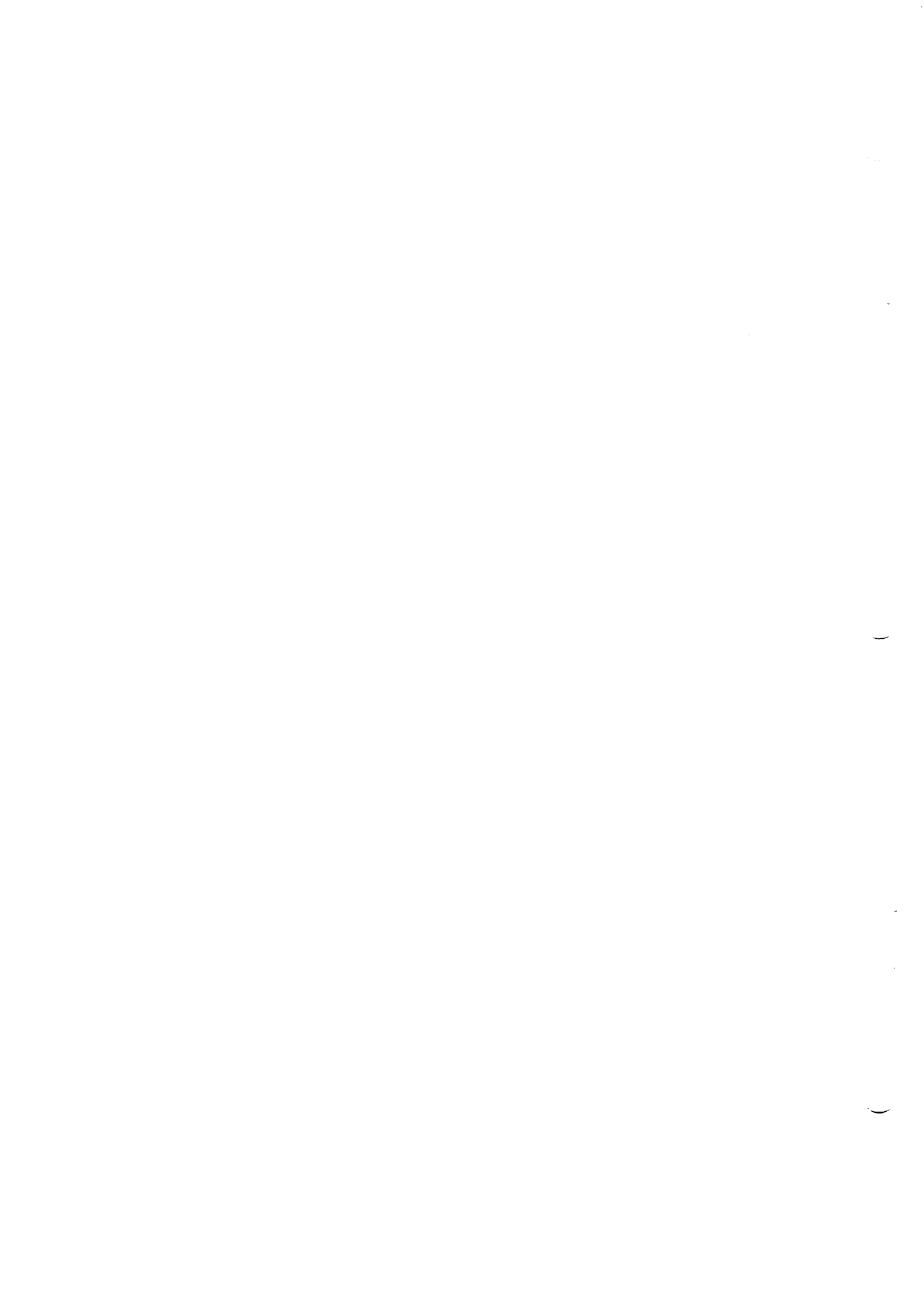


GC20-1792-1
File No. S370-34

Systems

**System/3 to Disk Operating
System/Virtual Storage
Extended (DOS/VSE)
Conversion Guide**

IBM



Systems

System/3 to Disk Operating System/Virtual Storage Extended (DOS/VSE) Conversion Guide

This guide is intended to assist the System/3 user who is converting to a DOS/VSE system, to be used as the primary reference document for the conversion. It presents a comparison of System/3 and DOS/VSE features and gives procedures for converting to DOS/VSE, either manually or with programming tools. Through numerous examples, the user can see how to change his current programs, files, and operation control language to comparable DOS/VSE programs, files and job control language. Where comparable functions do not exist, an alternate approach is suggested. The guide also presents a suggested migration plan and a checklist of things to consider, and directs the user to available migration tools.

The guide will be of interest primarily to people within the data processing department. The DP manager will find introductory information, migration guidelines, and system comparisons. The system programmer will find information on the system control program, system generation, and library considerations. The application programmer will find specific information on program conversion, control card conversion, sorts and utilities. In addition, management outside the data processing department may be interested in the overview, the advantages of DOS/VSE, and migration considerations.



Second Edition (November 1979)

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available outside the United States.

A form for reader's comments has been provided at the back of this publication. If this form has been removed, address comments to : IBM Corporation, Technical Publications, Dept. 824, 1133 Westchester Avenue, White Plains, New York 10604. IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation whatever. You may, of course, continue to use the information you supply.

© Copyright International Business Machines Corporation 1979

Introduction 1

Part I: Batch Systems. 1

Chapter 1. Overview. 2

I/O Devices. 2

Software 2

Conversion 3

Chapter 2. DOS/VSE and System/3 Characteristics and Differences. . . . 5

Disk Operating System/Virtual Storage Extended (DOS/VSE) 5

DOS/VSE System Functions 5

 Control Programs 5

 Supervisor 5

 Virtual Storage. 7

 Multiprogramming and Multitasking. 8

 Variable Partition Priority. 8

 Relocating Loader. 10

 Job to Job Communications area 10

 Automated System Initialization. 10

 Asynchronous Operator Communications 10

 Job Control. 10

 Data Management. 11

 Data Access Methods. 11

 Data Base Management 12

 Processing Programs. 12

 Service programs 12

 Spooling 12

 Library Service Programs 13

 Utilities. 13

 Linkage Editor 14

 Error Tracking 14

 Language Compilers and Assembler 14

 RPG II Compiler. 14

 DOS/VS COBOL compiler. 14

 FORTRAN Compiler 15

 Assembler. 15

 PL/I Compiler. 15

 Sort/Merge Program 15

 Additional DOS/VSE Facilities. 15

 Rush Jobs. 15

 Checkpoint/Restart 16

 System Generation. 16

 Data Communications. 16

 Assigning I/O Devices. 16

 Job Duration Printout. 17

 Job Accounting 17

Chapter 3. Transition Planning 18

Objectives 18

Considerations 19

 Program Conversion 19

 Automated Conversion 19

 Manual Translation 19

 Reprogramming and Redesign 20

 File Conversion. 20

 Converting Card Files. 20

 Converting Tape Files. 21

 Converting Disk Files. 21

 Converting Diskette Files. 21

 Job Control Statements 21

 Utilities and Sorts. 22

Preparing the Transition22
Personnel Requirements22
Education.22
Operator Training.23
User Departments23
Standards and Control.23
System Standards23
Programming Standards.24
Operating Standards.24
Control.24
Classifying Programs and Files25
Scheduling the Transition.26
Implementing the Transition.27
Preinstallation Tasks.27
System Generation.27
Conversion of Files.28
Postinstallation Tasks28
Checklist.28
Expanding the System32
Chapter 4. Data File Conversion.33
Converting Card Files.33
96-Column Cards.33
96- to 80-Column Card Conversion34
96-Column Card to Diskette Conversion.35
Converting Tape Files.35
Converting Diskette Files.36
Chapter 5. DASD File Conversion.37
Converting Disk Files.37
Virtual Storage Access Method.37
Sequential Access Method38
ISAM and DAM access methods.38
Other Access Method Considerations38
Summary.39
Physical Disk File Factors39
Data Differences39
Conversion Procedures for Disk Files40
VSE/IBM System/3-3340 Data Import program 5746-AM3.41
Intermediate devices41
3348 Data Module Considerations.42
Unload and Reload Programs42
VSE/VSAM Installation Steps.43
Testing.43
Conversion Schedule.43
Additional Data Considerations43
Chapter 6. Program Conversion.44
Conversion Procedures.44
General Program Conversion Guidelines.45
Compiler Workfiles45
Overlay Programs46
Printer Forms Control.46
RPG II Programs.46
Major RPG II Differences47
Other RPG II Differences50
Sample Jobs.51
COBOL Programs51
General Comments52
Environment Division52
Data division.53
Procedure Division54
Sample Jobs.54
FORTRAN Programs55
FORTRAN Statements56
General Differences.57
Sample Jobs.57
Assembler Programs58
Language Differences59

General Rules.59
Sample Jobs.59
Chapter 7. OCL to JCL Conversion61
Job Concepts61
Assigning Devices.62
Cataloged Procedures63
Label Information Area64
Conversion Procedures for OCL.64
OCL and JCL Comparison65
Chapter 8. Utilities72
Conversion Procedures.72
Evaluate Utility Functions72
Manually Recode Control Cards.72
Test72
Update Documentation73
DOS/VSE Utility Programs73
System Utilities73
VSE/VSAM Access Method Services.73
VSE/Data Interfile Transfer, Testing and Operations Utility74
Comparison of Utility Functions.74
Utility Program Examples76
Disk Initialization.76
Tape Initialization.76
Hard-copy File Print77
VTOC and File Label Display.78
Single File Display.79
Alternate Block Assign and Rebuild80
Dumping and Restoring a Disk Volume to Tape.80
File Delete.82
Copy a Disk Volume to Another Disk Volume.84
File-to-File Utilities86
Chapter 9. Library considerations.87
Library Contents87
Organization of Libraries on Disk.87
Library Management89
Library Maintenance Programs89
Library Program Examples91
Library Conversion Considerations.95
Chapter 10. Sorts.96
Differences.96
Converting Sequence Specifications96
Converting OCL Statements.97
Conversion Procedure98
Sort Example99
Chapter 11. Spooling	100
VSE/POWER with optional remote job entry feature	100
Advantages of Spooling	101
General Comparison	101
Job Options.	103
Operator Control of Spooling	105
Generation of Spooling Support	106
Summary of Spooling Differences.	107
Chapter 12. Operator Control of the System	108
The Operator's Duties.	108
System-Operator Communication.	108
Operator Console	109
DISPLAY UNIT	109
Operator Keyboard.	110
Control Panel.	110
Using the Display Unit	110
DOS/VSE Messages	111
Program Messages	112
Wait States.	112

Operator Commands.	112
I/O Devices.	112
Summary of Operation Considerations.	113
 Chapter 13. System Generation.	 115
Supervisor Generation.	115
Tailoring the Libraries.	115
Planning for System Generation	115
Scheduling System Generation	116
Updating Your DOS/VSE System	116
Maintain System History program.	116
 Chapter 14. System/3 Application Packages.	 118
Types of Program Packages.	118
Program Package Profile.	118
Conversion Alternatives.	119
Source Code Conversion of the System/3 Program Package	119
Selection of an Equivalent 4300 Program Package.	119
Design and Code Functionally Equivalent Programs	119
Selected System/3 Program Packages	119
Bill of Material Processor (BOMP).	120
DATA/3	120
Job Analysis System/3.	120
Terminal Query Facility for System/3 (TQF/3)	120
System/3 DITTO	120
 Part II: Data Communications	 120
 Chapter 15. Introduction to Data Communications.	 121
CCP-CICS/VS System Overview.	122
 Chapter 16. CCP-CICS/VS Description and Summary.	 131
CICS/VS Software Components.	131
System Management.	131
Task Management.	131
Task Management Summary.	132
Storage Management	133
Storage Management Summary	133
Program Management	133
Program Management Summary	134
Time Management.	134
Time Management Summary.	135
Terminal Management.	135
Terminal Management Summary.	136
File Management.	136
Transient Data Management.	136
Temporary Storage Management	137
File/Data Storage Management Summary	137
System Service	138
Sign-on/Sign-off	138
Sign-on/Sign-off Summary	138
Master Terminal.	138
Supervisor Terminal.	138
Operator Terminal.	139
Operator Terminal Summary.	139
System Statistics.	139
Terminal Test.	139
Application Service.	140
Basic Mapping Support.	140
BMS Map Generation	140
BMS Macro Instructions	142
BMS Summary.	145
System Monitoring.	145
Trace Management	145
Dump Management.	145
System Monitoring Summary.	146
System Support	146
Preparation of the DOS/VSE Environment	146
Generation of the CICS/VS Programs	147

Entry Level System (ELS)	147
Starter System	148
Generation of CICS/VS Tables	148
Preparation of Startup Parameters.	148
System Support Summary	149
Chapter 17. CCP-CICS/VS Conversion Considerations.	150
System Conversion Considerations	150
Task Management.	150
CCP-MRT Program.	150
CCP-SRT Program.	150
Storage Management	150
Program Management	151
Terminal Management.	151
File Management.	151
Sequential Files	151
Transient Data	151
Temporary Storage.	151
System Service	152
/MSG terminal operator command.	152
MSG system operator command.	152
/NAME terminal operator command	152
/FILE terminal operator command	153
/Q terminal operator command.	153
Application Service.	153
Application Program Conversion Considerations.	158
Program Interface to CICS/VS	158
Command Translator	160
EXEC Interface Block	160
Handling Exceptional Conditions.	160
Program Interface Summary.	160
Terminal Processing.	161
Terminal Processing Examples	161
Terminal Input	161
Terminal Output - Format Screen.	162
Terminal Output - Modify Screen.	163
Terminal Output - Display Data	165
3270 Erase All Unprotected	167
3270 Print	167
HANDLE AID command	168
RETURN command	168
Terminal Processing Summary.	169
File Processing.	170
File Processing Summary.	172
Data Areas	172
Data Area Summary.	173
Program Design	173
Terminal Acquisition	173
Disk Sort Program.	174
Recovery/Restart	174
Program Design Summary	174
CCP-CICS/VS Conversion Activities.	174
Chapter 18. System/3 to 4300 Conversion Alternatives	176
Conversion Alternatives Summary.	176
Program Descriptions	176
VSE/POWER RJE.	176
Basic Telecommunications Access Method Extended Support (BTAM-ES)	177
Program Conversion/Development Aids.	177
Testing Aids	177
Analysis Aids.	178
Appendix A Sample Configurations	179
Appendix B DOS/VSE Reference Manuals	180

LIST OF FIGURES

1. Operational functions and components of DMS and DOS/VSE.....	6
2. Influence of DOS/VSE automatic storage management in a data communications system.....	9
3. Comparison of the System/3 and DOS/VSE libraries.....	13
4. System input/output files.....	17
5. Examples of a progress record sheet.....	25
6. Example of a program inventory form.....	26
7. Example of a file inventory form.....	26
8. Compiler workfiles.....	46
9. Header options.....	48
10. Extension specifications.....	48
11. Line counter specifications.....	48
12. File description specifications.....	49
13. Calculation specifications.....	49
14. Output specifications.....	50
15. DOS/VS RPG II jobs.....	51
16. DOS/VS COBOL jobs.....	55
17. FORTRAN program statements.....	56
18. DOS/VS FORTRAN jobs.....	58
19. DOS/VSE ASSEMBLER jobs.....	60
20. DOS/VSE job stream.....	62
21. System logical devices.....	63
22. OCL and JCL comparison (part 1 of 2).....	66
22. OCL and JCL comparison (part 2 of 2).....	67
23. Disk label information.....	68
24. Tape label information.....	69
25. VSE/VSAM label information (part 1 of 2).....	70
25. VSE/VSAM label information (part 2 of 2).....	71
26. Comparison of utility functions.....	75
27. Disk initialization.....	77
28. Tape initialization.....	78
29. Hard copy file print.....	79
30. Volume table of contents (VTOC) display.....	79
31. Display information about a single file.....	80
32. Assign alternate block.....	81
33. Rebuild data.....	81
34. Dump and restore a disk volume to tape.....	82
35. File deletion.....	83
36. Delete contents of VTOC.....	84
37. Copy disk to disk.....	84
38. Backup and restore system libraries.....	85
39. System/3 and DOS/VSE libraries.....	87
40. Sample locations of private libraries.....	88
41. Relative location of the four system libraries.....	88
42. Library functions.....	90
43. Catalog to a library.....	91
44. Print a library entry.....	92
45. Print all directories.....	92
46. Print the system directory.....	93
47. Print and punch a library entry.....	93
48. Delete a library entry.....	94
49. Modify a source library entry.....	94
50. Rename a library entry.....	95
51. System/3 and DOS/VS sort job.....	99
52. VSE/POWER data flow.....	102
53. Sample spooling control statements.....	103
54. Spooling control statements and parameters.....	104
55. DOS/VSE and VSE/POWER jobs.....	105
56. Operator commands.....	106
57. Format of display screen.....	110
58. Format of message line.....	110
59. Console message example.....	111
60. Operator commands.....	114
61. CCP system overview.....	123
62. CICS/VS system overview.....	124

63.	CCP simple inquiry program (SRT).....	125
64.	CICS/VS simple inquiry program (one task).....	126
65.	CCP simple inquiry program (MRT).....	128
66.	CICS/VS simple inquiry program (two tasks).....	129
67.	CICS/VS storage management of two tasks.....	130
68.	Mapset definition.....	142
69.	Map definition.....	143
70.	Field definition.....	144
71.	DFE and BMS specifications.....	155
72.	Inquiry display.....	156
73.	DFE specifications coding.....	157
74.	BMS specifications coding.....	158

This manual should be read by current users of the System/3 series of computers who plan to move to a 4300 Processor. It contains information to help you plan the transition, as well as specific tools for implementing that plan. The manual is divided into two parts. Part I presents conversion information for a batch System/3. Part II presents conversion information for a System/3 running online applications. Each part has the same general structure. The first chapters present an overview of the 4300 Processor with its associated software, described in terms of the functions you currently perform on your System/3. Later chapters describe the detailed steps of the conversion, either batch or online, and identify the tools that will assist you in your conversion.

The individual who is controlling the conversion project should read the whole manual. While it is recommended that others involved in the conversion do so also, they may concentrate on the chapters that discuss their specific tasks.

This manual discusses factors that must be considered before the 4300 Processor is delivered. It is intended to give guidance during the transition planning; it does not attempt to solve particular problems in individual situations.

The usefulness of this guide will depend to a large extent upon the accuracy of the specific conversion procedures suggested and the examples given. Here, a consistent effort was made to maintain the strictest accuracy. The discussions cover the comparable functions of a System/3 and a DOS/VSE 4300 Processor, however not every hardware and software configuration could be taken into account without going into encyclopedic detail. It should be understood therefore that, for some configurations, these generalizations may not be valid.

Frequent reference will be made to the DOS/VSE Entry User's Guide (GC33-6047), which provides specific information on planning, installing, and using DOS/VSE. The DOS/VSE Entry User's Guide also contains a description of the VSE/POWER, VSE/VSAM and VSE/ICCF program products.

CHAPTER 1. OVERVIEW

The IBM 4300 Processors are growth systems for the System/3 user. For purposes of discussion in this manual we will assume you are converting to a 4331 Processor model I1 (a sample configuration is shown in appendix A). The examples in the manual and the discussions of various access methods pertain to fixed block architecture DASD devices, thus native mode operation with DOS/VSE. The 4300 Processors are fully compatible, thus providing upward growth with minimal conversion effort. The 4300 Processors are different from the System/3 but supports some of the same devices, also functions that the 4300 operating system provides are similar to the System/3.

High level languages also have a high degree of compatibility between the two systems. You may use RPGII, COBOL, Assembler, FORTRAN or PL/I languages for 4300 batch applications. Support for RPGII, COBOL, assembler and PL/I is also provided in the primary data base and terminal management programs used by the 4300 Processors.

I/O DEVICES

Several devices that attach to the System/3 can also be attached to a 4331 Processor, they are:

- 5424 Multi Function Card Unit, Models A1 and A2.
- 1403 Printer, Models 2, 7 and N1.
- 3410/11 Magnetic Tape Sub System, models 1, 2 and 3.
- 3340 Direct Access Storage Models A2 with attached 3340 model B1 and B2's.

The attachment of 3340's is recommended for migration time activity, however full systems performance is achieved by using a fixed block architecture DASD device. The VSE/IBM System/3 - 3340 Data Import program (5746-AM3) is a valuable program for converting System/3 3340 files to DOS/VSE SAM and VSE/VSAM files. 3340 DASD can be attached to the 4300 processor by use of the 3340 data import feature. Using VSE/IBM System/3 - 3340 Data import program you can then read the files and write them to a 4300 DASD device. (described in the DASD conversion section)

SOFTWARE

Disk Operating System Virtual Storage Extended (DOS/VSE), program no. 5745-020, is the system control program normally used with the 4300 Processor models I1 and J1. The licensed program VSE/Advanced Functions, program (5746-XE8), is required for the installation of the program products listed below. The use of Advanced Functions provides the necessary interface to the program products along with enhancements to the basic functions of the system control program (DOS/VSE). The description of DOS/VSE in this manual assumes the installation of VSE/Advanced Functions.

The 4300 Processor provides for the use of a wide range of system and application programs. The following is a partial list of the programs available for use with the 4300 Processor. These programs will be referenced in the manual.

Licensed Programs

VSE/Advanced Functions.....	5746-XE8
VSE/POWER with optional Remote Job Entry Feature.....	5746-XE3
VSE/Virtual Storage Access Method.....	5746-AM2
VSE/Interactive Computing and Control Facility.....	5746-TS1
VSE/IBM SYSTEM/3 - 3340 Data Import.....	5746-AM3
VSE/Data Interfile Transfer, Testing and Operations Utility...	5746-UT3
DOS/VS RPG II release 3.....	5746-RG1
System/3 DOS/VS RPG II Conversion Preprocessor.....	5735-CV1
DOS/VS COBOL and Library.....	5746-CB1
DOS FORTRAN IV Library Option I.....	5746-LM3
DOS/VS Sort/Merge Version 2.....	5746-SM2
CICS/DOS/VS.....	5746-XX3
BTAM-ES.....	5746-RC5

Not Licensed

DOS FORTRAN IV Compiler.....	360N-FO-479
DOS FORTRAN IV Library.....	360N-LM-480
Assembler	

CONVERSION

You may be planning a transition from the System/3 to a 4300 Processor for a number of reasons, such as:

- Program compatibility with the SYSTEM/370
- Greater growth capability of the 4300 Processor line
- Application growth
- Corporate standardization

As your company grows, you make greater use of your computer system for the steadily increasing volume of data. More information must be processed on customers, stock parts, general ledger accounts, etc. This may mean that you are outgrowing your System/3 in several areas, such as memory size, the number of partitions available for processing programs, disk capacity, print speed, online capability, etc. The 4300 Processors, with its wide range of processor models and I/O devices, may provide the answer to your data processing needs.

If you are running more and increasingly complex applications, you may have a need for improving the integration of all of your applications. The 4300 Processor has a wide range of software to assist you. Data base management systems such as DL/I can help you organize and integrate your company's data; data communications systems such as CICS/DOS/VS can help you provide timely information to your company's executives; query programs such as GIS/VS can facilitate processing one-time requests for information.

You may be planning a transition from your System/3 to a 4300 Processor because of corporate standardization. If you currently have a mixed System/3 and SYSTEM/370 environment, your data processing department may be more productive and less costly to operate with only one type of computer system. You can thus realize benefits from central program development and maintenance, as well as full data compatibility.

Whatever the reasons you are planning to upgrade, the 4300 Processor may provide additional capabilities that benefit you, depending upon which model and configuration of System/3 you have.

- Growth potential within the 4300 Processor family
- Higher DASD capacity
- More partitions
- Data base offering
- More flexible data communications
- Universal program library

This manual provides the information you will need to plan and implement conversion. You may currently have a System/3 running batch applications, or a System/3 with data communications. Each of these areas is addressed separately to clarify the compatibilities and incompatibilities between the System/3 and the 4300 Processor.

This chapter presents the major differences between the 4300 Processor using DOS/VSE with Advanced Functions and the System/3 using Disk System Management (DSM). It is meant to provide an overview of DOS/VSE and give technical direction that will help your data processing personnel to understand the conversion activity. It does not include the detailed information needed to convert from DSM to DOS/VSE. This is supplied in later chapters.

DISK OPERATING SYSTEM/VIRTUAL STORAGE EXTENDED (DOS/VSE)

DOS/VSE is a group of programs designed to make full use of the resources of a data processing system using the 4300 Processor hardware. The basic functions of DOS/VSE, how they work, are comparable to the System/3 Disk System Management (DSM). Sometimes the names of the programs are the same. The major differences are that DOS/VSE has more functions and that there is a different method used in communicating with the 4300 Processor: DSM uses operation control language (OCL); DOS/VSE uses job control language (JCL). OCL and JCL perform basically the same functions.

Figure 1 shows the operational functions and components of DSM, their functional equivalents in DOS/VSE, and a summary of the additional functions and components available with DOS/VSE. The System/3 DSM functions presented are based on the facilities available with the Model 15. (The Model 10 does not provide some of these functions.)

You can see from Figure 1 that all System/3 DSM functions are available in DOS/VSE. This does not imply that all components function in exactly the same way in DOS/VSE and DSM. In general, DOS/VSE system control programs have more capabilities than DSM system control programs, and they may be used in different ways. This is also true of IBM-supplied program products. You should not expect to find the same number of programs or identical program names in the DOS/VSE program library.

DOS/VSE SYSTEM FUNCTIONS

The following is a brief discussion of the specific DOS/VSE functions that are of interest when moving from a System/3 disk system to a 4300 Processor with DOS/VSE, and their relationship to the corresponding DSM functions.

CONTROL PROGRAMS

Control programs are designed to schedule and supervise the performance of data processing work by a computing system. DOS/VSE provides several control programs.

Supervisor

As on the System/3, DOS/VSE problem programs are executed under control of a supervisor, the major control program of the system. The basic functions and operational characteristics of the two supervisors are the same, but the DOS/VSE offers additional functions, especially in the area of resource management.

Some of the enhancements that DOS/VSE with Advanced Functions provides are:

- Virtual storage
- Multitasking
- Variable partition priority
- Relocating loader
- Job to job communications area
- Automated Systems Initialization
- Asynchronous Operator Communication

FUNCTION or COMPONENT	SYSTEM/3	4300
System Control and Service Programs	IPL	IPL
	Supervisor	Supervisor
	Operator Control Language (OCL)	Job Control Language (JCL)
	Data Management	Data Management
	Spooling	VSE/POWER
	Librarian	Librarian
	System Utilities	System Utilities
	Overlay Linkage Editor	Linkage Editor
	Sort	DOS/VS Sort/Merge
	File to File Utilities	VSE/VSAM Access Method Services
	DITTO	VSE/DITTO
		RAS (Reliability, Availability, Serviceability)
		MSHP (Maintain System History Program)
Language Compilers	Assembler	Assembler
	RPG II	RPG II
	FORTTRAN IV	FORTTRAN IV
	COBOL	COBOL
		PL/I Optimizing Compiler

Figure 1. Operational functions and components of DSM and DOS/VSE

Because of these enhancements and other functions, your supervisor will be considerably larger than the supervisor on your System/3. Some of these enhancements are explained below.

Virtual Storage

DOS/VSE uses virtual storage to extend the limits of physical main storage or real storage in a computer system. Virtual storage is addressable space that appears to the programmer as real storage. The addresses that DOS/VSE assigns are not limited by the size of 4300 Processor main storage. They are consecutive addresses that refer to a logically continuous address space, up to 16 million bytes. You define the size of this address space, and can write, compile, link-edit, load and execute your programs as if you had a processor with a main storage capacity as large as the address space. Although you can define a 16 million byte address space, in practice you should define only enough space to allow for the largest program you will execute.

The programmer needs not be aware of the mechanisms that allow substitution of direct access storage for main storage. The substitution is accomplished by a combination of 4300 hardware and DOS/VSE System Control Programming.

The increase in storage capacity provided by virtual storage can be compared to that provided by the System/3 RPG II automatic overlay feature. In both cases, you can write programs without taking into consideration the amount of actual main storage available.

On the System/3, the RPG II compiler segments the compiled program in order to make it fit into actual main storage. This segmentation is a very generalized technique, and does not allow you to divide a program into often used and seldom used segments. In other words, you cannot establish "priority" of program segments. Also, a segmented program uses more storage than a non-segmented one. When you want to run a segmented program on System/3 configurations with different main storage sizes, you must normally have multiple copies of the program in your program library.

In DOS/VSE, the supervisor and not the compiler "fits" the program into the available main storage. DOS/VSE divides the program into 2K-byte sections, called "pages". The DOS/VSE supervisor has a built-in algorithm that keeps the often used pages of a program in real storage and the seldom used pages on direct access storage called a "page data set". The pages of the program on direct access storage are brought into main storage or "paged in" as they are needed. In all cases, for an instruction to be executed, the instruction and its operands or data must all be in real storage.

In this way, performance may not be seriously affected when a program is segmented. The DOS/VSE supervisor automatically adapts the program to the available real storage size. You can reduce the time spent on programming because increased storage capacity reduces the need for creating program overlays for COBOL or FORTRAN programs. Furthermore, high-level languages can be used in situations where physical main storage constraints previously forced you to write subroutines or an entire program in Assembler language.

Virtual storage is especially useful on a system running data communications programs. When you run data communications applications in a system without virtual storage, as on the System/3, in order to have acceptable response time during maximum traffic periods, the main storage available to the partition must be large enough to hold the entire data communications program. This implies that, during periods of low traffic, part of the data communications partition contains dormant routines. As a result, your other applications suffer storage constraints during these low traffic periods. With the dynamic real storage allocation ability of DOS/VSE, your data communications program can get the real storage it needs during periods of high activity, but

during low-activity periods the batch program can use that real storage for its application processing. Figure 2 shows how using virtual mode improves storage use for data communications application.

Multiprogramming and Multitasking

All programs running under the control of DOS/VSE are executed from storage areas called "partitions". VSE/Advanced Functions allows up to seven partitions to be defined for use by the system. The number of partitions you generate to handle your workload may affect the total throughput of your system, depending upon the system resources available, such as processor memory size, and number of I/O devices, such as disk. The greater the resources available, the more partitions you will be able to efficiently support. DOS/VSE requires that you define a minimum of two partitions. If the resources are available it is advantageous to define the maximum of seven partitions since you can execute concurrently as many programs as there are partitions. When two partitions are defined, the result is a system similar to the System/3 Model 10 with the dual programming feature (DPF) or a Model 15 defined with two partitions; that is, you can execute two programs concurrently. Such concurrent program execution in DOS/VSE is called "multiprogramming". When a program currently being executed finds that it cannot continue because it must wait for an operation (such as input or output) to be completed, the system passes control to another program of lower priority. The system will enter the wait state (become inactive) only if none of the programs in the system is able to continue with useful work. As soon as an I/O or pending operation for a high-priority program is completed, execution of any lower-priority program is interrupted and control is passed to the high-priority program. Priorities are explained under "Variable Partition Priority" below. In addition, different parts of the same program can be executed concurrently where it is meaningful to do so. This is called "multitasking" and can be done within a given partition.

When you generate your DOS/VSE supervisor, you specify the number of partitions you require. When the system is initialized each of the partitions are allocated real and virtual memory. These allocations can be changed by the operator subsequent to the system being initialized. To increase the number of partitions available, you must generate a new supervisor specifying the new number of partitions.

Variable Partition Priority

Using DOS/VSE with multiprogramming, programs are executed according to an established set of priorities. The priority of a program for using the processor depends on the priority of the partition in which it is located. This is similar to the System/3 Model 15 partition priorities. The default priorities of DOS/VSE partitions, are established at system generation.

You can change the default priorities of the partitions at system generation or by using an operator command. Assigning a program to a given partition also assigns the priority of the partition to the program. You should assign the highest priority that is available to the partition to be used for rush jobs, such as an inquiry against a part file, for example. This ensures that rush jobs are executed as soon as possible. As long as the rush job partition is inactive, the other partitions are automatically given the next higher priority.

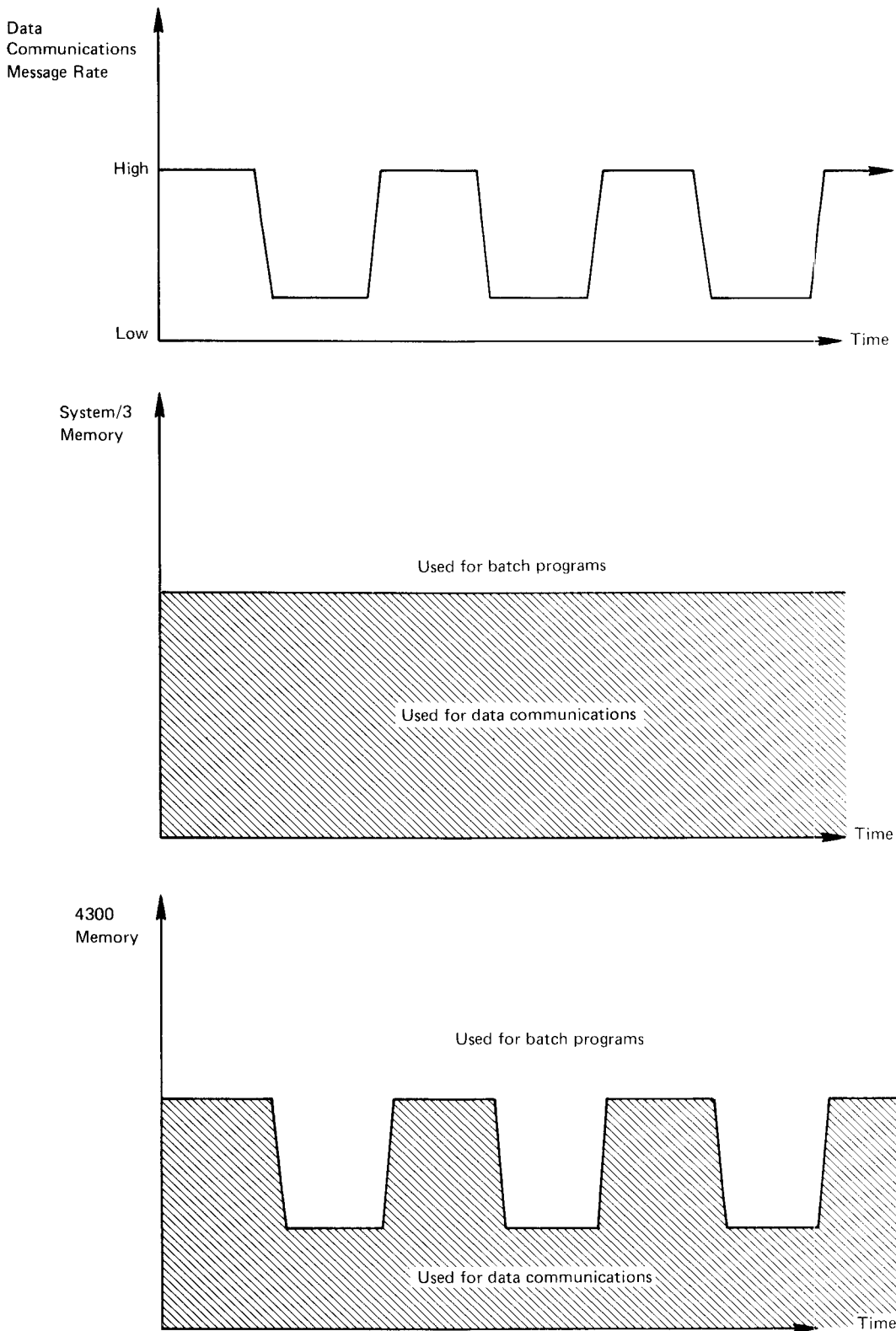


Figure 2. Influence of DOS/VSE automatic storage management in a data communications system

Relocating Loader

The relocating loader is an important feature of DOS/VSE. With this feature, you can dynamically change the virtual storage location from which a program will start execution. This is particularly important when you want to be able to execute a program in any DOS/VSE partition.

You need only one copy of the program in the core image library (the equivalent of the System/3 object library), even though the same program may be executed in more than one partition of different sizes. You need not link-edit a program again in order to execute it in a different partition from the one for which it was originally link-edited. This is also true in converting from one release of DOS/VSE to another when the size of the supervisor changes by adding some new functions.

As a result, the relocating loader saves programmer time, reduces the space required for the core image library by eliminating the need for multiple copies of programs, minimizes the number of linkage-editor runs, and reduces the transition effort from one DOS/VSE release to another.

Job to Job Communications area

The Job to Job communications area is an area located in the system where information can be stored and later accessed by another job. This allows information to be stored by one job and later accessed by another job. A special DOS/VSE macro provides the programmer access to the communications area.

Automated System Initialization

The initial program loader (IPL) initializes the system and loads the supervisor. The function is similar on DOS/VSE and the System/3, except that the procedures an operator must follow are different. DOS/VSE provides for an automated IPL that virtually eliminates operator interaction once the hardware IPL is accomplished. Automated systems initialization provides for starting the system and may also be used to initiate programs automatically. Programs such as "spoolers" and data communication programs that would normally run most of the day can be started automatically at Systems Initialization time.

Asynchronous Operator Communications

Asynchronous Operator Communications allows the operator of the system to defer response to one or more messages in order to issue commands, respond to higher-priority messages, or answer messages that only require a simple response. All messages that require a response are retained on the console screen until the operator responds to them. The console operator may also call the messages to the screen by providing the message identifier.

Job Control

As on the System/3, a program and the configuration on which it is to be executed must be defined using control statements. Where on the System/3 you used OCL statements, with DOS/VSE you use Job Control Language (JCL) statements. The DOS/VSE concept of JCL is much like that of the Model 15 OCL. You must provide the same kind of information on JCL cards as you did on OCL cards (that is, whether the program is to be compiled or executed, from which library or device the program is to be loaded, the files to be processed, and where these files are to be found). This does not imply that all information is provided in exactly the same way on the two systems. Occasionally, the statement identifier and the function performed are the same (for example, DATE and PAUSE), but in most cases JCL statements have different identifiers. For example, the information provided in the System/3 FILE statement is provided in the DOS/VSE // DLBL, // EXTENT, and // ASSIGN statements.

The unit of work submitted to DOS/VSE is called a "job"; a succession of jobs is called a "jobstream". The beginning of each job is indicated by a // JOB statement, and the end by a /& statement.

You may define one or more "steps" in a job. Defining a job as one jobstep produces a unit of work equivalent to a unit defined by one set of System/3 // LOAD and // RUN statements. Defining a job as a series of jobsteps gives a unit of work equivalent to a unit defined by a series of sets of System/3 // LOAD and // RUN statements, one set per jobstep.

Defining a series of related steps as a single job may sometimes be required or it may otherwise provide specific advantages. For instance, multiple steps within a single job would be preferred when execution of a later step is directly dependent upon successful completion of an earlier step. If a step terminates abnormally, the remaining steps in the job are bypassed, and the next job is begun.

As on the System/3, you can create "procedures" for frequently used sets of JCL statements, although in DOS/VSE these procedures cannot be nested.

DATA MANAGEMENT

Data management routines provide an interface between the user programs and the required data files, just as on your System/3. They handle the transfer of data between I/O devices and main storage. Data management consists of a "physical" and a "logical" portion. Physical data management is a group of routines that perform the actual transfer of data between main storage and an I/O device. Logical data management is a set of routines that perform functions such as:

- Opening and closing files
- Blocking and unblocking records
- Handling control operations such as space and skip
- Processing end of file

Logical data management, based on control information you provide, checks that the proper data files and I/O devices are being used.

As on the System/3, these routines are transparent to user programs.

Data Access Methods

The following access methods are available for use with the 4300 Processor equipped with fixed block architecture DASD devices. The Sequential Access Method (SAM) is provided with DOS/VSE. The Virtual Storage Access Method (VSE/VSAM) is a licensed program product. A prerequisite to the installation of VSE/VSAM is the licensed program product VSE/Advanced Functions. The functions of the access methods are:

Sequential Access Method (SAM): Records of a sequential file are arranged in the order in which they are created. Sequential files are accepted by all I/O devices except teleprocessing terminals. DOS/VSE Sequential Access files are similar to System/3 sequential files except that DOS/VSE sequential files may not be extended.

Virtual Storage Extended/Virtual Storage Access Method (VSE/VSAM): VSE/VSAM is an access method for direct or sequential processing of disk files.

The records in a VSE/VSAM file can be organized either in logical sequence according to the value of a key field (key-sequenced) or in the order in which the records are created (entry-sequenced) or based on a relative record number. You can read, add, delete, and modify records in a VSE/VSAM file sequentially or directly according to key or address.

A key-sequenced file has an index; its records can be accessed according to key or address. An entry-sequenced file does not have an index and records can be accessed only according to address; it can be processed like a sequential or direct file and it may be extended. VSE/VSAM Access Method Services provides a program to create and service VSE/VSAM files.

VSE/VSAM files are the most similar to System/3 files, and VSE/VSAM is the recommended file access method for your disk files.

Although the data management facilities are similar to those on your System/3, it will be necessary to reload your disk files because of the different physical disk formats, and to change access methods in some cases to get equivalent functions. Refer to the chapter on data file conversion for details.

DATA BASE MANAGEMENT

The conversion of applications to a data base concept should be considered prior to converting to the 4300 Processor or as an enhancement to the system after the 4300 Processor has been installed. Normally in a system with diverse applications there are multiple files containing data that is duplicated for each application. A data base is a concept for integrating, sharing and control of common data. A data base provides the flexibility of data organization by removing the direct association between the program and the physical storage of data. This allows changes to the data base without impacting all the programs that access the data. Some of advantages of a data base are:

- Control of data redundancy and reduction of duplicate data maintenance.
- Consistency through the use of the same data by all groups in the company.
- Application program independence from physical storage organizations and access methods.
- Reduction in overall application costs.
- Data designs usable for batch and online processing.
- A system-provided focal point for the control of data.

DOS/VS Data Language/I (DL/I), program number 5746-XX1, is a data management control system available for the 4300. DL/I provides an interface to application programs written in RPGII, COBOL, Assembler and PL/I. DL/I uses VSE/VSAM as its access method and provides utilities to define, create and reorganize the DL/I data base.

PROCESSING PROGRAMS

Processing programs run under control of the system control programs. They consist of service programs, language translators, and application programs.

Service programs

Service programs assist in the use of a computing system such as spoolers, utilities, or library programs.

Spooling

"Spooling" is the name of a technique used to increase the total throughput of a computer system. There is a larger discrepancy between the speed of the processor and unit record devices such as card reader,

printers and punches. A spooler program stores unit record input and output data on disk thus allowing the unit record devices to operate at near their maximum speed. This technique also allows the user to run programs in more than one partition, yet have only one printer, one reader and one punch. The System/3 and the 4300 Processor both have spooling programs. The DOS/VSE spooling program is the licensed program product VSE/POWER (5746-XE3). VSE/POWER has a high degree of flexibility and numerous functions that should enhance your computer operation. For more information see "spooling", Chapt 11.

Library Service Programs

As in the System/3, DOS/VSE programs and routines are stored in libraries. The number of libraries in DOS/VSE and the library names are different but the functions the libraries perform similar.

Programs and routines in the System/3 object library would be stored in the DOS/VSE core image and relocatable libraries, respectively. Source programs and procedures in the System/3 source library would be stored in the DOS/VSE source statement and procedure libraries, respectively. Figure 3 shows a comparison of the System/3 and DOS/VSE libraries. DOS/VSE provides a full range of library programs to service these libraries, just as on your System/3. The control cards used on DOS/VSE will be different. See the chapter on library considerations for further detail.

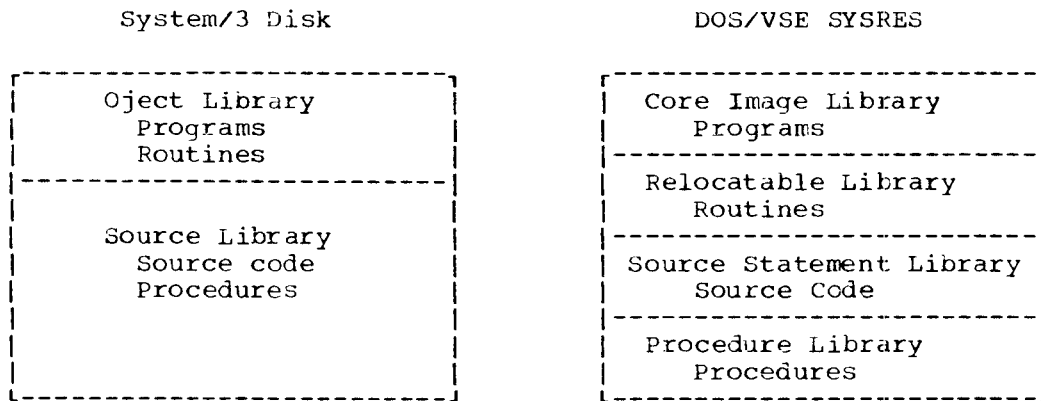


Figure 3. Comparison of the System/3 and DOS/VSE Libraries

Utilities

System utilities allow you to prepare and maintain your disks, prepare data files for processing, prepare I/O units to receive data, copy data from one medium to another, etc. The following is a partial list of the DOS/VSE system utilities. In most cases they are similar to System/3 utilities, but the control cards needed to request services will be different.

- Initialize magnetic tape
- Initialize disk
- Clear disk
- Display volume table of contents (VTOC)
- Copy and restore diskette
- Backup and restore system
- Assign alternate block or track
- Print hard-copy file
- Copy file or volume from one disk to another
- Blocking, deblocking and copying a file
- Updating object programs
- Dump/Restore file

Additional utility and file services are available with VSE/VSAM Access Method Services, (provided with VSE/VSAM, program no 5746-AM2), which provides file utility functions for VSE/VSAM files. The licensed program product VSE/DITTO (5746-UT3) is also available for utility and file services.

Details on the use of the System Utilities, Access Method Services, and VSE/DITTO are given in the chapter on utility conversion.

Linkage Editor

DOS/VSE provides a linkage editor with functions similar to the System/3 overlay linkage editor. It creates loadable programs from multiple relocatable object modules, and stores the output in the core image library. Each program you write must be link-edited before it can be executed. With the virtual storage and relocating loader of DOS/VSE, you need not bother with creating overlay programs. The control statements to use the linkage editor in DOS/VSE are different from those on the System/3. See the chapter on program conversion for how to use the linkage editor.

Error Tracking

Error statistics are continuously maintained by the system. Special system programs are available to analyze these statistics. Routines are provided in the DOS/VSE system control program code to detect, analyze, and circumvent machine malfunctions. Messages informing the operator of these conditions are automatically provided.

Storage printouts, called dumps, which include the status and contents of registers, can be automatically provided in the event of program errors such as invalid storage addresses, invalid arithmetic operations, and the like. The dump options are normally specified at system initialization or through the use of job control statements. The storage printouts are under control of DOS/VSE, which continues to the next job without operator intervention. Formatted dumps to facilitate error analysis can be produced by the DOS/VS RPG II, COBOL/VS, and PL/I language compilers.

Language Compilers and Assembler

Except for the assembler, DOS/VSE language compilers are generally compatible in both function and operating procedures with the corresponding DSM language compilers. Thus, if you used RPG II, COBOL, or FORTRAN as the primary programming language on your System/3 disk system you will be able to write new application programs using DOS/VS RPG II, COBOL/VS, or FORTRAN with little additional training. You will, however, have to learn how to resolve the differences between DOS/VS and System/3 RPG II, COBOL, FORTRAN, and Assembler.

RP II Compiler

RPG II is a programming language that can be used for a number of commercial data processing applications. DOS/VS RPG II provides the same kinds of functions as System/3 RPG II so you will not have to learn a new programming language for DOS/VSE. There is a high degree of compatibility between the two languages, but because of some differences, especially in disk file organization and printer control, you will need to make some changes to your source programs. The System/3 DOS/VS RPG II

Conversion Preprocessor program is available to help you automate the conversion process. See the section on RPG II programs in the program conversion chapter for more information.

DOS/VS COBOL compiler

DOS/VS COBOL is an English-like programming language especially designed for commercial data processing applications. The DOS/VS COBOL compiler translates ANS COBOL source program statements into executable object

code. Although the System/3 COBOL is a subset of DOS/VS COBOL, some changes must be made in your source decks, especially for device types and file organization. See the section on COBOL programs in the chapter on program conversion for more information.

FORTRAN Compiler

FORTRAN is a language primarily designed to solve scientific and mathematical problems. The DOS FORTRAN IV compiler and the FORTRAN IV Library Option 1 are available with DOS/VSE to translate FORTRAN source programs into object programs. There are some differences between System/3 FORTRAN and DOS FORTRAN, which are discussed in the section on FORTRAN programs in the program conversion chapter. Note that the commercial subroutine package is not available with the DOS FORTRAN IV compiler.

Assembler

The DOS/VSE Assembler prepares programs written in 4300 Assembler language for execution. As on the System/3, the assembler language lets you code machine and operating system functions. Because of the hardware differences between the System/3 and the 4300 Processor and the differences in the two assemblers, your assembler language programs and subroutines will have to be rewritten. See the section on Assembler programs in the program conversion chapter for details.

PL/I Compiler

PL/I is an additional language available with DOS/VSE designed for commercial and scientific applications. Since you do not have this language on your System/3, it is not discussed in this manual.

Sort/Merge Program

DOS/VS Sort/Merge enables you to sort several disk and tape files with randomly ordered records or to merge sequenced records into a sequentially ordered file. In addition the sort/merge provides:

- Data security by means of an option that erases work files
- Messages routed to either the console or the printer
- A storage dump, on request, after a critical message
- Record selection (include and omit)
- Record formatting
- Summary sort
- Alternate collating sequence

Because of the differences in control card formats, System/3 sort control statements must be converted to DOS/VS sort/merge control statements. The DOS/VS sort/merge does not provide for some of the functions performed by the System/3 sort such as multiple record type or the "force" function. Refer to the chapter on sort conversion for details.

ADDITIONAL DOS/VSE FACILITIES

Rush Jobs

At times you will need the facility, during normal operations, to execute a program immediately, using the results to make a business decision or continue program development. DOS/VSE does not require a "rollout/rollin" capability where you must interrupt a currently executing program, roll it out to disk, bring in an inquiry program, and roll in the original program when the inquiry program is finished. In DOS/VSE, by generating an extra partition in your system, and giving it the highest priority, you can load rush jobs or inquiry jobs without having to interrupt another job. DOS/VSE will let the high-priority rush job use the main storage resources it needs to do the work, and when the

job finishes, return the storage to other executing programs. In this way, DOS/VSE eliminates the need to interrupt a program in progress, and decreases operator intervention.

Checkpoint/Restart

DOS/VSE provides a checkpoint/restart capability, much like the System/3, for 4300 COBOL and Assembler programs.

Checkpoint records are written during the execution of a program. If processing is terminated due to a machine failure or an operator-initiated cancel, the job can be restarted at the last checkpoint, instead of rerunning the whole job. This is a very useful feature for long running jobs. The control cards for using this facility will be different in DOS/VSE.

System Generation

The system generation process enables you to tailor the DOS/VSE system to your specific needs. This process is similar to that on your System/3. By coding special control statements, you select which of the features of DOS/VSE you would like included in your supervisor(s). The I/O devices that are attached to your system are defined to DOS/VSE during system initialization. This allows you to redefine the I/O that are assigned to your system without having to generate a new supervisor. See the chapter on system generation for more detail.

Data Communications

DOS/VSE provides facilities to run data communications programs that process data from local or remote terminals. Application programs can be written in Assembler language using macros to request communications services similar to MLMP or MLTA, or in a high-level language running under control of CICS/VS, a control program similar to CCP. Although the functions are similar, you will have to recode, and in some cases, redesign your data communications programs. Details are covered in Part II, Data Communications.

Assigning I/O Devices

In DOS/VSE, processing programs that need access to a data file must inform the system of the type of device involved. This applies to DOS/VSE programs as well as your own programs. The actual device is not specified in your program, only a symbolic name of the form SYSxxx referring to a logical unit must be specified.

The user assigns a logical device name to a physical device during system initialization or during system operation. Figure 4 shows the DOS/VSE logical units used for system input and output compared with the SYSTEM/3 function.

Job Duration Printout

As previously discussed under Job Control, a logical unit of DOS/VSE work is called a job. The 4300 Processor has a standard hardware feature, the time-of-day clock, which DOS/VSE uses to produce an automatic printout of job duration on both the console display and the system printer.

Job Accounting

DOS/VSE has an optional facility that accounts for the processor and I/O resources used by each jobstep. This can assist you in charging user departments for computer services.

DOS/VSE Logical Unit	Use	System/3 Assignment
SYSRDR	Reading job control statements	// READER
SYSIPT	Input of system data, such as source statements for language translators, or control information for service programs	// READER
SYSIN	When SYSRDR and SYSIPT are assigned to the same device, i.e., both job control and data are read	// READER
SYSPCH	Punched output of the system	// PUNCH
SYSLST	Printed output of the system	// PRINTER
SYSLOG	Communication between the system and the operator and for logging job control statements.	// LOG

Figure 4. System input/output files.

CHAPTER 3. TRANSITION PLANNING

A well conceived and detailed plan for the transition from the System/3 to the 4300 Processor is essential to the success of the changeover. While no one plan will fit all needs, there are some general considerations common to most users. These alternatives and their interaction should be studied before a plan is drawn up.

A transition plan should establish the details of the conversion effort and the directions necessary to use the new system. In developing a conversion plan:

- Establish long- and short-range objectives
- Analyze the interaction between programs and files
- Become familiar with software and hardware differences
- Prepare for the changes in operating procedures and administrative activities
- Review the organization of the data processing department
- Assign responsibilities to implement new system functions
- Schedule activities and manpower requirements
- Decide what applications are needed and when, and which should be redesigned and in what priority.

OBJECTIVES

Moving to the 4300 Processor will in most cases give you a more flexible and powerful computer system. Therefore, before beginning the transition, study the new facilities offered by the system and then review your present objectives. You will probably select certain 4300 Processor facilities to be used immediately and others to be installed on a long-range basis.

The short- and long-range objectives you establish for the transition are linked. Your ultimate goals must be made clear early. You must choose which hardware devices to add to your system and when, and which operating facilities to adapt and in what order.

You can begin with gradual implementation of your DOS/VSE facilities and options, such as multiprogramming and spooling, and introduce new hardware functions such as larger disk capacity and increased storage size.

Gradually, you can write new applications and rewrite existing ones.

You should ensure that the new applications integrate smoothly into operations with improvements to the efficiency of existing applications. To do so:

- Use larger real storage, which will allow you to combine several smaller programs into one large one
- Increase the disk capacity of the system to keep more of your data accessible
- Integrate files containing redundant information and organize data into a data base to be used by more applications

- Catalog procedures so as to reduce the number of control cards required to execute a job
- Expand multiprogramming to include data communications permitting simultaneous processing of batch and communications applications
- Create accounting routines to monitor system use.

It is essential that you be aware that, although transition to the 4300 Processor should be swift, growth within it may be gradual. Radical transitions are generally costly. Functions that are beyond your current requirements should be included in your long-range plans.

CONSIDERATIONS

In the text that follows, we will discuss a wide range of conversion activities and suggest general guidelines rather than specific solutions. Additional details on conversion are given in later chapters.

PROGRAM CONVERSION

For a given application, you may have to do both file conversion and program conversion. Within each type, you may use a variety of techniques. The principal techniques of converting System/3 programs to DOS/VSE programs on a 4300 Processor are:

- Use of an automated conversion program
- Manual translation
- Reprogramming and redesign.

Each technique has immediate and long-term advantages and disadvantages. In all likelihood, you will use more than one technique to change to the new system. Some programs may be automatically translated, while others may require reprogramming for efficient operation.

Automated Conversion

Use of an automated conversion program can greatly reduce the amount of time spent in preparing a System/3 program to run under DOS/VSE on a 4300 Processor. Many of the language differences can be automatically translated by System/3 DOS/VS RPG II Conversion Preprocessor for your RPG II programs. Although some items may still require manual translation, your programmer will have more time to help with other conversion tasks.

If you have many programs written in another language such as COBOL or FORTRAN, it might be worthwhile to write your own conversion program to translate your application programs. You must decide whether it would take more time to write the program than to make the changes manually.

Manual Translation

Manual translation consists of transforming, by hand, a program written in a language used on the System/3 into a program in the same language but acceptable to DOS/VSE on a 4300 Processor. Since translation is rewriting, programs that have been translated must be recompiled. The advantage of translation is that it probably will require less time than entirely rewriting the program.

The effectiveness of translation and the programming effort involved depend on the state of the source code. After translation and recompilation are completed, you must examine your programs one by one and test the results. If a program requires many manual changes before compilation or many corrections after, complete reprogramming may be a more efficient alternative.

If you have chosen language translation as a conversion technique:

- Prepare a guide of language differences and use it to modify source code
- Compile programs to find special differences and possible errors, and recompile if necessary
- Rewrite control cards for DOS/VSE
- Convert data files for DOS/VSE
- Test programs with converted data and compare results.

You may find some difficulties in translating Assembler programs and routines. Differences between hardware instructions for the System/3 and 4300 Processor require a complete recoding, although the logic of the flow need not be changed.

Reprogramming and Redesign

Reprogramming enables you to combine the experience gained in creating your current applications with the new functions available in DOS/VSE. However, this technique by itself is often long and difficult. If you combine reprogramming with other conversion techniques, your programs can be rewritten gradually and your current applications can be run without competing for time with other conversion activities.

By definition, reprogramming means a statement-by-statement rewriting of each program. The most important step in any conversion plan is analysis and redesign, which involves rethinking your programming problems in terms of the hardware and software now available to you. In some cases it is preferable to develop entirely new applications. You must decide how best to integrate what is new and what has been redesigned, and whether to modify your configuration to fit the requirements of a new design. For instance, you might consolidate several applications and keep more data online for data communication applications.

FILE CONVERSION

The type of file conversion required depends on several factors:

- Whether your files are principally on cards, tape, disk, or diskette
- What configuration and which utility programs are available for the conversion.

Some general information on file conversion is contained in the following sections. Detailed techniques are discussed in the chapter on data file conversion.

Converting Card Files

If your 4300 configuration will have the same type of card I/O devices as your System/3, you should not have to convert your card data files. However, if you change card devices, you may have some conversion considerations.

If you change from 96-column cards to 80-column cards, you may have to modify the card layout of those files that have data extending beyond column 80, and then convert your permanent card file into the new format. You can write a simple user program and run it on a System/3, System/370 or 4300 with both 80- and 96-column card capability, or use tape as an intermediate storage medium. In addition, you will have to modify your programs to allow processing of 80-column cards.

Another consideration is the function that different card devices will

allow you to perform. For example, with the 5424 MFCU on your System/3 you can do functions such as merge, select, read, punch, and print on the same card. These functions may not be available on your 4300 card I/O devices. In this case, you may need to add a jobstep or consider using tape or disk for your files.

If you decide to keep the 96-column format either in cards or diskette, be aware that the system logical unit SYSIN supports only 80-byte records. This is not a problem if you use programmer logical units, SYSnnn, to read your card files.

Converting Tape Files

Your existing tape files created on the System/3 need not be modified to be used with DOS/VSE. However, it is important to note that, if your tapes have different densities or track formats from those used by your 4300 tape drives, you will have to use a utility program to convert densities and formats.

Converting Disk Files

Perhaps the most important item to consider in converting your data files is the conversion of your data on disk, since most of your major files are probably disk resident. Because of the different physical disk formats between the System/3 and the 4300 Processor, it will be necessary to reload all of your disk files onto the 4300 devices. This can be accomplished:

- By use of the VSE/IBM System/3 - 3340 Data Import, program number 5746-AM3, with the associated data import 4300 hardware feature.
- By dumping your files to tape or diskette on the System/3 and reloading the files on a 4300 Processor, using simple RPG II programs or utilities
- By punching disk data into cards on your System/3 and reloading on a 4300 using simple RPG II programs or utilities.

The file reloading can be accomplished at an IBM data center or on your own system after it is installed.

Another important consideration is the optimization of the disk data files in order to take advantage of the 4300 disk device. You will need to learn how the 4300 disks differ from the System/3 disks.

Converting Diskette Files

The primary consideration in converting your diskette data files is the size of the record. If the diskette files are processed as SYSIN, the record length must be 80 or 81 bytes (bytes 2-81 are used). If your data is processed through SYSnnn, there is no restriction. If you have a cardless system, and are using VSE/POWER, your records are supported as SYSIN only and must be 80 or 81 bytes. If you are reading data from a remote device with VSE/POWER RJE, again your record must be 80 or 81 bytes. This may mean that some of your programs will have to be modified because of the changed record format.

JOB CONTROL STATEMENTS

When moving from a System/3 to a 4300 Processor, you must replace the System/3 OCL statements with DOS/VSE JCL statements. Although the basic functions are the same, the difference in format does not usually permit a statement-for-statement conversion. OCL statements such as CALL, FILE, JOB, LOAD, PAUSE, and SWITCH have direct equivalents in JCL. Other statements require more conversion effort. You should thoroughly understand the concept of organization by job and jobstep in order to

use the additional facilities of the JCL language. After you have converted and tested the JCL, you can place some of these statements in the procedure library as cataloged procedures. The chapter on OCL to JCL conversion gives more detailed information.

UTILITIES AND SORTS

Control cards for your present utilities and sorts will need to be changed to a DOS/VSE format. This means you must understand the functions currently performed on your System/3 and learn how to do the same thing in DOS/VSE. If you have changed program functions or file formats, you must reevaluate the sorts and utilities to see if they are still needed or used in the same way. Each of the sorts and utilities must then be tested. Specific information on sorts can be found in the chapter on sort conversion. Information on utilities can be found in the chapters on utilities and library considerations.

PREPARING THE TRANSITION

Before the actual transition begins, you should do the following: identify personnel to work on the conversion; plan staff education, operator training, and orientation for user departments; establish standards and control procedures; and classify programs. These points are discussed below.

PERSONNEL REQUIREMENTS

One person should be placed in charge of the conversion effort, to allocate people and other resources to the jobs that need to be done. He should have the authority to make all necessary decisions. He can allocate responsibility for specific areas - for example, an application system to other individuals.

Someone must be designated to perform the system programming function, if you currently do not have such a function for your System/3. This function involves the planning and executing of system generation, planning the layout of files on disk volumes, and tuning the system for performance. System programming is an especially important function for a DOS/VSE system, because there are many more system facilities and options than on the System/3.

During conversion, you will still be running your existing applications on your System/3. This may mean that someone from the conversion team may have to debug a program or perform some program maintenance. In general, though, you should cease any new development work on your System/3. Development can be resumed for after the 4300 Processor is installed.

EDUCATION

Once the conversion team has been selected, you should schedule the necessary education. Most instruction should be provided well before you install your 4300 Processor. Some training may have to be scheduled during or just after installation, especially for computer operators who require "hands on" time. Many IBM educational facilities are available for training your programmers and computer center staff. In some cases, you can make use of independent study courses, so that personnel can learn the new subjects as they have time. Appendix B contains a list of suggested courses for various job functions. You should also review the current Customer Education Catalog and Schedule, G320-1244, and schedule the appropriate classes with your IBM marketing representative.

You should establish an education history and education plan for each staff member. This information can be keyed into your program inventory so that you know who is qualified to oversee and correct the running of each application. Perhaps one person can be sent to formal education classes, and then teach the rest of the staff what they need to know for their parts in the conversion.

Your education requirements may increase as you expand the functions of your system. You may find that self-education using IBM-supplied literature can prepare your staff for new system functions. The IBM System/360 and System/370 Bibliography, GA22-6822, and IBM System/370 Advanced Function Bibliography, GC20-1763, are regularly updated reports of current IBM publications.

Operator Training

Operator training must be an important part of your education plan. Because of basic differences in the nature of operator-system communication between DSM and DOS/VSE, your operators will have to learn a new set of procedures.

For DSM, the CPU console (or CRT on the Model 15) is the medium of communication between operator and processing program. The printer is also used by the system to produce messages from the language translators. The 5471 Printer-Keyboard on the Models 8, 10, and 12 is optional and is not normally used to communicate with the operator.

The operator can also initiate communication with the system or problem program by means of commands. He can enter commands from the keyboard and, when communicating with the job control program, from the system device assigned to read JCL.

Your operator should be included in all of the testing sessions you have at an IBM data center. In this way the operator can acquire the necessary "hands on" training and become familiar with the way your own programs will work on the 4300 Processor.

User Departments

Although the conversion of systems should be as transparent as possible to your users, manual procedures may change, or input data and output reports may change formats, perhaps because of new devices. In this case, the user departments must learn the new procedures or data formats. If you currently have terminals in your user departments, the people who use the terminals will have to learn about new terminal commands to request the functions they need to do their jobs.

The data processing department should hold special seminars for each user department, describing any changes. If the users know that a change is being made in the data processing equipment, they can be prepared for any minor scheduling changes caused by file conversions or equipment installation. The user can also help in the final phases of program testing, by providing a normal day's input of data. This can verify that the changed programs and procedures are working correctly.

STANDARDS AND CONTROL

When changing your system, the use of the new facilities and capabilities requires a review of the standards used.

System Standards

System standards apply to every group in a data processing installation, and include:

- Standards to ensure data and program security

- Execution time limits to prevent a looping program from monopolizing the system
- Volume serial numbers for tape and disk volume control
- Space allocation techniques in job control coding
- Operator messages requiring replies written in a form that cannot be confused with the standard system messages
- Job mix conventions
- Naming conventions
- Documentation standards
- Console message formats so that messages are readable
- VSE/POWER job classes and print/punch classes

Programming Standards

Programming standards apply to system and application programmer, and include:

- Specific language conventions
- Labeling conventions
- Access method conventions
- Buffer conventions
- Linkage conventions
- Device independence within modified programs

Operating Standards

Operating standards apply to the personnel responsible for the operation of the computing system and its supporting services, and include:

- IPL procedures
- Operating procedures for each job (run book)
- Error procedures
- Rerun procedures
- Procedures for submitting a job to be executed and returning the job to the submitter
- Job accounting procedures
- System backup procedures
- File backup and recovery

Control

A control procedure will help keep you informed of the progress of the conversion.

Draw a progress record chart that compares scheduling with activities accomplished. Information provided by your programmers each week enables you to compare work scheduled with work accomplished. Systematic checkpoints, used more frequently during the early stages of the transition, enable you to verify your assumptions and to adjust

estimates and dates if required.

Figure 5 is an example of a progress record chart. It details the information contained in the program and file revision schedule and in the programmer's weekly reports.

PROGRESS RECORD														West Publishing			
Program/File		Programmer	Days	Analysis	Start Date	Program File Revision	Start Date	Test	Start Date	Documentation	Start Date	Percent Complete	Total Days	Target Date	Comments		
Description	ID																
Accounts Receivable	AR 100	Jones	Scheduled														
			Completed														
	AR 110	Jones	Scheduled														
			Completed														
	AR 120	Jones	Scheduled														
			Completed														
	AR 130	Jackson	Scheduled														
			Completed														
AR 140	Jackson	Scheduled															
		Completed															
Inventory	IV 300	Allen	Scheduled														
			Completed														
	IV 310	Allen	Scheduled														
			Completed														

Figure 5. Example of a progress record sheet.

CLASSIFYING PROGRAMS AND FILES

By preparing inventories of existing programs and files, you can estimate the work to be performed by the new system and the effort needed to change the programs and data. A program inventory lists programs that have little or no activity and those used extensively. A file inventory describes each file used by your installation and shows the dependencies among groups of files, whether they have to reside on the same volumes, and the sequence of their conversion.

Figures 6 and 7 show examples of forms that can be used for inventories. Figure 6, a file inventory form, indicates the type of 4300 device and file. To improve performance, for example, you may decide to have a System/3 tape work file on a 4300 disk device. A program inventory form (Figure 7) may help you to decide how your programs interrelate. By determining the programming language used, the number of source statements, and the programmer responsible for the development and maintenance of each program, you identify key factors in the conversion workload. Try to identify any programs that use instructions and facilities in a nonstandard way.

Application	File Name or Number	Device Type and Model Number	Number of Tapes	File Organization						File Use	Comments	DOS/VSE File Organization (Proposed)	
				Record Format	Density	Parity	Number of Records	Logical Record Size	Physical Record Size				Labels
										Note 1	Note 2	Note 3	

Notes:

1. Possible entries are standard, non standard, and none
2. Possible entries are input, output, work, master, print, and update
3. Comment on any characteristics that could slow running on the 4300

Device Type	Number Tapes or Cyl	Record Format	Logical Record Size	Labels	Reasons for Selection

Figure 6. Example of a file inventory form.

Application	Program Name or Number	Development Programmer	Location of Documentation	Sort Output	User Exit	Description of Exit	Number of Source Statements	Program Size	Language Used	Frequency of Use					Comments
										Hrs./Run	Hrs./Week	Hrs./Month	Hrs./Quar.	Hrs./Year	
										Note 1					Note 2

Notes:

1. Show program frequency of use by comparing hours of run time with columns. For example, a 3-hour run of a quarterly report would show Hrs./Run = 3, Hrs./Quar. = 3, and Hrs./Year = 12
2. Comment on any characteristics that would slow running the program on the 4300 not previously noted. Be sure to note those programs that are without source code. List all file names and file numbers used by the program

Figure 7. Example of a program inventory form.

SCHEDULING THE TRANSITION

After establishing what has to be done and how it is to be done (translation, reprogramming, redesign), you should determine when to do it by developing a transition schedule. The schedule provides an overview of the entire transition project, emphasizing tasks that are executed in parallel.

Depending on the size and makeup of your installation, preparations should begin several weeks to several months before the arrival of your computer. It may take from a few days to many weeks until all the programs are running on the new machine.

In establishing a workable conversion timetable, decide whether you want to run your old and new systems simultaneously for a while. If you do, you will have more time to test and compare original with translated programs. However, budget and space limitations may prohibit you from running them in parallel. The changeover must then occur quickly. This necessitates more extensive preparations and tests during the preinstallation planning period.

After you have prepared your inventory of programs, plan the order in which they are to be translated. Once this is done, you will know how much file conversion is required.

IMPLEMENTING THE TRANSITION

The actual implementation of the transition can be considered in two distinct phases: preinstallation, including system generation and file conversion, and postinstallation.

PREINSTALLATION TASKS

Before receiving the 4300 Processor, you may wish to begin file conversion and program testing at the nearest IBM data center that has the configuration you need. You should order extra tapes to be used during conversion testing. The number you need will depend on the number of programs and the sizes of your data files.

Choose a sampling of programs and their corresponding files for translation. The usual procedure is to choose the programs you would run in a typical day's work, translating the easiest and most frequently used programs first. You can also begin documenting the execution time of translated programs.

Make certain that the programs chosen do not depend on the output of other programs you will not be translating. If possible, use real data and run the test program on the 4300 Processor and in parallel on your System/3. You may decide against using a complete file that tests all parts of a program in one run. Debugging such a test situation might be difficult. Instead, you might decide to create special test data, where the easier portions of the program would be tested first, the more difficult parts next, and exception situations last.

If possible, have your operator present during the entire testing period. Remember that your operating procedures will be different from what you are accustomed to on the System/3. It is advisable to begin creating new run books and operator documentation even in the testing period. It would also be wise to have a lead programmer present, both for training purposes and to assure smooth program operations.

When you first begin executing programs at the data center, you will be using their or some other installation's version of DOS/VSE. After having run some programs successfully, you will want to order a DOS/VSE system, generate it, and execute your translated programs, thereby duplicating the conditions at your installation once the 4300 Processor is delivered.

SYSTEM GENERATION

Planning for the generation of your system is one of the most important preinstallation tasks. One of the best ways to prepare for system generation is to hold a system review. All members on the conversion team should be present since they can help you:

- Determine location of and space allocations for DOS/VSE libraries
- Determine space allocation and disk location for your DOS/VSE files
- Decide partition sizes, if relevant
- Establish disk and tape label standards
- Install a documentation procedure

- Decide upon supervisor parameters best suited to your needs
- Calculate the number of 4300 disks you will need to hold your system and files.

After the review, you must begin to prepare the instructions for system generation and the jobstream you wish to process.

System generation is the preparation of a standard operating system tailored to your needs and based on the generalized DOS/VSE system distributed by IBM. Each installation will incorporate those functions required by the configuration and by the combination of system and application programs.

Although the system shipped by IBM is capable of immediate operation, most installations will:

- Generate supervisors adjusted to ensure maximum efficiency for the configuration
- Create private libraries
- Compile, assemble, and link-edit their own and IBM-supplied programs, and catalog them in the appropriate libraries
- Delete unnecessary components to free additional disk space.

If you plan in detail for system generation you will not have to repeat these procedures. Each option to be included in the supervisor must be evaluated in terms of the functions and performance needed and the real storage required. Planning the supervisor is one of the most important steps in system generation. The System Generation chapter provides additional information.

CONVERSION OF FILES

The conversion of data files must be timed so that they are not out of phase with the programs that use them. The program inventory form (Figure 7) contains a column for listing all files used by each program. Because reprogramming, translating, and replacing programs are generally done by application area, programs that use files from more than one application may have one or more data files out of phase. File maintenance programs are an example of programs that use files from more than one application. You should plan to execute a trial conversion of your files, before the actual conversion, to ensure your data conversion programs are working correctly.

POSTINSTALLATION TASKS

Once your 4300 Processor is installed you can begin running your converted applications on it, assuming they have all been tested and you have generated your own system. Before running your applications, you should ensure that the most current versions of your data files are available to be processed on the 4300. This may mean that you want to specifically schedule the conversion of certain data files - for example, after month-end processing for general ledger, or the end of a pay period for payroll.

CHECKLIST

This section contains a summary of the actions to be taken through the final stage of the transition from the System/3 to the 4300 Processor.

1. Organize your department for conversion
 - Establish objectives
 - Appoint conversion coordinator
 - Set up review procedures
 - Periodic system reviews within department
 - Checkpoint meetings with IBM personnel
 - Order new library of hardware and software publications
 - Determine manpower requirements for conversion and choose personnel
2. Establish education plan
 - Review IBM curricula for topics such as:
 - System generation
 - Job Control Language
 - Operating procedures
 - RPG II, COBOL, FORTRAN, Assembler
 - Data management
 - Supervisor and I/O macros
 - Sort/merge
 - System utilities
 - Schedule classes
3. Develop detailed conversion plan
 - Establish approach to program and file conversion
 - Develop plan for parallel testing to minimize production slowdown
 - Assign tasks to personnel
 - Establish program testing standards
 - Take inventory of programs and files and state interdependencies
4. Review data center facilities
 - Check availability of programming facilities well in advance, especially:
 - Supervisor requirements
 - Sort/merge and utilities availability
 - Check hardware configuration
 - Check device characteristics
 - Tape densities
 - Disk types
 - Model numbers
 - Verify device addresses and system main storage size
 - Determine whether alternate solutions or device substitutions are required
 - Determine backup procedures
 - Investigate number and location of test sites you require, for example:

- For dumping a System/3 with tape and disk in one location
- For file restoration, a 4300 Processor in another location
- Schedule time at data center
- Inquire about availability of scratch tapes and DASD files

5. Contact IBM services

- Field engineering hardware specialists
- Field engineering software specialists
- Systems engineers
- Sales personnel

6. Take inventory of tape and disk libraries

- Check whether you have enough tapes, disks, and diskettes
- Order more, if necessary
- Check delivery schedules

7. Implement conversion plan

- Convert programs (RPG II, COBOL, FORTRAN, Assembler)
- Convert existing sorts and utilities
- Convert OCL to JCL
- Convert files
- Investigate possibility of changing record sizes to take advantage of new DASD devices
- Check I/O area requirements
- Determine data set placement on each file
- Review file organization techniques
- Standardize terminology (label procedures, file names)
- Write any special programs required to dump and restore files

8. Review status of equipment on order

- Check for correctness and completeness
- Confirm delivery schedule

9. Review status of programming system

- Take inventory of current system usage and compare with DOS/VSE
- Order DOS/VSE release and check delivery schedule
- Order program products
- Order optional material (for example, source listings of components) if program logic descriptions are needed.

10. Lay out physical plan with IBM

- Check site and schedule improvements, if necessary, for:
 - Floors
 - Air conditioning

- Power
 - Cable lengths
 - Lighting
 - FE service clearance
- Review requirements for parallel testing and concurrent operations
 - Check communication equipment requirements
 - Ensure correctness of data files
 - If not correct, order correct ones and check delivery schedule
 - Check physical requirements for delivery
 - Door height and width
 - Corridor width
 - Elevator capacity
 - Rigging requirements
 - Disk and tape storage cabinets
 - Disk and tape library requirements
11. Plan for system generation
- Hold review with staff
 - Hold review with IBM systems and field engineers
 - Determine supervisor parameters and size
 - Begin plan for future expansion
 - Prepare system generation macros
 - Determine standard assignments and labels
 - Generate system
 - Test system with sample programs or with samples of your own programs
 - Generate system again if necessary
 - Review PTF requirements with IBM
12. Complete conversion to new system
- Have all programs running on the new system
 - Complete file conversion
 - Run parallel tests and verify results
 - Balance results with control procedures
 - Run new work on new system
13. Release old system
14. Hold periodic reviews
- Check system performance
 - Determine where incremental improvements can be made
 - Determine where performance and efficiency can be improved
15. Plan major redesigns and expansion

EXPANDING THE SYSTEM

Up to now, the principal objective has been to run existing and new applications smoothly using a DOS/VSE system that efficiently handles your programs. Once this is accomplished, it is time to begin implementing the full possibilities of your new system. It is important to grow gradually and not radically. Some of the steps you can take are:

- Create job accounting routines to help you monitor system usage
- Write checkpoint/restart routines for programs with long execution time
- Group related programs that depend on the previous program's output as the steps of one job. The entire job can thus be canceled automatically should one jobstep fail.
- Catalog your application programs in private libraries to give you greater flexibility in planning your disk space and to simplify operations.
- Improve comments to the operator in job control language statements, thereby speeding I/O setup
- Convert tape files to disk to improve execution time
- Organize disk storage space to provide the fastest possible access to your files
- Put record descriptions in the source statement library and consolidate terms used as labels
- Create an inventory of jobs and job characteristics, including information that will help document jobs and programs, such as:
 - Process-versus I/O-bound
 - Configuration requirements
 - Printer forms and number of copies required
 - Priority on daily schedule
 - Volume of unit record files and their processing time

Once your System/3 has been successfully replaced by the 4300 Processor, and you have become familiar with the functions and options of DOS/VSE, you can again consider new application development. You might want to install a data communication system to make your company's data easily available in the departments where it is needed. If you have remote processing locations, you might consider using remote job entry facilities so that your remote sites can benefit from the speed and power of your 4300. You might also consider organizing your data into a data base, making it easier for your programmers to implement new applications.

Your IBM marketing representative is available to provide any information you need.

This chapter will discuss the considerations for converting your data files from the System/3 to DOS/VSE. Since many of your System/3 devices can be attached to the 4300 Processor, you may not have to convert some of your files. The following sections will describe the data compatibilities and incompatibilities, and techniques for conversion.

Each of your data files must be evaluated to determine if it needs to be converted. Card, tape, and diskette files should need few changes to be processed with DOS/VSE. Your disk files will have to be converted, because of the different disk formats and logical file organizations available on the 4300. Be aware that some changes to files can affect other areas of your conversion effort such as OCL to JCL conversion, program conversion, utilities, or supervisor options. Each of these items will be discussed in the appropriate section.

CONVERTING CARD FILES

If you plan to use the same kind of card devices (80-column or 96-column) on your 4300 Processor as you did on your System/3, you should not have to change your card files. The physical deck of cards is readable on both systems.

If the card device(s) on your 4300 Processor will process the same kind of cards (80- or 96-column) as your System/3, but is a different physical device, you must evaluate the features on the two sets of card read/punch devices. If the capabilities are different, you may have to make some programming changes or changes to utilities. The capability may not be supported on some devices - such as card printing. (Card printing can be accomplished with the interpret feature on a key-punch or data recorder.)

Some considerations for processing 96-column cards on a 4300 Processor, or converting 96-column cards to 80-column cards are given below.

96-COLUMN CARDS

If you currently have 96-column cards and plan to keep that capability on the 4300, you must consider the following points.

1. The MFCU on the System/3 normally does not punch leading zeros in a numeric field (RPG II header option allows punching of leading zeros); the 5496 data recorder does not punch leading zeros in a right-adjust numeric field. The MFCU on the 4300 does punch leading zeros under DOS/VSE. These differences can affect the results of compares, particularly the LEVEL BREAK compares of RPG II. This problem would occur only with a mixed deck of cards, some punched with and some without leading zeros. This would probably be a master card file and not a transaction file. To eliminate the problem, you can write a simple RPG II program, defining the numeric fields, read the old deck and punch a new deck on your 4300; leading zeros will be punched. Then, to keep new input cards consistent, punch leading zeros on the 5496.
2. When you order your DOS/VSE software, be sure that you specify the feature for program maintenance on 96-column cards.
3. The MFCU on the System/3 and the MFCU on the 4300 Processor punch according to a different character set. The MFCU on the System/3 punches in 6 rows the 64-character set corresponding to the 96-column card code. On the 4300, it punches in eight rows representing a

256-character set. This will not be a problem unless you begin to use special characters that you did not use on your System/3. Eight-row punching in columns 33 - 96 can overpunch print positions 65 - 128, making the card difficult to read.

Object decks will be different on the 4300 Processor and the System/3, since they will be punched using a different coding method (8 rows vs 6)

If you anticipate the need to modify your object decks using card replacements with the REP card, then you should plan to have a special feature (8-bit Read/Punch, Feature number 3666) available on at least one of your 5496 data recorders. This feature will allow you to read and punch the 8-hole special characters. DOS/VSE provides a utility program, OBJMAINT, that allows you to make changes to object decks in the core image library if necessary. This program allows you to punch a plus symbol (+) instead of the standard 12-9-2 punch in column 1 of the REP card. This eliminates the need for a special feature on the 5496.

4. Another consideration with 96-column cards is that DOS/VSE supports its SYSIN files (system input) as 80- or 81-byte records only. (With 81-byte records, the data record is in bytes 2-81.) This means that although DOS/VSE can physically read the 96-column cards as SYSIN, it will make use of only 80 characters of data. This will affect you if you assign SYSIN as the logical unit for your card input files. Some compilers, RPG II for example, read their input as SYSIN. This means that tables and arrays must be changed to an 80-column format with the appropriate FILE, INPUT and EXTENSION specifications changed as well. If your own data files are specified as SYSnnn, where nnn is from 001 to the maximum for your system, then the full 96 columns can be read. Job control information must use only the first 80 columns of the 96 column card, however.
5. If you have card data files that will be transmitted to your 4300 from a remote terminal via VSE/POWER Remote Job Entry (RJE), you must be aware that VSE/POWER RJE will use only 80 or 81 bytes of the record. This may mean that you need to redesign those card files at the remote site to use no more than 80 columns.

96- TO 80-COLUMN CARD CONVERSION

If you plan to change from 96-column cards to 80-column cards when moving from the System/3 to the 4300 Processor, be aware of the following points.

1. Since the 80-column cards contain less data than 96-column cards, you will have to redesign the card format of any card having data extending past column 80. This may mean that a program now has to read two or more cards for the same information. Your program will have to be changed to handle the extra data input formats. You should consider putting special record codes in your cards so that your program can determine which data is processing. Program source decks with data extending beyond column 80, such as tables and arrays, will have to be changed to the 80-column format.
2. Another alternative may be to use the packed decimal data representation in the 80 column cards. This method allows the System/3 BCD data to be compressed into fewer columns in the 80 column card; for example, a five position System/3 field will only use three columns when packed decimal format is used. If the 96 column card has enough numeric fields it could fit into an 80 column card.
3. You will have to physically convert your 96-column cards to 80-column cards. You can do this in several ways.

- If you have access to a System/3, SYSTEM/370, or 4300 Processor with both types of card devices attached (80 and 96) you can write an RPG II or COBOL program that will read the 96-column card and punch one or more 80-column cards in the new formats that you have designed.
 - You can load your 96-column cards to tape on a System/3 or 4300 Processor with a tape device and a 96-column card reader. This program might be \$KCOPY, \$COPY, or DITTO on the System/3, VSE/DITTO on the 4300 Processor, or an RPG II or COBOL program you have written. You can then take the tape to a System/3 or a 4300 Processor with a tape device and an 80-column card punch. You will need to write an RPG II or COBOL program to read the tape file and punch the new 80-column card formats.
 - If the above configurations are not available or if your card data files are very small, you can simply repunch your data using the new formats on an 80-column keypunch or data recorder.
3. The 80-column card device on the 4300 Processor may not have all of the features of the 96-column MFCU on the System/3, such as the ability to print a card that was just read or the ability to stacker select. You must evaluate the features of the two card devices to determine if all of the needed functions can still be performed, or are still needed.

96-COLUMN CARD TO DISKETTE CONVERSION

If you have 96-column cards on the System/3, another option you have is to convert them to 96-byte records on diskettes, and use diskettes as data input to the 4300 instead of cards. This would eliminate the need to redesign and reprogram as when converting to 80-column cards from 96-column cards that have data extending past column 80. However, diskettes specified as SYSIN or those transmitted via VSE/POWER with Remote Job Entry will be processed as 80- or 81-byte records. Refer to "Converting Diskette Files" later in this chapter.

CONVERTING TAPE FILES

Tape data files are generally compatible from System/3 to the 4300 Processor, assuming that the track formats (7 or 9 track) and tape densities (usually 800 or 1600) are the same on your System/3 and 4300 Processor tape devices. If not, then you must physically convert the tape files. This can be accomplished in several ways.

1. If you have access to a 4300 or SYSTEM/370 with both types of tape that you need, you can use VSE/DITTO or write your own RPG II or COBOL program to copy tape to tape.
2. You can configure one or more of the tape devices on your own 4300 to include dual density (800 or 1600 bpi) or 7-track compatibility to process your special tape files. See your IBM marketing representative for more information.

Some other tape file considerations are the following:

1. System/3 standard labeled tapes are compatible with DOS/VSE standard labeled tapes. System/3 nonlabeled tapes are compatible with DOS/VSE nonlabeled tapes.
2. Both System/3 and DOS/VSE support multivolume tape files.
3. Both System/3 and DOS/VSE supports multifile tape volumes. However FORTRAN programs cannot process multifile volumes.

4. If you are storing packed data on your tape, be aware of the difference in the + sign between System/3 and DOS/VSE. On System/3 a packed + field has a sign of "F". On a 4300 the packed + field has a sign of "C" after an arithmetic operation. This difference can be significant in an alphanumeric character compare. The problem occurs if you are processing with a mixture of data, some created on the System/3 and some created on the 4300 Processor. (The 4300 can read the data properly as a packed numeric field.) You should reload the file on a 4300, specifying the particular field(s) as packed data. This will change the sign value.

The System/3 also allows nonnumeric information to be packed. This can happen when a previously unused portion of a record (initialized to blanks, an X'40') is used for packed data. Such a data field has an invalid sign to DOS/VSE and will cause a program check during arithmetic operations. However, DOS/VS RPG II recognizes this condition and creates a valid sign before processing the field, avoiding the program check. For other programming languages, you can write a special program to check fields for this condition and initialize them to zeros.

CONVERTING DISKETTE FILES

In general, your System/3 diskette files do not have to be converted to be used on a 4300 Processor. Some special considerations are listed below.

1. Diskette files of System/3 OCL will have to be reloaded with DOS/VSE JCL statements. (See the chapter on OCL to JCL for conversion information.) DOS/VSE will read only 80 characters of the JCL statements.
2. Diskette data files in your programs should be given a logical SYSnnn name (with nnn being from 001 to the maximum number allowable for your system). Diskette files specified as SYSIN will be processed as 80-byte records.
3. Diskette files can be read on a locally attached 3540 diskette reader, a user diskette reader(4300 only), or by VSE/POWER Remote Job Entry (RJE) from a 3741 or other devices. A 3741 cannot be locally attached to a 4300. If you have diskette files that will be transmitted to your 4300 from a remote terminal via VSE/POWER, VSE/POWER RJE will use only 80 or 81 bytes of the record. This may mean that you need to redesign those diskette files at the remote site that have data past column 80.
4. If you are planning to install a "cardless" 4300, some special points apply to your diskette files and other system options. You should read the Planning Guide for the IBM Disk Operating System/Virtual Storage (DOS/VS) "Cardless System", GC20-1786.
5. The DOS/VSE System Utilities contains a program called Copy and Restore Diskette (CRDR) to replace labels on a diskette, copy the entire contents of a diskette on to another diskette, and eliminate special records from the file. See the DOS/VSE System Utilities Manual GC33-5381 for further information.
6. The DOS/VS 3540 Diskette Utility (FDP 5798-CN2) provides additional utility functions for diskette files including diskette to card/tape/printer, card/tape to diskette, diskette scan, and alter diskette.

CONVERTING DISK FILES

When planning for the conversion of your data files from the System/3 to DOS/VSE, your disk files will probably take the most time. This is because, first, most of your data is probably disk resident, especially the master files containing your company's most important information. Second, there are more differences between System/3 and DOS/VSE disk files than with other types of files. The differences are in the areas of logical file access methods and the physical organization of data on disks. For these reasons all of your disk files will need to be converted.

In planning the conversion of your disk files, you must do the following:

- Select a file organization and access method for each file
- Select physical file factors such as block size (for non-fixed block architecture DASD) and placement of the files on disk
- Determine the impact of some data differences such as the sign of a packed field
- Choose the specific procedures to convert each disk file.

VIRTUAL STORAGE ACCESS METHOD

VSE/VSAM (program number 5746-AM2) is a file management system that allows three different methods of data organization, each of which permits different ways of processing. The three kinds of data organization are:

- Entry-Sequenced Data Set (ESDS)
- Key-Sequenced Data Set (KSDS)
- Relative Record Data Set (RRDS)

In an entry-sequenced file (ESDS) records are stored in the physical sequence in which they are loaded, like System/3 sequential files. New logical records are stored at the end of the file. A record may be retrieved sequentially or randomly based on its physical location within the file.

In a key-sequenced file (KSDS) records are stored in the collating sequence of a key field, such as customer number, and access to records is generally via an index like System/3 indexed files. A logical record may be retrieved randomly or sequentially according to its key.

In a relative record file (RRDS) records are stored according to their relative record numbers like System/3 direct files. A logical record may be retrieved randomly or sequentially based on its relative record number.

With these three kinds of file organization and the kinds of processing that they allow, VSE/VSAM provides almost total compatibility with the System/3 file organizations. VSE/VSAM files can be input to the DOS/VS Sort/Merge program. The entry-sequenced file may have records added to it at the end of the file. Records added to a key-sequenced file are placed in physical collating sequence, so that no long chains are created. VSE/VSAM key-sequenced files may be processed within limits set

by the program. Relative record files provide the capability of processing a file randomly by relative record number, as well as sequentially. VSE/VSAM also provides automatic space allocation for files. VSE/VSAM provides a program called Access Method Services, which includes a full range of file creation and service commands. VSE/VSAM also provides the capability to allow processing against one file from multiple partitions.

The file capabilities provided by System/3 that are not available with VSE/VSAM are: processing of a key-sequenced file in a consecutive manner without using the index, processing a key-sequenced file via an ADDROUT file, and processing a key-sequenced file via relative record number.

A key-sequenced file may be processed sequentially in a different order than its prime index by the creation of an alternate index. This would offer a facility similar to the ADDROUT processing facility.

SEQUENTIAL ACCESS METHOD

The DOS/VSE Sequential Access Method allows a programmer to store and retrieve the records of a file in consecutive order. In the System/3, a sequential file may be extended by having records added at the end of the file. In DOS/VSE you cannot extend a sequential file. VSE/VSAM should be used for those programs that need the facility for extending a sequential file. The System/3 allows a sequential file to be processed by relative record number to get the third, tenth or fiftieth record of the file. DOS/VSE allows only consecutive processing of the records. Both System/3 and DOS/VSE sequential files may be input to sort programs.

ISAM AND DAM ACCESS METHODS

Index Sequential and Direct Access methods are not supported for FBA DASD devices under DOS/VSE. It is recommended that the Index Sequential files be created as VSE/VSAM files. Programs can then be run using the ISAM interface function that is available with VSE/VSAM. With this approach, programs that have ISAM files require no changes in order to be run using FBA DASD devices. Programs using the DAM access method will have to be recoded in order to take advantage of FBA DASD devices.

OTHER ACCESS METHOD CONSIDERATIONS

There is an additional consideration to be made when converting System/3 disk files to DOS/VSE SAM. With the System/3, if you did not specify a location for your disk files, the system would automatically select a place to put the file and create the VTOC entry. This is useful for temporary files such as work files. With DOS/VSE SAM you must specify the relative block number where the file is located.

SUMMARY

As you can see, VSE/VSAM files are most functionally equivalent to the file organizations of the System/3. VSE/VSAM may require more main storage than the other DOS/VSE access methods, but the additional functions that are provided may eliminate file conversion problems and program changes. The RPG II Conversion Preprocessor Program normally will convert all of the file specifications for disk files in your RPG II programs to a VSE/VSAM format. The System/3 device specification is replaced by a VSE/VSAM data set type, for the following types of files:

<u>System/3</u>	<u>VSE/VSAM</u>
Table or array	ESDS
Sequential or direct	RRDS
Indexed, with unordered load	ESDS
Indexed, processed by ADDROUT	ESDS
Indexed, processed sequentially	RRDS
Indexed, processed randomly by relative record number	RRDS
Indexed, other	KSDS

ESDS = entry-sequenced file

KSDS = key-sequenced file

RRDS = relative record file

PHYSICAL DISK FILE FACTORS

For non-fixed block architecture DASD, the type of device your sequential records will be stored on affects the block size you choose. You will need to analyze the files you are converting, and choose an appropriate block size to avoid wasting space on disk.

When a fixed block architecture DASD device is used on the 4300, the programmer needn't be concerned with the physical characteristics of the disk or how data is written to disk. Data is written to and retrieved from disk out of areas called control intervals. A control interval may consist of one or more logical records but the program only sees the logical record it requested. When you define a SAM or VSE/VSAM file you specify the size of the logical record and optionally the size of the control intervals (CI). The programmer needn't specify any blocksize or blocking factor. The method of storing information in VSE/VSAM and SAM files is different, however.

Detailed information on selecting a control interval size can be found in the manual DOS/VSE Data Management Concepts GC24-5138.

VSE/VSAM - The information about your file is stored in a VSE/VSAM catalog, which is a special area of disk that you allocate for VSE/VSAM's use. Using information stored in the catalog VSE/VSAM manages where the files is written on disk. VSE/VSAM automatically acquires space for files from the area available to the VSE/VSAM catalog.

SAM - The area used for a SAM file is defined by the use of // DLBL and // EXTENT statements that are either in the job stream or stored permanently on disk.

DATA DIFFERENCES

System/3 disk data formats are compatible with DOS/VSE disk data formats, with the exception of the sign value in a packed positive

field. On System/3 a packed + field has a sign of "F". On the 4300 a packed + field has a sign of "C" after an arithmetic operation. This difference can be significant in a character comparison, especially if you use packed keys for your disk files. You can eliminate this problem when you load your DOS/VSE disk files by repacking the key in a DOS/VSE format or preferably by using unpacked keys.

One other consideration is that your System/3 disk file may contain packed blank (X'40') fields. This might have happened if you defined previously unused portions of your records (initialized to blanks) as packed data fields. If no data was actually placed in these fields, then they still contain blanks. Such fields have an invalid sign to DOS/VSE and will cause a program check during arithmetic operations. However, DOS/VS RPG II can recognize this condition (via an option in the 'H' control card) and create a valid sign before processing, avoiding the program check. For other programming languages, you can eliminate this problem by writing a program to check the field for this condition, and then initialize it to zeros. You may do this in your file load program.

CONVERSION PROCEDURES FOR DISK FILES

Once you have evaluated your disk files and selected their new logical organizations and physical formats, you must plan for the actual conversion of the files. This involves several steps:

- Determine the method of conversion
- Code any necessary programs or utilities
- Test the file conversion by creating test files for your application programs
- Determine the cutover schedule for your production files
- Convert your production files

There are several methods of converting your System/3 disk files to DOS/VSE. The method used will depend on the availability of the properly configured System/3 and 4300 Processor.

VSE/IBM System/3-3340 Data Import program 5746-AM3.

This utility is the most convenient method of conversion since it allows you to read directly from a System/3 3348 data module and write DOS/VSE SAM or VSE/VSAM files to a 3340 or fixed block architecture DASD device. The Data import utility is a licensed program product that requires the following:

VSE/Advanced Functions, program no. 5746-XE8

System/3-3340 data import feature on the 4300 Processor

VSE/VSAM, program no. 5746-AM2

The utility can read one or more files in a single run from the prime data area of a 3340 module and create data sets that are compatible with DOS/VSE SAM or VSE/VSAM. There are no other special programs required. The data import utility will run under DOS/VSE with Advanced Functions and requires a minimum of 256K bytes of virtual storage. Multivolume and multiversion files can also be converted. DOS/VSE naming conventions must be applied to multiversion files. The 3348 module must have been written on a System/3 using standard IBM data management routines. Only the main data area of the 3348 is copied. Files in one of the simulated areas of a 5444 have to be copied to the main data area using the System/3 \$SCOPY utility. The following figure illustrates the types of files that may be converted.

S/3 FILE TYPE	DOS/VSE DATA SET
SEQUENTIAL, DIRECT (Note 2)	SAM, VSE/VSAM ESDS OR RRDS (Note 1)
INDEXED ORDERED or UNORDERED	VSE/VSAM KSDS (Note 1)
INDEXED UNORDERED	VSE/VSAM ESDS (Note 3)

Note 1: Other VSE/VSAM file types possible, required by record format and/or application.

Note 2: Handled as sequential file.

Note 3: To get indexed capability, a secondary index must be built using the BLDINDEX function of VSE/VSAM.

Intermediate devices

If you chose to use an intermediate device for transferring files the following methods should be considered.

TAPE

If you plan to use tape as an intermediate step you would first code a utility to unload your System/3 disk file to tape. You could use the System/3 \$COPY or \$KCOPY program to do this, or write an RPG II or COBOL program. The program that you would need on the 4300 to reload your disk file depends on the data organization you have selected.

If your file will be a VSE/VSAM file, you can use the VSE/VSAM Access Method Services command REPRO to load the file, once the VSE/VSAM file has been DEFINED. You may also use VSE/DITTO or write your own program to load the VSE/VSAM file.

If the new disk file will be a sequential file, you can write your own program or use VSE/DITTO to load the file.

CARDS

If you plan to use cards as an intermediate step in converting your disk file, then the procedures to follow are similar to the procedures for tape. On the System/3, however, you must write your own programs to unload the disk files to cards if your disk data records are greater than the card size (80 or 96 bytes), since the utilities \$COPY and \$KCOPY will truncate the input record. If your records are shorter than 80 or 96 bytes, then you may use the utility programs. Again, the program that you will need on the 4300 to reload your file will depend on the type of file being loaded.

If your new file will be VSE/VSAM, you must load your cards onto a sequential disk or tape file to be able to use the Access Method Services REPRO command. You may also write your own program to read the cards and write the VSE/VSAM file.

If your new file will be a sequential file, then you can write your own program or use VSE/DITTO if your logical record is contained within one card.

DISKETTES

If you plan to use diskettes as an intermediate step in converting your disk files, you will have to write your own System/3 programs to unload the disk files to diskettes if your disk data records are greater than the fixed diskette data record size of 128 bytes. If your disk records are 128 bytes or less, \$COPY or \$KCOPY may be used to unload disks onto diskettes. In either case before you unload the disks you must create DOS/VSE header labels (HDR1) on the diskettes.

If your 4300 file is to be VSE/VSAM, unload the diskettes onto a sequential disk or tape file. You can then use the Access Method Services REPRO command to create the VSE/VSAM file. An alternative is to write your own program to read from diskette and write directly to the VSE/VSAM file.

3348 Data Module Considerations

When the System/3 Data Modules are used in conjunction with the VSE/IBM System/3 - 3340 Data Import program they are not altered. The Data Import program can read from the prime data area of the System/3 data modules and create files that are usable with the 4300.

When the System/3 files have been converted the 3348 data modules may be used on the 4300 Processor after some preparation. Before the 3348 data module is used on the 4300, you must run the reset alternate track program, \$RSALT, on your System/3 to reset the flags for the alternate tracks. The System/3 and 4300 Processor use different locations on the data module to assign alternate tracks. Once you have run the utility on your System/3, you can reinitialize the 3348 on your 4300 with the DOS/VSE Initialize Disk utility, INTDK. The 3348 data module is then ready to receive your 4300 files.

Unload and Reload Programs

You can save coding effort if you use the available System/3 and DOS/VSE utility programs to aid in your file conversions. The System/3 programs that are available are:

- \$COPY
- \$KCOPY
- DITTO
- Disk Sort (with tape output)

The DOS/VSE programs available are:

- Clear Disk (CLRDK)
- VSE/VSAM Access Method Services
- VSE/DITTO (program number 5746-UT3)

If these programs do not meet your needs, then you can write your own RPG II or COBOL programs. Two of the most common reasons for writing your own programs are to drop records that have been flagged for deletion on the System/3 or to reformat data fields as discussed under Data Differences earlier in this chapter. Once you have coded the program for one file, then the program can be used for your other data files with probably only minor changes.

VSE/VSAM Installation Steps

A good description of installing and using VSE/VSAM is included in the DOS/VSE Entry User's Guide. A summary of the steps required to create a file is given below. Subsequent file creation will require only the last two steps.

- Install the VSE/VSAM program product into your system libraries (including Access Method Services)
- Define the VSE/VSAM master catalog using Access Method Services
DEFINE MASTERCATALOG
- Define the space VSE/VSAM is to manage using Access Method Services
DEFINE SPACE
- Define the file attributes using Access Method Services
DEFINE CLUSTER
- Load the file using Access Method Services REPRO or your own program

Testing

A convenient way to verify your disk file conversion programs is to use them to create files for the testing of your application programs. Schedule a trial conversion sometime before the cutover date to the 4300 and the final conversion of your live files. You will then have time to correct any problems. For earlier version of test files, you might consider adding some code to the reload program to limit the number of records put on the files. This would give a more manageable file for test purposes.

Conversion Schedule

Because your disk files require the most conversion effort, and because information in those files is probably being added to or updated every day, you must carefully choose when to cut over from your System/3 to DOS/VSE. The decision you make will also depend on the availability of both a System/3 and a 4300 Processor configuration.

You may plan to convert all your files on a weekend, (while the 4300 is being installed), assuming that all of your 4300 programs have been tested and can begin processing the data on your own 4300 Processor.

If you plan to have both a System/3 and a 4300 Processor for some period of time, perhaps a month, then you plan to move your application systems to the 4300 at a convenient time for the application - perhaps after month-end processing for general ledger or after a pay period for payroll.

ADDITIONAL DATA CONSIDERATIONS

If you use special characters (such as an asterisk *) in numeric fields to indicate a unique action to be taken, you should be aware that in DOS/VSE these nonnumeric characters in numeric fields may cause a program to cancel with data checks. You should review your programs to determine if you will need to make program and/or system design changes.

This chapter will discuss the considerations for moving application programs from the System/3 to DOS/VSE. The first section will present the general procedures for program conversion, and the later sections will present specific information about each programming language including conversion tools, language differences, and JCL considerations.

CONVERSION PROCEDURES

Regardless of the programming languages that you use, the procedures for converting your System/3 application programs to run under DOS/VSE are similar for each language.

First, take an inventory of all of your application programs. In the process of doing this, you should also collect and organize all of the documentation about the program, including flowcharts and especially run instructions, since these will need to be updated. As you gather all of your program information, evaluate each program to see if it is still being used. You will probably find that you have many programs that were written for a one-time job, and are no longer required. These programs will not need to be converted. You should also establish a conversion priority. Programs that are run infrequently can be converted last, even after your 4300 has been installed. In planning the order of conversion, consider which disk or other data files are needed, so that when the program is converted there is a file to test with.

Next, decide how each program is to be converted. If you are planning to make major changes in your system or application design, then you are doing more than converting your programs. That is beyond the scope of this manual. System/3 to DOS/VSE conversion is easier if you simply convert your present systems, and then redesign once everything is running on the 4300 and you become familiar with the DOS/VSE and 4300 Processor capabilities. You may, however, have to do some minor redesign because of System/3 and DOS/VSE differences discussed elsewhere in this manual. There are several ways that your programs can be converted.

The easiest way to make program changes is to use an automated conversion tool, especially if you have a large number of programs. If you have RPG II programs, you should use the System/3 DOS/VS RPG II Conversion Preprocessor (5735-CV1) to convert your programs. If your programs are in another language, such as COBOL, you might want to write your own conversion program (especially if you have a large number of programs with many changes). You must determine whether it would take more time to design and develop a program than to make the changes manually.

Some manual changes may still be necessary, even with a conversion program, because some language differences cannot be detected or converted automatically.

If you do not use an automated conversion program to change your program, then you must make the changes manually. The specific language sections later in this chapter will give you a list of differences for each language. You should review the list for your programming language, decide if your programs will need the changes, and then make a list for your programmers of the items they need handle. You should also refer to Chapter 4, Data File Conversion, for additional programming differences.

Another way to discover some of the differences between the System/3 and DOS/VSE compilers is to compile your System/3 source program with the DOS/VSE compiler. The compiler will flag many of the changes that must be made.

Once you have made the necessary changes to your programs, you must compile them on a 4300. You may use an IBM data center 4300 Processor for this purpose. Review each of the messages that the compiler produces. Some may be error messages that require more changes to the program.

When you have a successful compilation of your program, then you can link-edit and catalog that program into the DOS/VSE core image library. When your test files are created, you can run your application program against the test files. You must have made the necessary changes in your JCL. Review the program output carefully for errors. Check the printed listings, and be sure to examine your disk or tape output to ensure the data is correct - field positions, data formats, etc. VSE/DITTO can be used for this purpose. If there are any errors, you must make the necessary program changes and recompile. Do not underestimate the time it will take you to complete this step of program conversion.

Once your programs for a particular application system are converted, you must review your program documentation and make the necessary changes. This will be particularly important for the operator's run sheets. At a minimum, the JCL needed to run a program will have changed.

After all the necessary documentation changes have been made, you should consider a "parallel" test of the 4300 application with the System/3 application. Make a DOS/VSE copy of the necessary "live" System/3 files, and then run the application programs twice - once on the System/3 and once on the 4300 Processor. Compare the results carefully. This is a good test of your new programs, since you are actually working with live data. It will show you how your entire program and system flows work, and if the operator instructions are accurate. It is a good idea to go through this parallel testing for two cycles. This will check the output of the first cycle as valid input for the second cycle. You should plan to do this testing at least 6 to 8 weeks before you install your own 4300 Processor so that you have time to make any necessary changes.

The following sections will discuss the specific conversion information for each language.

GENERAL PROGRAM CONVERSION GUIDELINES

Below are some conversion considerations that can apply to any programming language.

COMPILER WORKFILES

When you are assembling or compiling source programs you must establish work areas for the compiler as well as for the linkage editor if you want to link or catalog your programs.

The definition of these work areas is normally in the standard label area, in order to save the repeated definition of them in every compile or link-edit run. The standard labels will normally be loaded during automatic system initialization of the system.

Figure 8 illustrates label information for the necessary workfiles. COBOL makes use of the fourth work area.

EXAMPLE OF WORK AREAS

```
// DLBL IJSYSLN,'LINK.AREA',0
// EXTENT SYSLNK,,1,0,79000,1000
// DLBL IJSYS01,'WORK.AREA1',0
// EXTENT SYS001,,1,0,80000,2000
// DLBL IJSYS02,'WORK.AREA2',0
// EXTENT SYS002,,1,0,82000,2000
// DLBL IJSYS03,'WORK.AREA3',0
// EXTENT SYS003,,1,0,84000,2000
// DLBL IJSYS04,'WORK.AREA4',0
// EXTENT SYS004,,1,0,86000,2000
```

Figure 8. Compiler workfiles

OVERLAY PROGRAMS

Because of the virtual storage capability of DOS/VSE it is not necessary to fit programs into the available memory. DOS/VSE automatically handles this by keeping the most used parts of the program in storage, and storing the rest on a direct access device, where it is available if needed. This means that you do not have to write programs with overlays. If you currently have programs on your System/3 that are written with overlays, you should make the program(s) into one large, nonoverlaid program.

PRINTER FORMS CONTROL

If you will be using a forms control buffer (FCB) printer and you print reports on different sized forms, then you will need to create a forms control buffer load for each different type of report. An example of a buffer load program is presented in the DOS/VSE Entry User's Guide.

RPG II PROGRAMS

This section presents the conversion considerations for converting your RPG II programs from System/3 to DOS/VSE.

The DOS/VS RPG II compiler (5746-RG1) offers a high degree of compatibility with the System/3 RPG II compilers. It includes System/3 compatible functions, and native support of VSE/VSAM files. These features were not present in previous DOS RPG II compilers. However, System/3 batch application programs must have some changes made to be acceptable to the DOS/VS RPG II compiler.

Many of the changes that must be made to your System/3 programs are related to the type of files you will be processing - especially disk files. Other changes are necessary for forms control on printer files, and for options on the header (H) card. When converting your System/3 programs you will be converting RPG II source programs and/or Auto Report source programs.

The System/3 DOS/VS RPG II Conversion Preprocessor (5735-CV1) can be an important tool in the conversion of your RPG II and Auto Report programs. The conversion preprocessor can reduce much of the manual effort involved. It consists of a set of programs, written in RPG II. There are two versions. The System/3 version can do much of the program conversion work on your own System/3. The DOS/VS version can do any

conversion work that was not completed before your own 4300 Processor arrived.

The programs that the conversion preprocessor is to convert must be error free. You should correct programs, even with warning messages, before running the conversion preprocessor. Otherwise, the results might be unpredictable.

The conversion preprocessor can convert a single program at a time or a series of programs at one time. Control statements are provided to the conversion preprocessor supplying information about this particular conversion run.

Before you use the conversion preprocessor, you must have made some decisions about how your programs are to be converted. This information is then supplied on the control statements. Some of the decisions you must make are as follows:

- Will your System/3 DASD files be converted to VSE/VSAM or DOS/VSE SAM access method? The conversion preprocessor will normally convert your file description specifications to VSE/VSAM automatically. Specifications for DASD files not being converted to VSE/VSAM must be converted manually.
- Which 4300 devices will replace the System/3 devices? Many of the devices on your System/3 can be attached to a 4300. Some devices will be new.
- Which DOS/VSE symbolic device name (SYSXXX) will be associated with which DOS/VSE devices? DOS/VSE assigns physical devices to a file based on the logical filename.
- Which DOS/VSE symbolic device name will be associated with which filename?
- Which source statement sublibrary will RPG II Auto Report code use? The default sublibrary for RPG II is R.
- Will forms control on the printer be handled by spooling control cards or the special buffer load utility?

The specific requirements for the conversion preprocessor are discussed in the manual System/3 DOS/VS RPG II Conversion Preprocessor Installation and Reference, SC33-6035.

Below is a list of System/3 RPG II and DOS/VS RPG II major differences. You must consider all of these items if you manually convert your programs. The conversion preprocessor will convert most of these automatically for you. Those that it cannot, it will flag with a warning for you to do a manual conversion. Conversion effort will depend on how many of the following items are included in your programs.

MAJOR RPG II DIFFERENCES

Figures 9 - 14 indicate the major differences between System/3 RPG II and DOS/VS RPG II.

Function	COLUMN	System/3	DOS/VS RPG II
Core size to compile	7-9	No longer used	Not used, replace with blanks
Object output	10	Specifies link options and object program destination	Not used, use // OPTION JCL. Replace with blank.
Listing options	11	Controls source program listing	Use the // OPTION JCL. Replace with blank
Core size to execute	12-14	Maximum amount of storage for execution. Can cause overlays to be created	Not used. Replace with blank
Inquiry	37	Used for rollout/rollin	Not used. Replace with blank
Sign handling	40	Not used	For sign checking and forcing
Punch MFCU zeros	44	Blank = do not punch/print leading zeros 1 = punch/print	Not used, must be blank
Unprintable characters	45	Blank = halt on unprintable character 1 = do not halt, print space	Not used, replaced with blank
Shared I/O area	48	Shares I/O areas among 5444 disk files	Not used. Replace with blanks

Figure 9. Header options

Function	Column	System/3	DOS/VS RPG II
Table/array loading	11-18	One or more tables/arrays may be loaded from the same device. FROM filename is not unique	One or more tables/arrays may be loaded from the same device.

Figure 10. Extension specifications

Function	Column	System/3	DOS/VS RPG II
Form length/overflow line	15-24	Requires FL to specify forms length, OL to specify location of overflow line	Specifications for line counter normally not required

Figure 11. Line counter specifications

Function	Column	System/3	DOS/VS RPG II
Filename	7-14	Unique in 8 characters	Unique in 7 characters
Random load of a direct file via CHAIN	15-16, 28	Allowed	Allowed only with VSAM RRDS
File format	19	Blank is treated as an F	Same, but warning issued
Block length	20-23	Size of I/O buffer for program	Size of physical block (ignored for VSAM)
Default record length	24-27	Default is maximum record length for the device ADDROUT files is 3	Default is 80 VSAM ADDRUCUT is 5
Device type	40-46	Specifies System/3 device	Specifies device
Dual carriage feature	40-46	PRINTR2 for dual carriage	Feature not supported by DOS/VSE
SPECIAL file	40-46	CC 19 may be blank. Dual I/O areas allowed, CC 32	CC. 19 must be F, Dual I/O areas not allowed.
CONSOLE file	40-46	Can be used several ways	Can only be DISPLAY file
Symbolic device	47-52	Not used	Specifies logical unit SYSnnn
Labels	53	Not used	S must be specified for standard labels
Special IOCS routine parameter	54-59 (cont)	Table or array name to be used by SPECIAL routine	Allowed by specifying the table/array name in RLABL statement
In core file index	54-65 (cont)	Hold index in core	Not used. Must be blank Meaningless with VSAM
In core cylinder index	60-65	Used for index files	Not used for VSAM.
Number of extents	68 - 69	Required if greater than 1	This is a JCL function
External indicators	71- 72	U1-U8 permitted for file processed by a record address file	Must be blank for file processed by a record address file

Figure 12. File description specifications

Function	Column	System/3	DOS/VS RPG II
SQRT	28-32	Result of SQRT is always half adjusted	Must specify half adjust, if desired, CC 53
Subroutine linkage	28-32 RLABL	Assembler subroutines receive pointers to data areas defined by RLABL statements	Assembler subroutines must access RLABL statements via an EXTRN Subroutines must be rewritten

Figure 13. Calculation specifications

Function	Column	System/3	DOS/VS RPG II
Spacing on console	17-18	5471 must always space at least once before printing	Console must space at least once after printing
Skip before/after	19-22	Indicate line number to be skipped to	Indicate channel to be skipped to
Total output conditioned by LR	23-31	Always printed last	Total output performed in order of output lines. Resequence
L0 usage	23-31	Permitted on output	On calculations only
Overflow indicators	23-31	Permitted on AND line. May cause unusual results	Not permitted on AND line
Special page fields	32-37	PAGE3 - PAGE7 are not special RPG II words	PAGE3 - PAGE7 are reserved words
Multiple output operations	-	May output more than one record for a file in the same RPG II cycle for an update or combined file	No multiple output operations allowed to a combined or update file on same RPG II cycle

Figure 14. Output specifications

OTHER RPG II DIFFERENCES

1. In the System/3, compile time tables/arrays from cards can extend to 96 columns. DOS/VS compile time tables/arrays from cards can extend only to 80 columns.
2. In System/3, the sign of a positive field is X'F'. In DOS/VS, standard positive signs are forced to X'C' for packed fields and to X'F' for unpacked fields. Alphanumeric compares may produce undesired results.
3. If System/3 disk files are being converted to VSE/VSAM, additional entries may be necessary in the file descriptions. See the DOS/VS RPG II Reference Manual for VSE/VSAM specifications. Some processing functions permitted with VSE/VSAM are not permitted with the other DOS/VSE files. See the Data File Conversion chapter.
4. The System/3 program halts on errors. In DOS/VS the error processing procedure is predefined. Generally if an error is detected, the program will terminate unless the program contains appropriate error handling logic. In DOS/VS the H0 indicator is set when an error occurs, such as dividing by zero or an invalid array index. If H0 is not set off by the end of the cycle, the program cancels.
5. DOS/VS RPG II does not support teleprocessing (T) specifications. CICS/DOS/VS does have the capability of high level language support for RPG II. System/3 RPG programs would have to be converted or rewritten to take advantage of the high level interface. The chapter "CCP-CICS/VS conversion considerations" provides information and examples of the CICS/VS interface to RPG.
6. Subroutine linkage is different in DOS/VSE. RPG II programs that link to an assembler subroutine will have to be recoded.
7. Auto Report programs. The System/3 Auto Report language is a subset of the DOS/VS Auto Report language. The U-card and the /COPY statement specify different libraries and JCL.

SAMPLE JOBS

This section includes some sample Job Control Language (JCL) for typical jobs to compile an RPG II program and then catalog it to the core image library for execution.

The examples assume standard labels for the workfiles(see Figure 15).

```
COMPILE
// JOB RPGII COMPILE
// EXEC RPGII
-
- RPGII SOURCE STATEMENTS
-
/*
/ε

COMPILE, LINK, AND EXECUTE
// JOB RPGII COMPILE, LINK AND EXECUTE
// EXEC RPGII,GO
-
- RPGII SOURCE STATEMENTS
-
/*
-
- DATA CARDS
-
/*
/ε

COMPILE AND CATALOG
/ JOB RPGII COMPILE AND CATALOG
// OPTION CATAL
PHASE RPGPROG1,*
// EXEC RPGII
-
- RPGII SOURCE STATEMENTS
-
/*
// EXEC LNKEDT
/ε

EXECUTE
// JOB EXECUTE RPGPROG1
// EXEC RPGPROG1
-
- DATA CARDS
-
/*
/ε
```

Figure 15. DOS/VS RPG II jobs.

COBOL PROGRAMS

This section will present the conversion considerations for converting your COBOL application programs from the System/3 to DOS/VSE.

The DOS/VS COBOL Compiler (5746-CB1) offers a high degree of

compatibility with the System/3 COBOL compilers. However, your System/3 batch application programs must have some changes made to be acceptable to the DOS/VS COBOL compiler.

Many of the changes that need to be made to your COBOL programs are related to the types of files your programs will be processing - especially disk files. Other changes should be made in the Environment Division to properly document your programs. Also, in some cases the default data formats for the System/3 are different from DOS/VS.

Once you have made an inventory of your programs, you should prepare a list of the changes that must be made to them. If the number of programs is large, you could write your own program to make the necessary changes automatically, especially clerical changes such as the documentation items in the Environment Division. This would give your programmers more time to concentrate on the differences not easily detected automatically.

The following sections present the major differences between the System/3 COBOL and the DOS/VS COBOL compilers. It is not meant to be complete, in that it does not mention the extra facilities provided with the DOS/VS compiler, the increased ranges of values such as sizes for fields, or the clauses that the System/3 does not support. It attempts to point out areas important for the System/3 programmer converting a program. The sections are arranged in the order in which they appear in a COBOL program.

The compilers for both System/3 and DOS/VS are updated and enhanced periodically and compatibility items may change. You should review the IBM DOS Full American National Standard COBOL Reference Manual, GC28-6394, and the IBM DOS/VS COBOL Compiler and Library Programmer's Guide SC28-6478, for detailed information on DOS/VS COBOL as you make your program changes.

GENERAL COMMENTS

1. The System/3 will cause a skip to new page by placing a / in column 7 of a source statement; DOS/VS COBOL does not support this, but uses EJECT instead.
2. The System/3 DATE is defined as PIC 9(6) in the format yymmdd. The DOS/VS COBOL CURRENT-DATE is defined as PIC X(8) in the format nn/dd/yy.
3. The System/3 has a special register, LINKAGE-COUNTER for page control. DOS/VS COBOL does not support this.
4. The System/3 allows data fields defined in an FD to be referenced, even after a WRITE has been issued. In DOS/VS COBOL this can cause unpredictable results.
5. Reserved words may be different in System/3 and DOS/VS COBOL.
6. In System/3 the PROCESS statement is used to specify various compiler options. In DOS/VS COBOL the compiler options are specified in the CBL statement or the // OPTION JCL statement.

ENVIRONMENT DIVISION

1. CONFIGURATION SECTION

- a. SOURCE-COMPUTER and OBJECT-COMPUTER should indicate IBM-4300. These are comments to the compiler, but can serve as documentation to you to indicate when your programs have been converted. If IBM-4300 is not specified, a warning diagnostic will be issued.

b. DOS/VS ignores the MEMORY SIZE information.

2. INPUT/OUTPUT SECTION

a. FILE-CONTROL. This paragraph may require many changes because of the different physical 4300 devices and DOS/VS logical file organizations. Most differences occur in the ASSIGN statement where system-name-1 is defined. The DOS/VS format system-name is

SYSnnn-class-device-organization-name

You must provide a SYSnnn, which is the symbolic unit to which the file is assigned that relates to the Sysnn in the EXTENT statement. If your physical devices are different, you must specify the new device type. The name must be three through seven characters. The U in System/3 to indicate files opened for I/O is not used in DOS/VS.

VSE/VSAM files have different requirements for the ASSIGN statement. SYSnnn is required. The other fields are optional, but can be specified for documentation. Organization must be specified as AS for sequential files; it must not be specified for indexed VSE/VSAM files. System/3 does not reserve an additional buffer unless the RESERVE ALTERNATE AREA clause is specified. DOS/VS reserves an additional buffer unless the RESERVE NO ALTERNATE AREA is specified.

ACTUAL KEY in System/3 is a relative record number. In DOS/VS the field contains the relative or actual track address and the record address.

RECORD KEY is the same, but in DOS/VS the key should not include the first byte of the record in unblocked files, or a file in which records are to be deleted.

b. I/O CONTROL. Although the format of the RERUN clause is the same, the system name format is different. In DOS/VS it is the same as in the ASSIGN clause. The value for CORE-INDEX should be recalculated for DOS/VS.

DATA DIVISION

1. FILE SECTION - FD.

a. DOS/VS requires the RECORDING MODE clause.

b. DOS/VS does not support the LINAGE clause.

c. In the BLOCK CONTAINS clause, the value for the same file may be different from program to program for the System/3. For DOS/VS this value must be the physical block size, specified in bytes or number of logical records. If this is not specified on output for DOS/VS the file will be written unblocked.

2. Results of a REDEFINE of a COMP data item will be different, since System/3 truncates COMP items.

3. DOS/VS refers to zoned decimal as EXTERNAL DECIMAL and packed decimal as INTERNAL DECIMAL.

4. In the USAGE clause, in System/3 COMP is external decimal. In DOS/VS COMP is binary; DISPLAY is used for external decimal.

5. The 4300 performs arithmetic instructions on packed data fields only. All fields used in arithmetic operations must contain valid numeric characters to prevent program checks. You may have some blank packed fields in your data. The System/3 permitted this to happen. You

might write a special utility program to check for this situation and reinitialize those fields to zeros.

6. FILLER in System/3 causes a field to default to spaces. In DOS/VS a VALUE must be defined if spaces are desired.

PROCEDURE DIVISION

1. WRITE statement. DOS/VS reserves the first character of the data line for page control, if the ADVANCING option is used; the System/3 does not do this. This means that print lines may have to be defined as 133 characters. The System/3 supports the PAGE parameter for WRITE AFTER ADVANCING; DOS/VS does not.

In DOS/VS the DISPLAY, EXHIBIT, WRITE AFTER ADVANCING all cause the printer to space before printing. A simple WRITE OR WRITE BEFORE ADVANCING cause the printer to space after printing. The System/3 spaces the printer before for WRITE and after for DISPLAY and EXHIBIT. System/3 allows LINE or LINES in the WRITE AFTER ADVANCING statement; DOS/VS allows only LINES.

2. ACCEPT statement. System/3 allows ACCEPT DATE, DOS/VS does not.
3. Division by zero will cause a program check in DOS/VS if the ON SIZE ERROR is not coded. In the System/3 the operator is given an option to cancel or bypass the error.

SAMPLE JOBS

This section includes some sample Job Control Language (JCL) for typical jobs to compile a COBOL program and then catalog it to the core image library. See Figure 16. The examples assume standard labels for the workfiles.

```

COMPILE

// JOB COBOL COMPILE
// EXEC FCOBOL,SIZE=100K
-
-
- COBOL SOURCE STATEMENTS
-
/*
/8

COMPILE, LINK, AND EXECUTE

// JOB COBOL COMPILE,LINK AND EXECUTE
// EXEC FCOBOL,SIZE=100K,GO
-
-
- COBOL SOURCE STATEMENTS
-
-

/*
// ASSGN SYS011,FBA,VOL=DATA01,SHR
// DLBL MASTER,'MFILE'
// EXTENT SYS011,DATA01,0,1,100000,200
/8

COMPILE AND CATALOG

// JOB COBOL COMPILE AND CATALOG
// OPTION CATAL
  PHASE COBPROG1,*
// EXEC FCOBOL,SIZE=100K
-
-
- COBOL SOURCE STATEMENTS
-
-

/*
// EXEC LNKEDT
/8

EXECUTE

// JOB EXECUTE COBPROG1
// ASSGN SYS011,FBA,VOL=DATA01,SHR
// DLBL MASTER,'MFILE'
// EXTENT SYS011,DATA01,0,1,100000,200
// EXEC COBPROG1
/8

```

Figure 16. DOS/VS COBOL jobs

FORTRAN PROGRAMS

This section presents the conversion considerations for converting your FORTRAN application programs from the System/3 to DOS/VSE.

The FORTRAN programs for DOS come in three parts:

- DOS FORTRAN IV Compiler (360N-FO-479)
This is the basic FORTRAN IV compiler

- FORTRAN IV LIBRARY Subprograms (360N-LM-480)
This library contains a number of mathematical functions that can be included in your FORTRAN program. DOS does not provide a commercial subroutines package with FORTRAN.
- FORTRAN IV-DOS Library Option 1 (5746-LM3)
This library provides the routines required to support I/O devices such as the 3340 and fixed block architecture DASD devices.

Most of the differences between System/3 FORTRAN and DOS FORTRAN are in the areas of control cards to specify compiler and overlay options, and in the file specifications for the data files you will be processing.

Once you have made an inventory of your FORTRAN programs, you should prepare a list of the changes that must be made to them. If the number of programs is large, you may want to write your own program to make the necessary changes automatically. You must decide whether it would take more time to design, code, and test this program than to make the changes manually. For further information on FORTRAN you should consult the following manuals:

IBM System/360 and System/370 FORTRAN IV Language GC28-6515
FORTRAN IV Programmer's Guide GC28-6397
System/360 FORTRAN IV Library Subprograms GC28-6596

The following sections will present the major areas where you will have to make changes to your FORTRAN programs. The detailed specifications can be found in the above manuals.

FORTRAN STATEMENTS

Figure 17 indicates DOS FORTRAN statements that should replace certain System/3 FORTRAN statements.

Function	System/3	DOS/VSE
Provide specific options to the compiler	*PROCESS statement	OPTION JCL or FCT FORTRAN statement
Define input/output devices to be used at execution time	Device option statement	ASSGN JCL statement
Specify amount of storage to be used by the load module	CORE statement	-
Specify a priority for a subprogram in an overlay environment	CATEGCRY statement	-
Assign a name to a main program	PROGRAM statement	PHASE statement
Cause a named program to overlay an invoking program and receive control	INVOKE statement	-
Provide for the sharing of a main storage area by two or more main programs, and the variables and arrays to be contained in it	GLOBAL statement	CCMMCN statement
Indicate that for FORTRAN-supplied functions having several names, depending on argument type, the correct function is to be selected by the FORTRAN compiler	GENERIC statement	Not available. Name specific function desired

Figure 17. FORTRAN program statements

GENERAL DIFFERENCES

The logical unit numbers for specific devices are different in System/3 and DOS FORTRAN. The System/3 allows up to 32,767 logical units per program; DOS allows a maximum of 15. On the System/3 you could process any number of disk and tape files up to the maximum. In DOS you can process a combination of up to 11 disk and tape files per program. If a program references more than 11 disk and/or tape files, it must be redesigned by making two programs or reducing the number of files. Small files might be placed in main storage as arrays.

The structure of an overlay program, and the way in which it is specified with control cards, will be different in DOS. With the virtual storage of DOS/VSE you should not have as much of a need for overlay programs.

DOS FORTRAN does not support multifile tape volumes. It can process only one file per volume. DOS FORTRAN cannot process multivolume or multiextent disk data files.

Since DOS FORTRAN has certain record length and block format requirements for its disk files, you should evaluate these requirements for programs of any other language that may process these files, to ensure that they can handle the data correctly. See the FORTRAN IV Programmer's Guide for these requirements.

SAMPLE JOBS

This section includes some sample Job Control Language (JCL) for typical jobs to compile a FORTRAN program and then catalog it to the core image library. See Figure 18. The examples assume standard labels for the workfiles.

```

COMPILE

// JOB FORTRAN COMPILE
// EXEC FFORTRAN
-
-
- FORTRAN SOURCE STATEMENTS
-
/*
/£

COMPILE, LINK, AND EXECUTE

// JOB FORTRAN COMPILE, LINK AND EXECUTE
// EXEC FFORTRAN,GO
-
-
- FORTRAN SOURCE STATEMENTS
-
/*
// ASSGN SYS005,FBA,VOL=DATA01,SHR
// ASSGN SYS006,FBA,VOL=DATA02,SHR
/£

COMPILE AND CATALOG

// JOB FORTRAN COMPILE AND CATALOG
// OPTION CATAL
  PHASE FORPROG1,*
// EXEC FFORTRAN
-
-
- FORTRAN SOURCE STATEMENTS
-
/*
// EXEC LNKEDT
/£

EXECUTE

// JOB EXECUTE FORPROG1
// ASSGN SYS005,FBA,VOL=DATA01,SHR
// ASSGN SYS006,FBA,VOL=DATA02,SHR
// EXEC FORPROG1
/£

```

Figure 18. DOS FORTRAN jobs

ASSEMBLER PROGRAMS

This section presents the conversion considerations for converting your Assembler application programs and subroutines from the System/3 to DOS/VSE.

Because of the differences in System/3 and 4300 architecture and hardware instruction sets, your Assembler language programs will have to be completely rewritten. The program design is still valid, however, because the logic and flow of the program generally need not change.

Since the machine instructions and language features of System/3 and DOS/VSE are different, you should first learn about the 4300 Assembler.

You can do this by taking the Assembler Coding-PI, course code K3600, or the Assembler Language Coding independent study program. Your IBM marketing representative can provide you with additional information.

Since you must rewrite all of your Assembler programs, you can incorporate some of the hardware instructions that the 4300 provides, but which you must do with several instructions on the System/3, such as multiply or divide.

For detailed information about Assembler specifications, macro formats, and how to use the Assembler control statements, you should refer to the following manuals:

OS/VS, DOS/VSE, VM/370 Assembler Language, GC33-4010
DOS/VSE Macro User's Guide, GC24-5139
DOS/VSE Macro Reference, GC24-5140
Guide to the DOS/VSE Assembler, GC33-4024

LANGUAGE DIFFERENCES

The following list contains some of the major differences between the System/3 and DOS/VSE Assemblers. It is not meant to be all-inclusive. It can help you determine the conversion workload as well as what your programmers must learn to do the conversion. The above manuals should be referenced before rewriting an Assembler program.

GENERAL RULES

1. A 4300 instruction is always two, four, or six bytes in length. The operation code determines the length of the instruction.
2. 4300 instructions should be halfword aligned for best performance.
3. The 4300 has 16 general purpose registers. Each register is bytes (32 bits) long.
4. The 4300 provides register-to-register instructions, so that data does not have to be placed in storage as an intermediate step.
5. The label of a constant addresses the leftmost byte of the field.
6. The 4300 provides hardware multiply and divide instructions, and pack and unpack, so it is not necessary to use RPG II or other subroutines for that purpose.
7. Macro instruction formats are not compatible from System/3 to DOS/VSE.
8. The System/3 works with zoned decimal and the 4300 Processor works with packed decimal for arithmetic operations.
9. Linkage conventions for subroutines and subprograms are different for System/3 and DOS/VSE.
10. The Assembler instructions have different operation codes and different parameters for System/3 and DOS/VSE.

SAMPLE JOBS

This section includes some sample Job Control Language (JCL) for typical jobs to compile an assembler program and then catalog it to the core image library for execution. See Figure 19. The examples assume standard labels for the workfile.

```

ASSEMBLE
// JOB ASSEMBLE
// EXEC ASSEMBLY,SIZE=64K
      - ASSEMBLER SOURCE STATEMENTS
/*
/ε

ASSEMBLE, LINK, AND EXECUTE
// JOB ASSEMBLE LINK AND EXECUTE
// EXEC ASSEMBLY,SIZE=64K,GO
      - ASSEMBLER SOURCE STATEMENTS

/*
// ASSGN SYS011,FBA,VOL=DATA01,SHR
// DLBL SAMLOAD,'SEQ.FILE',99/365,SD
// EXTENT SYS011,DATA01,1,0,12000,800
/ε

ASSEMBLE AND CATALOG IN RELOCATABLE LIBRARY
// JOB ASSEMBLE AND CATALOG IN REL
// OPTION DECK
// DLBL IJSYSHP,'DISK.PCH/IPT.FILE'
// EXTENT SYSPCH,DATA01,1,0,30000,500
ASSGN SYSPCH,FBA,VOL=DATA01,SHR
// EXEC ASSEMBLY,SIZE=64K
      PUNCH 'CATALR ASMPROG1'
      - ASSEMBLER SOURCE STATEMENTS
/*
CLOSE SYSPCH,OOD
// DLBL IJSYSIN,'DISK.PCH/IPT.FILE'
// EXTENT SYSIPT
ASSGN SYSIPT,FBA,VOL=DATA01,SHR
// EXEC MAINT
CLOSE SYSIPT,OOO
/ε

ASSEMBLE AND CATALOG IN CORE IMAGE LIBRARY
// JOB ASSEMBLE AND CATALOG
// OPTION CATAL
      PHASE ASMPROG1,*
// EXEC ASSEMBLY,SIZE=64K
      - ASSEMBLER SOURCE STATEMENTS
/*
// EXEC LNKEDT
/ε

EXECUTE
// JOB EXECUTE ASMPROG1
// ASSGN SYS011,FBA,VOL=DATA01,SHR
// DLBL SAMLOAD,'SEQ.FILE',99/365,SD
// EXTENT SYS011,DATA01,1,0,12000,800
// EXEC ASMPROG1
/ε

CATALOG FROM RELOCTABLE LIBRARY
// JOB CATALOG ASMPROG1 FROM REL
// OPTION CATAL
      PHASE ASMPROG1,*
      INCLUDE ASMPROG1
// EXEC LNKEDT
/ε

```

Figure 19. DOS/VSE ASSEMBLER jobs.

This chapter will discuss the considerations for converting your job control statements from the System/3 to DOS/VSE. The first section will present a comparison of the System/3 and the DOS/VSE "job", the rules for identifying which devices a program is to use and the facilities that DOS/VSE provides to reduce the need for numerous job control statements in each job, including cataloged procedures. Later sections will present conversion approaches and a comparison of OCL and JCL statements.

JOB CONCEPTS

The DOS/VSE concept of a job is very much like that of the System/3 Model 15. The unit of work to be done is called a job. A succession of jobs is called a jobstream. A job may consist of the execution of only one program, called a jobstep. The job and its environment are defined to the system by job control statements. Job control may be used to define the program that is to be executed, the library that contains the program, the files that are to be processed by the program, and on which physical I/O devices the files reside.

DOS/VSE provides a number of facilities relating to the control of your jobs through the Job Control and Supervisor programs. These facilities include:

- Automatic job-to-job transition, requiring a minimum of operator intervention
- Based on JCL statements, assignment of an actual I/O device to a symbolic device named in the program
- Loading of executable programs from disk libraries into storage for processing
- Handling of program termination.

A DOS/VSE job is identified by certain control statements. It begins with a //JOB statement and ends with a /& statement. Other job control statements may appear between these two. All job control statements are identified by a // in the first two positions of the statement, with these exceptions:

```
/* indicates end of data
/& indicates end of job
* indicates a comment
/+ end of procedure
```

The name of the DOS/VSE job is specified on the //JOB statement. DOS/VSE does not allow the use of step names, although these may be recorded on the JCL statements as comments. Figure 20 illustrates the jobs within a DOS/VSE jobstream.

In DOS/VSE, as on the System/3 Model 15, you can define one or more jobsteps within a job. It may be advantageous to do this when a jobstep is dependent on the successful completion of the previous step. If a step fails, then DOS/VSE flushes the rest of the jobsteps and will allow no further execution of that job. DOS/VSE will then continue with the next job. The job-to-job transition of DOS/VSE is much like that of a System/3 running in NOHALT mode. DOS/VSE does not have a HALT mode of operation.

```

// JOB jobname -----> beginning of first job
.
. (additional job control statements)
.
/& -----> end of first job
// JOB jobname -----> beginning of second job
.
. (additional job control statements)
.
/& -----> end of second job

```

Figure 20. DOS/VSE jobstream

ASSIGNING DEVICES

Job control statements are used to specify the actual I/O devices to be used during a particular execution of a program. When you write a program in DOS/VSE, you specify the type of device needed for a file and not the actual device. This is done by specifying a symbolic name of the form SYSxxx for each file referring to a logical, rather than a physical unit. The assignment of the logical device name to a physical device (channel and unit number) is made by the user either at system initialization or during the actual operation of the system. These can be entered by the operator, but would normally be provided by the programmer in the job control statements of his job. This means that you can still run your programs, perhaps on a different device, if there are hardware failures or for testing on another processor with different devices. DOS/VSE uses certain logical devices that are assigned to a physical device. Figure 21 shows some of the system logical devices and explains what they are used for.

The application programmer has 241 logical devices available. they are SYS000 through SYS240. You should plan to standardize the logical device names that your programmers use for files such as SYS008 for card or diskette files, SYS009 for punch files and SYS010 for printer files. Some IBM programs, especially the utilities, expect to use certain SYSnnn units. When these are specified, in the program documentation, you must use them. The RPG II compiler, for example, makes use of SYS001, SYS002, and SYS003 for its workfiles.

Devices may be assigned using either actual addresses or they may assigned generically. The following examples are assigns using actual addresses:

```

// ASSGN SYS007,00E      (assigns SYS007 to the printer at 00E)
// ASSGN SYS020,240      (assigns SYS020 to the DASD device at 240)

```

The following are generic assigns :

```

// ASSGN SYS007,PRT1      (assigns SYS007 to a PRT1 type printer)
// ASSGN SYS020,FBA,VOL=DATA01,SHR (assigns SYS020 to a FBA type
                                device with a vol ID of DATA01)

```

Generic assigns are recommended because they reduce the amount of changes to JCL when actual devices are changed.

Device	DOS/VSE USE
SYSRDR	Assigned to the device that reads job control statements
SYSIPT	Assigned to the device that reads job data
SYSIN	Used when assigning SYSRDR and SYSIPT to the same physical device
SYSPCH	Assigned to the system punch device
SYSLST	Assigned to the system printer
SYSOUT	Used to assign both SYSPCH and SYSLST to a tape unit
SYSLOG	Assigned to the device that logs job control, normally the system console
SYSLNK	Assigned to the device that has the work area for linking programs to the core image library
SYSSLB	Assigned to the DASD device where the private source library is located
SYSRLB	Assigned to the DASD device where the private relocatable library is located
SYSCLB	Assigned to the DASD device where the private core image library is located
SYSREC	Assigned to the DASD device that records system activity
SYSDMP	Assigned to the DASD device that dumps are written to in case of a program failure
SYSCAT	Assigned to the DASD device where the VSE/VSAM catalog is located

Figure 21. System logical devices

CATALOGED PROCEDURES

DOS/VSE provides a facility to place sets of frequently used job control statements in card image format in the DOS/VSE procedure library. These sets of control statements are then called cataloged procedures. With the execution of a cataloged procedure (initiated with a // EXEC PROC=procname), one statement can replace many other JCL statements. This can greatly reduce the errors caused by mishandling cards or the time that an operator spends replacing bent or damaged cards.

The cataloged procedure facility of DOS/VSE is similar to that provided by the System/3, with some important differences. DOS/VSE does not permit the "nesting" of procedures, where one procedure can request the execution of another procedure. This means that you may have to replan the contents of your procedures, to eliminate those that are nested.

The System/3 provides the capability to modify procedures in the source library by using the \$MAINT program to remove, replace, or insert statements.

DOS/VSE does not allow an entry in the procedure library to be modified. The procedure must be recataloged if permanent changes are to be made. However, DOS/VSE procedures can be temporarily modified during execution

by the inclusion of "overwrite" statements in the jobstream, to add, delete, or replace statements. See the Library Considerations chapter for examples of cataloging and modifying a procedure. As on the System/3, data can be included in a cataloged procedure. The data must be referred to in your program as SYSIPT data. This data might be control statements for the utility programs or the sort. Normally you would not include the data for your own application programs in a cataloged procedure, especially if the data changes frequently. Data in a cataloged procedure cannot be overwritten at execution time.

LABEL INFORMATION AREA

DOS/VSE provides another facility that can reduce the number of job control statements within a program. DOS/VSE lets you store the information about your disk files in a special area on disk called the label information area. This area is a part of the system residence file (SYSRES) and is located after the last library defined. If you do not specify information about your disk file in your job (the // DLBL and // EXTENT statements), then DOS/VSE will search the label information areas for that information. You would use the label information areas to contain labels for system workfiles used by the compilers and linkage editor, other system files - SYSRES, or your own master files. DOS/VSE with VSE/Advanced Functions allows you to define a label area on SYSRES for the labels. When the DOS/VSE system is backed up and restored from tape the size of the label area may be expanded.

To make effective use of the label information areas, you should standardize the names of your files, so that different programs always refer to the same file by the same name.

CONVERSION PROCEDURES FOR OCL

Although the System/3 OCL and DOS/VSE JCL statements perform similar functions, the formats of the two are different. All of your System/3 OCL must be rewritten into the DOS/VSE formats. In some cases the DOS/VSE equivalent statement is almost identical to the System/3, such as the // IATE statement. In other cases, information contained in one System/3 statement will appear in several DOS/VSE statements, such as the // FILE that must be converted to the // DLBL, // EXTENT and // ASSGN statements. In still other cases, there is no DOS/VSE equivalent for the System/3 statement, because the statement is not needed in DOS/VSE, such as the // LOCKOUT of the System/3 Model 10.

Your conversion procedure will consist of several steps. First, collect all the OCL that must be converted, including procedures that are cataloged in your procedure libraries.

Next, develop some standards for your system. Some of the points you should consider are the assignment of certain SYSxxx names to your files, whether you will use physical device addresses or device type designations to refer to the physical devices at execution time (//ASSGN), and which of your disk labels will be placed in the label information area as standard labels. These and other considerations are discussed in the DOS/VSE Entry User's Guide.

Once these decisions are made, you can physically convert your OCL statements to JCL statements. There are several ways you might do this, for example:

- You may write an automated conversion program that reads OCL and produces JCL. Control cards might be used to provide SYSxxx names, device addresses, file label information, etc.
- You can manually recode all of your OCL statements into JCL format.

The following section presents a comparison of OCL and JCL to guide you during conversion.

After you have converted your JCL statements, you must test them to ensure the conversion was correct. You will use your converted JCL statements when you begin to test your application programs.

Once your JCL statements have been tested, you will want to update your documentation, especially the operator's run sheets.

OCL AND JCL COMPARISON

This section presents several figures comparing System/3 OCL with DOS/VSE JCL. Figure 22 shows System/3 OCL statements with their DOS/VSE equivalents. In some cases, the OCL statement may be replaced with a DOS/VSE command used by the operator. Figures 23 and 24 present more detailed information about the formats for supplying disk and tape label information to DOS/VSE. information to DOS/VSE. Figure 25 illustrates the information you must provide to DOS/VSE about your VSE/VSAM files. Further information and detailed formats of the DOS/VSE JCL statements can be found in the DOS/VSE System Control Statements Manual, GC33-5376.

System/3	DOS/VSE	DOS/VSE Functions/Comments/Differences
// LOAD OR	// EXEC progname OR	program name (8 characters) indicates the program to be executed. DOS/VSE retrieves a program from the assigned core image library.
// stepname LOAD	// EXEC PGM=progname	The EXEC also performs the function of System/3 RUN.
// LOAD* OR //stepname LOAD*	no equivalent	In DOS/VSE the program must be link-edited into the core image library before it can be executed. The closest DOS/VSE job stream would be the following: // JOB jobname // OPTION LINK INCLUDE /* // EXEC LNKEDT // EXEC / &
// LOG	LOG command // OPTION LOG // OPTION NOLOG	The device assigned to SYSLOG is used for logging. The DOS/VSE LOG command is equivalent to the System/3 model10 // LOG ON Causes control statements to print on SYSLSST Terminates printing of control statements
// NOHALT	no equivalent	DOS/VSE operates in NOHALT mode
// PAUSE	// PAUSE	Equivalent function. Comments to operator precede the // PAUSE so the operator can take the necessary actions
// PRINTER	// ASSGN SYSLSST set linect=nn	Specifies physical device for system printer Sets the number of lines to be printed on the system printer (SYSLSST)
// PUNCH	// ASSGN SYSPCH	Specifies physical device for system punch
// READER	// ASSGN SYSIN	Specifies the physical device or control statements and SYSIPT data
// RUN	// EXEC	The // EXEC causes the program to be loaded and executed, combining the functions of the System/3 // LOAD and // RUN
// SWITCH	// UPSI	DOS/VSE resets UPSI byte to zeros at beginning of each job
/ &	no equivalent	DOS/VSE has no delimiter for end of job step, / & in DOS/VSE indicates end of job
/*	/ &	Indicates end of job
*	* Comment	Same as System/3, DOS/VSE must have a blank in column 2
/*	/*	Same as System/3, end of a data file

Figure 22.OCL and JCL comparison (part 1 of 2)

System/3	DOS/VSE	DOS/VSE Functions/Comments/Differences
// FORMS (model10)	// ASSGN SYSLST,cuu SET LINECT=nn	See the // PRINTER OCL statement
// PARTITION (model10)	ALLOC F=nnk command	DOS/VSE partitions are defined by the ALLOC command, normally at IPL time
// LOCKOUT (model10)	no equivalent	DOS/VSE partitions have control depending on priority
// BSCA	// ASSGN	No equivalent, a DOS/VSE BSCA device is assigned a SYSnnn number. At execution it is assigned to a physical line address
// CALL	// EXEC PROC= procname	PROC= indicates the name of the procedure to be executed from the procedure library
// COMPILE	ASSGN PHASE	No direct equivalent. DOS/VSE uses the ASSGN for the library to be used and PHASE for the name to be given to the compiled program
// DATE	// DATE mm/dd/yy	The DATE applies to the job currently being executed. Date is reset to system date at the end of the job being executed.
// FILE	For Disk/Diskette // ASSGN // DLBL // EXTENT For Tape // TLBL // ASSGN Other files // ASSGN	Information about the file is contained in up to three statements for Disk and Diskette files. Contains information about file name, file ID retention period and file type. The ASSGN statement associates a logical file with a physical device.
// HALT	no equivalent	DOS/VSE operates in NOHALT mode. A // PAUSE may be used to allow the operator to respond to instructions.
// IMAGE	Use the System Buffer Load	The SYSBUFLD program loads the universal character set buffer. The buffer load data can be in the core image library or read in from SYSIPT. The LUCB command can also be used.
// jobname JOB	// JOB jobname	Jobname is one to eight characters. Indicates the beginning of a job

Figure 22. OCL and JCL comparison (part 2 of 2)

SYSTEM/3		DOS/VSE		Differences/Comments
Statement	Parameter	Statement	Parameter	
// FILE	NAME=filename	// DLBL	filename	System/3 1-8 characters DOS/VSE 1-7 characters
	LABEL=filename		'file-id'	System/3 1-8 characters DOS/VSE 1-8 characters
	Date-mmddyy RETAIN-T RETAIN-S RETAIN-P		date (blank) 0 99/365	DOS/VSE checks the creation date on input files. The date value specifies the file retention period.
			codes	In DOS/VSE indicates type of file being used.
			DSF	DOS/VSE data secured file
		// EXTENT	sysnnn	Symbolic unit is the same as that specified in the DOS/VSE program.
	PACK=name		serial number	One to six character volume identifier of disk.
			type	Specifies type of extent.
			sequence number	Specifies sequence number of the extent.
	LOCATION-location		relative block	Specifies the beginning of the data file relative to zero.
	TRACKS-number RECORDS-number		number of blocks	Specifies size of the file.
	SPLIT-tracks/ cylinders			Not used by fixed block architecture DASD devices
	UNIT=unit	// ASSGN	sysnnn, cuu	In System/3, unit is disk unit number. In DOS/VSE cuu is the physical disk address.
	HIKEY- 'high key'			Not used by DOS/VSE
	VERIFY			Not used by DOS/VSE

Figure 23. Disk label information

System/3		DOS/VSE		Differences/Comments
// FILE	Name-filename	// TLBL	filename	System/3 1-8 characters DOS/VSE 1-7 Characters
	LABEL-name		'file-ID'	System/3 1-8 Characters DOS/VSE 1-7 Characters
	RETAIN-nnn DATE-mmddyy		date	DOS/VSE checks creation date of input files. Date specifies the retention period of output files
	REEL-nnnnn		file serial number	Volume identification of tape
			volume sequence number	Used by DOS/VSE to control the sequence of volumes in a multiple volume file
	SEQNUM-nnn		file sequence number	Number of a file in a multi-file volume
			gen- number	Generation number. In DOS/VSE used to modify the file-ID
			version- number	In DOS/VSE used to modify the generation number
	UNIT-unit	// ASSGN	SYSnnn,cuu	In DOS/VSE SYSnnn is logical file name. The cuu is the physical address of the device
	DENSITY- PARITY- TRANSLATE- CONVERT-		ss	The values specified for ss are used in place of the four System/3 parameters
	ASCII BLKL RECL RECFM DEFER END			Not used in DOS/VSE

Figure 24. Tape label information

DOS/VSE Statement	Operand	VSE/VSAM Specifications
<pre>// DLBL ['file-ID'], [date], [codes], [DSF], [BUFSP=n], [CAT=filename]</pre> <p>note: a comma is inserted for each positional operand that is omitted if subsequent operands are used.</p>	filename	<p>The filename is 1-7 alphameric, the first of which must be alphabetic. It is identical to (1) dname of the FILE(dname) parameter in a Access Method Services command, and (2) the DDNAME=filename parameter of the Access Method Control Block (ACB) in the processing program that identifies the file.</p>
	'file-ID'	<p>The file-ID must be specified when an existing input file is being processed. The file-ID is identical to the name specified in the DEFINE command of Access Method Services. The file-ID must be coded as follows:</p> <ul style="list-style-type: none"> • One to 44 Alphameric(A-Z,0-9,@,\$,#) characters. • After each group of eight or less characters, a period(.) must be inserted. • No imbedded blanks allowed. • The first character of the file-ID and the first character following a period must be alphabetic or notational (A-Z,@,\$,#).
	date	<p>This parameter overrides the expiration date specified in the DEFINE command of Access Method Services.</p>
	DSF	<p>Ignored by VSAM. All files are data secured.</p>
	BUFSP=n	<p>Specifies the number of bytes of virtual storage to be allocated for buffer space for this file. Overrides buffer space allocated when the file was defined or the amount specified in the ACB provided BUFSP=n is a higher value. The accepted value is (0-999999)</p>
	CAT=filename	<p>This specifies the filename of the catalog owning this file. Specify this operand if you want to override the system or job catalog.</p>

Figure 25. VSE/VSAM label information(part 1 of 2)

DOS/VSE Statement	Operand	VSE/VSAM Operands
// EXTENT [symbolic-unit], [serial-number], [type], [sequence-number], [physical block no], [number of blocks]	symbolic unit	A six character field indicating the symbolic unit(SYSnnn) of the volume for which this extent is effective.
	serial number	Volume serial number of the volume. One to six characters.
	type	One character indicating the type of extent. VSAM data area value is a 1.
	sequence number	Not used by VSAM
	physical block number	One to ten characters indicating the start of the file relative to zero. Used by VSAM for UNIQUE files only.
	number of blocks	Specifies the number of blocks in the file. Used by VSAM for UNIQUE files only
// ASSGN sysnnn, [cuu FBA], [temp perm], [VOL=volserno], [SHR]	SYSnnn	Specifies the programmer logical unit. The range of nnn is from 000 to one less than the number specified at system generation.
	cuu FBA	Indicates the channel and unit number. Fixed block architecture DASD. Used when you are interested in the type of device but no particular physical unit.
	VOL= serialno	Specifies the volume serial number of the device required.If VOL is specified the system searches in sequence for the volume label. The operator receives a message to mount the requested volume if it is not online.
	PERM TEMP	Indicates whether the assignment should be temporary or permanent.
	SHR	Applies only to DASD devices. Should be used to allow a DASD device to be assigned that is already assigned to another partition.

Figure 25. VSE/VSAM label information (PART 2 of 2)

CHAPTER 8. UTILITIES

One of the most valuable tools that you have in your data processing environment is the set of utility programs that enable you to easily do many of the system maintenance and file manipulation jobs necessary for a smooth system operation. This chapter will help you to convert your System/3 utilities to equivalent DOS/VSE utilities. Library utilities will be considered in a later chapter.

CONVERSION PROCEDURES

The conversion procedure for creating DOS/VSE utilities consists of the following steps:

- Evaluate utility functions
- Manually recode control cards and JCL
- Test
- Update documentation

EVALUATE UTILITY FUNCTIONS

The first task that you must do in the conversion of your utility programs is to inventory and evaluate each one. If, as part of your whole transition effort, you have changed program functions or file formats, you may no longer need certain utilities and you may have to add new ones. In general, the DOS/VSE utilities contain commands to do the equivalent functions that you now do on your System/3. But the names of the programs will be different, and the control cards will have different parameters. Some of the System/3 utilities have no equivalent in DOS/VSE, because they perform functions unique to the System/3 and not necessary on DOS/VSE, such as the 5445 Data Interchange Utility.

To evaluate your utilities:

- Determine the function being performed
- Decide if the function is still necessary or if another must be added
- Select a way to do it under DOS/VSE.

Detailed information on the functions provided by DOS/VSE utilities can be found below under DOS/VSE Utility Programs.

MANUALLY RECODE CONTROL CARDS

Once you have evaluated each utility function and determined how you want to code it in DOS/VSE, you must manually code the control cards and JCL for the DOS/VSE utilities. Samples of coding for the most common System/3 and DOS/VSE utility functions are given later in this chapter under Utility Program Examples.

TEST

After you have recoded the utility programs, you will want to test them to ensure that they perform the appropriate function. You can use the same data files that you create to test your application programs for this purpose.

UPDATE DOCUMENTATION

When your utility programs are running correctly, you should then update your documentation to reflect added, deleted, or changed utility functions.

DOS/VSE UTILITY PROGRAMS

A set of system utility programs is provided with DOS/VSE. Two programs products are also available to perform utility functions, VSE/VSAM ACCESS METHOD SERVICES (available with VSE/VSAM, program number 5746-AM2) and VSE/DITTO.

SYSTEM UTILITIES

The DOS/VSE System Utilities is a set of programs that performs a variety of general system functions, including the following:

- Assigns an alternate block on disk when a block has been proven defective
- Clears one or more areas of a disk
- Copies an entire file or volume onto another specified medium and restores the file or volume to its original medium (copy/restore diskette, copy/restore disk to card, copy/restore disk to tape, copy disk to disk, fast-copy disk volume)
- Initializes disk
- Initialize a tape with IBM and ANSI standard volume labels
- Displays the volume table of contents (VTOC)
- Prints the hard-copy file
- Blocks, deblocks, and copies a file; prints job control statements and file contents; selects jobs from a blocked file
- Updates object programs
- Saves the DOS/VSE system and private libraries on tape and restores these libraries to a disk

Detailed information on these utilities can be found in the IOS/VSE System Utilities Manual, GC 33-5381.

VSE/VSAM ACCESS METHOD SERVICES

The program product VSE/VSAM provides a general purpose utility called Access Method Services, which allows you to perform useful functions for creating and maintaining your VSE/VSAM files. Commands are available to do the following functions:

- Define and copy VSE/VSAM catalogs
- Allocate space for a file
- Define (create), print, copy, or reorganize VSE/VSAM files
- Load records into a file
- Create a backup copy of a file
- Alter, delete, or list catalog entries
- Convert a sequential or indexed-sequential file to the VSE/VSAM format

- Build one or more alternate indexes for a file
- Define catalog entries for paths and non-VSE/VSAM files
- Recover from certain types of damage to a file
- Control command execution by testing or setting condition codes
- Specify diagnostic aids and printed output options.

Detailed information about VSE/VSAM Access Method Services can be found in the manual Using VSE/VSAM Commands and Macros SC24-5144.

VSE/DATA INTERFILE TRANSFER, TESTING AND OPERATIONS UTILITY

The licensed program VSE/DITTO (program number 5746-UT3) provides a series of commands that perform useful functions for card, tape, disk, and printer files. Commands are available to let you examine, create, and modify files in a testing and production environment. VSE/DITTO performs many of the normal card, tape, and disk utility functions in one program, which are performed by one or more standard utility programs. You may currently have a DITTO program on your System/3. The VSE/DITTO program contains many of the same functions, but they are related to a 4300 environment.

Some of the functions are as follows:

- Card/tape/disk to card/tape/disk/print
- SAM, VSE/VSAM file creation
- Initialize tape
- Change disk volume label
- Alter bad data on tape or disk
- Tape positioning
- Scan tape/disk

COMPARISON OF UTILITY FUNCTIONS

There is not one-to-one correspondence between System/3 and DOS/VSE utilities. Services performed by one System/3 program may be performed by two or more DOS/VSE programs, and one DOS/VSE program may perform the function of two or more System/3 programs.

Figure 26 presents a chart of the most common utility functions you now perform, naming the System/3 and corresponding DOS/VSE program that provide the function.

Utility Functions	System/3	DOS/VSE
Initialize a disk volume	\$INIT	INTDK
Rename a disk volume	\$INIT	VSE/DITTO (DID)
Print the hard-copy file	\$HIST	PRINTLOG
Initialize a tape volume	\$TINIT	INTTP
		VSE/DITTO (INT)
Display tape labels	\$TINIT	VSE/DITTO (TP)
Alternate Block assign	\$ALT	ALTBK
Alternate Block rebuild	\$BUILD	ALTBK
Disk volume VTOC display	\$LABEL	LVTOC
		IDCAMS (LIST CAT)
Disk file label display	\$LABEL	IDCAMS (LIST ENT)
Erase disk data	\$DELET	OBJMAINT
File delete	\$DELET	IDCAMS (DELETE)
Copy a disk volume	\$COPY	FCOPYB
Copy a disk file	\$CCPY	OBJMAINT, IDCAMS (REPRO)
Dump disk to tape	\$DCOPY	FCOPYB
Restore tape to disk	\$DCOPY	FCOPYB
Tape error program	\$TVES	EREP
Recover Index program	\$RINDX	IDCAMS
Display label area	N/A	LSERV
3340 simulation area copy	\$SCOPY	N/A
1000 file VTOC conversion	\$WVTOC	N/A
5445 data interchange utility	\$VTOC	N/A
File compress	\$FCOMP	N/A

Note: IDCAMS is a program that is available with the program product VSE/VSAM (program number 5746-AM2). VSE/DITTC is a program product (program number 5746-UT3). The other are programs are DOS/VSE system utilities.

Figure 26. Comparison of utility functions.

UTILITY PROGRAM EXAMPLES

The following pages contain a more detailed discussion of various System/3 utilities and their DOS/VSE equivalents. The IDCAMS utilities are available with VSE/VSAM. Coding examples are provided for some of the most common functions to help you convert your utility programs. Where two or more DOS/VSE programs can perform the same function, alternate methods will be suggested. Examples are based on the System/3 and 4300 Processor configurations given in Appendix A. Each DOS/VSE utility program generally requires some job control statements (JCL) to name the program to be executed and to assign any required I/O devices. Each utility will use certain symbolic device assignments (SYSnnn). You may find it helpful to review the general discussion of JCL in Chapter 6.

DISK INITIALIZATION

In DOS/VSE, as on the System/3, all disks must be initialized before they are used. The IBM fixed block architecture DASD modules are preinitialized by IBM.

The disk initialization that you perform (called quick initialization) will create the volume table of contents (VTOC) and write a label on the volume to identify the disk, optionally you can clear the disk area to zeros. You can also rename a disk volume that has been previously initialized. Figure 27 shows two examples of disk initialization using the DOS/VSE utility INTDK.

In order to perform a surface analysis of a fixed block architecture DASD you may use the stand alone utility provided with the DOS/VSE system. This program would be run if the number of alternate blocks assigned exceeds the amount available. The utility does a surface analysis of the device and reclaims blocks if possible. The utility identifies the device and volume to the operator, checks for defined VSAM space and also checks for protected or unexpired files. After the surface analysis program is run the disk initialization program should be executed.

TAPE INITIALIZATION

In DOS/VSE all tapes must be initialized before use. Initialization writes a volume label on the tape for a labeled tape or a tape mark for an unlabeled tape. The tape initialization functions can be performed in DOS/VSE by using the DOS/VSE tape initialization program or VSE/DITTC. The volume display function can be accomplished by using a combination of several VSE/DITTO commands. Figure 28 shows an example of tape initialization.

FIRST INITIALIZATION

```
(1)
// LOAD $INIT,F1                // JOB INIT DISK
// RUN                          // ASSGN SYS000,241 (2)
// UIN UNIT-D2,TYPE-PRIMARY    // EXEC INTDK (1)
// VOL PACK-222222            // UID IQ
// END                          // VTOC STRTADR=126000 (4)
                                // (3)
                                VOL1222222
                                // END
                                /&
```

REINITIALIZATION

```
(1)
// LOAD $INIT,F1                // JOB INIT DISK
// RUN                          // ASSGN SYS000,241 (2)
// UIN UNIT-D2,TYPE-CLEAR      // EXEC INTDK (1)
// VOL PACK-222222,OLDPACK-222222 // UID IS
// END                          // VTOC STRTADR=126000 (4)
                                // (3)
                                VOL1222222
                                // END
                                /&
```

- (1) Program to be executed
- (2) Unit-address of the disk to be initialized
- (3) New volume label
- (4) VTOC location
- (5) The DOS/VSE operator must confirm initialization if there are any unexpired files on the disk

Figure 27. Disk initialization

HARD-COPY FILE PRINT

In DOS/VSE the same kind of information that is in the System/3 Model 15 system history area (SHA) is in a file called the hard-copy file. You can print the entire contents of the DOS/VSE hard-copy file, entries not previously printed, or other selected groups of messages with a program called PRINTLOG. In DOS/VSE the printing options are entered through the operator console. Normally the system operator will see a list of all the available options, and he can select the one(s) he wants. Figure 29 shows examples of printing the hard-copy file.

```

(1)
// LOAD $INIT,F1 // JOB INIT TAPE(VSE/DITTO)
// FUN // ASSGN SYS000,330,C0 (2) (4)
(2) (3) (4) // VOL UNIT-T1,REEL-TAPE01,DENSITY-1600 // ASSGN SYS001,UA
// END // EXEC INTTP (1)
// INTT CARD
(3)
VOL1TAPE01
// END
/&
// JOB INIT TAPE (VSE/DITTO)
(1)
// EXEC DITTO
* The NEXT 4 LINES ARE ENTERED
THROUGH THE OPERATOR CONSOLE
(5)
INT
(2) (4)
330C0
(3)
TAPE01
EOJ
/&
(1) Program to be executed
(2) Unit-address of the tape to be initialized
(3) New volume label
(4) Density of the tape
(5) VSE/DITTO function to use

```

Figure 28. Tape initialization

VTOC AND FILE LABEL DISPLAY

In DOS/VSE the VTOC Display program displays all the file labels contained in the VTOC. There is no option to list a particular data file only, and the VTOC entries cannot be printed in a sorted order. The program used to display the VTOC is named LVTOC. Figure 30 shows how to list the VTOC.

```

(1)
// LOAD $HIST,F1                // JOB PRINT HARDCOPY FILE
// RUN                          // EXEC PRINTLOG
(2)                              (2)
// PRINT HISTORY-ALL           * ENTERED BY OPERATOR: ALL
// END                          /&
(1)
// LOAD $HIST,F1                // JOB PRINT HARDCOPY FILE
// RUN                          // EXEC PRINTLOG
(3)                              (3)
// PRINT HISTORY-CURRENT       * ENTERED BY OPERATOR: NEW
// END                          /&
(1) Program to be executed
(2) Print entire contents of the file
(3) Print entries not previously printed

```

Figure 29. Hard-copy file print

```

(1)
// LOAD $LABEL,F1              // JOB DISPLAY VTOC
// RUN                          // ASSGN SYS004,240
(2)                              (2)
// DISPLAY UNIT-D2,LABEL-VTOC // ASSGN SYS005,00E
// END                          // EXEC LVTOC
(3)                              (1)
// END                          /&
(1) Program to be executed
(2) Unit-address of the disk containing the VTOC to be
    displayed
(3) Unit-address of output medium (printer, tape, or disk)

```

Figure 30. VTOC display

SINGLE FILE DISPLAY

If you have converted your System/3 files to VSE/VSAM, you can use the VSE/VSAM Access Method Services program, IDCAMS, to list all entries in the VSE/VSAM catalog or only the entry for a particular file. Figure 31 shows an example of printing the information about one file.

```

(1)
// LOAD $LABEL,F1                // JOB LIST VSE/VSAM FILE KSDS1
// RUN                            (1)
//                               // EXEC IDCAMS,SIZE=AUTC
//                               (3)
// DISPLAY UNIT-D2,LABEL-KSDS1    LIST ENT(KSDS1) ALL
// END                            /*
//                               /&

```

- (1) Program to be executed
- (2) Unit-address of the disk containing the file to be displayed ** In DOS/VSE the information about the file is kept in the VSE/VSAM catalog
- (3) Name of the file

Figure 31. Display information about a single file

ALTERNATE BLOCK ASSIGN AND REBUILD

In DOS/VSE, a program is used to assign an alternate block on disk, and to rebuild data on the new block. This program is called ALTBLK. To run the DOS/VSE utility, you must know the Physical Block Number of the defective block. Figure 32 shows an example of assigning an alternate block. Figure 33 shows an example of rebuilding data.

DUMPING AND RESTORING A DISK VOLUME TO TAPE

DOS/VSE provides a utility program to dump a whole disk volume to tape for backup purposes, and then later to restore this tape to disk. The program is called fast copy, FCOPYB, and performs the same kind of functions as \$DCOPY on the System/3.

The FCOPYB program can run under the control of DOS/VSE or you can create a special version of FCOPYB to be loaded standalone from card or diskette. See the chapter on Fast-Copy Disk Volume in the DOS/VSE System Utilities Manual. Figure 34 illustrates the dump of a disk volume to tape and the restore of tape to disk.

```

(1)
// LOAD $ALT,F1 // JOB ASSIGN ALTERNATE BLOCK
// RUN // ASSIGN SYS000,241 (2)
// ALT PACK-222222,UNIT-D2,ASSIGN-1209 // EXEC ALTBLK (1)
// END (3) (4)
PBN=550 VOLID=222222 LIST

/6

(1) Program to be executed
(2) Unit-address of the disk
(3) Location of the defective block (track on System/3)
(4) Lists the defective block on SYSLST

```

Figure 32. Assign alternate block

```

(1)
// LOAD $BUILD,F1 // JOB REBUILD DATA
// RUN // ASSIGN SYS000,241 (2)
// REBUILD PACK-222222,UNIT-D2, // EXEC ALTBLK (1)
(3) (3) (3) (3) (4)
// TRACK-120900,LENGTH-20,DISP-0 PBN=550 VOLID=222222 UPDATE LIST

-SUBSTITUTE DATA-
-1st 32 bytes of data-
-2nd32 bytes of data-
-
-last 32 bytes of data-

/*

/6

(1) Program to be executed
(2) Unit-address of the disk
(3) Physical Block Number (Track Number on System/3)
(4) Lists disk block data on the device assigned to SYSLST

```

Figure 33. Rebuild data

```

(1)
// LOAD $DCOPY,F1
// FILE NAME-BACKUP,UNIT-T1 (2)
// LABEL-'BACKUP TAPE' (3)
// RUN
// COPYPACK FROM-D2,PACK-222222 (4) (5)
// END

(1)
// LOAD $DCOPY,F1
// FILE NAME-BACKUP,UNIT-T1,REEL-TAPE01
// LABEL-'BACKUP TAPE' (2)
// RUN
// COPYPACK TO-D2 (4)
// END

(1)
// JOB DUMP
// ASSGN SYS004,241 (4)
// ASSGN SYS005,330 (2)
// TLBL UOUT,'BACKUP.TAPE' (3)
// EXEC FCOPIB (1)
DUMP VOLUME IV=222222
/*
/&

(1)
// JOB RESTORE
// ASSGN SYS004,330 (2)
// ASSGN SYS005,241 (4)
// TLBL UIN,'BACKUP.TAPE' (3)
// EXEC FCOPIB (1)
RESTORE VOLUME FROM TAPE -
OV=WCRK01 NOVERIFY
/*
/&

(1) Program to be executed
(2) Unit-address of the tape
(3) Filename on tape
(4) Unit-address of the disk
(5) Volume label of the disk

```

Figure 34. Dump and restore a disk volume to tape

FILE DELETE

On the System/3, data files are deleted by using a program called \$DELETE. In DOS/VSE there are several methods to use, depending on the type of file to be deleted. If you have VSE/VSAM files, then you can use the Access Method Services command DELETE. If you have SAM files defined in the VSE/VSAM catalog, you can also use the Access Method Services DELETE command to delete these files from the VSE/VSAM catalog and scratch their entries from the VTOC. If you are not using VSE/VSAM, you can delete a file writing a new file over it. If the old file(s) is unexpired, DOS/VSE will give the operator a message asking if the file should be deleted or the job canceled. To delete the file, the operator can key DELETE. To delete the entire contents of the VTOC, you can define a new file with extents as large as the capacity of the disk, or you can use the IS function of the Disk Initialization program. Figure 35 shows an example of deleting individual files. Figure 36 shows an

example of deleting all files. In the case of DOS/VSE the files are removed by deleting the volume table of contents (VTCC).

```

(1)
// LOAD $DELET,F1                // JOB DELETE VSE/VSAM FILE
// RUN                            // ASSGN SYS007,FBA,VOL=DATA01,SHR
// REMOVE PACK-222222,UNIT-D2,    // DLBL VSAMFIL,,,VSAM
// LABEL-FILE01                  // EXTENT SYS007
// END                            // EXEC IDCAMS,SIZE=AUTO

DELETE-
(3)
FILE01 CLUSTER FILE(VSAMFIL) PURGE

/*
/ε

// JOB DELETE SEQUENTIAL FILE
(4)
// ASSGN SYS004,SYSIPT
(2)
// ASSGN SYS005,241
(5)
// DLBL UOUT,'DUMMY FILE',0
(6)
// EXTENT SYS005,,1,0,102000,1000
(1)
// EXEC OBJMAINT

./ COPY

** DUMMY RECORD

/*
/ε

(1) Program to be executed
(2) Volume to be deleted
(3) The name of the file to be deleted
(4) Indicates SYSIPT will be input device
(5) A dummy file name with a retention period of zero days
(6) Extent to be overwritten

```

Figure 35. File deletion

```

(1)
// LOAD $DELET,F1 // JOB DELETE ALL FILES
// RUN // ASSGN SYS000,240 (3)
// REMOVE PACK-222222,UNIT-D2,LABEL-VTOC // EXEC INTDK (1)
// END // UID IS
// VTOC STRTADR=126000
(2)
VOL1222222
// END
/ &

```

- 11) Program to be executed
- (2) Volume label of the disk
- (3) Unit-address of the disk

Note: The Method will work if all the files are sequential. If there are VSE/VSAM files they must be deleted with the IDCAMS program.

Figure 36. Delete contents of VTOC

COPY A DISK VOLUME TO ANOTHER DISK VOLUME

If you do not have tape on your system, you may want to copy a disk volume to another disk volume for backup. On the System/3 you would use \$COPY; on DOS/VSE you would use the FCOPYB program. Figure 37 shows an example of copying disk to disk.

```

(1)
// LOAD $COPY,F1 // JOB COPY DISK TO DISK
// RUN // ASSGN SYS004,FBA,VOL=SYSRES,SHR (2)
// COPYPACK FROM-D1,TO-D2, // ASSGN SYS005,FBA,VOL=222222,SHR (3)
(2) (3) // EXEC FCOPYB,REAL (1)
(4) (5) COPY VOLUME IV=SYSRES OV=222222 NV=222222 (4) (5) (6)
// END /*
/ &

```

- (1) Program to be executed
 - (2) Volume to be copied from
 - (3) Volume to be copied to
 - (4) Volume label of the input disk
 - (5) Volume label of the output disk
 - (6) Volume label of the output disk on completion of the copy operation
- If you leave this parameter out, the volume label of the output disk will be the same as the volume label of the input disk.

Figure 37. Copy disk to disk

Another DOS/VSE program, BACKUP, can be used if at least one tape drive is available. BACKUP offers you the facility of a backup on tape of your DOS/VSE system and/or private libraries. Only the active entries will be put on tape. You can restore the tape created by the BACKUP program with the RESTORE program. You can use the same or different allocations for the libraries. Since only the active entries were put on tape, the restored libraries will be condensed. Figure 38 illustrates BACKUP and RESTORE.

```

                                BACKUP OF SYSTEM LIBRARIES

// JOB BACKUP
// ASSGN SYS005,FBA,VCL=SYSRES,SHR   sysres file
// ASSGN SYS006,8809                 backup tape
// ASSGN SYS007,UA                   no private core image
// ASSGN SYS008,UA                   no private relocatable
// ASSGN sys009,UA                   no private source
// DLBL IJSYSRS,'DCS.SYSRES.FILE'
// EXTENT SYS005,SYSRES
// EXEC BACKUP,SIZE=100K
// SA,SORT                           stand alone,sorted entries
/*
// MTC REW,SYS006                     rewind tape
/6

                                RESTORE OF SYSTEM LIBRARIES

The standalone tape may be restored to the SYSRES volume by the
console operator as follows:
1) Key the address of the backup tape unit into the program
load screen
2) ENTER key
3) Respond to the console prompts and select the restore program
4) The RESTORE program will give the following prompts
SPECIFY ADDRESS OF INPUT DEVICE CUU
330      (operator response)
SPECIFY TYPE OF INPUT DEVICE XXXXY
8809     (operator response)
SPECIFY ADDRESS OF SYSRES DISK CUU OR EOB
240     (operator response)
SPECIFY TYPE OF DISK XXXXY
fba     (operator response)
ANY PRIVATE LIBRARIES TO BE RESTORED? YES/NO
no      (operator response)
8R43D TYPE NOVERIFICATION OR PRESS ENTER FOR WRITE VERIFICATION
noverify (operator response)
8R01D ** GIVE SYSTEM LIBRARY ALLOCATIONS **
ENTER key (operator response for each allocation if accepting
the allocations stored on the backup tape)
8R15D TYPE GO IF ALLOCATION IS CCRRECT
go      (operator response)

The final message to appear on the screen will indicate that the
restore process is complete.

```

Figure 38. Backup and restore system libraries

FILE-TO-FILE UTILITIES

You often need to copy files or parts of files into another file of the same or different type for backup or testing. On the System/3 you used the \$COPY or \$KCOPY program for this. In DOS/VSE several programs can perform these functions - VSE/VSAM Access Method Services REPRO, VSE/DITTO, DOS/VSE System Utilities.

VSE/VSAM Access Method Services will allow you to copy a VSE/VSAM file into another VSE/VSAM file, a sequential file into a VSE/VSAM file, optionally specifying a key range or address range or count of records to be copied. The DOS/VSE Copy Disk Utility will allow you to copy a disk file (SAM) from one volume to another volume, with the same extent limits from which it was copied. You may also copy disk to tape or cards, to restore the file later.

For diskette files, the Copy and Restore Diskette program will allow you to recreate labels, copy a diskette to another, create a backup file, and eliminate special records. VSE/DITTO provides many file-to-file functions for card, tape, printer, and disk SAM and VSE/VSAM files.

This chapter presents a comparison of DOS/VSE and System/3 library contents and organization, use, and library management. The library utility programs in DOS/VSE and System/3 are covered.

LIBRARY CONTENTS

In DOS/VSE, as in System/3, libraries are special areas on disk used for storing programs, routines, and sets of control statements. The System/3 has a source library and an object library. The object library is used to store object programs and routines. The source library is used to store procedures and source statements.

DOS/VSE has four libraries, whose functions correspond to the four used by the System/3 libraries (see Figure 39).

System/3	Contents of System/3 and DOS/VSE	DOS/VSE
Object Library	Executable Programs	Core Image Library
-----	Routines	Relocatable Library
Source Library	Source Statements	Source Statement Library
-----	Procedures	Procedure Library

Figure 39. System/3 and DOS/VSE libraries

The core image library contains entries called phases, which are programs or subroutines that can be loaded by DOS/VSE for execution. Phases are like the System/3 object programs.

The relocatable library contains entries called modules, which are programs or routines that need to be link-edited before they can be loaded for execution. Modules are like the System/3 routines.

The source statement library contains entries called books, which are sets of source language statements. Books are like sets of System/3 source statements. The DOS/VSE source statement library is divided into sublibraries. Each sublibrary is identified by a character and is used by a particular program. COBOL, for example, uses the C sublibrary.

The procedure library contains sets of job control statements, and may contain data such as utility modifier or librarian control statements, like the System/3.

ORGANIZATION OF LIBRARIES ON DISK

There are two types of DOS/VSE libraries, system libraries and private libraries. The core image, relocatable, and source statement libraries may exist as both system and private libraries, but the procedure library may exist only as a system library. The system libraries, by definition, are contained in the system residence file (called SYSRES), and can be accessed by all partitions. The system residence file contains the basic operating system and all the supervisor code, in

addition to your application programs. Private libraries are contained on separate disk drives, and can be accessed only by programs in the partitions to which the libraries are assigned. Private libraries would probably contain your application programs and routines. You can think of the system libraries as being on the DASD device from which you IPL (such as F1 on the System/3) and private libraries as being on other drives (such as R1, F2, or R2 on the System/3). See Figure 40 for a representation of the location of system and private libraries.

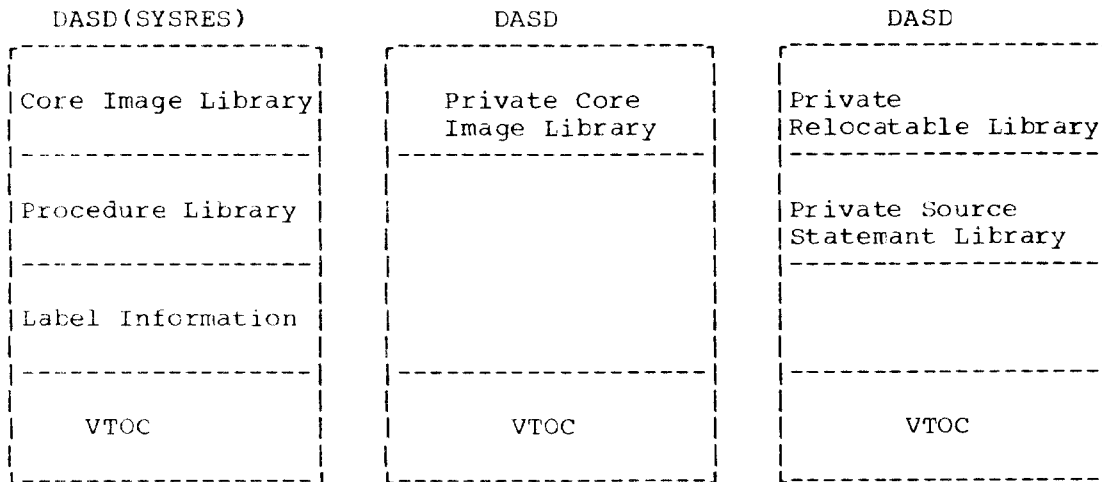


Figure 40. Sample locations of private libraries.

In DOS/VSE the private libraries may be located anywhere on disk, but the system libraries must start at the beginning of a disk, after certain system information. The location of system libraries in relation to each other is always the same, even if one or more do not exist (see Figure 41).

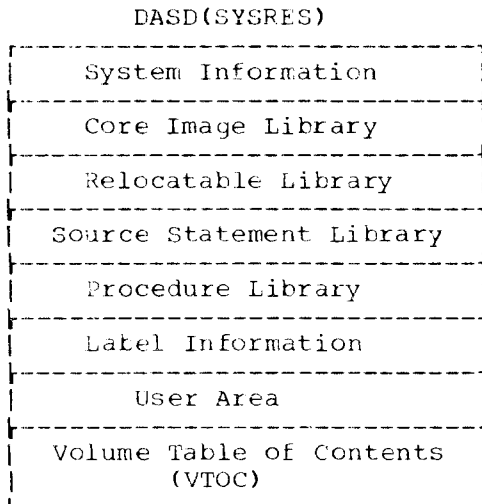


Figure 41. Relative location of the four system libraries.

The boundaries of all the DOS/VSE libraries are fixed by your specifications when you build the system. The size of your system libraries can be changed by using the reallocation function of the MAINI program. Private libraries must be recreated to change their size.

If you have tape drives installed, DOS/VSE provides two very helpful utilities that you can use. The BACKUP program allows you to create a magnetic tape backup of your DOS/VSE system and private libraries. It is normally used in conjunction with the RESTORE utility, which uses the BACKUP-produced tape and reloads the condensed libraries. The new libraries can be of the same or different sizes as those of the original libraries.

LIBRARY MANAGEMENT

Unlike the System/3, DOS/VSE does not allow temporary entries in the libraries, with the exception of the core image library. The relocatable, source statement, and procedure libraries contain only permanent entries.

Before any program can be executed in DOS/VSE, it must be stored in the core image library by the linkage editor. This process is referred to as cataloging a program. Programs are stored either temporarily or permanently, depending on the OPTION specified at link-edit time.

- If the LINK option is specified, the program is stored temporarily for immediate execution in the same job, and it will be overwritten by the next program that is link-edited.
- If the CATAL option is specified, the program is stored permanently and can be executed any time after the catalog job. It can be deleted only by the library maintenance program, or by cataloging another program with the same name.

When preparing to load a program for execution, DOS/VSE will search for programs in the core image library in a special way. Normally DOS/VSE first searches for a program in the private core image library (if assigned), and then in the system core image library. For programs beginning with a dollar sign (\$), usually IBM-provided programs, DOS/VSE reverses the search order.

DOS/VSE will not attempt to reuse space in any of the libraries. Once an entry has been deleted, its space is unused. You can reclaim this space by condensing or reorganizing the library.

In DOS/VSE, as in the System/3, each library has its own directory, containing information about each library entry.

LIBRARY MAINTENANCE PROGRAMS

The System/3 has one program, the Library Maintenance program (\$MAINT), to create, maintain, and service the System/3 libraries. In DOS/VSE there is a set of programs, together called the Librarian, to perform the same kinds of services for the DOS/VSE libraries.

Figure 42 shows the functions that can be performed for the DOS/VSE libraries (functions that you now do on your System/3 and which DOS/VSE program and specific control card should be used. You can find additional information about library services in the DOS/VSE System Control Statements Manual (GC33-5376) in the Librarian chapter.

Most of these functions can be performed on both system libraries and private libraries. To perform a function on a private library, it must be assigned. To perform a function on a system library, the corresponding private library must be unassigned.

Both the DOS/VSE system and private libraries can be condensed and/or changed in size by using the BACKUP/RESTORE utilities.

	Core Image Library	Relocatable Library	Source Statement Library	Procedure Library
Allocate Function				
Create libraries	CORGZ	CORGZ	CORGZ	CORGZ
Change the sizes of libraries	MAINT (ALLOC)	MAINT (ALLOC)	MAINT (ALLOC)	MAINT (ALLOC)
Delete libraries	NOTE	MAINT (ALLOC)	MAINT (ALLOC)	MAINT (ALLOC)
Reorganize libraries	MAINT (CONDS)	MAINT (CONDS)	MAINT (CONDS)	MAINT (CONDS)
Copy Function				
Add or replace one or more library entries	LINKAGE EDITOR	MAINT (CATALR)	MAINT (CATALS)	MAINT (CATALP)
Copy one library entry	CORGZ (COPYC)	CORGZ (COPYR)	CORGZ (COPYS)	CORGZ (COPYP)
Copy library entries that have names beginning with certain characters	CORGZ (COPYC)	CORGZ (COPYR)	N/A	N/A
Copy all library entries	CORGZ (COPYC)	CORGZ (COPYR)	CORGZ (COPYS)	CORGZ (COPYP)
Print one library entry	CSERV (DSPLY)	RSERV (DSPLY)	SSERV (DSPLY)	PSERV (DSPLY)
Print library entries that have names beginning with certain characters	CSERV (DSPLY)	RSERV (DSPLY)	N/A	N/A
Print all library entries	CSERV (DSPLY)	RSERV (DSPLY)	SSERV (DSPLY)	PSERV (DSPLY)
Print entries from all directories including the system directory	DSERV (DSPLY)	DSERV (DSPLY(S))	DSERV (DSPLY(S))	DSERV (DSPLY(S))
Print system directory only	DSERV	DSERV	DSERV	DSERV
Punch one library entry	CSERV (PUNCH)	RSERV (PUNCH)	SSERV (PUNCH)	PSERV (PUNCH)
Punch library entries that have names beginning with certain characters	CSERV (PUNCH)	RSERV (PUNCH)	N/A	N/A
Punch all library entries	CSERV (PUNCH)	RSERV (PUNCH)	SSERV (PUNCH)	PSERV (PUNCH)
Print and punch one library entry	CSERV (DSPCH)	RSERV (DSPCH)	SSERV (DSPCH)	PSERV (DSPCH)
Print and punch library entries that have names beginning with certain characters	CSERV (DSPCH)	RSERV (DSPCH)	N/A	N/A
Print and punch all library entries	CSERV (DSPCH)	RSERV (DSPCH)	SSERV (DSPCH)	PSERV (DSPCH)
Delete Function				
Delete an entry from a library	MAINT (DELETC)	MAINT (DELETR)	MAINT (DELETS)	MAINT (DELETP)
Delete library entries that have names beginning with certain characters	MAINT (DELETC)	MAINT (DELETR)	N/A	N/A
Delete all library entries	MAINT (DELETC)	MAINT (DELETR)	MAINT (DELETS)	MAINT (DELETP)
Modify Function				
This function is used for maintenance of source statements by using card input	N/A	N/A	MAINT (UPDATE)	N/A
Resequence a source library entry	N/A	N/A	MAINT (UPDATE)	N/A
List the statements in a source library entry that are being modified	N/A	N/A	MAINT	N/A
Remove statements from a source library entry	N/A	N/A	MAINT (DEL)	N/A
Replace source library statements	N/A	N/A	MAINT (REP)	N/A
Insert statements into a source library entry	N/A	N/A	MAINT (ADD)	N/A
Rename Function				
Change the name of a library entry	MAINT (RENAMC)	MAINT (RENAMR)	MAINT (RENAMS)	MAINT (RENAMP)
Merge Function				
Create control cards to merge the contents of two libraries	CORGZ	CORGZ	CORGZ	CORGZ
NOTE: MAY BE USED FOR PRIVATE CORE IMAGE LIBRARY ONLY				

Figure 42. Library Functions

Figures 43-50 contain examples of some of the more common library functions that you will be performing on your DOS/VSE libraries, to help you convert your current System/3 library utility programs. Examples are presented for both the System/3 and DOS/VSE. The specific DOS/VSE library is usually indicated by a single letter.

C is the core image library
 R is the relocatable library
 S is the source statement library
 P is the procedure library

```

                (1)
// LOAD $MAINT,F1                // JOB CATAL SOURCE PROGRAM
// RUN                            // EXEC MAINT
// COPY FROM-READER,(2) (3)      // EXEC MAINT
// COPY FROM-READER,LIBRARY-S,NAME-Prog01, CATALS A.PROG01
// TO-F1,RETAIN-PBKEND
** Source Statements **          ** SOURCE STATEMENTS **
// CEND                            BKEND
// END                            /*
                                   /*
(1) Program to be executed
(2) The last character indicates the DOS/VSE library
    ** You can catalog in the core image library
    only by link-editing the program
(3) Name of the source book
(4) Sublibrary identifier

```

Figure 43. Catalog to a library

```

(1)
// LOAD $MAINT,F1 // JOB PRINT SOURCE PRCG
// RUN // EXEC SSERV
(2) (3) (4) (5) (3)
// COPY FROM-F1,LIBRARY-S,NAME-PROG01, DSPLY A.PROG01
(4)
// TC-PRINT /*
// END /&

```

- (1) Program to be executed
- (2) The first character in the DOS/VSE program name indicates the library
- (3) Name of the source book
- (4) Requested function
- (5) Sublibrary identifier

Figure 44. Print a library entry

```

(1)
// LOAD $MAINT,F1 // JOB PRINT ALL
// RUN // EXEC DSERV
(2) (3) (2)
// COPY FROM-F1,LIBRARY-ALL,NAM-DIR, DSPLYS CD,RD,SD,PD
(3)
// TO-PRINT /*
// END /&

```

- (1) Program to be executed
- (2) Indicates all libraries
- (3) Requested function
Output of the DOS/VSE function DSPLYS is sorted

Figure 45. Print all directories

```

          (1)
// LOAD $MAINT,F1                // JOB PRINT DIRECTORY
// RUN                            // EXEC DSERV
// COPY FROM-F1,LIBRARY-SYSTEM,NAME-DIR, /&
// TO-PRINT
// END

(1) Program to be executed
(2) The absence of control cards in DOS/VSE
    indicates that only the system directory
    is to be printed.

```

Figure 46. Print the system directory

```

          (1)
// LOAD $MAINT,F1                // JOB PRINT AND PUNCH
// RUN                            // EXEC PSERV
// COPY FROM-F1,LIBRARY-P,NAME-PRCC01, DSPCH PROC01
// TC-PRTPCH                      /*
// END                            /&

(1) Program to be executed
(2) The first character in the DOS/VSE program
    name indicates the library
(3) Procedure name
(4) Requested function

```

Figure 47. Print and punch a library entry

```

      (1)
// LOAD $MAINT,F1                               // JOB DELETE CIL PHASE
// RUN                                           // EXEC MAINT
      (2)      (3)      (2)      (3)
// DELETE FROM-F1,LIBRARY-O,NAME-RUN01,        DELETC RUN01
// RETAIN-P                                     /*
// END                                           /&

```

- (1) Program to be executed
- (2) The last character indicates the library
- (3) Name of the load module

Figure 48. Delete a library entry

```

      (1)
// LOAD $MAINT,F1                               // JOB MODIFY A SL BOOK
// RUN                                           // EXEC MAINT
      (2)      (3)      (3)      (4)      (2)
// MODIFY NAME-PROG01,FROM-F1,LIBRARY-S,        UPDATE  A.PROG01
      (5)      (5)
// WORK-D2,RESER-YES                            ) DEL SEQN01,SEQN02
      (5)      (5)
// REMOVE FROM-SEQN01,TO-SEQN02                ) REP SEQN03,SEQN04
      (5)      (5)
// REPLACE FROM-SEQN03,TO-SEQN04                ** REPLACE STATEMENTS **
** REPLACE STATEMENT(S) **                      ) ADD SEQN05
      (5)
// INSERT AFTER-SEQN05                          ** INSERT STATEMENTS **
** INSERT STATEMENT(S) **                       ) END
// CEND                                         /*
// END                                           /&

```

- (1) Program to be executed
- (2) Name of the source library book
- (3) Update can be performed only on source library books in DOS/VSE
- (4) Identifier for Assembler sub-library
- (5) Sequence numbers that correspond to source statements of a program in the source library

Figure 49. Modify a source library entry

```

(1)
// LOAD $MAINT,F1 // JCB RENAME RI MOD
// RUN // EXEC MAINT
// RENAME FROM-F1,LIBRARY-R,NAME-ROUT01, (2) (3) (4)
// NEWNAME-ROUT02 RENAMR ROUT01,ROUT02
// END /*
// &

(1) Program to be executed
(2) The last character indicates the library
(3) Oldname of the routine/module
(4) Newname of the routine/module

```

Figure 50. Rename a library entry

LIBRARY CONVERSION CONSIDERATIONS

When planning for your upgrade from System/3 to DOS/VSE you should decide what types of libraries you will use, how many of each type you will have, and where they will be located. Most of the considerations that you reviewed in planning your System/3 libraries are still valid for DOS/VSE. The libraries should be small enough not to waste disk space, but large enough to allow for some expansion of the system. You also want your libraries to be accessed with maximum efficiency. You should answer the following question:

- Which libraries are required?
- How many disk drives are available, and where on these devices should the individual libraries be placed?
- How large should each of the libraries be and what should they contain?

You should also decide how often to backup your system (and data) files, the program to be used (perhaps FCOPY or BACKUP/RESTORE), and determine the type and frequency of maintenance to be performed (such as reorganization).

Information on planning your libraries can be found in the chapter Planning the System in the DOS/VSE System Management Guide, GC33-5371. Information on disk space requirements for libraries and their directories can be found in the Librarian chapter of the DOS/VSE System Control Statements Manual, GC33-5376

This chapter will discuss the considerations for migrating from your System/3 Sort to DOS/VS Sort/Merge. The first section will present a comparison of sort features, and later sections will present specific conversion information.

Sort statements written for the System/3 Disk Sort programs must be completely rewritten when converting to DOS/VSE, as neither System/3 CCL statements nor System/3 Sort sequence specifications are compatible with DOS/VSE job control statements or DOS/VS Sort/Merge control statements.

The sections "Converting Sequence Specifications" and "Converting OCL Statements" discuss some of the points you should consider when converting a System/3 Disk Sort application into a DOS/VS Sort/Merge application. Features of the DOS/VS Sort/Merge for which there is no equivalent in System/3 Disk Sort are not discussed.

DIFFERENCES

Most of the facilities available to the user of the System/3 Disk Sort are also available in the DOS/VS Sort/Merge Program Product 5746-SM2.

- DOS/VS Sort/Merge can merge sequenced input files as well as sort.
- DOS/VS Sort/Merge can link to user-written assembler routines at points in the sort/merge called program exits. At these exits, the user-written routines may write or check labels, open or close files, take checkpoints, insert, modify, or delete records, read the input file, write the output file, or process I/O errors.
- DOS/VS Sort/Merge allows use of more than one workfile, and allows use of a multi-extent workfile, but job control statements specifying a workfile must always be present.
- DOS/VS Sort/Merge does not support forced control fields; if this function is required, it must be performed by a user-written routine at a program exit.
- DOS/VS Sort/Merge allows only one INCLUDE/OMIT statement; but the statement can contain many selection conditions (equivalent to many record type specifications).
- DOS/VS Sort/Merge does not support multiple record types directly; control fields must have the same relative position in each record. However, any record type (or types, if the control fields are equivalent) can be accessed by selecting that type (or types) with an INCLUDE/OMIT statement.
- DOS/VS Sort/Merge does not support comparison of zone or digit parts only of a field. All fields specified must be a whole number of bytes long and must begin and end on a byte boundary.

The following sections contain some of the considerations in converting your sorts from System/3 to DOS/VSE. Additional information can be found in the DOS/VS Sort/Merge Programmer's Guide, SC33-4044

CONVERTING SEQUENCE SPECIFICATIONS

This section discusses some of the points you should consider when converting System/3 Disk Sort sequence specifications to DOS/VS

Sort/Merge control statements.

1. The information on the header specification line and the field description lines for control fields is used on the SORT and MERGE control statement. Here you specify your control fields and their formats. You will get a tag-along sort (with the output records identical to the input records) unless you specify:
 - The ADDROUT option on an OPTION statement (to get an addrout sort)
 - Summary fields on a SUM statement (to get a summary sort)
 - Record reformatting by means of the OUTREC statement (partial tag-along).
2. The information on the field description lines about data fields need not be specified if you want a tag-along sort where the output records are identical to the input records.
3. To specify a tag-along sort with record reformatting, where the output records are not identical with the input records, code an OUTREC statement, specifying which fields from the input record are to appear in the output records, and how the fields are to be aligned within the records.
4. To drop control fields from records, code an OUTREC statement, specifying which fields from the input record are to appear in the output records, and how the fields are to be aligned within the records.
5. To select records for inclusion in or omission from the sort, code an INCLUDE/OMIT statement. Even where the System/3 Disk Sort requires more than one record type specification, all the selection conditions can be coded on one INCLUDE/OMIT statement. Note that the DOS/VS Sort/Merge accepts only one INCLUDE/OMIT statement in a sort application.
6. To specify an addrout sort, code an OPTION statement with the keyword ADDRROUT. Note: The length of the disk addresses produced is different for SAM and VSE/VSAM files.
7. To specify a summary sort, code a SUM statement to define the fields to be summarized. Note that you do not define summary overflow fields; if DOS/VS Sort/Merge detects an overflow condition, the pair of records involved remain unsummarized.
8. To handle signed control fields, you need only specify the correct format parameter on the SORT statement when the field is defined as a control field. No special coding is involved.
9. You must always code a RECORD statement to describe your records; You will need to code an INPFIL and an OUTFIL statement to describe your input and output respectively, unless the default values assumed will be satisfactory.

CONVERTING OCL STATEMENTS

This section discusses some of the points you should consider when converting the System/3 OCL statements necessary for a sort to the DOS/VSE job control statements necessary for Sort/Merge. Additional information on OCL to JCL conversion can be found in Chapter 7.

1. The // LOAD statement is replaced by the // EXEC statement, which must appear last of the job control statements, immediately before the sort/merge control statements.
2. Each // FILE statement must be expanded into a // ASSGN statement, a // DLBL statement, and one or more // EXTENT statements. If tape files are to be used, // TLBL statements may be necessary. You may

not need the // ASSGN statement if your installation has a suitable permanent assignment for the file in question.

3. The // RUN statement is not used.
4. A // JOB statement must be coded and placed first, before all other statements. This is then followed by all the necessary // ASSGN, // DLBL, // EXTENT, or // TLBL statements. The // EXEC statement must appear next, followed by the sort/merge control statements, /*, then the /6

CONVERSION PROCEDURE

The conversion procedure for creating DOS/VS Sort/Merge programs consists of the following steps:

- Review the System/3 and DOS/VSE sort differences
- Make manual coding changes necessary, including JCL, using the guidelines presented.
- Test the converted programs
- Update the documentation when the programs are running correctly

In reviewing the differences between the sort programs, note especially the types of files that are and are not supported by the DOS/VS Sort/Merge.

- VSE/VSAM and SAM files are supported
- If you are currently using the System/3 Sort program to sort card or diskette input, you must code an exit to the DOS/VS Sort/Merge to be able to process these files. The DOS/VS Sort/Merge Programmer's Guide gives a sample of the necessary coding for a card file. You may also load the cards or diskette onto a disk file, then sort the records.
- If you have diskette files to be sorted, another alternative is to use the Diskette Sort program (Installed User Program 5796-PGJ). This IUP consists of source macros that can be assembled into a user exit of the DOS/VS Sort/Merge program to allow processing of your diskette input files.
- If you are currently sorting System/3 indexed files and creating an ADDRROUT file, and are planning to convert this file to VSE/VSAM for DOS/VS, you cannot use the ADDRROUT option of the sort. The reason for this is that when you add records to a VSE/VSAM file, other records already in the file can be physically moved to accommodate the addition. In this case an ADDRROUT sort made before the addition is no longer accurate.

If you are using the record handling features of the System/3 Sort such as multiple record types, zone or digit-only control fields, etc., you may want to write preprocessor programs for your sort to extract the records wanted, and also to do special formatting of the control fields. This extracted file can then be input to the sort.

SORT EXAMPLE

Following is an example of a System/3 sort recoded for the DOS/VSE Sort/Merge program.

Assume that a file to be sorted contains records in the following format:

COL	1-3	ITEM CD
	4-9	BLANK
	10-15	ITEM NO
	16-19	BLANK
	20-25	PRICE
	26-34	BLANK
	35-40	BALANCE
	41-44	BLANK
	45-51	RECORD
	52	DELETE CODE

The output record should be in the following format after sorting the records, with ITEMCD as the major control field and ITEMNO as the intermediate control field.

COL	1-3	ITEM CD
	4-9	ITEM NO
	10-16	RECORD
	17-22	BALANCE

Records that have the character 'X' in the DELETE CODE should be ignored.

Figure 51 is an example of System/3 Disk Sort and DOS/VSE Sort/Merge coding, which includes reformatting of the above-mentioned fields.

```
// LOAD $DSORT,F1 // JOB SORT
// FILE NAME-INPUT,PACK-222222,UNIT-D2, // ASSGN SYS001,201
// LABEL-WORKA,RETAIN-S // DLBL SORTOUT,'SORT OUTPUT AREA'
// FILE NAME-WORK,PACK-222222,UNIT-D2, // EXTENT SYS001,,1,0,20000,2000
// TRACKS-20,RETAIN-S // ASSGN SYS002,201
// FILE NAME-OUTPUT,PACK-222222,UNIT-D2, // DLBL SORTIN1,'SORT INPUT AREA'
// LABEL-WORKB,TRACKS-20,RETAIN-T // EXTENT SYS002,,1,0,22000,2000
// RUN // ASSGN SYS003,201
// HSORTR 9A X22 // DLBL SORTOUT,'SORT WORK AREA',0,DA
0 C 52 52EQCX // EXTENT SYS003,,1,0,24000,4000
FNC 1 3 // EXEC SORT
FNC 10 15 SORT FIELDS=(1,3,A,10,6,A),FORMAT=CH
FDC 1 3 RECORD TYPE=F,LENGTH=52
FDC 10 15 INPFIL BLKSIZE=2080
FDC 45 51 OUTFIL BLKSIZE=2200
FDC 35 40 OMIT COND=(52,1,EQ,C'X'),FORMAT=CH
// END OUTREC FIELDS=(1,3,10,6,45,7,35,6)
OPTION PRINT=ALL
/*
/8
```

Figure 51. System/3 and DOS/VSE sort job

VSE/POWER WITH OPTIONAL REMOTE JOB ENTRY FEATURE

VSE/POWER, licensed program (5746-XE3), is the spooling program used with DOS/VSE. If you currently have a System/3 you are probably familiar with the concept of spooling. VSE/POWER will provide you with additional functions, thus greater operational flexibility. VSE/POWER performs spooling of card image data read from a tape unit, diskette reader or card reader. To install VSE/POWER the program product VSE/Advanced Functions is required. Up to 6 partitions can be spooled by VSE/POWER. Some of the additional functions of VSE/POWER not provided by System/3 are:

- Remote Job Entry (RJE) facility that permits DOS/VSE jobs to be entered into the system from remote terminals and output to be received at a batch terminal
- Job Entry Control Language (JECL) to give a operator greater flexibility in specifying the operating environment of programs
- More operator freedom in starting, stopping, and controlling the spooling function
- Jobs can be held on the reader queue after execution, and output can be held on the print/punch queues after printing/punching
- Access by VSE/POWER of job control and data that is stored in the source statement library

Figure 52 shows the data flow through VSE/POWER. The paragraphs that follow discuss the steps in the figure.

1. Input

VSE/POWER reads job streams from card, disk, or diskette devices for the partitions it controls. This input consists of DOS/VSE job control statements, data, and optionally VSE/POWER job entry control language (JECL) that can indicate special handling for the job or jobsteps.

2. Reader Task

This card, tape, or diskette input is read by a reader task and placed onto disk. Depending on the selected JECL options execution is scheduled directly, or must be scheduled by the operator. Jobs will execute according to the job's priority.

3. Intermediate storage

Intermediate storage used by VSE/POWER consists of three files, a queue file (which contains control information about each job), a data file (which contains the actual job input and output), and optionally an account file (which contains statistics about the number of cards read, lines written, etc.). Job input is stored in the data file reader queue.

4. Execution Processor Task

As a partition controlled by VSE/POWER is executing, there are one or more execution processor tasks initiated by VSE/POWER to process the unit record input and output for the job in that partition. The execution read task retrieves data records from intermediate storage

and presents them to the user partition where they are processed. The execution writer tasks intercept the output from the user partition and transfer it to intermediate storage.

5. Intermediate Storage

Printed and punched output is stored in the list queue and punch queue of the VSE/POWER data file.

6. Writer Tasks

The list and punch tasks print and punch data from intermediate storage, based on information supplied in the JECL.

7. Output

Output is physically printed or punched by the VSE/POWER partition while other jobs are being executed in the user partitions.

8. Operator Communications Task

The operator communications task handles all the communications between VSE/POWER and the console operator. It is always present and active in VSE/POWER to allow the operator to start and stop the reader and writer tasks, and modify jobs in the VSE/POWER queues.

ADVANTAGES OF SPOOLING

In addition to the overlapping of unit record input and output with job processing, spooling will provide some other advantages.

With spooling, you will need less I/O equipment than you would with basic multiprogramming. Both the System/3 and DOS/VSE require a separate reader, printer, and punch device for each partition that is not being spooled. When a spooling program is in operation, one card reader, punch, and printer can perform all the unit record I/O operations for all partitions being spooled.

Using a spooling program will mean that you have greater availability of your system. Without spooling, a job that produces printed output, for example, is dependent on having a printer available. If the printer has a hardware problem and is unavailable, that job cannot be run. With spooling, whenever a spooled reader, punch, or printer becomes inoperative, the system can continue processing with those jobs already in the reader queue on disk and store the output in the punch and print queues. When the I/O unit becomes available again, reading, punching, and printing can continue.

GENERAL COMPARISON

The overall spooling function of System/3 and DOS/VSE is similar. The spooling program handles the overlapping of unit record input and output with the execution of jobs. The differences arise mainly in how job options are specified to the spooling program, and the commands the operator uses to communicate with the spooling program. VSE/POWER provides some additional facilities not available with System/3 spooling, such as remote job entry. There is usually a greater range in the options that can be specified. For example, job priority is from 0 to 5 for the System/3, from 0 to 9 for DOS/VSE. These extra facilities will not be discussed here, but you can refer to the VSE/POWER Installation Guide and Reference, SH21-5329, for additional information.

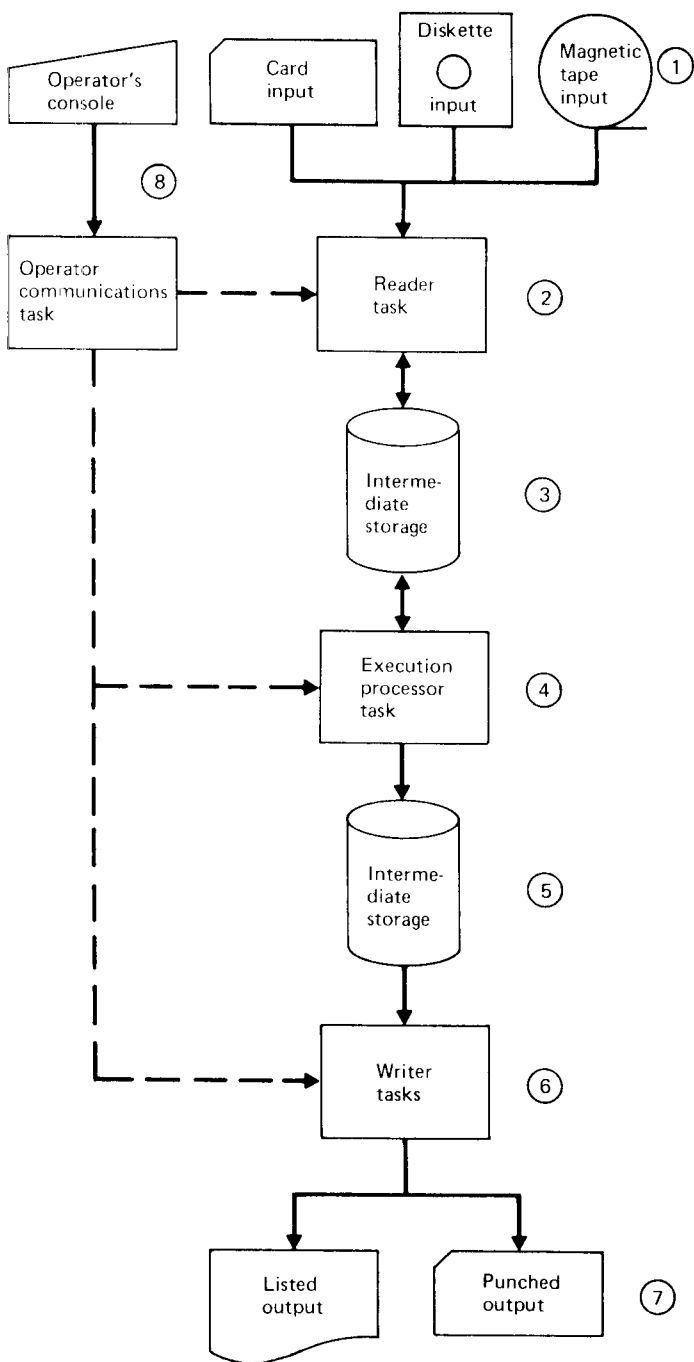


Figure 52. VSE/POWER Data flow.

JOB OPTIONS

In the System/3, three OCL statements are used to provide information about individual job options to the spooling program. These are the // JOB, // PRINTER and // PUNCH statements.

In VSE/POWER special information about job processing is provided in VSE/POWER job entry control language (JECL). Information appearing in the above-named System/3 statements would appear on the VSE/POWER * \$\$ JOB,* \$\$ LST and * \$\$ PUN statement, respectively. All VSE/POWER JECL statements for jobs entered locally at the processor begin with a * \$\$ in columns 1 through 4. All the JECL examples shown in this chapter will be for local devices.

Figure 53 illustrates SYSTEM/3 spooling control statements, and how the same options would be indicated using VSE/POWER JECL.

Figure 54 compares the parameters that control spooled jobs for the System/3 and VSE/POWER. The additional parameters of the DOS/VSE JECL statements are not shown. For further information, see the manual Entry Level User's Guide and the manual VSE/POWER Installation Guide and Reference.

<u>System/3</u> // FIRST JOB PRIORITY-3,PARTITION-1 // PRINTER FORMSNC-123,COPIES-2,PRIORITY-2 // PUNCH CARDNO-050,COPIES-3,PRIORITY-4 /*
<u>VSE/POWER</u> * \$\$ JOB JNM=FIRST,PRI=3,CLASS=A * \$\$ LST FNO=123,COPY=2,PRI=2 * \$\$ PUN FNO=050,COPY=3,PRI=4 * \$\$ EOJ

Figure 53. Sample spooling control statements

JECL statements may be placed anywhere in a DOS/VSE job or jobstream, which means that one DOS/VSE job may be the same as one VSE/POWER job; one DOS/VSE job may be several VSE/POWER jobs; or several DOS/VSE jobs may be one VSE/POWER job. Normally, through, a DOS/VSE job would be the same as a VSE/POWER job. If this is the case, then JECL is not required, but will probably still be used to supplement the DOS/VSE JCL. Figure 55 illustrates several combinations of DOS/VSE and VSE/POWER jobs.

SYSTEM/3	VSE/POWER	Comments
//JOBNAME JOB	* \$\$ JOB JNM=jobname	Indicates the job name to the spooling program
PRIORITY-priority	PRI=priority	Indicates job priority. System/3 PRIORITY-0 is equivalent to VSE/POWER DISP=H of a * \$\$ JOB
PARTITION-partition	CLASS=class	System/3 specifies the partition number. VSE/POWER jobs run in a partition according to the class specified.
Spool=yes/no	---	If the spooler(VSE/POWER) is inactive,DOS/VSE jobs treat the spooler control statements as comments.
COR=core	---	DOS/VSE virtual partition allocated large enough to execute largest job
//PRINTER FORMSNO=formstype	* \$\$ LST FNO=formstype	Formstype for System/3 is 1 to 3 characters.For VSE/POWER formstype is 1 to 4 characters.
COPIES-number	COPY=number	Specifies number of copies.
DEFER=yes/no	RBS=number	Normal VSE/PCWER mode is like DEFER=yes in the System/3. RBS= specifies number of pages to build before printing starts.
ALIGN=yes/no	---	VSE/POWER PSETUP command is for forms alignment. Normal operation is like System/3 ALIGN=no.
PRIORITY-priority	PRI=priority	Priority of the printed output.
//PUNCH CARDNO-cardtype	* \$\$ PUN FNO=cardtype	For the System/3,1 to 3 characters.For VSE/POWER 1 to 4 characters.
COPIES-number	COPY=number	Number of copies
DEFER=yes/no	RBS=number	RBS= command allows for printing before job has completed. Normal mode of VSE/POWER is like DEFER=yes on System/3.
PRIORITY-priority	PRI=priority	Priority of the punched output.
/*	* \$\$ EOJ	End of job

Figure 54. Spooling control statements and parameters.

VSE/POWER JOB NUMBER	DOS/VSE JOBSTREAM	COMMENTS
1	// JOB ONE * // EXEC STEPA * /E	One DOS/VSE job is one VSE/POWER job. If run in background JECL isn't necessary
2	* \$\$ JOB TWO // JOB TWO * // EXEC STEPB * /E * \$\$ EOJ	One DOS/VSE job is one VSE/POWER job. optional JECL is used
3	* \$\$ JOB THREE // JOB THREE * // EXEC STEPC * /E // JOB THREEB * // EXEC STEPD * /E * \$\$ EOJ	multiple DOS/VSE jobs in one VSE/POWER job. (* \$\$ JOB and * \$\$ EOJ are both required)
4	* \$\$ JOB FOUR // JOB FOURFIVE * // EXEC STEPE * \$\$ EOJ	Multiple VSE/POWER jobs for one DOS/VSE job
5	* \$\$ JOB FIVE * // EXEC STEPF * /E * \$\$ EOJ	

Figure 55. DOS/VSE and VSE/POWER jobs

JECL statements need not be removed from a job if you are not running VSE/POWER. Because all JECL statements begin with an*, they are treated as comments by DOS/VSE. The options the JECL statements would specify to VSE/POWER will be ignored when read by DOS/VSE.

A * \$\$ RDR JECL statement is used to insert a diskette file into input being read from the card reader. The diskette is a SYSIN device (80 bytes) when no physical number is specified. It is a data input device (up to 128 bytes) when the physical unit is specified.

OPERATOR CONTROL OF SPOOLING

As in System/3 spooling, a DOS/VSE operator can communicate with the spooling program to start and stop reader and writer tasks, and to modify jobs in the queues. VSE/POWER provides a set of operator commands that can be used to perform the various functions. In general, there is a one-to-one correspondence between the commands used by a System/3 operator and a DOS/VSE operator. Since VSE/POWER provides additional facilities, the DOS/VSE commands will have some extra parameters. As in

System/3, the VSE/POWER commands may be abbreviated. The commands may be entered through the operator console on the 4331 Model II or J1. Figure 56 presents a summary of the functions an operator can perform to control spooling.

FUNCTION	System/3 Command/Abbr	VSE/POWER Command/Abbr	Comments
Start a reader/writer task	START/ S	PSTART/ S	Same
Start partition	START/ S	PSTART/ S	Brings a partition under DOS/VSE control
Stop a reader/writer task	STOP/ P	PSTOP/ P	Same
Stop partition	STOP/ P	PSTOP/ P	Releases a partition from DOS/VSE control
Restart printed or punched	RESTART/ T	PRESTART/ T	VSE/POWER can restart from the beginning, or plus or minus a number of pages within the LST queue file
Hold a job in the reader/printer/punch queues	HOLD/ H	PALTER/ A	VSE/POWER can hold the job in queue by using PALTER to change the disposition to hold (DISP=H)
Release a job in the reader/printer/punch queues	RELEASE/ R	RELEASE/ R	VSE/POWER can release job by using PALTER to change the disposition to dipatch (DISP=D)
Change the characteristics of a job in the reader/printer/punch queues	CHANGE/ G	PALTER/ A	VSE/POWER can cchange PRI=,CLASS=,and COPY= of jobs in the queues
Change partition priority	PTY/ Y		Use DOS/VSE command PRTY to alter priority
Delete an entry from the reader/printer/punch queues	CANCEL/ CN	PDELETE/ L	PDELETE will not delete queue being processed
Cancel a program executing in a partition	CANCEL/ CN	PFLUSH/ F	Same function
Display queue entries	DISPLAY/ D	PDISPLAY/ D	Same function

Figure 56. Operator Commands.

GENERATION OF SPOOLING SUPPORT

With the System/3 you decide whether you want spooling in the system at system generation time. With System/Exx, VSE/Advanced Functions is required to use the spooling program VSE/POWER. Prior to installing VSE/POWER you must plan how you will use the spooling-form numbers, job priorities, etc. The steps in including VSE/POWER in your system, and the decisions to be made are discussed below. Detailed information on planning for, generating, and using VSE/POWER can be found in the

DOS/VSE Entry User'S Guide.

You may want to create your own version of the VSE/POWER program. A preassembled VSE/POWER program is supplied with the DOS/VSE system. If this program does not meet your needs, then you can generate a version that includes the appropriate options. Next you must catalog the VSE/POWER program into your core image library.

VSE/POWER requires disk space for its files: the queue file, the data file, and the optional accounting file. You must calculate how much space VSE/POWER will use on your direct access volumes. Most of the considerations for determining disk space are the same as for the System/3. The following can impact the amount of disk space needed:

- Number of records in the input stream
- Number of lines or cards of output
- Length of time an entry will remain in the queue
- How many entries will be in the queues at any time

VSE/POWER allows multiple extents for its disk files, and these extents can reside on several disk volumes. The System/3 allows a maximum of one volume.

Once you have determined the disk requirements for VSE/POWER, you should place the label information in the label information area as standard labels, and consider permanently ASSIGNing the DASD devices VSE/POWER will use.

You should develop some standards for the options that can be specified on JECL statements.

- In which partition will the jobs be run?
- What are the job priorities?
- What names will be used for special card and printer forms?

You will also have to develop procedures for your operator to use in starting VSE/POWER, shutting it down, and communicating with VSE/POWER. You will probably want to make the startup procedures automatic, so the operator is not required to make many responses to system messages. These procedures should be documented in the operator's run book.

SUMMARY OF SPOOLING DIFFERENCES

1. In System/3, the spooling program does not reside in a separate partition. The VSE/POWER program resides in one of the DOS/VSE partitions. In each case the spooling program can control all of the other partitions defined.
2. In the System/3, the operator must decide at IPL time to cancel spool or let it run. If spooling is canceled and later must be used, the operator must re-IPL. In DOS/VSE, the operator can initiate and shut down VSE/POWER at any time. Normally, however, it is initiated after IPL and runs throughout the day's work.
3. System/3 spooling jobs are controlled with OCL and by the operator. VSE/POWER jobs are controlled by JECL.

CHAPTER 12. OPERATOR CONTROL OF THE SYSTEM

This chapter will discuss some of the operational differences between a System/3 and a 4300 running DOS/VSE. It is not intended to be an operator's manual, but rather a guide to point out the different procedures for loading and running jobs, and responding to system messages. A complete description of operation procedures is found in the manual DOS/VSE Operating Procedures, GC33-5378.

Operational characteristics and commands of spooling are covered in the Spooling chapter.

THE OPERATOR'S DUTIES

Operator-system communication plays a vital part in the efficient operation of a data processing installation. DOS/VSE provides facilities that ensure timely and effective interaction between operator and machine.

The System/3 operator and the DOS/VSE operator will perform similar duties when operating the system. The operator's primary duties in a computer installation are to:

- Start up the system
- Prepare the I/O devices; for example, mount tapes for each individual job
- Initiate and control the execution of jobs
- Interpret and respond to system or program requests for information or actions.

More specifically, the operator of a DOS/VSE-controlled computing system can:

- Initiate and control multiprogramming operation
- Interrogate the system to obtain information about its status
- Cancel the execution of a job, for instance, in the case of an unending program loop.
- Diagnose problems with the help of problem determination aids.

For its part, the system:

- Alerts the operator when a condition requiring his intervention occurs
- Provides information such as start and stop times of jobs and run times.

SYSTEM-OPERATOR COMMUNICATION

Communication from the system to the operator is in the form of messages written on the operator communications device. The messages describe the situation that requires operator action. Each message is identified by a unique number. This number serves as an entry point into the DOS/VSE Messages Manual, (GC33-5379) where an explanation is given for each

message, along with a description of the action required.

Operator responses to messages are generally short, one-word answers, such as RETRY, IGNORE, or CANCEL. Alternatively, the operator can allow the system default option to take effect in most situations requiring intervention.

Communication from the operator to the system is in the form of commands and replies to messages. There are three types of commands:

- Job control commands: almost identical in format and scope to the job control statements.
- Operator commands: statements for performing controlling duties (for example, changing the size of partitions), or for asking for specific information (for example, assigning I/O devices)
- Screen control commands: statements for manipulating the information displays on the operator console.

The operator can intercommunicate with the system at any time by typing in commands. In addition, he can instruct the system to suspend processing after the current job or jobstep in any partition, for example, in order to allow time for changing printer forms or device assignments.

In addition to operator-system communications, there can be direct operator problem program communication, provided that the problem program has a special operator communications routine.

These points are discussed further in the following sections.

OPERATOR CONSOLE

DOS/VSE allows system-operator communication through the operator console. The operator console is similar to the System/3 Model 15 CRT/keyboard in its manner of operation. It replaces the conventional processor console panel with its numerous controls and indicators.

The operator console is used to:

- Log messages from the system and user programs
- Accept operator responses to messages
- Accept operator commands
- Display messages about operating conditions you should be aware of, and incorrect use of the console
- Display warning messages that must be resolved
- Display system hardware status.

If you are currently a System/3 Model 10 user, then you can think of the operator console as replacing the functions performed by the display lights on the processing unit, and the logging function of the printer or the 5471 Printer-Keyboards.

The operator console consists of two parts: the display unit and the operator keyboard with control panel.

DISPLAY UNIT

The operator console display unit can display 25 lines of information. The top 20 lines of the operator console are used for the display of messages and the entry of data. The cursor is automatically positioned

to the line where the operator can enter data. Figure 57 shows the format of the display screen.

Each line on the screen is a maximum of 80 characters. The format of the messages is shown in Figure 58.

Line	1	.
		.
		• Lines 1 through 20
		• are used for the
		• display of messages
		• and entry of data
		• by the operator.
		.
		.
	20	
	21	Machine Status
	22	Manual Operations Information
	23	Reserved for Service Personnel
	24	Reserved for Service Personnel
	25	Status of Screen and Keyboard

Figure 57. Format of display screen.

Blank	LINE NUMBER	BLANK or *	PARTITION INDICATOR	MESSAGE TEXT
1	2	4	5	7----->through 80

Figure 58. Format of message line.

Operator Keyboard

The keyboard contains keys similar to those of a typewriter, which you use to give commands to the system. It also contains some special keys for positioning of the display screen cursor or interrupting processing.

The 4300 operator console does not have program function keys as does the CRT/keyboard of the System/3 Model 15.

Control Panel

The control panel, which is located at the top of the keyboard, contains a number of switches, keys, and lights. The keys are used for basic tasks such as making the system operational. The lights alert you to check conditions in the system.

Using the Display Unit

DOS/VSE messages, job control statements, and operator responses that are displayed on the operator console are also stored in a special area on disk called the hard-copy file.

Since not all of the messages can be retained on the screen DOS/VSE

provides an operator command to let you redisplay console messages which have rolled off the screen. The D command is used to redisplay this information from the hard-copy file.

There is also a special utility program named PRINTLOG that enables you to list the contents of the hard-copy file on SYSLST, the system printer. It is discussed in the chapter Utilities.

DOS/VSE MESSAGES

DOS/VSE does not use a system of halt codes to provide the information to the operator, but rather displays a message number and abbreviated description of the message.

The system communicates with the console operator by issuing messages to SYSLOG, the logical unit to which the console is assigned. Each message is preceded by a partition identifier and a message code, which includes an action indicator. This action indicator shows what operator response is required, grouped into four types of messages:

- Action (A) messages indicate an action the operator must perform
- Decision (D) messages indicate the operator must make a choice of actions
- Information (I) messages do not require specific operator action
- Eventual-action (E) messages indicate some action must be taken eventually
- System Wait-action (W) operator action is required because a hardware failure occurred. It may be necessary to set hardware switches and/or run recovery programs.

Do not rely on your memory but make a habit of looking up each message that is issued by the system. This will save you time and trouble. Full details of all DOS/VSE messages, including operator action and responses, are contained in DOS/VSE Messages, GC33-5379.

Figure 59 shows an example of an action-type message. The format for the other types of messages is the same, except that the A will be replaced by a D, I, or E.

```
01*BG 1C10A    PLEASE ASSGN SYSRDR

01 - Line number 1 on the screen
* - Is set to blank when the message is responded to
BG - Message refers to the background partition
1 - A job control message
C - Either job initiation or job termination issued
    the message
10 - This is the message number
A - Indicates that the message requires an operator
    to respond, in this case the operator would
    assign SYSRDR to a physical device
PLEASE ASSGN READER - This is the message text
```

Figure 59. Console message example.

PROGRAM MESSAGES

Messages that are issued from application programs should be documented in the operator's run book, along with the proper response formats. This information is supplied by the application programmer. You should be aware that although the characters you type in appear on the screen as uppercase letters, they are actually treated by the system as lowercase. Either the application program should accept both upper- and lowercase, or you must remember to use the shift key when typing responses to program messages.

WAIT STATES

Whenever operator action or a decision is necessary, the program that issued the message waits until you have entered an appropriate response to the message.

A number of errors may cause the system to enter an uninterruptable wait state. This is indicated by the wait light on the processor console.

OPERATOR COMMANDS

Although the functions that you perform in communicating with and controlling DOS/VSE are similar to what you did on the System/3, the formats of the commands and the parameters you specify will be different. Figure 60 provides a summary of the DOS/VSE operator commands that you would use to perform various functions. You should use the Operator's Library DOS/VSE Operating Procedures, GC33-5378, for detailed information on each command and procedure.

The DOS/VSE Entry User's Guide contains information that will be helpful to you as an operator. It provides a good description of starting the system and using the DOS/VSE spooling program, VSE/POWER. It should not, however, be used as an operator's guide.

I/O DEVICES

As an operator of a DOS/VSE System, you will need to learn about the operating procedures for the devices on your system. Many of these may be the same as the ones that were on your System/3, and they will operate in a similar fashion on your 4300. Other devices will be different, and you will have to learn some new procedures. The hardware manuals for the specific device can provide you with that information.

SUMMARY OF OPERATION CONSIDERATIONS

1. DOS/VSE is always operated like the System/3 in NOHALT mode.
2. DOS/VSE does not use halt codes, but provides the operator with an error message number and an abbreviated description of the message.
3. The DOS/VSE unit of work is called the job. It begins by a // JOB statement and ends with a /& statement. A job may consist of the execution of several programs.
5. The DOS/VSE system logical devices that are of most interest to the operator are:
 - SYSIN - the system input device
 - SYSLST - the system print device
 - SYSPCH - the system punch device
 - SYSLOG - the system logging device
6. DOS/VSE does not provide a rollout/rollin facility as does the System/3. With the extra partitions available with DOS/VSE, it is not necessary to interrupt an already executing program. Simply generate one more partition than you normally use, giving it the highest priority. When a job must be executed immediately, you can run it in this extra 'hot job' partition.

FUNCTION	System/3 (model 15)	DOS/VSE	COMMENTS
Cancel program operation in a partition	CANCEL	CANCEL	Same
Set the system date	DATE	SET DATE	DOS/VSE date is set at IPL time
Display history area (hard copy file)	DISPLAY	D	Same
Cause dump of partition or all of main storage	DUMP	DUMP	Same
Place system in HALT or NOHALT mode	HALT or NOHALT	---	DOS/VSE operates in nohalt mode
Turn on/off automatic delete of information messages	IDDELETE NOIDDELETE	---	DOS/VSE type I messages are deleted automatically
Set partition priority	PTY	PRTY	Same
Change System input device for a partition	READER	ASSGN SYSIN	Same
Interrupt program to allow another to execute	ROLLOUT	---	For DOS/VSE use an extra partition
Change partition size	SET	ALLOC	Same
Start a partition	START	START	Same
Stop a partition	STOP	STOP	Same
Set the system time of day	TIME	SET CLOCK	DOS/VSE time set at IPL time
Enable or disable system trace	TRACE	---	SDAIDS and PLAIDS supply trace
List I/O assignments	DISPLAY	LISTIO	Same
List Virtual and real storage allocations	DISPLAY	MAP	Same
Interrupt processing at end of job or job step	HALT	PAUSE	Same
Deactivate a partition	SET (to zero)	UNBATCH	Same
Make a device available or unavailable	---	DVCUP DVCDN	
Magnetic tape commands	---	MTC	

Figure 60. Operator Commands.

This chapter will discuss the general DOS/VSE system generation tasks and identify those areas that are different from a System/3 system generation.

In general, the procedures are the same for system generation on a System/3 and for DOS/VSE. You receive a standard system distributed by the IBM. This system must then be tailored to your own needs. You will generally create your own supervisor by specifying options relating to the features or programs you are to have on your system. You will also tailor the libraries to include your supervisor, other system programs provided by IBM such as system utilities, processing programs, and your own application programs. You will also install any IBM licensed program. You will plan for the disk space to contain the jobs to be spooled.

SUPERVISOR GENERATION

Supervisor generation involves the selection of options and features your system is to support. On the System/3 you would have only one supervisor containing the options needed. DOS/VSE will include several pregenerated supervisors. You may want to generate other supervisors that contain different options for use in special situations such as program testing. Each DOS/VSE supervisor is generated with a different name and is stored in the core image library. When the system is initialized the correct supervisor is selected. The procedure that automatically starts the system may be altered if a different supervisor is desired.

On the System/3, particularly the Model 10, you select supervisor options mainly on the basis of hardware devices attached to your system. If a piece of equipment is attached to your system, you indicate the support is to be included in your supervisor.

With DOS/VSE, the hardware devices are identified by the automated start up procedure. If you are testing on a 4300 Processor with different I/O you need only alter the startup procedure to identify the devices to DOS/VSE.

The programs that you will install on the 4300 Processor affect the options that you chose in generating DOS/VSE. You should be aware of which options are required for each program you plan to install.

TAILORING THE LIBRARIES

On the System/3, the libraries are created by the options you specify during system generation. Your new libraries are built including only the programs, routines, and source code needed to support your system. With DOS/VSE, the system generation procedure first creates the libraries containing all of the DOS/VSE code. You then tailor the libraries by deleting unwanted code. The procedure library contains cataloged procedures that you can use for this purpose. For both System/3 and DOS/VSE you specify at system generation time the initial sizes your libraries will have. These sizes must be large enough to contain the necessary code.

PLANNING FOR SYSTEM GENERATION

In most DOS/VSE installations, system generation is a system programmer function. There are many areas of DOS/VSE where the selection of an

option or feature can impact the performance of the system as a whole. Therefore, it is important that one person in your installation be responsible for the overall planning and implementing of your DOS/VSE system. This person will also be involved in the developing and implementing of standards for your DOS/VSE system such as naming conventions for programs, files and cataloged procedures, or the placement and use of disk files.

The following list is a summary of the decisions which must be made before system generation. It is important to evaluate and plan for each of these considerations carefully, because the decisions you make here can impact the rest of your conversion effort. The DOS/VSE Entry User's Guide provides detailed information on the planning and generation of DOS/VSE.

- Number and size of partitions to be supported
- Standard disk file labels
- Use of logical unit names
- System workfiles
- Libraries
- Printer buffer load programs
- Planning the implementation of program products

SCHEDULING SYSTEM GENERATION

Before your own 4300 arrives, you will want to generate your own DOS/VSE system. A good time to do this is when you begin to have application programs that are error free. When they have been successfully tested on a 4300 processor, you will want to place these programs in your own core image library.

You should schedule magnetic tapes to be delivered when you plan to start testing your programs. The number of tapes that you require would depend on the size and number of data files you have. You would require tapes to backup the system and private libraries and additional tapes to back up your data files. If you are installing a non-fixed block architecture DASD device you will have to order enough extra disk packs to back up the system and private libraries and your data files or use tape as a backup facility.

You can then plan to have DOS/VSE snipped to you, so that you can generate a system to support your testing requirements. You may find that you have to include some options different from what your own system will be, to accommodate your test facilities. You may choose to alter the automated start up procedure to include I/O devices that are different on the test system. After your own 4300 Processor is installed, you should use the supervisor that has been tailored for your own system, eliminating the options that were necessary only for testing.

UPDATING YOUR DOS/VSE SYSTEM

As with the System/3 Disk System Management programs, the DOS/VSE system has periodic updates and new releases that provide users with more features. You should generate a new system when required in order to take advantage of the additional features and also to keep your system at a current level.

MAINTAIN SYSTEM HISTORY PROGRAM

The maintain system history program provides a means of keeping an accurate history of your DOS/VSE system. It provide the following functions:

- Creation of a system history file when DOS/VSE is installed
- Updating of this history file whenever a product, component, or feature is installed.
 - 1.Product: Program package that contains "components" and "features" in the same distribution library.
 - 2.Component: Program package like CICS/VS,VSE/POWER,etc.
 - 3.Feature: Program package that contains functions not available in the originally supplied program.
- Checking of the history file during installation of a component or feature to insure that the prerequisite components, features or Program Temporary Fixes(PTFs) are present
- Application of PTFs, recording of these changes, and reversing the application of the corrections if required.
- Listing of the contents of the system history file in a variety of different formats.
- Backup and restore of the system history file.

The manual DOS/VSE Entry User's Guide explains the use of the maintain system history program and shows specific examples for using the program.

Some of the programs used on your System/3 may be application or other special purpose program packages. The extent to which program packages are used varies widely depending upon the type of application and user requirements within an application. Typically a System/3 used to process applications in the manufacturing industry will make greater use of program packages than System/3s used in other industries. This chapter will help you evaluate System/3 to 4300 conversion alternatives for your System/3 program packages.

The following sections discuss different types of IBM program packages and present guidelines to help you select a conversion alternative.

Later sections will suggest conversion alternatives for some of the more highly used System/3 program packages.

TYPES OF PROGRAM PACKAGES

For purposes of this discussion, a program package is considered to be one or more programs or routines that perform special functions. The following types of program packages are available from IBM.

- Program products (PP)
- Application Program Products (PPA)
- Field developed programs (FDP)
- Installed user programs (IUP)
- Programming RPQ's (PRPQ)

Each type of program package may have different programming maintenance services and license fees. These two points should not be overlooked in evaluating conversion alternatives.

Another way of classifying program packages when considering conversion is as follows:

- Generalized applications (e.g., Project Control using Job Analysis System/3)
- Industry applications (e.g., System/3 IPICS Engineering and Production Data Control)
- System support (e.g., System/3 DITTO)
- Data communications (e.g., DATA/3)

These classifications will be referenced in the following sections.

PROGRAM PACKAGE PROFILE

Before conversion alternatives are investigated the following information should be obtained for each of your program packages.

Type of program package (PP, PPA, FDP, IUP, or PRPQ)
Source language
Number of programs
Number, size, and type of data files
Retraining of end users
Number of modifications, if any, you have made to the package
Amount of library space needed to execute and maintain programs

Each of the above has a potential impact on the conversion method you

choose.

CONVERSION ALTERNATIVES

There are three major alternatives available when converting System/3 program packages to the 4300.

- Source code conversion of the System/3 program package
- Selection of an equivalent 4300 program package
- Design and code functionally equivalent programs

These alternatives are discussed in the following paragraphs.

SOURCE CODE CONVERSION OF THE SYSTEM/3 PROGRAM PACKAGE

The feasibility of this alternative for a particular program package can be assessed using the program package profile. Program packages written in System/3 RPG II can be converted using the System/3 DOS/VS RPG II Conversion Preprocessor. Those written in System/3 Assembler language must be manually recoded. The type of program, i.e., FDP, IUP, or PP and the programming services available/required must also be considered. In particular where many end users are involved, for example with manufacturing application packages, source code conversion may be your best alternative. You should review the program maintenance and license considerations with your IBM marketing representative. If you convert a System/3 licensed program to the 4300, the conversion constitutes a customer modification to that licensed program.

SELECTION OF AN EQUIVALENT 4300 PROGRAM PACKAGE

Details of available program packages can be found in the booklet "Program Information" (GE21-9949). The availability of 4300 program packages is constantly changing, particularly in field developed programs and installed user programs. You should investigate the availability of a functionally equivalent 4300 program package with your IBM marketing representative. Equivalent 4300 program packages are more likely to be available in the generalized application, systems support and data communications categories.

Data files and the data formats required by both the System/3 and 4300 packages must be considered when evaluating this alternative.

DESIGN AND CODE FUNCTIONALLY EQUIVALENT PROGRAMS

This alternative is suitable when you have a specific installation requirement that is not met by either of the other alternatives. An example of such a requirement is to install a data base using Data Language/I (DL/I). In this case the application area where the System/3 program package was used may need to be redesigned and recoded to take advantage of the DL/I facilities.

If a System/3 program package has been extensively modified to meet your requirements and your personnel are familiar with the application and programs, this alternative may be a practical long-term solution.

SELECTED SYSTEM/3 PROGRAM PACKAGES

This section presents several highly used System/3 program packages with suggested conversion alternatives.

Bill of Material Processor (BOMP)

The System/3 BOMP program product is used both as a standalone package and also in conjunction with other software available as program products or field developed programs. Generally the same conversion alternatives apply to System/3 BOMP and the System/3 program packages based on System/3 BOMP.

Both System/3 BOMP and the associated System/3 program packages are written in RPG II and can be source code converted using the RPG II Conversion Preprocessor. Thus one alternative available in this area is to convert the System/3 RPG II source code and use VSE/VSAM files. If DL/I data base facilities are chosen, then redesigning and coding as outlined above is required.

DATA/3

DATA/3 is a data communications-type program package. Ongoing programming services requirements and maintenance for new devices make source code conversion of such a package undesirable. A functionally equivalent 4300 program product is Display Management System/VS (DMS/VS) (5746-XC2). CICS/VS is a prerequisite for the installation of DMS/VS. DATA/3 programs would need to be recoded to DMS/VS specifications.

Job Analysis System/3

Job Analysis System/3 is written in System/3 Assembler. A similar project control-type program product for the 4300 is Project Analysis and Control DOS/VS (PROJACS) 5746-XP1.

Terminal Query Facility for System/3 (TQF/3)

A similar facility to TQF/3 can be provided by Generalized Information System (GIS/DOS/VS) (5799-ALX). GIS/DOS/VS programs execute in a batch partition of DOS/VSE. To provide an interactive terminal facility both CICS/VS and Source Program Maintenance Online II (5798-CFT) are prerequisites.

System/3 DITTO

VSE/Data Interfile Transfer, Testing and Operations (VSE/DITTO) program (5746-UT3) provides the functions available with SYSTEM/3 DITTO.

Part II compares CCP to the program product Customer Information Control System/DOS/VS (CICS/DOS/VS) (program number 5746-XX3). For convenience, this product is generally referred to in this manual as "CICS/VS".

CICS/VS functions that are equivalent to CCP functions are illustrated. This can help you carry out conversion planning from CCP to CICS/VS.

Particular emphasis is given to (1) CCP facilities commonly used in a 3270 terminal network, DFF (display format facility) environment; and (2) programs that process inquiry, inquiry/update, and data entry/order entry transactions in a transaction-oriented system.

References will be made throughout Chapters 15, 16, 17, and 18 to the following manuals:

Customer Information Control System/Virtual Storage (CICS/VS) Entry Level System User's Guide (DOS/VS) SC33-0086

Customer Information Control System/Virtual Storage (CICS/VS) Application Programmer's Reference Manual (Command Level), SC33-0077

Customer Information Control System/Virtual Storage (CICS/VS) Application Programmer's Reference Manual (RPG II), SC33-0085

Customer Information Control System/Virtual Storage (CICS/VS) General Information Manual SC33-0066

The design objectives of both CCP and CICS/VS are similar: to insulate the application program from the communications network and system environment.

The method of implementation and the degree to which particular facilities are implemented vary considerably, however. In many cases the difference is simply another technique of accomplishing the same function, e.g., DFF (Display Format Facility) and BMS (Basic Mapping Support). In a few cases, programs must be redesigned because of differences in the software architecture of CCP and CICS/VS. CICS/VS generally requires the programmer to perform fewer functions than CCP. For example, under CICS/VS, an application program is concerned only with a single terminal.

Designing a data communications system involves many activities, for example, a feasibility study, and system design, including screen, form, file, recovery/restart, and program design.

Design differences between CCP and CICS/VS will generally affect only program design and application coding with resultant changes in documentation and training. Most components of the original system design may remain unaltered.

Information on conversion from System/3 to 4300 is presented below as follows:

- CCP-CICS/VS Systems Overview

This section will illustrate design and implementation differences between CCP and CICS/VS. The overview is supported by diagrams of an inquiry program executing in both systems.

- CCP-CICS/VS Description and Summary

Chapter 15 will present the CICS/VS facilities available to the application programmer, terminal and system operator. A summary of the CCP and CICS/VS facilities is included.

- CCP-CICS/VS Conversion Considerations

Chapter 16 will present conversion considerations at both the system and application program levels. The differences between CCP and CICS/VS are summarized.

- CCP-CICS/VS Conversion Activities

This section summarizes the conversion activities and provides a checklist for CCP to CICS/VS conversion planning.

- System/3-4300 Processor Conversion Alternatives

This chapter summarizes the conversion alternatives available for System/3 data communications software other than CCP. Development, testing, and analysis aids for CICS/VS are included.

CCP-CICS/VS SYSTEM OVERVIEW

A diagrammatic representation of the CCP and CICS/VS data communications systems is shown in Figures 61 and 62. As you can see, CICS/VS additional software components, e.g., task management, transient data management, and temporary storage management.

To illustrate major differences in implementation between CCP and CICS/VS, Figures 63 to 67 show several simple inquiry programs executing in both a CCP and a CICS/VS system. Software components descriptions will refer to these diagrams. A description of the CICS/VS components as they would be used in a typical inquiry application is in the CICS/VS Entry Level User's Guide.

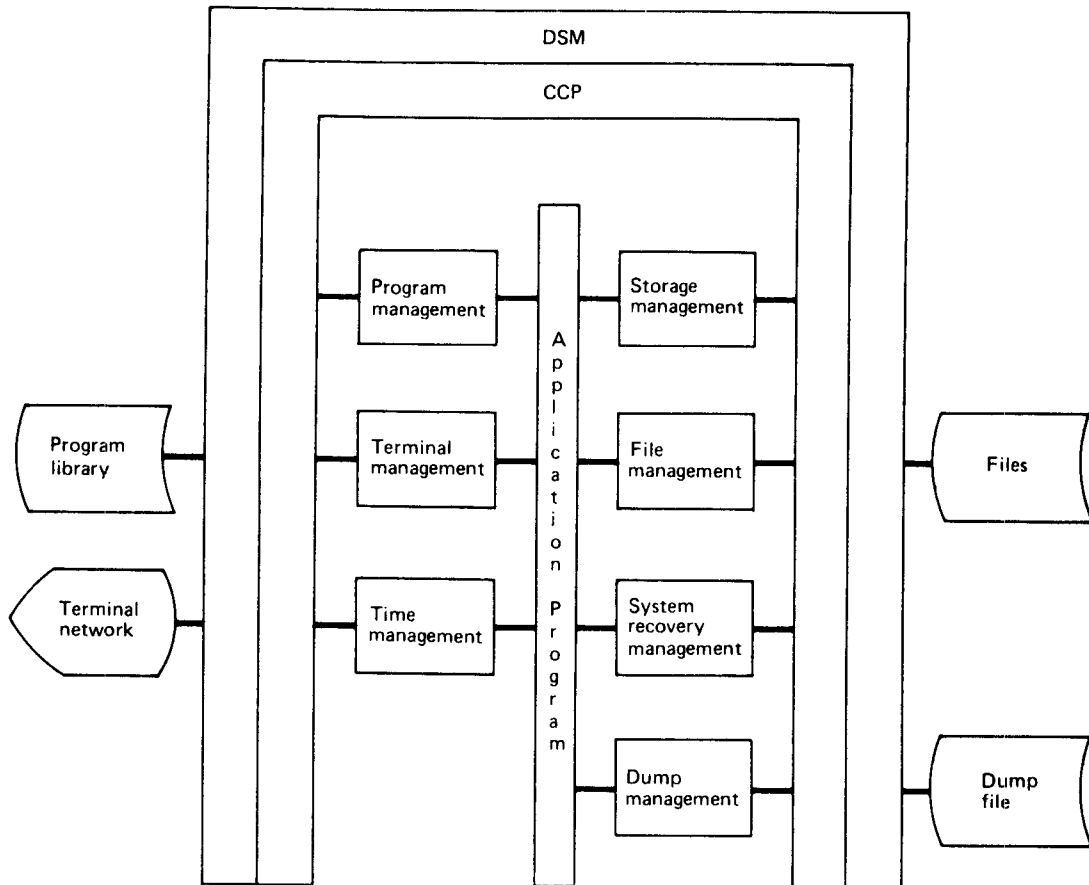


Figure 61. CCP system overview.

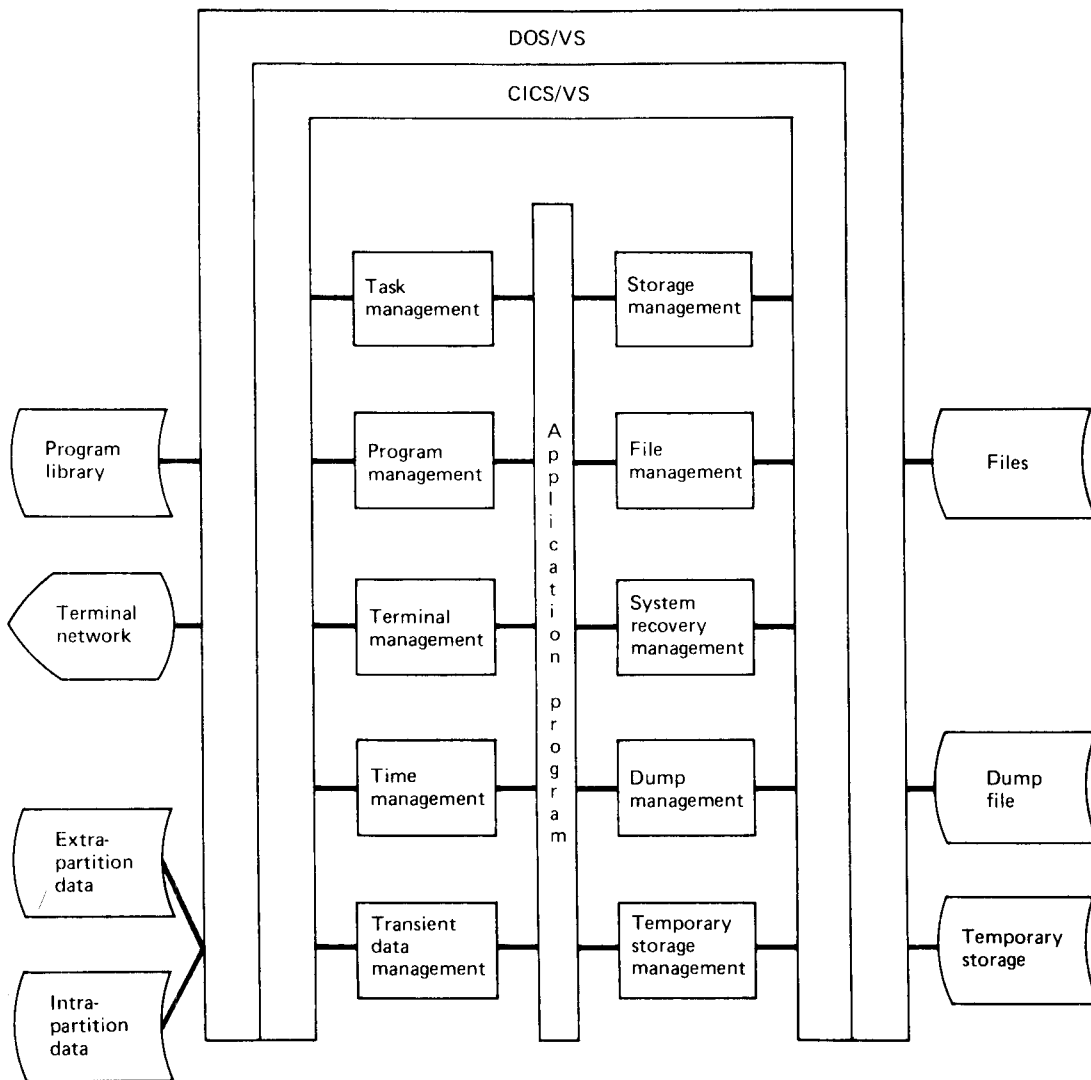


Figure 62. CICS/VS System Overview

In Figure 63 note that:

- Program A is a CCP/SRT program using display format facility (DFF)
- Data work areas and both terminal and file I/O areas allocated within the program
- Terminal 1 is under the control of program A.

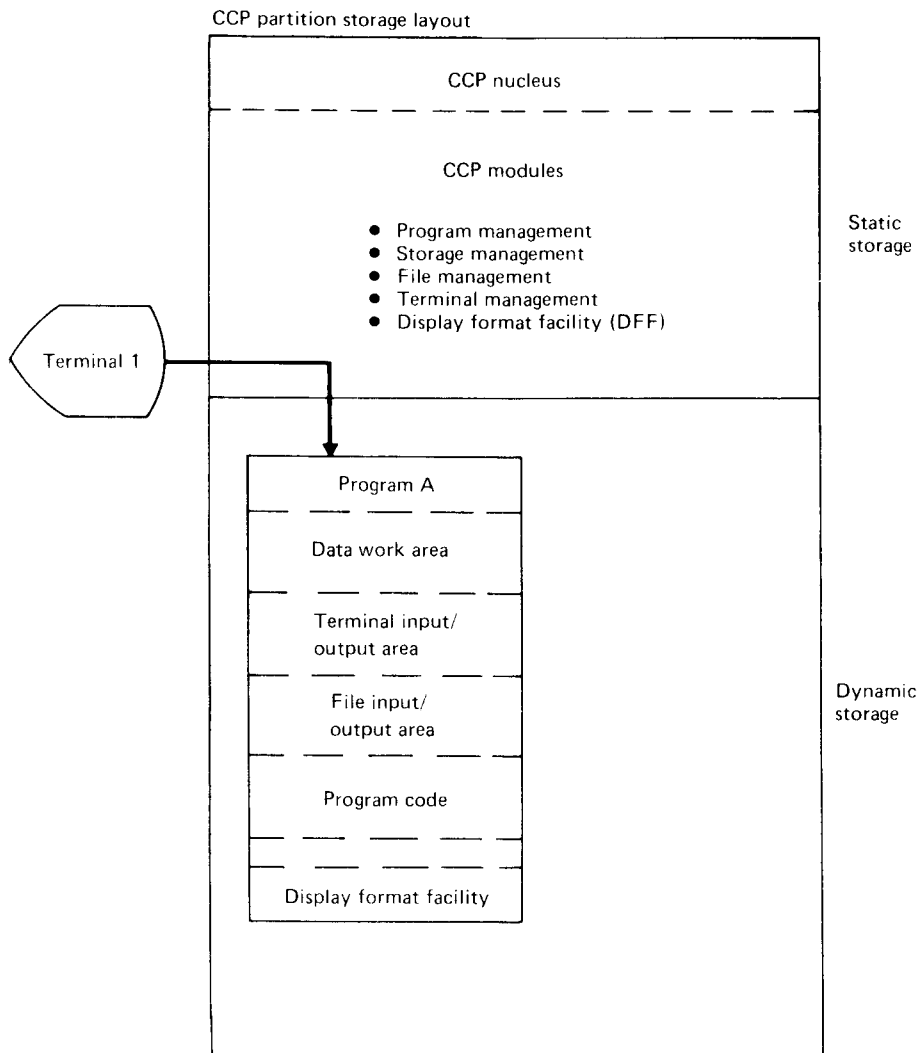


Figure 63. CCP simple inquiry program (SRT).

In Figure 64 note that:

- Program B is a CICS/VS program using Basic Mapping Support (BMS)
- CICS/VS dynamically acquires storage and creates a task control area on receiving a transaction from a terminal
- Program B consists of mainly executable code
- CICS/VS dynamically acquires storage for data work areas and terminal and file I/O areas, as needed, which are unique to the task
- Terminal 1 is under control of task 1.

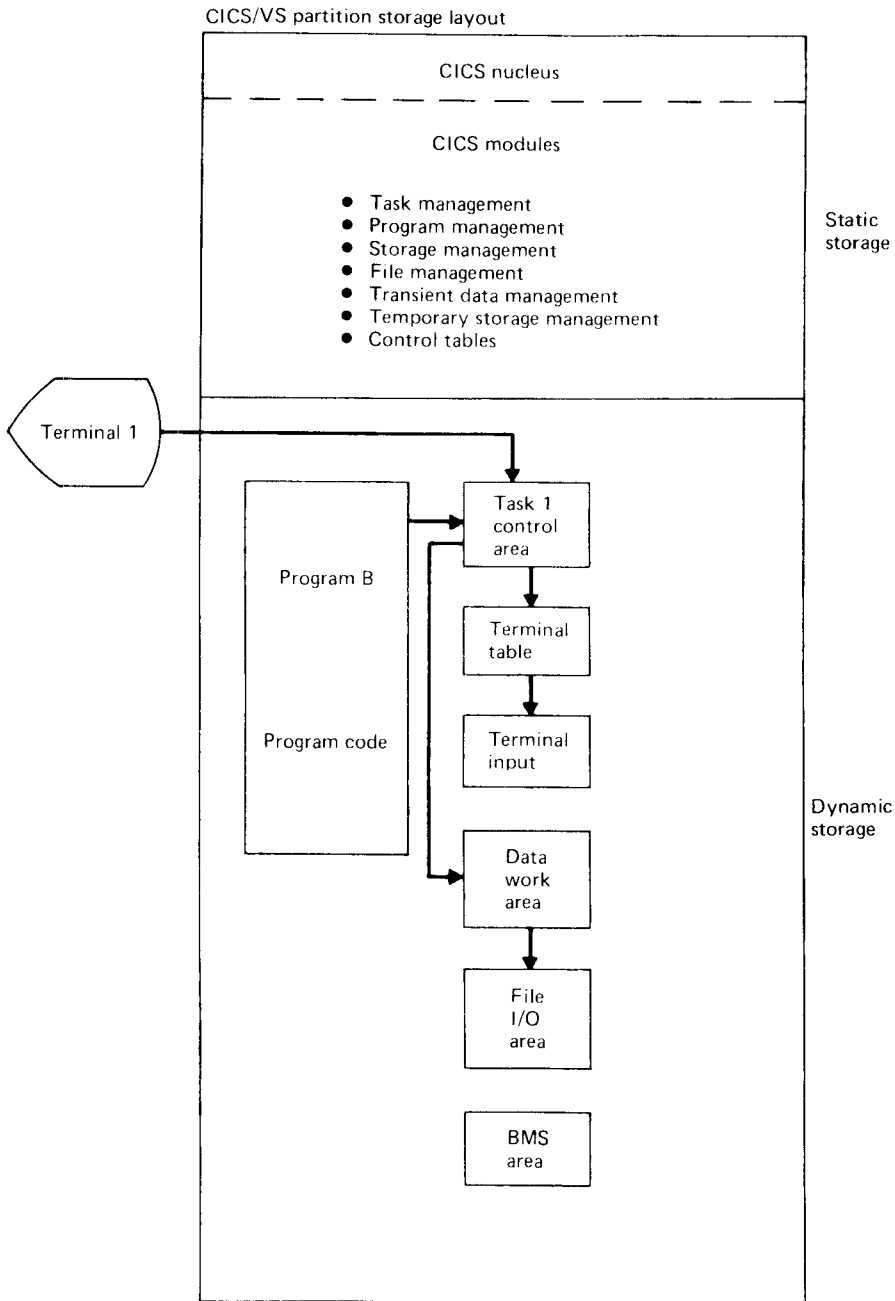


Figure 64. CICS/VS simple inquiry program (one task)

In figure 65 note that:

- Program C is a CCP/MRT program using DFF
- Data work areas for terminals 1 and 2 must be allocated and managed by the program
- Only one program/CCP interface exists and all transactions share the same terminal I/O area in the program. One transaction at a time can be processed by a particular program (single thread)
- Data in program work areas may be shared by transactions and terminals
- Terminals 1 and 2 are under control of program C

In Figure 66 note that:

- Program B is a CICS/VS program using EMS
- CICS/VS has created a task control area for each transaction being processed
- Each task has unique data work areas and I/O areas. The storage has been dynamically allocated to the task as required and in this case will be released at task termination.
- Multiple program/CICS/VS interfaces exist enabling multiple tasks to be executed concurrently (multithread)
- Data in task work areas cannot be shared. Facilities such as temporary storage or transient data can be used to enable data sharing.
- Terminal 1 is under control of task 1 and terminal 2 is under control of task 2.

In Figure 67 note that:

- Program B is a CICS/VS program using EMS
- Task 1 and task 2 are at different stages of transaction processing
- Task 1 has just been created to process a transaction from terminal 1 and is processing data in the terminal input area.
- Task 2 has processed the data in the terminal input area and released the storage. A data work area and file I/O area have been acquired by task 2 to continue processing the transaction.
- Terminals 1 and 2 are under control of tasks 1 and 2 respectively.

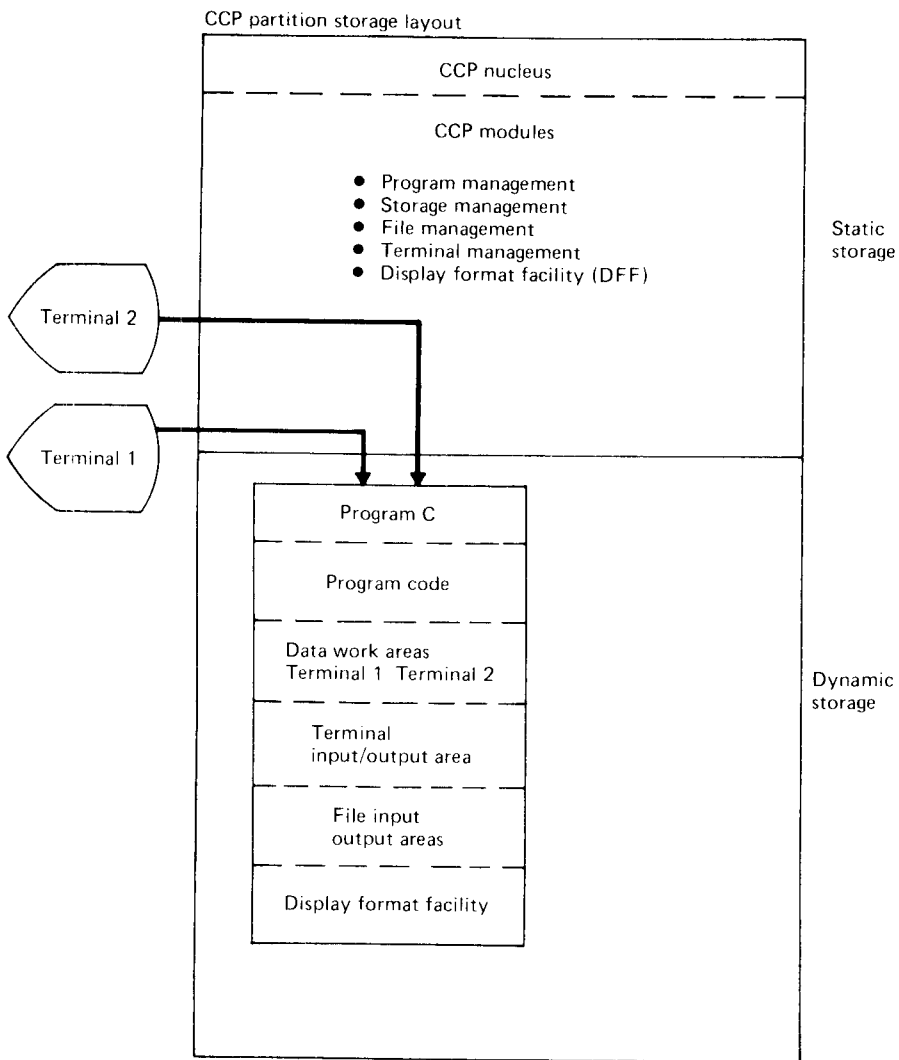


Figure 65. CCP simple inquiry program (MRT)

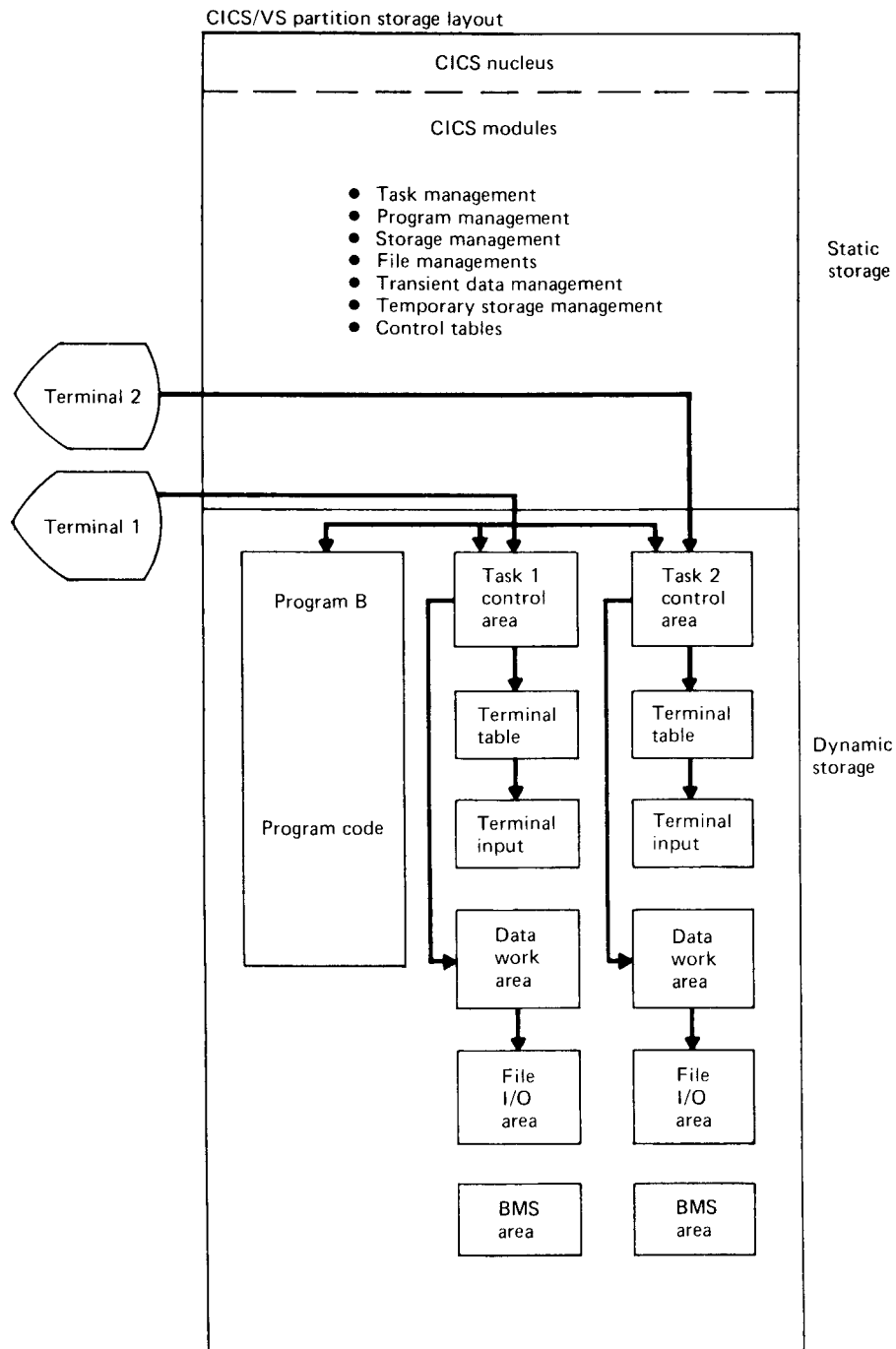


Figure 66. CICS/VS simple inquiry program (Two tasks)

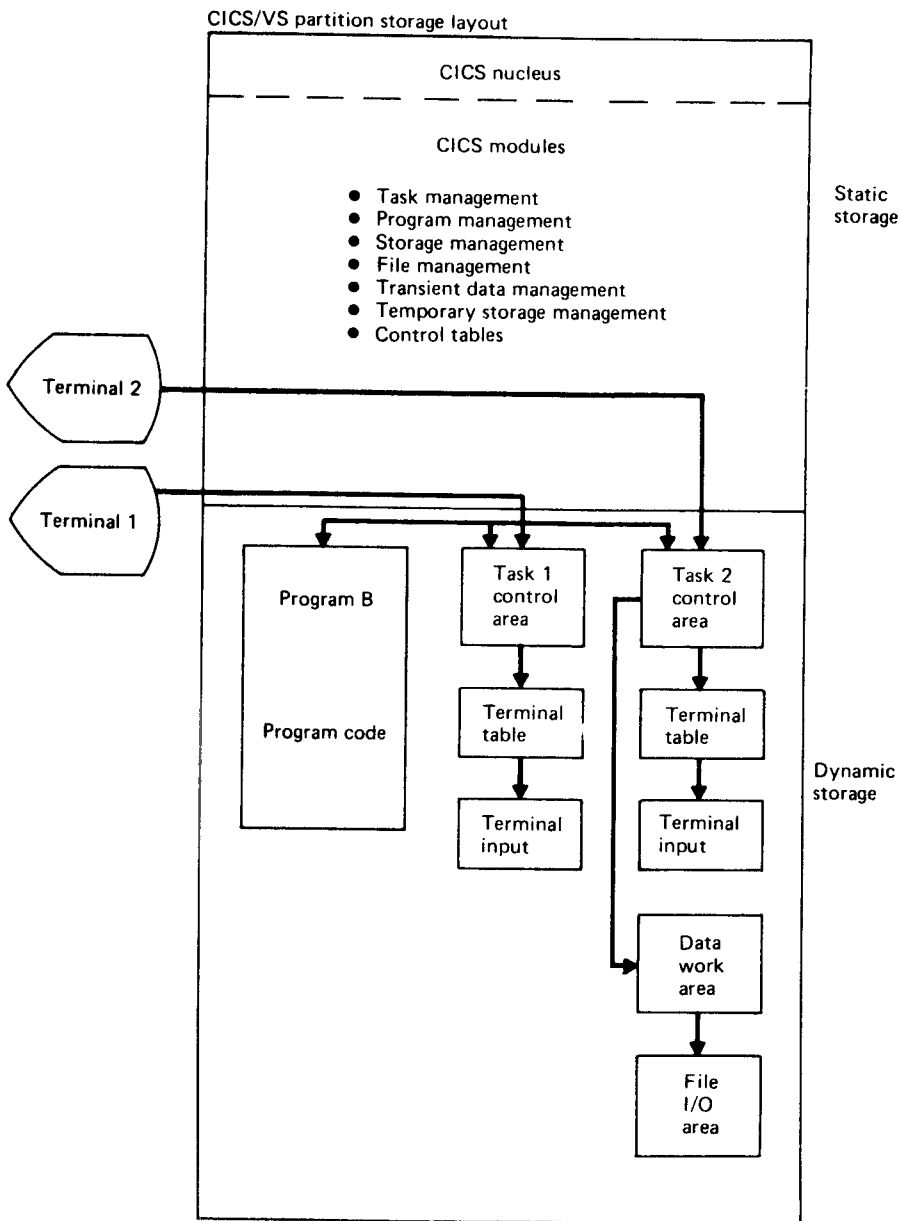


Figure 67. CICS/VS storage management of two tasks.

The software components of CICS/VS are listed below. Each CICS/VS components will be briefly described and then a summary for both CCP and CICS/VS presented. Further information on the CICS/VS software components is available in the CICS/VS General Information Manual (SC33-0066).

Only components whose functions are equivalent to those of CCP or may be required during conversion will be discussed in this chapter. CICS/VS offers many additional functions that should be considered for use in new online applications. Conversion from CCP to CICS/VS is considered as being carried out on an equivalent function basis, rather than a redesign, to take advantage of additional CICS/VS facilities (particularly recovery/restart, which is predominantly a user system design/program function under CCP).

CICS/VS SOFTWARE COMPONENTS

System management component:

- Task management
- Storage management
- Program management
- Time management
- Terminal management
- File management
- Transient data management
- Temporary storage management

System service component:

- Signon/signoff
- Master terminal
- Supervisor terminal
- Operator terminal
- System statistics
- Terminal test

Application service component:

- Basic Mapping Support
- EXEC interface program

System monitoring component:

- Trace management
- Dump management

System support component:

- System generation
- Environment definition
- System initialization

SYSTEM MANAGEMENT

TASK MANAGEMENT

There are some differences in software architecture and terminology between CCP and CICS/VS. To avoid confusion in later sections some key terms are defined below.

CCP: User task. Application code, such as an inquiry program, executing independently in the CCP program partition. Under Model 15 CCP, up to 15 user tasks can execute concurrently. Multiple terminals may use the same

user tasks concurrently.

CICS/VS: Transaction. Application code, such as an inventory update "transaction", that can be used by one or more terminal operators concurrently. The number of transactions that can operate concurrently is limited only by the available resources of a specific system.

CICS/VS: Task. The execution of a transaction for a particular terminal operator. A given task can relate to only one terminal operator. Multiple tasks may use the same transaction concurrently.

As you can see, a CCP "user task" and a CICS/VS "transaction" are related, whereas a CCP "user task" and a CICS/VS "task" have different meanings.

CCP controls a potential system overload/stall condition by providing a queuing facility when the program request is received from the terminal. The task management component of CICS/VS controls the resources allocated to a task. A potential overload/stall condition is controlled by queuing requests for resources as they are requested by tasks during the processing of a transaction.

Additional facilities provided by CICS/VS in this area are:

- More extensive priority scheduling for tasks
- Runaway task detection, which corrects a program loop condition
- Statistics on number of concurrent tasks.

Task Management Summary

- | CCP | CICS/VS |
|--|--|
| <ul style="list-style-type: none">• Program Responsibility to manage resources where multiple terminals are attached to the same program | <ul style="list-style-type: none">• Task management responsible for allocation of resources to multiple tasks using the same copy of a program |
| <ul style="list-style-type: none">• Multiple transactions using same program may not execute concurrently (single thread) | <ul style="list-style-type: none">• Multiple tasks using same program may execute concurrently (multi-thread) |
| <ul style="list-style-type: none">• Priority system | <ul style="list-style-type: none">• Priority scheduling system |
| <ul style="list-style-type: none">• Terminals may share data within the program | <ul style="list-style-type: none">• Each task has some unique data working areas not sharable between tasks. Other CICS/VS facilities such as transient data, temporary storage, or the common work area (CWA) are available to enable data sharing. |
| <ul style="list-style-type: none">• Program request queuing facility available during system overload | <ul style="list-style-type: none">• Task queuing facility available during system overload |
| <ul style="list-style-type: none">• Maximum 15 tasks | <ul style="list-style-type: none">• Maximum 999 tasks |
| <ul style="list-style-type: none">• Program must be designated MRT to allow multiple terminals to be controlled and code must be written to control multiple terminals | <ul style="list-style-type: none">• Ability to have multiple terminals (tasks) attached to the same program needs no special designation or program code |
| <ul style="list-style-type: none">• Program may acquire/release terminals | <ul style="list-style-type: none">• One CICS/VS task may create another task and pass data to the task created. A terminal may be under control of the created task. |

STORAGE MANAGEMENT

Figures 63 and 64 show storage layouts of a simple inquiry program under CCP and CICS/VS. As can be seen from the diagrams, storage is allocated within the CCP program for I/O areas and data working areas, while the CICS/VS task requests storage only as required.

Figure 67 shows the storage layout for two tasks where each is at a different stage of processing a transaction, and only the storage needed for the particular stage has been acquired and allocated to the program. This method of dynamic storage allocation allows better use of available storage and consequently more concurrent tasks than would be otherwise possible.

A part of CICS/VS dynamic storage is kept in reserve and used only to meet requests for storage when other storage is in active use. This reserve storage is called the storage cushion and is, in effect, a safety valve that enables CICS/VS to dynamically regulate the amount of processing it can support. CICS/VS stops inviting terminals to send transactions for processing until the currently executing tasks return sufficient storage to CICS/VS to reestablish the storage cushion.

Storage Management Summary

- | CCP | CICS/VS |
|--|--|
| <ul style="list-style-type: none">• Storage allocated by CCP to programs | <ul style="list-style-type: none">• Storage allocated by CICS/VS to tasks and programs |
| <ul style="list-style-type: none">• I/O areas for terminal I/O areas for files and working storage static in program | <ul style="list-style-type: none">• Dynamic I/O areas for terminal and files and working storage acquired dynamically and then released |
| <ul style="list-style-type: none">• Transaction queuing facility during system overload condition | <ul style="list-style-type: none">• Storage cushion concept to detect potential system overload condition |
| <ul style="list-style-type: none">• NEP program to save program load time | <ul style="list-style-type: none">• Programs may be preloaded into virtual storage |
| <ul style="list-style-type: none">• Queuing at program request when storage is allocated to program which contains I/O areas, etc. | <ul style="list-style-type: none">• Queuing for storage may occur during transaction processing |
| <ul style="list-style-type: none">• Number of times program requested is available | <ul style="list-style-type: none">• Comprehensive statistics to enable system tuning and better storage management |
| <ul style="list-style-type: none">• Storage occupied by program released at program termination | <ul style="list-style-type: none">• All storage acquired by a task is chained together and may be released by the task during transaction processing or automatically by CICS/VS at task termination |

PROGRAM MANAGEMENT

Program management services within CICS/VS are controlled by the program control program. Program control commands are used to load programs, link or transfer control to programs, and return control from a program. Single copies of application programs in dynamic storage are controlled to allow concurrent use by multiple tasks.

When a requested application program is already in dynamic storage, the control is transferred directly to that program. Each program location on a direct access volume and in storage is kept in the processing program table (PPT). The status of each program is maintained in the

PPT, particularly whether it is in dynamic storage and is active.

Application programs remain resident in dynamic storage when not being used unless there is an indication that storage resources have become overloaded. In this case programs not currently active are deleted from dynamic storage.

Application programs must be written so that they are serially reusable between the entry and exit points of the program (i.e., reentrant). Entry and exit points of an application program coincide with the use of CICS/VS commands. A serially reusable portion of an application program is executed by only one task at a time.

The user can use common work areas between entries and exits, and is required to use unique storage areas for only those data items that must be retained when the user program exits to CICS/VS.

Reentrant programs allow a single copy of an application program to be used to process several transactions concurrently.

Program Management Summary

- | CCP | CICS/VS |
|---|---|
| <ul style="list-style-type: none">• CCP load programs from an object library | <ul style="list-style-type: none">• CICS/VS loads programs from a core image library |
| <ul style="list-style-type: none">• Programs cannot be reentrant if written in RPG II | <ul style="list-style-type: none">• Programs must be reentrant so that multiple tasks can use the same copy of the program |
| <ul style="list-style-type: none">• Languages supported:
RPG II
COBOL
FORTRAN
Assembler | <ul style="list-style-type: none">• Languages supported:
RPG II
COBOL
PL/I
Assembler |
| <ul style="list-style-type: none">• One program may load another program using the task chain facility with data transfer | <ul style="list-style-type: none">• One program may execute the following functions in relation to another program:
LINK
RETURN
TRANSFER CONTROL (XCTL) |
| <ul style="list-style-type: none">• Program requests CCP functions by setting up parameters and branching to a communications sub-routine | <ul style="list-style-type: none">• Program uses CICS/VS commands |
| <ul style="list-style-type: none">• Run sorts under control of CCP | <ul style="list-style-type: none">• Not available |
| <ul style="list-style-type: none">• Maximum number of user programs is 256 | <ul style="list-style-type: none">• Maximum number of user programs is unlimited--based on resource availability |
| <ul style="list-style-type: none">• Display Format Facility (DFF) | <ul style="list-style-type: none">• Basic Mapping Support (BMS) |

TIME MANAGEMENT

The following services are performed by CICS/VS based on intervals of time specified by the user during system initialization.

- CICS/VS exit time interval control: the maximum interval of time for which CICS/VS wishes to release control to the operating system in the event there are no transactions ready to resume processing.

- System stall detection and correction: automatic detection when the CICS/VS dynamic storage resource becomes overloaded to the point where no active transactions can continue and no new transactions can be initiated. Corrective action involves the purging of transactions designated as purgeable by the user.
- Runaway task detection and correction: automatic detection when an application program develops a "loop" within the program logic. Corrective action involves the abnormal termination of the transaction.

The following services are performed by time management in response to a request by an application program or another CICS/VS function.

- SUSPEND permits a task to temporarily suspend itself for a given period of time.
- CANCEL allows a transaction to terminate its own or another transaction request for a SUSPEND.

The ability to retrieve the data and time of day is also provided.

Time Management Summary

- | CCP | CICS/VS |
|--|---|
| • Program may request time of day (Model 15) | • Program may request time of day |
| • Program may request WAIT function (Model 15) | • Program may request SUSPEND and CANCEL functions. |

TERMINAL MANAGEMENT

Terminal management provides for communications between terminals and user-written application programs through the terminal control program. The terminal control program uses data that describes the communication lines and terminals. This data is kept in the terminal control table (TCT). The terminal control program uses BASIC Telecommunications Access Method Extended Support (BTAM-ES) (program number 5746-RC5) to perform terminal input and output. When there is a need for a task to process the message, terminal control requests the creation of a task by task control.

Permanent transmission errors are handled by CICS/VS and additional action can be taken by a user program. The following sequence of events takes place when a permanent error occurs for a terminal.

- The terminal is placed out of service, i.e., no polling
- The terminal abnormal condition program is attached as a CICS/VS task
- Error data is written to a destination in transient data control defined by the user
- The terminal abnormal condition program then links to the user-written program (DFHTEP) for further appropriate action, which may include:
 - Terminal restored to in-service
 - Line placed out of service
 - Transaction (task) in process abnormally terminated.

The above error handling system means individual user programs need not have as many error routines for certain types of terminal errors.

Terminal Management Summary

- | CCP | CICS/VS |
|--|--|
| <ul style="list-style-type: none">• Access methods supported<ul style="list-style-type: none">- MLMP- MLTA• Terminals supported (a complete list is given in the CCP General Information Manual) | <ul style="list-style-type: none">• Equivalent access method<ul style="list-style-type: none">- BTAM-ES• Terminals supported (a complete list is given in the CICS/VS General Information Manual) |

FILE MANAGEMENT

File management consists of the file control program, the file control table, and access methods. Statistical data is maintained on file operations. The following access methods are recommended for use with CICS/VS since they support the use of a FBA DASD device:

- Virtual Storage Extended/Virtual Storage Access Method (VSE/VSAM)
- Sequential Access Method (SAM)

The additional access methods available to the user for non-FBA devices are:

- Indexed Sequential Access Method (ISAM)
- Direct Access Method (DAM)

Transient data uses the sequential access method (SAM) for extrapartition data sets.

File management provides the following basic services and features using VSE/VSAM, ISAM or DAM.

- Random record retrieval
- Random record update
- Random record addition
- Random record deletion (VSE/VSAM only)
- Sequential record retrieval
- Locate mode, read-only retrieval (VSE/VSAM only)
- Logical open/close of data sets
- Exclusive control of records during update operations

Additionally, CICS/VS provides increased function to the facilities provided by the system access methods.

- Segmented records
- Indirect accessing
- Mass record insertion (VSE/VSAM only)
- File sharing with update integrity within the CICS/VS partition
- Cross-partition file sharing with update integrity (VSE/VSAM only)

A description of these facilities is in the CICS/VS General Information Manual.

TRANSIENT DATA MANAGEMENT

Transient data management provides a generalized queuing facility where data can be queued for subsequent internal or offline processing. Selected data can be routed to or from predefined symbolic destinations either extrapartition or, optionally, intrapartition.

Intrapartition destinations are queues of data on direct access devices developed for input to one or more CICS/VS transactions. Data directed to or from these internal destinations is called intrapartition data and

consists of variable-length records only. Examples of data queued for intrapartition processing are:

- Transaction that require processes to be performed serially i.e., allocation of order numbers
- Accumulation of update data for a file to allow update in a particular sequence
- Batching of input data

Extrapartition destinations are data sets external to CICS/VS. Data directed to or from these external destinations can consist of sequential records that are fixed- or variable-length, blocked or unblocked.

Extrapartition destinations give CICS/VS a sequential file capability.

Data can be placed on an extrapartition data set by CICS/VS for subsequent input to CICS/VS or for offline processing.

When data is sent to an intrapartition destination and the number of entries in the queue reaches a predetermined level the user can optionally specify that a transaction be automatically initiated to process the data in the queue. This facility enables message switching to terminals to be implemented if required.

TEMPORARY STORAGE MANAGEMENT

Temporary storage management provides the services necessary for an application program to temporarily store data in dynamic storage or optionally on a direct access device. Data is stored and retrieved symbolically, thus facilitating data sharing among transactions.

The temporary storage control program provides temporary direct access storage that can be used to accumulate data during a transaction that has multiple inputs from the terminal. Although the data can be accumulated in dynamic storage, the user may prefer to place a mass of transaction data on a direct access device until the end of the transaction. The temporary storage control program puts the data on direct access storage using VSE/VSAM. Data can be retrieved in either a sequential or direct manner.

File/Data Storage Management Summary

- | CCP | CICS/VS |
|---|---|
| • Sequential, indexed, and direct files supported | • VSE/VSAM files supported. Sequential file facility available by using extra-partition transient data. Indexed Sequential (ISAM) and Direct Access (DAM) for non FBA DASD devices. |
| • Files may be shared between partitions with some restrictions on update | • Files may be shared between partitions if necessary with only one partition updating the file. (VSE/VSAM allows multiple partitions to update under certain conditions) |
| • Files defined in the application program same as batch | • Files defined in file control tables not in the application program |

SYSTEM SERVICE

Sign-on/Sign-off

The sign-on/sign-off function is optional and can be used to varying degrees. The CICS/VS sign-on transaction, CCSN, makes a security check on the name and password parameters supplied by the terminal operator. Both must be present, correctly entered, and in agreement with an existing entry in the sign-on table. If the verification is positive the operator conditions are set to indicate the operator is signed on. If the verification is negative a message is logged to indicate a possible security violation.

During sign-on a priority and security key are established. The operator's security key is used in a security check for all transactions subsequently entered.

Several options are also available for the CICS/VS sign-off transaction, CSSF.

Sign-on/Sign-off Summary

- | CCP | CICS/VS |
|---|--|
| <ul style="list-style-type: none">• /ON password
The sign-on procedure allows the operator access to the system | <ul style="list-style-type: none">• CCSN PS=password,NAME=name
The sign-on procedure establishes a security key that allows the operator to invoke only the appropriate transactions for which he is authorized. |
| <ul style="list-style-type: none">• /OFF HOLD
/OFF DROP
The HOLD option allows subsequent sign-on while DROP disconnects the terminal | <ul style="list-style-type: none">CSSF
CSSF GOODNIGHT
The GOODNIGHT option puts the terminal into RECEIVE ONLY status |

MASTER TERMINAL

Individuals within the customer organization can be defined in the sign-on tables as having authority to perform master terminal functions. The processor console may be used as the master terminal. A designated operator can sign on at any terminal, causing that terminal to have the capabilities of a master terminal. It is from the master terminal that the overall CICS/VS operation is controlled. The master terminal operator can dynamically vary system parameters as well as change the status of each line, terminal, and data set. All processing for master terminal operation is controlled by conversational terminal interaction.

Master terminal functions include the ability to:

- Alter transaction and/or terminal priority
- Disable and enable various table entries (PPT,PCT,FCT,DCT)
- Dynamically list active tasks (trace)
- Purge active tasks
- Dynamically open/close selected data sets
- Switch to an alternative dump data set

SUPERVISOR TERMINAL

Individuals within the organization can be defined in the sign-on table as having authorization to perform supervisory functions. A designated individual can sign on at any terminal, causing that terminal to be designated as a supervisory terminal. A supervisor can change the

service status and-or the processing status of his terminal or of any terminal under his supervision. A terminal can be places either in service or out of service. Its processing status can be such that it can initiate transactions and receive messages on request; receive messages automatically; receive messages only; or send messages to CICS/VS only.

Examples of operating procedures using the above facilities are contained in the CICS/VS Entry Level User' Guide.

An optional feature of CICS/VS is the support of the processor console as a CICS/VS terminal. This enables the user to run the master terminal program (CSMT) and other system administration tasks at his processor site.

OPERATOR TERMINAL

The status of a terminal may be changed by entering the appropriate transaction identification, depending on the terminal designation.

- Operator terminal - CSOT transaction
- Supervisor terminal - CSST transaction
- Master terminal - CSMT transaction

The system service program DFHCWTO with the associated transaction identification (CWTO) provides the terminal operator with the capability of sending messages to the processor console operator.

A CICS/VS transaction is initiated from a terminal by the operator entering a valid one- to four-character transaction code. Depending on the transaction, additional data may entered with the transaction code.

Operator Terminal Summary

- | CCP | CICS/VS |
|---|---|
| • Programs can be requested by the operator keying in the program name and, optionally input data | • Transactions can be initiated by the operator keying in the transaction code and, optionally input data |
| • /MSG MESSAGE sends messages to the system operator | • CWTO transacation sends a message to the console operator |
| • System operator console supported by CCP | • Processor console can be supported as a CICS/VS terminal |

System Statistics

System statistics are maintained by CICS/VS management programs during the execution of CICS/VS. These statistics can be displayed during the day in part or in their entirety at the request of any terminal operator with the appropriate security code. The statistics are printed automatically when the system is terminated.

The statistical information maintained by CICS/VS is detailed in the CICS/VS Operator's Guide (SC33-0080). CICS/VS operating statistics are a powerful tool for use in tuning the data communications system.

Terminal Test

Terminal test facilities can be provided by both CCP and CICS/VS. In both systems the terminal operator can initiate test functions. The CICS/VS test function is initiated by entering the transaction identification CSFE.

APPLICATION SERVICE

The CICS/VS facility that provides equivalent functions to that of the CCP display format facility (DFF) is basic mapping support (BMS). The description of BMS facilities is restricted to those equivalent to DFF.

BASIC MAPPING SUPPORT

The basic mapping support (BMS) function allows the CICS/VS application programmer to have access to input and output data streams without including device-dependent code in the CICS/VS application program.

Maps are assembled offline through use of CICS/VS macro instructions. The user defines and names fields and groups of fields that can be written to and received from the devices supported by BMS. The assembled maps contain all the device-dependent control characters necessary for the proper manipulation of the data stream.

Associated with each map is a table of field names that is copied into each application program that uses the map. Data is passed to and from the application program under these field names. The application program is written to manipulate the data under the various field names so that alteration of a map format does not necessarily lead to changes in program logic. New fields can be added to a map format without making it necessary to reprogram existing applications.

Output data may be supplied from the application program by placing the data in the table under the appropriate field name. As an alternative, output maps can contain field default data that is sent when data supplied by an application program is not present. This facility permits the specification of titles, headers, etc., for output maps.

Optionally, the displaying of all default data can be suppressed by the application program for any output map. Each time a map is used, the application program can temporarily modify the attributes of any named field in the output map. Output map fields with no field names can contain default data, but the application program cannot replace the default data or modify the attributes of unnamed fields.

For input, the user assembles a map defining the fields that can be written to and received from a particular device. Any data received for a particular field is moved across under the field name in the symbolic storage definition for the map. Pen-detectable fields defined in an input map are flagged as detected if present in a 3270 input stream. An input map for a particular case can specify a subset of the fields potentially receivable; any fields received and not presented in that map are discarded. This permits the number of keyable and selectable fields from a map to be changed without it being necessary to reprogram applications that currently receive data from the map.

Maps are stored in the CICS/VS program load library. When a map stored in the program load library is referenced by BMS, a copy is automatically retrieved by CICS/VS without application program action. Multiple users of a map contained in the program load library share a single copy in main storage.

BMS permits any valid combination of field attributes to be specified by the user when generating maps. Inclusion of BMS in CICS/VS is a system generation option.

BMS MAP GENERATION

This section describes how to prepare 3270 screen layouts for use by CICS. This is done with basic mapping support (BMS) macro instructions

which produce a map table and a map area. The use of the map table and map area will now be defined and the BMS map creation macro instructions explained.

MAP TABLE

A map table is cataloged in the core image library for each screen layout used. It is given a name that is referenced by the program whenever basic mapping support is invoked.

The map table contains the screen position of every field defined in the layout. When the ENTER key or PF key is pressed, fields having the modified data indicator on are sent to the computer. When the selector light pen causes transmission, the addresses of modified fields are sent to the computer. The screen position in the map table is used by CICS to identify the fields received. On output operations, the position is used to send fields to a specific location on the terminal. Field addressing permits widely separated fields to be transmitted with the smallest number of characters, thus speeding transmission.

For the 3270 the map table also contains the attribute character of each field. On output operations, the attribute character is sent to the screen, creating a field with specific characteristics. On input operations, the attribute character causes input data to be justified left or right and empty positions to be filled with blanks or zeros. Numeric-only attributes cause right justification and zero fill. All other attributes cause left justification and blank fill.

The map table may also contain initial data for selected fields. On output operation, titles, headings, and keyword fields are automatically sent to the terminal from the map table and not from the program.

The application program can modify the attribute characters and initial data as required, thereby permitting a single map to be used under different circumstances.

MAP AREA

A map area record description is created for each screen layout used by the program. The map area record description is stored in the source statement library and copied into the application program.

On 3270 input operations, BMS places the data from all modified fields in the map area. Unmodified fields contain X'00' characters. The application program retrieves the input data using the data field names in the map area record description.

On output operations, the application program places the output data in the data fields found in the map area record descriptions. Fields that are not to be sent are left as X'00' in the map area. BMS can combine all data placed in the map area with any initial data contained in the map table. If a field has data in both the map area and the map table, the map area data is chosen. The combined data is then written onto the terminal. The programmer also has the option of sending map area data only or map table data only to the terminal.

On 3270 output operations, the application program may place attribute characters in the attribute fields defined in the map area. These attribute characters are sent to the terminal in place of the attribute characters contained in the map table.

Each field in the map table corresponds to three entries in the map area record description: a data length, an attribute, and the data. A common field name is shared by all three entries with a 1-character suffix added to make the entry names unique. For example, a field named FLDX would have an entry FLDXL for the length, an entry FLDXA for the attribute, and an entry FLDXI for the input data. The output data would redefine the input entry with an entry FLDXO.

BMS Macro Instructions

Three BMS macro instructions are used to create the map table and map area record description: DFHMSD, DFHMDI, and DFHMDF. The DFHMSD macro instruction is always the first and last instruction describing a page or screen layout. The DFHMDI macro instruction is always the second macro instruction in describing a layout. DFHMDI is used to name the map and specify the page or screen size. Every field in the layout is defined with a DFHMDF macro instruction.

The BMS macros are coded in the following sequence:

```
mapset      DFHMSD
mapname     DFHMDI
            DFHMDF
            -----
            DFHMDF
            DFHMSD TYPE = FINAL
```

The format of the DFHMSD macro is illustrated below in Figure 68.

```
mapset      DFHMSD  TYPE=DSECT
              MAP
              FINAL
              MODE=IN
              OUT
              INOUT
              LANG=ASM
              COBOL
              PLI
              CTRL=(PRINT
                  L40,L64,L80,HONEOM
                  FREEKB
                  ALARM
                  FRSET)
```

Figure 68. Mapset definition

The meaning of the parameters is defined below.

mapset is the one-to seven-character name of this map set.

TYPE= indicates the generation function of the macro instruction

DSECT indicates that this is a symbolic description map generation run to create the list of field names to be copied into an application program.

MAP indicates that this is a physical map generation run to create the control information block used by BMS to perform mapping.

FINAL must be coded as part of the last macro instruction of a map definition.

MODE=

IN indicates an input map generation.

OUT indicates an output map generation

INOUT indicates that the map definition is to be

used for both input and output mapping operations.

LANG= specifies the language in which the application program referring to a symbolic description map is written and, hence, is applicable for only a DFHMSD TYPE=DSECT macro instruction

ASM indicates that the symbolic description map is to be referred to by an Assembler-language program.

PLI indicates that the symbolic description map is to be referred to by a PL/I program.

CTRL= is used to specify device characteristics related to terminals of the 3270 Information Display System.

PRINT must be specified if the printer is to be started; if omitted, the data is sent to the printer buffer but is not printed. This operand is ignored if the map set is used with 3275s without the Printer Adapter feature or with 3277s.

L40, L64, L80, HONEOM are mutually exclusive options that control the line length on the printer. L40, L64, and L80 force a carrier return/line feed after 40, 64 or 80 characters, respectively. HONEOM causes the printer to honor all new-line (NL) characters and the first end-of-message (EM) character that appear in displayable fields of the data stream. If the latter option is specified, the application program must insert the NL and EM characters into the data stream. If the NL character is omitted, a carrier return/line feed occurs at the physical end of the carriage. If the EM character is omitted, printing stops at the end of the 3270 buffer.

FREEKB specifies that the keyboard should be unlocked after this map is written out. If omitted, the keyboard remains locked; further data entry from the keyboard is inhibited until this status is changed.

ALARM activates the 3270 audible alarm feature.

FRSET indicates that the modified data tags (MDT's) of all fields currently in the 3270 buffer are to be reset to a not-modified condition (that is, field reset) before any map data is written to the buffer. This allows the DFHMDF ATTRB specification for the requested map to control the final status of any fields written or rewritten in response to a BMS command.

The format of the DFHMDF macro is illustrated below in Figure 69.

```
mapset DFHMDF SIZE=(line,column)
```

Figure 69. Map definition

The meaning of the parameters is defined below.

mapname is the one-to seven-character name of this map, to be specified in the map operand of any BMS command that refers to this map.

SIZE= gives the dimensions of a map in terms of length and width.

line is a value from 1 to 240, indicating the length

of a map as a number of lines.

column

is a value from 1 to 240, indicating the width of a map as a number of characters per line. Space for the attribute byte should be included in the column specification.

A DFHMDF macro instruction is specified for every field on the terminal. It is illustrated in Figure 70.

```
-----  
| fieldname  DFHMDF  POS=(row,column)  
|                                     LENGTH=field length  
|                                     ATTRB=(UNPROT,NUM,PROT,ASKIP  
|                                           NORM,BRT,DRK  
|                                           DET  
|                                           FSET  
|                                           IC)  
|                                     INITIAL='any user data'  
|-----
```

Figure 70. Field definition

The meaning of the parameters is defined below.

fieldname

Specify a one- to seven -character field name only if the programmer is to have access to the data in this field. BMS will add a suffix of L, A, I, and O, to identify length, attribute, input, and output fields, respectively. If the field name is omitted, no map area entry is provided for this field, and neither the field initial data nor the field attribute can be modified.

POS=

Specify the row and column on the terminal of the attribute character for this field, relative to row 1, column 1. Hard-copy terminals should specify the position preceding the first data character of a field.

LENGTH=

Specify the length of the data field, not including the attribute character. A field must not extend beyond the terminal limit as defined in the DFHMDF size operand. Length can range from one to 256.

ATTRB=

Specify the characteristics to be contained in the attribute character.

UNPROT - UNPROTECTED
NUM - NUMERIC ONLY
PROT - PROTECTED
ASKIP - AUTOSKIP
NORM - NORMAL INTENSITY
BRT - BRIGHT INTENSITY
DRK - DARK INTENSITY
DET - DETECTABLE (LIGHT PEN)
FSET - FIELD SET (MODIFIED DATA TAG ON)
IC - INSERT CURSOR INDICATOR

if the ATTRB operand is omitted, a normal-intensity AUTOSKIP attribute character is provided in the map table. If the ATTRB operand is coded, UNPROT and NORM are the defaults.

INITIAL=

Specify any constant data to be contained in the map table for this field. If the initial operand is omitted, no constant data for this field is provided from the map

table on output operations.

BMS SUMMARY

Display layouts usually require that the following design decisions be made:

- Number and size of headings and whether data in the heading may be altered by the program
- Number and size of data fields and the class and type of each data field
- Number and size of operator messages
- System/operator interaction and display modification using put overrides
- Cursor positioning after each system/operator interaction.

Display layout design with the associated program functional requirements (i.e., specification of put override facilities required) form a large part of the total design effort. As DFF and BMS offer essentially the same facilities, this design work and the display layout may be unchanged.

It is in the area of specification of the display layout that conversion effort is required. A comparison of these differences is given in the Conversion Considerations chapter, Figure 71.

- | DFF | BMS |
|--|---|
| • Display format specification sheet coded from layout | • BMS macros coded from display layout |
| • Different sheets used for display and printer specifications | • Same BMS macros but different operands of the macros used |

SYSTEM MONITORING

TRACE MANAGEMENT

The optional trace function provides a trace table containing entries that reflect the execution of various CICS/VS commands by application programs and CICS/VS management programs. The trace facility is especially useful in program testing and debugging. If generated, it can be turned on or off through the CICS/VS master terminal facility.

Additionally the trace entries can be written to a sequential disk file with information on task dispatching, initiation and termination to aid problem determination and tuning.

DUMP MANAGEMENT

The dump control program records dumps of transactions on sequential data sets on either magnetic tape or direct access storage. CICS/VS optionally provides the capability to open/close the active dump data set during CICS/VS execution. Two dump data sets can be defined so that the dump utility program can be printing one of the data sets in batch mode.

The printed output of the dump utility is formatted so that each area,

program, and table entry is identifiable.

SYSTEM MONITORING SUMMARY

- | CCP | CICS/VS |
|--|---|
| <ul style="list-style-type: none">• Trace facility provided by DSM by loading trace routine before CCP startup• An abnormal termination of a user program by CCP causes a dump of the supervisor and the CCP partition to be written to the \$CCPFILE• One dump data set permitted | <ul style="list-style-type: none">• A trace may be provided for the following:<ul style="list-style-type: none">A transactionSelected system componentsAll system components• Dump requests may be requested by an application program as a diagnostic facility. An abnormal termination of a user program will also cause a dump to be written to the dump data set.• Optionally two dump data sets can be defined |

SYSTEM SUPPORT

The system support component consists of several functions that are required to support the data communications system. Most of the support functions are performed offline.

CICS/VS consists of systems programs, system tables and user application programs. CICS/VS generation tailors the system programs and tables to a user's specific needs. The CICS/VS distribution material contains both source code and preassembled modules and tables. The source code can be cataloged into a source statement library and used to generate CICS/VS. The preassembled modules and tables can be cataloged into a core image library and executed immediately. This significantly reduces the time to become productive using CICS/VS.

Four of the activities necessary to provide a user with an operational data communications system are discussed in this section. The four activities are:

- Prepare the DOS/VSE environment
- Generate the CICS/VS system programs
- Generate the CICS/VS tables to define the environment
- Prepare a set of parameters for a particular start-up of CICS/VS

PREPARATION OF THE DOS/VSE ENVIRONMENT

CICS/VS requires certain functional capabilities, e.g., teleprocessing support, to be generated in the DOS/VSE supervisor. The DOS/VSE supervisor generation parameters must be reviewed so that the CICS/VS requirements are incorporated.

Consideration must also be given to other parameters during supervisor generation, e.g., the number of I/O devices depending on the terminal network. A summary of the DOS/VSE supervisor generation macros can be found in the CICS/VS Entry Level System User's Guide

Library sizes for CICS/VS programs and tables and user programs must also be estimated and established within the DOS/VSE system. Labels defining the libraries should be loaded to either the standard or partition standard label area.

GENERATION OF THE CICS/VS PROGRAMS

CICS/VS provides several executable management programs that can be selected and generated by the user to perform various functions. These management programs include:

- Task management
- Storage management
- Program management
- Time management
- Terminal management
- Trace management
- Dump management
- File management
- Transient data management
- Temporary storage management

Additionally a number of system service programs are provided by CICS/VS. These programs run as application programs under CICS/VS.

- Master terminal
- Supervisor terminal
- Operator terminal
- System statistics
- Abnormal condition handling
- Terminal abnormal condition handling
- System termination
- Time-of-day control
- Sign-on/sign-off
- Terminal test
- Basic mapping support

Macros are provided to enable easy generation of each of the above programs. Multiple copies of each program may be generated each with a unique suffix to provide a range of functions for different runs of CICS/VS.

User-written programs, e.g., Terminal Error Program (TEP), can be incorporated to handle specific installation requirements.

There are several approaches to preparing CICS/VS for use on your system. The method used depends on the functions and facilities that you require. The methods available are:

Entry Level System (ELS)

Starter System

Entry Level System (ELS)

The Entry Level System provides the first-time user with a pre-assembled CICS/VS system. The system is a subset of full CICS/VS and is designed for the user that requires a limited working set and has a low volume of concurrent transactions. CICS/VS application programs that are used with ELS are object compatible with full CICS/VS. When the ELS user migrates to full CICS/VS the system programs would need to be reassembled or appropriate system programs selected from the starter system. The various tables that are used by CICS/VS must be assembled whether ELS or full CICS/VS is used. The facilities of ELS are:

- Command Level Interface-CICS/VS accepts instructions written in a syntax similar to the language. (Support for RPG II, COBOL, ASSEMBLER and PL/I).
- Terminal Support - Basic Mapping support for Screen Layout and Input/Output to terminals, (3270 only).
- Storage Control - Facility for dynamically obtaining working storage.

- Dump Control - Provides dumps to aid program testing.
- Trace Control - Trace facility to aid in tracking program execution.
- File Control - Direct access processing (ISAM, VSE/VSAM).
- Transient data control - Sequential queueing of data on disk (Input and Output).
- Temporary storage control - Storage of data for later retrieval.
- Program control - Transfer of control between programs.
- Access to System Information - Use of CICS/VS storage areas to pass information between programs.
- Interval control - Initiating transactions according to time of day or elapsed time.

Starter System

The starter system provides at least one pre-generated version of all the system programs. In some cases there are several versions of each system program. The user may select those pre-assembled programs that fit his needs saving considerable assembly time. The balance of the system programs are assembled, along with the CICS/VS tables, to prepare the CICS/VS system for operation.

GENERATION OF CICS/VS TABLES

CICS is a table-driven system. Its operation is based upon the unique environment described by tables. All parameters regarding the terminals, files, programs, transactions and operator identification is contained in the tables. Examples of these tables are:

- DCT - destination control table
- FCT - file control table
- PCT - program control table
- PPT - processing program table
- SIT - system initialization table
- SNT - sign-on table
- TCT - terminal control table
- TEPT - terminal error processing table

A suffix system is also used with the table assembly and allows multiple copies of a table to be cataloged to a core image library. A general description of the CICS/VS tables is in the CICS/VS General Information Manual

PREPARATION OF STARTUP PARAMETERS

Before startup several activities must be performed

- Volume/space file planning for direct access devices for both CICS/VS files and user files
- Creation of file label sets for CICS/VS and user files

- Creation of CICS/VS startup jobstream

The startup jobstream may include override parameters that determine the particular program or table to be loaded for this execution run. This is done by specifying the required suffix for the program or table. Override parameters may also be used to change the following:

- Maximum number of tasks
- Maximum number of active tasks
- Maximum number of tasks within a class
- System partition exit time
- Runaway task time interval
- Stall time interval
- Storage cushion size
- Trace table size
- Message level

SYSTEM SUPPORT SUMMARY

- | CCP | CICS/VS |
|---|---|
| <ul style="list-style-type: none"> • High-level macros provided to generate CCP object code. Sample generation macros provided for user modification | <ul style="list-style-type: none"> • High-level macros provided to generate CICS/VS object code. Starter system available with macro facilities to generate individual components |
| <ul style="list-style-type: none"> • Assignment sets created to define parameters for a particular execution run | <ul style="list-style-type: none"> • Tables assembled to define parameters for a particular execution run |
| <ul style="list-style-type: none"> • Startup override facilities available to system operator | <ul style="list-style-type: none"> • Startup override facilities available in jobstream and also to system operator. Ability to specify different component programs as well as tables |

Conversion considerations are outlined at both the system and application level.

SYSTEM CONVERSION CONSIDERATIONS

The CICS/VS software components were used to summarize differences between the two systems in Chapter 16, CCP-CICS/VS Description and Summary. The same structure is used in this chapter to explain conversion considerations at the system level. Suggested techniques and possible solutions are described under the appropriate CICS/VS software component where necessary. Some of the software components by their very nature (e.g., system support) are not considered for conversion and are excluded from this chapter.

TASK MANAGEMENT

Since the CCP program is controlled by CCP in a somewhat similar manner to a CICS/VS task, the various types of CCP programs need to be considered during conversion.

CCP-MRT Program

- Evaluate data work areas within the program to determine data unique to a transaction from a particular terminal and data that may be shared by terminals. The sharable data is discussed under storage and file management.
- Redesign the program so that each terminal will have unique data work areas allocated during program execution. The techniques to do this depend on the programming language. The CICS/VS Entry Level System User's Guide(DOS/VS) illustrates the techniques for COBOL programs.
- Remove the terminal management logic and code required to manage multiple terminals from the program.
- Evaluate the requirement under CICS/VS to acquire terminals. If the requirement still exists then the START command should be used to initiate another task and any data necessary passed to the new task. The terminal acquired is under control of the new task and CICS/VS and is released by the new task normally at task termination.

CCP-SRT Program

- Evaluate the program to determine data work areas unique to each transaction from a particular terminal as all CICS/VS programs should have the capability of having multiple tasks attached to the same copy of the program.
- Terminal acquisition can be converted as explained previously for an MRT program.

STORAGE MANAGEMENT

- NEP programs should be made resident in virtual storage so that upon receipt of transaction a program load is not performed. DOS/VSE allocates real storage only to active programs, thus no real storage will be used if the NEP type program is not active.

PROGRAM MANAGEMENT

CICS/VS programs may be written using either macro level coding or command level coding. Macro level coding supports ASSEMBLER, COBOL and PL/I application programs, while Command Level Coding supports RPG II, ASSEMBLER, COBOL and PL/I. If the macro level coding is used, System/3 RPG II programs would have to be completely recoded into ASSEMBLER, COBOL or PL/I language. When command level coding is used the coding can be accomplished with RPG II. This allows you evaluate what portion of the System/3 RPG II program logic can be retained and combined with high level command coding. Command Level Coding allows the application programmer to request CICS/VS services by including CICS/VS commands at appropriate points in the program. The command level processor translates the commands into the appropriate language which can then be compiled and link edited.

System/3 RPG II programs could also be converted by the Display Management System/VS (DMS/VS), particularly if the program performs an inquiry or inquiry with update function. DMS II is described in the chapter System/3 to 4300 Conversion Alternatives. Another alternative available is to recode the System/3 RPG II programs in COBOL/VS or PL/I.

TERMINAL MANAGEMENT

Evaluate procedures for handling terminal errors in CCP programs and define functions required in the terminal error program.

FILE MANAGEMENT

Sequential Files

A sequential file facility is available in CICS/VS by using extrapartition transient data. Both input and output operations are supported. In addition the files may be blocked and may contain variable-length records. CCP sequential output files used for logging or storing data for batch processing can be converted to extrapartition transient data files.

Conversion to direct files should be considered where either an update or add function is required to a sequential file. The advantage of the direct file is in the area of recovery/restart.

Transient Data

Intrapartition transient data facilities are included in the Entry Level system but this facility should not be required during conversion of CCP to CICS/VS as no equivalent facility exists in CCP.

Temporary Storage

Use of temporary storage will depend on conversion techniques and application requirements. For example if the programs are coded in COBOL the use of temporary storage under the control of CICS/VS will depend on data areas being defined in the working storage or the linkage section.

The use of temporary storage directly by an application program should not be necessary during conversion.

A more detailed discussion of files and data areas is available under Application Program Conversion Considerations in this chapter.

CCP commands that may be entered by the terminal and system operator are discussed below. The most appropriate conversion technique will depend on the use of the CCP command within a particular application and customer environment.

/MSG terminal operator command

The CICS/VS equivalent is the CWTO transaction to send a message to the system operator.

MSG system operator command

A CICS/VS Entry Level system supports the processor console as a CICS/VS terminal. The user can run the CICS/VS master terminal control program (CSMT) and other system administration tasks at his processor site. The facilities provided are detailed in the CICS/VS Entry Level User's Guide. Using the processor console as a CICS/VS terminal provides similar facilities to the system operator as are available with CCP.

A user-written transaction could be provided to enable the system operator to send a message to a terminal operator. If more comprehensive message switching facilities were required, the CICS/VS message switching (MSMG) transaction could be added to the CICS/VS system. A simple user-written transaction will provide the equivalent facility of the MSG command available to the CCP system operator.

/NAME terminal operator command

The /NAME command has no direct equivalent in CICS/VS. The two major uses of this command are:

- For security reasons where the program checks the terminal name.
- To enable an operator to use an alternative terminal where the program expects a particular terminal name.

The security requirement can be implemented in CICS/VS using a combination of terminal operator name, password and security key in the sign-on table.

The second requirement is largely due to the MRT program, which is not necessary under the task-oriented structure of CICS/VS. During conversion of an MRT program the terminal management code that uses the terminal name as a key should be removed. In many cases under CICS/VS the application program need know only the terminal name for statistical purposes or information.

If the /NAME command or an equivalent facility is required by an application, two alternatives are available:

- Enter the terminal name as data at transaction initiation and modify the application program accordingly
- Request a user-written transaction that allows keying of the terminal name and the subsequent transaction code. This front-end transaction could then start the second transaction and pass the terminal name to it as data.

/FILE terminal operator command

The /FILE command has no direct equivalent in CICS/VS. The file specifications in CICS/VS are in the file control table and access from a program is by using a symbolic file name. Two methods of implementing an equivalent facility, if required, are:

- Enter the file name as data and modify the program accordingly
- Request a user-written transaction that allows keying of file name and subsequent transaction code for the /NAME command.

/Q terminal operator command

The /Q command has no direct equivalent in CICS/VS. At program request time if resources required by program are unavailable the request is queued by CCP. The unavailable resources would typically be main storage or terminals. Since virtual storage and task management remove the necessity for MRT-type programs and multiple copies of SRT-type programs, there is less need for such a command.

Resources are acquired as needed by a CICS/VS task. This means that in most cases the transaction can be started, but may be slightly delayed in execution if resources (e.g., storage) are temporarily unavailable. Thus the absence of a /Q command in CICS/VS should not present operational difficulties.

Data mode escape, /RUN, /RELEASE - terminal operator commands

The facilities available by the above commands are not directly available in CICS/VS. CICS/VS does not examine data input from a terminal for specified series of characters. To achieve an equivalent facility, if required, the test for the six characters would need to be coded into the application program and then another task initiated to perform the function required (i.e., send a message to the operator).

Application design should enable a terminal operator to cancel/terminate a transaction at several logical points during processing either by keying specified data or by depressing the CLEAR, PA, or PF keys. Such a terminal operator action under CICS/VS would thus be able to send messages to the system operator or initiate other transactions. The dynamic task creation and allocation of resources under CICS/VS with appropriate application design reduce the requirement for commands such as data mode escape, /RUN and /RELEASE.

APPLICATION SERVICE

The Display Format Facility (DFF) specifications for screen layouts should be converted to BMS specifications. Listed below are details of the DFF specifications and the equivalent BMS specification. These DFF/BMS specification differences are summarized in the following table and then illustrated in an example using a simple inquiry screen layout. Additional information on DFF is contained under "Application Program Conversion Considerations" in this chapter.

DFF

BMS

Display control format entries

- | | |
|--|--|
| • Display name | • Map name in DFHMDI macro |
| • Field name for initial cursor position | • Use of ATTRB=IC in DFHM4DF macro for appropriate field |

- Display size
- Clear before writing
- WCC (write control character)
- Use of SIZE=(line, column) in DFHMDDI macro
- Specified at execution time by the ERASE operand when sending a map to a terminal
- Use of CTRL=(FREEKB, ALARM, FRSET) in DFHMDD macro

DFD

BMS

Field definition form entries

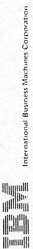
- | | |
|--|---|
| <ul style="list-style-type: none"> • Fieldname (1-6 chars.) • Field starting location (data) • Field length • Field class/type • Data source
G - generation defined
F - generation defined
E - execution time • Input, automatic skip • Data • CCP assignment values calculated and printed by generation routine • Printed output of input and output program specifications by generation routine | <ul style="list-style-type: none"> • Fieldname (1-7 chars.) in DFHMDD macro • Use of POS=(row, column) in DFHMDD macro for attribute byte • Use of LENGTH= in DFHMDD macro • Field class or type unnecessary as the ATTRB= parameter of DFHMDD allows specification of equivalent characteristics. As illustrated in Figure 71, the input/output type specification is at the map level in the DFHMDD macro. • For F, specification fields should not have a field name on DFHMDD macro. The E/G distinction is unnecessary as all named fields can be modified at execution time • Use of ATTRB = ASKIP in DFHMDD macro for appropriate field • Use of INITIAL=' in DFHMDD macro • Storage areas are dynamically obtained by CICS/VS • Use of TYPE=DSECT in DFHMDD to generate a list of field names for cataloging into the source statement library. The field names may be copied from the library into the application program during compilation |
|--|---|

Figure 71 illustrates equivalent DFD and BMS specifications for field characteristics. An example of both DFD and BMS specifications for a simple inquiry display is given in Figures 72 - 74.

		DFF Specification						BMS Specification
Field Class	Characteristics							DFHMDF macro ATTRB =
	Type	Designator	Protection	Data Type	Modified Data Tag	Intensity	Detection	
Output	1		PRO	A/N	OFF	NOR	NDT	(PROT, NORM)
	2		PRO	A/N	OFF	HI	DET	(PROT, BRT, DET)
	5		PRO	A/N	OFF	NDS	NDT	(PROT, DRK)
Input	1		UNP	A/N	OFF	NOR	NDT	(UNPROT, NORM)
	2		UNP	A/N	OFF	HI	DET	(UNPROT, BRT, DET)
	3		UNP	NUM	OFF	NOR	DET	(UNPROT, NUM, DET, NORM)
	4		UNP	NUM	OFF	HI	DET	(UNPROT, NUM, BRT, DET)
	5		PRO	A/N	OFF	NDS	NDT	(PROT, DRK)
	6		PRO	NUM	OFF	NDS	NDT	(PROT, NUM, DRK)
	7		UNP	A/N	OFF	NDS	NDT	(UNPROT, DRK)
	8		UNP	NUM	OFF	NDS	NDT	(UNPROT, NUM, DRK)
Output/Input	1		UNP	A/N	ON	NOR	NDT	(UNPROT, FSET, NORM)
	2		UNP	A/N	ON	HI	DET	(UNPROT, BRT, FSET, DET)
	3		UNP	NUM	ON	NOR	NDT	(UNPROT, NUM, FSET, NORM)
	4		UNP	NUM	ON	HI	DET	(UNPROT, NUM, BRT, FSET, DET)
	5		PRO	A/N	OFF	NDS	NDT	(PROT, DRK)
	6		PRO	NUM	OFF	NDS	NDT	(PROT, NUM, DRK)
	7		PRO	A/N	ON	NDS	NDT	(PROT, DRK, FSET)
	8		PRO	A/N	ON	NOR	NDT	(PROT, FSET, NORM)
Selector Pen Detectable (SPD)	1	?	PRO	A/N	OFF	NOR	DET	(PROT, DET, NORM)
	2	?	PRO	A/N	OFF	HI	DET	(PROT, BRT, DET)
	3	>	PRO	A/N	ON	NOR	DET	(PROT, FSET, DET, NORM)
	4	>	PRO	A/N	ON	HI	DET	(PROT, BRT, FSET, DET)
	5	?	PRO	A/N	OFF	NDS	NDT	(PROT, DRK)
	6	∕	PRO	A/N	OFF	NOR	DET	(PROT, DET, NORM)
	7	∕	PRO	A/N	OFF	HI	DET	(PROT, BRT, DET)

Figure 71. DFF and BMS specifications

Printer/Display Layout



International Business Machines Corporation

LINE	1-10	11-20	21-30	31-40	41-50	51-60	61-70	71-80	81-90	91-100	101-110	111-120	121-130	131-140
01														
02														
03														
04														
05														
06														
07														
08														
09														
10														
11														
12														
13														
14														
15														
16														
17														
18														
19														
20														
21														
22														
23														
24														
25														
26														
27														
28														
29														
30														
31														
32														
33														

CUSTOMER NO. ###
CUSTOMER NAME X
SHIP TO A
BALANCE XXX,XXX.XX
CREDIT LIMIT XXX,XXX.XX
PURCHASES
THIS MONTH XXX,XXX.XX
Y-T-D XXX,XXX.XX

AID	PF1	PF2	PF3	PF4	PF5
1	ENTER	CLEAR	RECALL	SELECTOR	PRINT
2	SEARCH	RECALL	SELECTOR	PRINT	W

AID	PF6	PF7	PF8	PF9	PF10
6	PF11	PF12	PA1	PA2	PA3
7	PF11	PF12	PA1	PA2	PA3
8	PF11	PF12	PA1	PA2	PA3
9	PF11	PF12	PA1	PA2	PA3
10	PF11	PF12	PA1	PA2	PA3

Figure 72. Inquiry Display



International Business Machines Corporation

DISPLAY FORMAT SPECIFICATIONS

GX21 9175-4 UM/050*
Printed in U.S.A.

Application: _____ Completed by: _____
 Purchases Instructions: _____
 Page of _____
 Date: _____

Field Type	Field Name	Field Starting Location	Field Length	Field Class	Field Description
C	Display Name	1	12	Text	Customer Name
C	Field Name for Invoice	13	12	Text	Customer Name
C	Customer Position	25	12	Text	Customer Position
C	Display Size (L)	37	12	Text	Customer Name
C	Screen Default (X)	49	12	Text	Customer Name
C	Printer Default (X)	61	12	Text	Customer Name
C	US World Trade Format (I)	73	12	Text	Customer Name
C	Day Source Print	85	12	Text	Customer Name
C	Day Source Format	97	12	Text	Customer Name
C	Clear Before Writing (Y/N)	109	12	Text	Customer Name
C	Printer Before Writing (Y/N)	121	12	Text	Customer Name

Field Type	Field Name	Field Starting Location	Field Length	Field Class	Field Description
F	HEADR1	1	12	Text	Customer Name
F	CUSNO	13	12	Text	Customer Number
F	HEADR2	25	12	Text	Customer Name
F	CUSNM	37	12	Text	Customer Name
F	HEADR3	49	12	Text	Customer Name
F	SHPNM	61	12	Text	Customer Name
F	HEADR4	73	12	Text	Customer Name
F	BALAN	85	12	Text	Customer Name
F	HEADR5	97	12	Text	Customer Name
F	CRLIM	109	12	Text	Customer Name
F	HEADR6	121	12	Text	Customer Name
F	HEADR7	133	12	Text	Customer Name
F	PURMO	145	12	Text	Customer Name
F	HEADR8	157	12	Text	Customer Name
F	PURTD	169	12	Text	Customer Name

Figure 73. DFF specifications coding

```

MAPSET    DFHMDS TYPE=MAP,MODE=INOUT,CTRL=(FREEKB,FRSET),LANG=COBOL
MAP001    DFHMDS SIZE=(24,80),TIOPFX=YES
          DFHMDF POS=(06,05),LENGTH=12,INITIAL='CUSTOMER NO.'
CUSNO     DFHMDF POS=(06,20),LENGTH=3,ATTRB=(PROT,BRT)
          DFHMDF POS=(07,05),LENGTH=13,INITIAL='CUSTOMER NAME'
CUSNM     DFHMDF POS=(07,20),LENGTH=22,ATTRB=(PROT,BRT)
          DFHMDF POS=(08,05),LENGTH=7,INITIAL='SHIP TO'
SHPNM     DFHMDF POS=(08,20),LENGTH=22,ATTRB=(PROT,BRT)
          DFHMDF POS=(11,05),LENGTH=7,INITIAL='BALANCE'
BALAN     DFHMDF POS=(11,20),LENGTH=10,ATTRB=(PROT,BRT)
          DFHMDF POS=(13,05),LENGTH=12,INITIAL='CREDIT LIMIT'
CRLIM     DFHMDF POS=(13,20),LENGTH=10,ATTRB=(PROT,BRT)
          DFHMDF POS=(16,05),LENGTH=9,INITIAL='PURCHASES'
          DFHMDF POS=(17,08),LENGTH=10,INITIAL='THIS MONTH'
PURMO     DFHMDF POS=(17,20),LENGTH=10,ATTRB=(PROT,BRT)
          DFHMDF POS=(18,08),LENGTH=5,INITIAL='Y-T-D'
PURTD     DFHMDF POS=(18,20),LENGTH=10,ATTRB=(PROT,BRT)
          DFHMDS TYPE=FINAL

```

Figure 74. BMS specification coding

APPLICATION PROGRAM CONVERSION CONSIDERATIONS

CICS/VS programs differ from batch programs in the following ways:

- The initial input is read by CICS/VS before the program is entered
- I/O and work areas may be defined outside the program and addressability must be established to such areas.
- Output consists of terminal responses instead of reports
- Data files are opened and closed by CICS/VS, not by the programs.

Command Level coding allows the application programmer to request CICS/VS functions by inserting the commands in his program. Command Level coding for COBOL, PL/I and ASSEMBLER is identical except for the delimiters used, while RPG II also uses a different syntax. In the following section on conversion the Command Level examples will concentrate on COBOL and RPG II.

Conversion considerations will be discussed under the following headings.

- Program interface to CICS/VS
- Terminal processing
- File processing
- Data areas
- Program design

PROGRAM INTERFACE TO CICS/VS

An application program requests CICS/VS functions by the use of CICS/VS commands. A list of the commands and examples of their use can be found in the CICS/VS Application Programmer's Reference Manual (RPG II) and for COBOL, CICS/VS Application Programmer's Reference Manual (Command Level).

These take the general form:

COBOL Example:

EXEC CICS function (options) END-EXEC

where "function" could be READ, WRITE, or any other CICS function.

The format of CICS/VS command is as follows:

1. The verb EXECUTE or its abbreviation EXEC
2. The identifier CICS
3. The function
4. A sequence of options
5. The statement terminator END-EXEC

RPG II Example:

Command level coding is entered on the RPG II calculation specifications sheet. The commands must be in specific columns as shown below.

*** CALCULATION SPECIFICATIONS ***							
Col		1	2	3	4	5	NOTES
6	7-8	8	8	3	3	6	
C	(blank) Ln SR	(func-name)	EXEC	(file-name)		(ind)	1
		(key-word)	ELEM	(literal-constant)	(v-name)		2

NOTES:

1. Col 6 - Always a C

COL 7-8 May be blank, a level number, or SR (subroutine)

Col 18-27 (func-name) - Always indicates the function to be performed such as READ, WRITE, SEND, etc.

Col 28-32 EXEC - Is required as the first statement.

Col 33-42 (file-name) - Optional parameter. Associated with the file defined in the RPG II file description.

Col 56-57 (ind) - The EXEC command requires an indicator. If none is specified indicator 13 is automatically assigned.

2. Col 18-27 (key-word) - Is the name of the option to be specified for the EXEC command such as data set, map, etc.

Col 28-32 ELEM - There may be multiple element statements depending on the EXEC command being performed.

Col 33-42 (literal-constant) - Optional parameter.

Col 43-48 (v-name) - Optional parameter. Denotes name of a variable that describes an RPG II field, array-element or data structure.

The RPG II examples that follow in this Chapter won't illustrate the coding in columns 6, 7, 8, and 56 of the Calculation

Command Translator

A program in which CICS/VS commands have been coded is processed by a command translator. The translator accepts the source program in which CICS/VS commands have been translated into CALL statements. At execution time the CALL statements invoke the EXEC interface program, which requests the function from the appropriate CICS/VS control program.

EXEC Interface Block

Each task in a command environment has a control block called the EXEC interface block (EIB) associated with it. Each application program has automatic access by name to the fields within the task's EIB. The EIB contains information that is useful during the execution of an application program, such as the transaction identifier, the time and data, and the cursor position on a display device.

Handling Exceptional Conditions

A number of exceptional conditions can occur during the execution of each CICS/VS command. Associated with each exceptional condition is a default action, which is the action that the system will take automatically if the application program does not specify a way to handle the condition. Usually the default action is to terminate the task abnormally. If the application program is to handle an exceptional condition, it issues a HANDLE CONDITION command, specifying the name of the condition and a program label to which control is to be transferred if the condition occurs, for example:

COBOL Example:

```
EXEC  CICS HANDLE CONDITION
      ENDFILE (END-ROUTINE)
      ERROR (TERMINATE)
      END-EXEC
```

RPG II Example:

```
*** CALC SPECS ***
COL  1      2      3
     8      8      3
     HANDLE   EXEC
     CONDITION ELEM
     ERROR    ELEM   TERMINATE
     ENDFILE  ELEM   ENDRT
```

The above example indicates that the application programmer will handle the ENDFILE condition specifically, and will deal with all other error conditions in a generalized way, with the exception of an I/O error, for which the system default action is to be taken.

Program Interface Summary

CCP

- Program interface usually by a SPECIAL file and operation codes in an array or record area
- Exceptional conditions cause return codes to be set in array and indicators set on

CICS

- Program interface by use of CICS/VS commands
- Exceptional conditions can result in system default action or a branch to a label in the program

- Program compiled in system set up specifically for CCP programs
- Program processed by command translator before compilation
- CCP returns information to program in an array and record area
- CICS/VS returns information to the program in the EXEC interface block

TERMINAL PROCESSING

Terminal Processing is carried out by the use of CICS/VS commands that invoke BMS facilities. Types of terminal processing are:

- Terminal input
- Terminal output - format screen
- Terminal output - modify screen
- Terminal output - display data
- 3270 Terminal - erase all unprotected
- 3270 terminal - print

TERMINAL PROCESSING EXAMPLES

Terminal Input

The BMS input function reads data, and/or converts data previously read by terminal control, from a terminal I/C area (TIOA) to a map area.

It removes 3270 data stream control characters, so that the program may conveniently operate on the data.

COBOL Example:

				NOTE

EXEC	CICS	RECEIVE	[MAP (map-name)]	1
			[INTO (data-area)]	
			[SET (pointer-ref)]	2

Notes:

1. RECEIVE
Specify RECEIVE to read data originating at the terminal and map it into the map area.

MAP (map-name)
Specify the name (1 to 7 characters) of the map table as defined in the processing program table.
2. INTO (data-area)
Specify the data area into which the mapped data is to be written. This option need not be specified if the map name is specified as a literal constant when the name of the data area is assumed to be the map name with the addition of the suffix "1".

SET (pointer-ref)
Specify the data pointer that is to be set to the address of the mapped data.

*** CALC SPECS ***				
COL1	2	3	4	NOTES:
8	8	3	3	
RECEIVE	EXEC			1
MAP	ELEM	'map-name'		2
INTO	ELEM		(data-area)	3
SET	ELEM		(pt-ref)	4

NOTES:

1. Specify that data is to be read from a terminal.
2. Specify the map name as a literal constant.
3. Specify if the map name was not specified. A data area name where the data is to be mapped may be coded.
4. Specify the set option if BMS should provide an area where the data is mapped and supply the address in the pointer reference.

Terminal Output - Format Screen

This BMS output function sends initial data from a map table to the terminal through terminal control. A 3270 screen may be erased before the data is displayed.

The command references a map table, which may contain attributes bytes, titles, and constants. No program modifications to the map table are permitted in this example.

COBOL Example:

				NOTES:

EXEC	CICS	SEND	MAPONLY	1
			MAP (map-name)	2
			[FREEKB] [FRSET] [ALARM]	3
			[CURSOR (data-value)]	4
			[ERASE]	5

NOTES:

1. SEND MAPONLY Specify to write the map constants to a terminal. The map constants are written unchanged.
2. MAP (map-name) Specify the name (1 to 7 characters) of the map table as defined in the processing program table.
3. Specify FREEKB to signal that the keyboard should be unlocked after the map is written.

Specify FRSET to indicate all modified data tags are to be reset before writing the map.

Specify ALARM to activate the audible alarm (special feature).

If these options are omitted, the options specified in the map table are used.

4. CURSOR (data-value) Specify the position of the cursor after the write. "data-value" is a halfword binary value in the range 0 to 1919. If the CURSOR operand is omitted, the cursor is positioned as defined in the map table.
5. ERASE Specify ERASE to erase the 3270 screen before data is displayed.

RPG II Example:

**** CALC SPECS ****					
Col	1	2	3	4	NOTES:
	8	8	3	3	
	SEND	EXEC			1
	MAPONLY	ELEM	'map-name'		2
	FREEKB	ELEM			3
	FRSET	ELEM			4
	ALARM	ELEM			5
	CURSOR	ELEM		(d-value)	6
	ERASE	ELEM			7

NOTES:

1. Indicates data to be sent to the terminal
2. Specifies that only the map constants are to be transmitted using the map name specified as a literal constant.
3. Specify to signal that the keyboard should be unlocked after the map is written.
4. Specify if all modified data tags are to be reset before writing the map.
5. Specify to activate the audible alarm (special feature)
6. Specify the position of the cursor after the write. If omitted, the cursor is positioned as defined in the map table.
7. Specify to erase the 3270 screen before data is displayed.

Terminal Output - Modify Screen

This BMS function modifies a portion of the data on a 3270 screen. This might be a message sent to the operator. BMS references a map table that sends only variable data from the map area.

COBOL Example:

	NOTES:

. (program instructions)	1
:	
:	
EXEC CICS SEND DATAONLY	2
MAP (map-name)	3
[FREEKB] [FRSET] [ALARM]	4
[CURSOR (data-value)]	5
[FROM (data-area)]	6

NOTES:

1. These instructions set up the data in the map area. Unused portions of the map area must be cleared to binary zeros. Attribute bytes may be modified by moving one of the following to an attribute field:

Symbolic Name	Field Attribute
DFHBMUNP	Unprotected
DFHBMUNN	Unprotected numeric only
DFHBMPRO	Protected
DFHBMASK	Autoskip
DFHBMERY	Bright intensity
DFHBM DAR	Dark intensity
DFHBMFSE	MDT on (field set)

DFHEMPRF	Protected and MDT on
DFHEMASF	Autoskip and MDT on
DFHEMASB	Autoskip and bright intensity

- SEND DATAONLY Specify to write from the map area to a terminal, only data supplied by the application program.
- MAP (map-name) Specify the name (1 to 7 characters) of the map table as defined in the processing program table.
- Specify FREEKB to signal that the keyboard should be unlocked after the map is written.

Specify FRSET to indicate all modified data tags are to be reset before writing the map.

Specify ALARM to activate the audible alarm.

If these options are omitted, the options specified in the map table are used.

- CURSOR (data-value) Specify the position of the cursor after the write. "data-value" is a halfword binary value in the range 0 to 1919. If the CURSOR operand is omitted, the cursor is positioned as defined in the map table.
- FROM (data-area) Specifies the data to be mapped. This option may be omitted if the map name is specified as a literal constant when the name of the data area is assumed to be the map name with the addition of the suffix "0".

RPG II Example:

Col	1	2	3	4	NOTES:
	8	8	3	3	
	SEND	EXEC			1
	DATAONLY	ELEM	'map name'		2
	FREEKB	ELEM			3
	FRSET	ELEM			4
	ALARM	ELEM			5
	CURSOR	ELEM		(d-value)	6
	FROM	ELEM		(d-area)	7

NOTES:

- Send indicates transmission of data to the terminal. Prior to this instruction the data should be set up in the map area. Unused portions of the map area must be cleared to binary zeros. Attribute bytes may be modified by moving one of the following to an attribute field.

Symbolic Name	Field Attribute
DFHEMUNP	Unprotected
DFHEMUNN	Unprotected numeric only
DFHEMPRO	Protected
DFHEMASK	Autoskip
DFHEMBRY	Bright intensity
DFHEMDAR	Dark intensity
DFHEMFSE	MDT on (field set)
DFHEMPRF	Protected and MDT on
DFHEMASF	Autoskip and MDT on
DFHEMASB	Autoskip and bright intensity

2. Specify for sending data only and indicate the map name as a literal constant.
3. Specify to signal that the keyboard should be unlocked after the map is written.
4. Specify to indicate that all modified data tags are to be reset before writing map.
5. Specify to activate audible alarm.
6. Specify the position of the cursor after the write.
7. Specify if the map name is not coded. Indicate the area where the data is to be mapped.

Terminal Output - Display Data

The BMS function erases the 3270 screen. Data placed in a map area by the application program and previously merged with initial data from a map table is sent to a terminal. This might be a map for a file record with field keywords from the map table and data from the file.

COBOL Example:

	Notes

. : (program instructions)	1
. EXEC CICS SEND MAP (map-name)	2
[FREEKB] [FRSET] [ALARM]	3
[CURSOR (data-value)]	4
[FROM (data-area)]	5

NOTES:

1. These instruction set up the data in the map area. Unused portions of the map area must be cleared to binary zeros. Attribute bytes may be modified by moving one of the following to an attribute field:

Symbolic Name	Field Attribute
DFHBMUNP	Unprotected
DFHBMUNN	Unprotected numeric only
DFHBMPRO	Protected
DFHBMASK	Autoskip
DFHMBRY	Bright intensity
DFHBM DAR	Dark intensity
DFHBMFSE	MDT on (field set)
DFHBM PRF	Protected and MDT on
DFHBMASF	Autoskip and MDT on
DFHBMASB	Autoskip and bright intensity

2. SEND
Specify SEND to write data and map to the terminal

MAP (map-name)
Specify the name (1 to 7 characters) of the map table as defined in the processing program table.
3. Specify FREEKB to signal that the keyboard should unlocked after the map is written.

Specify FRSET to indicate all modified data tags are to be reset before writing the map.

Specify ALARM to activate the audible alarm.

If these options are omitted, the options specified in the map table are used.

4. CURSOR (data-value)
Specify the position of the cursor after the write. "data-value" is a halfword binary value in the range 0 to 1919. If the CURSOR operand is omitted, the cursor is positioned as defined in the map table.
5. FROM (data-area)
Specifies the data to be mapped. This option may be omitted if the map name is specified as a literal constant when the name of the data area is assumed to be the map name with the addition of the suffix "0".

RPG II Example:

**** CALC SPECS ****						
Col	1	2	3	4	NOTES:	
	8	8	3	3		
	SEND	EXEC				1
	MAP	ELEM	'map name'			2
	FREEKB	ELEM				3
	FRSET	ELEM				4
	ALARM	ELEM				5
	CURSOR	ELEM		(d-value)		6
	FROM	ELEM		(d-area)		7

NOTES:

1. Send command indicates a transmission to the terminal. Prior to this instruction the unusual portions of the map area must be cleared to binary zeros. Attribute bytes may be modified by moving one of the following to an attribute field.

Symbolic Name	Field Attribute
DFHBMUNP	Unprotected
DFHBMUNN	Unprotected numeric only
DFHBMPRO	Protected
DFHBMASK	Autoskip
DFHBMERY	Bright intensity
DFHBM DAR	Dark intensity
DFHBMFSE	MDT on (field set)
DFHBM PRF	Protected and MDT on
DFHBMASF	Autoskip and MDT on
DFHBMASB	Autoskip and bright intensity

2. Specify the map name as a literal constant.
3. Specify to signal that the keyboard should be unlocked after the map is written.
4. Specify to indicate all modified data tags are to be reset before writing the map.
5. Specify to activate the audible alarm.
6. Specify a value to indicate the position of the cursor after the write.
7. Specify if the map name is not coded. Indicate the area where the data is to be mapped.

3270 Erase All Unprotected

The erase all unprotected function causes all unprotected fields on a 3270 screen to be erased, and the cursor to be positioned at the first unprotected field.

This may be useful in applications such as continuous data entry or add, where a terminal operator enters records on a formatted screen. A screen with protected keyword fields may be sent initially to a 3270 to guide the operator in entering each field. After reading in each record, the ISSUE ERASEAUP command can be issued to erase the entered fields, leaving the original screen with the protected keyword fields for entry of the next record. This function saves line time by eliminating the need for retransmission of the screen format.

COBOL Example:

```
EXEC CICS ISSUE ERASEAUP
```

RPG II Example:

```
**** CALC SPECS ****
| Col   1           2
|       8           8
|
|       ISSUE      EXEC
|       ERASEUP    ELEM
```

3270 Print

The print function prints the contents of a 3270 screen on the first eligible and available 3270 printer. A printer is considered eligible if:

- The printer is attached to the same control unit as the display
- The printer has a buffer size equal to or larger than the display
- The printer and the display are described in the same DFHTCT TYPE=GPENTRY instruction
- The printer is flagged as eligible by specifying: DFHTCT TYPE=GPENTRY,TRMFEB=(P)

A printer is considered available if it is not busy printing and it is operational (power on, paper loaded, etc.). If the screen cannot be printed immediately, the data is automatically queued in temporary storage until it is printed.

COBOL Example:

```
EXEC CICS ISSUE PRINT
```

RPG II Example:

```
*** CALC SPECS ***
| Col   1           2
|       8           8
|
|       ISSUE      EXEC
|       PRINT     ELEM
```

HANDLE AID command

CICS/VS provides two sets of constants that can be copied into a program. The first set is used to determine which 3270 attention key was pressed to initiate a transaction. The second set contains some commonly used 3270 attribute bytes that may be used to modify the attribute bytes on a screen.

The action taken by a program when an attention interrupt occurs is determined by a HANDLE AID command, which specifies a label to which control will pass after a program attention or function key has been pressed.

The following example shows a HANDLE AID request that sets up one label for the PA1 key AID and a second label for the PA2 and PA3 key AID's, all of the PF key AID's except PF10 and the CLEAR key AID.

COBOL Example:

```
EXEC CICS HANDLE AID
      PA1 (LAB1)
      ANYKEY (LAB2)
      PF10
      END-EXEC
```

RPG II Example:

**** CALC SPECS ****				
Col	1	2	3	4
	8	8	3	3
	HANDLE	EXEC		
	AID	ELEM		
	PA1	ELEM		LAB1
	ANYKEY	ELEM		LAB2
	PF10	ELEM		

RETURN command

A RETURN function in a program passes control to a program at the next higher logical level. With a single-program transaction (one that issued no LINK or XCTL) the RETURN signals the end of the task and control is passed to CICS/VS.

A transaction identifier can be specified so that the next input from the terminal indicates this as the next transaction and data can be passed to the called transaction. An example is shown below.

COBOL Example:

```
EXEC CICS RETURN TRANSID (TRANSACTION
                    IDENTIFIER)
                    COMMAREA (data-area)
                    LENGTH (data-value)
```

RPG II Example:

	****	CALC	SPECS	****	
Col	1		2	3	4
	8		8	3	3
	RETURN		EXEC		
	TRANSID		ELEM	'TRAN-ID'	(d-area)
	COMMAREA		ELEM		
	LENGTH		ELEM	(d-value)	

This facility to specify the transaction identifier for the next input from a terminal and pass data to the called transaction enables transaction type processing to be implemented easily. A CCP program and screen format using PRUF where the program name (2 to 6 character) is written to the screen to enable initiation of the next program can be converted without change to the screen format. The transaction identifier (1 to 4 characters) can be passed to CICS/VS by using the RETURN function and the 2-6-character program name in the data stream ignored on input.

The terminal error program incorporated in the CICS/VS system can perform certain types of error recovery for all terminals and programs. This reduces the need for error handling.

Data sent to a terminal printer is queued by CICS/VS should the printer be busy. The program thus does not have to wait and reissue the write to the printer.

TERMINAL PROCESSING SUMMARY

- | CCP | CICS/VS |
|--|--|
| <ul style="list-style-type: none">• Program uses invite input and accept input operations to receive terminal input• Program uses put message and program request under format put message operations to format screen• Program uses put overrides operation to modify screen. A list of names of the fields to be overridden and override information are placed in the record area. Attribute bytes may be changed with certain restrictions• Program sets up display format name in record area• WCC set up at format generation time | <ul style="list-style-type: none">• Terminal input read by CICS/VS terminal control program. Program uses RECEIVE function to map data into the map area• Program uses SEND function with option MAP ONLY to format screen. Cursor positioning and other options can be used at program execution• Program uses SEND function with option DATA ONLY to modify screen. Suffix field names are used by the program to set up the data. Attribute bytes may be changes as required.• Program issues CICS/VS command with mapname in MAP option• WCC set up at map generation time can be modified at execution time |

- | CCP | CICS/VS |
|---|---|
| <ul style="list-style-type: none">• Erase all unprotected function achieved by program using erase all unprotected operations | <ul style="list-style-type: none">• Erase all unprotected function achieved by program using ISSUE function with ERASEUP option |

- Copy operation transfers data from one 3270 terminal to another
- Symbolic terminal name available in array
- Data in display format supplied at generation time may not be modified at execution time
- AID byte available in input record area
- Format field names coded in program form format generation output
- Transaction processing (programs with only one TERMINAL INPUT AND output) implemented using PRUF (program request under format)
- Program function to handle terminal printer busy condition
- A 3270 screen may be printed by the program using ISSUE function with PRINT option
- Symbolic terminal name available in EXEC interface block
- All data in map may be modified if fieldname is used at generation time
- AID byte available in EIB to program
- Map field names copied into program from source statement library
- Transaction processing can be implemented using RETURN function to give next transaction identifier for a terminal to CICS/VS or by storing transaction identifier on the screen in a PRUF type technique
- CICS/VS function to queue data for terminal printer if printer is busy when program writes data to terminal printer

FILE PROCESSING

CCP application programs contain file descriptions and statements to issue I/O operations. For each CCP program the type of file processing required is also specified in the assignment set. Several restrictions apply within the CCP system as to the type of file processing one program may do in relation to another program for the same file. These restrictions are necessary to maintain file integrity within the CCP system.

CICS/VS application programs do not contain file descriptions, only file control commands. No specification is required for the processing required by a particular program. The file processing required within the CICS/VS system is specified by the file control program and the file control table (FCT). The CICS/VS implementation allows file integrity to be maintained almost without any restriction on the type of access a program may require. A CICS/VS program may issue file commands without considering file commands issued by other CICS/VS programs.

The file commands that may be issued to CICS/VS are:

READ	- read a record from a file
READ UPDATE	- read a record from a file for update
REWRITE	- write an update record for a file
WRITE	- write a record to be added to a file
UNLOCK	- release exclusive control initiate by a READ UPDATE

Exclusive control places all tasks with the exception of the first into a wait queue so that only one task at a time updates the record and returns it to the file before another task may access the record. The UNLOCK command enables exclusive control to be released where the program logic determines a REWRITE will be bypassed.

To avoid tying up storage and exclusive control of records unnecessarily and to prevent a lockout situation, some programming sequences of file

commands are preferable to others. For example sequence 1 below is better than sequence 2. In general a REWRITE should be issued as soon as possible after the READ UPDATE for a file.

Sequence 1

```
EXEC CICS READ      UPDATE  FILE (A)
EXEC CICS REWRITE   FILE (A)
EXEC CICS READ      UPDATE  FILE (B)
EXEC CICS REWRITE   FILE (B)
```

Sequence 2

```
EXEC CICS READ      UPDATE  FILE (B)
EXEC CICS READ      UPDATE  FILE (A)
EXEC CICS REWRITE   FILE (A)
EXEC CICS REWRITE   FILE (B)
```

File layouts should be cataloged into the source statement library and copied into CICS/VS application programs as an installation standard.

Examples of file commands are given in the CICS/VS Entry Level User's Guide (DOS/VS).

A task may gain exclusive control of a file by issuing an enqueue/dequeue request to CICS/VS. This type of resource control should be avoided because of possible performance implications.

CICS/VS supports the Indexed Sequential Access Method (ISAM), Direct Access Method (DAM) and Virtual Storage Access Method (VSE/VSAM) for non-FBA devices. Support for FBA devices is VSE/VSAM exclusively. Conversion considerations from System/3 to the 4300 Processor in relation to file organization is discussed in Chapter 5, DASD File Conversion.

CICS/VS also offers a sequential file capability through transient data. The file commands for transient data are:

```
READQ TD - read a record from a sequential file
WRITEQ TD - write a record to a sequential file
```

To assist recovery/restart, direct access files should be considered for some user log files where the file is reformatted and the next available location for storing data is under program control.

The file processing facilities provided by the CICS/VS system are determined by the options specified during generation of the file control program and the file control table. For each file the services that CICS/VS application programs may request is specified in the FCT. For all files the following services may be provided:

```
GET      - records will be read
BROWSE   - records will be sequentially retrieved
NEWREC   - records will be added
UPDATE   - records will be updated
```

Additionally the following services may be provided for VSE/VSAM files:

```
DELETE   - records will be deleted
SHARE    - records will be shared within the DOS/VSE system
```

As explained earlier data integrity is preserved within the CICS/VS system when programs perform updates or additions from files. All files may be shared within the DOS/VSE system where only one partition (either the CICS/VS partition or batch) requires the update facility. If both partitions require an update facility VSE/VSAM files must be used.

Two facilities exist within CICS/VS to support unit record files:

- Transient data - a transient data sequential file may

have a printer specified in the destination control table (DCT)

- Sequential terminals - sequential terminals specified in the terminal control table may have a card reader as input and a printer as output.

Printed output from the CICS/VS partition may be spooled using VSE/PCWER if required. Examples of file control tables and destination control tables can be found in the CICS/VS System Programmers Reference Manual.

File Processing Summary

CCP	CICS/VS
<ul style="list-style-type: none">• File description in application program• File services requested by I/O language statements• File organizations supported<ul style="list-style-type: none">- consecutive- indexed- direct• Restrictions for data access within CCP for application programs	<ul style="list-style-type: none">• No file description in application program• File services requested by CICS/VS file commands• File organizations supported<ul style="list-style-type: none">- SAM (transient data)- ISAM (non FBA DASD)- DAM (non FBA DASD)- VSE/VSAM• Restrictions do not apply to CICS/VS application programs

CCP	CICS/VS
<ul style="list-style-type: none">• //DISKFILE and //PROGRAM assignment control statements specify CCP file processing facilities• Card reader and printer files specified in file description within CCP program• File integrity for update implemented within CCP partition• CCP files may be shared with the other partition under certain conditions• A file may be specified as non-sharable for a particular program	<ul style="list-style-type: none">• Terminal control program and table (TCT) and destination control table (DCT) and optionally terminal control table (TCT) specify CICS/VS file processing facilities• Printer supported by both DCT and TCT. Card reader supported by TCT• File integrity implemented for update implemented within CICS/VS partition• CICS/VS files may be shared across partitions. For update from both partitions VSE/VSAM files must be used• Program may issue CICS/VS requests to enqueue/dequeue the file for exclusive control

DATA AREAS

A CICS/VS application program may use several types of data areas. The storage for these data areas may be within the program or in CICS/VS dynamic storage, depending on the source language and programming technique of the program. An ANS COBOL or RPG II program can typically provide storage for the following data areas:

- Working storage area (variable, flags)
- Terminal map areas

- DASD file areas
- Log file areas (transient data)

The following areas provided by CICS/VS are described in the linkage section of an ANS COBOL program. RPG II provides access to these areas by use of the ADDRESS EXEC command.

- Common work area (CWA)
- Transaction work area (TWA)
- Terminal control table user's area (TCTUA)

Additionally, temporary storage may be requested using the WRITEQ TS command to store a single record with a unique name. A READQ TS command can retrieve the stored records sequentially.

Selection of the appropriate data areas for a program will depend on the following considerations:

- Size of data to be stored
- Data to be shared by multiple tasks
- Duration of data area
 - CICS/VS duration
 - Program duration
 - Task duration

Installation standards are necessary to coordinate the use of the CICS/VS-defined data areas. CICS/VS System Programmers Reference Manual describes the various data areas available to an ANS COBOL program. The manual CICS/VS Entry Level User's Guide (DOS/VS) describes the data areas for RPG II and ANS COBOL.

Generally conversion of System/3 CCP programs to 4300 CICS/VS programs should not require use of the temporary storage facility. Redesign of programs to be transaction- rather than multiple-function oriented may require temporary storage but consideration should be given first to the use of the other CICS/VS-defined data areas.

Data Area Summary

CCP	CICS/VS
<ul style="list-style-type: none"> • Most data areas are contained within CCP application program 	<ul style="list-style-type: none"> • Data areas may be within application program or in CICS/VS dynamic storage depending on the use of the data, availability to other tasks, and required duration of the data
<ul style="list-style-type: none"> • All terminals of an MRT program may access program data areas 	<ul style="list-style-type: none"> • Selected data areas may be accessed by all terminals using the same copy of a program, e.g., CWA
<ul style="list-style-type: none"> • Use of data areas dependent on programming standards 	<ul style="list-style-type: none"> • Use of data areas dependent on installation CICS/VS standards and programming standards

PROGRAM DESIGN

Differences between the facilities provided by CCP and CICS/VS require some changes in program design. Only differences that affect conversion and have not been previously discussed are explained in this section.

Terminal Acquisition

CCP allows a program to acquire a terminal. The acquired terminal has access to data contained in the CCP program and data or a message may be

sent to the acquired terminal for information or operator action. CICS/VS offers an equivalent facility by allowing one task to start another task.

The START command is used to start a task at some specified time and, optionally, to pass data to that task. CICS/VS stores the data and queues the request until the specified time. Then, as soon as all necessary resources (e.g., the terminal) are available, the task is started. By specifying a time that has already elapsed, the task may be started immediately.

Data may also be passed between the task via temporary storage if required.

Disk Sort Program

A disk sort program is not available under CICS/VS. Two alternatives are available to provide an equivalent facility.

1. User-written program

A user-written sort program may be used to sort records for a particular application. The program control commands enable the user-written sort program to be executed as required by the application.

2. DOS/VSE Sort/Merge program

The input data to be sorted could be written to an extrapartition transient data file that can be dynamically closed by the master terminal operator. A batch partition of the DOS/VSE system could then be used to sort the records. After completion, the sorted output file could be opened as an input transient data file. The sorted records are thus available to a CICS/VS program.

Recovery/Restart

CCP does not provide recovery/restart facilities. The recovery/restart procedures depend on the application, file, and program design. During conversion equivalent CICS/VS functions should be chosen so that the same recovery/restart procedures can be implemented.

Program Design Summary

CCP	CICS/VS
<ul style="list-style-type: none">• A program can acquire a terminal	<ul style="list-style-type: none">• A task can start another task that requires a terminal as a resource
<ul style="list-style-type: none">• Disk sort program available	<ul style="list-style-type: none">• Equivalent facility can be provided by:<ul style="list-style-type: none">- User-written program- Use DOS/VSE sort program in a batch partition
<ul style="list-style-type: none">• Recovery/restart capability dependent on application and program design	<ul style="list-style-type: none">• Equivalent facilities should be chosen so that the same recovery/restart capability is available

CCP-CICS/VS CONVERSION ACTIVITIES

An education plan should be a part of the overall conversion plan. An important element of the education plan will be the CICS/VS education. A checklist of major activities to be carried out after CICS/VS education has been completed is given below. A detailed conversion plan showing actual conversion details should be developed from this summary.

- Review CCP applications and develop a list of significant CCP-CICS/VS differences that require conversion
- Convert DFF specifications to BMS macros
 - Generate BMS maps
- Review program design in the following areas
 - CCP operations
 - File operations
- Recode the relevant sections of the program to CICS/VS requirements with CICS/VS commands for terminal and file operations
- Review recovery/restart procedures
- Review security requirements
- Review CICS/VS facilities required to support the hardware and application programs
- Review method of CICS/VS generation to decide on method of generation.
- Prepare the DOS/VSE environment
 - DOS/VSE supervisor
 - Creation and allocation of libraries
- Generate the CICS/VS system
 - Generate CICS/VS programs
 - Generate CICS/VS tables
- Prepare the CICS operating environment
 - DASD volume planning
 - DASD file label sets
 - CICS/VS startup jobstream
 - CICS/VS startup override parameters
 - CICS/VS documentation
- Convert data files
- Test application programs
- Review terminal operator's guide
- Review system operator's guide
- Train terminal operators
- Install CICS/VS operational system

Terminal-oriented systems may be supported by several programs on both the System/3 and 4300 Processor. This section summarizes the 4300 alternatives available for the following System/3 programs:

- CCP and DATA/3
- RPG II Telecommunications Features
- MLMP/MLTA subroutines

CICS/VS is recommended as the data communications system for the 4300. CICS/VS allows you flexibility in the development of future online applications.

A brief description of the 4300 programs (i.e., ICCF, VSE/POWER with RJE, and BTAM-ES) is included in this section. Development, testing, and diagnostic aids are also included for CICS/VS.

The feasibility of a particular conversion alternative will depend on specific hardware and application requirements for the individual user.

Your IBM marketing representative should be consulted for the most recent information on new versions of program products and particularly newly released field developed programs to assist conversion from System/3 to 4300.

CONVERSION ALTERNATIVES SUMMARY

System/3	4300 Processor
<ul style="list-style-type: none">• CCP, DATA/3	<ul style="list-style-type: none">• CICS/VS and optionally DMS/VS
<ul style="list-style-type: none">• RPG II Telecommunications Feature	<ul style="list-style-type: none">• CICS/VS• VSE/POWER RJE
<ul style="list-style-type: none">• MLMP/MLTA subroutines	<ul style="list-style-type: none">• CICS/VS• Assembler subroutine using BTAM-ES macros

PROGRAM DESCRIPTIONS

The following paragraphs describe the above alternative programs.

VSE/POWER RJE

The VSE/POWER remote job entry feature (RJE), program number (5746-XE8), provides an efficient method to enter jobs from terminals into the system for execution and to obtain output centrally or at a workstation. VSE/POWER supports the following BSC terminals:

- 2770 Data Communication System
- 2780 Data Transmission Terminal
- 3741 Data Station (model 2)
- 3741 Programmable WORK STATION (model 4)
- 3770 Data Communication System (in BSC or SNA mode)
- 3780 Data Communications Terminal
- 3790 Communication System, (only SDLC)

Only BSC point-to-point connections are supported, except for the 3770. The 3770 can also be used by VSE/POWER RJE in SDLC mode.

The 3790 Data Communication System is supported only in SDLC mode by VSE/POWER RJE.

Basic Telecommunications Access Method Extended Support (BTAM-ES)

Program number 5746-RC5, may be used to design a dedicated telecommunications system within a single partition. The support of binary synchronous communications and a variety of start-stop devices gives BTAM-ES considerable flexibility and a wide range of applications.

A BTAM-ES subroutine linked into a DOS/VS RPG II application program can provide a conversion facility for both System/3 RPG II Telecommunications Feature and System/3 MLMP subroutines. A detailed check of hardware features required to be supported and line protocol requirements is necessary to evaluate the feasibility of a BTAM-ES subroutine as a conversion facility.

PROGRAM CONVERSION/DEVELOPMENT AIDS

VSE/Interactive Computing and Control Facility (VSE/ICCF) (Program Product 5746-TS1)

The VSE/ICCF System is an entry level, interactive system designed to extend the power of the computer to multiple terminal users concurrently. The functional capabilities brought to the terminal users can provide significant productivity benefits in many different user environments. For instance, a programmer may use the VSE/ICCF System, not only for program development, but for library maintenance and the creation of more complex job streams. Assembler, COBOL, PL/I, and RPG programs may be compiled and tested directly from the terminal with results received back at the terminal. A submit-to-batch capability is also provided for the execution of programs that do not lend themselves to execution in a time shared environment. The RPG II Source Entry Facility (RSEF) used in conjunction with ICCF provides an easy to use, interactive RPG II program entry and modification facility.

Display Management System/VS (DMS/VS) (Program Product 5746-XC2)

DMS/VS is a user-oriented set of preprogrammed functions to aid in the implementation of online 3270 applications under CICS/VS. Forms are provided that require the user only to fill in the blanks, to produce screens, file descriptions, data transfers, file search and message routine instructions to the system. DMS/VS also enables the user to provide for online inquiries, and paging, data entry, and file updates without being knowledgeable in CICS or 3270 functional characteristics.

TESTING AIDS

CICS/3270 Simulator (FDP 5798-AXC)

CICS/3270 Simulator permits testing of CICS/3270 application programs using a sequential terminal. As an extension to the CICS sequential test facility, it accepts simple statements describing the test data and builds 3270 input data streams including all necessary control characters. It also accepts 3270 output data streams, interprets all control information, and prints the stream as it would appear on a real 3270 terminal. Thus the effort of preparing test data and interpreting test results is significantly reduced.

CICS Online Test/Debug (IUP 5796-AEF)

CICS Online Test/Debug provides the programmer with the facilities to test and debug application programs or user files while CICS is operating by entering command statements via a 3270 terminal. In effect,

the program enables a programmer with access to a 3270 to "console debug" programs without interrupting the normal operation of the installation.

CICS Volume Test Facility (FDP 5798-CDJ)

CICS Volume Test Facility is designed to provide the user with the capability to stress the total teleprocessing system in a production-like environment. The need for volume testing when undertaking changes in terminals, in network configuration, in central hardware systems, in control programs, or in applications is addressed by this FDP.

ANALYSIS AIDS

CICS/3270 Simulator (FDP 5798-AXC) (see above description)

CICS Dynamic Map (FDP 5798-AXR)

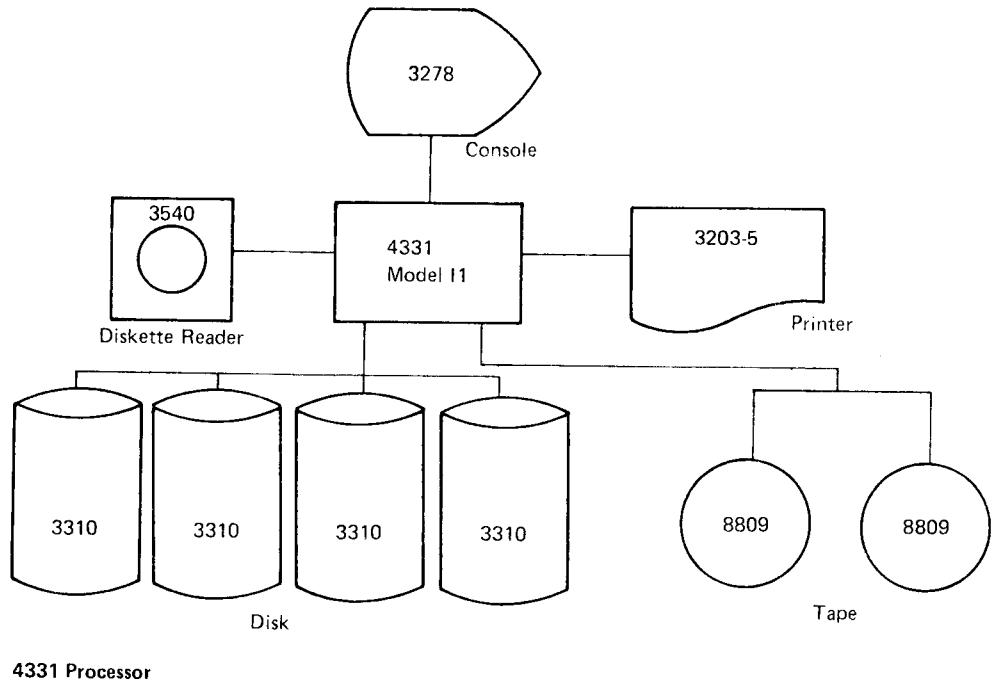
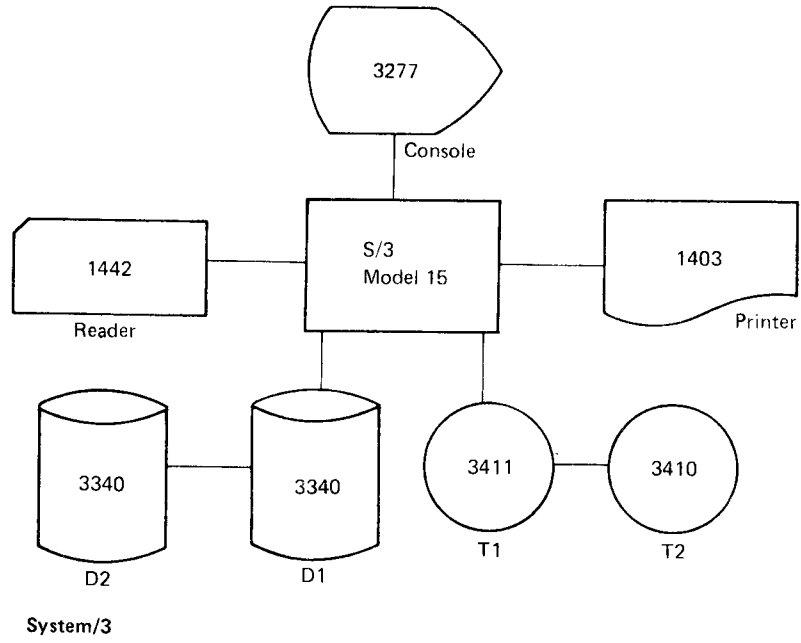
CICS Dynamic Map provides visibility into the realtime status and composition of an active CICS partition, by combining a statistic-gathering capability with its own output writer facility. By using CICS Dynamic Map the user can see how various changes in the dynamic free storage area can effect system performance and the utilization of the free memory storage area.

CICS Performance Analyzer (FDP 5798-AZN)

CICS Performance Analyzer provides the CICS user with the ability to collect and summarize selected information regarding resource utilization of the CICS system. It assists in identifying inefficient and/or heavily used applications.

CICS Performance Analyzer II (FDP 5798-CFP)

This program is an enhanced version of 5798-AZN above.



APPENDIX B DOS/VSE REFERENCE MANUALS

Subject	Title	Manual Number
<u>DOS/VSE</u>		
General Information	Introduction to DOS/VSE	GC33-5370
	DOS/VSE Entry User's Guide	GC33-6047
Data Set Organization	DOS/VSE Data Management Concepts	GC24-5138
Job Control and Linkage Editor	System Control Statements	GC33-5376
	Operating Procedures	GC33-5378
Librarian	DOS/VSE System Management Guide	GC33-5371
Messages	DOS/VSE Messages Manual	GC33-5379
Macros	DOS/VSE Macro User's Guide	GC24-5139
	DOS/VSE Macro Reference	GC24-5140
Operator Control	DOS/VSE Operating Procedures	GC33-5378
Labels	DOS/VSE Tape Labels	GC33-5374
	DOS/VSE DASD Labels	GC33-5375
System Generation	DOS/VSE System Generation	GC33-5377
	DOS/VSE System Management Guide	GC33-5371
Security	Data Security under DOS/VSE	GC33-6077
Assembler	OS/VS, DOS/VSE, VM/370 Assembler Guide to DOS/VSE assembler	GC33-4010 GC33-4024
Cardless Systems	Guide to Cardless Systems	GC20-1786
Utilities	DOS/VSE System Utilities	GC33-5381
Service-Debugging	DOS/VSE OLTEP	GC33-5383
	DOS/VSE Serviceability Aids and Debugging Procedures	GC33-5380
System History	Maintain System History (MSHP) User's Guide	GC33-6060
<u>DOS/VSE Program Products</u>		
VSE/POWER	General Information	GH12-5128
	Installation and Operations	SH12-5329
	VSE/POWER Messages	SH12-5520
180 System/3 to DOS/VSE Conversion Guide		

DOS/VS Sort Merge	General Information Installation Reference Manual Programmer's Guide	GC33-4030 SC33-4045 SC33-4044
DOS/VS COBOL	General Information ANS COBOL Installation Manual COBOL Programmer's Guide	GC28-6473 SC28-6479 SC28-6478
Fortran	FORTTRAN IV Library Option System 360/370 FORTTRAN IV FORTTRAN IV Programmer's Guide	SC28-6883 SC28-6515 SC28-6478
RPG II	General Information RPG II Language RPG II Messages RPG II Conversion Preprocessor	SC33-6030 SC33-6031 SC33-6033 SC33-6035
VSE/VSAM	General Information Programmer's Reference VSE/VSAM Messages	GC24-5143 GC24-5145 GC24-5146
CICS/VS	General Information CICS/VS Entry Level Guide Operator's Guide Application Reference (RPG II) Messages and Codes	GC33-0066 SC33-0086 SC33-0080 SC33-0085 SC33-0081
VSE/Advanced Functions	General Information System Information	GC33-6106 GC33-6107
Conversion	VSE/IBM System/3 - 3340 Data Import Installation Reference	SC33-6063

This form may be used to communicate your views about this publication. They will be sent to the author's department for whatever review and action, if any, is deemed appropriate. Comments may be written in your own language; use of English is not required.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation whatever. You may, of course, continue to use the information you supply.

Note: *Copies of IBM publications are not stocked at the location to which this form is addressed. Please direct any requests for copies of publications, or for assistance in using your IBM system, to your IBM representative or to the IBM branch office serving your locality.*

Possible topics for comment are:

Clarity Accuracy Completeness Organization Coding Retrieval Legibility

If you wish a reply, give your name and mailing address:

Note: Staples can cause problems with automated mail sorting equipment. Please use pressure sensitive or other gummed tape to seal this form.

What is your occupation? _____

Number of latest Newsletter associated with this publication: _____

Thank you for your cooperation. No postage stamp necessary if mailed in the U.S.A. (Elsewhere, an IBM office or representative will be happy to forward your comments or you may mail directly to the address in the Edition Notice on the back of the title page.)

Reader's Comment Form

Fold and tape

Please Do Not Staple

Fold and tape



NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES



BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 40 ARMONK, N.Y.

POSTAGE WILL BE PAID BY ADDRESSEE:

International Business Machines Corporation
Department 824
1133 Westchester Avenue
White Plains, New York 10604

Fold and tape

Please Do Not Staple

Fold and tape



International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, N.Y. 10604

IBM World Trade Americas/Far East Corporation
Town of Mount Pleasant, Route 9, North Tarrytown, N.Y., U.S.A. 10591

IBM World Trade Europe/Middle East/Africa Corporation
360 Hamilton Avenue, White Plains, N.Y., U.S.A. 10601

Cut or Fold Along This Line
System 390 to Disk Operating System/Virtual Storage Extended (File No. S370-34) Conversion Guide Printed in U.S.A. GC20-1792-1

This form may be used to communicate your views about this publication. They will be sent to the author's department for whatever review and action, if any, is deemed appropriate. Comments may be written in your own language; use of English is not required.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation whatever. You may, of course, continue to use the information you supply.

Note: *Copies of IBM publications are not stocked at the location to which this form is addressed. Please direct any requests for copies of publications, or for assistance in using your IBM system, to your IBM representative or to the IBM branch office serving your locality.*

Possible topics for comment are:

Clarity Accuracy Completeness Organization Coding Retrieval Legibility

If you wish a reply, give your name and mailing address:

What is your occupation? _____

Number of latest Newsletter associated with this publication: _____

Thank you for your cooperation. No postage stamp necessary if mailed in the U.S.A. (Elsewhere, an IBM office or representative will be happy to forward your comments or you may mail directly to the address in the Edition Notice on the back of the title page.)

Note: Staples can cause problems with automated mail sorting equipment.
Please use pressure sensitive or other gummed tape to seal this form.

Reader's Comment Form

Cut or Fold Along Line

System/3 to Disk Operating System/Virtual Storage Extended (File No. S370-34) Conversion Guide Printed in U.S.A. GC20-1792-1

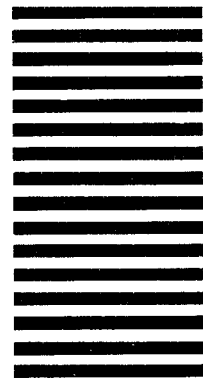
Fold and tape

Please Do Not Staple

Fold and tape



NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES



BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 40 ARMONK, N.Y.

POSTAGE WILL BE PAID BY ADDRESSEE:

International Business Machines Corporation
Department 824
1133 Westchester Avenue
White Plains, New York 10604

Fold and tape

Please Do Not Staple

Fold and tape



International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, N.Y. 10604

IBM World Trade Americas/Far East Corporation
Town of Mount Pleasant, Route 9, North Tarrytown, N.Y., U.S.A. 10591

IBM World Trade Europe/Middle East/Africa Corporation
360 Hamilton Avenue, White Plains, N.Y., U.S.A. 10601



International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, N.Y. 10604

IBM World Trade Americas/Far East Corporation
Town of Mount Pleasant, Route 9, North Tarrytown, N.Y., U.S.A. 10591

IBM World Trade Europe/Middle East/Africa Corporation
360 Hamilton Avenue, White Plains, N.Y., U.S.A. 10601