtime
:file ocalist;dev=epoc     --direct output to epoc printer
!indirect;brief;offline    --scan the indirect file created above
:showtime
//                         --terminates OCA
:eoj
```

Figure B-2. Job Stream to Execute OCA.

OCA Report Formats

A report is automatically generated for each file that OCA scans. There are two formats of the report:

- Brief format
- Detailed format

The brief report format provides only information directly related to the ability of the program or SL file to migrate successfully. This information includes a list of potential incompatibilities detected when OCA scanned the selected program or SL files.

NOTE

It is highly recommended that you run the brief report first. Run the detailed report only on programs which have potential incompatibilities as identified in the brief report, or if the additional information is needed.

The detailed report format provides information about the program in addition to the information provided in the Brief Report format.

Another purpose of the detailed report is to obtain the code location of potential incompatibilities. This should be done only on programs which have incompatibilities as reported by the OCA brief report (and not disqualified by RTM). A detailed report can be used to locate privileged segments or provide a better understanding of the program's structure. The content of this report differs slightly depending on whether the file is a program file or an SL file.

If you have the source file, following are two alternatives to using the detailed report to obtain the code location of incompatibilities. After you have used the OCA brief report format to identify the names of incompatible external procedures:

- Use an editor to find the exact location in your source file of the call to the incompatible procedure, or
- Use the compiler listing and a cross reference to find the exact location of the call to the incompatible procedure.

Output Device Specification

By default, OCA report and HELP text output is directed to the job/session list device $STDLIST. If you want to redirect output to a different device, you can create a file equation using the :FILE command to redefine $STDLIST.

In addition, if you use the ;OFFLINE option, you can direct OCA report or HELP text output to the OCA list file OCALIST. By default this file is assigned to the system line printer (DEV=LP). If you want to redirect output to a different device, you can create an MPE file equation using the :FILE command (either at the SCAN prompt or prior to running OCA) to redefine the characteristics for OCALIST. This file equation is invoked for file specification only if you specify ;OFFLINE. In the following example, the :FILE command is invoked at the SCAN prompt to create a file equation for OCALIST, then a file name is specified with the ;OFFLINE option:

```
SCAN>:FILE OCALIST;DEV=EPOC
SCAN>@.TOOLS.SWAT;OFFLINE
SCAN>
```

Contents of Brief and Detailed Report Formats

The following tables summarize the types of information returned in the brief and detailed reports created by OCA. The tables mirror the sequence of sections of the OCA report. Sections printed only for program files are noted in the table description:

- General Information, returned for program files only
- Segment Information
- Resolved External Procedures
- Unresolved External Procedures returned for program files only
- Potential Incompatibilities
- Intrinsic Mechanism Information

General Information

Table B-1 summarizes the information returned in the "General Information" section of an OCA report. This information is returned only for program files; segmented library (SL) files do not have equivalent characteristics. The first two categories of information indicate potential migration issues. You should pay particular attention to whether or not the program file was prepped with Privileged Mode (PM) capability. In this case, check to see if your program file contains privileged mode segments. In general, use of privileged mode code in an application is much more likely to cause migration related problems.

Table B-1. General Information (program files only)

Information	Brief	Detailed
Program capabilities (for example IA, BA, DS, PM)	X	X
Are there privileged segments?	X	X
Initial stack info: size of program global area (DB-QI)		X
Initial stack info: size of initial DL area (DL-DB)		X
Initial stack info: size of initial stack (QI-Z)		X
Initial stack info: program MAXDATA size (DL-Z)		X
Was a program prepped with ;FPMAP option		X

Segment Information

Table B–2 summarizes the information returned in the "Segment Information" section of an OCA report. The "Segment Information" section returns information about all program/SL segments, as well as information about possible patch areas if the program/SL file is patched. If the "General Information" section of an OCA report indicates that the program file contains privileged mode segments, the "Segment Information" section is useful for determining which program/SL segments are privileged.

Table B–2. Segment Information

Information	Brief	Detailed
Table containing the segment name (SL files only), segment number, segment length, and mode (User or Priv)		X
Summary information: total code size		X
Summary information: average code segment size		X
Patch area information: program name (program file patch areas only)		X
Patch area information: segment name		X
Patch area information: prep date/time		X
Patch area information: patch data/time		X
Patch area information: whether or not the patch area has been altered		X
Patch area information: an octal dump of the patch area (only if the patch area has been altered)		X

Resolved External Procedures

Table B-3 summarizes the information returned in the "Resolved External Procedures" section of an OCA report. The "Resolved External Procedures" section of the report provides a complete list of all external procedures called by a program directly or indirectly. This section also provides a cross reference of all the calls. External procedure calls originating from a segment in the system library SL.PUB.SYS will not be in the list. This section may be very long for large programs and for programs that use SL files. Hewlett-Packard recommends that you request this information only if you need to "get a feel" for what a program is doing, and how it does it, for example, if you are migrating a program you did not write.

Table B-3. Resolved External Procedures

Information	Brief	Detailed
The name of the external procedure		X
Library in which the external procedure resolved (Group, Pub, System)		X
Segment number of the segment(s) calling the procedure		X
SST number(s) for the PCAL to the procedure		X
PB relative offsets for the calls to the procedure		X

Unresolved External Procedures

Table B-4 summarizes the information returned in the "Unresolved External Procedures" section of an OCA report. The "Unresolved External Procedures" section is printed only if the program file contains one or more unresolved external procedure references. This usually occurs when you do not specify the correct library at the SCAN prompt using the LIB= option. For best results, you should scan the program file using the same LIB= option as you would when you run the program.

Table B-4. Unresolved External Procedures (program files only)

Information	Brief	Detailed
A list of the external procedures which were unresolved	X	X
The reason why the procedure was unresolved	X	X

Potential Incompatibilities

Table B-5 summarizes the information returned in the "Potential Incompatibilities" section of an OCA report. The "Potential Incompatibilities" section lists the potential migration issues you should expect to encounter. You should read the messages encountered in this section, and refer to Appendix D for a more detailed explanation of the cause of the incompatibility, as well as possible corrective actions to take to remedy the problem.

It is possible that the OCA report may list calls to system intrinsics that you will be unable to locate in your code. This should not alarm you. Many MPE V/E compilers generate calls to system intrinsics on your behalf. For example, the COBOL II/V compiler generates a call to the system intrinsic XCONTRAP if you compile with the $CONTROL DEBUG option enabled. When you recompile your source code using an MPE XL Native Mode compiler, all compiler-generated external references will be properly handled by the compiler.

Table B-5. Potential Incompatibilities

Information	Brief	Detailed
A list of all the incompatibilities detected	X	X
The name of the incompatible procedure that is referenced	X	X
A brief message indicating the nature of the incompatibility	X	X
An indication of the mode in which a user would expect to encounter the problem (ie: CM only, CM/NM, NM only)	X	X
A message number which can be cross referenced with a number from RTM and looked up in Appendix D	X	X

Summary Information

Table B-6 summarizes the information returned in the "Summary Information" section of an OCA report. The "Summary Information" section reports total numbers of events listed in the "Potential Incompatibilities" section of the report.

Table B-6. Summary Information

Information	Brief	Detailed
Total number of CM only incompatibilities detected	X	X
Total number of CM/NM incompatibilities detected	X	X
Total number of NM only incompatibilities detected	X	X
Total number of user-defined incompatibilities detected	X	X
Total number of uncallable procedures detected	X	X

Intrinsic Mechanism Information

Table B-7 summarizes the information returned in the "Intrinsic Mechanism Information" section of an OCA report. The "Intrinsic Mechanism" section briefly describes the intrinsic mechanism issue related to migration, and lists all of the names of the system intrinsics your program references. Using this list, you can consult your source code to determine if you actually used the intrinsic mechanism. The intrinsic mechanism issue related to migration is discussed below.

Although you may not recognize it by the name intrinsic mechanism, the intrinsic mechanism has been an MPE V/E feature for a long time. All of the programming languages supported on MPE V/E allow users to call system supplied routines, called intrinsics, without explicitly declaring the intrinsic parameters. The compilers determine if you are calling the procedure correctly by examining the file SPLINTR.PUB.SYS which contains the declarations for all the system intrinsics.

The MPE V/E intrinsic mechanism is provided as a convenience for users. It is a shorthand notation for calling external procedures, but it does not cause the compiler to generate code any differently than if the procedure is declared as an external procedure and called as in the example above. MPE XL also supports a set of Native Mode compilers which handle the intrinsic mechanism in a slightly different manner.

On MPE XL, the intrinsic mechanism plays a more important part in the compilation process. The intrinsic file on MPE XL, SYSINTR.PUB.SYS contains more information about the procedure which is being called. For example, it contains information about which mode the procedure executes in (Compatibility Mode or Native Mode) and about how it can be accessed via the Switch subsystem. Furthermore, it contains crucial information about the parameter alignment necessary for calling the procedure.

If you plan to migrate your application to Native Mode on MPE XL, you must use the intrinsic mechanism to get correct results. By mandating the use of the intrinsic mechanism and enhancing it to contain several new features, MPE XL will be able to provide a programmatic interface which is more flexible and extensible than the one currently provided by MPE V/E.

Each of the MPE V/E programming language reference manuals contain a section on how to call system intrinsics. These manuals can be consulted to determine the appropriate syntax for the language you are using. If you are already using the intrinsic mechanism (not explicitly declaring the procedures or allowing the compiler to do it implicitly), you will not encounter any problems in this area when migrating to Native Mode on MPE XL.

There will be other migration guides that detail language-specific concerns. These guides will provide you with additional information on the subject of the intrinsic mechanism and how to deal with any potential issues which may arise when recompiling to Native Mode. Some of the MPE XL Native Mode compilers support special migration related compiler options which can be of great assistance in determining if you have a problem in this area. COBOL II/XL, for example, supports a `$CONTROL CALLINTRINSIC` option which will generate a warning if your program contains calls to system intrinsics without using the intrinsic mechanism.

It is possible that the OCA report may list calls to system intrinsics that you will be unable to locate in your code. This should not alarm you. Many of the MPE V/E compilers will generate calls to MPE V/E system intrinsics on your behalf. For example, most of the compilers will generate a call to the system intrinsic TERMINATE which handles terminating your process when your program is done executing. Any procedure names which you find in the list generated by OCA but cannot find references to in your source code fall into this category.

When you recompile your source code on MPE XL using Native Mode compilers, all compiler-generated external references to MPE XL system intrinsics are handled through the intrinsic mechanism.

Table B-7. Intrinsic Mechanism Information

Information	Brief	Detailed
A message briefly explaining the intrinsic mechanism issue	X	X
A list of all the intrinsics that the application references	X	X

Sample Brief Report

The following lists a sample brief OCA report.

```
OCA HP30365A.00.00 (Cat A.00.00) - THU, OCT 23, 1986, 11:01 AM  Page No.  1
Report for STICP.SOOL.MPEV ;LIB=G ;BRIEF

                    GENERAL  INFORMATION

Program was prepped with ;CAPS=BA,IA.
Program contains only user mode segments.

                    POTENTIAL  INCOMPATIBILITIES

Incompatibilities detected in the program file "STICP.SOOL.MPEV".

FCONTROL:  *** System Defined Incompatibility ***
    - Some FCONTROL controlcodes are no longer valid
      on MPE XL (CM/NM 500).
      These control codes include:  03, 48.

Incompatibilities detected in the group SL (SL.SOOL.MPEV).

CREATEPROCESS:  *** System Defined Incompatibility ***
    - CREATEPROCESS may encounter byte pointer problems
      in NM on MPE XL (NM 107).

FCONTROL:  *** System Defined Incompatibility ***
    - Some FCONTROL controlcodes are no longer valid
      on MPE XL (CM/NM 500).
      These controlcodes include:  03, 48.

                    SUMMARY  INFORMATION

NM Only Incompatibilities:           1
CM/NM Incompatibilities:             2
Uncallable Procedures Detected:      0

                    INTRINSIC  MECHANISM  INFORMATION

Applications being recompiled to native mode must
use the intrinsic mechanism for system intrinsics.

Number of procedures requiring the intrinsic mechanism:  3


Procedures from STICP.SOOL.MPEV requiring the intrinsic mechanism:
FCONTROL

Procedures from SL.SOOLMPEV requiring the intrinsic mechanism:
CREATPROCESS       FCONTROL
```

Sample Detailed Report

The following lists a sample detailed OCA report.

```
OCA HP30365A.00.00 (Cat A.00.00) - THU, OCT 23, 1986, 11:01 AM  Page No.  1
Report for STICP.SOOL.MPEV ;LIB=G ;DETAILED

        G E N E R A L   I N F O R M A T I O N

Program was prepped with ;CAPS=BA,IA.
Program contains only user mode segments.

Program was prepped with ;CAPS=BA,IA.
Program contains only user mode segments.

Initial Stack Information:
    Size of initial DL area (DL-DB):           0  (%000000)
    Size of program global area (DB-QI):     284  (%000434)
    Size of initial stack (QI-Z):           4256  (%010240)
    ;MAXDATA= was not specified on :PREP command.

        S E G M E N T   I N F O R M A T I O N


Segment information for the program file "STICP.SOOL.MPEV".
Total number of segments: 1
Starting segment number:  0

 Segment           Segment
 Number            Length (%octal)          Mode
 ================================================
    0                  76 (%00114)           USER


 Total code segment size: 76   Average code segment size: 76


 Information for referenced segemnts in the group SL (SL.SOOL.MPEV).
 Number of segments referenced: 1


 Segment            Segment        Segment
 Name               Number         Length (%octal)        Mode
 ============================================================
 STICK                 0               72 (%00110)        USER


 Total code segment size: 72   Average code segment size: 72


 Information for referenced segments in the account SL (SL.PUB.MPEV).
 No externals resolved to this SL.
 NOTE :  This application does not require this SL to migrate.


 Entry Point       PB Relative Address     Stt Number
 ================================================
 OB'                     %000013            %000001
```

RESOLVED EXTERNAL PROCEDURES

External references of the program "STICP.SOOL.MPEV".

Procedure Name	Called At Seg Num	STT Num	Resolved To LIB
FCONTROL	0	8	SYSTEM

**** OFFSETS ****

%000061

Total Number of calls in segment 0: 1

Total number of calls to FCONTROL: 1
Number of segments which call FCONTROL: 1

Procedure Name	Called At Seg Num	STT Num	Resolved To LIB
P'CLOSEIO	0	5	SYSTEM

**** OFFSETS ****

%000072

Total Number of calls in segment 0: 1

Total number of calls to P'CLOSEIO: 1
Number of segments which call P'CLOSEIO: 1

Procedure Name	Called At Seg Num	STT Num	Resolved To LIB
P'INITHEAP'3000	0	6	SYSTEM

**** OFFSETS ****

%000016

Total Number of calls in segment 0: 1

Total number of calls to P'INITHEAP'3000: 1
Number of segments which call P'INITHEAP'3000: 1

Procedure Name	Called At Seg Num	STT Num	Resolved To LIB
P'RESET	0	3	SYSTEM

**** O F F S E T S ****

%000052

Total Number of calls in segment 0: 1

Total number of calls to P'RESET: 1
Number of segments which call P'RESET: 1

Procedure Name	Called At Seg Num	STT Num	Resolved To LIB
P'REWRITE	0	4	SYSTEM

**** O F F S E T S ****

%000035

Total Number of calls in segment 0: 1

Total number of calls to P'REWRITE: 1
Number of segments which call P'REWRITE: 1

Procedure Name	Called At Seg Num	STT Num	Resolved To LIB
PROCA	0	7	GROUP

**** O F F S E T S ****

%000067

Total Number of calls in segment 0: 1

Total number of calls to PROCA: 1
Number of segments which call PROCA: 1

Procedure Name	Called At Seg Num	STT Num	Resolved To LIB
TERMINATE'	0	2	SYSTEM

**** O F F S E T S ****

%000073

Total Number of calls in segment 0: 1

Total number of calls to TERMINATE': 1
Number of segments which call TERMINATE': 1

External references inherited from the group SL (SL.SOOL.MPEV).

	Called At		Resolved To
Procedure Name	Seg Num	STT Num	LIB
CREATEPROCESS	0	2	SYSTEM

**** O F F S E T S ****

%000072

Total Number of calls in segment 0: 1

Total number of calls to CREATEPROCESS: 1
Number of segments which call CREATEPROCESS: 1

	Called At		Resolved To
Procedure Name	Seg Num	STT Num	LIB
FCONTROL	0	3	SYSTEM

**** O F F S E T S ****

%000032

Total Number of calls in segment 0: 1

Total number of calls to FCONTROL: 1
Number of segments which call FCONTROL: 1

P O T E N T I A L I N C O M P A T I L I T I E S

Incompatibilities detected in the program file "STICP.SOOL.MPEV".

FCONTROL: *** System Defined Incompatibility ***
- Some FCONTROL controlcodes are no longer valid
  on MPE XL (CM/NM 500).
  These controlcodes include: 03, 48.

Report for STICP.SOOL.MPEV ;LIB=G ;DETAILED

Incompatibilities detected in the group SL (SL.SOOL.MPEV).

```
CREATEPROCESS: *** System Defined Incompatibility ***
  - CREATPROCESS may encounter byte pointer problems
    in NM on MPE XL (NM 107).

FCONTROL: *** System Defined Incompatibility ***
  - Some FCONTROL controlcodes are no longer valid
    on MPE XL (CM/NM 500
    These controlcodes include: 03,48.
```

SUMMARY INFORMATION

```
NM Only Incompatibilities:        1
CM/NM Incompatibilities:          2
Uncallable Procedures Detected:   0
```

INTRINSIC MECHANISM INFORMATION

Applications being recompiled to native mode must use the intrinsic mechanism for system intrinsics.

Number of procedures requiring the intrinsic mechanism: 3

Procedures from STICP.SOOL.MPEV requiring the intrinsic mechanism:
FCONTROL

Procedures from SL.SOOL.MPEV requiring the intrinsic mechanism:
CREATEPROCESS FCONTROL

C

Using the Run Time Monitor

Introduction

This appendix describes how to use the Run Time Monitor. Run Time Monitor is intended to be used in conjunction with its companion Migration Toolset utility, the Object Code Analyzer (OCA) during the analysis and planning stage of the migration effort. Run Time Monitor runs on MPE V/E-based systems. Included in this appendix are discussions on the following subjects:

- Run Time Monitor Operation, a general overview on how the tool operates.
- Setting up Run Time Monitor, how to install RTM on your MPE V/E-based system.
- Using RTMSYS, how to enable the RTM event monitoring facility.
- Using RTMREP, how to generate RTM event reports.

Run Time Monitor Operation

Run Time Monitor is a tool designed to aid in migrating applications from an MPE V/E-based system to an MPE XL-based system. The task of Run Time Monitor is to monitor executing applications for occurrences of pre-defined events. An event is defined as an MPE V/E intrinsic feature that will be incompatible when executed on an MPE XL-based system.

Run Time Monitor is comprised of three parts:

- A controlling program, `RTMSYS.PUB.SYS`, you use to enable Run Time Monitor and specify the event classes you want to monitor.
- A report program, `RTMREP.PUB.SYS`, you use to generate reports for monitored applications.
- An SL, `RTMSL.PUB.SYS`, used by Run Time Monitor to control event monitoring.

Run Time Monitor uses the MPE V/E System Logging facility to log events during program execution. Therefore, system logging must be enabled for Run Time Monitor to log events. To determine if Run Time Monitor is operating, execute an event that you know will be logged, close the current log file and create a new log file with `:SWITCHLOG`, and generate a report with RTMREP.

RTMSYS

A user with SM or OP capability can use RTMSYS to enable or disable Run Time Monitor and monitoring of specific event classes. These classes are described in Table C-1.

Table C-1. Run Time Monitor Event Classes

Class Number	Class Name	Description
0	Logging Enabled	Enable/disable Run Time Monitor operation.
1	Native Mode Events	Enable/disable logging of uniquely Native Mode events (in addition to events common to both Compatibility Mode and Native Mode.
2	Command Intrinsic Events	Enable/disable logging of any incompatible command executed by the COMMAND intrinsic. Commands are not associated with a specific mode. Therefore, the Native Mode Events (class 1) does not effect this event class.
3	FFILEINFO Intrinsic Events	Enable/disable logging of FFILEINFO related incompatibilities. There are several differences relevant to FFILEINFO mainly associated with device information.
4	FGETINFO Intrinsic Events	Enable/disable logging of FGETINFO related incompatibilities.
5	FCONTROL Intrinsic Events	Enable/disable logging of FCONTROL related incompatibilities.
6	General File System Events	Enable/disable logging of all other file system incompatibilities.
7		Reserved for future use.
8	Miscellaneous Events	Enable/disable logging of events that do not fall under the other event classes.

NOTE

RTMSYS adds at least 8, and as many as 100, words on the user stack during execution of monitored intrinsics. This may result in stack overflow. If stack overflow occurs, you should either rerun the program with the :NOCB option, modify the program to use less stack space, or rely only on OCA's output for detecting incompatibilities.

RTMREP

A report of events can be generated by using the RTMREP program. RTMREP reads the specified system log files and generates a report to the formal file designator RTMLIST of the events found. Two report formats are available:

- A brief report (the default) is a program summary which shows the events generated by each program. A counter shows the number of times that a particular call caused the event. The report is sorted by date, program name, and event class.
- A detailed report shows all events logged, sorted by date, program name, and event class. All information in the log record is formatted and displayed in chronological order.

RTMSL

Run Time Monitor uses an SL file, RTMSL.PUB.SYS, to find areas of incompatibility by intercepting calls to specific system procedures. The SL is bound to the program at load time and allows you to control which events are logged. The relevant information is logged by Run Time Monitor, which then completes the procedure call by calling the actual procedure in SL.PUB.SYS. With Run Time Monitor enabled, the LOADER inserts RTMSL.PUB.SYS into the load sequence when any program not residing in PUB.SYS is executed or allocated.

Managing RTM Disc Space Usage

RTM uses the DMPE system logging facility to keep track of migration events as they occur on your system. As log file records are generated, RTM will consume disc space on your system. The rate of consumption varies based on several different factors. These factors include the following:

- Other activities also being logged on your system (for example, Console logging).
- System load (for example, numbers of users, extent of activity on your system).
- Which RTM event classes have been enabled using RMTSYS.PUB.SYS.
- Period of operation of RTM.
- Number of migration issues encountered in the programs being monitored.

If you are concerned about disc space usage on your system you may wish to monitor the system log files when you first enable RTM (or when you change the event classes being monitored via RTMSYS.PUB.SYS). One easy way to do this is to issue the following :LISTF command.

```
:listf log####.pub.sys,2
ACCOUNT= SYS                GROUP= PUB

FILENAME CODE            LOGICAL RECORD                          SPACE
                        SIZE TYP      EOF        LIMIT R/B      SECTORS #X MX

LOG1327                 1022W VB     27165        1023   1         8192    16 16
LOG1328                 1022W VB     13012        1023   1         4096     8 16
LOG1329                 1022W VB      5402        1023   1         2048     4 16
```

By totaling the SECTORS column you can determine how much disc space is being consumed by the system log files (in the example above, 14336 sectors have been used).

If you find that disc space is being consumed too rapidly the following corrective/preventative steps are recommended:

1. Store system log files on tape for future analysis as they are generated.
2. Disable monitoring for event classes once you feel you have sufficient data on events in those classes.
3. Temporarily disable other types of system logging while using RTM.

Setting Up Run Time Monitor

System logging and the process event type (type 16) must be enabled through the MPE V/E SYSDUMP facility for Run Time Monitor to work. Since any program already loaded or allocated is not linked through RTMSL.PUB.SYS when run-time event monitoring is enabled, it is necessary to deallocate all programs first, or make sure that Run Time Monitor is enabled immediately after a system start.

NOTE

The auto-allocate feature of MPE V/E does not affect Run Time Monitor operation. RTMSYS ensures that auto-allocate is disabled before any Run Time Monitor status changes are made. Once the changes are made, RTMSYS enables autoallocate if it was enabled prior to the change.

Initially, all event classes recognized by RTMSYS are off. For Run Time Monitor to work, you need to turn on the desired event classes.

There are two methods available to automatically enable Run Time Monitor after a WARM/COOL/COLDSTART:

- Through an operator logon UDC
- Through the system startup state configurator file

Operator Logon UDC.

The first method you can use to automatically enable Run Time Monitor is to have OPERATOR.SYS execute a logon UDC that runs the RTMSYS program with an info string. The info string contains the name of a file that has the event classes turned on or off, as desired. Each entry in the file looks exactly as if it was typed interactively.

```
******
EVENTUDC
OPTION LOGON
:RUN RTMSYS.PUB.SYS;info="RTEVENTS.PUB.SYS"
******
```

The file RTEVENTS.PUB.SYS is a standard EDIT/3000 compatible file, either numbered or unnumbered, that you create. Each line contains a class number and the desired status for that particular class.

```
1       0,ON
2       1,ON
3       4,OFF
```

System Startup File

The second method you can use to enable RTMSYS by is streaming a job stream file from the startup state configurator file, SYSTART.PUB.SYS. A job stream file must be used since the :RUN command can not execute from the startup state configurator file.

A sample job stream file to be invoked from a startup state file:

```
!JOB RTMINIT,OPERATOR.SYS
!RUN RTMSYS.PUB.SYS;INFO="RTEVENTS.PUB.SYS"
!TELLOP; **** RTM ENABLED
!EOJ
```

Using RTMSYS

Run Time Monitor is controlled by the program RTMSYS. RTMSYS can be run interactively or in batch mode by any user possessing OP or SM capability by entering:

```
RUN RTMSYS.PUB.SYS
```

You only need to run RTMSYS to change the status of an event class.

Figure C-1 charts the flow of control with RTMSYS prompts. Additional information on the prompts is provided in subsequent headings. Defaults for all prompts are shown in upper case.

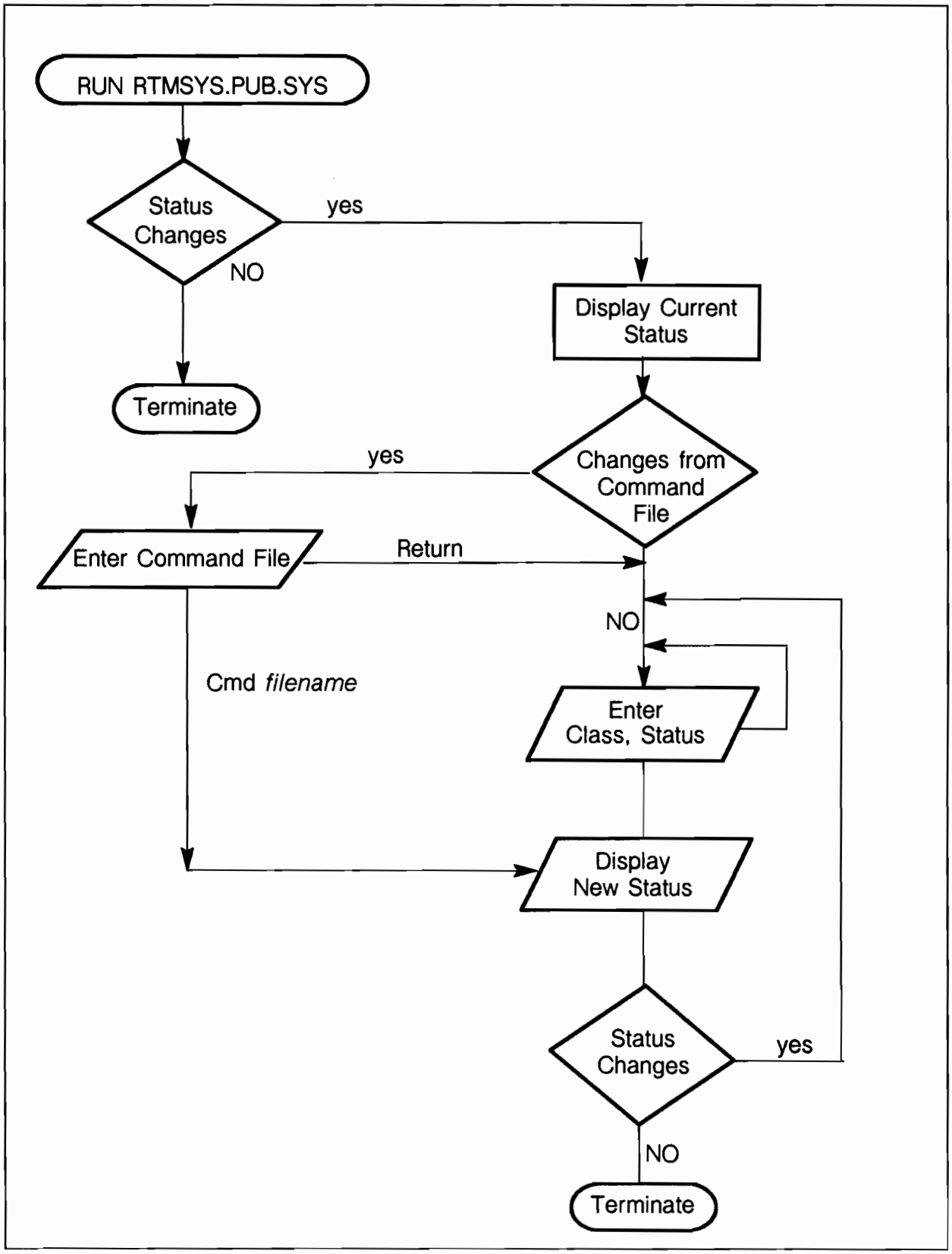


Figure C-1. RTMSYS.

Status Changes Prompt

RTMSYS initially displays all event classes and their current status. The first prompt asks you if you want to make changes to the classes of events that Run Time Monitor detects:

- A positive response causes RTMSYS to continue and takes you to the Command File prompt.
- A negative response, or [Return], terminates RTMSYS.

```
HP30364A.00.00 Run Time Monitor (C) HEWLETT PACKARD COMPANY 1986

Enter a question mark (?) at any prompt for help.
Default answers are shown in upper case.

Do you want to make RTM status changes [yes/NO]?


CLASS DESCRIPTION       STATUS

0 - LOGGING ENABLED      OFF
1 - Native Mode EVENTS   OFF
2 - COMMAND INTRINSIC    OFF
3 - FFILEINFO            OFF
4 - FGETINFO             OFF
5 - FCONTROL             OFF
6 - GENERAL FILESYS      OFF
7 - RESERVED
8 - MISCELLANEOUS        OFF
```

Command File Prompt

```
Do you want to make changes from a command file [yes/NO]?
```

You are being asked if you have a file that contains information regarding the classes of events that you want to alter the status of.

- A positive response takes you to the Enter Command File prompt.
- A negative response (the default), takes you to the Enter Class prompt.

Enter Command File Prompt

```
Enter name of command file:
```

At this prompt, you should enter the name of the Command File. Valid responses include any valid file name of a file containing the event classes and desired status.

This file can be any standard, EDIT/3000 compatible file, either numbered or unnumbered, that contains records indicating the class number and desired status. Each line contains a class number and the desired status for that particular class.

```
1       0,ON
2       1,ON
3       5,OFF
```

Enter Class Prompt

```
ENTER CLASS, ON/OFF?
```

You are being prompted for event class changes. To change the status of an event class, enter the class number and the desired status, ON or OFF, separated by a comma. Continue to enter class numbers and desired status until all changes have been made. When all desired changes have been entered press Return. This returns you to the Status Changes prompt.

Exiting RTMSYS

A negative response, or Return, at the Status Changes prompt causes RTMSYS to terminate.

Running RTMSYS in Batch Mode

Here are three sample job stream files that illustrate how you can enable and disable RTM events from batch mode, and generate a report on all programs run from a specified account.

```
:job rtmstart, manager.sys, pub
:comment
:comment check to see if rtmstart.pub.sys exists
:comment
:setjcw cierror=0
:listf rtmstart.pub.sys
:comment
:comment if rtmstart.pub.sys does not exist then create it
:comment
:if cierror = 907 then
:editor
a
0,on
1,on
2,on
3,on
4,on
5,on
6,on
7,on
8,on
//
k rtmstart.pub.sys
e
:endif
:comment
:comment run rtmsys.pub.sys with command file created above
:comment
:run rtmsys.pub.sys;info="rtmstart.pub.sys"
:eoj
```

Figure C-2. Job Stream to Enable RTM for all Event Classes.

```
:job rtmstop, manager.sys, pub
:comment
:comment check to see if rtmstart.pub.sys exists
:comment
:setjcw cierror=0
:listf rtmstop.pub.sys
:comment
:comment if rtmstop.pub.sys does not exist then create it
:comment
:if cierror = 907 then
:editor
a
0,off
1,off
2,off
3,off
4,off
5,off
6,off
7,off
8,off
//
k rtmstop.pub.sys
e
:endif
:comment
:comment run rtmsys.pub.sys with command file created above
:comment
:run rtmsys.pub.sys;info="rtmstop.pub.sys"
:eoj
```

Figure C-3. Job Stream to Disable RTM for all Event Classes.

```
:job rtmrep, manager.sys, pub
:comment
:comment  The responses to the RTM prompts below have the
:comment  following meanings:
:comment
:comment          Response                          Meaning
:comment
:comment          :file rtmlist;dev=epoc      redirect the report to the
:comment                                      epoc printer.
:comment          n                           do not want a detail line for
:comment                                      each incompatibility detected
:comment          1300                        start with log file 1300
:comment          1305                        end with log file 1305
:comment          y                           subset the data
:comment          @.@.finance                 report on all programs run
:comment                                      out of the finance account
:comment          1,2,3,4,5                   report on events in event
:comment                                      classes 1 through 5
:comment
:run rtmrep.pub.sys
:file rtmlist;dev=epoc
n
1300
1305
y
@.@.finance
1,2,3,4,5
:eoj
```

Figure C-4. Job Stream to Generate a Report for all Finance Programs.

RTMSYS Program Error Messages

The error and warning messages listed in Table C-2 are returned by the two programs that make up the Run Time Monitor utility, RTMSYS and RTMREP.

Table C-2. Run Time Monitor Error Messages

Message Number	Description
12	Classes must be numeric, in the range of 1..8.
14	Is incomplete - a maximum of 100000 records per run allowed.
31	User must have OP capability to run RTMSYS.
39	Changes from Command File not made due to file error.
41	Input must be in the form: "1, ON" or "1, OFF".
42	Class must be numeric, for example "1, ON".
43	Class is out of range, valid classes are 0 through 8.
44	Specify "ON" or "OFF", for example "1, ON".
45	WARNING - system logging is disabled - use SYSDUMP to enable it.
46	WARNING - Logging class 16 is disabled - use SYSDUMP to enable it.
48	The VUFs of RTM segments in SL.PUB.SYS and RTMSL.PUB.SYS do not match.
49	RTM is not enabled. Version information from RTMSL.PUB.SYS is unavailable.
54	RTM ERROR: LOADPROC intrinsic error occurred.
55	WARNING - RTMSL.PUB.SYS does not exist.

Using RTMREP

Run Time Monitor does not automatically generate reports. However, all run-time events that are captured are recorded in a system log file. A report of events logged can be generated by running RTMREP.

There are two report formats.

Run Time Monitor does not automatically generate reports.

- A program summary (the default), a brief report that shows all events generated by each program. A counter shows the number of times that a particular segment/offset caused the event. The report is sorted by date, program name, and event class.
- A detailed report shows all events logged. These events are sorted by Date, program name, and event class. All information in the log record is formatted and displayed in chronological order.

NOTE

RTMREP cannot access the current system log file. If you want to generate a report from the current system log file, you must first close it with the :SWITCHLOG command.

To generate a report, enter the following:

```
RUN RTMREP.PUB.SYS
```

RTMREP prompts you for the choice of a program summary or detailed report.

Additionally, LISTLOG5.PUB.SYS can be used to examine the log file for entries. Refer to the *MPE V Utilities Reference Manual* (32033-90008) for more information on LISTLOG5.

RTMREP has a HELP facility can be invoked at any time by entering ?. An explanation of the current prompt is then displayed, followed by the same prompt. You may also issue any MPE V/E command that can be executed programmatically be prefixing it with a colon (:).

Output Device Specification

The output formal file designator is RTMLIST, which by default is opened with device class LP. The :FILE command can be used to redirect the output as necessary.

Figure C-5 charts the flow of control with RTMREP prompts. The prompts are described under the following headings. Defaults for all prompts are shown in upper case.

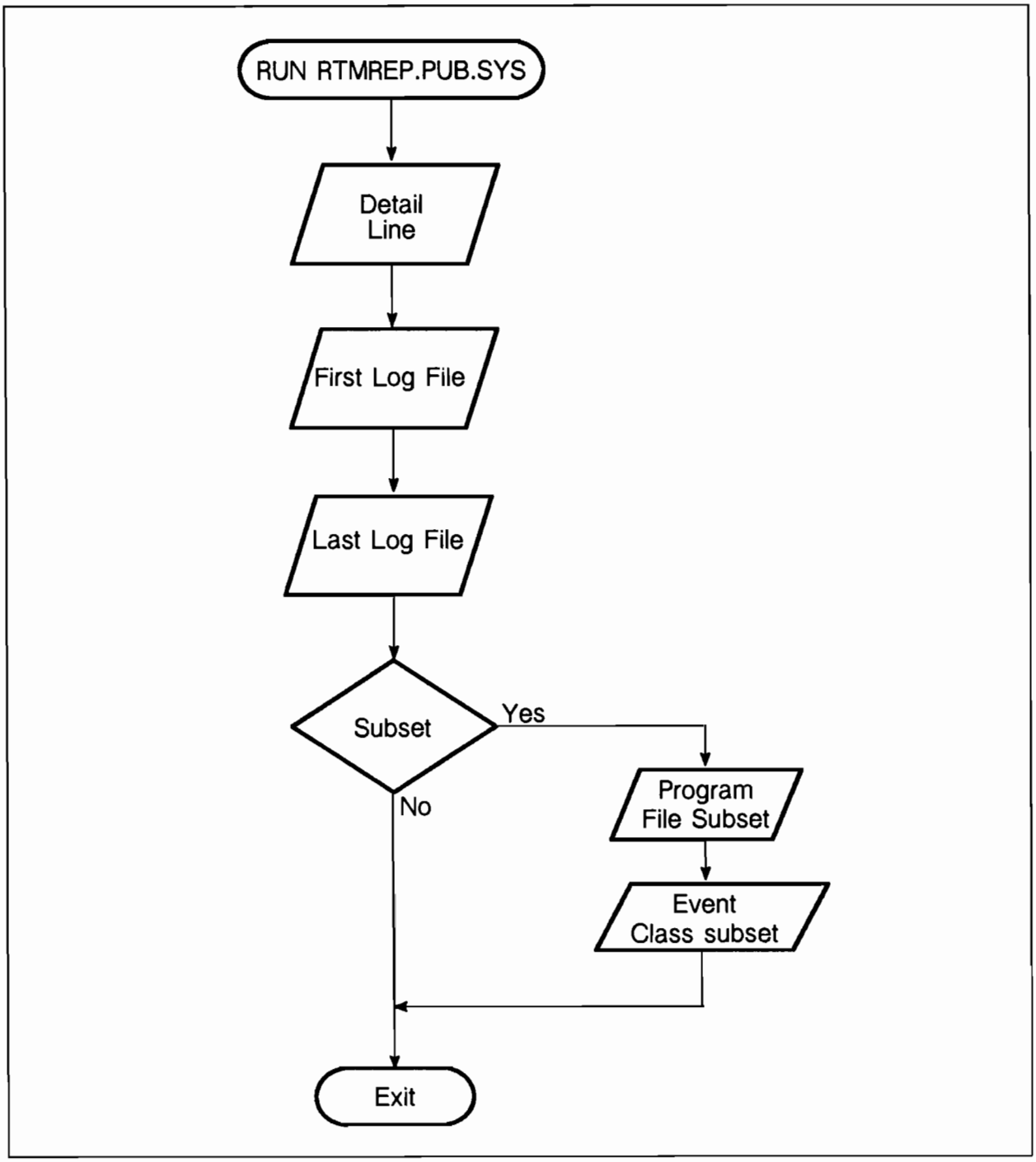


Figure C-5. RTMREP.

Detail Line Prompt

```
HP30364A.00.00 Run Time Monitor (C)
HEWLETT PACKARD COMPANY 1986

Enter a question mark (?) at any prompt for help.
Default answers for all prompts are shown in upper case.

Do you want a detail line for each event [yes/NO]?
```

- A positive answer to this prompt gives you the detailed report.
- A negative answer (the default) to this prompt gives you the summary report. Either response takes you to the "Log File Specification" prompts.

Log File Specification Prompt

```
You may specify a range of log files to be analyzed for the report
Enter first log file:
Enter last log file:
```

To examine a range of log files, type the starting log file number and the last log file number.

To examine only one log file, enter the LOG file number in response to the Enter First Log File prompt and a Return to the Enter Last Log File prompt.

Either response takes you to the "Subset" prompt.

Subset Prompt

```
Do you want to report on a subset of available data [yes/NO]?
```

- A positive response allows you to tailor the report to your needs by selecting specific a subset of programs. This takes you to the "Enter Program File Subset" prompt.
- A negative response (the default) reports on all available information.

Enter Program File Subset Prompt

```
Enter program file subset [CR will select all]:
```

You are being prompted for the name of a file(s) you want RTMREP to create a report about. In addition, you can specify a set of program files using wild card characters (@, #, ?) in conjunction with MPE V/E file specifications. For example, the following file set specification indicates all program files that start with the letter "C":

```
C@.mygroup.myacct
```

Enter Class Numbers Prompt

```
Enter class numbers, separated by commas, or CR for all:
```

You are being prompted for the event classes you want included in the report RTMREP creates.

Exiting RTMREP

After report specifics are complete, RTMREP prints the report, then exits.

Sample Report

The following lists a sample Run Time Monitor report.

NOTE

The RTM report format uses 132 columns. The report that follows has been truncated to 80 columns to fit this page. A complete report would include a field to the far right listing a brief description of the incompatibility.

```
1HP30364X.01.02  RUN TIME EVENTS BY PROGRAM FILE     LOG DATE: THU, OCT 23,1986
0PROGRAM FILE          TIME       #J/S       SEGMENT      DELTAP   STATUS   EVENT

 STICP.SOOL.MPEV      10:29 AM    #S127      GSL  %000    000072   060002   107
                      10:29 AM    #S127      GSL  %000    000032   060002   503
                      10:29 AM    #S127      PROG %000    000061   060001   503
```

D

Incompatibilities

Introduction

This appendix lists all known incompatibilities between MPE V/E and MPE XL. These incompatibilities are divided into two categories, undetected incompatibilities and detected incompatibilities.

Undetected Incompatibilities

Incompatibilities that cannot be detected by the Migration Toolset (Object Code Analyzer and Run Time Monitor) are considered undetected incompatibilities. Undetected incompatibilities fall into the following groups:

- Peripheral dependent incompatibilities.
- Intrinsics incompatibilities.
- Subsystem and compiler incompatibilities.

Peripheral Dependent Incompatibilities

Peripherals that are supported on MPE V/E-based systems may not be currently supported on MPE XL. The *HP 3000 Computer Systems System Configuration Guide* (5954-9354) provides a list of the peripherals currently supported on MPE XL. The following incompatibilities exist as long as the associated peripherals are not supported:

- FREADBACKWARD will result in a run-time error since no tape drives with reverse read capability are supported on MPE XL.
- FWRITE *controlcodes* %311, %312, %313, %314-317 are associated with devices that are not currently supported on MPE XL.

Intrinsic Incompatibilities

The following intrinsic incompatibilities cannot be detected by RTM or OCA:

- Applications that use undocumented intrinsics, execute in Privileged Mode, call routines that require Privileged Mode, or use privileged machine instructions may need to be modified when moved to the 900 Series. Only a detailed study of the program will determine this, as appropriate modifications must be made on a case-by-case basis. The Migration Toolset cannot provide sufficient details.
- Various MPE XL subsystems require file numbers for their own use. Therefore, you should make no assumptions about file numbers returned to your program. All hardcoded file numbers should be removed and replaced with appropriate variables that contain the file number values returned by system intrinsics.
- Intrinsic parameter types and conventions have changed in Native Mode. You should use the intrinsic mechanism to ensure compatibility.
- FOPEN, on MPE XL, specifying a maximum number of extents, other than one, allows the file system to determine the number and size of extents independently.
- FOPEN, specifying an *foption* of zero or an *aoption* of zero always behaves as if the parameters were defaulted.
- FOPEN, if errors 90 and 91 are applicable at the same time for a failed FOPEN call, error 90 is the only error returned via the FCHECK intrinsic.
- FOPEN, the default for initial allocation is now zero extents rather than one extent in MPE V/E.

Subsystem and Compiler Incompatibilities

This section summarizes non-Operating System specific incompatibilities. Other areas that have changed from MPE V/E to MPE XL include:

- Applications which use SPL to assemble any of the following nonprivileged machine code instructions must be removed before moving to the 900 Series:

  RMSK

  RSW

  RCLK

- On the 900 Series, XON/XOFF control characters are not stripped from the data stream while XON/XOFF is set OFF. Applications using this feature must be modified.
- Applications that depend on specific DSERROR codes may have to be modified, since the error codes on MPE XL comply with the new NS 3000/XL error codes.
- Only compiled BASIC/V programs run in Compatibility Mode on MPE XL. Interpreted BASIC/V programs must be compiled before being moved to the 900 Series.

- Pascal applications written on MPE V/E based systems which depend on the specific data layout using the variant parts of records may require modification before recompilation into Native Mode. This is because pointers are sized and aligned on 32-bit boundaries.

- Pascal applications which take advantage of undetected errors may have to be modified. These undetected errors are listed in Appendix E of the *Pascal/3000 Reference Manual* (32106-90001).

- HP FORTRAN 77 programs that use COMMON or EQUIVALENCE should be modified so that 32-bit data types are aligned on 32-bit boundaries and so that 64-bit data types are aligned on 64-bit boundaries before being moved to Native Mode, in order to ensure maximum performance.

- SPL is available in Compatibility Mode both for run-time support and for development of SPL applications. Your SPL applications and SPL procedures will run in Compatibility Mode. However, because of its close relationship to the MPE V/E based architecture, HP does not support a Native Mode SPL compiler on the 900 Series. The Object Code Translator can be used to increase the performance of MPE V/E object code.

- High-level language applications that will be migrated to Native Mode and that call user supplied SPL procedures will require Switch Stubs to access these procedures.

- Procedures passed as parameters in HP FORTRAN 77 or Pascal require 16-bit parameters on MPE V/E and 32-bit parameters on MPE XL.

- The passing of a field of a record, an element of an array, or an equivalenced variable as a reference parameter and requesting that more data be copied than is represented by the item size may produce unexpected results on MPE XL.

- The passing of a pointer as an SPL integer or by value to a Compatibility Mode SPL routine.

- SPL procedures called with `OPTION VARIABLE`. Explicit parameter masks are supported in Compatibility Mode only. Native mode code calling a user SPL routine in Compatibility Mode must do so with a Switch Stub. All Native Mode calls to intrinsics must go through the intrinsic mechanism.

- Variables located in programs should be properly initialized. Uninitialized variables that did not cause problems on MPE V/E-based systems may cause programs to abort on MPE XL-based systems.

Detected Incompatibilities

Incompatibilities that are detected by the Migration Toolset are listed in Table D–1. Table D–1 lists incompatibilities by number as reported by the Migration Toolset. This table contains:

- A message number as returned by the Migration Toolset.
- The complete incompatibility message.
- The mode in which the incompatibility would arise and the Migration tool used to detect the incompatibility. Also specified is the event class you must enable before RTM will detect the incompatibility.
- The cause of the incompatibility.
- Action to take to work around the incompatibility.

Table D-1. Incompatibilities detected by OCA and RTM.

101	**MESSAGE**	XARITRAP may require a new trap handler in MPE XL Native Mode.
	MODE	Native Mode incompatibility detected by OCA and RTM (Event Class 1).
	CAUSE	HP Precision Architecture is a 32-bit architecture, while the architecture of MPE V/E-based computer systems is a 16-bit architecture. In Native Mode, memory addresses represented as pointers or PLABELs must be 32-bit entities, while in MPE V/E and in Compatibility Mode, pointers and PLABELs must be 16-bit entities. The XARITRAP intrinsic uses PLABELs.
	ACTION	Redeclare the variables you pass through the required *plabel* and *oldplabel* parameters of XARITRAP to be 32-bit entities.
102	**MESSAGE**	XCONTRAPmay require a new trap handler in MPE XL Native Mode.
	MODE	Native Mode incompatibility detected by OCA and RTM (Event Class 1)
	CAUSE	HP Precision Architecture is a 32-bit architecture, while the architecture of MPE V/E-based computer systems is a 16-bit architecture. In Native Mode, memory addresses represented as pointers or PLABELs must be 32-bit entities, while in MPE V/E and in Compatibility Mode, pointers and PLABELs must be 16-bit entities. The XCONTRAP intrinsic uses PLABELs.
	ACTION	Redeclare the variables you pass through the required *plabel* and *oldplabel* parameters of XCONTRAP to be 32-bit entities.

103	**MESSAGE**	XLIBTRAP may require a new trap handler in MPE XL Native Mode.
	MODE	Native Mode incompatibility detected by OCA and RTM (Event Class 1).
	CAUSE	HP Precision Architecture is a 32-bit architecture, while the architecture of MPE V/E-based computer systems is a 16-bit architecture. In Native Mode, memory addresses represented as pointers or PLABELs must be 32-bit entities, while in MPE V/E and in Compatibility Mode, pointers and PLABELs must be 16-bit entities. The XLIBTRAP intrinsic uses PLABELs.
	ACTION	Redeclare the variables you pass through the required *plabel* and *oldplabel* parameters of XLIBTRAP to be 32-bit entities.
104	**MESSAGE**	XSYSTRAP may require a new trap handler in MPE XL Native Mode.
	MODE	Native Mode incompatibility detected by OCA and RTM (Event Class 1).
	CAUSE	HP Precision Architecture is a 32-bit architecture, while the architecture of MPE V/E-based computer systems is a 16-bit architecture. In Native Mode, memory addresses represented as pointers or PLABELs must be 32-bit entities, while in MPE V/E and in Compatibility Mode, pointers and PLABELs must be 16-bit entities. The XSYSTRAP intrinsic uses PLABELs.
	ACTION	Redeclare the variables you pass through the required *plabel* and *oldplabel* parameters of XSYSTRAP to be 32-bit entities.

105	**MESSAGE**	SEARCH may encounter pointer problems in MPE XL Native Mode.
	MODE	Native Mode incompatibility detected by OCA and RTM (Event Class 1)
	CAUSE	HP Precision Architecture is a 32-bit architecture, while the architecture of MPE V/E-based computer systems is a 16-bit architecture. In Native Mode, memory addresses represented as pointers or PLABELs must be 32-bit entities, while in MPE V/E and in Compatibility Mode, pointers and PLABELs must be 16-bit entities. The SEARCH intrinsic may use pointers.
	ACTION	If you use the optional *defn* parameter of SEARCH, you must redeclare the variable you pass through it to be a 32-bit entity.

106	**MESSAGE**	MYCOMMAND may encounter pointer problems in MPE XL Native Mode.
	MODE	Native Mode incompatibility detected by OCA and RTM (Event Class 1).
	CAUSE	HP Precision Architecture is a 32-bit architecture, while the architecture of MPE V/E-based computer systems is a 16-bit architecture. In Native Mode, memory addresses represented as pointers or PLABELs must be 32-bit entities, while in MPE V/E and in Compatibility Mode, pointers and PLABELs must be 16-bit entities. The MYCOMMAND intrinsic uses pointers.
	ACTION	Redeclare the variable you pass through the required *params* parameter of MYCOMMAND to be a an array of 32-bit entities. In addition, if you use the optional *defn* parameter, you must redeclare the variable you pass through it to be a 32-bit entity.

107	**MESSAGE**	CREATEPROCESS may encounter byte pointer problems in MPE XL Native Mode.
	MODE	Native Mode incompatibility detected by OCA and RTM (Event Class 1).
	CAUSE	HP Precision Architecture is a 32-bit architecture, while the architecture of MPE V/E-based computer systems is a 16-bit architecture. In Native Mode, memory addresses represented as pointers or PLABELs must be 32-bit entities, while in MPE V/E and in Compatibility Mode, pointers and PLABELs must be 16-bit entities. The CREATEPROCESS intrinsic may use pointers.
	ACTION	If you use the optional *itemnums* and *items* parameters of CREATEPROCESS, you must redeclare the variables you pass through them to be arrays of 32-bit entities.
200	**MESSAGE**	Possible programmatic execution of incompatible commands.
	MODE	Compatibility Mode and Native Mode incompatibility detected by OCA.
	CAUSE	Some MPE V/E commands have either been modified for use on MPE XL or are no longer supported on MPE XL. The COMMAND intrinsic may execute one of the following modified or unsupported commands: `:LISTF, :LISTACCT, :LISTUSER, :LISTGROUP, :PTAPE, :TUNE, SHOWCACHE, :CACHECONTROL, STARTCACHE, STOPCACHE, :SHOWLOG, :SWITCHLOG, RESUMELOG, :SHOWCOM, :DSCONTROL, and :SPEED.`
	ACTION	If OCA reports this event, enable Event Class 2 of RTM to detect when COMMAND executes one of the commands listed above. Check for event numbers 201 through 218, which indicate that RTM detected execution of a modified or unsupported command. In all cases, analyze the program to determine whether the command executed was acquired interactively or embedded in code and evaluate how the command is used.

201	**MESSAGE**	Programmatic :LISTF.
	MODE	Compatibility Mode and Native Mode incompatibility detected by RTM (Event Class 2).
	CAUSE	Because of differences between HP Precision Architecture and the architecture of MPE V/E-based computer systems, :LISTF -1 is not supported on MPE XL. In addition, because of changes in extent management on MPE XL, the MX (maximum extents) field of :LISTF -2 will contain an asterisk if a maximum number of extents was not specified at file creation.
	ACTION	Remove code dependent upon :LISTF -1. Determine what data you needed from :LISTF -1 and evaluate alternative ways to retrieve that data; for example, through the FLABELINFO intrinsic. Furthermore, MPE XL enhancements to the :LISTF command may also provide this data. If your code is dependent upon data retrieved from the MX field output by :LISTF 2, you may have to modify your code to handle the asterisk.
202	**MESSAGE**	Programmatic :LISTACCT to a file.
	MODE	Compatibility Mode and Native Mode incompatibility detected by RTM (Event Class 2).
	CAUSE	Because of enhancements to the MPE XL operating system, output from the :LISTACCT command appears in ASCII format.
	ACTION	If you are doing programmatic analysis of the :LISTACCT output, you must modify your code to handle the new output format.

203	**MESSAGE**	Programmatic :LISTUSER to a file.
	MODE	Compatibility Mode and Native Mode incompatibility detected by RTM (Event Class 2).
	CAUSE	Because of enhancements to the MPE XL operating system, output from the :LISTUSER command appears in ASCII format.
	ACTION	If you are doing programmatic analysis of the :LISTUSER output, you must modify your code to handle the new output format.
204	**MESSAGE**	Programmatic :LISTGROUP to a file.
	MODE	Compatibility Mode and Native Mode incompatibility detected by RTM (Event Class 2).
	CAUSE	Because of enhancements to the MPE XL operating system, output from the :LISTGROUP command appears in ASCII format.
	ACTION	If you are doing programmatic analysis of the :LISTGROUP output, you must modify your code to handle the new output format.
205	**MESSAGE**	Programmatic :PTAPE command not supported.
	MODE	Compatibility Mode and Native Mode incompatibility detected by RTM (Event Class 2).
	CAUSE	Modifications have been made to the attached peripheral environment of MPE XL, resulting in paper tape readers being unsupported on MPE XL–based computer systems. The :PTAPE command is not supported on MPE XL–based computer systems.
	ACTION	Remove all :PTAPE invocations and dependent code. If you are using :PTAPE for uses other than reading from paper tape devices, you must determine the original need for the call to :PTAPE, and develop alternative solutions.

207	**MESSAGE**	Programmatic :TUNE.
	MODE	Compatibility Mode and Native Mode incompatibility detected by RTM (Event Class 2).
	CAUSE	Because of enhancements to the MPE XL operating system, the required *minclockcycle* parameter of the :TUNE command is ignored.
	ACTION	No action is required, as the *minclockcycle* parameter of :TUNE is ignored by the operating system.
208	**MESSAGE**	Programmatic :SHOWCACHE.
	MODE	Compatibility Mode and Native Mode incompatibility detected by RTM (Event Class 2).
	CAUSE	HP Precision Architecture features include complete operating system management of all cache activities. Programmatic control of cache activities is not allowed. MPE V/E cache management commands, including the :SHOWCACHE command are not supported on MPE XL.
	ACTION	Remove all :SHOWCACHE invocations and dependent code.
209	**MESSAGE**	Programmatic :CACHECONTROL.
	MODE	Compatibility Mode and Native Mode incompatibility detected by RTM (Event Class 2).
	CAUSE	HP Precision Architecture features include complete operating system management of all cache activities. Programmatic control of cache activities is not allowed. In addition, BLOCKONWRITE on a system-wide basis is not supported on MPE XL. MPE V/E cache management commands, including the :CACHECONTROL command, are not supported on MPE XL.
	ACTION	Remove all :CACHECONTROL invocations and dependent code. BLOCKONWRITE on a file-by-file basis is available through the FSETMODE intrinsic.

210	**MESSAGE**	Programmatic :STARTCACHE.
	MODE	Compatibility Mode and Native Mode incompatibility detected by RTM (Event Class 2)
	CAUSE	HP Precision Architecture features include complete operating system management of all cache activities. Programmatic control of cache activities is not allowed. MPE V/E cache management commands, including the :STARTCACHE command, are not supported on MPE XL.
	ACTION	Remove all :STARTCACHE invocations and dependent code.
211	**MESSAGE**	Programmatic :STOPCACHE.
	MODE	Compatibility Mode and Native Mode incompatibility detected by RTM (Event Class 2).
	CAUSE	HP Precision Architecture features include complete operating system management of all cache activities. Programmatic control of cache activities is not allowed. MPE V/E cache management commands, including the :STOPCACHE command, are not supported on MPE XL.
	ACTION	Remove all :STOPCACHE invocations and dependent code.
212	**MESSAGE**	Programmatic :SHOWLOG.
	MODE	Compatibility Mode and Native Mode incompatibility detected by RTM (Event Class 2)
	CAUSE	This event is not a migration issue. There is no incompatibility.
	ACTION	Ignore this event.
213	**MESSAGE**	Programmatic :SWITCHLOG.
	MODE	Compatibility Mode and Native Mode incompatibility detected by RTM (Event Class 2).
	CAUSE	This event is not a migration issue. There is no incompatibility.
	ACTION	Ignore this event.

214	**MESSAGE**	Programmatic :RESUMELOG.
	MODE	Compatibility Mode and Native Mode event detected by RTM (Event Class 2).
	CAUSE	This event is not a migration issue. There is no incompatibility.
	ACTION	Ignore this event.

215	**MESSAGE**	Programmatic :SHOWCOM.
	MODE	Compatibility Mode and Native Mode event detected by RTM (Event Class 2).
	CAUSE	Modifications have been made to the attached peripheral environment of MPE XL; the Intelligent Network Processor (INP) and the associated :SHOWCOM command are not supported on MPE XL.
	ACTION	Remove all :SHOWCOM invocations and dependent code. Evaluate your original need for :SHOWCOM and determine alternative solutions. Equivalent functionality may be available through the NS 3000/XL subsystem.

216	**MESSAGE**	Programmatic :DSCONTROL.
	MODE	Compatibility Mode and Native Mode event detected by RTM (Event Class 2).
	CAUSE	Modifications have been made to the attached peripheral environment of MPE XL, resulting in the DS/3000 subsystem being replaced on MPE XL with the NS 3000/XL subsystem (an enhanced version of the NS 3000 subsystem available on MPE V/E). The :DSCONTROL command is not supported on MPE XL.
	ACTION	Remove all :DSCONTROL invocations and dependent code. Evaluate your original need for :DSCONTROL and determine alternative solutions, for example, the :NSCONTROL command currently available through the NS 3000/V subsystem In addition, equivalent functionality may be available through the NS 3000/XL subsystem.

217	**MESSAGE**	Programmatic :STARTSESS.
	MODE	Compatibility Mode and Native Mode event detected by RTM (Event Class 2).
	CAUSE	Modification have been made to the attached peripheral environment of MPE XL; the following TERMTYPEs are not supported: 4, 6, 12, 13, 14, 15, 16, 19, and 20. The required *logonstring* of the :STARTSESS command may specify an unsupported TERMTYPE.
	ACTION	If the required *logonstring* parameter of :STARTSESS includes a TERMTYPE, you may have to modify the variable passed through *logonstring* to specify a supported TERMTYPE (10, 18, 21, and 22).
218	**MESSAGE**	Programmatic :SPEED.
	MODE	Compatibility Mode and Native Mode event detected by RTM (Event Class 2).
	CAUSE	Modifications have been made to the attached peripheral environment of MPE XL; the following baud rates are not supported: 110, 150, and 600. The :SPEED command uses a baud rate.
	ACTION	Modify only the :SPEED invocations that specify the unsupported baud rates listed above, replacing unsupported baud rates with supported baud rates.

250	**MESSAGE**	Error numbers returned by the COMMAND intrinsic may be different.
	MODE	Compatibility Mode and Native Mode incompatibility detected by OCA and RTM (Event Class 2). When RTM detects events 250, 251, and 252, only event 250 is actually logged to a system log file located on disc. This minimizes disc space consumption associated with each detected invocation of the COMMAND intrinsic.
	CAUSE	Because of differences between HP Precision Architecture and the architecture of MPE V/E based computer systems, a few MPE V/E Command Interpreter (CI) error numbers and their associated messages have been replaced on MPE XL with more precise error numbers and messages. The required *error* parameter of the COMMAND intrinsic may , in a very few cases, return a different CI error number on MPE XL-based computer system than it would on an MPE V/E-based computer system for the same type of error.
	ACTION	If you have code that relies upon specific non-zero values returned in the required *error* parameter of COMMAND, you may have to modify your code. Analyze your source code and list the error numbers upon which your MPE V/E code relies. When MPE XL error numbers are available, check for incompatibilities. In addition, if this event is logged by RTM, analyze your code to determine whether the incompatibility is associated with event 250, 251, or 252.

251	**MESSAGE**	The meaning of the value returned in parm has changed for the COMMAND intrinsic.
	MODE	Compatibility Mode and Native Mode incompatibility detected by OCA. If RTM logs event 250, event 251 may apply to the event detected by RTM. Analyze your code to determine if this is the case.
	CAUSE	Because of enhancements to the MPE XL operating system, a value returned by the required *param* parameter of the COMMAND intrinsic has a different interpretation on MPE XL than it does on MPE V/E. On MPE V/E, a positive value *n* less than 12 indicates that an error occurred in the *nth* parameter of the character array passed through the *comimage* parameter; a positive value greater than 12 is a file system error number. On MPE XL, a negative value indicates the character position within the *comimage* array which contains the first character of the parameter where an error occurred; a positive value indicates a file system error number.
	ACTION	If you have code that relies upon the value returned by the *parm* parameter of COMMAND, determine what, if any, changes you need to make.
252	**MESSAGE**	COMMAND intrinsic supports process creation on MPE XL.
	MODE	Compatibility Mode and Native Mode incompatibility detected by OCA. If RTM logs event 250, event 252 may apply to the event detected by RTM. Analyze your code to determine if this is the case.
	CAUSE	Because of enhancements to the MPE XL operating system, you can use the COMMAND intrinsic to invoke commands that create processes, for example, :RUN, :SEGMENTER, and so forth.
	ACTION	If your program is using the COMMAND intrinsic to invoke a CI command that has been input by a user, you should investigate whether or not you want the user to invoke process creation commands not allowed on MPE V/E.

300	**MESSAGE**	Some FFILEINFO *item* values may not be valid on MPE XL.
	MODE	Compatibility Mode and Native Mode incompatibility detected by OCA.
	CAUSE	Some values returned in the optional *itemvalue* parameter of the FFILEINFO intrinsic either have modified meanings or have no meaning on an MPE XL–based computer system. Affected *itemvalues* are those associated with the following values passed in an associated *itemnums* parameter: 5, 7, 13, 15, 16, 40, 41, 42, 44, 47, 48, and 49.
	ACTION	If OCA reports this event, enable Event Class 3 of RTM to detect when FFILEINFO returns one of the *itemvalues* listed above. Check for event numbers 305 through 349 which indicate that RTM detected that FFILEINFO returned a modified or meaningless *itemvalues*.
305	**MESSAGE**	FFILEINFO *itemvalue* 5 device type/subtype.
	MODE	Compatibility Mode and Native Mode incompatibility detected by RTM (Event Class 3).
	CAUSE	When the specified file is a Standard disc file (as opposed to a KSAM, RIO, or CIR disc file),MPE XL does not distinguish between the supported disc devices' device types. In this specific case, the value returned by *itemvalue* 5 always represents device type 3, subtype 8 (indicating a 793X disc drive).
	ACTION	If the Standard disc file specified in the FFILEINFO call will not reside on a 793X disc drive, you must remove dependency upon *itemvalue* 5.

307	**MESSAGE**	FFILEINFO *itemvalue* 7 hardware device address.
	MODE	Compatibility Mode and Native Mode incompatibility detected by RTM (Event Class 3).
	CAUSE	MPE XL does not support the DRT number/unit number hardware address. When the specified file is located on a disc, the value returned by *itemvalue* 7 always represents DRT number 8, unit number 0, meaningless values. For MPE XL disc files, the DRT number/unit number representation for a hardware device address has been replaced by device path, available through *itemvalue* 75 of the FFILEINFO intrinsic.
	ACTION	Remove dependency upon *itemvalue* 7 of FFILEINFO. Evaluate why your program used this value. You may be able to perform an equivalent action using the device path returned by *itemvalue* 75 of FFILEINFO.
313	**MESSAGE**	FFILEINFO *itemvalue* 13 physical I/O count.
	MODE	Compatibility Mode and Native Mode incompatibility detected by RTM (Event Class 3).
	CAUSE	Because of enhancements to the MPE XL operating system, the value returned by *itemvalue* 13 of the FFILEINFO intrinsic is the physical I/O count only for files residing on devices other than disc; for files residing on disc, *itemvalue* 13 returns the logical I/O count.
	ACTION	If your code relies upon *itemvalue* 13 of FFILEINFO returning the physical I/O count for disc files, you may have to either remove the dependency or modify your code to rely instead upon the logical I/O count.

315	**MESSAGE**	FFILEINFO – Item 15 extent size.
	MODE	Compatibility Mode and Native Mode incompatibility detected by RTM (Event Class 3).
	CAUSE	On MPE XL, the file system manages extent allocation, where variable numbers of extents, each of variable size, are allocated based upon a need basis. There is no fixed extent size for a file. Extent information specified at file creation is meaningless on an MPE XL–based computer system and is stored in the file label only to maintain backward compatibility with MPE V/E–based computer systems. Therefore, *itemvalue* 15 of the FFILEINFO intrinsic returns a value that is meaningless on MPE XL.
	ACTION	If your code relies upon the extent size returned by *itemvalue* 15 of FFILEINFO, you will have to modify your code to remove dependency upon extent size.
316	**MESSAGE**	FFILEINFO – Item 16 maximum number of extents.
	MODE	Compatibility Mode and Native Mode incompatibility detected by RTM (Event Class 3).
	CAUSE	On MPE XL, the file system manages extent allocation, where variable numbers of extents, each of variable size, are allocated based upon a need basis. There is no practical limit to the number of extents for a file. Extent information specified at file creation is meaningless on an MPE XL–based computer system and is stored in the file label only to maintain backward compatibility with MPE V/E–based computer systems. Therefore, *itemvalue* 16 of the FFILEINFO intrinsic returns a value that is meaningless on MPE XL.
	ACTION	If your code relies upon the number of extents returned by *itemvalue* 16 of FFILEINFO, you will have to modify your code to remove dependency upon number of extents.

340	**MESSAGE**	FFILEINFO – Item 40 disc device status.
	MODE	Compatibility Mode and Native Mode incompatibility detected by RTM (Event Class 3).
	CAUSE	Because of differences between HP Precision Architecture and the architecture of MPE V/E-based computer systems, the disc device status returned in *itemvalue* 40 of the FFILEINFO intrinsic has no meaning on MPE XL, but always returns a 0. This *itemvalue* is kept only to maintain backward compatibility with MPE V/E-based computer systems.
	ACTION	If your code relies upon the disc device status returned by *itemvalue* 40 of FFILEINFO, you will have to modify your code to remove the dependency.
341	**MESSAGE**	FFILEINFO – Item 41 device type.
	MODE	Compatibility Mode and Native Mode incompatibility detected by RTM (Event Class 3).
	CAUSE	When the specified file is a standard disc file (as opposed to a KSAM, RIO, or CIR disc file), MPE XL does not distinguish between the supported disc devices' device types. In this specific case, the value returned by *itemvalue* 41 of the FFILEINFO intrinsic always represents device type 3 (indicating a 79XX disc drive).
	ACTION	If the standard disc file specified in the FFILEINFO call will not reside not a 79XX disc drive, you must remove dependency upon *itemvalue* 41.
342	**MESSAGE**	FFILEINFO – Item 42 device subtype.
	MODE	Compatibility Mode and Native Mode incompatibility detected by RTM (Event Class 3).
	CAUSE	When the specified file is a standard disc file (as opposed to a KSAM, RIO, or CIR disc file), MPE XL does not distinguish between the supported disc devices' device types. In this specific case, the value returned by *itemvalue* 42 always represents device subtype 8 (indicating a 793X disc drive).
	ACTION	If the standard disc file specified in the FFILEINFO call will not reside not a 793X disc drive, you must remove dependency upon *itemvalue* 42.

344	**MESSAGE**	FFILEINFO – Item 44 last extent allocated.
	MODE	Compatibility Mode and Native Mode incompatibility detected by RTM (Event Class 3).
	CAUSE	Because of enhancements to the MPE XL operating system, *itemvalue* 44 of the FFILEINFO intrinsic returns a value that represents the number of extents that have been accessed during the current open. On MPE V/E-based computer systems, this same *itemvalue* returns the cardinal number of the last extent accessed.
	ACTION	If your code relies upon the information returned in *itemvalue* 44 of FFILEINFO, you will have to either remove the dependency or modify your code to rely upon the MPE XL interpretation of *itemvalue* 44.
347	**MESSAGE**	FFILEINFO – Item 47 DRT number.
	MODE	Compatibility Mode and Native Mode incompatibility detected by RTM (Event Class 3).
	CAUSE	MPE XL does not support the DRT number/unit number hardware address. When the specified file is located on a disc, the value returned in *itemvalue* 47 of the FFILEINFO intrinsic always represents DRT number 8, a meaningless value. For MPE XL disc files, the DRT number/unit number representation for a hardware device address has been replaced by device path, available through *itemvalue* 75 of FFILEINFO.
	ACTION	Remove dependency upon *itemvalue* 47 of FFILEINFO. Evaluate why your program used this value. You may be able to perform an equivalent action using the device path returned by *itemvalue* 75 of FFILEINFO.

348	**MESSAGE**	FFILEINFO – Item 48 device unit number.
	MODE	Compatibility Mode and Native Mode incompatibility detected by RTM (Event Class 3).
	CAUSE	MPE XL does not support the DRT number/unit number hardware address. When the specified file is located on a disc, the value returned the value returned by *itemvalue* 48 of the FFILEINFO intrinsic always represents unit number 0, a meaningless value. For MPE XL disc files, the DRT number/unit number representation for a hardware device address has been replaced by device path, available through *itemvalue* 75 of FFILEINFO.
	ACTION	Remove dependency upon *itemvalue* 48 of FFILEINFO. Evaluate why your program used this value. You may be able to perform an equivalent action using the device path returned by *itemvalue* 75 of FFILEINFO.
349	**MESSAGE**	FFILEINFO – Item 49 software interrupt plabel (MSG files).
	MODE	Compatibility Mode and Native Mode incompatibility detected by RTM (Event Class 3).
	CAUSE	HP Precision Architecture is a 32-bit architecture, while the architecture of MPE V/E-based computer systems is a 16-bit architecture. In Native Mode, memory addresses represented as pointers or PLABELs must be 32-bit entities, while in MPE V/E and in Compatibility Mode, pointers and PLABELs must be 16-bit entities. An integer equivalent to a PLABEL of a software interrupt handler is returned in *itemvalue* 49 of the FFILEINFO intrinsic.
	ACTION	Redeclare the variable you use to hold the value returned by *itemvalue* 49 of FFILEINFO to be a 32-bit entity.

400	**MESSAGE**	Some FGETINFO item values may not be valid on MPE XL.
	MODE	Compatibility Mode and Native Mode incompatibility detected by OCA.
	CAUSE	The values returned in the optional *devtype*, *haddr*, *physcount*, *extsize*, and *numextents* parameters of the FGETINFO intrinsic have either modified meanings or no meaning on an MPE XL–based computer system.
	ACTION	If OCA reports this event, enable Event Class 4 of RTM to detect when FGETINFO returns one of the optional parameters listed above. Check for event numbers 405 through 416 which indicate that RTM detected that FGETINFO returned a modified or meaningless parameter value.
405	**MESSAGE**	FGETINFO *devtype* parameter.
	MODE	Compatibility Mode and Native Mode incompatibility detected by RTM (Event Class 4).
	CAUSE	When the specified file is a standard disc file (as opposed to a KSAM, RIO, or CIR disc file), MPE XL does not distinguish between the supported disc devices' device types. In this specific case, the value returned by the optional *devtype* parameter of the FGETINFO intrinsic always represents device type 3, subtype 8 (indicating a 793X disc drive).
	ACTION	If the standard disc file specified in the FGETINFO call will not reside on a 793X disc drive, you must remove dependency upon the *devtype* parameter.

407	**MESSAGE**	FGETINFO *haddr* parameter.
	MODE	Compatibility Mode and Native Mode incompatibility detected by RTM (Event Class 4).
	CAUSE	MPE XL does not support the DRT number/unit number hardware address. When the specified file is located on a disc, the value returned by the optional *haddr* parameter of the FGETINFO intrinsic always represents DRT number 8, unit number 0. For MPE XL disc files, the DRT number/unit number representation for a hardware device address has been replaced by device path, available through *itemvalue* 75 of the FFILEINFO intrinsic.
	ACTION	Remove dependency upon the *haddr* parameter of FGETINFO. Evaluate why your program used this value. You may be able to perform an equivalent action using the device path returned by *itemvalue* 75 of FFILEINFO.
413	**MESSAGE**	FGETINFO *physcount* parameter.
	MODE	Compatibility Mode and Native Mode incompatibility detected by RTM (Event Class 4).
	CAUSE	Because of enhancements to the MPE XL operating system, the value returned by the optional *physcount* parameter of the FGETINFO intrinsic is the physical I/O count only for files residing on devices other than disc; for files residing on disc, *physcount* returns the logical I/O count.
	ACTION	If your code relies upon the value returned in the *physcount* parameter of FGETINFO for files residing on disc,you must either remove the dependency, or modify your code to rely instead upon the logical I/O count of disc files.

415	**MESSAGE**	FGETINFO *extsize* parameter.
	MODE	Compatibility Mode and Native Mode incompatibility detected by RTM (Event Class 4).
	CAUSE	On MPE XL, the file system manages extent allocation, where variable numbers of extents, each of variable size, are allocated based upon a need basis. There is no fixed extent size for for a file. Extent information specified at file creation is meaningless on an MPE XL-based computer system and is stored in the file label only to maintain backward compatibility with MPE V/E-based computer systems. Therefore, the optional *extsize* parameter of the FGETINFO intrinsic returns a value that is meaningless on MPE XL.
	ACTION	If your code relies upon the extent size returned by the *extsize* parameter of FGETINFO, you will have to modify your code to remove dependency upon extent size.
416	**MESSAGE**	FGETINFO *numextents* parameter.
	MODE	Compatibility Mode and Native Mode incompatibility detected by RTM (Event Class 4).
	CAUSE	On MPE XL, the file system manages extent allocation, where variable numbers of extents, each of variable size, are allocated based upon a need basis. There is no practical limit to the number of extents for a file. Extent information specified at file creation is meaningless on an MPE XL-based computer system and is stored in the file label only to maintain backward compatibility with MPE V/E-based computer systems. Therefore, the optional *numextents* parameter of the FGETINFO intrinsic returns a value that is meaningless on MPE XL.
	ACTION	If your code relies upon the number of extents returned by the *numextents* parameter of FGETINFO, you will have to modify your code to remove dependency upon number of extents.

500	**MESSAGE**	Some FCONTROL *controlcodes* are not valid on MPE XL.
	MODE	Compatibility Mode and Native Mode incompatibility detected by OCA.
	CAUSE	Some values that may be contained in the required *param* parameter of the FCONTROL intrinsic have either been modified or have no meaning on an MPE XL-based computer system. Affected *params* are those associated with *controlcodes* 3 and 48.
	ACTION	If OCA reports this event, enable Event Class 5 of RTM to detect when FCONTROL calls *controlcode* 3 and 48.
503	**MESSAGE**	FCONTROL *controlcode* 3 Read hardware status word.
	MODE	Compatibility Mode and Native Mode incompatibility detected by RTM (Event Class 5).
	CAUSE	Because of differences between HP Precision Architecture and the architecture of MPE V/E-based computer systems, the system hardware status word returned by FCONTROL has no meaning on MPE XL. A meaningless value is returned in the required *param* parameter of the FCONTROL intrinsic when a value of 3 is passed in required *controlcode* parameter.
	ACTION	Remove dependencies upon *controlcode* 3 of FCONTROL.

548	**MESSAGE**	FCONTROL *controlcode* 48 Enable/Disable software interrupts.
	MODE	Compatibility Mode and Native Mode incompatibility detected by RTM (Event Class 5).
	CAUSE	HP Precision Architecture is a 32-bit architecture, while the architecture of MPE V/E-based computer systems is a 16-bit architecture. In Native Mode, memory addresses represented as pointers or PLABELs must be 32-bit entities, while in MPE V/E and in Compatibility Mode, pointers and PLABELs must be 16-bit entities. The PLABEL of a software interrupt handler may be passed through the required *param* parameter of FCONTROL if the required *controlcode* parameter passes the value 48.
	ACTION	Redeclare the variable you pass through the *param* parameter of FCONTROL to be a 32-bit entity when you also pass a *controlcode* value of 48.
601	**MESSAGE**	The PTAPE intrinsic is not available on MPE XL.
	MODE	Compatibility Mode and Native Mode incompatibility detected by OCA and RTM (Event Class 6).
	CAUSE	Modifications have been made to the attached peripheral environment of MPE XL, resulting in paper tape readers being unsupported on MPE XL-based computer systems. The PTAPE intrinsic is not supported on MPE XL-based computer systems.
	ACTION	Remove all PTAPE invocations and dependent code. If you are using PTAPE for uses other than reading from paper tape devices, you must determine the original need for the call to PTAPE and develop alternative solutions.

602	**MESSAGE**	The FCARD intrinsic is not available on MPE XL.
	MODE	Compatibility Mode and Native Mode incompatibility detected by OCA and RTM (Event Class 6).
	CAUSE	Modifications have been made to the attached peripheral environment of MPE XL; the HP7260 series of card readers is not supported on MPE XL-based computer systems. Because the FCARD intrinsic accesses only these card readers, FCARD likewise is not supported on MPE XL.
	ACTION	Remove all FCARD invocations and dependent code.
800	**MESSAGE**	CS data communication routines are not available on MPE XL.
	MODE	Compatibility Mode and Native Mode incompatibility detected by OCA.
	CAUSE	Modifications have been made to the attached peripheral environment of MPE XL; CS data communication routines are not supported on MPE XL.
	ACTION	Remove all CS routines and dependent code. Analyze the original reasons for having the CS routines and develop alternative solutions. Equivalent functionality on MPE XL may be available through the NS 3000/XL subsystem.

801	**MESSAGE**	PTOP data communication routines are not available on MPE XL.
	MODE	Compatibility Mode and Native Mode incompatibility detected by OCA.
	CAUSE	Modifications have been made to the attached peripheral environment of MPE XL; PTOP data communication routines are not supported.
	ACTION	Remove all PTOP routines and dependent code. Analyze the original reasons for having the PTOP routines and develop alternative solutions. For example, you may be able to immediately replace PTOP routines with equivalent NetIPC routines available in the NS 3000/V subsystem. Equivalent functionality on MPE XL may be available through the NS 3000/XL subsystem.

801	**MESSAGE**	STARTSESS *logonstr* parameter may specify an unsupported TERMTYPE.
	MODE	Compatibility Mode and Native Mode incompatibility detected by RTM (Event Class 8).
	CAUSE	Modification have been made to the attached peripheral environment of MPE XL; the following TERMTYPEs are not supported: 4, 6, 12, 13, 14, 15, 16, 19, and 20. The required *logonstr* parameter of the STARTSESS intrinsic may specify an unsupported TERMTYPE.
	ACTION	If the required *logonstr* parameter of STARTSESS includes a TERMTYPE, you may have to modify the variable passed through *logonstring* to specify a supported TERMTYPE (10, 18, 21, and 22).

802	**MESSAGE**	RFA (1000 to 3000) data communication routines are not available on MPE XL.
	MODE	Compatibility Mode and Native Mode incompatibility detected by OCA.
	CAUSE	Modification have been made to the attached peripheral environment of MPE XL; RFA and DEXEC intrinsics are not supported on MPE XL.
	ACTION	Remove all RFA and DEXEC routines and dependent code. Analyze the original reasons for having the these routines and develop alternative solutions. Equivalent functionality on MPE XL may be available through the NS 3000/XL subsystem.

802	**MESSAGE**	GENMESSAGE may encounter pointer problems in MPE XL Native Mode.
	MODE	Native Mode incompatibility detected by RTM (Event Class 8).
	CAUSE	HP Precision Architecture is a 32-bit architecture, while the architecture of MPE V/E-based computer systems is a 16-bit architecture. In Native Mode, memory addresses represented as pointers or PLABELs must be 32-bit entities, while in MPE V/E and in Compatibility Mode, pointers and PLABELs must be 16-bit entities. The GENMESSAGE intrinsic may use pointers.
	ACTION	If you use the optional *param* parameter of GENMESSAGE, you must redeclare the variable you pass through it to be a 32-bit entity.

850	**MESSAGE**	Data capture intrinsics are not available on MPE XL.
	MODE	Compatibility Mode and Native Mode incompatibility detected by OCA.
	CAUSE	Data capture intrinsics are not supported on MPE XL.
	ACTION	Analyze the original reasons your program required the data capture intrinsics and determine if equivalent functionality is available using VPLUS intrinsics available on MPE V/E.
851	**MESSAGE**	Printer Support Package (PSP) intrinsics are not callable from Native Mode on MPE XL.
	MODE	Native Mode incompatibility detected by OCA.
	CAUSE	On MPE XL PSP intrinsics are not directly callable from a Native Mode program.
	ACTION	On MPE XL, programs accessing the PSP intrinsics must either run in Compatibility Mode, or use the Switch subsystem from Native Mode to access the PSP intrinsics.

E

Notes on Migrating VPLUS

Introduction

For the most part, VPLUS operates the same on MPE XL-based machines as it does on MPE V/E-based machines. This is because all of VPLUS operates in Compatibility Mode. Native Mode stubs are supplied for all of the VPLUS intrinsics so that they may be called from Native Mode applications. The following VPLUS utilities are available in Compatibility Mode:

- FORMSPEC
- REFSPEC
- REFORMAT
- ENTRY

Migration Issues

Terminal Configuration

All terminals must be configured for XON/XOFF handshaking.

Supported Terminals

The following terminals are supported:

- HP2392A
- HP2394A
- HPHP150
- HP2393A
- HP2397A
- HP2625A

- HP2628A

The following terminals are supported if the firmware revision conditions in parentheses are met:

- HP2622A (rev D 1818-3199)
- HP2623A (rev B 1818-3223)
- HP2624B (rev B 1818-3139)
- HP2627A (rev B 1818-3487)

Unsupported Terminals

The following terminals are not supported:

- HP2382A
- HP2621A
- HP2621B
- HP2621P
- HP2626A
- HP2626W
- HP264x
- HP3075
- HP3076
- HP3081A

Pascal Integers

In HP Pascal/XL all references to integers in the COMAREA must be of type SHORTINT (a new 2-byte compiler-defined type for the HP Pascal/XL compiler) as opposed to type INTEGER (-32768..32767) which is interpreted as 4 bytes.

Data Alignment

VPLUS intrinsics continue to expect data that is aligned on 16-bit boundaries. For example, in a Pascal program, the record structures declared for the VPLUS INFO intrinsics may need to be packed to ensure 16-bit data alignment.

Real Data Types

Real data type conversions continue to assume Hewlett Packard real format for VGETREAL, VGETLONG, VPUTREAL, and VPUTLONG. New IEEE conversion intrinsics (VGETIEEEREAL, VGETIEEELONG, VPUTIEEEREAL, and VPUTIEEELONG) are available.

NM Stubs

Programs with incorrect parameter values which run successfully in Compatibility Mode may not run in Native Mode. Many of these incorrect parameters will be detected by the native mode stubs which have an additional level of parameter checking not present in the VPLUS intrinsics.

Call Intrinsic

In general, the Native Mode COBOL II/XL compiler requires that intrinsics be called with the Call Intrinsic mechanism. The only exceptions to this are the VPLUS and IMAGE intrinsic calls. For compatibility reasons the COBOL II/XL compiler will recognize these calls and assume CALL INTRINSIC. A warning will be issued when using Call Intrinsic on MPE V/E (see Warning).

WARNING

On MPE V, the Call Intrinsic mechanism examines the SPLINTR file and determines the type of addressing that the intrinsic expects.

VPLUS examines the language id and, if it is 0 (i.e. COBOL), VPLUS assumes all the addresses are word addresses. If the Call Intrinsic mechanism is used, this is an invalid assumption and erroneous results are likely. Therefore, the programmer should "fool" VPLUS by using a language ID of 5 (PASCAL).

On MPE XL, the COBOL II/XL compiler generates byte addresses and the VPLUS stub will prevent VPLUS from doubling the address (in an attempt to convert from word addresses to byte addresses).

It is recommended that customers preparing for migration of COBOL applications use the Call Intrinsic construct and a language ID of 5.

Non-VPLUS I/O

Often, application designers resort to using FREAD, FWRITE, and input/output verbs of a programming language to display and retrieve information from the terminal within the VPLUS context. The situation becomes more complex when the application also controls the terminal settings as well as the driver settings. A likely end result is an application which runs in a restricted environment such as on an ATP and on one kind of terminal only. The same application may not run in a different environment such as over X.25 or on a different kind of terminal. Four new VPLUS procedures are provided to assist the application designer in producing more portable and easier to maintain applications. These four procedures are:

- VTURNOFF
- VTURNON
- VBLOCKWRITE
- VBLOCKREAD

VTURNON/VTURNOFF

VOPENTERM configures the driver and the terminal for block mode access. VCLOSETERM configures the terminal and driver for character mode access. These procedures initialize the terminal in many ways including clearing the screen image. VTURNON and VTURNOFF reconfigure the terminal and driver but leave the terminal screen image intact. The procedure VTURNOFF allows the application to momentarily switch to character mode from block mode without disturbing the screen. A call to VTURNON will reconfigure the terminal back to block mode.

For example, a program which accesses a printer slaved off a terminal while that terminal is using VPLUS should use VTURNON and VTURNOFF. VTURNOFF would be called, followed by the FWRITEs to the printer. Then VTURNON should be called.

Refer to pages E-14 through E-18 for syntax and further discussion of these two intrinsics.

VBLOCKREAD/VBLOCKWRITE

The VPLUS terminal input/output procedures for terminal presentation and retrieval such as VSHOWFORM and VREADFIELDS are restricted to input/output associated with a predefined form and are used in block/format mode. VBLOCKREAD and VBLOCKWRITE provide the means for the application to communicate directly between the application buffer and the terminal. No form is associated with these procedures.

An example use of this new pair of procedures would be to transfer an unedited block of text to the terminal so that the end-user can edit the block of text using the special terminal keys. The edited text can then be read directly into the application buffer. Other more complex operations can also be performed using these procedures in conjunction with a form (for example, appending many lines of text to the end of a form displayed on the screen). A very complex screen would be one that involves a VPLUS form and an additional area for unformatted input/output. This kind of use is exemplified by the Field Menu in FORMSPEC, where the bottom half of the screen is used for unformatted input/output. These combinations of operations are extremely complex. Therefore, they should be used by advanced users only.

Use of VBLOCKWRITE to control terminal settings related to communications protocol, such as the terminal straps should be avoided as much as possible. The procedures VOPENTERM, VCLOSETERM, VTURNON, and VTURNOFF are more appropriate for a systematic approach to controlling driver settings.

The recommended practice for programming VPLUS applications is to use only VPLUS procedures to perform terminal input/output operations when within the VPLUS context. If the application must perform terminal input/output operations using other input/output methods, either VCLOSETERM or VTURNOFF must be used to switch out of the VPLUS context before doing so. Following this simple programming guideline assures compatibility of the application with different HP 3000 drivers and terminals and improves the maintainability of the application.

Refer to pages E-19 through E-24 for syntax and further discussion.

VGETIEEEREAL

Copies character coded data from a specified field in the form data buffer in memory into an application, converting the numeric value to IEEE floating point format.

Syntax

```
VGETIEEEREAL {COMAREA,FIELDNUM,VARIABLE}
```

Parameters

COMAREA

Must be COMAREA name specified when the forms file was opened with VOPENFORMF. If not already set, the following COMAREA items must be set before calling VGETIEEEREAL:

- CSTATUS. Set to zero.
- COMAREALEN. Set to total number of 2-byte words in COMAREA.

VGETIEEEREAL may set the following COMAREA fields:

- CSTATUS. Set to nonzero value if call is unsuccessful or if field's error flag is set.

FIELDNUM

Two-byte integer variable containing the field number assinged by FORMSPEC to the field in the data buffer from which the value is retrieved. Note that the field identified by FIELDNUM may not be longer than 80 characters.

VARIABLE

Variable of type IEEE floating-point real (32-bits) in which the converted value will be placed.

Discussion

This procedure reads the field identified by its field number from the data buffer. (Note that this field number is a unique number assigned to each field by FORMSPEC and is totally independent of the field position in the data buffer.) The field's value must be numeric, but its data type need not be. That is, numbers in a character type field can be converted.

The numeric value, stored in the buffer in character coded form, is converted to IEEE format and then moved to the variable in the application. If errors occur during

conversion, CSTATUS is set to an error code. If the requested field's error flag is set, its value is moved to the variable, but CSTATUS is set to an error code.

Note that a field's number never changes unless the batch command, RENUMBER, is used. It is not changed even if the position of the field in the form is changed, or its length or other characteristics are changed. The field number should not be confused with the position of the field in the data buffer, which is based on the field position within the form. Thus, the field number provides a way to locate fields regardless of their position.

Examples

The following examples illustrate a call to VGETIEEEREAL:

COBOL:

```
CALL "VGETIEEEREAL" USING COMAREA FIELDNUM VARIABLE.
```

FORTRAN:

```
CALL VGETIEEEREAL(COMAREA,FIELDNUM,VARIABLE)
```

Pascal:

```
VGETIEEEREAL(COMAREA,FIELDNUM,VARIABLE);
```

VGETIEEELONG

Copies character-coded data from a specified field in the form data buffer in memory into an application, converting the numeric value to IEEE long floating point format.

Syntax

```
VGETIEEELONG {COMAREA,FIELDNUM,VARIABLE}
```

Parameters

COMAREA

Must be the COMAREA name specified when the forms file was opened with VOPENFORMF. If not already set, the following COAMREA items must be set before calling VGETIEEELONG:

- CSTATUS. Set to zero.
- COMAREALEN. Set to total number of 2-byte words in COMAREA.

VGETIEEELONG may set the following COMAREA fields:

- CSTATUS. Set to non-zero value if call is unsuccessful or if field's error flag is set.

FIELDNUM

Two-byte integer variable containing the field number assigned by FORMSPEC to the field in the data buffer from which the value is retrieved. Note that the field identified by FIELDNUM may not be longer than 80 characters.

VARIABLE

Variable of type IEEE long floating-point real (64-bits) in which the converted value will be placed.

Discussion

This procedure reads the field identified by its field number from the data buffer. (Note that this field number is a unique number assigned to each field by FORMSPEC and is totally independent of the field position in the data buffer). The field's value must be numeric, but its data type need not be. That is, numbers in a character type field can be converted.

The numeric value, stored in the buffer in character coded form, is converted to IEEE long format and then moved to the variable in the application. If errors occur during

conversion, CSTATUS is set to an error code. If the requested field's error flag is set, its value is moved to the variable, but CSTATUS is set to an error code.

Note that a field's number never changes unless the batch command, RENUMBER, is used. It is not changed even if the position of the field in the form is changed, or its length or other characteristics are changed. The field number should not be confused with the position of the field in the data buffer, which is based on the field position within the form. Thus, the field number provides a way to locate fields regardless of their position.

Examples

The following examples illustrate a call to VGETIEEELONG:

COBOL:

```
CALL "VGETIEEELONG" USING COMAREA FIELDNUM VARIABLE.
```

FORTRAN:

```
CALL VGETIEEELONG(COMAREA,FIELDNUM,VARIABLE)
```

Pascal:

```
VGETIEEELONG(COMAREA,FIELDNUM,VARIABLE);
```

VPUTIEEEREAL

Writes a floating point number in IEEE standard format from an application to a specified field in the form data buffer in memory, converting the value to character set coded external representation.

Syntax

```
VPUTIEEEREAL {COMAREA,FIELDNUM,VARIABLE}
```

Parameters

COMAREA

Must be the COMAREA name specified when the forms file was opened with VOPENFORMF. If not already set, the following COMAREA items must be set before calling VPUTIEEEREAL:

- CSTATUS. Set to zero.
- COMAREALEN. Set to total number of 2-byte words in COMAREA.

VPUTIEEEREAL may set the following COMAREA fields:

- CSTATUS. Set to non-zero value if call is unsuccessful.
- NUMERRS. Will be decremented if new value replaces the value of a field with an error.

FIELDNUM

Two-byte integer variable containing the field number assigned by FORMSPEC to the field in the data buffer to which the is written. Note that the field identified by FIELDNUM may not be longer than 80 characters.

VARIABLE

Variable of type IEEE floating-point real (32-bits) that contains the value to be converted to character set coded external representation and copied to a field in the data buffer.

Discussion

This procedure converts an IEEE floating-point real number to its character-coded form and writes the converted value to a particular field in the data buffer, right justified. The exact format of the written data depends on the type of the destination field.

For example, if the number "34.56" were to be put in a field of type DIG, the result would be "34", since a field of type DIG may only contain integer values. The destination field is identified by the field number assigned by FORMSPEC. The field to which the value is written must be defined as a numeric field (type NUM, IMP, or DIG).

Note that a field's number never changes unless the batch command, RENUMBER, is used. It is not changed even if the position of the field in the form is changed, or its length or other characteristics are changed. The field number should not be confused with the position of the field in the data buffer, which is based on the field position within the form. Thus, the field number provides a way to locate fields regardless of their position. If the specified field had an error, then VPUTIEEEREAL will clear the field's error flag and decrement NUMERRS.

Examples

The following examples illustrate a call to VPUTIEEEREAL:

COBOL:

```
CALL "VPUTIEEEREAL" USING COMAREA FIELDNUM VARIABLE.
```

FORTRAN:

```
CALL VPUTIEEEREAL(COMAREA,FIELDNUM,VARIABLE)
```

Pascal:

```
VPUTIEEEREAL(COMAREA,FIELDNUM,VARIABLE);
```

VPUTIEEELONG

Writes a floating point number in IEEE long standard format from an application to a specified field in the form data buffer in memory, converting the value to character set coded external representation.

Syntax

```
VPUTIEEELONG {COMAREA,FIELDNUM,VARIABLE}
```

Parameters

COMAREA

Must be the COMAREA name specified when the forms file was opened with VOPENFORMF. If not already set, the following COMAREA items must be set before calling VPUTIEEELONG:

- CSTATUS. Set to zero.
- COMAREALEN. Set to total number of 2-byte words in COMAREA.

VPUTIEEELONG may set the following COMAREA fields:

- CSTATUS. Set to non-zero value if call is unsuccessful.
- NUMERRS. Will be decremented if new value replaces the value of a field with an error.

FIELDNUM

Two-byte integer variable containing the field number assigned by FORMSPEC to the field in the data buffer to which the value is written. Note that the field identified by FIELD may not be longer than 80 characters.

VARIABLE

Variable of type IEEE long floating-point real (64-bits) that contains the value to be converted to character set coded external representation and copied to a field in the data buffer.

Discussion

This procedure converts an IEEE long floating-point real number to its character-coded form and writes the converted value to a particular field in the data buffer, right justified. The exact format of the written data depends on the type of the destination field.

For example, if the number "34.56" were to be put in a field of type DIG, the result would be "34", since a field of type DIG may only contain integer values. The destination field is identified by the field number assigned by FORMSPEC. The field to which the value is written must be defined as a numeric field (type NUM, IMP, or DIG).

Note that a field's number never changes unless the batch command, RENUMBER, is used. It is not changed even if the position of the field in the form is changed, or its length or other characteristics are changed. The field number should not be confused with the position of the field in the data buffer, which is based on the field position within the form. Thus, the field number provides a way to locate fields regardless of their position. If the specified field had an error, then VPUTIEEELONG will clear the field's error flag and decrement NUMBERS.

Examples

The following examples illustrate a call to VPUTIEEELONG:

COBOL:

```
CALL "VPUTIEEELONG" USING COMAREA FIELDNUM VARIABLE.
```

FORTRAN:

```
CALL VPUTIEEELONG(COMAREA,FIELDNUM,VARIABLE)
```

Pascal:

```
VPUTIEEELONG(COMAREA,FIELDNUM,VARIABLE);
```

VTURNOFF

Turn off VPLUS block mode and enable character mode access without disturbing the terminal screen.

Syntax

```
VTURNOFF {COMAREA}
```

Parameters

COMAREA

Must be comarea named when file was opened by VOPENTERM. If not already set, the following COMAREA items must be set before calling VTURNOFF.

- CSTATUS. Set to zero.
- COMAREALEN. Set to total number of 2-byte words of COMAREA.

VTURNOFF may set the following COMAREA items:

- CSTATUS. Set to nonzero value if call is unsuccessful.
- FILERRNUM. Set to file error code if MPE file error.

Discussion

VTURNOFF is used for momentarily switching from VPLUS block mode to character mode. This procedure is designed for use after a terminal has been previously opened by VOPENTERM or after a VTURNON.

VTURNOFF reconfigures the terminal and driver for character mode operations without disturbing the screen image on the terminal. The following operations normally performed in VCLOSETERM are not performed in VTURNOFF:

- clear local form storage
- enable the USER/SYSTEM keys
- disable touch reporting, delete touch fields
- clear screen
- unlock keyboard
- close terminal file

Note especially that TURNOFF does not close the terminal file. To close the file and completely reset the driver and the terminal, VCLOSETERM must be used.

Examples

The following examples illustrate a call to VTURNOFF:

COBOL:

```
CALL "VTURNOFF" USING COMAREA.
```

BASIC:

```
200 CALL VTURNOFF(C(*))
```

FORTRAN:

```
CALL VTURNOFF(COMAREA)
```

SPL:

```
VTURNOFF(COMAREA);
```

Pascal:

```
VTURNOFF(COMAREA);
```

VTURNON

Turn on VPLUS block mode without disturbing the terminal screen.

Syntax

```
VTURNON {COMAREA,TERMFILE}
```

Parameters

COMAREA

The COMAREA name must be unique for each open forms file. The COMAREA must be the same COMAREA used in VOPENTERM. The following COMAREA items must be set before the call if not already set:

- CSTATUS. Set to zero.
- LANGUAGE. Set to code that identifies the programming language of the calling program.
- COMAREALEN. Set to total number of 2-byte words in COMAREA.

VTURNON may set the following COMAREA fields:

- CSTATUS. Set to nonzero value if call is unsuccessful.
- FILERRNUM. Set to file error code if MPE file error.
- FILEN. Set to MPE file number of terminal
- IDENTIFIER. Set to appropriate VPLUS/V terminal ID.
- LAB'INFO. Set to appropriate number and length of labels.

TERMFILE

Must be the same terminal file name used in VOPENTERM.

Discussion

VTURNON is normally used in an application when the terminal is already opened by VOPENTERM, and VTURNOFF was called to switch out of VPLUS block mode. VTURNON switches the application back to VPLUS block mode without disturbing the image on the terminal screen.

VTURNON reconfigures the terminal and the driver without performing the following operations which are normally performed by VOPENTERM:

- initialize local form storage
- clear screen
- enable the USER function keys
- disable or enable the USER/SYSTEM key as specified in the SHOWCONTROL word

Unlike VOPENTERM, VTURNON will not ask you to press the BLOCK MODE key, if you are using an HP2640B or HP2644 terminal when the terminal is not in block mode.

Examples

The following examples illustrate a call to VTURNON:

COBOL:

```
CALL "VTURNON" USING COMAREA, T1.
```

BASIC:

```
90 T1$=" "

100 CALL VTURNON(C(*),T1$)
```

FORTRAN:

```
T1="  "

VTURNON(COMAREA,T1);
```

SPL:

```
MOVE T1:="    ";

VTURNON(COMAREA,T1);
```

Pascal:

```
T1:=' ';

VUTURNON(COMAREA,T1);
```

VBLOCKWRITE

Write a block of characters to a terminal in block mode.

Syntax

```
VBLOCKWRITE {COMAREA,BUF,LEN,TMODE,LOC,TC}
```

Parameters

COMAREA

The following COMAREA fields must be set before calling VBLOCKWRITE if not already set:

- LANGUAGE. Set to code identifying the programming language of the calling program.
- COMAREALEN. Set to total number of 2-byte words in COMAREA.

VBLOCKWRITE may set the following COMAREA fields:

- CSTATUS. Set to nonzero value if call is unsuccessful.
- FILERRNUM. Set to file error code if MPE file error.

BUF

Byte array containing characters to be written to the terminal.

LEN

Number of bytes in the BUF array (2-byte integer).

TMODE

Terminal Mode (2-byte integer).

0 = do not change terminal mode

1 = change to format mode

2 = change to unformatted mode

LOC

Start position of write (array of two 2-byte integers). Absolute cursor addressing is not allowed in format mode. An error will be returned.

[0] [0] = home cursor before WRITE

[x] [y]= start from absolute row x column y (not allowed in format mode)

[-1][0] = start from current cursor position

[-2] [0] = start from first available line of display memory, for example, the first available line after the end of a previous form (not allowed in format mode)

TC

Terminal Control (2-byte integer)

0 = do not lock keyboard at the beginning of write ,unlock at the end of write

1 = lock keyboard at the beginning of write ,unlock at the end of write

Discussion

This procedure writes the content of a user buffer to a terminal. TMODE options can be used to change the terminal to format or unformatted mode before the write. LOC options allow the programmer to specify the position on the screen where the write is to begin. Terminal Control (TC) options can be used to control keyboard lock for the protection of data as it is being written to the terminal. TC = 1 is recommended for applications which do multiple writes to the terminal with no intervening reads. Procedures such as VBLOCKREAD or VREADFIELDS lock the keyboard as soon as the terminal begins transmitting data when triggered by the ENTER key or a function key.

VOPENTERM must be called before using this procedure. This procedure is intended only for advanced programmers who are proficient with terminal control operations and VPLUS terminal settings. Terminal keyboard operations such as PREV PAGE and NEXT PAGE can be performed programmatically by sending the appropriate escape sequences to the terminal via VBLOCKWRITE. VBLOCKWRITE can also be used to write large blocks of unformatted text or multiple report lines in between uses of predefined VPLUS forms. To assure portability of the application from one driver to another, alteration of terminal straps using VBLOCKWRITE is not recommended. See VTURNON and VTURNOFF for more information on how to switch between character mode and block mode without disturbing the screen.

Examples

The following examples illustrate a call to VBLOCKWRITE:

COBOL:

```
CALL "VBLOCKWRITE" USING COMAREA @BUF LEN TMODE LOC TC.
```

BASIC:

```
CALL VBLOCKWRITE(C(*),B1$,L1,M1,L(*),T1)
```

FORTRAN:

```
CALL VBLOCKWRITE(COMAREA,BUF,LEN,TMODE,LOC,TC)
```

SPL:

```
VBLOCKWRITE(COMAREA,BUF,LEN,TMODE,LOC,TC);
```

Pascal:

```
VBLOCKWRITE(COMAREA,BUF,LEN,TMODE,LOC,TC);
```

VBLOCKREAD

Read a block of characters from a terminal in block mode.

Syntax

```
VBLOCKREAD {COMAREA,BUF,LEN,ACTLEN,TMODE,LOC,BC,TC}
```

Parameters

COMAREA

The following COMAREA fields must be set before calling VBLOCKREAD if not already set:

- LANGUAGE. Set to code identifying the programming language of the calling program.
- COMAREALEN. Set to total number of 2-byte words in COMAREA.

VBLOCKREAD may set the following COMAREA fields:

- CSTATUS. Set to nonzero value if call is unsuccessful.
- FILERRNUM. Set to file error code if MPE file error.

BUF

Byte array to receive data from terminal.

LEN

Maximum number of bytes to read from terminal (2-byte integer).

ACTLEN

Actual number of bytes read from terminal (2-byte integer).

TMODE

Terminal setting at the time of read (2-byte integer).

1 = assume terminal is in format mode

2 = assume terminal is in unformatted mode

LOC

Array of two 2-byte integers containing start position of read.

[0] [0] = home cursor before read

[-1] [0] = start from current cursor position

BC

Buffer Control (2-byte integer) - not currently used. Must initialize to zero.

TC

Terminal Control (2-byte integer) - not currently used. Must initialize to zero.

Discussion

This procedure reads a block of data from the terminal with a number of options. There are two major differences between VREADFIELDS and this procedure. First, it provides more options for reading data from the terminal. Second, data read is returned directly to the application buffer. There is no VPLUS form associated with the read.

Like the companion intrinsic VBLOCKWRITE, this procedure is recommended only for advanced programmers who are proficient with terminal input/output. VOPENTERM must be called before using VBLOCKREAD. The keyboard must be unlocked before calling VBLOCKREAD (See keyboard unlock options in VBLOCKWRITE). VBLOCKREAD will lock the keyboard immediately after the ENTER or a function key is pressed to ensure data integrity.

This procedure is primarily designed for unformatted reads. For users who do not use VREADFIELDS but use VBLOCKREAD to read in data in format mode, the application data interpretation algorithm should accommodate both MDT (Modified Data Tag) and non-MDT inputs. When MDT is on, unmodified blanks and data are not transmitted from the terminal. Refer to the appropriate terminal reference manuals for further explanation of the MDT feature.

Examples

The following examples illustrate a call to VBLOCKREAD:

COBOL:

```
CALL "VBLOCKREAD" USING COMAREA @BUF LEN ACTLEN TMODE LOC BC TC.
```

BASIC:

```
CALL VBLOCKREAD(C(*),B1$,L1,L2,M1,L(*),U1,U2)
```

FORTRAN:

```
CALL VBLOCKREAD(COMAREA,BUF,LEN,ACTLEN,TMODE,LOC,BC,TC)
```

SPL:

```
VBLOCKREAD(COMAREA,BUF,LEN,ACTLEN,TMODE,LOC,BC,TC);
```

Pascal:

```
VBLOCKREAD(COMAREA,BUF,LEN,ACTLEN,TMODE,LOC,BC,TC);
```

F

Tools for Application Development

Introduction

Application development tools for the 900 Series HP 3000 computer family are very similar to those available on MPE V/E-based systems.

These tools include:

- A wide range of programming languages, including optimizing compilers.
- Run-time library support.
- Toolset/XL (an integrated application development environment).
- Debugging tools.
- Database tools.
- Forms design and screen handling tools.
- Report generation tools.
- System Dictionaries.
- Distributed data processing.

Table F-1 outlines the availability of Hewlett-Packard products for programmers on MPE XL. Product names with the suffix "/XL" are specifically designed for use in Native Mode. All product names that previously had the suffix "/3000" now have the suffix "/V". This indicates they were designed for use on MPE V/E, but will also operate on MPE XL in Compatibility Mode.

NOTE

Although a limited number of compilers are available in Compatibility Mode, MPE XL provides run-time support, via Compatibility Mode libraries, for all HP-supported MPE V/E-based languages. Thus, any program that was compiled on an MPE V/E-based system using an HP supported compiler, can be executed in Compatibility Mode on an MPE XL-based system.

Table F-1. MPE XL Application Development Tools

MPE V/E	MPE XL CM	MPE XL NM
HP FORTRAN 77/V	HP FORTRAN 77/V (F)	HP FORTRAN 77/XL
FORTRAN 66/V	FORTRAN 66/V (F)	NA
COBOL II/V	COBOL II/V (F)	COBOL II/XL
HP Pascal/V (F)	HP Pascal/V (F)	HP Pascal/XL
Pascal/V	Pascal/V (F)	NA
RPG/V	RPG/V	RPG/XL (F)
HP Business BASIC/V	HP Business BASIC/V	HP Business BASIC/XL (F)
BASIC/V	BASIC/V (F)	NA
SPL/V	SPL/V	NA
NA	NA	HP C/XL (F)
NS 3000/V	NA	NS 3000/XL
TurboIMAGE/V	NA	TurboIMAGE/XL
IMAGE/V	NA	NA
HP SQL/V	NA	NA
NA	NA	ALLBASE/XL
KSAM/V	KSAM/V	KSAM/XL (F)
Toolset/V	NA	HP Toolset/XL
Transact/V	Transact/V	Transact/XL (F)
Report/V	Report/V	NA
Business Report Writer/V	Business Report Writer/V	Business Report Writer/XL (F)
VPLUS/V	VPLUS/V	VPLUS/XL (F)
HP System Dictionary/V	NA	HP System Dictionary/XL
Dictionary/V	Dictionary/V	NA
Inform/V	Inform/V	Inform/XL (F)
HP Access Central	HP Access Central	NA
DEBUG	DEBUG/XL	DEBUG/XL

NA - Not currently planned
(F) - Planned for a future release of MPE XL

Index

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X