

AS/400 Advanced Series



System/36 Environment Programming

Version 3

AS/400 Advanced Series



System/36 Environment Programming

Version 3

Take Note!

Before using this information and the product it supports, be sure to read the general information under "Notices" on page xi.

First Edition (September 1995)

This edition applies to the licensed program IBM Operating System/400, (Program 5716-SS1), Version 3 Release 6 Modification 0, and to all subsequent releases and modifications until otherwise indicated in new editions. Make sure you are using the proper edition for the level of the product.

Order publications through your IBM representative or the IBM branch serving your locality. If you live in the United States, Puerto Rico, or Guam, you can order publications through the IBM Software Manufacturing Solutions at 800+879-2755. Publications are not stocked at the address given below.

A form for reader comments is provided at the back of this publication. If the form has been removed, you can mail your comments to:

| Attn Department 542
| IDCLERK
| IBM Corporation
| 3605 Highway 52 N
| Rochester, MN 55901-9986 USA

or you can fax your comments to:

United States and Canada: 800+937-3430
Other countries: (+1)+507+253-5192

If you have access to Internet, you can send your comments electronically to IDCLERK@RCHVMW2.VNET.IBM.COM; IBMMAIL, to [IBMMAIL\(USIB56RZ\)](mailto:IBMMAIL(USIB56RZ)).

When you send information to IBM, you grant IBM a non-exclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you or restricting your use of it.

© **Copyright International Business Machines Corporation 1995. All rights reserved.**

Note to U.S. Government Users — Documentation related to restricted rights — Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract with IBM Corp.

Contents

Notices	xi	Chapter 3. Configuring the System/36 Environment	3-1
Trademarks And Service Marks	xi	System Values Affecting the System/36 Environment	3-1
About System/36 Environment Programming (SC41-4730)	xiii	Starting Another AS/400 Subsystem	3-2
Who Should Use This Book	xiii	System/36 Environment Configuration	3-2
Chapter 1. Introduction	1-1	Commands for Configuring the System/36 Environment	3-2
System/36 Function in the System/36 Environment	1-1	AS/400 Device Identification	3-3
Operating in the System/36 Environment Configuration	1-1	AS/400 Device Configuration	3-3
Printed Output	1-1	Considerations for System/36 Environment Installation or PTF Application	3-4
File and Library Storage	1-2	Security Attributes for Multiple Requester Terminals (MRTs)	3-4
Libraries	1-2	Configuration of the System/36 Environment Change S/36 Environment Configuration Display	3-4
Files	1-2	Function keys	3-5
Folders	1-2	Change S/36 Environment Configuration Display	3-5
Diskette and Magnetic Tape Storage	1-2	Change S/36 Environment Attributes	3-12
Security	1-2	Display S/36 Configuration Display	3-13
Designing Records	1-2	Retrieve S/36 Environment Configuration Attributes	3-13
Communications	1-3	Work with System/36 Environment Configuration	3-13
Menus and Displays	1-3	Removing Display IDs	3-14
Messages and Message Members	1-3	Chapter 4. Printed Output	4-1
Programs and Procedures	1-3	Creating and Controlling Printed Output	4-1
Mixing System/36 Environment and AS/400 Functions	1-3	Printer Data Management Output	4-1
Jobs and Job Processing	1-3	System List Output	4-3
Error Prevention, Detection, and Recovery	1-4	Print Spooling	4-3
National Language Support	1-4	Using Output Queues	4-3
Licensed Programs	1-4	Changing the Output Queue for a Job	4-4
Utilities and Application Development Tools	1-4	Controlling Print Spooling	4-4
Query	1-4	Spool Writer Messages	4-4
OfficeVision for OS/400	1-4	Printer Control Guidelines	4-4
Client Access/400	1-5	Changing the Session Printer	4-4
Programming Languages	1-5	Changing the Print Key Printer	4-5
Chapter 2. Operating in the System/36 Environment	2-1	Changing the System List Device	4-5
System/36 Environment User Profile Attribute	2-1	Changing the System Printer	4-5
Commands to Access the System/36 Environment Functions	2-2	Changing the Printer Configuration Information	4-5
Accessing System/36 Environment Functions from Batch Jobs	2-2	Changing Printer Information in a Procedure	4-5
Subconsoles	2-3	Controlling or Displaying Print Spooling Information	4-5
System Request Menu	2-4	Copying and Displaying Output from an Output Queue	4-6
Attention Key	2-5		
OS/400 CL Commands for the System/36 Environment	2-5		

Printing Output by Forms Number	4-7	Listing Members and Library Information	6-6
Combining Several Print Files in One Job	4-7	Saving and Restoring Libraries	6-6
Assigning the Delayed Status to Printed Output	4-8	Copying Libraries and Library Members	6-7
Assigning Priorities to Printed Output	4-8	Securing Libraries	6-7
Programming Considerations	4-8	Listing Files	6-7
Use of Print Files by the System/36 Environment	4-9	Renaming Libraries or Library Members	6-7
Printed Output Attributes	4-9	Removing Libraries or Library Members	6-7
Chapter 5. Library, File, and Folder Overview	5-1	Coexistence Considerations	6-7
Comparison of System/36 and AS/400 Addressing Models	5-1	Library Lists	6-7
System/36 Addressing Model	5-1	Search Order	6-9
AS/400 Addressing Model	5-2	User Auxiliary Storage Pools for the System/36 Environment	6-10
System Information	5-3	Moving from System/36 to the System/36 Environment	6-10
Library QSSP	5-3	Chapter 7. Files	7-1
System Library (#LIBRARY)	5-4	Using Files	7-1
User Information Stored on Disk	5-4	Creating Files in the System/36 Environment	7-1
Output Queues	5-4	Naming a Physical File	7-1
Job Queue	5-4	Specifying a File in a Program	7-3
Journal Files	5-4	Placing Data in Files	7-3
Licensed Program Libraries	5-4	Removing a File from Disk or Diskette	7-3
User Files	5-4	Securing Files	7-3
User Libraries	5-4	Copying Files	7-3
User Folders	5-4	Printing or Displaying Files	7-4
Naming Conventions for Files, Libraries, and Folders	5-4	Store Deleted Files in Cache	7-4
Dynamically Created Files	5-5	File Organization	7-5
Programming Considerations	5-5	Sequential File Organization	7-5
Listing the Disk Volume Table of Contents	5-5	Direct File Organization	7-7
Measuring Disk Activity	5-5	Indexed File Organization	7-8
Chapter 6. Libraries	6-1	Multiple Indexes for a File	7-9
Libraries for the System/36 Environment	6-1	Processing Files	7-12
Library Names	6-1	Current Record Pointer	7-12
Group Libraries	6-1	Nonkeyed and Keyed Processing	7-13
Library Members	6-2	File Processing Methods	7-13
Library Member Names	6-2	Choosing a File Organization	7-19
Using Libraries	6-2	File Use	7-19
Assigning Libraries	6-3	Activity of the File	7-20
Sharing Libraries	6-3	Disk Space	7-21
Changing Libraries in a Job	6-4	File Attributes	7-21
Specifying Authority for Libraries	6-4	Scratch Files	7-21
Making Backup Copies and Recovering from Errors	6-4	Job Files	7-21
Recovering from Damage to #LIBRARY	6-4	Resident Files	7-21
Recovering from Damage to Library QSSP	6-5	Extendable Files	7-23
Library Sector-Mode and Record-Mode Files	6-5	Delete-Capable Files	7-24
Programming Guidelines for Libraries	6-6	Blocking Records	7-25
Creating Libraries	6-6	Sharing Files	7-26
Creating Library Members	6-6	File Sharing Considerations	7-26
		Levels of File Sharing	7-26
		Waiting for Files to Become Available	7-27
		Record Protection	7-28
		Releasing Locked Records	7-29
		File Deadlock Conditions	7-29

File Change Errors	7-30	I-Data Exchange Format	9-2
Using Multiple Names to Access a Single File	7-30	Storing Information on Diskette	9-2
Programming Considerations	7-31	Types of Diskette Files	9-2
Using System/36 Environment Files Library	7-31	Diskette Data Compression	9-3
Using the Library List Support for Files in the System/36 Environment	7-31	Diskette File Expiration Dates	9-3
Using System/36 Environment Files and AS/400 Files	7-33	Programming Considerations	9-4
Using File Members and Date-Differentiated Files	7-34	Preparing Diskettes	9-4
Using Override Database File CL Command	7-35	Copying, Saving, and Restoring Information	9-4
Extending Files	7-35	Listing Information from Diskette	9-5
Shared Files and System/36 Environment Share Levels	7-36	Removing Information from Diskette	9-6
Non-System/36 Environment Programs in the System/36 Environment	7-37	Allocating the Diskette Drive to a Job	9-6
Shared File Opens within the Same Job	7-37	Coexistence Considerations	9-6
Duplicate Keys and Key Sorting	7-38	Restoring the AS/400 System to System/36	9-7
Remote Files	7-38	Restoring System/36 to the AS/400 System	9-7
Moving from System/36 to the System/36 Environment	7-38	Restoring the AS/400 System to the System/36 Environment	9-7
Chapter 8. Folders and Data Dictionaries	8-1	Moving from System/36 to the System/36 Environment	9-8
Migration Considerations	8-1	Chapter 10. Magnetic Tape Storage	10-1
Using Folders	8-1	Tape Drives Supported	10-1
Folders and Folder Members	8-1	Tape Formats	10-1
Securing Folders	8-1	IBM Standard Label	10-1
Creating a Folder	8-1	Nonlabeled	10-2
Accessing a Folder	8-1	Tape Files	10-2
Listing Folder Information	8-1	Exchanging Tape Files with Other Systems	10-3
Deleting a Folder	8-1	Tape File Expiration Dates	10-3
Renaming a Folder	8-1	Tape Security	10-4
Reorganizing a Folder	8-2	Securing Write Access to Tapes	10-4
Saving and Restoring Folders and Folder Members	8-2	Using Tapes Secured by Other Systems	10-4
Using Data Dictionaries	8-2	Programming Considerations for Tape Processing	10-4
Working with a Data Dictionary	8-2	Automatically Advancing to Next Tape Drive	10-4
Working with Data Dictionary Definitions	8-2	Using REWIND, LEAVE, and UNLOAD Tape Cartridge Processing	10-5
Using Data Dictionary Definitions	8-2	Preparing Tapes	10-5
Saving and Restoring a Data Dictionary	8-2	Allocating the Tape Drive to a Job	10-5
Programming Considerations	8-2	Copying, Saving, Restoring, and Listing Information	10-6
Coexistence Considerations	8-3	Removing Information from Tape	10-7
Moving from System/36 to the System/36 Environment	8-3	Using Multiple Tape Drives	10-7
Chapter 9. Diskette Storage	9-1	Creating a Sequential Set of Files on Tape	10-7
Diskette Types and Storage Capacities	9-1	Coexistence Considerations	10-8
Diskette Exchange Formats	9-1	Restoring from an AS/400 System to System/36	10-9
Basic Data Exchange Format	9-1	Restoring System/36 Files and Members to the AS/400 System	10-9
H-Data Exchange Format	9-2		

Restoring Files and Members from an AS/400 System to the System/36 Environment	10-9	Considerations for System/36 Environment Program Start Requests	13-8
Moving from System/36 to the System/36 Environment	10-10	Errors on Program Start Requests Subsystem	13-11
Chapter 11. Security	11-1	Descriptions/Communications Entries Subsystem Communications Device	13-15
System Security Levels	11-1	Allocation	13-16
Sign-On Security	11-1	OS/400 Intersystem Communications Function (ICF)	13-16
User Profiles and Special User Authority	11-2	ICF Files	13-17
User Class	11-2	Tying the Application to Communications Configurations	13-17
Special Authority	11-2	Communications Operations	13-20
Initial Program Security	11-4	Return Codes and Messages	13-28
Menu Security	11-4	Testing Communications Applications	13-28
Limited Capability	11-4	File Transfer Subroutines	13-30
Group Profile	11-4	File Transfer Subroutine Parameters	13-31
Resource Security	11-4	File Transfer Support Considerations	13-33
Authority for a User to a Resource	11-5	Asynchronous Communications	13-34
Public Authority	11-7	Asynchronous Configuration Considerations	13-34
Library-Level Security	11-7	Asynchronous Programming Considerations	13-35
Authorization Lists	11-7	BSCCEL	13-36
Authority Holders	11-7	BSCCEL Terminology Considerations	13-36
Moving from System/36 to the System/36 Environment	11-8	BSCCEL Configuration Considerations	13-37
System/36 User Identification File	11-8	BSCCEL Programming Considerations	13-37
System/36 Resource Security File	11-8	Finance Considerations	13-38
Additional Security Considerations	11-11	Configuration Considerations	13-38
Saving and Restoring Authorities	11-12	Programming Considerations	13-38
Chapter 12. Designing Records	12-1	Retail Considerations	13-39
Identifying Required Fields	12-1	Configuration Considerations	13-39
Naming Fields	12-1	Programming Considerations	13-39
Using Numeric Fields	12-1	Intrasystem Communications	13-39
Zoned Decimal Format	12-2	Programming Considerations	13-39
Packed Decimal Format	12-3	System/36 APPC to AS/400 APPC	13-40
Binary Format	12-4	System/36 Peer to AS/400 Advanced Program-to-Program Communications (APPC)	13-41
Floating-Point Format	12-5	System/36 BSC/CICS to AS/400 SNA Upline Facility	13-41
Using Alphanumeric Fields	12-5	Remote Host Support Considerations	13-41
Using Keys	12-5	SNUF Programming Considerations	13-41
Allowing for Deletion of Records	12-6	System/36 BSC/IMS to AS/400 SNA Upline Facility	13-42
Determining Field Size	12-6	Remote Host Support Considerations	13-42
Defining Record Length	12-6	SNUF Programming Considerations	13-42
Allowing for New Fields	12-6	Using CL Override Commands	13-44
Describing Record Layout	12-6	General Programming Considerations	13-45
Chapter 13. Communications	13-1	Migration Considerations	13-45
Configuring the Communications Environment	13-1	Automatic Dial and Telephone Number List Support	13-45
System/36 Background	13-1	X.21	13-46
Communications Procedures Examples ENABLE and VRYCFG Hierarchy and Examples	13-2		
OS/400 Subsystem Considerations for System/36 Users	13-7		

System Network Architecture Distribution		Moving from System/36 to the System/36	
Services (SNADS)	13-46	Environment	14-25
Object Distribution	13-47	System/36 Display File Enhancements	14-25
Sending AS/400 Objects	13-47	Optimizing Performance of Display Files	14-26
Receiving Objects as AS/400 Objects	13-47		
Sending AS/400 System/36 Environment		Chapter 15. Messages and Message	
Objects	13-47	Members	15-1
Receiving Objects in the AS/400		Types of Messages	15-1
System/36 Environment	13-48	Message Concepts	15-1
		Message Member Concept	15-1
Chapter 14. Menus and Displays	14-1	Message Files and the System/36	
Menus	14-1	Environment	15-2
Function Key Differences	14-1	Inserting Variable Data into Displayed	
User Menus	14-2	Messages	15-3
Menu Option Logging	14-3	Converting Message Text	15-4
Menu Security	14-3	Supplying Default Responses for Messages	15-5
Menu Formats	14-3	Default Response Process	15-5
Designing Menus	14-4	Severity Levels	15-5
Creating and Changing Menus	14-7	Considerations for Default Responses to	
Creating and Displaying Online Help		Messages	15-7
Information for Menus	14-7	Displaying Response Messages	15-7
Using Color or Highlighting on Menus	14-9	Displaying Informational and Prompting	
Displays	14-10	Messages	15-8
Display Data Management	14-10	Formatting Messages with Control	
Data Types	14-11	Characters	15-9
Attributes	14-12	System Operator Displays	15-9
Display Data Management Operations	14-12	User Authority to the System Operator	
Designing Displays	14-13	Message Queue	15-9
Types of Displays	14-14	Sending System/36 Environment	
Creating Display Formats	14-17	Messages	15-9
Creating Online Help Information for		Handling Defaults for System Operator	
Your Displays	14-18	Messages	15-11
Using Display Formats with the		Sending Messages	15-11
Programming Languages	14-19	Message Handling Considerations	15-11
Using Display Formats within a		Error Messages Not Displayed	15-12
Procedure	14-20	Problems with the Operator Messages	15-12
Using the Read-Under-Format		Console (System) Operator Messages	15-13
Technique	14-20	Automatic Reply Handling When	
Using Data Description Specifications		QSYSOPR Is in Default Mode	15-13
(DDS) and Screen Format Generator		Dual-Routed Messages	15-13
(\$SFGR)	14-21	Enhancements and Restrictions	15-13
Using \$SFGR to Change SFGR to DDS	14-21	Programming Guidelines	15-14
SFGR Printed Output	14-21	Creating or Changing Message Source	
Creating, Adding, Changing, or Deleting		Members	15-14
Display File Formats	14-21	Assigning Default Responses and	
Replacing a System/36 Load Member		Severity Levels	15-14
with an AS/400 Display File	14-22	Specifying a Message Member to Be	
Maximum Number of Display Devices	14-23	Used within a Procedure	15-15
Public Authority to Use SFGR Display		Message Member and Message File	
Files	14-23	Considerations	15-15
FORMAT Procedure Parameters	14-24	Displaying Messages from Procedures	15-16
Differences between System/36 SFGR		Checking Entries for Required	
and AS/400 DDS	14-24	Parameters	15-17
		Using Messages with Programs	15-17

Using Messages with Displays	15-17
Moving from System/36 to the System/36 Environment	15-17
Chapter 16. Programs and Procedures	16-1
Designing Programs and Procedures	16-1
Programs	16-1
Batch and Interactive Programs	16-1
Program Characteristics	16-2
Program Types	16-2
Comparison of Program Types	16-4
Summary Table of Users and Requesters	16-4
Designing Applications	16-5
Programming Considerations	16-6
Procedures	16-21
Procedure Attributes	16-21
Parts of a Procedure	16-21
Using Procedures	16-22
Procedures with Menus	16-23
Calling a Procedure from Another Procedure	16-23
Considerations for Multiple Requester Terminal Procedures	16-23
Delaying MRT Termination	16-25
Internal Processing of MRT Jobs	16-25
Designing Procedures	16-26
Naming a Procedure	16-26
Procedure Performance and Coding Techniques	16-26
Programming Considerations for Procedures	16-27
Moving from System/36 to the System/36 Environment	16-29
Chapter 17. Mixing System/36 Environment and AS/400 Functions	17-1
Using AS/400 Architectural Features in System/36 Programs	17-1
Using AS/400 CL Commands in the System/36 Environment	17-2
Entering AS/400 CL Commands Interactively	17-2
Adding AS/400 CL Commands to System/36 Procedures	17-3
Program Control in the System/36 Environment	17-5
// LOAD and // RUN OCL Statements	17-5
High-Level Language CALL Statement	17-8
File Processing in System/36 Environment	17-8
Database Files	17-8
Printer Files	17-11
Display and Communications Files	17-12
Other Device Files	17-14

Chapter 18. Jobs and Job Processing	18-1
Using Jobs and Job Processing	18-1
Jobs and Job Steps	18-1
Starting and Ending Jobs	18-1
Starting Jobs	18-1
Running Jobs	18-2
Using the System/36 Environment Command Processor	18-2
Using the Initiator Function	18-4
Processing OCL Statements and Procedure Control Expressions	18-4
Ending Jobs	18-6
Managing and Scheduling Jobs	18-7
Job Priorities	18-7
Using Batch Job Immediate Support	18-8
Using the Job Queue	18-8
Evoking Other Jobs	18-11
Submitting Jobs to Run Later	18-11
Using Job Queue to Run Jobs Later	18-11
WAIT OCL Statement	18-11
Submitting Jobs by Security Classification	18-12
Preventing Users from Ending Jobs	18-13
Preventing Interrupted Jobs	18-13
Preventing Informational Messages from Appearing	18-13
Running Jobs during Initial Program Load (IPL)	18-13
Running Jobs without Operators	18-14
End-of-Day Processing	18-14
Job Date and Date Format	18-14
Chapter 19. Error Prevention, Detection, and Recovery	19-1
Types of Failures and Errors	19-1
System Failures	19-1
Disk Device Failures	19-1
Power Failures	19-1
Equipment Failures	19-1
Programming Errors	19-1
System Operator Errors	19-2
User Errors	19-2
Error Prevention	19-2
Using the Automatic Response Function	19-2
Preventing Unscheduled Ending of Jobs	19-2
Testing and Debugging Programs	19-2
Using the WAIT and FILE OCL Statements	19-3
Allocating the Diskette or Tape Drive to a Job	19-3
Error Detection	19-3
Error Detection Subroutines	19-3
Program Language Error Detection	19-3
System/36-Compatible COBOL Language	19-4
System/36-Compatible RPG II Language	19-4

User-Coded Error Detection Routines	19-4	Handling Synonyms	A-3
Checking Return Codes in Procedures	19-4	Indexed File with Keys	A-4
Referring to the Job Log	19-4	Randomizing Techniques	A-6
Backup and Recovery	19-4		
Equipment Backup	19-4	Appendix B. \$SFGR Specification Forms	B-1
Data Backup and Recovery	19-5	Display Control (S) Specifications	B-1
Backup and Recovery Methods	19-6	Help Definition (H) Specifications	B-8
Service Aid Procedures	19-8	Field Definition (D) Specifications	B-13
Error-Handling Considerations	19-8		
Disk Storage Full	19-8	Appendix C. Merging Graphics and Text	C-1
Display Station Device Error		Printing a Graphics File Only	C-1
Considerations	19-8	Graphics File Printout Example	C-1
Display Station Device Error Recovery	19-10	Printing a Graphics File Along with Other	
Printer Device Error Considerations	19-10	Output	C-2
ICF Error Considerations	19-10	Example of an Included Graphics File	C-2
Database File Error Conditions	19-11	Programming Considerations	C-3
Chapter 20. System/36 Environment		Appendix D. Intelligent Printer Data	
National Language Support	20-1	Stream (IPDS) Advanced Function	
System/36 Environment Multiple Language		Support	D-1
Support	20-1	Calling the Subroutines	D-1
Multiple Language Support for		COBOL Subroutines	D-1
IBM-Supplied Data	20-1	RPG II Subroutines	D-1
Multiple Language Support User-Supplied		RPG II and COBOL Printer Parameters	D-1
Data	20-1	Using the IPDS Advanced Function Support	D-2
Multilingual System Environment	20-2	Setting Printer Options	D-2
System/36 Environment Double-Byte		Printing Graphics Using Subroutines	D-7
Character Support	20-3	Printing Forms and Graphs	D-11
AS/400 Double-Byte Character Set		Printing Bar Codes	D-12
System Value	20-3	Sample Form	D-14
AS/400 Double-Byte Character Set Job			
Attribute	20-3	Appendix E. Security Considerations for	
System/36 Environment Double-Byte		the System/36 Environment	E-1
Character Job Attribute	20-6	System/36 Procedures	E-1
IGC Procedure	20-6	System/36 Operator Control Commands	E-20
Setting the Library List for DBCS Session	20-7	System/36 OCL Statements	E-22
System/36 Environment DBCS Printer		System/36 Procedure Control Statements	E-24
Support	20-9	OS/400 System/36 Commands	E-25
Writing Applications for Translating			
Considerations	20-9	Bibliography	H-1
		General AS/400-Related Books	H-1
Appendix A. Access Algorithms for Direct		Programming Language and Utility Books	H-2
Files	A-1	Communications Books	H-2
Choosing an Access Algorithm	A-1	Migration Books	H-3
Handling Synonym Records	A-1		
Examples of Access Algorithms	A-2	Index	X-1
Defining the Algorithm	A-2		

Notices

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any of the intellectual property rights of IBM may be used instead of the IBM product, program, or service. The evaluation and verification of operation in conjunction with other products, except those expressly designated by IBM, are the responsibility of the user.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, 500 Columbus Avenue, Thornwood, NY 10594, U.S.A.

| Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact the software interoperability coordinator. Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

| Address your questions to:

| IBM Corporation
| Software Interoperability Coordinator
| 3605 Highway 52 N
| Rochester, MN 55901-9986 USA

This publication could contain technical inaccuracies or typographical errors.

This publication may refer to products that are announced but not currently available in your country. This publication may also refer to products that have not been announced in your country. IBM makes no commitment to make available any unannounced products referred to herein. The final decision to announce any product is based on IBM's business and technical judgment.

Changes or additions to the text are indicated by a vertical line (|) to the left of the change or addition.

This publication contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

Trademarks And Service Marks

The following terms, denoted by an asterisk (*) in this publication, are trademarks of the IBM Corporation in the United States or other countries or both:

Application System/400

APPN

AS/400

CICS

COBOL/400

DisplayWrite

IBM

IPDS

Intelligent Data Printer Stream

OfficeVision

Operating System/400

OS/2

OS/400

RPG/400

SAA

System/36

System/38

Systems Application Architecture

VTAM

400

About System/36 Environment Programming (SC41-4730)

This book contains information about System/36 environment utilities, programming languages, and licensed programs. It also contains information about System/36 functions available in the System/36 environment.

For information about other AS/400 publications, see either of the following:

- The *Publications Reference* book, SC41-4003, in the AS/400 Softcopy Library.
- The *AS/400 Information Directory*, a unique, multimedia interface to a searchable database containing descriptions of titles available from IBM or from selected other publishers. The *AS/400 Information Directory* is shipped with your system at no charge.

For a list of publications related to this book, see the “Bibliography.”

Who Should Use This Book

This book supplies programmers with the information needed to develop, maintain, and support application programs and data to be used in the System/36 environment. It gives an overview of the system parts and how the system works.

Before you use this book, you should be familiar with System/36, the AS/400 system, and your workstation.

In this book, the term *user* refers to the application user. The term *operator* refers to the system operator.

Chapter 1. Introduction

This chapter introduces the IBM* System/36 environment. The System/36 environment is a function of the Operating System/400* (OS/400*) operating system that processes System/36 operation control language (OCL) statements and other procedure statements to run System/36 applications. In addition, the System/36 environment also allows control language (CL) commands to be processed. The System/36 environment helps you develop, maintain, and support a common set of programs, data, and other system elements on the System/36 and the AS/400* system.

System/36 Function in the System/36 Environment

The System/36 environment supplies the procedures, operation control language statements, utility control statements, and control commands that you use to:

- Create and maintain System/36 programs on the AS/400 system
- Run System/36 programs and procedures on the AS/400 system
- Create and maintain System/36 disk files on the AS/400 system
- Sort System/36 files on the AS/400 system
- Create and maintain System/36 libraries on the AS/400 system
- Create and maintain System/36 folders on the AS/400 system
- Process System/36 information on diskettes and tapes on the AS/400 system
- Create and maintain System/36 display formats, menus, and message members on the AS/400 system
- Create and maintain System/36 procedures on the AS/400 system

The following sections supply an overview of the System/36 environment.

Operating in the System/36 Environment

Operating in the System/36 environment consists of using procedures and operator control commands. A **control command** in the System/36 environment is a command used by an operator to control the system or a work station. A control command does not run a procedure and cannot be used in a procedure. A **work station** is a device used to transmit information to or receive information from a computer; for example, a display station or printer. The AS/400 control language (CL) commands are used to start and end operations in the System/36 environment. The **control language (CL)** is the set of all commands with which a user requests system functions.

See Chapter 2, "Operating in the System/36 Environment," for more information.

Configuration

The System/36 environment consists of the following:

- The AS/400 configuration
- System/36 environment configuration

See Chapter 3, "Configuring the System/36 Environment," for more information.

Printed Output

Printed output consists of reports and lists.

Printer data management allows your programs to use printers. The system uses print spooling, a system function that saves printer output on disk for later printing of print requests.

Several printers are available for the system. They have different speeds, character sets, and other options. Printers can be attached directly or remotely to the system.

See Chapter 4, "Printed Output," for more information.

File and Library Storage

Files and libraries are stored on disk. Blocks and sectors are the units of measure for disk storage. A **block** in the System/36 environment is a 2560-byte area of disk storage used when creating or referencing disk files. Files and libraries stored in disk devices have the location of each data record directly addressed to allow for direct access.

See Chapter 5, "Library, File, and Folder Overview," for more information.

Libraries

A library is a named area on disk that contains other objects, such as programs and related information. You can use libraries to find specific objects on the system.

The system can contain the following types of libraries:

- The System/36 environment libraries (QSSP and #LIBRARY)
- The system library (QSYS)
- Licensed program libraries
- Application libraries

See Chapter 6, "Libraries," for more information.

Files

A file is a set of related records treated as a unit. There are several ways to organize and access files. The following are examples of files:

- Transaction
- Master
- Memo

See Chapter 7, "Files," and Chapter 16, "Programs and Procedures," for more information.

Folders

A folder is a named area on disk that contains members created and used by the word processing function of OfficeVision for OS/400 and Client Access for OS/400. A **document**, in the System/36 environment is one or more lines of text that can be named and stored in a folder.

You can group folders in several ways. Each folder can contain multiple members of the same type. Folders serve as directories to documents and other folders.

See Chapter 8, "Folders and Data Dictionaries," for more information.

Diskette and Magnetic Tape Storage

You use diskettes and tapes to make backup copies of information and to store files and libraries outside of the system.

The AS/400 system supports several types of diskettes. You can use magnetic tape on reel or cartridge. Tapes hold more information than diskettes.

See Chapter 9, "Diskette Storage," and Chapter 10, "Magnetic Tape Storage," for more information.

Security

Security is the protection of data, system operation, and system devices. The System/36 environment uses the AS/400 security functions. Security levels on the AS/400 system include:

- Physical (level 10)
- Password (level 20)
- Password and Resource (level 30)
- Password, Resource, and operating system integrity (level 40)
- Password, Resource, and enhanced operating system integrity (level 50)

Note: Accessing objects using interfaces not supported on the system causes programs to fail.

See Chapter 11, "Security," for more information.

Designing Records

A record is a collection of fields. When you design a record you must include all required input and output fields. The way you design records depends on the type and format of the records needed, the type of fields and files used, and how the records are used.

See Chapter 12, “Designing Records,” for more information.

Communications

The AS/400 system uses data communications to send and receive information from different devices and systems. The system acts as a host system to remote work stations, acts as a secondary station to a remote host system, or communicates with another system as a peer.

See Chapter 13, “Communications,” for more information.

Menus and Displays

A menu is a displayed list of options from which a user makes a selection. You can use screen design aid (SDA) or the Build Menu (BLDMENU) procedure to create a menu. The **screen design aid (SDA)** is a function of the Application Development Tools licensed program that helps the user design, create, and maintain displays and menus.

The user enters data on a display to communicate with a program. The program uses a display to show data to the user. You can use SDA or the FORMAT procedure to create a display.

See Chapter 14, “Menus and Displays,” for more information.

Messages and Message Members

The AS/400 system uses messages to communicate with you. Your programs use messages to communicate with users. Messages supply information, prompt the user to enter data, or indicate an error has occurred. A **message member** in the System/36 environment is a library load member that defines the text of each message and its associated message identification code. On the OS/400 operating system, a message member is a message file (*MSGF) object.

The following are the types of messages available on the AS/400 system:

- Informational
- Prompting

- Error

See Chapter 15, “Messages and Message Members,” for more information.

Programs and Procedures

A procedure is a collection of statements that can cause one or more programs to run. The procedure statements are in a library member called a procedure member. On the OS/400 operating system, procedure members are stored as members of source physical file QS36PRC.

To run a procedure, you can enter a procedure command, which is the name of the procedure member in the library. Enter procedure commands with information that tells the procedure what to do.

See Chapter 16, “Programs and Procedures,” for more information.

Mixing System/36 Environment and AS/400 Functions

An application that is migrated from a System/36 to the System/36 environment is called a System/36 application. Within the System/36 environment, you can change these migrated applications to use AS/400 functions. These changed applications are called mixed mode applications. You can eventually change the application entirely to an AS/400 application.

However, when you start mixing System/36 environment functions and AS/400 functions, you must follow certain rules. See Chapter 17, “Mixing System/36 Environment and AS/400 Functions” for more information.

Jobs and Job Processing

On System/36, a **job** is a unit of work composed of one or more programs. A **job step** is a unit of work done by one program. Jobs can have one or more job steps.

See Chapter 18, “Jobs and Job Processing,” for more information.

Error Prevention, Detection, and Recovery

Create a backup and recovery plan for your system, once you understand the errors that can occur and how to prevent them and recover from them.

See Chapter 19, "Error Prevention, Detection, and Recovery," for more information.

National Language Support

The System/36 environment allows you to work in your own national language. This national language can be a single- or double-byte character language.

See Chapter 20, "System/36 Environment National Language Support," for more information.

Licensed Programs

The following sections describe the licensed programs, supplied by IBM, that you can use in the System/36 environment.

Utilities and Application Development Tools

The application development tools offer the following functions for creating and maintaining parts of applications:

- **Data file utility (DFU).** Use DFU to create and maintain simple data entry programs, file update programs, file inquiry programs, and report printing programs. You can use DFU to interactively create and maintain programs instead of coding in a programming language.
- **Programming development manager (PDM).** Use PDM to create and maintain procedures and source programs.
- **Screen design aid (SDA).** Use SDA to create and maintain displays and menus.
- **Source entry utility (SEU).** Use SEU to create and maintain procedures and source programs.

- **Business graphics utility (BGU).** Use BGU to design and produce business and scientific charts.
- **Character generator utility (CGU).** Use CGU to define and maintain user-defined double-byte characters and related sort information.

Query

Query allows you to request a variety of reports based on information in your files. It uses a series of displays to prompt you to specify:

- The information you want in your report
- Whether you want to print or display the report
- Whether you want to store the query data in a disk file
- How you want the report to look

You can save a query in a library once it is created. You can change your saved queries, copy them, or delete them from the library.

The *Query/400 Use* book has more information about Query/400.

OfficeVision for OS/400

The OfficeVision for OS/400 functions include word processing and office tasks.

Word Processing: You can use the word processing function to do the following:

- Create documents (such as letters, memos, and reports).
- Create online help information for application programs.

The *Using OfficeVision/400 Word Processing* book has more information about the word processing function.

Office Tasks: The office task function supplies automatic ways to handle office tasks, such as:

- Electronic mail handling
- Calendar management
- Directory support
- Distribution lists
- Message handling
- Administrative support

The *Using OfficeVision/400* book has more information about the office task function of OfficeVision for OS/400.

Client Access/400

The Client Access/400 licensed program allows you to use a personal computer as a work station attached to your AS/400 system. The Client Access/400 has the following features:

- PC files are stored in folders.
- Virtual printer support lets you use printers attached to the AS/400 system as if they were attached to your personal computer.
- Transfer function support lets you transfer AS/400 source members, procedure members, and files to the personal computer. It also lets you transfer personal computer files to the AS/400 system.

Refer to the *Client Access/400 for DOS with Extended Memory User Guide* or *Client Access/400 for OS/2 User Guide*, and the *Client Access/400 for DOS and OS/2 Technical Reference* books for more information about Client Access/400.

Programming Languages

The AS/400 system supports several programming languages, including RPG/400*, COBOL/400*, and PL/I. The System/36 environment also supports System/36-compatible RPG II and System/36-compatible COBOL. When you use a programming language, you control:

- How information appears on the users' displays
- How information appears on printed reports
- What processing the program does

Chapter 2. Operating in the System/36 Environment

This chapter discusses operating in the System/36 environment.

Operating in the System/36 environment consists of using procedures and operator control commands. AS/400 control language (CL) commands are used to start and end operations in the System/36 environment.

A user can choose one of two methods for running in the System/36 environment:

- A user profile attribute that indicates a user should always have access to the System/36 environment functions
- Commands to access the System/36 environment functions

System/36 Environment User Profile Attribute

Use the special environment attribute in a user's profile (the special environment (SPCENV) parameter on the Create User Profile (CRTUSRPRF) CL command and the Change User Profile (CHGUSRPRF) CL command) to indicate that you should have access to the System/36 environment functions. Following are the values you can specify for this attribute:

***NONE**

If you specify *NONE for the special environment attribute in your user's profile, you do *not* automatically have access to the System/36 environment functions when signing on to the system. If you need to run a System/36 environment procedure, operator control command, and so on, you must use the Start System/36 (STRS36) CL command to access the System/36 environment functions.

***S36**

If you specify *S36 for the special environment attribute in your user's profile, you automatically access the System/36 environment when you sign on to the system.

***SYSVAL**

If you specify *SYSVAL for the special environment in your user profile, system value QSPCENV determines whether the system automatically gives you access to System/36 environment functions when you sign on. The QSPCENV value can be *S36 or *NONE. Use the Display System Value (DSPSYSVAL) and Change System Value (CHGSYSVAL) CL commands to display and change the QSPCENV value.

If most users want access to the System/36 environment functions, set system value QSPCENV to *S36 and specify *SYSVAL for the special environment value when you create user profiles (*SYSVAL is the default when creating user profiles). If you do not want an individual user to automatically have access to System/36 environment functions, specify *NONE for the special environment attribute in the user's profile. The special environment value *NONE, when in the user's profile, overrides the value in the system value QSPCENV.

If most users do not want access to the System/36 environment functions, set system value QSPCENV to *NONE and specify *SYSVAL as the special environment value when you create user profiles. If you want an individual user to automatically have access to System/36 environment functions, specify *S36 for the special environment attribute in the user's profile. The special environment value *S36, when in the user's profile, overrides the value in the system value QSPCENV.

Note: The subsystem routing entry used to start the job must be QCMD to automatically access the System/36 environment functions. QCMD is the default for the IBM-supplied subsystems QBASE, QINTER, QBATCH, and so on. A **subsystem** is an operating environment, defined by a subsystem description, in which the system coordinates processing and resources. See the *Work Management* book for more information on the routing entries.

Commands to Access the System/36 Environment Functions

If you do not have automatic access to the System/36 environment functions, you can use commands to enter the System/36 environment. Use the STRS36 command to run System/36 environment functions. Either use the End System/36 Environment (ENDS36) command to exit the System/36 environment, or use the sign off (SIGNOFF) command to sign off the system and exit the System/36 environment functions. You can also exit the System/36 environment by pressing F3 or F12 on the menu displayed by the command if the FRCMNU (*YES) option was specified on the STRS36 command.

If you need to run a single System/36 environment procedure, use the Start System/36 Procedure (STRS36PRC) command. STRS36PRC runs the procedure in the System/36 environment and automatically returns you to your previous environment.

When you use the STRS36 or STRS36PRC CL commands, the System/36 environment saves the name of the current library. A **current library** is the library specified to be the first user library searched for objects requested by a user. You can specify the name for the current library on the sign-on display or in a user profile. When you specify an object name (such as the name of a file or program) on a command but do not specify a library name, the system searches the libraries in the system part of the library list and then searches the current library before searching the user part of the library list. The current library is also the library that the system uses when you create a new object if you do not specify a library name. When you leave the System/36 environment (using the ENDS36 CL command or at the end of the procedure started with the STRS36PRC CL command), the current library is restored to the value it was when you entered the System/36 environment.

You can enter System/36 environment procedures and CL commands from the System/36 environment Command Entry display or on the command line of any menu. To access the System/36 Environment Command Entry display, type in an

asterisk on the command line of any menu in the System/36 environment. To exit the System/36 Environment Command Entry display, press either F3 (Exit) or F12 (Cancel).

Notes:

1. The STRS36 CL command blanks the local data area when you enter the System/36 environment.
2. The ENDS36 CL command blanks the local data area when you leave the System/36 environment.
3. The STRS36PRC CL command does not blank the local data area.
4. You cannot leave the System/36 environment with the Exit (F3) or Cancel (F12) keys unless FRCMNU (*YES) is specified on the STRS36 command. Use the ENDS36 command to return to the environment you left when you entered the STRS36 CL command.
5. You cannot use the STRS36PRC command when a System/36 environment procedure is active. For example, a CL program called from a System/36 environment procedure cannot use STRS36PRC to call a System/36 environment procedure.

See Chapter 6, "Libraries," for information on how the library list is changed by the STRS36 and STRS36PRC CL commands.

See the *CL Reference* book for information about the STRS36, ENDS36, and STRS36PRC CL commands.

Accessing System/36 Environment Functions from Batch Jobs

The following types of System/36 environment jobs automatically access System/36 environment functions:

- // EVOKE OCL statement
- JOBQ operator control command
- // JOBQ OCL statement
- F6 from the help prompt (\$HELP) for the System/36 environment procedure
- Multiple requester terminal (MRT) programs

- Nonrequester terminal (NRT) programs
- Intersystem communications function (OS/400-ICF, hereafter referred to as ICF) start requests for procedures

The **intersystem communications function (ICF)** is a function of the operating system that allows a program to communicate with another system. An **operating system** is a collection of system programs that control the overall operation of a computer system. A **system program** in the System/36 environment is an IBM-supplied program that is installed on the system. The System/36 environment utility program \$MAINT is an example.

Jobs created by the Submit Job (SBMJOB) CL command use the special environment attribute of the user profile for the batch job to determine whether the job has automatic access to System/36 environment functions.

When jobs of this type are submitted, all information regarding the submitting work station is lost, such as the work station printer's identity. If this information is necessary, use a System/36 environment command or OCL statement to submit the job. The **operation control language (OCL)** in the System/36 environment is a language used to identify a job and its processing requirements to the System/36 environment. If you used the SBJOB command to submit the STRS36PRC command, this information is also lost.

If you use a System/36 environment procedure, command, or OCL statement to submit the batch job (for example, JOBQ, EVOKE, and so on), much information is copied from the submitting job to the submitted job by the System/36 environment to set up the environment for the batch job. Most of this information usually involves the current session values, but also includes the work station ID of the submitting job. A **session** in the System/36 environment is the length of time that starts when a user signs on to the System/36 environment and ends when the user signs off the System/36 environment. A **work station ID** in the System/36 environment is a 2-character identifier assigned to each display station and printer on your system.

If the special environment value for your user profile is *S36 (or your user profile value is *SYSVAL and the system value QSPCENV is

*S36), the batch job can access System/36 environment functions. If the special environment value for your user profile is *NONE (or the user profile value is *SYSVAL and the system value QSPCENV is *NONE), the job does *not* access System/36 environment functions. Use the STRS36PRC command in a job that does not access System/36 environment functions to run a System/36 environment procedure. You cannot use the STRS36 and ENDS36 commands in batch jobs.

Note: The subsystem routing entry you use to start the job must be QCMD to gain automatic access to System/36 environment functions. This routing entry is the default for IBM-supplied subsystems, such as QBASE, QINTER, and QBATCH. See the *Work Management* book for information about routing entries.

Subconsoles

The concept of a subconsole is not supported by the System/36 environment. This section describes the steps you can perform to map System/36 subconsole support to the AS/400 message handling support:

- The system automatically creates a message queue for each display station named by the display station ID. The queue holds messages sent to the device. For example, a message from the console to X1 is placed in display station message queue X1. A **console** is a display station from which an operator can control and observe the system operation.
- You must run the Change Device Printer (CHGDEVPR) or Create Device Printer (CRTDEVPR) commands for each printer controlled by a subconsole. Specify the display station message queue to receive messages for a printer. For example, if X1 is the subconsole for printers P1 and P2, you must run the CHGDEVPR command for printers P1 and P2, and you must specify a message queue of QSYS/X1.
- When you sign on to a display station, the system puts the message queue for the display station in notify mode. If a message is on the message queue for the display at the time you sign on, the message-waiting light will be turned on. When a message is sent to

the message queue the message-waiting light is turned on if it is not already on.

To display the messages for your display station, use the Display Message (DSPMSG) CL command.

The Display Messages display appears as follows:

```

Display Messages
Queue . . . . . : QSYSOPR      System : RCH38360
Library . . . . : QSYS        Program . . . . : *DSPMSG
Severity . . . . : 72         Library . . . . :
                        Delivery . . . . : *HOLD

Type reply (if required), press Enter.
Controller RCHASG11 contacted on line LANLINE.
All sessions ended for device RCHASLAM.
An adapter has inserted or left the token-ring on line LANLINE.
An adapter has inserted or left the token-ring on line LANLINE.
All sessions ended for device RCHAS41501.
All sessions ended for device S1010107.
An adapter has inserted or left the token-ring on line LANLINE.
An adapter has inserted or left the token-ring on line LANLINE.
An adapter has inserted or left the token-ring on line LANLINE.
An adapter has inserted or left the token-ring on line LANLINE.
SYS1631 Options ( 123)
Empty slot or picker failure during DELETE.
Reply . . . . .
Bottom

F3=Exit      F11=Remove a message      F12=Cancel
F13=Remove all  F16=Remove all except unanswered  F24=More keys

```

Because the messages are saved in a message queue, you can (with security authorization to the message queue) display and respond to these messages.

The format and options for the Display Message display are the AS/400 message display format and options.

You are not given special authority to spool commands when your display station is defined to receive the messages for a printer. Authorization does not depend on the display station from which you are operating.

System Request Menu

Use the System Request key to interrupt a job or display console messages. The System Request menu has a format similar to the following in the System/36 environment:

```

System Request
System: RCH38360

Select one of the following:

1. Display sign on for alternative job
2. Cancel job and close files; new data is saved
3. Display current job
4. Display messages
5. Send a message
6. Display system operator messages
7. Display work station user
10. Start system request at source system
11. Transfer to (source/target) system
20. Set inquiry condition for S/36 program

80. Disconnect job

90. Sign off

Selection
—
F3=Exit      F12=Cancel
(C) COPYRIGHT IBM CORP. 1988, 1992.

```

Consider the following factors when using the System Request menu:

- The wording of options is tailored to the environment in which you are running.
- Options not allowed because of the ATTR OCL statement are not shown on the display.
- The system suspends AS/400 and single requester terminal (SRT) jobs.
- In a multiple requester terminal (MRT), only the MRT's use of the display station from which the System Request key was pressed is suspended. The MRT can continue to access other display stations.
- You can restrict access to the system request function by revoking access to the system request panel group (*PNLGRP) named QGMNSYSR in library QSYS.
- On System/36, if a utility uses the diskette drive, and the system issues a message that requires you to initialize a diskette, you can request a command display with inquiry option 1. From the new session, you can then run the INIT procedure and initialize a diskette. In the System/36 environment, when the system issues a message that requires you to initialize a diskette, use message option INZ to initialize the diskette. The system does *not* allow a secondary job running at the display station to use the diskette when the system allocates the diskette to the primary job.
- On System/36, CMD1 ended the secondary session and returned to the previous session. On the AS/400 system, the SIGNOFF command is used for this function.
- In a MRT program, processing of the System Request key is delayed until the MRT has an input operation outstanding for the display

station from which the system request key was used. A **multiple requester terminal (MRT) program** in the System/36 environment is a program that can process requests from more than one display station or ICF session at the same time using a single copy of the program.

- System request is ignored for acquired display stations.

Attention Key

A user-written application can define a program to be called when you press the Attention key. Use the Set Attention Program (SETATNPGM) command or the Attention key program attribute in the user profile to define this program. However, the Attention key does not work when the work station is running a MRT program. For more information about setting up the Attention key, see the *Work Management* book.

Note: You should not access System/36 environment functions from the Attention key program.

OS/400 CL Commands for the System/36 Environment

This section gives a brief description of commands and procedures you can use in the System/36 environment. For a detailed description of these and other CL commands, see the *System/36 Environment Reference* and the *CL Reference* books.

Change System/36 Message List (CHGS36MSGL)

Use the CHGS36MSGL command to specify the action to be taken by the system when an error occurs on a CL command in a procedure.

Change System/36 (CHGS36)

Use the CHGS36 command to change the System/36 environment. This command displays a series of screens with which you can tailor the System/36 environment.

Change System/36 Environment Attributes (CHGS36A)

Use the CHGS36A command to change values in the System/36 environment configuration. You can use this command while others are in the System/36 environment.

Change System/36 Procedure Attributes (CHGS36PRCA)

Use the CHGS36PRCA command to change the System/36 attributes of a procedure member.

Change System/36 Program Attributes (CHGS36PGMA)

Use the CHGS36PGMA command to change the System/36 attributes of a program.

Change System/36 Source Attributes (CHGS36SRCA)

Use the CHGS36SRCA command to change the System/36 attributes of a source member.

Create System/36 COBOL Program (CRTS36CBL)

Use the CRTS36CBL command to create a System/36-compatible COBOL program.

Create System/36 Display File (CRTS36DSPF)

Use the CRTS36DSPF command to create a display file from a screen format generator (SFGR) source member. The **screen format generator (SFGR)** in the System/36 environment is a utility on the AS/400 system that creates AS/400 display files from System/36 SFGR source statements. This command is equivalent to the System/36 FORMAT procedure. You can use this command to convert the System/36 SFGR source to the AS/400 DDS source.

Create System/36 Menu (CRTS36MNU)

Use the CRTS36MNU command to create a menu from a source member in System/36 format. It creates a display file and message file from a message text source member and \$SFGR source member or option text message source member. This command is equivalent to the System/36 BLDMENU procedure.

Create System/36 Message File (CRTS36MSGF)

Use the CRTS36MSGF command to create a message file from a message source member in System/36 format. This command is equivalent to the System/36 CREATE procedure. Also, you can use it to convert the System/36 message source to the AS/400 CL source.

Create System/36 Message File Menu (CRTMSGFMNU)

Use the CRTMSGFMNU command to create a menu from source that is in System/36 format.

This command creates a display file from an option text message file or command text message file.

Create System/36 RPG II Program (CRTS36RPG)

Use the CRTS36RPG command to create a System/36-compatible RPG II program.

Display System/36 Environment (DSPS36)

Use the DSPS36 command to display or print the System/36 environment configuration. This command displays a series of panels that show the values specified with the CHGS36 command.

Edit System/36 Procedure Attributes (EDTS36PRCA)

Use the EDTS36PRCA command to edit the System/36 attributes of one or more procedures.

Edit System/36 Program Attributes (EDTS36PGMA)

Use the EDTS36PGMA command to edit the System/36 attributes of one or more programs.

Edit System/36 Source Attributes (EDTS36SRCA)

Use the EDTS36SRCA command to edit the System/36 attributes of one or more source members.

End System/36 (ENDS36)

Use the ENDS36 command to end a System/36 environment session started with the STRS36 command.

Restore System/36 File (RSTS36F)

Use the RSTS36F command to restore a file or a group of files saved in the System/36 save and restore format using the SAVE procedure.

Restore System/36 Folder (RSTS36FLR)

Use the RSTS36FLR command to restore a folder saved in the System/36 save and restore format.

Restore System/36 Library Member (RSTS36LIBM)

Use the RSTS36LIBM command to restore library members (source and procedure) saved in the System/36 save and restore format using the FROMLIBR or SAVELIBR procedures.

Retrieve System/36 Environment Attributes (RTVS36A)

Use the RTVS36A command to retrieve System/36 environment configuration values.

Save System/36 File (SAVS36F)

Use the SAVS36F command to save a single file or multiple files in the System/36 save and restore format.

Save System/36 Library Member (SAVS36LIBM)

Use the SAVS36LIBM command to save library members (source and procedure) in the System/36 save and restore member format.

Start System/36 Procedure (STRS36PRC)

Use the STRS36PRC command to run a System/36 environment procedure and return to the environment from which you entered the System/36 environment.

Start System/36 (STRS36)

Use the STRS36 command to start a System/36 environment session, and optionally display a menu.

Work with System/36 Configuration (WRKS36)

Use the WRKS36 command to change, display or print the System/36 environment configuration. This command shows you the name of the System/36 environment configuration object. By selecting option 2, you can change your configuration. By selecting option 5, you can display your configuration. By selecting option 6, you can print your configuration.

Work with System/36 Procedure Attributes (WRKS36PRCA)

Use the WRKS36PRCA command to change or display the System/36 attributes for your procedures. The command shows you a list of procedures and their descriptions. By selecting option 2, you can change the attributes. By selecting option 5, you can display the attributes.

Work with System/36 Program Attributes (WRKS36PGMA)

Use the WRKS36PGMA command to change or display the System/36 attributes for your programs. The command shows you a list of programs and their descriptions. By selecting option 2, you can change the attributes. By selecting option 5, you can display the attributes.

**Work with System/36 Source Attributes
(WRKS36SRCA)**

Use the WRKS36SRCA command to change or display the System/36 attributes for your source members. The command shows you a

list of source members and their descriptions. By selecting option 2, you can change the attributes. By selecting option 5, you can display the attributes.

Chapter 3. Configuring the System/36 Environment

Configuration consists of the following parts:

- AS/400 system processes for configuring the system hardware and software
- Configuration of the System/36 environment

This chapter describes the process and characteristics of configuring the System/36 environment. It does not describe system configuration.

System Values Affecting the System/36 Environment

The following system values affect how the AS/400 system and the System/36 environment work together:

QCONSOLE

The System/36 environment uses the display device name specified to determine whether the OCL IF CONSOLE test is evaluated as true or false.

Display the system value using the following command:

```
DSPSYSVAL QCONSOLE
```

QPRTDEV

The System/36 environment uses the printer device name specified as the default printer or system printer when you do not direct output to a specific printer.

Change the system value using the following command:

```
CHGSYSVAL QPRTDEV name
```

name is the AS/400 device name of a printer.

QSPCENV

If your user profiles have *SYSVAL for the SPCENV parameter (the default), the QSPCENV system value determines whether you run in the System/36 environment.

To ensure that all users operate in the System/36 environment at all times, each user profile should have the SPCENV parameter value *SYSVAL and the QSPCENV system value *S36. To change the system value to *S36, type CHGSYSVAL QSPCENV *S36. When you configure your system with this

command, the AS/400 system you are on can run System/36 applications. Use this same command to configure the system if you have migrated from a System/36.

You can change the QSPCENV value during initial program load (IPL). If you select Y to display additional options, the AS/400 system presents an option to change the default special environment. Set the option to *S36 to change the QSPCENV value. You can tailor your system at IPL time.

QDEVNAMING

If this value is *S36, the AS/400 system assigns a 2-character device name to all displays, printers, tapes, and diskettes automatically configured by the AS/400 system. The device names follow the System/36 naming convention.

When the AS/400 configuration notifies the System/36 environment configuration that a new device has been added (and it has a 2-character name that follows the System/36 naming conventions), the System/36 environment tries to make the System/36 ID the same as the AS/400 device name. That way, error messages that the System/36 environment shows for displays, printers, tapes, and diskettes, match the configured names.

Change the system value for QDEVNAMING using the following command:

```
CHGSYSVAL QDEVNAMING *S36
```

You can change the QDEVNAMING value at IPL time. If you select Y to display additional options, the System/36 environment presents an option to change the device configuration names. Set the option to *S36 to change the QDEVNAMING value.

If the AS/400 device name for the first display that becomes active is set to DSP01, and you change QDEVNAMING to *S36, you may need to do an IPL to change the DSP01 name to W1.

If automatic-configuration is on and has already configured the displays, printers, tapes, or diskettes with non-System/36 naming conventions, do the following to

rename your devices with System/36 naming conventions:

1. Vary the device off by using either the WRKCFGSTS or VRYCFG command.
2. Delete the device description by using the DLTDEVD command.
3. Turn off the device.
4. Wait about 10 seconds and turn the device back on.
5. Use the WRKCFGSTS command to ensure that the device was automatically-configured based on the System/36 naming conventions.

Do this for each device you want to have renamed.

Note: If you use the automatic advance function (AUTO-YES) support for T1 and T2, the tape drives assigned to these 2-character device names must support the same tape density. If tape reels are used, they must also be the same density.

For more information on AS/400 device names, see "AS/400 Device Configuration" on page 3-3.

Starting Another AS/400 Subsystem

After loading the System/36 environment onto the AS/400 system, starting the next AS/400 subsystem causes the subsystem to:

- Find the current AS/400 device configurations for displays, printers, tapes, and diskettes.
- Create #LIBRARY if it does not exist.
- Create the object QS36ENV of type *S36 in #LIBRARY (if a System/36 environment configuration does not exist in #LIBRARY).
- Create a default System/36 ID for each 10-character AS/400 device name. This data is placed in the object QS36ENV of type *S36 in #LIBRARY.

If the AS/400 device name is a valid System/36 device ID, it tries to create the System/36 environment device ID to match the AS/400 device name.

- Provide default values for 3270 device emulation, general System/36 environment values, and MRT security. The **3270 device emu-**

lition is the operating system support that allows an AS/400 system to appear as a 3274 Control Unit in a BSC multipoint network or SNA/SDLC network.

System/36 Environment Configuration

The *Local Device Configuration* book explains how to configure your AS/400 system. The information in the following sections explains how to configure the necessary data to run System/36 applications on your AS/400 system.

You need a System/36 environment configuration to specify values used to run jobs in the System/36 environment.

Commands for Configuring the System/36 Environment

The following commands process the System/36 environment configuration data:

Change System/36 (CHGS36)

Use the CHGS36 command to change values in the System/36 environment configuration. If you use this command, no one else is allowed in the System/36 environment at the same time.

Change System/36 Attributes (CHGS36A)

Use the CHGS36A command to change values in the System/36 environment configuration while the System/36 environment is active. This command can be used in a noninteractive job.

Display System/36 (DSPS36)

Use the DSPS36 command to view or print the System/36 environment configuration data.

Retrieve System/36 Attributes (RTVS36A)

Use the RTVS36A command to retrieve System/36 configuration environment values. This command can be used in a noninteractive job.

Work with System/36 Configuration (WRKS36)

Use the WRKS36 command to change or display the System/36 environment configuration. This command shows you the name of the System/36 environment configuration object. By selecting option 2, you can change

your configuration. By selecting option 5, you can display your configuration. By selecting option 6, you can print your configuration.

AS/400 Device Identification

AS/400 device description names can be up to 10 characters. System/36 device IDs can be only 2 characters. Much of the information you specify for the System/36 environment concerns mapping between AS/400 device names and the System/36 environment device IDs. The following items depend on a 2-character ID to run without change:

OCL	Application programs
\$SFGR displays	Menus

The System/36 environment configuration maps between the AS/400 device description names and the System/36 IDs for the following devices:

Displays	Printers
Tapes	Diskette

Notes:

1. A maximum of 1128 devices can be supported with the 2-character System/36 names. The first character can be one of the letters A–Z, \$, #, or @. The second character can be one of the letters A–Z, a number 0–9, \$, #, or @. The IDs F1, #D, and #P are not used; T1, T2, TC, and I1 are reserved for tapes and diskettes respectively.

Once the System/36 names have been used, the System/36 environment configuration ignores any new devices added to the system.

2. Communications devices are not mapped and you must use the AS/400 device description name.
3. Along with mapping between the AS/400 device names on the System/36 IDs, you can also specify the System/36 environment values for files libraries, default session libraries, date-differentiated files, printer information, 3270 device emulation, and MRT security.

AS/400 Device Configuration

When a new display, printer, tape, or diskette is configured on the AS/400 system, the AS/400 device configuration informs the System/36 environment configuration that a new device has been configured.

If the AS/400 device name follows the System/36 naming conventions (QDEVNAMING is set to *S36), and the System/36 environment does not have a System/36 device ID by that name, then the System/36 environment configuration sets the System/36 device ID to match the AS/400 device name.

If the AS/400 device name follows the System/36 naming conventions, but the System/36 environment configuration is already using this ID, or if the AS/400 device name follows AS/400 standards (QDEVNAMING is set to *NORMAL), the System/36 environment configuration creates a System/36 device ID.

If the AS/400 device names for displays, printers, tapes, or diskettes follow the System/36 naming convention (QDEVNAMING is set to *S36), and some of the AS/400 device names no longer have matching System/36 device IDs, you can do the following to help match them up:

1. Issue the CHGS36 command.
2. Select a 2 option to change all of the device sections.
3. Erase or blank out the System/36 ID.
4. Press F5 to refresh the screen.
5. Press F10 to set the System/36 IDs.

If a 2-character AS/400 device name is found, and it is a valid System/36 ID for this device type, the System/36 2-character ID is set to match the AS/400 2-character device name. For any AS/400 device names that do not meet the above requirement, a System/36 ID is generated for the remaining AS/400 device names.

6. Press the Enter key to save the changes.
7. After you have changed all of the device sections and returned to the Change S/36 Environment Configuration display, press the Enter key to save all of the changes.

Considerations for System/36 Environment Installation or PTF Application

You should not be in the System/36 environment while you do any of the following:

- Install, delete, or save the System/36 environment licensed program.
- Apply or remove program temporary fixes (PTFs) to or from the System/36 environment licensed program. A **program temporary fix (PTF)** is a temporary solution to, or bypass of, a defect in a current release of a licensed program.

You must do an IPL for the system after you do any of the following:

- Install or delete the System/36 environment licensed program.
- Apply or remove PTFs to or from the System/36 environment licensed program.

To check if you are in the System/36 environment, enter `STATUS SESSION` from the command line of a menu. If the Display Session Status display appears, you are in the System/36 environment. If you are not in the System/36 environment, you receive an error message, such as `Command STATUS in library *LIBL not found.`

Another way to determine whether you are in the System/36 environment is to specify the Display Job (DSPJOB) CL command and select option 1 to display your job status attributes. If your job is processing in the System/36 environment, the special environment attribute for your job is *S36.

If you entered the System/36 environment with the Start System/36 Environment (STRS36) CL command, you can use the End System/36 Environment (ENDS36) CL command to leave the System/36 environment.

If you entered the System/36 environment because the special environment value for your user profile is *S36, or your user profile is *SYSVAL and system value QSPCENV is *36, you can do the following to leave the System/36 environment:

1. Change the special environment of your user profile from *S36 to *NONE by using the

Change User Profile (CHGUSRPRF) CL command.

2. Sign off the system by using the Sign Off (SIGNOFF) CL command.
3. Sign back on the system.

After signing back on the system, you will not be in the System/36 environment.

Security Attributes for Multiple Requester Terminals (MRTs)

If the Security Attributes for Multiple Requester Terminals (MRTs) option appears, your user profile has a user class attribute of *SECOFR. Use the MRT security option to specify security and authority attributes for your MRT program. You can specify that the MRT program has the security and authority attributes of the initiator of the MRT program, or of the owner of the MRT program. An **initiator** in the System/36 environment is the part of the system that reads and processes operation control language statements from the system input device.

In addition, you can specify whether the system checks the authority of all users of a MRT program, or only the authorities of the initiator of the MRT program against the files the MRT program may use.

Depending on the security needed (and on some performance considerations), you can determine whether MRT security works as it does on System/36, or if you should tailor it to work differently. For more information on security, see Chapter 11, "Security." For information on the Change S/36 MRT Security and Performance display, see page 3-11.

Configuration of the System/36 Environment

This section describes the System/36 environment configuration displays and prompts that you see when you use Change System/36 Environment Configuration (CHGS36) command, or after typing a 2 in the *Option* column, next to the configuration description of the Change S/36 Environment Configuration display.

Change S/36 Environment Configuration Display

The Change S/36 Environment Configuration display looks similar to the following:

```

Change S/36 Environment Configuration
S/36 environment . . . . . : #LIBRARY
Type options, press Enter.
2=Change

Option Configuration Description
- S/36 display IDs
- S/36 printer IDs
- S/36 tape IDs
- S/36 diskette IDs
- S/36 3270 device emulation values
- S/36 environment values
- S/36 MRT security and performance

F3=Exit F12=Cancel

```

Function keys

The following function keys are valid for the Change S/36 Environment Configuration display:

Enter

All entries are verified and, if correct, saved when no options are on the display. If incorrect entries were made, you are returned to the display with the entries. If no incorrect entries were made, the configuration information is sorted and saved.

If one or more options were selected on the Change S/36 Environment Configuration display, the configuration option is shown.

Page Up or Roll Down

Moves backward to show additional information for this display.

Page Down or Roll Up

Moves forward to show additional information for this display.

Print

Prints the information currently shown on your display.

Help

Provides more information about using the display.

F1=Help

Provides more information about using the display.

F3=Exit

Shows a display that lets you specify if you want to save the changes made on the configuration displays.

F12=Cancel

Shows a display that lets you specify if you want to save the changes made on this display.

To save the changes you make in this display, or in any of the configuration menus for the System/36 environment, press the Enter key.

Change S/36 Environment Configuration Display

The following sections describe the displays you may see when making changes to each of the seven configuration descriptions of the Change S/36 Environment Configuration display.

S/36 Display IDs: If you type a 2 in the *Option* column, next to the *S/36 display IDs Configuration Description* of the Change S/36 Environment Configuration display and press the Enter key, your display might look as follows:

```

Change S/36 Display IDs
S/36 environment . . . . . : #LIBRARY
Type new/changed values, press Enter.

AS/400 S/36 S/36 Default
Display Display ID Printer ID
DSP11 WA P3
DSP12 WB
DSP13 WC
DSP10 W0
DSP01 W1
DSP02 W2
DSP03 W3
DSP04 W4
DSP07 W7
DSP08 W8 P3
DSP09 W9
X5 X5

F3=Exit F5=Refresh F6=Sort S/36 display IDs More...
F10=Set S/36 display IDs F12=Cancel

```

You can specify the printer that you want your output to go to when you run in the System/36 environment from that display station.

The following function keys are valid for the Change S/36 Display IDs display:

F3=Exit

Shows a display that lets you specify if you want to save the changes made on the configuration displays.

F5=Refresh

All AS/400 system display device names are shown. To use a display device when running in the System/36 environment, it must have a System/36 display ID. To add an AS/400 system display device to the System/36 environment, type a System/36 printer ID and press the Enter key, or press F10 (Set S/36 displays).

F6=Sort S/36 display IDs

The information for AS/400 system display device names that do not have corresponding System/36 display station IDs is removed from the list, and the list is sorted to remove blank lines.

F10=Set S/36 display IDs

If an AS/400 system display device name does not have a corresponding System/36 environment ID, one is assigned. The AS/400 system device name is used as the System/36 environment ID if:

- The AS/400 system display device name is 2 characters in length.
- The characters are allowed for a System/36 environment display ID.
- The characters are not already being used as a System/36 environment ID.

Otherwise, a unique System/36 environment ID is created for the AS/400 system device.

A System/36 environment ID is assigned for each AS/400 system device name that does not have a corresponding System/36 environment ID.

F12=Cancel

Shows a display that lets you specify if you want to save the changes made on this display.

S/36 Printer IDs: If you type a 2 in the *Option* column, next to the *S/36 printer IDs Configuration Description* of the Change S/36 Environment Configuration display and press the Enter key, your display might look as follows:

```
Change S/36 Printer IDs
S/36 environment . . . . . : #LIBRARY
Type new/changed values, press Enter.
AS/400      S/36      Lines      Characters
Printer     Printer ID  Per Inch  Per Inch  Font
P1          P1         4         15        05
P3          P3         -         -         -

Bottom
F3=Exit F5=Refresh F6=Sort S/36 printer IDs F10=Set S/36 printer IDs
F12=Cancel
```

You can specify the lines per inch, characters per inch, and font for any printer. If the printer cannot support the function, the specified value will be ignored.

The following function keys are valid for the Change S/36 Printer IDs display:

F3=Exit

Shows a display that lets you specify if you want to save the changes made on the configuration displays.

F5=Refresh

All AS/400 system printer device names are shown. To use a printer device when running in the System/36 environment, it must have a System/36 printer ID. To add an AS/400 system printer device to the System/36 environment, type a System/36 Printer ID and press the Enter key, or press F10 (Set S/36 printers).

F6=Sort S/36 printer IDs

The information for AS/400 system printer device names that do not have corresponding System/36 printer IDs are removed from the list, and the list is sorted to remove blank lines.

F10=Set S/36 printer IDs

If an AS/400 system printer device name does not have a corresponding System/36 environment ID, a System/36 environment ID is assigned. The AS/400 system device name is used as the System/36 environment ID if:

- The AS/400 system printer device name is 2 characters in length.
- The 2 characters are allowed for a System/36 environment printer ID.

- The 2 characters are not already being used as a System/36 environment printer ID.

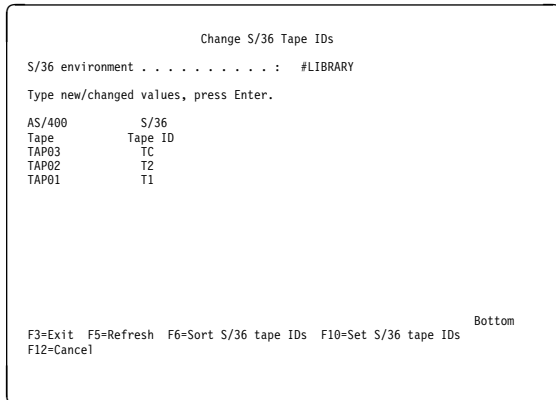
Otherwise, a unique System/36 environment ID is created for the AS/400 system device.

A System/36 environment ID is assigned for each AS/400 system device name that does not have a corresponding System/36 environment ID.

F12=Cancel

Shows a display that lets you specify if you want to save the changes made on this display.

S/36 Tape IDs: If you type a 2 in the *Option* column, next to the *S/36 tape IDs Configuration Description* of the Change S/36 Environment Configuration display and press the Enter key, your display might look as follows:



Note: If you use the automatic advance feature (AUTO-YES) supported for T1 and T2, the tape drives assigned to these 2-character device names must be of the same model.

The following function keys are valid for the Change S/36 Tape IDs display:

F3=Exit

Shows a display that lets you specify if you want to save the changes made on the configuration displays.

F5=Refresh

All AS/400 system tape device names are shown. To use a tape device when running in

the System/36 environment, it must have a System/36 tape ID. To add an AS/400 system tape device to the System/36 environment, type a System/36 tape ID and press Enter, or press F10 (Set S/36 tapes).

F6=Sort S/36 tape IDs

The information for AS/400 system tape device names that do not have corresponding System/36 tape IDs are removed from the list, and the list is sorted to remove blank lines.

F10=Set S/36 tape IDs

If an AS/400 system tape device name does not have a corresponding System/36 environment ID, a System/36 environment ID is assigned. The only System/36 environment IDs allowed for a tape device are T1, T2, and TC. The AS/400 system device name is used as the System/36 environment ID if:

- The AS/400 system device tape name is 2 characters in length.
- The 2 characters are allowed for a System/36 environment tape ID.
- The 2 characters are not already being used as a System/36 environment tape ID.

Otherwise, a unique System/36 environment ID is created for the AS/400 system device.

An System/36 environment ID is assigned for each AS/400 system device name that does not have a corresponding System/36 environment ID.

Up to three tape IDs can be specified.

F12=Cancel

Shows a display that lets you specify if you want to save the changes made on this display.

S/36 Diskette IDs: If you type a 2 in the *Option* column, next to the *S/36 diskette IDs Configuration Description* of the Change S/36 Environment Configuration display and press the Enter key, your display might look as follows:

```

Change S/36 Diskette IDs
S/36 environment . . . . . : #LIBRARY
Type new/changed values, press Enter.
AS/400          S/36
Diskette        Diskette ID
DKT01          I1

F3=Exit  F5=Refresh  F10=Set S/36 diskette  F12=Cancel  Bottom

```

The following function keys are valid for the Change S/36 Diskette ID display:

F3=Exit

Shows a display that lets you specify if you want to save the changes made on the configuration displays.

F5=Refresh

All AS/400 system diskette device names are shown. To use a diskette device when running in the System/36 environment, it must have a System/36 diskette ID. To add an AS/400 system diskette device to the System/36 environment, type a System/36 diskette ID and press Enter, or press F10 (Set S/36 diskette).

F10=Set S/36 diskette

If an AS/400 system diskette device name does not have a corresponding System/36 environment ID, a System/36 environment ID is assigned. The only System/36 environment ID allowed for a diskette device is I1, and only one diskette device can be assigned to I1.

F12=Cancel

Shows a display that lets you specify if you want to save the changes made on this display.

S/36 3270 Device Emulation Values: If you type a 2 in the *Option* column, next to the *S/36 3270 device emulation values Configuration Description* of the Change S/36 Environment Configuration display and press the Enter key, your display might look as follows:

```

Change S/36 3270 Device Emulation Values
S/36 environment . . . . . : #LIBRARY
National language
UK multinational
Type choice, press Enter.
Location name . . . . . ENGLAND
Type option, press Enter.
2=Change national language for emulation
Option National Language for Remote Displays
- Austria/Germany
- Austria/Germany multinational
- Belgium
- Belgium multinational
- Canada
- Canada multinational
- Denmark
- Denmark multinational
F3=Exit  F12=Cancel  Bottom

```

Use the 3270 emulation option to specify translation on a remote display.

You can specify a location name and select one national language to be used by 3270 device emulation when running from a remote display. The AS/400 system uses the location name specified in the System/36 environment configuration when you do not specify a location name on the ES3270 procedure.

Press the Page Down key to continue to the next Change S/36 3270 Device Emulation Values display. If you continue to press the Page Down key, you will see a total of eight displays containing the following choices of national languages for remote displays:

- Austria/Germany
- Austria/Germany multinational
- Belgium
- Belgium multinational
- Canada
- Canada multinational
- Denmark
- Denmark multinational
- Finland
- Finland multinational
- France (Azerty)
- France (Azerty) multinational
- France (Qwerty)
- France (Qwerty) multinational
- International
- International multinational
- Italy
- Italy multinational
- Norway
- Norway multinational
- Portugal
- Portugal multinational
- Spain

- Spain multinational
- Latin American Spanish-Speaking
- Latin American Spanish-Speaking multinational
- Sweden
- Sweden multinational
- Switzerland/French
- Switzerland/French multinational
- Switzerland/German
- Switzerland/German multinational
- UK
- UK multinational
- United States/Canada
- United States multinational
- Japan English
- Japan English multinational
- Japan Katakana
- Netherlands
- Netherlands multinational
- Iceland
- Iceland multinational
- ROECE-Latin
- Turkey
- Yugoslavia multinational
- Cyrillic
- Greece
- Thailand
- Cursive language–right-to-left
- Noncursive language–right-to-left

The following function keys are valid for the Change S/36 3270 Device Emulation Values display:

F3=Exit

Shows a display that lets you specify if you want to save the changes made on the configuration displays.

F12=Cancel

Shows a display that lets you specify if you want to save the changes made on this display.

S/36 Environment Values: If you type a 2 in the *Option* column, next to the *S/36 environment values Configuration Description* of the Change S/36 Environment Configuration display and press the Enter key, your display might look as follows:

```

Change S/36 Environment Values
S/36 environment . . . . . : #LIBRARY
Type choices, press Enter.

S/36:
Default session library . . . . . QS36F
Default files library . . . . . N           Y=Yes, N=No
Use library list for files . . . . . N       Y=Yes, N=No
Date differentiated files . . . . . Y       Y=Yes, N=No
Shared opens of files . . . . . Y          Y=Yes, N=No
Record blocking when sharing files . . . . N   Y=Yes, N=No
Store deleted files in cache . . . . . Y     Y=Yes, N=No
Default lines per page . . . . . 066       1-112
Default forms . . . . . *STD
Default message action . . . . . *CONTINUE *CONTINUE, *IGNORE
                                           *HALT, *CANCEL
Halt options . . . . . 03                 More...

F3=Exit  F5=Refresh  F10=Set to default values  F12=Cancel

```

The following general System/36 environment values can be changed on this display:

- The default session library. This library works as it did on the System/36, except the library name does not appear on the AS/400 sign-on display.
- The default files library. This is the library where you can find your System/36 environment files. On the AS/400 system, files must reside in a library.
- Use library list for files. Specify N for this field if you do not want the System/36 environment to do an automatic search of the library list for database files.
- Date differentiated files. Specify Y for this field if any application uses date-differentiated files.
- Shared opens of files. Specify N for this field if you do not want the System/36 environment to use automatic shared opens of files.
- Record blocking when sharing files. Specify Y for this field if you want the System/36 environment to use record blocking for all sequentially accessed files, including those shared with other jobs.
- Store deleted files in cache. Specify N if you want to delete a database file through the DELETE OCL statement. Otherwise, the default value Y causes deleted files to be placed in a cache. When you create a database file, the System/36 environment searches the cache for a file with the same attributes and inserts the file into the library.
- Print information
 - The default lines per page. This option works as it did on System/36.

- The default forms value. This option works as it did on System/36.
- Default message action. You can specify the default action to be taken when an error occurs on a CL command within a procedure. Specify *HALT if you want the job to halt with options when an error occurs. Specify *CONTINUE if you want the job to continue without stopping when an error occurs. The substitution expression ?MSGID? will be set to the message ID of the escape message. Specify *IGNORE if you want the job to continue without stopping when an error occurs. The substitution expression ?MSGID? will not be set to the message ID of the escape message. Specify *CANCEL if you want the job to be canceled when an error occurs.
- Halt options. You can specify the halt options that will be allowed when the job halts because of an error on a CL command within a procedure. Only shown if the Default message action is *HALT.

To change additional System/36 environment values, press the Page Down or Roll Up key. After doing this, the next Change System/36 Environment Values display appears. Your display looks similar to the following:

```

Change S/36 Environment Values
S/36 environment . . . . . : #LIBRARY
Type choices, press Enter.
EVOKE job initiation . . . . . *IMMED      *IMMED, *JOBQ
Storage pool . . . . . *BASE      *BASE, *CURRENT
Job priority . . . . . 50          1-99, *SUBMITTER
Source file record length . . . . . 132    52-132
Allow CHGS36 while active . . . . . N      Y=Yes, N=No
Add only S/36 environment users . . . . . N      Y=Yes, N=No
Substitute ICF procedure data . . . . . Y      Y=Yes, N=No
Description . . . . . #LIBRARY

F3=Exit  F5=Refresh  F10=Set to default values  F12=Cancel

Bottom

```

The following general System/36 environment values can be changed on this display:

- EVOKE job initiation. You can specify the method to be used to start EVOKE jobs and job steps started with the // ATTR RELEASE-YES OCL statement. *IMMED specifies the job is started by circumventing the JOBQ. This offers a performance improvement over the *JOBQ processing, but offers less flexibility than the *JOBQ support.

- *JOBQ specifies the job is started on the JOBQ.
- Storage pool. You can specify the storage pool to be used for jobs started with the *IMMED option. *BASE is the default and specifies that the job is started in the *BASE storage pool. *CURRENT specifies the job is started in the same storage pool as the submitting job. This value is only shown if *IMMED is specified for the EVOKE job initiation value.
- Job priority. You can specify the priority at which to start the job when the job is started with the *IMMED option. The valid priority values are 1–99 and *SUBMITTER. *SUBMITTER specifies to start the job at the same priority as the submitting job. The default is 50.
- Source file record length. You can specify the record length to be used whenever a new source file is created to hold your procedure statements (QS36PRC) or your source statements (QS36SRC). These source files are created when the BLDLIBR or TOLIBR procedures are run. They are also created when the Restore System/36 Library member (RSTS36LIBM) command is run. The value should be based on the maximum statement length in your procedure and source members plus 12 bytes required by the system. If your statements are 80 characters or less in length, you should choose 92 for the record length. The valid record lengths are 52 to 132. The default is 132.

- Allow CHGS36 while active. You can specify Y if you want to allow the configuration object to be updated with the CHGS36 command while others are signed onto the System/36 environment.
- Add only S/36 environment users. You can specify Y if you want to add workstation devices to the configuration only when the device signs on to the System/36 environment. Specify N if you want to add workstation devices to the configuration when the device is created.
- Substitute ICF procedure data. You can specify Y if you want data received on an ICF start request to be scanned for substitution expressions (except for ICF start requests from retail or finance devices). If the data

might contain question marks that should not be treated as substitution expressions, you should specify N.

- Description. You can specify a description of the System/36 environment. The description can be up to 40 characters, and any characters may be used. If no description is specified, #LIBRARY is used.

The following function keys are valid for the Change S/36 Environment Values display:

F3=Exit

Shows a display that lets you specify if you want to save the changes made on the configuration displays.

F5=Refresh

Removes any changes you made and shows the original values.

F10=Set to default values

Removes the values typed and shows the default values.

F12=Cancel

Shows a display that lets you specify if you want to save the changes made on this display.

S/36 MRT Security and Performance:

If you type a 2 in the *Option* column, next to the *S/36 MRT security and performance Configuration Description* of the Change S/36 Environment Configuration display and press the Enter key, your display might look as follows:

```

Change S/36 MRT Security and Performance
S/36 environment . . . . . : #LIBRARY
Type choices, press Enter.
User profile used by MRT program . . . . 1      1=First user of MRT
                                           2=Owner of MRT

Check authority of user to files used
by MRT program . . . . . 1      1=All users
                                           2=First user

Seconds to wait before ending a
non-NEP MRT . . . . . 0      0-32767
MRT job initiation . . . . . *IMMED *IMMED, *JOBQ
Storage pool . . . . . *BASE *BASE, *CURRENT
Job priority . . . . . 20     1-99, *SUBMITTER

F3=Exit  F5=Refresh  F10=Set to default values  F12=Cancel

```

The Change S/36 MRT Security and Performance display shows the user profile used by the MRT program. It also shows which user's authority is

examined by the MRT program when the system is running in the System/36 environment. It also shows the number of seconds the system is to wait before ending a MRT that is not a never-ending program (NEP). To change the user profile, user authority, and seconds to wait, type over the value and press the Enter key.

- User profile used by MRT program. For security checking, specify which user profile the MRT program should run under. The options are:

1=First user of MRT

The MRT program runs under the user profile that starts this MRT program. This is the default value.

2=Owner of MRT

The MRT program runs under the owner profile of the MRT program. This owner could be the programmer who created the MRT or another user made the owner by the Change Object Owner (CHGOBJOWN) command.

The default value for the user profile used by the MRT program option is 1.

- Check authority of user to files used by MRT program. For security checking, specify which user's access to the files the MRT program uses. The options are:

1=All users

The system checks all users of the MRT program to determine whether they have the correct authority to the files the MRT program uses. This is the default value.

2=First user

The system checks only the initiator of the MRT program to determine whether he has the correct authority to the files that the MRT program uses.

- Seconds to wait before ending a non-NEP MRT. When a MRT program is not a NEP, and the MRT program releases the last ICF session or last display station, the program is given a return code instructing it to go to the end of the program. The system waits the number of seconds specified on this display before going to the end of the program. The value allowed is 0 through 32767. The default value for the seconds to wait is set automatically by the System/36 environment.

- MRT job initiation. This option specifies the method to be used to start MRT jobs.

The valid values are:

***IMMED**

By specifying *IMMED for the MRT job initiation value, the job is started by circumventing the JOBQ. This offers a performance improvement over the *JOBQ processing but offers less flexibility than the *JOBQ support. *IMMED is the default for the system.

***JOBQ**

By specifying *JOBQ for the MRT initiation value, the job is started using the JOBQ.

- Storage pool. This option specifies the storage pool to be used for the jobs started with the *IMMED option.

The valid values are:

***BASE**

By specifying *BASE, any job started with the *IMMED option uses the *BASE storage pool in the subsystem. *BASE is the default for the system.

***CURRENT**

By specifying *CURRENT, any job started with the *IMMED option uses the same storage pool as the submitting job.

- Job priority. This option specifies the priority at which to start the job when the job is started with the *IMMED option.

The valid values are:

1-99

By specifying a value from 1 to 99, the job starts with an initial run priority of the specified value.

The default is 20.

***SUBMITTER**

By specifying *SUBMITTER, the job is started with the same run priority as the submitter at the time the job is submitted.

The following function keys are valid for the Change S/36 MRT Security and Performance display:

F3=Exit

Shows a display that lets you specify if you want to save the changes made on the configuration displays.

F5=Refresh

Removes any changes you made and shows the original values.

P10=Set to default values

Removes the values typed and shows the default values.

F12=Cancel

Shows a display that lets you specify if you want to save the changes made on this display.

Change S/36 Environment Attributes

If you use the prompt function (F4) for the Change System/36 Environment Attributes (CHGS36A) command, a display shows similar to the following:

```

Change S/36 Environment Attr (CHGS36A)
Type choices, press Enter.
Default session library . . . . *SAME____ Name, *SAME
Default files library . . . . QS36F____ Name, *SAME
Use library list for file . . . *NO_____ *YES, *NO, *SAME
Date differentiated files . . . *NO_____ *YES, *NO, *SAME
S/36 shared opens of files . . . *YES_____ *YES, *NO, *SAME
Shared file record blocking . . *NO_____ *YES, *NO, *SAME
Store deleted files in cache . . *YES_____ *YES, *NO, *SAME
Default lines per page . . . . 66_____ 1-112, *SAME
Forms type . . . . . *STD_____ Character value, *STD, *SAME
Default message action . . . . *HALT_____ *CONTINUE, *HALT, *IGNORE...
Halt options . . . . . 03_____ *SAME, 0, 1, 2, 3, 01, 02...
Evoke job initiation . . . . . *IMMED_____ *IMMED, *JOBQ, *SAME
Evoke storage pool . . . . . *BASE_____ *BASE, *CURRENT, *SAME
Evoke job priority . . . . . 50_____ 1-99, *SUBMITTER, *SAME
Source file record length . . . 132_____ 52-132, *SAME
Allow CHGS36 while active . . . *NO_____ *YES, *NO, *SAME
F3=Exit F4=Prompt F5=Refresh F12=Cancel F13=How to use this display More...
F24=More keys

```

```

Change S/36 Environment Attr (CHGS36A)
Type choices, press Enter.
Add only S/36 users . . . . . *NO_____ *YES, *NO, *SAME
Substitute ICF procedure data . *YES_____ *YES, *NO, *SAME
MRT user profile . . . . . *FRSTUSR *OWNER, *FRSTUSR, *SAME
Check authority to files . . . . *ALLUSR_ *ALLUSR, *FRSTUSR, *SAME
MRT delay value . . . . . 0_____ 1-32767, *SAME
MRT job initiation . . . . . *IMMED_____ *IMMED, *JOBQ, *SAME
MRT storage pool . . . . . *BASE_____ *BASE, *CURRENT, *SAME
MRT job priority . . . . . 20_____ 1-99, *SUBMITTER, *SAME
F3=Exit F4=Prompt F5=Refresh F12=Cancel F13=How to use this display Bottom
F24=More keys

```

For more information on the attributes that can be changed and a description of the function keys, see “S/36 Environment Values” on page 3-9.

Display S/36 Configuration Display

If you use the Display System/36 Environment Configuration (DSPS36) command, or select option 5 on the Work with S/36 Environment Configuration display, you can see the Display S/36 Environment Configuration display.

```

Display S/36 Environment Configuration
S/36 environment . . . . . : #LIBRARY
Type options, press Enter.
5=Display
Option Configuration Description
- S/36 display IDs
- S/36 printer IDs
- S/36 tape IDs
- S/36 diskette ID
- S/36 3270 device emulation values
- S/36 environment values
- S/36 MRT security and performance

F3=Exit      F12=Cancel

```

```

Retrieve S/36 Environment Attr (RTVS36A)
Type choices, press Enter.
Environment name . . . . . #LIBRARY NAME
CL var for SLIB (8) _____ Character value
CL var for FLIB (10) _____ Character value
CL var for LIBL (4) _____ Character value
CL var for DATDIFF (4) _____ Character value
CL var for S36SHARE (4) _____ Character value
CL var for RCDBLK (4) _____ Character value
CL var for CACHEDLTF (4) _____ Character value
CL var for LPPAGE (3) _____ Character value
CL var for FORMTYPE (4) _____ Character value
CL var for DFTMSGACN (9) _____ Character value
CL var for HALTOPT (4) _____ Character value
CL var for EVKJOBINIT (6) _____ Character value
CL var for EVKJOBPOL (8) _____ Character value
CL var for EVKJOBPTY (10) _____ Character value
CL var for SRCRCLEN (3) _____ Character value
More...
F3=Exit F4=Prompt F5=Refresh F12=Cancel F13=How to use this display
F24=More keys

```

```

Retrieve S/36 Environment Attr (RTVS36A)
Type choices, press Enter.
CL var for CHGACT (4) _____ Character Value
CL var for ADDS36ONLY (4) _____ Character Value
CL var for ICFUSBST (4) _____ Character Value
CL var for MRTUSRPRF (8) _____ Character Value
CL var for MRTAUT (8) _____ Character Value
CL var for MRTDLY (5) _____ Character Value
CL var for MRTJOBINIT (6) _____ Character Value
CL var for MRTJOBPOL (8) _____ Character Value
CL var for MRTJOBPTY (10) _____ Character Value
Bottom
F3=Exit F4=Prompt F5=Refresh F12=Cancel F13=How to use this display
F24=More keys

```

When you select option 5 for any of the configuration descriptions on the Display S/36 Environment Configuration display and press the Enter key, the displays that show would be similar to those that show when you use the CHGS36 command, or type 2 in the *Option* column of the Change S/36 Environment Configuration display. The information that appears after selecting option 5, could have been changed with either the CHGS36 or CHGS36A commands. You cannot change any information using the DSPS36 command. If you use the DSPS36 command with OUTPUT(*PRINT), the configuration information is printed in a spooled file, and you will not see any panels.

Retrieve S/36 Environment Configuration Attributes

If you use the prompt function (F4) for the Retrieve S/36 Environment Configuration Attributes (RTVS36A) command, the displays will be similar to the following:

For more information on the attributes that can be retrieved and a description of the function keys, see “S/36 Environment Values” on page 3-9.

Work with System/36 Environment Configuration

If you use the Work with System/36 Environment Configuration (WRKS36) command, the Work with S/36 Environment Configuration display should look similar to the following:

```

Work with S/36 Environment Configuration
Type options, press Enter.
2=Change 5=Display 6=Print
Opt Object Type Library Text
_ QS36ENV *S36 #LIBRARY
Bottom
F3=Exit F5=Refresh F12=Cancel

```

If you select option 2 for the System/36 Environment Configuration object shown, you will see the Change S/36 Environment Configuration display. This display is the same as the one you see when you use the CHGS36 command.

If you select option 5 for the System/36 Environment Configuration object shown, you will see the Display S/36 Environment Configuration display. This display is the same as the one you see when you use the DSPS36 command.

If you select option 6 for the System/36 Environment Configuration object shown, the configuration information is formatted and printed.

Removing Display IDs

You can use the CHGS36 command to remove display IDs from the configuration table by blanking out display ID values on the Change S/36 Display IDs display.

An easier way to remove display IDs from the configuration table is to use the QEXRMVDE API. This program is described in detail in the *System/36 Environment Reference* book.

Chapter 4. Printed Output

This chapter describes how to produce and control printed output for the System/36 environment. It does not describe how to design printed output or print text with graphics. For information on designing printed output, see the *Printer Device Programming* book. For information on printing graphs, see Appendix C, "Merging Graphics and Text." For information on intelligent printer data stream* (IPDS*) printer functions, see Appendix D, "Intelligent Printer Data Stream (IPDS) Advanced Function Support."

Creating and Controlling Printed Output

When deciding how to create printed output, consider the following:

- The source of the output
- The output length
- Whether to print or display the output
- Whether to add control codes

Printed output on the system is processed either by printer data management or a special system program called **system list**. The system list program processes the output created by some of the System/36 environment utilities (for example, the \$MAINT utility program).

The following types of programs use printer data management:

- User-written programs
- Print key
- OS/400 data communications programs
- Sort
- Application Development Tools products:
 - Data file utility (DFU)
 - Programming development manager (PDM)
 - Screen design aid (SDA)
 - Source entry utility (SEU)
- Business Graphics Utility (BGU)
- System/36-compatible RPG II compiler
- System/36-compatible COBOL compiler
- Word processing function of OfficeVision for OS/400
- Query/400

The following procedures and utility programs use the system list program to list their printed output:

Procedure	Utility
BLDMENU	\$BMENU
CREATE	\$MGBLD
FORMAT	\$SFGR
CATALOG	\$LABEL (local files, libraries, and folders)
LISTLIBR	\$MAINT

The following table shows the procedure and utility programs that use the system list program to determine the system list device (CRT or printer). If it is a printer, the printed output is produced by an OS/400 command through a printer device file.

Procedure	Utility	Printer Device File
CATALOG	\$LABEL (diskette) \$LABEL (tape) \$LABEL (remote files)	QPDSPKT QPTAPDSP QWRKDDM
LISTDATA	\$COPY	QSYSPRT
LISTFILE	\$BICR \$COPY \$TCOPY \$MAINT	QSYSPRT QSYSPRT QSYSPRT QPSRODSP

Notes:

1. \$BMENU and \$SFGR use printer data management, but are partially controlled by the system list program.
2. User-written RPG II and COBOL programs can use the system list program.

For more information about these procedures or utilities, refer to the *System/36 Environment Reference* book.

Printer Data Management Output

Your output prints with the paging and spacing that your program requested.

You can send the printed output from a program to a specific printer by using a PRINTER operation control language (OCL) statement. If you do not specify a PRINTER OCL statement, the system

sends printed output to the session printer for the display station. To use the session printer, set the OUTQ and PRTDEV parameters in your user profile to *WRKSTN. *WRKSTN is the default value. You also need to set the display station device description OUTQ and PRTDEV parameters to the name of your desired session printer. You can use the STATUS SESSION control command to display the ID of your session printer. You can use the PRINT procedure to change your session printer. This procedure changes the printer until you sign off the display station. The session printer only affects System/36 environment output, not AS/400 output while in the environment.

When you sign on, the session printer is determined by the value specified during System/36 environment configuration. The STATUS SESSION control command shows the configured session printer. Use the SET procedure to change your configured session printer. This procedure changes the printer until you run another SET procedure or change the system. The change remains in effect the next time you sign on the display station.

When the session printer is changed, the AS/400 job is also changed. This is done as though a CHGJOB command was run with the PRTDEV parameter specified. This ensures that the printed output generated in the System/36 environment defaults to the same printer. This change occurs when the System/36 environment is started and when the session printer changes. The job remains changed for the AS/400 job.

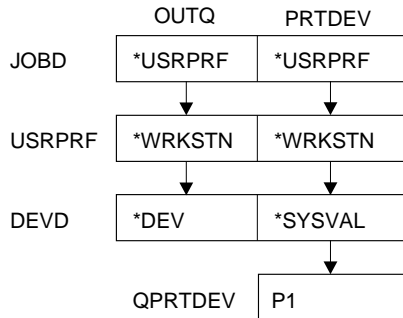
Whenever a Change Job (CHGJOB) command is issued to change the PRTDEV value of a current job, the System/36 environment session printer value is changed. Whenever the PRINT procedure or // FORMS OCL statement is used to change the session printer, the native PRTDEV value is also changed.

To determine which printer and output queue is used at job initiation time, the system looks at the following objects:

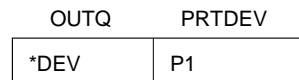
- Job description (JOBDD)
- User Profile (USRPRF)
- Workstation device description (DEVD)

The SET procedure and the Change System/36 Configuration (CHGS36) command change the PRTDEV parameter of the workstation device description instead of keeping the session printer value in its own configuration object.

The following diagram shows the path taken when the system determines the output queue and the print device for a job. The defaults are shown.

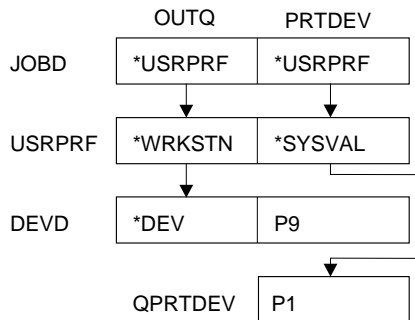


The result from the diagram above:

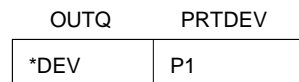


RV2W109-0

In the following example, the user specified a printer device in the device description.



The result from the diagram above:



RV2W110-0

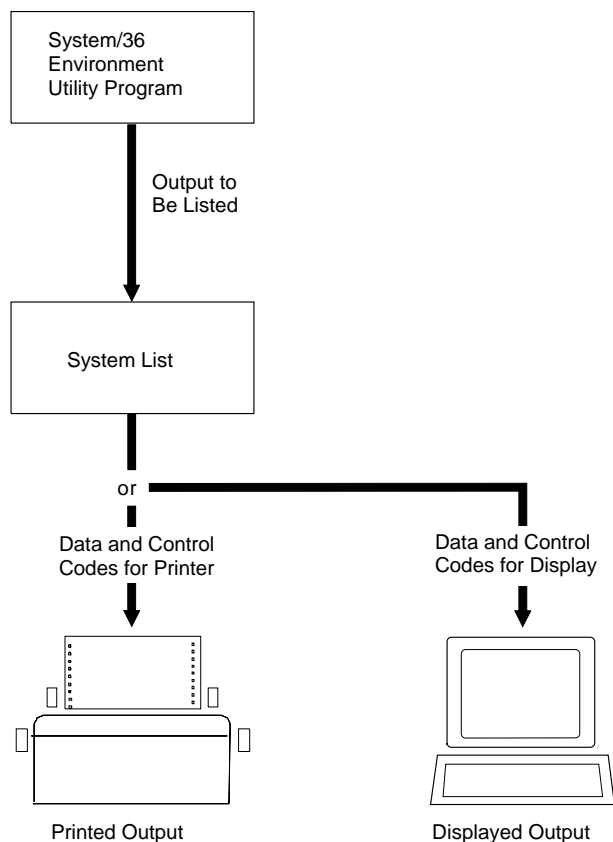
By setting the PRTDEV to *SYSVAL in the user profile, the session printer value set in the workstation device descriptions was never looked at. The session printer value was not reset because it was never used.

System List Output

The output from some of the System/36 environment utilities is handled by a special system program called system list. System list allows output to go to an output queue associated with a printer or to a display station.

Note: The output queue for your job must be *DEV. If you identify a specific output queue, all output will go to that queue regardless of printer DM or SYSLIST.

The following figure shows how the system processes system list output:



RSLW008-3

The system list program does the following:

1. Takes the computer printout requests from the utility program
2. Determines whether the output is to be printed or displayed
3. Sends that information to the output queue for printers associated with the correct device

The printer or display station that receives the system list output is called the **system list device**.

When you sign on, the system list device is the same as the session printer. To display the current system list device, use the STATUS SESSION control command. Use the SYSLIST procedure to change the system list device. This procedure changes the system list device until:

- You run another SYSLIST procedure
- You run a PRINT procedure or the FORMS OCL statement
- You sign off the display station

The *System/36 Environment Reference* book has more information about using the SYSLIST procedure.

Print Spooling

The system uses print spooling (the capability to store printed output in an output queue for later printing) for all printing requests. For more information about print spooling, refer to the *Printer Device Programming* book.

Using Output Queues

The System/36 environment uses **output queues** as System/36 uses the spool file, except there is one output queue for each printer configured on the system. The output queue is created during configuration.

If you want print spooling to be similar to System/36, configure your printers with a two-character device name such as P1 or P2. Use the Change System Value (CHGSYSVAL) command with the SYSVAL (QDEVNAMING) VALUE (*S36) parameters. This command causes 2-character device names to be created when the system automatically configures your devices. At configuration time, an output queue is created with the same name as this 2-character device name. For example, printer P1 is associated with output queue P1 in library QUSRSYS.

To have the System/36 environment run like System/36, specify that the output created by a job should be put in output queue *DEV. Specifying the *DEV value indicates that output for a

printer should be placed in the output queue for that printer. For example, output for printer P1 will be placed in output queue P1, output for printer P2 will be placed in output queue P2, and so on. See “Changing the Output Queue for a Job” for information on how to specify the output queue for a job.

Changing the Output Queue for a Job

Use any of the following methods to specify the output queue that will contain the printed output for a job:

- Specify the name of the output queue that is to contain the output (or specify *DEV to place the output in the output queue for the printer) in the job description for the job. Use this method when multiple users or jobs use the same job description and want their output in the same output queue.
- Specify that the user profile defines the output queue (use the *USRPRF special value) in the job description for the job. In each user’s profile, specify the name of the output queue that is to contain the output for the user. This method should be used when multiple users of jobs use the same job description and want their output in different output queues.
- After the job has been started, use the Change Job (CHGJOB) command to specify the name of the output queue that is to contain the output for the job. This method should be used if you temporarily need to change the output queue that is used by the job. The CHGJOB command will change the routing of the output for the duration of the job or until another CHGJOB command is entered.

Controlling Print Spooling

The operators control all output queues and print entries in the output queue. The operator can, for example, start or stop the printers and change the number of copies to print.

Users have some control over the print entries they create. For example, they can change the number of copies to print for spool files they created.

The commands used to control print spooling are shown in “Controlling or Displaying Print Spooling Information” on page 4-5.

Spool Writer Messages

Spool writer messages are sent to the message queue specified during configuration. To send all messages to the system operator queue, specify the QSYSOPR message queue in library QSYS.

To display the status of all active printer spool writers on the system, use the STATUS WRT control command. You can select an option to display the message queue associated with an active writer. See the *Printer Device Programming* book for complete information about how print spooling works.

Printer Control Guidelines

The following sections describe several functions you can use to manage or control printer output.

Changing the Session Printer

During System/36 environment configuration, you specify a printer to receive output from each display station. This printer is called the **session printer**. The session printer is used for all printed output created in the System/36 environment while a user is signed on unless a // PRINTER OCL statement has been processed for a particular print file. Use the PRINT procedure to change the following values:

- The printer ID used for printed output, including system list
- The number of lines per page
- The vertical print density (lines per inch)
- The horizontal print density (characters per inch)
- The forms number to be used

The PRINT procedure changes these values for the session. The settings return to the defaults when the session ends.

Note: Not all printers allow changing lines per inch or characters per inch. The *System/36 Environment Reference* book has more information about the PRINT procedure.

Changing the Print Key Printer

On the System/36, you can direct your Print key output to a unique printer. In the System/36 environment on the AS/400 system, all Print key output is sent to the printer assigned to the session printer. When the session printer is changed, the printer for Print key output is also changed.

The PRINTKEY procedure, the WORKSTN OCL statement, and the SETPK utility control statement of the \$SETCF utility program can be used to change whether a heading or border is to be printed with the Print key output. If a printer ID is specified for the Print key printer, the value is syntax-checked and ignored.

The SET procedure and the ID, BORDER, and HEADER parameters on the SETCF utility control statement of the \$SETCP utility program do not support changing the Print key printer, the Print key border, or the Print key header information. If any are specified, an error is issued.

Changing the System List Device

The system list device is the default printer specified during system configuration. Use the SYSLIST procedure to change the system list device to another printer, direct the output to the display station, or to turn this function off (no output is listed). The *System/36 Environment Reference* book has more information about the SYSLIST procedure.

Changing the System Printer

If the session printer is not set, and you do not specify a system list device, printed output uses the default system printer. You can specify or change the system printer by using the CHGSYSVAL command with the SYSVAL(QPRTDEV) parameter.

Changing the Printer Configuration Information

Use the SET procedure to specify the following printer-related items for the display station:

- Printer ID to use for session output
- Forms number

- Number of lines per page

These values remain in effect after the system operator signs off. However, when using the SET procedure to specify a printer ID, the values remain in effect after signing off only if you specify PRTDEV(*WRKSTN) in the user profile. The *System/36 Environment Reference* book has more information about the SET procedure.

Changing Printer Information in a Procedure

In a procedure, you can use the FORMS or PRINTER OCL statements to control how output prints. For example, you can change the following values:

- The printer ID printed output uses
- Number of lines per page
- Vertical print density (lines per inch)
- Horizontal print density (characters per inch)
- Forms number
- Number of copies to print
- Whether the paper needs to be aligned before the output prints

Whenever a RUN OCL statement is processed, a generic printer override is generated through an AS/400 OVRPRTF FILE (*PRTF) command. The information from the PRINT procedure (the FORMS OCL statement) is used as parameters on this command. Printing done by an application or a utility uses the parameter from this command.

If a PRINTER OCL statement is specified, RUN OCL statement processing generates a specific printer override using information from the PRINTER OCL statement. This override applies to the utility or application printer file that uses the same name as that used on the PRINTER OCL statement. See the *Data Management* book for more information on overrides.

The *System/36 Environment Reference* book has more information about these OCL statements.

Controlling or Displaying Print Spooling Information

The system operator controls print spooling with control commands. To display information about the status of the spooled files, use the STATUS or STATUSF control command with the PRT param-

eter. To display information about the status of the spool writer, use the STATUS control command with the WRT parameter.

You can use the following commands to control print spooling. Except where noted, use these System/36 environment commands as on System/36.

CANCEL PRT

The CANCEL PRT command cancels one or more spooled file entries.

CHANGE COPIES

The CHANGE COPIES command changes the number of copies of output to print.

CHANGE DEFER

The CHANGE DEFER command changes the delayed attribute of an spooled file entry. This attribute indicates whether an entry can begin printing before all the data has been produced by the program.

CHANGE PRTY

The CHANGE PRTY command changes the priority of the spool writer for a printer. This function is supported by the JOBPTY parameter on the CHGJOB command.

CHANGE SEP

The CHANGE SEP command changes the number of separator pages used to separate spooled file entries for a printer.

Note: You can use this command only when the spool writer is active.

CHANGE PRT

The CHANGE PRT, spool ID command moves a spooled file entry to the top of the output queue. The CHANGE PRT, spool ID, spool ID command is supported by the Change option on the Work with Spool Files (WRKSPLF) display. The OUTPTY parameter can be used on the Change Spool File Attributes (CHGSPLFA) display.

CHANGE FORMS

The CHANGE FORMS command changes the forms number of output to print.

CHANGE ID

The CHANGE ID command changes the printer ID assigned to output to print.

HOLD PRT

The HOLD PRT command holds selected spooled file entries to prevent them from printing.

RELEASE PRT

The RELEASE PRT command releases selected held spooled file entries to allow them to print.

RESTART PRT

The RESTART PRT command restarts the printing of a spooled file entry.

Note: You can use this command only when the spooled file entry has a status of writer (WTR).

START PRT

The START PRT command starts the spool writer so entries can print.

STOP PRT

The STOP PRT command stops the spool writer so entries cannot print.

For complete information on using these control commands, and their possible parameters and requirements, see the *System/36 Environment Reference* book.

Copying and Displaying Output from an Output Queue

Operators can copy output from an output queue to a disk file and display printed output from the disk file by using the COPYPRT procedure. You can do the following using the COPYPRT procedure:

- Copy reports from the output queue to a disk file.
- Display printed reports from a disk file at a display station before they are printed.
- Print reports copied to the disk file. You can also print selected pages of a report.

This procedure allows users to look at reports and decide which pages are to be printed.

Before operators use the COPYPRT procedure, they should be sure that the specified report is held on the output queue by using the HOLD PRT control command or the PRIORITY-0 parameter of the PRINTER OCL statement.

Note: The COPYPRT procedure does not copy spool print records that were created using operations that allow user programs to send their own data stream to the printer.

You can save the disk file used with the COPYPRT procedure on diskette and print it later. See the *System/36 Environment Reference* book for more information about the COPYPRT procedure.

Printing Output by Forms Number

Output in an output queue is printed on a first-in/first-out basis, with no regard to forms number. If, for example, two or more jobs use different forms numbers, the system operator controlling a printer has to change the forms often. The system allows operators to control the printing of output using different forms numbers.

The system has two methods of printing by forms number. Using the START PRT control command, you can print by a specific form, or by groups of forms. This method allows operators to print all reports that use the same forms number together. For example, you can print all jobs in the output queue with special invoice forms together.

To print by groups of forms (using the FORMS parameter of the START PRT command), the spool writer prints all entries in the output queue using the forms on the printer. When the spool writer finishes printing the entries, it prompts you to change forms in the printer. This method reduces the number of times you have to change the paper in a printer.

To print by a specific forms number, the spool writer prints only those entries that have the specified forms number. When there are no more

entries in the output queue with that forms number, the spool writer stops printing.

For example, to print reports that require forms number A112 on printer P2, a user types the following command string:

```
START PRT,P2,A112
```

The spool writer prints only reports that required forms number A112 on printer P2. If there are no entries in the output queue with forms number A112 for that printer, the spool writer does not print any entries.

The START PRT control command in the *System/36 Environment Reference* book has more information about printing by forms type. See the *Printer Device Programming* book for information about assigning forms numbers.

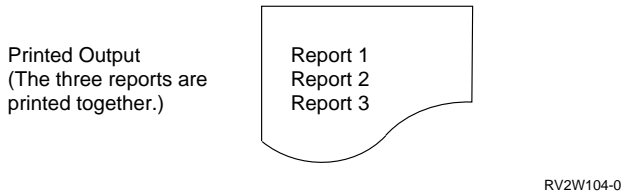
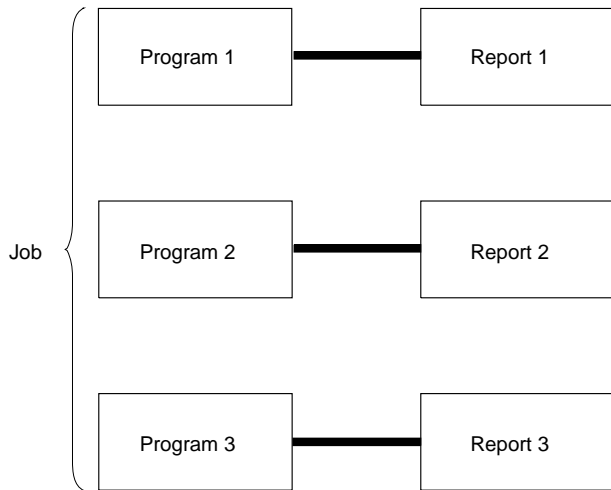
Combining Several Print Files in One Job

The system allows you to combine printer output from several programs into one print file. A **print file** is a device file that determines which attributes apply to the printed output. A particular printer may or may not support all of the attributes specified in a printer file. Users can create their own printer files. The programs must:

- Run in the same job
- Specify the same printer for the output

The first program of the job creates the print file, and the remaining programs in the job add output to the same print file.

For example, you have three order-entry programs within a job, and each program produces one report. You can have one print file instead of three separate print files to allow all reports produced by the order-entry programs to be printed at the same time. The following figure shows how this process works.



Use the CONTINUE parameter of the PRINTER OCL statement to combine multiple reports into one print file. See the *System/36 Environment Reference* book for more information about the PRINTER OCL statement.

Note: The size of the print file is determined by the record length of the first program's print file. If any program's printed output has a different length, the first program's record length must be equal to or greater than the longest record length required in the job while the CONTINUE parameter is in effect. For more information, see "CONTINUE-YES Processing" on page 17-11.

Assigning the Delayed Status to Printed Output

Use the DEFER parameter of the PRINTER OCL statement to specify the delayed status of a spooled file. The delayed status indicates whether a spooled file can begin printing before the file is completed by the program. Normally, the output is not printed until it is complete (when the job ends or the program closes the spooled file).

Specify DEFER-NO to print the output when there is data to print. For example, a 60-page report can begin printing after the first page is complete. When the program has created page 60, 20 pages of the report may already have been printed.

Using DEFER-NO can slow your printers. No spooled files can print until the spooled file using the DEFER-NO parameter has been printed. If you use the default value (DEFER-YES), other spooled files can begin printing while yours is being spooled.

Assigning Priorities to Printed Output

Use the PRINTER OCL statement to assign a priority to printed output. You can assign printed output a priority of 0 through 5. Output with a priority of 5 prints first. The system assigns output a priority of 1, if you do not assign a priority. Output with a priority of 0 means that the output is held on the output queue until an operator releases the output by typing the RELEASE control command. The PRINTER OCL statement in the *System/36 Environment Reference* book has more information about assigning priorities to printer output.

The following table shows how the System/36 environment printed output priorities are mapped to the AS/400 printed output priorities:

System/36 Environment Priority	AS/400 System Priority
0	7 and HOLD (*YES)
1	7
2	6
3	5
4	4
5	3

Programming Considerations

The following sections describe the programming considerations for the use of print files in the System/36 environment, and provides programming considerations for printed output attributes.

Use of Print Files by the System/36 Environment

For every printer defined in the System/36 environment, a print file is created with the same name as the System/36 environment printer. For example, if P1 is a System/36 environment printer ID, the library named #LIBRARY will contain a print file named P1. When a System/36 environment application creates printed output, the System/36 environment support uses these print files to define some of the attributes of the spooled file.

Some of these attributes, such as separator pages, cannot be overridden using the PRINTER, FORMS, or WORKSTATION OCL statements. You can change the attributes of these print files with the Change Print File (CHGPRTF) CL command.

When you use the Create Printer Device (CRTPRTDEV) command to create a printer device, the printer file in #LIBRARY is created asynchronously as another job. The CRTPRTDEV command may complete before the printer file has been created. If you create a printer device, wait for the printer file to be created before trying to change it.

Note: The CHGS36 command creates the System/36 environment print files and changes those that exist. For this reason, changes you make to a System/36 environment print file may be lost when you run CHGS36.

The *Printer* column of the STATUS PRT display identifies the application program name for printed output. For example, if the Printer OCL statement looks like the following example, the *Printer* column will contain REPORT for this printed output.

```
// PRINTER NAME-REPORT,DEVICE-P1
```

If an application program creates multiple printed output files, the information in the *Printer* column

can uniquely identify the printed output. The information in the *Printer* column can also be used to identify the print file that the System/36 environment used when creating the printed output. The System/36 environment first looks for a print file name that matches the application program name for the printed output. If a print file is found (in the current library, #LIBRARY, or the job's library list) that matches the application program name for the printed output, the matching print file is used. If a print file is not found, the System/36 environment will use the print file for the device that the output was created for. For example, if the Printer OCL statement looks like the following example, the System/36 environment will search for a print file name REPORT.

```
// PRINTER NAME-REPORT,DEVICE-P1
```

If the REPORT print file is found, the attributes of the REPORT print file will be used for the printed output. If the REPORT print file is not found, the attributes of the P1 print file will be used.

Printed Output Attributes

The attributes of printed output are determined by merging the following information in the specified order:

1. Application-program-specified information
2. Printer OCL statement information
3. Printer information defined by the System/36 environment configuration in the CHGS36 CL command
4. Print file
5. Printer information defined by the LINES, PRINT, SET procedures, or FORMS OCL statement

For example, if you specify forms 0001 on the printer OCL statement, forms 0002 for the PRINT procedure, and forms 0003 for the print file, the forms specified on the printer OCL statement (forms 0001) are used.

Chapter 5. Library, File, and Folder Overview

This chapter discusses the placement of System/36 libraries, files, and folders in the System/36 environment.

A **disk** is a storage device made from a flat, circular sheet of metal with magnetic surfaces. Programs, files, libraries, and system work areas that the AS/400 system uses to process programs are stored on disk. A **disk drive** is the device that reads and writes information on a disk.

Comparison of System/36 and AS/400 Addressing Models

One of the basic differences between the System/36 and the AS/400 system is how the information on the computer is stored and addressed.

System/36 Addressing Model

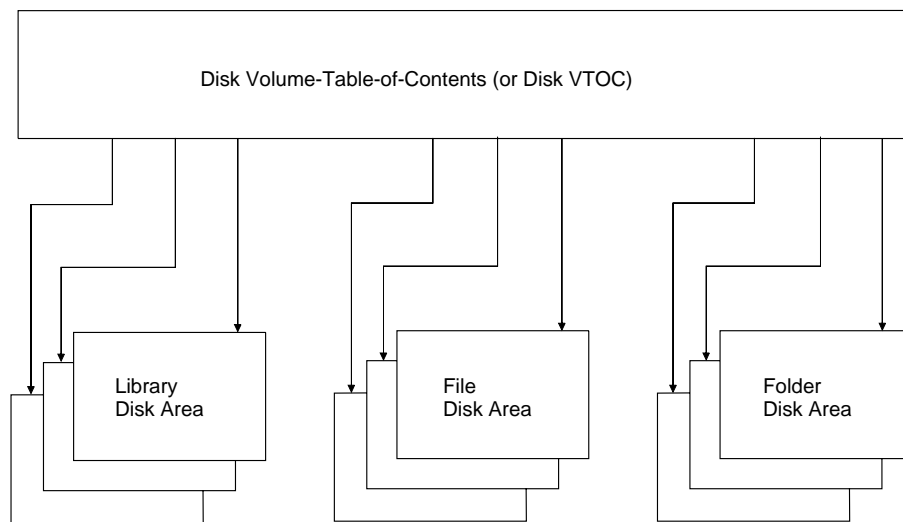
The System/36 has a disk volume-table-of-contents (disk VTOC) located at a fixed address on disk. The disk VTOC contains the addresses of areas of disk reserved for three types or classes of information, as follows:

- Libraries
- Files
- Folders

Each library, file, and folder stored on the system has a VTOC entry. Figure 5-1 shows System/36 addressing for libraries, files, and folders.

The disk area for a library or folder consists of a directory area followed by the area which the System/36 library management or folder management programs can subdivide for library or folder members. The directory area contains the relative address of each active member, along with member attributes. A document is an example of a folder member. Procedures, display formats, message members, and compiled programs are examples of library members.

The disk area for a file contains a set of related data records which can be accessed by an application program. For an indexed file, the disk area also contains an index which can be used to access the records by key. For an alternative index, the disk area contains only an index, with the index entries addressing records in a sequential, direct, or indexed file.



RSLW105-0

Figure 5-1. System/36 Addressing for Libraries, Files, and Folders

AS/400 Addressing Model

On the AS/400 system, the addresses of stored units of information are not externalized to the user. Units of information are called **objects**. Objects are contained in libraries. Each object is uniquely identified in the system by the object name, the library which addresses the object, and the type of information contained in the object (also referred to as the **object type**).

An AS/400 library is different from its System/36 counterpart in that its disk area contains only a directory to the objects that are addressed through that library. The disk storage for an object that is addressed through a library is physically a separate area of disk space from the library in which the object is located.

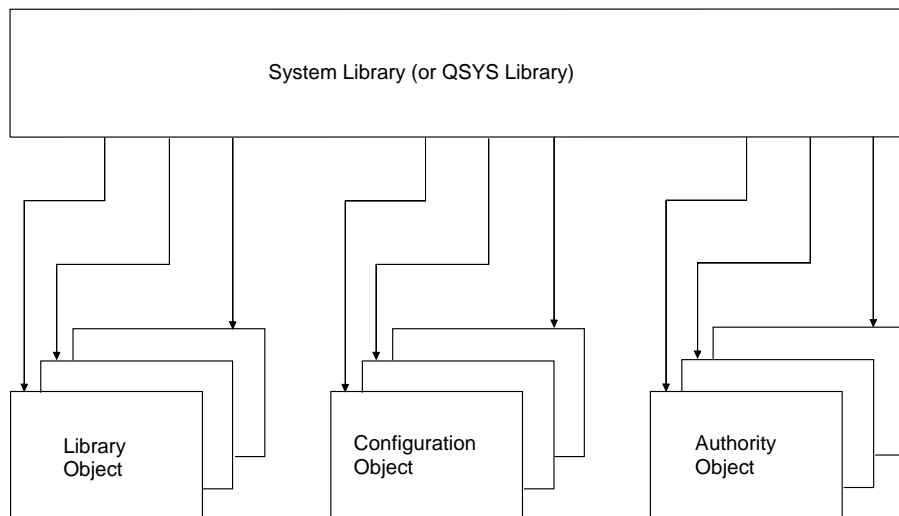
Figure 5-2 shows how the system library (QSYS) is used to satisfy the addressing of library objects, configuration objects (line descriptions, controller descriptions, device descriptions), and authority objects (user profiles and authorization lists). A **system library** is the library shipped with the system that contains objects, such as authorization lists and device descriptions created by a user; and the licensed programs, system commands, and any other system objects shipped with the system. The system identifier is QSYS. The system libraries in the System/36 environment includes a library for containing the System/36 environment (QSSP) and a library for non-IBM objects (#LIBRARY). An authorization list is a list

of two or more user IDs and their authorities for system resources. These objects are always addressed through the system library. For that reason, functions which create and delete these types of objects do not require you to specify a library name. Also, you cannot move any of these objects to another library. The system library is the closest AS/400 analogy to the System/36 disk VTOC.

Figure 5-3 on page 5-3 shows how an AS/400 library addresses each object contained in that library.

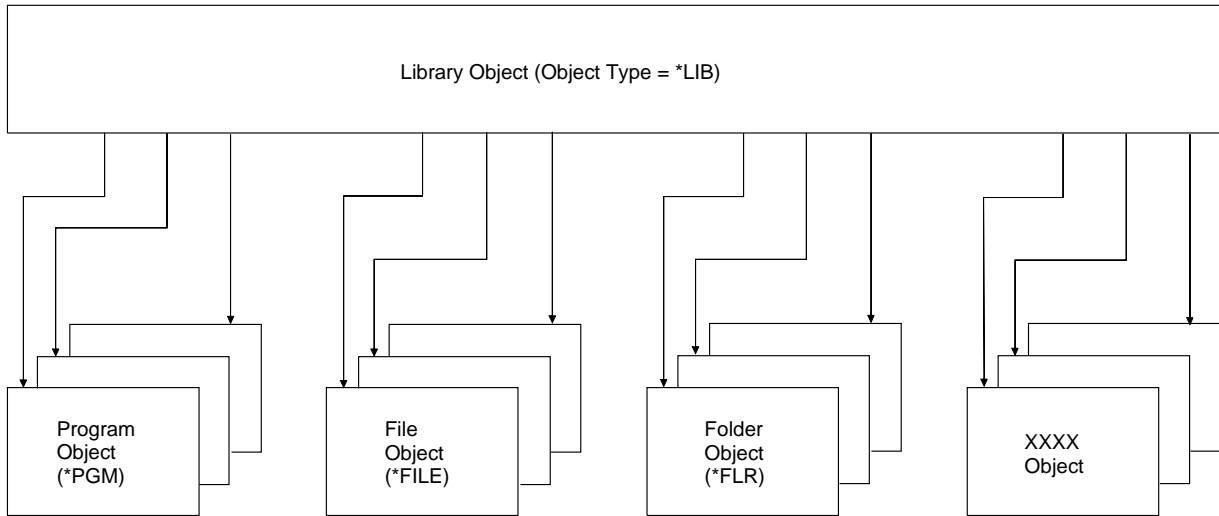
Note: A file object can only be addressed through a library. The System/36 environment normally addresses all data files and alternative index files in the current files library. The System/36 environment default files library name can be displayed using the Display System/36 (DSPS36) command, and can be changed using the Change System/36 (CHGS36) command or the Change System/36 Environment Attributes (CHGS36A) command. The session files library for a System/36 environment session can be changed using the FLIB procedure or the FILELIB OCL statement. The current files library for a System/36 environment job can be changed using the FILELIB OCL statement. See the *System/36 Environment Reference* book for a complete description of the FLIB procedure and FILELIB OCL statement.

The data for a sequential, indexed, or direct file is stored in a part of the file called a file member.



RSLW104-0

Figure 5-2. Objects Addressed through the AS/400 System Library (QSYS)



RSLW103-0

Figure 5-3. Addressing Objects through an AS/400 Library

Folders are also objects that must be addressed through a library. On the AS/400 system, a folder is a document library object (DLO).

The following table shows the mapping of some of the System/36 types of information to objects on the AS/400 system.

System/36 Object	AS/400 Object
Alternative index	Logical file
Compiled display formats	Display file
Compiled message member	Message file
Compiled program	Program
Data dictionary folder	Data dictionary and set of files
Date-differentiated files	Multiple member physical file
Direct file	Physical file
Document	Document library object (DLO)
Document folder	Folder document library object (DLO)
Folder	Folder document library object (DLO)
Indexed file	Physical file
Library	Library
Library #LIBRARY	Library #LIBRARY

System/36 Object	AS/400 Object
Load member	Program Display file Message file
Network resource directory (NRD)	DDM files
Procedure member	Member of source physical file QS36PRC
Sequential file	Physical file
Source member	Member of source physical file QS36SRC
Subroutine member	Program

System Information

This section describes system information stored on disk.

Library QSSP

When you install the System/36 environment on an AS/400 system, you restore library QSSP. QSSP contains the programs, procedures, utilities, files, and so forth, to allow System/36 applications to run on an AS/400 system.

With a new IBM-released version of the System/36 environment is installed, all objects in library QSSP are deleted and new objects restored. Because of this, do not place any user-created objects in library QSSP. If you must

change an IBM-supplied object in library QSSP, copy the object to library #LIBRARY and change the copied version.

System Library (#LIBRARY)

#LIBRARY contains user-created procedures, programs, messages, and so forth. Because the system searches #LIBRARY when locating objects, use it to contain objects common to many System/36 environment users.

User Information Stored on Disk

This section describes the contents of user information stored on disk.

Output Queues

Output Queues contain printed output stored on disk for later printing. For more information about output queues, see “Print Spooling” on page 4-3.

Job Queue

The job queue contains information about jobs waiting to be run. For information about the job queue, see “Using the Job Queue” on page 18-8.

Journal Files

Journal files are created when you are journaling system events. For more information, see the *DB2 for OS/400 Database Programming* book.

Licensed Program Libraries

The licensed program libraries contain the programs necessary to compile your programs or use the utility programs. The system stores the display formats, messages, procedures, and programs needed by the language compilers and the utility programs in these libraries. Each programming language or utility has its own library.

The following table shows a group of licensed program libraries:

Licensed Program	Library Name
COBOL	#COBLIB

Licensed Program	Library Name
Data file utility (DFU)	#DFULIB
RPG II	#RPGLIB
Screen design aid (SDA)	#SDALIB
Character generator utility (CGU)	#CGULIB
Development support utilities (DSU)	#DSULIB
Source entry utility (SEU)	#SEULIB

User Files

User files contain the data your programs need. For more information about files, see Chapter 7, “Files.”

User Libraries

User libraries contain the programming information for your jobs. For more information about libraries, see Chapter 6, “Libraries.”

User Folders

Folders contain members created by OfficeVision for OS/400. For more information about folders, see Chapter 8, “Folders and Data Dictionaries.”

Naming Conventions for Files, Libraries, and Folders

Use the following conventions when naming objects for use in the System/36 environment:

- Use 8 characters or less for names.
- Do not put quotation marks around special characters. System/36 environment internally puts quotation marks on names that have special characters. Specify names that have special characters as you did on System/36.
- System/36 environment code uses an algorithm to process object names. The algorithm does the following:
 - Checks that the name follows System/36 syntax rules. The first character must be uppercase A through Z, #, @, or \$. If the name fails the check, the system sends an error message.
 - Determines whether to put names that meet System/36 naming conventions in

quotation marks, using the following guidelines:

- If the name is a simple name, it is not internally marked with quotation marks. Simple names are:
 - First character (uppercase A through Z, \$, #, or @)
 - Remaining characters (uppercase A through Z, \$, #, @, underline, period, or digits 0 through 9)
- If the name is an extended name, it is internally marked with quotation marks.

Extended names can have any displayable character (code point greater than hex 3F) except an embedded blank, asterisk (*), single quotation mark ('), double quotation mark ("), or question mark (?).

Note: Folder names cannot be extended. Therefore, the information on extended names does not apply to folder names.

- If the name contains embedded blanks, asterisk, single quotation mark, double quotation mark, or question mark, the System/36 environment issues an error message.

Dynamically Created Files

The DISP-NEW keyword on the FILE statement for disk files requests that the system create new files as the applications use them. The System/36 environment creates files in the middle of open processing when the FILE statement requests DISP-NEW and the file does not already exist.

While processing FILE statements that have requested DISP-NEW, the system uses a system-wide index to prevent another System/36 environment job from attempting to create the same file.

Programming Considerations

This section describes programming considerations for managing disk storage.

Listing the Disk Volume Table of Contents

Use the CATALOG procedure to list the table of contents of the disk. The list shows the names of the files in the System/36 environment files library, libraries, folders, and data dictionaries. The *System/36 Environment Reference* book describes the CATALOG procedure.

Measuring Disk Activity

See the *Performance Tools/400* book for information about measuring disk activity.

Chapter 6. Libraries

This chapter describes the libraries for the System/36 environment, and explains how you can use libraries for your applications.

A **library** is an object that contains groups of other objects, such as files, message queues, and programs. You use libraries to find specific objects on the system.

Libraries for the System/36 Environment

The System/36 environment contains the following types of libraries:

- The **System/36 environment system library (QSSP)**. The QSSP library contains the IBM-supplied programs, procedures, files, and so on for the System/36 environment.
- The **System/36 environment user library (#LIBRARY)**. #LIBRARY contains user-written applications. Since the system searches #LIBRARY for objects, you can place applications run by many users in #LIBRARY.
- The **system library (QSYS)**. System library QSYS contains the IBM-supplied programs, files, messages, commands, and so on for the AS/400 system.
- Other **licensed program libraries**. These libraries contain the IBM-supplied programming support for the programming languages and utilities.
- Your **application libraries**. These are the libraries you create to store your programming information.

Library Names

A library name in the System/36 environment can be up to 8 characters long. The name must begin with an alphabetic character (uppercase A through Z, #, \$, or @). The remaining characters can be any combination of numeric, alphabetic, and special characters. The following names are not allowed:

#LIBRARY	F1
READER	DISK
PRINT	ALL
TAPE	

Avoid using the following characters for names since they have special meanings in procedures: commas (,), apostrophes ('), question marks (?), slashes (/), greater than signs (>), equal signs (=), plus signs (+), and hyphens (-). Avoid using names that begin with a Q because the system creates libraries beginning with a Q.

Do not place double quotation marks (") around names containing special characters (extended names). The System/36 environment internally places quotation marks around extended names. Extended names include any displayable character (code points greater than hex 3F) except embedded blanks, asterisks (*), single quotation marks ('), double quotation marks ("), or question marks (?). If a name contains these characters, it is invalid and you receive an error message.

You can create meaningful library names by abbreviating the name of the application that uses the library. For example:

Application	Library name
Accounts receivable	ACCTLIBR
Payroll	PAYLIB
Miscellaneous programs	PROGLIBR
Messages library	MSGLIBR

Group Libraries

Group libraries are a set of libraries collectively identified by one name. When you create a group library, specify a name for it. Each library in the group is named as an extension of the group name. The group library name and the library name are separated by one or more periods.

For example, a group library name is M.TEST. Libraries in the group are named M.TEST.1, M.TEST.2, and M.TEST.3. The group library in this example could also be considered part of a group named M.

Group libraries allow you to secure all libraries in the group. See Chapter 11, “Security,” for more information about securing group libraries.

Library Members

A **library member** in the System/36 environment is a named collection of source statements or data. A library contains the following types of library members:

- **Source members.** A source member in the System/36 environment is a member of source file QS36SRC in the specified library. Source members contain information that is input to another process (such as a compilation). High-level language source statements, message source, and display format source are source members.

On the AS/400 system, source members are contained in a physical source file named QS36SRC. A **source file** is a file created by the specification of FILETYPE(*SRC) on the Create Physical File (CRTPF) command or by using the Create Source Physical File (CRTSRCPF) command. A source file can contain source statements for such items as high-level language programs and data description specifications. Use the Source Entry Utility (SEU) procedure, the Development Source Utility (DSU) procedure, the Start Source Entry Utility (STRSEU), or the Start Programming Development Manager (STRPDM) CL commands to change source members.

- **Procedure members.** Procedure members contain operation control language (OCL) statements interpreted by the System/36 reader/interpreter. On the AS/400 system, procedure members are contained in a physical source file named QS36PRC. Use the SEU procedure, the DSU procedure, STRSEU CL command, or the Start Programming Development Manager (STRPDM) CL commands to change procedure members.
- **Load members.** Load members are the internal form of objects. Compiled programs, display formats, and message members are load members.

On the AS/400 system, load members consist of three different object types. Programs are objects of type *PGM, display files are objects

of type *FILE (subtype display file), and message files are objects of type *MSGF.

Load members are not compatible with System/36 load members and cannot be migrated back to System/36.

- **Subroutine members.** Subroutine members are the output from a process such as compilers, query, or data file utility (DFU). On System/36, program subroutines are combined to create load members.

On the AS/400 system, subroutine members consist of only program objects. Program subroutines are objects of type *PGM.

The record length to be used when the System/36 environment functions create the QS36PRC and QS36SRC source file can be specified when configuring the System/36 environment. This is discussed in Chapter 3, “Configuring the System/36 Environment.”

Library Member Names

A library member name can be up to 8 characters long, and must begin with an alphabetic character (uppercase A through Z, #, \$, or @). The remaining characters can be any combination of numeric, alphabetic, and special characters. The following special names are not allowed:

- DIR
- SYSTEM
- NEW
- ALL

Avoid using the following characters in member names since they have special meanings in procedures: commas (,), apostrophes ('), question marks (?), slashes (/), greater than signs (>), equal signs (=), plus signs (+), periods (.), and hyphens (-). Avoid using names that begin with a Q because the system creates library members beginning with a Q.

Using Libraries

You use a library as a directory to a group of related objects.

There are several advantages to using libraries:

- You can group certain objects for individual users. Doing so helps you to manage the

objects on your system. For example, place all the procedures that user MARY can use in library MARYLIB.

- You can group all objects used for an individual application into a single library. For example, place all your order-entry files and programs into an order-entry library called ORDLIB. In this way, you need to add only one library (containing your order-entry files) to the library list to ensure that all your order-entry files and programs are in the list (see “Library Lists” on page 6-7). Use this method when you do not want to specify a library name every time you use an order-entry file or program.
- You can ensure security. You specify which users have authority to use each library, and what they can do with the library. For example, you can specify that a user has read authority only for a specific library (see Chapter 11, “Security”).
- You can simplify save and restore operations. You can group objects that are saved and restored at the same time into one library. You can then save all the objects with one Save Library (SAVLIB) command, instead of saving each object individually with Save Object (SAVOBJ) commands.
- You can use different libraries for testing and for production.
- You can use multiple production files. For example, use one production library for source files and for creating objects, one for application programs and files, one for objects infrequently saved, and one for objects frequently saved.

Assigning Libraries

You can assign libraries to display stations, operators, or jobs in the following ways:

- **Sign-on library.** You can designate the library to be used by a display station when an operator signs on to that display station. If a library name is not specified on the sign-on screen, and the current library defined in the user’s profile has the special value *CRTDFT (indicating a user library is not defined), the System/36 environment sets the current library to the sign-on library specified by the SET

procedure. If the SET procedure was not run at the display station, the current library is set to the default System/36 environment sign-on library defined by the Change System/36 (CHGS36) CL command or by the Change System/36 Environment Attributes (CHGS36A) CL command. If the CHGS36 or the CHGS36A CL commands were not run to set the System/36 environment default sign-on library, the current library is set to #LIBRARY.

For more information on the SET procedure, see the *System/36 Environment Reference* book. For more information on the CHGS36 and CHGS36A CL commands, see Chapter 3, “Configuring the System/36 Environment.”

- **Current library.** A current library is associated with each job running. The current library is usually the sign-on library, but you can specify a session library or a job library as the current library.
 - **Session library.** You can designate a library for the session. The system searches the session library first for any members needed for a job. If the job information is not found in the session library, the system searches the System/36 environment user library (#LIBRARY). If the job information is not found in #LIBRARY, the system searches the library list (*LIBL). Use the SLIB procedure or the MENU control command to change the session library.
 - **Job library.** You can designate a library for each job or job step by using the LIBRARY OCL statement.

Sharing Libraries

Two or more programs can read from the same library at the same time. However, the following library functions require that only one program at a time uses the same library:

- Restoring a library using the Restore Library (RESTLIBR) procedure
- Removing all library members or all library members of one type using the REMOVE procedure
- Renaming a library using the RENAME procedure
- Deleting a library using the DELETE procedure

Changing Libraries in a Job

When using multiple libraries, you may need to change libraries in the middle of a job. Use the LIBRARY OCL statement to specify the library to be used for a job or a specific job step. The *System/36 Environment Reference* book has information about the LIBRARY OCL statement.

Specifying Authority for Libraries

You can specify authority both for user libraries and the system library in the System/36 environment. You can also specify the authority of objects created in a library. See Chapter 11, "Security," for information about authority for libraries and objects within libraries.

Making Backup Copies and Recovering from Errors

Use library backup and recovery to make sure your data is correct, and to minimize recovery time for each program. Consider the following points:

- Make a backup copy of a library whenever you change the library or library members significantly. Use the Save Library (SAVELIBR) procedure to save the library on diskette or tape. If a machine or program error occurs, you do not have to enter the changes again. Use the Restore Library (RESTLIBR) procedure to restore the library from diskette or tape.
- If an abnormal ending occurs when you are copying a member to a library (for example, using the TOLIBR procedure) the member may not have been copied correctly. If you are copying several members to a library, some members may not have been copied. Check each member to make sure it is correct. If the member is not correct or has not been copied, repeat the copy procedure.
- If a program ends abnormally when you are creating or changing a member in a library, the changes may or may not have been added to the member. To help you recover from abnormal endings, the source entry utility (SEU) automatically creates a work file. A **work file** in the System/36 environment is a file used for temporary storage of data being processed. See the *ADTS/400: Source Entry*

Utility book for more information about SEU and using the work file to recover from errors.

- You must be able to reconstruct your libraries if a failure occurs, to allow your applications to continue to operate.

For more information about backup and recovery, see Chapter 19, "Error Prevention, Detection, and Recovery."

Recovering from Damage to #LIBRARY

If the System/36 environment user library #LIBRARY is damaged, take the following steps to restore #LIBRARY from diskette or tape:

1. Have all users sign off the system and end all batch jobs.
2. Bring the system to a restricted state. For information on the restricted state see the End Subsystem (ENDSBS) command in the *CL Reference* book.
3. Rename library #LIBRARY to library #LIBRARY2.
4. Restore library #LIBRARY from diskette or tape.
5. If you have changed objects in #LIBRARY since you created the backup version you can recover the changed versions even though #LIBRARY has since become damaged. The process to recover the objects may take a *very* long time. Before recovering the objects, see the description of the Reclaim Storage (RCLSTG) CL command in the *CL Reference* book for information on the time required to recover objects that existed in #LIBRARY before it was damaged. Take the following steps to recover objects that existed in #LIBRARY before it was damaged:
 - a. Run the RCLSTG CL command to recover the objects that were in #LIBRARY when it was damaged.
 - b. When the reclaim storage function ends, library QRCL contains the system objects that were not in a library. Move the objects that existed in #LIBRARY before it was damaged to #LIBRARY from QRCL.

Note: Not all of the objects in library QRCL may have come from #LIBRARY.

You must determine which objects previously were in #LIBRARY when #LIBRARY became damaged.

6. Run the CHGS36 CL command and specify the following items:
 - a. From the Change S/36 Environment Configuration display, type 2 in the *S/36 display IDs* field.
 - b. Press the Enter key to return to the Change S/36 Environment Configuration menu.
 - c. When the Change S/36 Environment Configuration menu appears again, press the Enter key to update the System/36 environment configuration information.
7. Do an initial program load (IPL) for the system from disk.

Recovering from Damage to QS36ENV

***S36 in #LIBRARY:** If the System/36 environment configuration object (QS36ENV *S36) becomes damaged, take the following steps to restore the object from diskette or tape:

1. Have all users sign off the system and end all batch jobs.
2. Bring the system to a restricted state. For information on the restricted state see the description of the ENDSBS command in the *CL Reference* book.
3. Rename #LIBRARY to #LIBRARY2.
4. Restore #LIBRARY from diskette or tape.
5. Move objects that were not saved on the backup version of #LIBRARY from #LIBRARY2 to #LIBRARY.
6. Run the CHGS36 CL command and specify the following values:
 - a. From the Change S/36 Environment Configuration menu, type 2 in the *S/36 display IDs* field.
 - b. Press the Enter key from the Change S/36 Display IDs screen to return to the Change S/36 Environment Configuration menu.
 - c. When the Change S/36 Environment Configuration menu appears again, press the Enter key to update the System/36 environment configuration information.

7. Do an IPL for the system from disk.

Recovering from Damage to Library QSSP

If the System/36 environment system library QSSP is damaged, take the following steps to restore library QSSP from diskette or tape:

1. If system value QUSRLIBL or QSYSLIBL contains library QSSP, note the current setting of the system value and remove library QSSP from the list of libraries for the system value.
2. Have all users sign off the system and end all batch jobs.
3. Run an IPL for the system from disk, and specify that only the console be started.
4. When the system IPL finishes, delete QSSP from the disk. If the request fails because another job has a lock on library QSSP, use the Work Object Lock (WRKOBJLCK) CL command to determine the job that has a lock on library QSSP.
5. Follow the installation procedures for the System/36 environment in the *Software Installation* book.
6. If you changed the system values QUSRLIBL or QSYSLIBL, change them back to their original value.
7. Do an IPL for the system from disk and specify that all devices should be started.

Library Sector-Mode and Record-Mode Files

You can copy library members to disk, diskette, or tape files in sector-mode or record-mode format.

Sector-Mode Files: Sector-mode files contain library members that are stored in the internal format used by the AS/400 system. These files are created only by the FROMLIBR and SAVELIBR procedures or by the \$MAINT utility program. Using this format, you can migrate files to the System/36 environment that were created on the System/36.

You can use sector-mode files only with another AS/400 system. The *System/36 Environment Ref-*

erence book has more information about using sector-mode files.

Record-Mode Files: Records in a record-mode file have a special format. One statement from the member is contained in each record. All records have the same length, and the record can be from 40 to 120 characters long. The AS/400 system fills the record with blanks or truncates the statements in the library member to match the specified record length.

The first record in the file is a COPY statement that defines the library member. The last record in the file must be a CEND statement. The records in between contain the statements in the library member.

If you use the FROMLIBR procedure or the \$MAINT utility program to create a record-mode file from a source or procedure member, the COPY and CEND statements are placed in the file automatically. Otherwise, you must specify the COPY and CEND statements (for example, if a program you coded is to create a file that becomes a library member).

You can use record-mode files on other systems if the files are written as basic data exchange files. The *System/36 Environment Reference* book has more information about using record-mode files.

Programming Guidelines for Libraries

The following sections describe the functions you can do with libraries and library members in the System/36 environment, and they indicate the procedures you use to do the functions. The *System/36 Environment Reference* book has more information about these procedures.

Creating Libraries

Use the System/36 environment Build Library (BLDLIBR) procedure to create libraries on disk. BLDLIBR also creates the QS36SRC and QS36PRC source files. The BLDLIBR procedure always creates libraries in the system auxiliary storage pool (ASP). The authority to use the library defaults to the system value of QCRTAUT. The authority of objects created into the library, if

not specified in a command, defaults to the create authority of the library. See Chapter 11, “Security,” for information about the authorities for libraries and objects within libraries. To build libraries in user ASPs, use the Create Library (CRTLIB) CL command.

Creating Library Members

To create a library source or procedure member, use SEU or PDM. If you do not have SEU or PDM, use the \$MAINT utility program.

To create and change source members for display formats and menus, use the screen design aid (SDA) utility, source entry utility (SEU), or programming development manager (PDM).

To create load members, use compilers or utilities.

Listing Members and Library Information

When your library is on disk, use the LISTLIBR procedure to:

- List the library's directory
- List members from the library

When you have a library saved on diskette or tape, or you have one or more members in a diskette or tape file, use the LISTFILE procedure to list information about the members.

Saving and Restoring Libraries

Use the SAVELIBR procedure to save an entire library on diskette or tape. Use the RESTLIBR procedure to restore an entire library from diskette or tape to disk.

The System/36 environment libraries (#LIBRARY, QSSP, and QS36F) are saved when you save all nonsystem libraries using the following CL command:

- SAVLIB LIB(*NONSYS)

They can be restored using the following CL commands:

```
RSTLIB SAVLIB(*NONSYS)
```

or

```
RSTLIB SAVLIB(QSSP)
RSTLIB SAVLIB(#LIBRARY)
RSTLIB SAVLIB(QS36F)
```

#LIBRARY and QS36F are also saved when you save all user libraries using the following CL command:

```
SAVLIB LIB(*ALLUSR)
```

They can be restored using the following CL commands:

```
RSTLIB SAVLIB(*ALLUSR)
```

or

```
RSTLIB SAVLIB(#LIBRARY)
RSTLIB SAVLIB(QS36F)
```

Copying Libraries and Library Members

Use the LIBRLIBR procedure to copy library members from one library to another. Use the FROMLIBR procedure to copy library members from a library to a disk, diskette, or tape file. Use the TOLIBR procedure to copy library members from a file (either disk, diskette, or tape) to a library.

Securing Libraries

See Chapter 11, "Security," for information about securing libraries.

Listing Files

Use the CATALOG procedure to list information about files. The *System/36 Environment Reference* book describes the CATALOG procedure.

Renaming Libraries or Library Members

Use the RENAME procedure to rename a library, file, or folder. Use the CHNGEMEM procedure to rename a library member.

Removing Libraries or Library Members

Use the DELETE procedure to remove an entire library. Use the REMOVE procedure to remove library members.

Coexistence Considerations

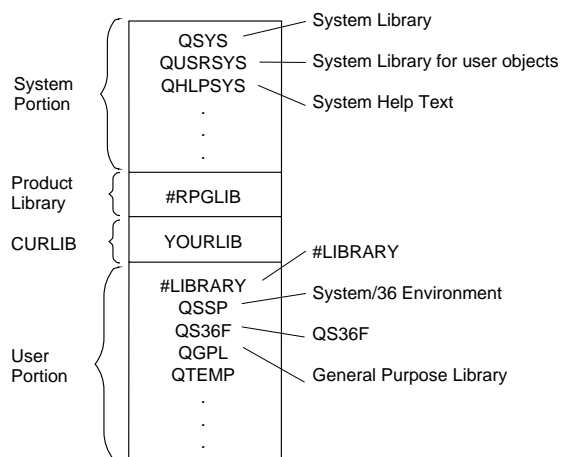
The following are considerations for using libraries in the System/36 environment.

Library Lists

A **library list** is an ordered list of libraries. Each job has a library list. When you use a library list, each library in the list is searched in the order of its occurrence until the system finds an object of the specified name and type. If more than one object of the same name and type exists in the list, the system uses the object from the library that appears first in the library list.

The library list illustrated in Figure 6-1 consists of the following:

- System part. The system libraries, specified by the QSYSLIBL system value, that contain objects needed by the system. Library QSYS is an AS/400 system library.
- Product libraries. Up to two product libraries used to support languages and utilities for the job.
- Current library. The current library for the job. This can be a duplicate of a library in the user part of the library list.
- User part. The objects referred to by the system's users and applications.



RV2W105-0

Figure 6-1. Example of Library List Parts

When you use the STRS36 or STRS36PRC CL commands, the System/36 environment saves the name of the current library. When you leave the System/36 environment (with the ENDS36 CL command, or at the end of the procedure started with the STRS36PRC command), the current library is restored to the value it had when you entered the System/36 environment. For example, if your current library is LIB1 and you enter STRS36PRC PRC(SLIB) CURLIB(#LIBRARY) when the STRS36PRC command ends, your current library is restored to LIB1.

Batch job specific information: When you start a batch job, the library list for that job is copied from the library list of the submitting job. This includes the system portion, the current library, and the user portion.

MRT job specific information: When you create a job in the System/36 environment, the system sets the current library from the Sign-On display, the Start System/36 (STRS36) command, the Start System/36 Procedure (STRS36PRC) command, or the Submit Job (SBMJOB) command. If no current library is specified, the current library is set to #LIBRARY. System/36 did not have library lists.

If #LIBRARY, QSSP, and the System/36 environment files library are not in the library list, they are added to the user part of the library list after the current library. If one of these libraries is already in the library list, its position in the list does not change. Library QSSP is shipped with a *USE authority to allow the objects to be used but not changed by users. #LIBRARY and the System/36

environment files library are created by the system. The authority for these two libraries is determined by the system value QCRTQUT.

Note: Never remove library QSSP from the library list while in the System/36 environment. This library contains objects necessary for performing System/36 environment functions, and results cannot be predicted if the library is removed.

On System/36, the current library for the MRT initiator's SRT job was copied to the MRT job.

In the System/36 environment, the current library for the MRT initiator's SRT job is copied to the MRT job (as was done on System/36). The rest of the MRT initiator's SRT library list is not copied to the MRT.

When you start a MRT job in the System/36 environment, the library list is set up as defined in the QS36MRT job description in library QGPL. The elements in the following list make up the default setting for this job description:

- The system portion of the library list is set to the libraries contained in system value QSYSLIBL. Use the Display System Value (DSPSYSVAL) CL command to display the value of QSYSLIBL, and the Change System Value (CHGSYSVAL) CL command to change the value of QSYSLIBL for all users.
- The current library is set to the current library of the job that started the MRT job.
- The user portion of the library list is set to the libraries contained in system value QUSRLIBL. Use the Display System Value (DSPSYSVAL) CL command to display the value of QUSRLIBL, and the Change System Value (CHGSYSVAL) CL command to change the value of QUSRLIBL for all users.
- Since this is a System/36 environment job, the system also does the following when starting a MRT job:
 - Adds the System/36 environment files library to the top of the user portion of the MRT job's library list, if not already in the list. If the files library is already in the library list, its position in the list does not change.
 - Adds QSSP to the top of the user portion of the MRT job's library list, if not already

in the list. If QSSP is already in the library list, its position in the list does not change.

- Adds #LIBRARY to the top of the user portion of the MRT job's library list, if not already in the list. If #LIBRARY is already in the library list, its position in the list does not change.
- If #LIBRARY, QSSP, and the System/36 environment files library are not in the library list, they are added to the user part of the library list after the current library. If one of these libraries is already in the library list, its position in the list does not change.

Search Order

This section describes the search order in the System/36 environment for:

- Subroutines
- Programs
- Procedures
- Migration commands
- Display files
- ICF files
- Print files
- Source and load members (other than PROCs and compiled code)
- Database files

Note: Chapter 20, “System/36 Environment National Language Support,” describes how the System/36 environment searches for display files, messages, menus, prompts, and so on.

The following objects are not discussed. The search order for these objects is always to the library specified or is managed by the product defined.

- Alternate indexes (located in the System/36 environment files library)
- Folders (defined and managed by office systems)
- Data dictionaries (defined and managed by the interactive data definition utility (IDDU))
- Documents (defined and managed by office systems)
- Mail logs (defined and managed by office systems)

Searching for Subroutines (Compiled

Callable Code): The System/36 OLINK procedure (to link edit subroutines into the system) is not supported for the System/36 environment. Instead, the system searches the //LOAD library when specified (or *CURLIB), followed by #LIBRARY and *LIBL to locate the subroutines when they are to be called.

Searching for Programs (Compiled

Callable Code): On System/36, for //LOAD, the system searches the library (specified or *CURLIB) and then #LIBRARY. For the System/36 environment, the system searches the specified library, #LIBRARY, and then the library list (*LIBL).

The // IF LOAD- and // IF SUBR- procedure control expressions do not search the library list. This situation can cause application errors. For example, you can use a special library to contain tailored code. You use the // IF test to determine if the requested tailored module is in that special library, and if it is, to load the module. If the module is not in the library, the system searches the application library (which might be *CURLIB) for the code.

Searching for Procedures: Procedures are stored in the QS36PRC file in a library. The search order is:

1. Specified library or *CURLIB
2. #LIBRARY
3. Library list (*LIBL)

Searching for Source Members: The System/36 environment follows the same search order used on System/36. The search order for source members depends on the procedure or command being processed. The procedures search only the library specified. If no library is specified, the procedure searches #LIBRARY.

Searching for Migration Commands:

There are control language commands that allow you to migrate objects restored from System/36 into the AS/400 objects. If the file name is specified (with an explicit name or *CURRENT), only that library is searched. If *LIBL is allowed, the job library list is searched to locate the object.

Searching for Database Files: Database files on the AS/400 system are put in libraries. For programs and utilities that are run within the System/36 environment, you can search for database files using the following methods:

- You can search for all files in the current files library.

The current files library, by default, is the session files library. It is set whenever a new job step is initiated. Within a job, the current files library can be changed with the File Library (FILELIB) OCL statement. See the FILELIB OCL statement in the *System/36 Environment Reference* book.

The session files library is the default files library in the System/36 environment configuration. It is set whenever a new session is initiated. Within a session, the session files library can be changed with the File Library (FLIB) procedure or the FILELIB OCL statement. See the FLIB procedure and the FILELIB OCL statement in the *System/36 Environment Reference* book.

The default files library is set when the System/36 environment is defined. It can be changed using the Change System/36 (CHGS36) command or the Change System/36 Environment Attributes (CHGS36A) command. For more information, see Chapter 3, “Configuring the System/36 Environment.”

- You can search for all files using the library list.

The libraries referred to in the library list are searched when the current library list search indicator is set to YES. The current files library is not searched when the current library list search indicator is set to YES, unless the current files library is in the library list.

The current library list search indicator is the session library list search indicator by default. It is set whenever a new job step is initiated. Within a job step, the current library list search indicator can be changed with the FILELIB OCL statement. See the FILELIB OCL statement in the *System/36 Environment Reference* book.

The session library list search indicator is the default library list search indicator in the System/36 environment configuration. It is set whenever a new session is initiated. Within a session, the session library list search indicator can be changed using the FLIB procedure or the FILELIB OCL statement. See the FLIB procedure and the FILELIB OCL statement in the *System/36 Environment Reference* book.

The default library list search indicator is set when the System/36 environment is defined. It can be changed using the CHGS36 command or the CHGS36A command. For more information, see Chapter 3, “Configuring the System/36 Environment.”

User Auxiliary Storage Pools for the System/36 Environment

The System/36 environment supports the use of auxiliary storage pools (ASP), but does not support creating libraries in ASPs. To create a library in an ASP, you can use the Create Library (CRTLIB) command. The Build Library (BLDLIBR) procedure always creates a library in the system storage pool.

For more information on ASPs see the following publications:

- *Backup and Recovery – Advanced*
- *CL Reference*
- *DB2 for OS/400 Database Programming*

Moving from System/36 to the System/36 Environment

You can migrate libraries created on System/36 to the System/36 environment. You can use SAVELIBR or FROMLIBR procedures on the System/36 to save procedures or source members to tape or diskette. To restore these procedures or source members on an AS/400 system, use the Restore System/36 Library Member (RSTS36LIBM) command.

Chapter 7. Files

This chapter describes related sets of records called files. It describes how to use, process, and organize files, and how to use file attributes, block records, and share files.

Using Files

This section describes how to do the following:

- Create files
- Name files
- Specify a file for a program
- Place data in files
- Remove files from disk
- Copy files
- Print or display files

See the *System/36 Environment Reference* book for more information about the commands and procedures described in the following sections.

Creating Files in the System/36 Environment

When you create files in the System/36 environment, the system stores them in the current files library of the System/36 environment. To change the system default files library, use the Change System/36 (CHGS36) command or the Change System/36 Environment Attributes (CHGS36A) command. The default library name is QS36F. For information about the CHGS36 and CHGS36A commands, see Chapter 3, "Configuring the System/36 Environment."

The default files library becomes the session files library when you start the System/36 environment. When you enter statements from the keyboard, the session files library and current files library are the same. When starting a procedure, the session files library is made the current files library for the procedure. Use the FILELIB OCL statement and FLIB procedure to change the current and session files library. For more information, see the *System/36 Environment Reference* book.

You can create files on disk using:

- The BLDFILE procedure. For example, if you want to create a sequential file named NEWFILE, and you have enough space for

one hundred 256-byte records, enter the following statement:

```
BLDFILE NEWFILE,S,R,100,256,,T
```

- The data file utility (DFU) to automatically create sequential, direct, and indexed files.
- A program that does output operations. The FILE OCL statement can have DISP-NEW or no DISP parameter specified. For example, if you want to have your program create an output file named OUT1 with the label OUTPUT and you have enough disk space for 10 blocks of data, enter the following FILE OCL statement:

```
// FILE NAME-OUT1,UNIT-F1,LABEL-OUTPUT,BLOCKS-10,  
//      RETAIN-T,DISP-NEW
```

- The RESTORE, TRANSFER, or TAPECOPY procedure to copy files from diskette or from tape to disk.
- The COPYDATA procedure to copy files from disk to disk.
- The Query data entry function.
- The interactive data definition utility (IDDU).
- From a remote location, using the File Transfer procedure or distributed data management (DDM) procedure.

Select one of the following values when you specify the size of files being created:

- The number of records the file will contain
- The number of blocks of disk space for the file (1 block is 2560 bytes)

For example, you can create a sequential file with enough room for five hundred 256-byte records, or with 50 blocks of disk space allocated for it.

Naming a Physical File

The System/36 environment requires you to uniquely identify each file through a file name and a file label. The file name indicates how the program refers to the file. The file label indicates how the system refers to the file on disk. A file name or file label can be up to 8 characters long and must begin with an alphabetic character (A through Z, #, \$, or @). Characters (other than the

beginning character) can be any combination of characters (numeric, alphabetic, and special).

Do not name files ALL. Apostrophes ('), question marks (?), asterisks (*), and quotation marks (") are not allowed. Do not use the following characters because they have special meanings in procedures: commas (,), slashes (/), greater than signs (>), equal signs (=), plus signs (+), and hyphens (-). If you use any of these characters, the System/36 environment puts double quotation marks around the file name and then uses this expanded name as the actual name on disk. For more information, see "Naming Conventions for Files, Libraries, and Folders" on page 5-4.

Create meaningful file labels by abbreviating the name of the application that uses the file. For example:

File Label	Application or Type of File
ACCTRECV	Accounts receivable
CUSTMAST	Master customer file
CUSTORDS	Customer orders

NAME and LABEL are parameters on the FILE OCL statement. Specify the LABEL parameter only when the name of the file on disk is different from the name used in the program. If you do not specify the LABEL parameter, the program uses the name specified as the NAME parameter. For example, in the COBOL coding and the FILE OCL statement shown below, the file name used by the program is ITEMMAST, but the file label is INVMAST (the actual name of the file on disk).

Following is the COBOL program segment:

```
INPUT-OUTPUT SECTION.
FILE-CONTROL.
    SELECT INPUT-FILE ASSIGN TO DISK-ITEMMAST.
```

Following is the FILE OCL statement:

```
// FILE NAME-ITEMMAST,LABEL-INVMAST
```

Using File Dates: You can assign the same file label to more than one file if each file has a different creation date. The **creation date** is the system date when an object is created. A **system date** is the date assigned in the system values when the system is started. During system configuration, you can specify that the system prevent a new disk file with a different date from being created if it has the same label as an existing file. Date differentiated files are treated as separate members in a single file on the AS/400 system.

See Chapter 3, "Configuring the System/36 Environment," for more information.

When creating these date-differentiated files, make sure the following attributes of the date-differentiated file are the same:

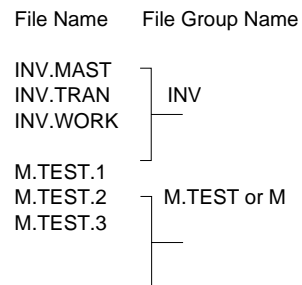
- Record length
- Delete-capable/nondelete-capable
- Duplicate keys allowed/not allowed
- Keyed file/nonkeyed file
- Key position and length (if keyed file)
- Initial allocation
- Form of initial allocation (blocks or records)
- Extend value
- File size
- File type

If these attributes are not the same, an error appears when you try to create a new file for input or output.

Using Group Files: A group file is a set of files identified by a file name with identifiers separated by one or more periods. The characters preceding a period identify the file group.

The limit of 8 characters for a file name applies to names for group files. The period is one of the 8 characters. A group name should not be more than 6 characters long plus a period (.). A 7-character group name plus a period (.) does not allow more characters to differentiate between files in a group.

The following figure shows examples of file names that identify group files.



RSLW068-0

You can save, restore, or delete all the files in a group file in one step by using the SAVE, RESTORE, or DELETE procedure. You can use group files for all files used by an application or portion of an application.

You can secure group files. See Chapter 11, “Security,” for more information about securing group files.

Renaming a File: Use the RENAME procedure to rename files. Do not rename files used by other programs, procedures, or utilities. If you rename a file used by several procedures, all those procedures would need to know the new name to be able to refer to the file. See the *System/36 Environment Reference* book for more information.

Specifying a File in a Program

To use a file in a program, you must specify the file name. You can also specify the file label in the program. When the program is compiled, the compiler takes file information from the file description in the program. The system uses this information to define the record formats to be processed and to open the file.

The file description in the program contains a description of the following:

- Record formats including field names, data types (numeric, alphanumeric, or character), and field lengths
- Processing method
- File organization as specified in the high-level language program

All records in a file are fixed length and can have the same or different field descriptions. You can subdivide a field or redefine the field for processing in your high-level language program. Records are described in Chapter 12, “Designing Records.”

Your high-level language program must open a file before issuing a request to read, write, change, or delete records in a file. A request to open a file allows the program to process the file. A request to close a file prevents the program from processing the file. In some high-level languages you must code the request in your program. In other languages, the open or close operation is performed automatically.

Some high-level languages allow you to specify file parameters in the program; others require a FILE OCL statement to specify these parameters. Before a file is opened, the system checks to see

if a FILE OCL statement has been entered to override the file specified in the program. The file description in the program is merged with the parameters on the file-overriding FILE OCL statement.

See the *System/36 Environment Reference* book for more information.

Placing Data in Files

Place data in files using one of the following methods:

- High-level language program. See the *System/36 Environment Reference* book for more information.
- DFU. The *ADTS/400: Data File Utility* has more information.
- Query. *Query/400 Use* book has more information.
- Diskette to disk. Use the TRANSFER or the RESTORE procedure.
- Tape to disk. Use the TAPECOPY or RESTORE procedure.
- COPYDATA procedure as described in “Copying Files.”

Removing a File from Disk or Diskette

Use the DELETE procedure to remove a file from disk or diskette.

Securing Files

Secure your files using the AS/400 commands. You should decide who can do the following:

- Read data from the file.
- Create or delete the file.
- Change data in a file.

See Chapter 11, “Security,” for more information about how to secure files.

Copying Files

The COPYDATA procedure copies a file on disk to another file on disk. During the copy operation, the COPYDATA procedure can do the following:

- Remove deleted records
- Include or omit specific records

- Create the copied data in the same or a different file organization

Printing or Displaying Files

You can use the following methods to print or display files:

- High-level language program. See the appropriate language book.
- DFU. The *&dfulist*. has more information.
- Query/36. See the *Query/400 Use* book for more information.
- LISTDATA procedure. Use the LISTDATA procedure to selectively list data from a file, no matter what its organization is. The data can be printed or displayed.

Use the LISTDATA options to:

- Print or display the records in either character or hexadecimal representation.
- Specify the maximum number of records to be printed.
- Make the record length either larger or smaller than the original record length.
- Include or omit selected records.
- LISTFILE procedure. Use the LISTFILE procedure to list the contents of files or libraries from disk, diskette, or tape.

Store Deleted Files in Cache

System/36 environment has a configuration option to store files being deleted in a cache. When you create a database file, the System/36 environment searches the cache for a file with the same attributes, inserts the file into the library, and removes it from the cache. Using a file stored in the cache is a faster method of creating a file.

Note: When a file that is stored in the cache is reused, its creation date and time and record format name are not updated.

Database files meeting the following requirements are stored in the cache:

- File only has one member
- File is not an alternate index file or a remote file (DDM)

- File is not used by an alternate index or a logical file
- File is not being journaled
- File is not linked to a data dictionary
- File was created by a System/36 environment function (for example, the BLDFILE procedure)

Files stored in the cache do use additional storage. You can monitor this storage by displaying the System/36 environment values (DSPS36 command) and watching the number of kilobytes of storage used by the cache. To remove all files from the cache, have a user with QSECOFR authority call the QEXCLNCI program. The QEXCLNCI program submits a request for a database server job to remove all the files from the cache. The system also submits a request for a database server job to remove all the files from the cache when the system is started or when the System/36 environment value for storing the deleted file in cache is set to N (No) by the Change System/36 (CHGS36) command or the Change System/36 Environment Attributes (CHGS36A) command.

Every library has an Object Information Repository (OIR). The OIR is where information about the objects contained in the library is kept. When a file is deleted and stored in the cache, the OIR entry for that file is not removed. The OIR entry is replaced if the file is subsequently reused from the cache with the same name in the same library. Otherwise, the OIR entry persists. If enough of these unused OIR entries accumulate for a library, users may be prevented from adding new objects to the library.

The QEXCLNCI program can be called by a user with QSECOFR authority to remove these unused OIR entries. The program must be called with a single parameter specified. When a library name is specified for the parameter, the program attempts to remove the unused entries from the OIR of that library. When *ALL is specified for the parameter, the program attempts to remove unused entries from the Object Information Repositories of every library on the system. To reduce conflicts with other system activity, this function of the QEXCLNCI program should be performed during times of low system use.

File Organization

This section describes sequential, direct, and indexed file organization.

File organization is the way records are placed in a file. The system allows sequential, direct, and indexed file organizations. Records in sequential files are in the order in which they were entered into the file. Records in direct files have a record number that specifies the location of a record in relation to the beginning of the file. Indexed files have the key and position of each record recorded in a separate portion of the file called an index.

Sequential File Organization

A sequential file is a file whose records are in the sequence they were written.

The program that creates the file arranges records in the same sequence they were placed in the file. The program places the first record in the first position in the file, the second record in the second position, and so forth.

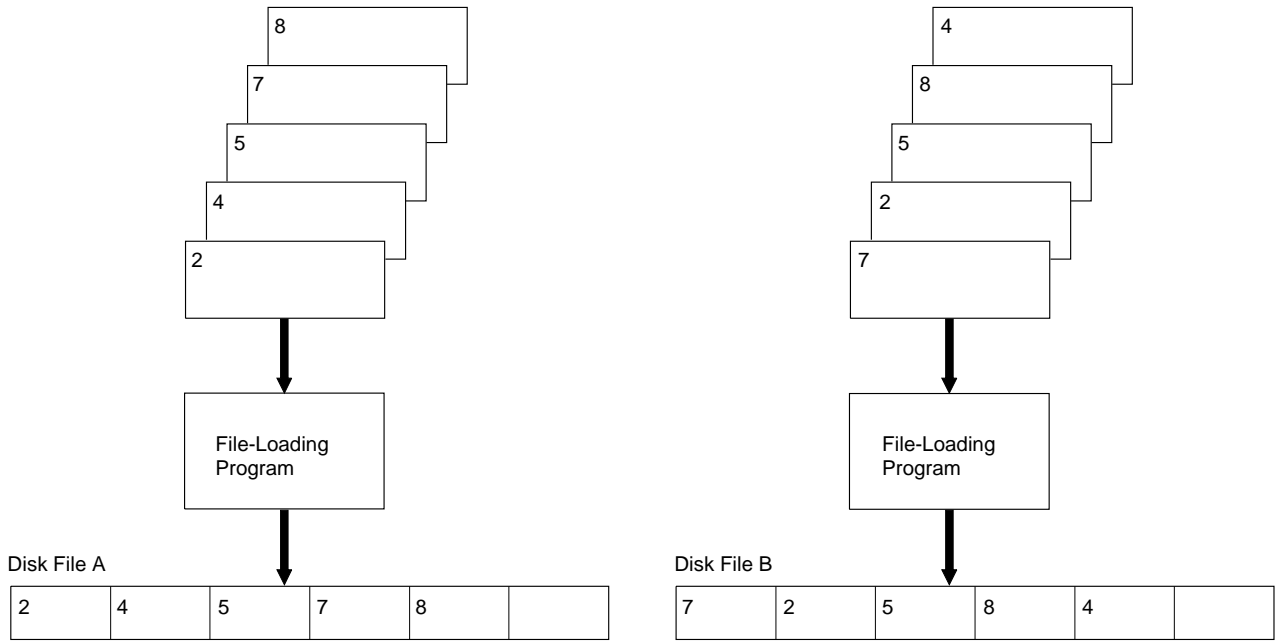
Figure 7-1 shows the creation of two sequential files. The file loading program reads the records on the left in the following sequence: 2, 4, 5, 7, 8. Because the records are loaded onto a disk in this same sequence, the disk file is a sequential file. The file loading program reads the records on the right in the sequence: 7, 2, 5, 8, 4. The records are loaded onto a disk in the same sequence, so this is a sequential file also.

The loading program can load records in an ordered or unordered sequence. In an ordered sequence, the file loading program arranges records in ascending or descending sequence, based on the value of a control field. The records in Disk File A in Figure 7-1 are loaded in an ordered sequence. The records in the Disk File B in Figure 7-1 are loaded in an unordered sequence. The way you write the file loading program (*not* the sequence of the input source records) determines whether the program loads a file in an ordered or unordered sequence.

A sequential file takes less space than a direct file or an indexed file. Direct files have gaps in the file for missing records. Indexed files use extra space for the index. See the *DB2 for OS/400 Database Programming* book for information on determining how much space a sequential file requires.

When a record is added to a sequential file, it is placed at the end of the file, not between existing records. In Figure 7-1, when the file loading program adds record 6, it is placed after record 8, not between records 5 and 7. If it is important that record 6 appear between records 5 and 7, you can run a sort program to arrange the records into a new ordered sequence.

You can process sequential files consecutively, randomly by relative record number, or by the generalized processing method (both consecutively and randomly). See "File Processing Methods" on page 7-13 for more information.



RSLW013-1

Figure 7-1. Organization of a Sequential File

Direct File Organization

In a direct file, records are loaded into assigned places within the file. Normally, the file loading program uses the value of a field from the input record to point to the position in the disk file where the record should be placed. The pointer (the value of a field from the input record) is called the **relative record number**. The relative record number identifies the position of the record relative to the beginning of the file. For example, if the decimal value of a relative record is 8, that record is placed in the eighth record position in the file.

Regardless of the sequence in which the file loading program reads the records, it places the records in the direct file at the proper location. For example, Figure 7-2 shows two sequences of input records.

The input records on the left in Figure 7-2 are in the sequence: 2, 4, 5, 7, 8. The input records on the right are in the sequence: 7, 2, 5, 8, 4. However, in both cases, the file loading program places the records into the same positions in the direct file. Spaces are left for records 1, 3, and 6. When records are added to a direct file, they are placed in the spaces that were left for them. For

example, record 6 is added between records 5 and 7.

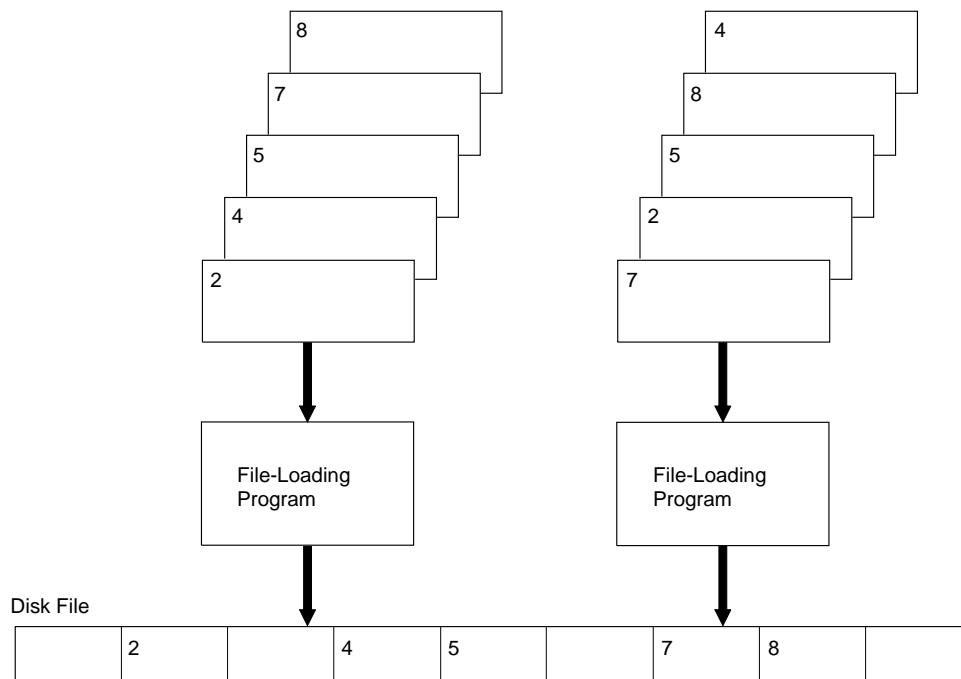
When the file loading program creates a direct file, all record positions in the file are initialized to blanks (hex 40s) if the file is not delete-capable. If the file is delete-capable, unused record positions are marked as deleted records.

All relative record numbers are decimal and start with 1.

The relative record number is based on a value stored in the record or is a calculated value calculated using the values of fields in the record.

Direct file organization can use a great deal of disk storage space if the formula you choose for calculating record positions leaves many unused record positions in the file. For example, if your calculations create relative record numbers from 1 to 1000, space is reserved in the file for 1000 records. If you use only 100 records, the space for the other 900 records is not used.

Occasionally, when you use formulas to determine relative positions, more than one record has the same calculated position in the file. These records are called **synonym records**.



RSLW014-1

Figure 7-2. Organization of a Direct File

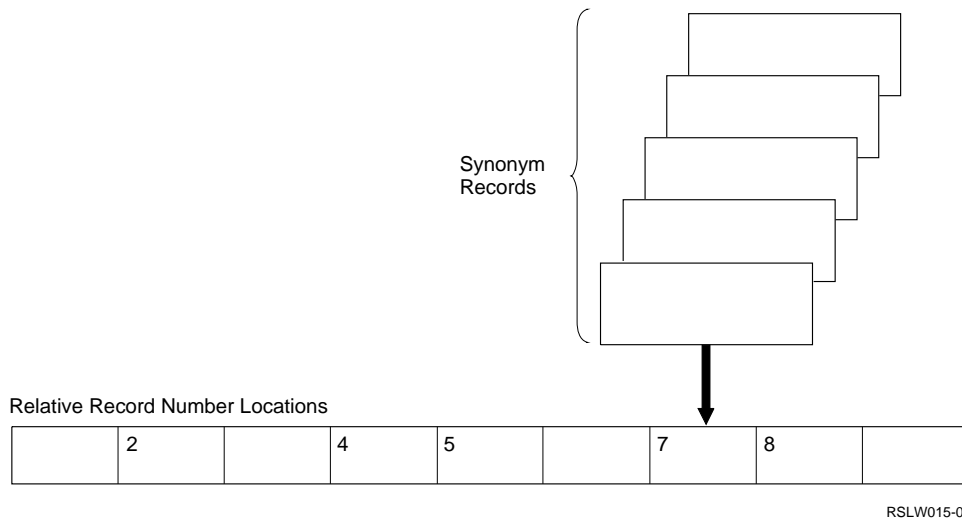


Figure 7-3. Synonym Records

Figure 7-3 shows an example of synonym records.

Only one synonym record can be stored in a calculated record position. Therefore, if you use formulas to create direct files, you must plan how to store and retrieve synonym records from various locations within the file. See Appendix A, “Access Algorithms for Direct Files” for more information.

You can access direct files consecutively, randomly by relative record number, or by the generalized processing method. See “File Processing Methods” on page 7-13 for more information.

Indexed File Organization

In an indexed file, the file loading program arranges records in the sequence they were written. As the records are added, an index, called the **primary index**, is built for the file. Thus, an indexed file contains two parts: an index and the sequentially organized records.

The index is a table containing an entry for each record in the file. Each index entry identifies a record by the value of its key and locates the

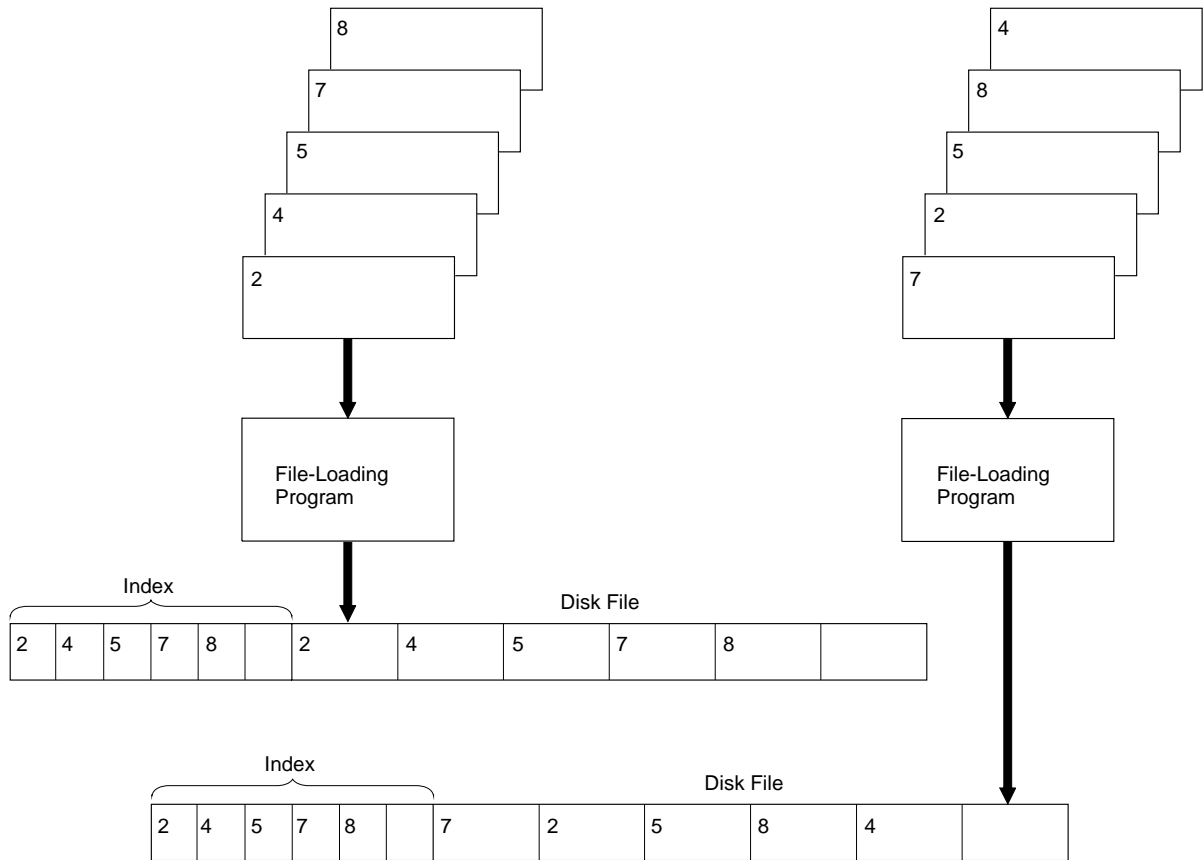
record by its address in the file. The key (also called the key field or the record key) is the portion of the record containing information that identifies the record. For information about the rules for forming keys, see “Using Index Keys” on page 7-11.

With an index, a program processes required records by referring to the key of the record. For example, if you have an indexed file with order records containing customer number, amount ordered, and balance due, your program can use the customer number as the key to find a record for a particular customer without reading any other records.

Figure 7-4 on page 7-9 shows how indexed files are organized.

You can process indexed files consecutively, sequentially by key, randomly by relative record number, randomly by key, or by the generalized processing method. See “File Processing Methods” on page 7-13 for more information.

See the *DB2 for OS/400 Database Programming* book for information on how to tell how much space an indexed file requires.



RSLW016-1

Figure 7-4. Organization of an Indexed File

Multiple Indexes for a File

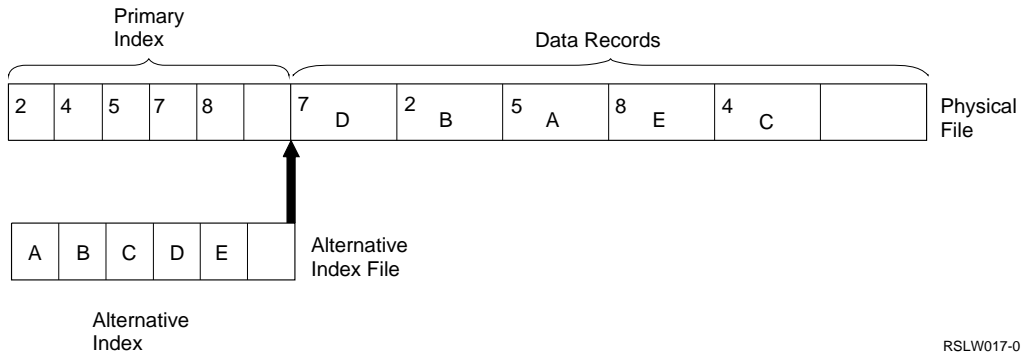
A **physical file** is a file that contains data records. After you create a physical file, you can create indexes (called **alternative indexed files**) for that file without creating new data records (see Figure 7-5).

You can create any number of alternative indexed files. The files you create to define each of the logical files are called alternative indexed files. You can create alternative indexed files for any file type: sequential, direct, or indexed. You can create alternative indexed files with the BLDINDEX procedure. The *System/36 Environment Reference* book has more information about the BLDINDEX procedure. Names for alternative indexed files must follow the same rules as physical files.

Figure 7-5 shows an example of multiple indexes for an indexed file.

All the indexes point to the same data records in the file, but each index can use a different portion of the record as the key. The field(s) used as the key in an index can overlap the field(s) used as the key in other indexes. Therefore, you can process records from the file in various sequences, depending on which index you use. For example, for a personnel file, you can use the employee number as one key and the department number as a second key. A high-level language program treats a multiple-index file the same way it treats a normal indexed file.

When changes are made to the physical file, all indexes (primary and alternative) are changed as required to reflect the change (if restrictions for duplicate keys or changing keys are followed). For example, if you use the employee number as one key and the department number as a second key, changes you make to the file (while using the index based on the employee number) are automatically made in the index based on the department number. You must have a FILE OCL statement for each index that the program uses.



RSLW017-0

Figure 7-5. Multiple Indexes for an Indexed File

Note: System/36 would not allow the user to update the primary index fields. The OS/400 program does not differentiate between indexes, and all index fields can be modified.

Using Multiple-Index Files: When you use multiple-index files, be aware of the following factors:

- Unlike the System/36, date-differentiated alternative indexed files cannot be created. However, an alternative indexed file can be built for a physical file that is date-differentiated.
- You cannot delete the physical file if alternative indexed files are specified for it. The RETAIN-S parameter of the FILE OCL statement is not allowed for the physical file or for the alternative indexed files.

To delete a physical file, you must first delete all alternative indexed files. Use the DELETE procedure to delete the alternative indexed files. You can delete a physical file and all indexes by using the file group naming convention and the DELETE procedure to delete the entire group. See "Using Group Files" on page 7-2 for more information about file groups.

- You cannot rewrite (overlay using DISP-OLD on the FILE OCL statement) an existing physical file if it has alternative indexed files. Before you can overlay the data in the physical file, you must delete the alternative indexed files.

You cannot rewrite an alternative indexed file. You can only add to or change alternative indexed files.

- You can specify a multiple-index file in a procedure substitution expression that retrieves

information about the file's size (?F'S'? or ?F'A'?). If the specified file is an alternative indexed file, the system can substitute the number of blocks or records allocated for the physical file.

- You can change key values on change operations. When you change an index, consider the following:
 - In a COBOL program, you can change a key that has been used to retrieve the record. Therefore, do not retrieve the record by the field to be changed.
 - In an RPG II program, between the record retrieval (CHAIN, READ, READE, or READP operation) and the record change, do not make another retrieval operation to the file (this ensures that you change the correct record).
- When you change a record, and a key for any index is changed or a record is added, duplicate keys can occur. The duplicate key can cause an error message to appear for an index the program is not using. The error message occurs only for indexed files that do not allow duplicate keys. You indicate whether an indexed file allows duplicate keys when you create the file. For more information about specifying duplicate keys, see "Specifying Duplicate Keys" on page 7-11.

Saving Multiple-Index Files: When you use the SAVE procedure to save an indexed file, the system saves the data and a description of the primary index.

When you save an alternative indexed file, the system saves only a description of the index. It does not save the index itself or the data in the file. Saving the physical file does not save the

alternative indexed files. You can save a physical file, and all indexes for that file, using the file group naming convention and the SAVE procedure to save the entire group. When you use the SAVE procedure to save a file group, the system saves the physical file, and then saves the alternative indexed files. See “Using Group Files” on page 7-2 for more information about file groups.

The description of the SAVE procedure in the *System/36 Environment Reference* book has more information about saving multiple-index files.

Restoring Multiple-Index Files: When you use the RESTORE procedure to restore an indexed file, the system restores the data and rebuilds the index from the description that was saved.

When you restore an alternative indexed file, the system rebuilds the alternative indexed file from the description saved. The data used for this rebuild is the data in the physical file that has the same date as the alternative indexed file being restored. Therefore, when you restore the indexes individually, the physical file must be restored before any alternative indexed files are restored. If the physical file is not on disk, and you are restoring an alternative indexed file for the file, an error message appears.

When you use RESTORE ALL to restore a multiple-index file, the system restores the physical file and then restores the alternative indexed files. A physical file on tape or diskette cannot be restored to an existing physical file on disk until the alternative indexed files for that file are deleted from disk.

Note: Because the system rebuilds alternative indexed files when they are restored, restoring a multiple-index file can be time-consuming. The time depends on the number of records in the file, the number of alternative indexed files defined for the file, and the key length of the file being restored.

The description of the RESTORE procedure in the *System/36 Environment Reference* book has more information about restoring multiple-index files.

Using Index Keys: The following rules apply to keys in the System/36 environment:

- A key can be up to 120 bytes in length, although in some cases the maximum length is lower. For example, DFU and RPG II restrict the maximum length of a key to 99 bytes.
- A key is treated as alphanumeric, even if the data is numeric. If a key is numeric, it should not contain a sign (+ or -).
- You can start a key anywhere within the record.
- You can combine as many as three fields to form a key.
- Key fields for an index file cannot overlap.
- Each field in a key must be defined to contain a unique set of positions in the record.
- You can specify key fields in any order, regardless of their place in the record.
- For a indexed physical file, the fields in a key must be next to each other in the record. For alternative indexed files, keys do *not* need to be next to each other in the record.

For example, in Figure 7-6, you could use fields 1 and 2 together as the primary key, fields 5, 9, and 7 together as an alternative indexed file key, and fields 1 and 3 as another alternative indexed file key.

Using Duplicate Keys: The system allows duplicate keys in indexed or alternative indexed files. For example, if you want to process employee records sequentially by department number, you can build an indexed file that uses the department number as the key. Because there would be more than one employee record for each department, the file must allow duplicate keys.

Specifying Duplicate Keys: You can specify whether a file allows duplicate keys when you create the file. If you use the BLDFILE procedure to create the file, you specify duplicate keys by using the DUPKEY parameter. If a program creates the file, you specify duplicate keys with the DUPKEY parameter of the FILE OCL statement. You can specify whether an alternative indexed file will contain duplicate keys using the DUPKEY parameter on the BLDINDEX procedure.

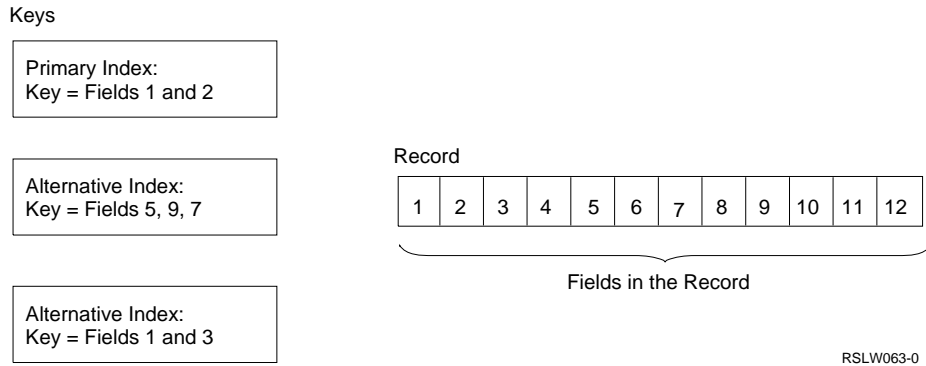


Figure 7-6. Key Fields

Checking for Duplicate Keys: If a file does not allow duplicate keys, the system checks for duplicate keys before an index entry is added to the file (either during add operations or during change operations that change a key field). If an operation would cause a duplicate key in an index that does not allow duplicate keys, the operation is not allowed.

Sequence of Duplicate Keys: If an indexed file has duplicate keys, the sequence of the duplicate keys is maintained by the relative record number of the records in the file. That is, the first record entered or added to the file is represented by the first entry in the index. Change operations do not change the position of records in the file. If a change operation changes a key, the new key is put in sequence by the relative record number of the original record.

Processing a File with Duplicate Keys: When you process a file with duplicate keys randomly by key, only one of the records having duplicate keys is available for processing. The available record is the record with an index entry first in the set of duplicate entries. Random processing by key is discussed in "Random Processing by Key" on page 7-17.

If you use the generalized processing method, you can retrieve the other records with duplicate keys by requesting operations that read the next record in the file. The generalized processing method is discussed in "Generalized Processing Method" on page 7-17.

You can process records with duplicate keys sequentially by key either for the entire file, or for records within limits. Sequential processing by key is discussed in "Sequential Processing by

Key" on page 7-15. When you specify limits for keys with duplicates, the lower limit is set to the first duplicate key in the index, and the upper limit is set to the last duplicate key. The upper limit is determined as records are read from the file. The upper limit includes records added by other jobs.

Using Alternative Indexed Files: If you change, delete, or add records to a multiple-index file, the system must change all the indexes. Therefore, the number of alternative indexed files directly affects the performance of the program.

It is more efficient to build alternative indexed files for a file only when needed. For example, if a program requires a certain index to produce a monthly report, that index could be built just before running the job that produces the monthly report.

Processing Files

Processing files involves the following concepts:

- Current record pointer
- Nonkeyed processing
- Keyed processing
- File processing methods

Current Record Pointer

When you open a file, the system establishes a current record pointer. The current record pointer points to a particular record position within the file. It positions a record for reading and maintains that position for changing or deleting the record.

The System/36 environment changes the current record pointer when it successfully completes an

input operation or when an end-of-file completion code is returned.

The current record pointer is always at one of the following positions:

- Beginning of file. In this position, the pointer is before the first record in the file. If a request to read the next record is made, the system reads the first record in the file. After the system opens a file, the pointer is set at the beginning of the file.
- End of file. In this position, the pointer is beyond the last record in the file. If the system receives a request to read the previous record, it reads the last record in the file.
- Record position. In this position, the pointer points to a record position in the file. The record at that position may be an active record or a deleted record.

Nonkeyed and Keyed Processing

You can process files without using a key (non-keyed processing) or according to the value of the key (keyed processing).

Nonkeyed Processing: In nonkeyed processing, the records are processed in the sequence in which they are stored in the file. The system uses this sequence to process records randomly or consecutively. The system bases the current record position on the relative position of the record in the file. When the system opens the file, it sets the current record pointer at the beginning of the file. The current record position changes as read operations occur for the file. The system performs change, delete, and release operations on the record at the current record position. Adding a record does not change the position of the current record pointer.

Nonkeyed processing allows the following operations for sequential, direct, and indexed files:

- Read the first record in the file
- Read the last record in the file
- Read the next record in the file
- Read the previous record in the file
- Read the record at the current record position + N
- Read the record at the current record position - N
- Read the record at the relative record number

- Add a record at the end of data
- Add a record at the relative record number
- Change the current record
- Delete the current record
- Release the current record

Note: Not all high-level languages allow every operation in the preceding list. Refer to the appropriate language book for information about the operations allowed in a particular language.

Keyed Processing: In keyed processing, the records are in sequence by their key values. This sequence allows the system to process records randomly or sequentially by key. During file processing, the system maintains a current record pointer. When the system opens the file, it sets the pointer at the beginning of the first key in the index. The current record position changes as read operations occur for the file. The system changes, deletes, and releases the record whose index entry is at the current record pointer. Add operations do not change the position of the current record pointer.

Keyed processing allows the following operations for indexed files:

- Read the record that has a specific key value
- Read the first record in the file
- Read the last record in the file
- Read the next record in the file
- Read the previous record in the file
- Read the record that has an equal or greater key value
- Read the record that has a greater key value
- Read the record if it has a key equal to a specified value
- Add a record at the end of data
- Change the current record
- Delete the current record
- Release the current record

Note: Not all high-level languages allow every operation in the preceding list. Refer to the appropriate language book for information about the operations allowed by a particular language.

File Processing Methods

Before you create the file, you must select a processing method. **Processing method** is the term used to describe the way a program retrieves disk records for processing. The processing method defines a set of functions that the high-level lan-

guage program uses to retrieve records from the file. The processing methods are:

- Consecutive
- Sequential by key
- Random by relative record number
- Random by key
- Generalized processing

Consecutive processing allows you to process records in the order in which they appear in the file. Sequential processing by key allows you to process indexed files by the sequence of the keys in the index. Random processing by relative record number allows you to process a record by specifying the record's relative record number. Random processing by key allows you to process, in an indexed file, a record by specifying the record key. Generalized processing allows you to process a file using a combination of the above methods.

Do not confuse these processing methods with file organizations. However, the file organization plays a significant role in determining which processing method you can use in a program. The following table indicates which processing method you can use for each file organization.

Processing Method	Organization		
	Sequential	Direct	Indexed
Consecutive	Yes	Yes	Yes
Sequential by key	No	No	Yes
Random by relative record number	Yes	Yes	Yes
Random by key	No	No	Yes
Generalized processing method (nonkeyed processing)	Yes	Yes	Yes
Generalized processing method (keyed processing)	No	No	Yes

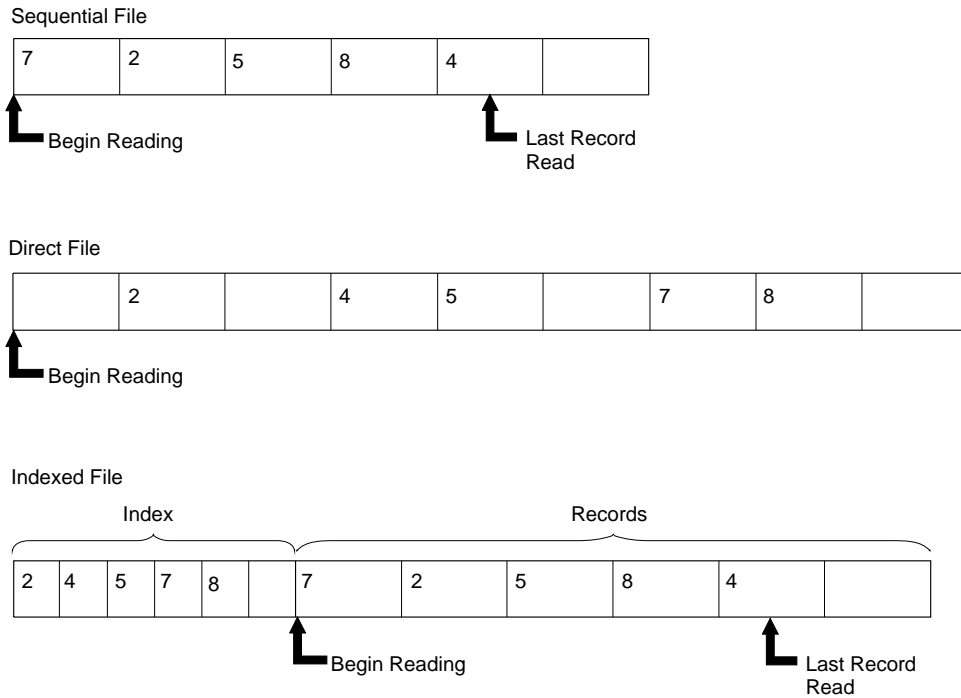
Consecutive Processing Method: The consecutive processing method reads records in the order in which they appear in the file, one after another from first to last, as shown in Figure 7-7.

You can use the consecutive processing method for all three file organizations.

Sequential Files: When the system processes a sequential file consecutively, the space at the end of the file reserved for new records is *not* read. Therefore, record 4 is the last record read in the sequential file in Figure 7-7.

Direct Files: When the system processes a direct file consecutively, if it is not delete-capable, the program reads the gaps that were left for new records. Therefore, the program must test for a blank record each time it reads a record. When a program processes a direct file that is delete-capable, the deleted records are bypassed. A **direct file** in the System/36 environment is a disk file in which records are referenced by the relative record number. A direct file is created as an OS/400 physical file.

Indexed Files: When the system processes an indexed file consecutively, the program ignores the index portion of the file as it reads the records. However, if change operations change keys, or if records are added or deleted, the system automatically changes all indexes for the file. The space at the end of the file that is reserved for new records is *not* read. Therefore, record 4 is the last record read in the indexed file in Figure 7-7.



RSLW019-0

Figure 7-7. Consecutive Processing Method

Sequential Processing by Key: When the system processes an indexed file sequentially by key, the program processes the records according to the sequence of the keys in the index, as shown in Figure 7-8.

Although the system processes the keys consecutively, it processes the records randomly because the index entries are sorted but the records are not. The system processes records in ascending order of key value.

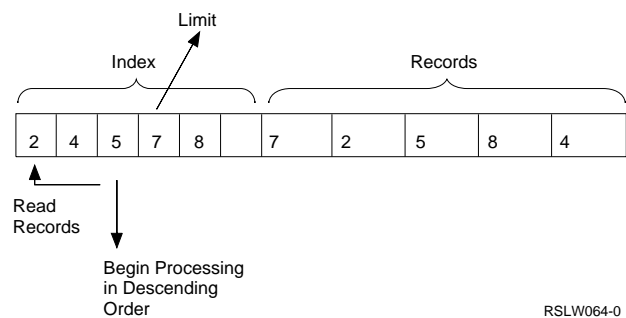
If there are duplicate keys in the index, the system processes records with duplicate keys in the order of the relative record number. For more information about duplicate keys, see "Using Duplicate Keys" on page 7-11.

The system bypasses deleted records in a delete-capable file.

Sequential Processing by Key within Limits: The sequential-by-key processing method usually processes all records in a file. However, you can

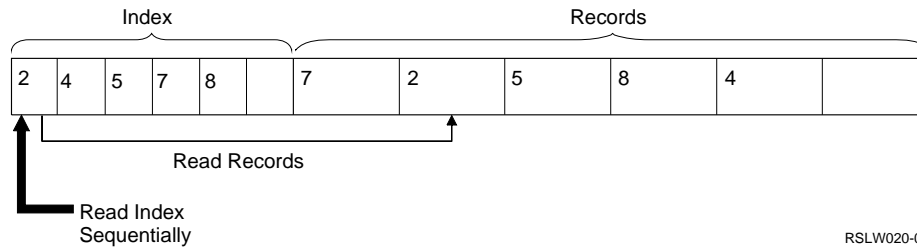
specify the upper and lower limits of the key values of the records you process sequentially by key in ascending order, as shown in Figure 7-9.

You can also specify the limit of the key values for records you process sequentially by key in descending order, as shown in the following figure.



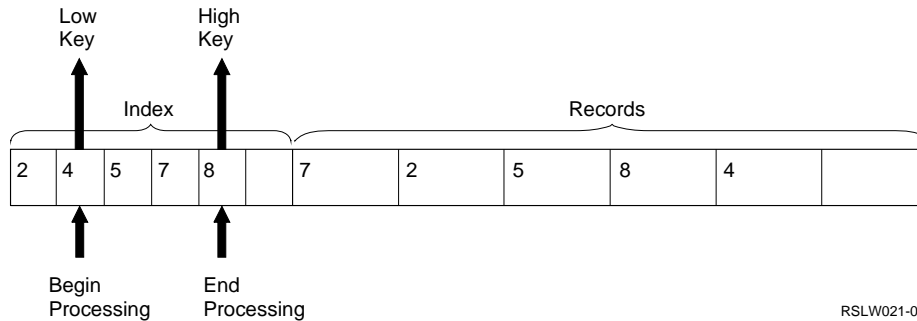
RSLW064-0

You can process an indexed file sequentially by key within limits to process a specific group of records in the file. You can process the file in ascending order or descending order.



RSLW020-0

Figure 7-8. Sequential Processing by Key



RSLW021-0

Figure 7-9. Sequential Processing by Key in Ascending Order within Limits

When you process files in ascending order:

- The lower limit is the key value at which processing begins, and the upper limit is the key value at which processing ends.
- If duplicate keys are in the index, the system uses the first duplicate of the appropriate key as the lower limit, and the last duplicate of the appropriate key as the higher limit.

When you process files in descending order:

- Processing begins at the key value just below the limit and continues in descending order to the first record in the file.
- If duplicate keys are in the index, the first key the system returns is the last key in the group of duplicates.

The limits you specify for processing the file remain in effect until:

- New limits are set
- A random read operation occurs
- The file is closed

Random Processing by Relative Record Number:

The system uses the random processing method by relative record number to read only the record the program needs. It ignores all the other records in the file. Therefore, you determine the order in which disk records are processed. The relative record numbers indicate the positions of the records in the file relative to the beginning of the file. Relative record numbers are positive whole numbers that the system converts into the disk addresses of the records.

The relative record number processes all three file organizations randomly. When a delete-capable file is processed randomly by relative record number, the program cannot read the deleted records. However, the deleted records do take up space in the file, so they affect the relative record number of other records.

Sequential Files: You can processing sequential files randomly by relative record number when you process only a few of the records in the file and you know the relative record numbers of the records.

Direct Files: Figure 7-10 shows an example of a direct file the system processed randomly by relative record number.

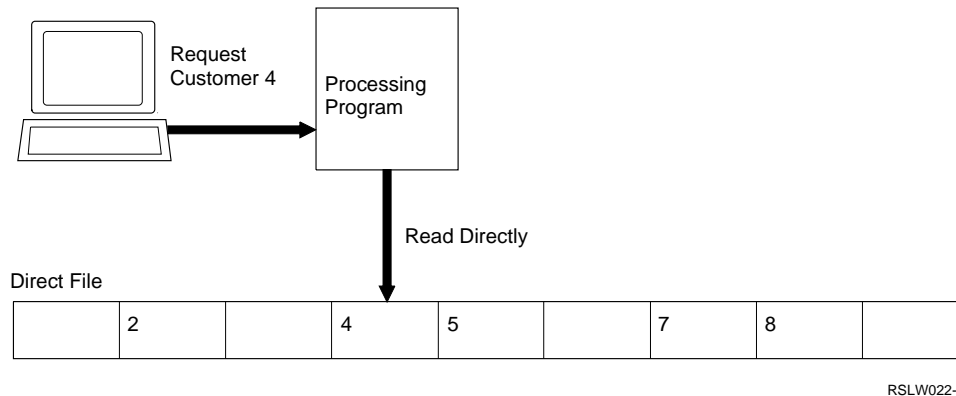


Figure 7-10. Random Processing by Relative Record Number for a Direct File

Blank records are not a problem when retrieving records from a direct file, because the program reads only the desired records. In the example in Figure 7-10, the direct file contains customer records stored at record positions based on the customer number. The program receives a request to read the record for customer 4. The system uses the customer number as the relative record number. The program retrieves the requested record (the fourth record relative to the beginning of the file) without reading any other record. The program must determine whether there is information in the record.

Indexed Files: When the system processes an indexed file randomly by relative record number, it processes the records by their relative record number values. If change operations change any of the keys in the records, or adds or deletes any records from the file, the system changes all indexes for the file.

Random Processing by Key: The system processes only indexed files randomly by key. Figure 7-11 shows an example of random processing by key for an indexed file. In the example in Figure 7-11, the program receives a request to read the record for customer 4. The system uses the customer number as the key. It retrieves the record for customer 4 in two steps:

1. The system searches the index for a value that matches the requested key. The index entry with the matching key value also contains the relative record number of the record for that key.
2. The system reads the record at that record position from the data portion of the file.

Generalized Processing Method: Use the generalized processing method to process a disk file randomly, consecutively, or sequentially while changing, deleting, or adding records. When using the generalized processing method, you can do nonkeyed or keyed processing.

For example, assume you have a file of employee records containing employee number, department number, and location code, and the file is an indexed file with location code as the key. Using the generalized processing method with keyed processing, your program can process the records for all the employees at a location.

As Figure 7-12 on page 7-18 shows, the program can process the file randomly by key to find the first record that has the location code.

As Figure 7-13 on page 7-18 shows, after the program finds that record, it can process the file sequentially by key to process the records for the other employees at that location.

As Figure 7-14 on page 7-19 shows, while processing the file, the program can add records to the file.

Sequential or Direct Files: Use the nonkeyed generalized processing method to process sequential or direct files consecutively or randomly by relative record number.

Indexed Files: Use the nonkeyed generalized processing method to process indexed files by using:

- Consecutive processing
- Random processing by relative record number

Use the keyed generalized processing method to process indexed files using:

- Sequential processing by key
- Random processing by key

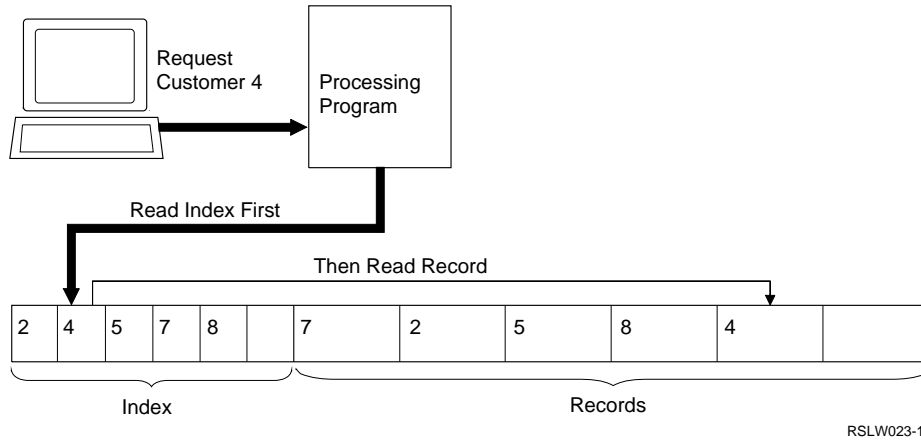


Figure 7-11. Random Processing by Key for an Indexed File

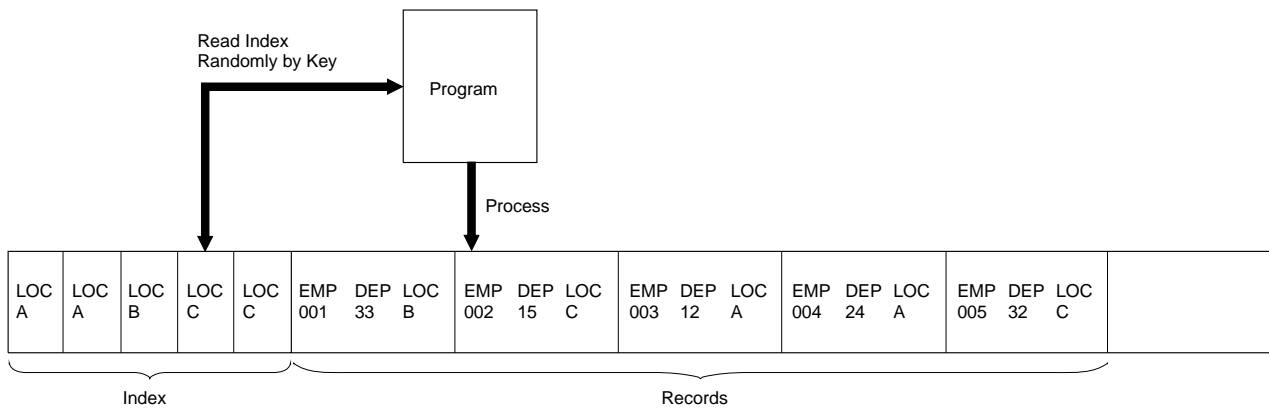


Figure 7-12. Generalized Processing Method Randomly by Key

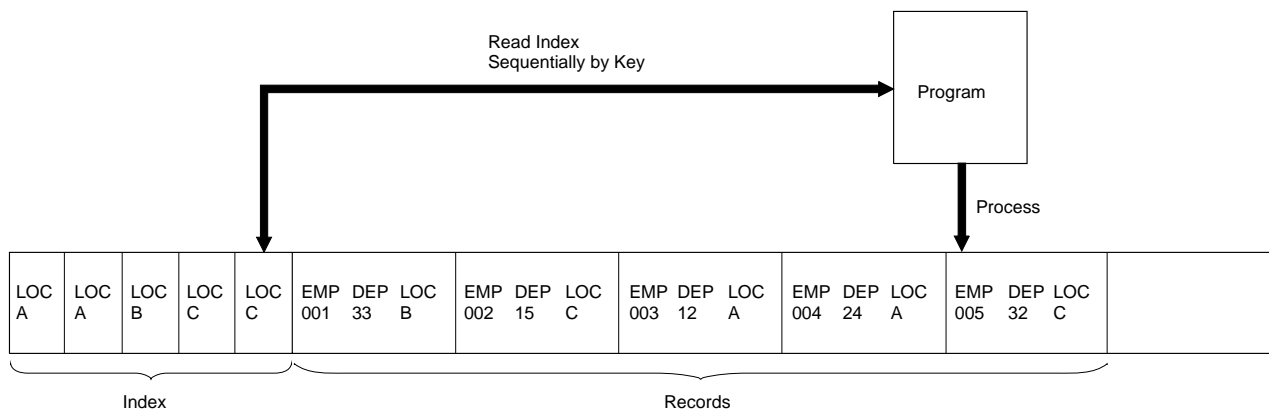


Figure 7-13. Generalized Processing Method Sequentially by Key

Relative Record Number	Contents
1	Next available relative record number First relative record number Last relative record number
2	Next available relative record number First relative record number Last relative record number
.	.
.	.
.	.
10	W1 transaction 1
11	W1 transaction 2
12	W1 transaction 3
.	.
.	.
.	.
110	W2 transaction 1
111	W2 transaction 2
.	.
.	.
.	.

Note: The records entered from each display station require a control record. This direct file reduces the possibility that a display station operator tries to process a record within a sector assigned to another program. However, the number of records that can be entered from a display station is limited, and gaps can exist between the end of one section of the file and the beginning of the next section.

For more information about transaction files, see the *DB2 for OS/400 Database Programming* book.

Processing Method: You determine the file organization when you create the file. If you create a file for an existing application in which you use a particular processing method, your choice of file organization may depend on that processing method.

If you use random processing, the program can process specific records in a file (by relative record number or by key) without processing the entire file. For example, when display station operators process telephone orders, they want to process specific data in an indexed file using the customer number as a key.

If you use consecutive processing, the program processes all the records in the file. For example, to produce invoices for all customers, the program would process the file consecutively. In that case, the file organization could be sequential.

Application: The type of application affects your choice of file organization. For processing all the records in a transaction file arranged in customer-number sequence (and used as input for a report), use sequential file organization. For processing a master file of 10,000 records that has few additions or deletions (and that is used for high-speed inquiry), use direct organization. For processing a master file of employee addresses to print addresses on 15% of the payroll checks, use indexed organization.

Batch Processing and Interactive Processing: It is important to identify whether the application uses batch processing or interactive processing.

In batch processing, the system accumulates and processes groups of data at specific times, such as daily, weekly, or monthly. Applications processed once a week (for example, payroll) are perfectly suited for batch processing.

In interactive processing, the system processes individual records or transactions at the time the transaction occurs. For example, in an interactive order entry application, as soon as you make a sale, the system subtracts the quantity of merchandise sold from the quantity on hand in the appropriate master file.

The type of processing can require a particular file organization. Batch processing, which does not require random processing to the data, might require sequential files. Interactive processing requires immediate access to the file, so direct or indexed files are more appropriate.

Activity of the File

Activity means how often you process the file. You measure activity as a fraction produced by the number of transactions to the file divided by the number of records in the file. This fraction is usually expressed as a percentage.

You can randomly process a relatively inactive file and have a direct or an indexed organization. As activity increases, consecutive processing is better

because it becomes more likely that the record to be processed is available in a buffer and that it can be processed without reading from, or writing to, the disk. Therefore, very active files could be sequential (processed consecutively) or indexed (processed sequentially by key).

You can reduce the total activity of an indexed master file by sorting a transaction file (so that only one retrieval of a master file record is needed for a group of transactions with the same key). Also, you can reduce activity by sorting the data in the master file to match the sequence of the index.

Disk Space

A sequential or direct file takes less space than an indexed file because an indexed file requires additional space for the index.

File Attributes

File attributes include the following:

- Scratch files
- Job files
- Resident files
- Extendable files
- Delete-capable files

File attributes indicate how long a file will be retained, and whether you can extend or delete records in a file.

The file retention attributes are scratch, resident, and job. Resident files are permanent files. Scratch files store temporary data for one program. Job files store temporary data needed only from one job step to the next. The system can automatically extend an extendable file to prevent a program from ending abnormally when the file is not large enough to add more records. Delete-capable files allow your programs to use their delete statements or codes to delete records from a file.

Scratch Files

Scratch files are files that have RETAIN-S specified on the FILE OCL statement. Programs cannot share scratch files. You normally use scratch files as temporary work files for a single job step. At the end of the job step in which the scratch files are created, the scratch files release the disk space they used. Thus, you can use a scratch file in only one job step.

Job Files

Job files have RETAIN-J specified on the FILE OCL statement. Job files exist from the time you create them until the job ends, or until you delete them. The system releases the disk space used by the job file when the last job step ends.

The system releases the disk space used by a job file when you specify the RETAIN-S parameter of the FILE OCL statement in a job step for the file.

Job files usually contain a limited number of records from a particular file, and these records are used by various programs within the same job. For example, you can place portions of a master file into a job file, and many programs can use this job file within the same job. Programs in different jobs cannot share a job file.

Resident Files

Resident files have RETAIN-T specified on the FILE OCL statement. Resident files are permanent files. For example, a master file is a resident file. With resident files you can share data among various jobs. You can save and restore resident files using diskette or tape. Resident files remain on the disk until you do one of the following operations:

- Run the DELETE procedure.
- Change the file retention parameter on the FILE OCL statement to S (scratch) in a particular job step and allocate the file by the job.

Note: The file cannot be shared with another job while the file is being deleted or scratched.

Using Resident Files from One Job

Step to Another: If you specify JOB-YES on the FILE OCL statement for a resident file, the file is kept allocated to this job for other job steps until the end of the job. The other parameters on the FILE statement remain in effect until the end of the job, or until a FILE statement in another job step that has the same NAME parameter overrides them. You cannot override the location and size (RECORDS or BLOCKS) parameters after the file is created. You can add new parameters by a FILE statement in another job step that has the same NAME parameter.

If you specify JOB-YES on a FILE statement in another job step later in the job, the parameters you specified on that FILE statement remain in effect until the end of the job, or until you override them in a later job step. The parameters you specify with JOB-YES on the FILE statement in a previous job step no longer apply.

You can specify the JOB-YES parameter only for FILE statements that are outside the LOAD and RUN statements. If you place a FILE statement outside a LOAD and RUN pair, the system tries immediately to acquire ownership of the specified disk file for use by the job. If the FILE statement is within the LOAD and RUN pair, the system waits until it encounters the RUN statement before it acquires ownership of the file.

If you specify JOB-NO on a FILE statement in a job step, the program gives up ownership of the file at the end of the job step.

The following example shows how the JOB-YES parameter affects the FILE OCL statement for a new file during each of three job steps:

```
* Job step 1:
// FILE NAME-A,JOB-YES,RECORDS-10,EXTEND-50,DBLOCK-200,
//     DISP-NEW
// LOAD PROG1
// RUN
```

In job step 1, the JOB-YES parameter means the FILE OCL statement parameters specified for file A remain in effect until the end of the job, or until you override them in a later job step. Following are the parameters specified for file A:

- The size of the file is 10 records.
- The file is extended by 50 records whenever additional space is needed.

- 200 records are moved between main storage and disk for each input/output operation.

If the program does not use file A during job step 1, it does not create file A.

If job step 2 includes the FILE statement for file A:

```
* Job step 2:
// LOAD PROG2
// FILE NAME-A,BLOCKS-20,EXTEND-60,DISP-NEW
// RUN
```

The program uses the BLOCKS parameter in job step 2 instead of the RECORDS parameter in job step 1. Therefore, if the program creates file A in job step 2, its size is 20 blocks instead of 10 records.

Also, the EXTEND parameter in job step 2 overrides the EXTEND parameter in job step 1, but only for a single job step (because JOB-YES is not specified on the FILE statement for job step 2). Therefore, if the program creates file A in job step 2, the file is extended by 60 blocks instead of 50 records whenever it requires additional space.

If the program does not create file A in job step 2, it resets the EXTEND parameter to 50 at the end of the job step. The DBLOCK parameter specified in job step 1 remains in effect in job step 2 because the program does not override it.

RECORDS and BLOCKS parameters are the only exceptions to the rule that parameters are reset to the value specified on the FILE statement with JOB-YES specified. If job step 3 contains no FILE statement, and the program did not create file A in job step 2, the BLOCKS parameter stays at 20.

Note: If you specify the RECORDS, BLOCKS, or LOCATION parameters with JOB-YES on a FILE statement for a file that already exists, the system ignores the parameters.

In job step 2, the program resets the EXTEND parameter to 50 because that is the value specified on the FILE statement with JOB-YES in job step 1. The DBLOCK parameter specified in job step 1 remains in effect into job step 3.

Resident files with the JOB-YES parameter can cause a file lock when the file is shared. If one program within a job acquires the file as a shared file, another program in another job cannot acquire

the same file as a nonshared file until one of the following occurs:

- The program that acquired the file as a shared file goes to the end of the job.
- You specify JOB-NO for a particular job step in the job that was sharing the file and that job step ends.

Figure 7-15 is an example of OCL statements used with two jobs sharing a file for which you specified JOB-YES. In this example, program C wants exclusive use of file A. Therefore, program C must wait until program B in job Y ends.

Also, a file deadlock can occur if two or more jobs are using two or more resident files with JOB-YES specified. If the jobs try to use files that do not permit sharing, and that have the JOB-YES parameter specified, both jobs can have to wait.

If you are running a job that contains a MRT procedure, and you want the file to be used by the other job steps, the MRT procedure must contain the FILE OCL statement on which you specified the JOB-YES parameter.

The *System/36 Environment Reference* book has more information about using the FILE OCL statement.

Extendable Files

An extendable file is a disk file for which the system automatically attempts to allocate more space each time the file becomes full. Specifying an extendable file prevents your program from ending abnormally when there is no room in the file for additional records.

Specifying an Extendable File: You can specify a file as extendable using either of the following methods:

- The EXTEND parameter of the FILE OCL statement specifies the number of blocks or records to extend the file.
- BLDFILE procedure. The number of blocks or records to extend the file is a parameter.

The extension value must be a numeric value that indicates the amount of additional space needed

for the extension. If you specified the file size in blocks when the file was created, the extension value is in blocks. This value must be large enough to contain at least one record.

If you specified the file size in records when the file was created, the extension value is in records. The amount of the file extension is the number of records or blocks specified, rounded up to a block boundary.

If you specify an extension value when the file is created, or when a file is rewritten with new information (DISP-OLD), the extension value becomes an attribute of the file. The file is extended, if required, by any program using the file.

Notes:

1. When you rewrite a file with new information, the old extension value is not saved for the file. You must specify the value again, as if the file were a new file.
2. If you use EXTEND on a FILE OCL statement that is *not* for a new file, the EXTEND value is ignored and the system uses the value provided when the file was created. You can change the EXTEND value for an existing file using the Change Physical File (CHGPF) command.
3. If no EXTEND value is specified when creating new files with the FILE OCL statement or the BLDFILE procedure, a default extend value of 32 767 divided by the record length is used.
4. When accessing files by relative record number, no extending of files is done regardless of the EXTEND value of the field.

The system automatically extends a file when records are added using consecutive, sequential-by-key, or random-by-key access.

You can extend a file any number of times, up to 8 000 000 records. A file is not extended if there is not enough disk space, or if a disk input/output error occurs. You cannot cancel a file extension.

If you have used the maximum number of extensions, the system sends a message to the operator asking whether the file should be extended again.

```

Job X
* Job step 1 for job X
// FILE NAME-A,JOB-YES,DISP-SHR // FILE NAME-A,JOB-YES,DISP-SHR
// LOAD PROGA // LOAD PROGB
// RUN // RUN

* Job step 2 for job X
// LOAD PROGC
// FILE NAME-A,DISP-OLD
// RUN

```

Job Y

Job Y

```

// FILE NAME-A,JOB-YES,DISP-SHR
// LOAD PROGB
// RUN

```

Figure 7-15. OCL Statements Used with Two Jobs Sharing a File

Delete-Capable Files

Programs can delete records in a delete-capable file. Specify a file as delete-capable to allow your programs to delete unwanted records when processing the file. If you need the data that was in a deleted record, do not use a delete-capable file. Instead, have your program place a delete code in the record. Then, when the file is processed, your program can check for this code.

Creating a Delete-Capable File: To create a delete-capable file, do either of the following:

- Specify the DFILE parameter on the BLDFILE procedure.
- Specify DFILE-YES on the FILE OCL statement.

When you create a delete-capable direct file, the system initializes it to contain all deleted records. The system initializes non-delete-capable direct files with blank records.

Deleting Records from a Delete-Capable File: When you delete records from a delete-capable sequential or indexed file, the records are not physically removed from the file (unless you use the COPYDATA procedure to remove them). They are marked as deleted records. Therefore, the data that was in the

record before the system deleted it is no longer available to the program.

When the system deletes a record from a multiple-index file, it deletes the key for that record from all indexes.

In RPG II, if you use an address output (addrout) file to process records in a file, and you delete a record, the record is deleted from the file you process but not from the addrout file. To delete the record from the addrout file, you must delete the entry for the record in the addrout file or re-create the addrout file.

The following table lists the statements various programming languages use to delete records from a delete-capable file.

Programming Language	Statements Used to Delete Records
COBOL	DELETE statement
RPG II	U in column 15 of file description specifications DEL in columns 16–18 of output specifications

Processing a File Containing Deleted

Records: When the system processes a file containing deleted records consecutively or sequentially by key, it bypasses each deleted record and reads the next record in the file.

When the system processes a file containing deleted records randomly by key or randomly by relative record number, it returns a record-not-found completion code to the program when it processes a deleted record.

Adding Records to a Delete-Capable

File: You can add records to delete-capable sequential, direct, and indexed files using relative record numbers.

Using RPG II to Add Records to Delete-Capable Files:

For sequential and direct files, you must first place the relative record number of the record to be added to the file in the RECNO field. The system defines the RECNO field on the continuation line of the file description specifications. The relative record number must be the record number of a deleted record. Then, to add a record to the file, code output specifications that contain ADD in columns 16 through 18. RPG II uses the relative record number from the RECNO field to locate where the record is to be added to the file. If the relative record number is not the number of a deleted record, a stop occurs and the system displays a message that a duplicate record exists in the file.

You add records to indexed files randomly by key using chaining. **Chaining** compares the key field of the record to be added with the key fields already in the index. This ensures that the record to be added is not a duplicate of a record already in the file. With chaining you can design your program to handle any duplicate key fields it finds, without requiring the operator to respond to an error message.

The *System/36-Compatible RPG II User's Guide and Reference* book has more information about using chaining to add a record to an indexed file.

Using COBOL to Add Records to Delete-Capable Files:

If you specify relative organization for the file, you can add records to delete-capable files. When you specify ACCESS IS RANDOM or ACCESS IS DYNAMIC, new

records are inserted into the file. The RELATIVE KEY you specified for the file must contain the desired relative record number for this record before the system can perform a WRITE operation. When the system performs a WRITE operation, the record is placed at the specified relative-record-number position in the file. If the relative record number is not the number of a deleted record, the system stops the program and returns a file-status indicating that a duplicate record exists.

Using DFU with Delete-Capable Files:

DFU changes, lists, or inquires into delete-capable files. You cannot use DFU to create delete-capable files, or to delete records from a delete-capable file.

Blocking Records

This section describes blocking of records and the differences between physical and logical input/output operations.

A record block is the number of records transferred as a unit of information between a disk file and a buffer in main storage. Although only one record at a time is available for processing by your program, one or more records can be transferred into the data buffer at a time. The block length specifies the amount of main storage used for a data buffer in your program.

You can change the buffer size using the DBLOCK parameter of the FILE OCL statement. You can specify a DBLOCK value up to 65 535, but any value over 32 767 results in a blocking value of 32 767. The block length does not affect the way records are stored on the disk, but does affect when they are stored.

The System/36 environment allows record blocking to be used only when the following conditions are true:

- The file is opened for input or output.
- Sequential processing is used.
- The file is not being shared with other jobs (DISP-SHR on the FILE OCL statement is not specified).

To use record blocking when sharing files, do one of the following:

- Use the CHGS36 command or the CHGS36A command to change the System/36 environment value to always use record blocking when sharing files. This affects the default for all System/36 environment jobs on the system. See “Commands for Configuring the System/36 Environment” on page 3-2 for more information.
- Use the System/36 environment-supplied program QEXRCDBK to change your job to always use record blocking when sharing files. See the appendix on IBM-supplied programs in the *System/36 Environment Reference* book for more information about QEXRCDBK.
- To specify record blocking for an individual file, specify SEQONLY(*YES) on an OVRDBF command. See the *CL Reference* book for a description of the OVRDBF command.

Note: If you use record blocking when sharing files, functional differences from System/36 could occur. When blocking on output, records are not immediately written to disk and do not have immediate access to another job. When blocking on input, updates from another job may be missed. Make sure your programs are not dependent on the specific time the records are written to disk.

Considerations for Efficient Record

Blocking: Block length is a multiple of record length. For example, if the record length is 64 characters, and the blocking factor is four, the block length is 256 characters. In that case, four records are transferred at one time.

Blocking is useful if you are likely to process multiple records in a data buffer. By specifying a large blocking factor, you reduce the number of times the system must read from and write to the disk. For example, if your program reads a file consecutively when records are blocked 100 per data buffer, to read the first record, the system must transfer 100 records from disk to the data buffer. This takes a relatively long time. However, the next 99 times the program reads a record, the record is already in main storage, so no time is required to read the disk file.

You can reduce the number of disk reads and writes required by your program by increasing the size of your data buffers. However, increasing the size of your data buffers increases the amount of

main storage required to run your programs. This affects your program and system performance.

If you do not specify a DBLOCK value in the FILE OCL statement, the system uses default blocking. Default blocking uses a buffer of 4096 bytes.

Sharing Files

File sharing occurs when two or more programs process the same file at the same time.

Figure 7-16 on page 7-27 shows an example of file sharing.

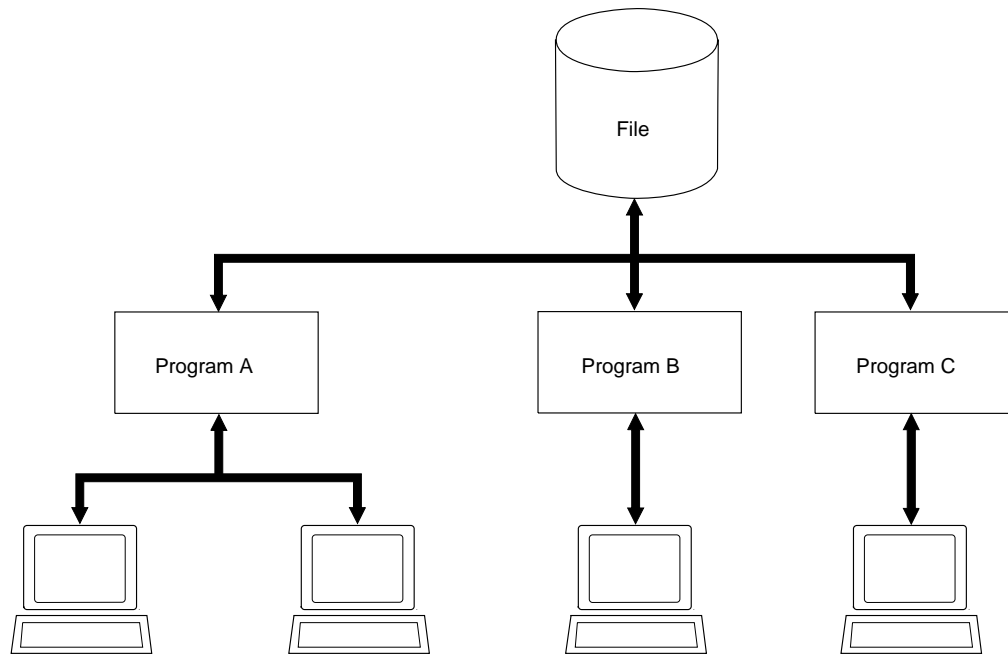
File Sharing Considerations

If you want more than one program to share a file, consider the following guidelines:

- Only resident files can be shared.
- Files you are creating cannot be shared.
- If you change a resident file to a scratch file by specifying a RETAIN-S parameter on the FILE OCL statement, you cannot share the file.
- The system protects records read for change by one program from being changed by another program using the same file. For more information, see “Record Protection” on page 7-28.
- If programs share more than one file, all programs should process the files in the same sequence to reduce the chances of a file deadlock occurring. For more information, see “File Deadlock Conditions” on page 7-29.
- If you share a file, record blocking is not used. See “Blocking Records” on page 7-25 for more information.

Levels of File Sharing

The system allows several levels of file sharing. The Disposition (DISP) parameter of the FILE OCL statement determines the level of file sharing. The System/36 environment maps the DISP parameter share levels into AS/400 lock states. The System/36 environment does not check to see if the application is using a file with a valid share level.



RSLW027-1

Figure 7-16. File Sharing Diagram

When one or more programs are using a file, the programs that own the file determine which other programs can share the file. The share level tells the system how the program uses the file and what types of processing other programs can do while sharing the file. For example, if you specify DISP-SHRMR, the program changes the file while sharing it with other programs that can only read the file. Once the system lets the program use the file, it does not allow other programs that want to change the file to use the file.

The following table shows:

- The levels of file sharing you can specify
- The type of processing you can do when you own the file
- The type of processing other programs can do while your program owns the file

Share Level	The Program That Owns the File Can:	Other Programs Can:
SHR	Read and change	Read and change
SHRMM	Read and change	Read and change
SHRMR	Read and change	Read only
SHRRM	Read only	Read and change

Share Level	The Program That Owns the File Can:	Other Programs Can:
SHRRR	Read only	Read only
OLD	Read and change	Not allowed to process file
NEW	Read and change	Not allowed to process file
Not specified	Read and change	Not allowed to process file

Note: Changing includes change, delete, and add operations to a file.

Waiting for Files to Become Available

If another program uses a file, and the file is at a share level that does not permit your program to use the file, the system waits for the file.

If the file is not available, the system automatically waits for the file until the program using the file goes to the end of the job step. If the waiting program is requesting no share, it must wait until the job that owns the file reaches a job step in which the file is not used. While the program is waiting, all other programs that request the file also have to wait. When the file becomes avail-

able, the program gains ownership of the file and begins running. If other programs are also waiting to use the file, the system checks whether they can use the file at their requesting share level.

Figure 7-17 shows whether another program can share a file at a requested level when one program owns the file.

For example, if your program requests to share a file by using DISP-SHRMM on the FILE OCL statement, the system would allow your program to share the file only if the programs that own the file specified either DISP-SHRRM, DISP-SHRMM, or DISP-SHR on the FILE OCL statement.

When several programs are sharing a file, other programs can share the file only when their requesting share levels are compatible with all other share levels.

Using the WAIT Parameter: Use the WAIT parameter on a FILE OCL statement that is not between LOAD and RUN statements to determine (within your jobs) whether a file is available. You can specify WAIT-YES or WAIT-NO.

If you specify WAIT-NO, the system tries to acquire the file for the program at the desired shared level. If the file is unavailable to the program, the system returns completion code 2031 to the procedure. By using the ?CD? substitution expression, and an IF conditional expression within your procedure, you can choose the processing steps done within a job if the file is unavailable. For example, if a file you need as input to the first program of a job is unavailable, you can decide not to run the remaining parts of the job.

If you specify WAIT-YES or no WAIT parameter, the program waits until the file becomes available. This wait condition lasts until the program can use the file. For example, you may decide to submit a program using a particular file and not have this program use the file until all other programs using the file are finished.

Note: The System/36 environment does not issue an error message if it is waiting for a NEP or MRT.

The following example uses the WAIT parameter to determine whether a file is unavailable (busy) for more than 30 minutes. If the file is unavailable for more than 30 minutes, the system sends the system operator a message to run the job later.

```
* Parameters used:
* Parameter 1 = Number of minutes between tries.
*             Range of 1 to 59 minutes.
*             Default 5 minutes.
* Parameter 2 = Maximum number of tries.
*             Default is 6 tries.
* Parameter 64 = Number of times the program
*               tried to get the file
*****
// IFF ?1?=' ' EVALUATE P1=?1?00'
// EVALUATE ?64F'1'? P1,6=?1'000500'?
// TAG LOOP
// FILE NAME-TEST,WAIT-NO
// IF ?CD?=0000 GOTO GOTFILE
// IF ?64?>?2'6'? GOTO NOFILE
// EVALUATE P64=?64?+1
// WAIT INTERVAL-?1?
// GOTO LOOP
*
// TAG NOFILE
// ** 'FILE "TEST" IS BEING USED; RUN JOB LATER'
// RETURN
*
// TAG GOTFILE
// LOAD PROG1
// RUN
```

Note: If you place the FILE statement before the LOAD statement, the system attempts to acquire the file for the job immediately. For example:

```
// FILE NAME-INPUT,UNIT-F1,LABEL-MASTER,RETAIN-T,
//     DISP-OLD,WAIT-YES
// ATTR CANCEL-NO,MRTMAX-20,NEP-NO,PRIORITY-HIGH,
//     RELEASE-YES
// LOAD ORDPRG,ORDERLIB
// PRINTER NAME-REPORT,ALIGN-YES,SPOOL-YES
// RUN
```

Record Protection

Record protection prevents two or more programs from changing a record in a shared file at the same time. Record protection applies to programs that process the file with SHR or SHRMM share levels. A program that tries to change a record already being changed by another program is forced to wait until the first program releases the record.

Figure 7-17. File Share Level Ownership

Share Level Requested by another Program	Share Level for the Program That Owns the File					
	Program	SHRRM	SHRMM ¹	SHRMR	SHRRR	No Share ²
SHRRM	Yes	Yes	Yes	Yes	Yes	No
SHRMM ¹	Yes	Yes	No	No	No	No
SHRMR	Yes	No	No	No	No	No
SHRRR	Yes	No	No	No	Yes	No
No Share ²	No	No	No	No	No	No

¹ DISP-SHRMM is the same as DISP-SHR.

² No share is specified by the DISP-OLD, DISP-NEW, or no DISP parameter being specified.

Releasing Locked Records

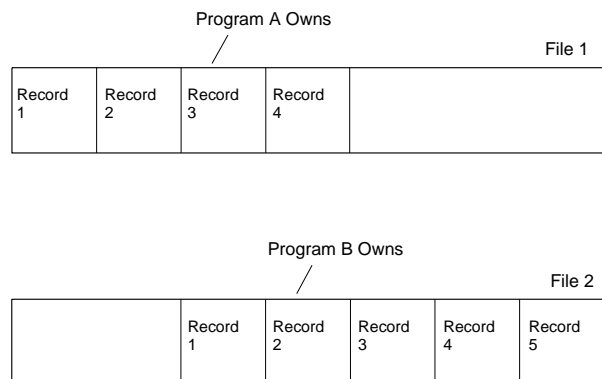
The system releases a record when any of the following conditions occur:

- The program reads another record from the file.
- The program does a data management operation that causes an error to occur.
- The program changes the record, and file sharing is used (for example, DISP-SHR on the FILE OCL statement).
- The program deletes the record from the file.
- The program does a release operation. How your program releases records depends on the high-level language you use. For example, in RPG II you can release a locked record by writing a record with no output fields.
- The program closes the file.
- The program ends.

File Deadlock Conditions

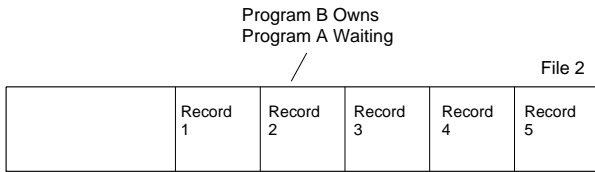
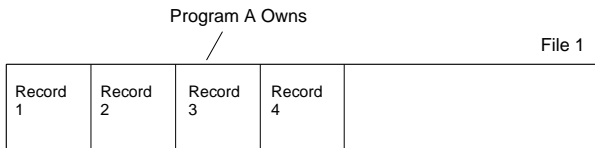
The condition of programs waiting for each other is called a FILE DEADLOCK. A file deadlock condition can occur when programs share two or more change files.

For example, program A and program B are changing shared files 1 and 2. As the following figure shows, program A reads record 3 for changing from file 1, and program B reads record 2 for changing from file 2.



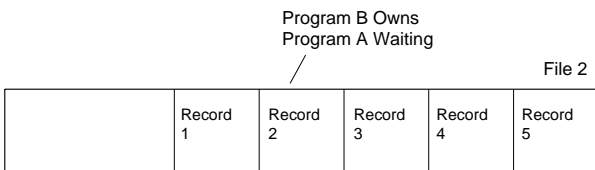
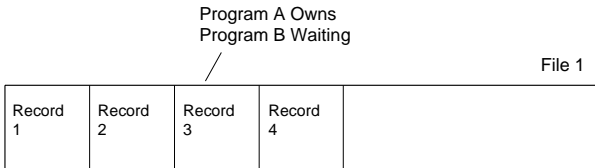
RSLW032-0

The following figure shows a situation where program A tries to read record 2 from file 2. Program A must wait because program B is using the record.



RSLW033-0

As the following figure shows, if program B tries to read record 3 from file 1, program B must wait because program A is using that record.



RSLW034-0

To ensure that file deadlocks do not occur, you should always release a record before reading a record from another shared change file.

File Change Errors

A change performed by another program sharing the file can be lost. For example, suppose program A reads a record from file X and displays it at display station 1. Then suppose program A reads another record from file X and displays it at display station 2. The second read operation from file X causes the System/36 environment to free the first record. Therefore, a second program-sharing file X can change the first record. Then, if display station 1 reads the record again, and changes the record using the original field values, the changes made by the second program might be lost.

Note: This error can occur when the program is an SRT with acquired work stations or a MRT, and when the program supports more than one work station at the same time.

You can avoid the preceding error by using one of the following techniques:

- Before doing a change, the program should read the record again and check that none of the fields being changed have been changed since the record was displayed for changing. If any of the fields were changed, the program should display the field again for changing or, if possible, use the field values currently in the record to do the change.
- Protect records being changed by establishing a field in the record to be used as a busy indicator that indicates the record is being changed. For example, a busy indicator might be the display station ID and the program name. Subsequent attempts to process the same record should test for the busy indicator and, depending on the value of the indicator, allow or not allow the record to be changed. The busy indicator should be removed from the record when the change is performed by the requesting program or if no change is performed. If records in a file can be changed at the same time by two different programs, both programs should test and use the same busy indicator.

If the program ends abnormally, and you are not going to start it again, you should run another program that turns off the busy indicators in records that were being changed by the program when it ended. This allows programs that check the busy indicator to handle the record properly.

Using Multiple Names to Access a Single File

A program can use multiple names to access one disk file. For example, a program can be written to process two files called FILEA and FILEB, which are the same physical file, by using the following OCL statements:

```
// FILE NAME=FILEA,LABEL=MASTER,DISP=SHRMM
// FILE NAME=FILEB,LABEL=MASTER,DISP=SHRMM
```

Defining a disk file by using two or more names allows you to process one file by two separate

processing methods in one program. For example, one part of the program can process a master file randomly by key, and the other part of the program can process the same file randomly by relative record number.

Using the generalized processing method is another way to process a file by two separate processing methods (randomly and consecutively). For information about the generalized processing method, see “Generalized Processing Method” on page 7-17.

A single physical file accessed by multiple names can also be used in a program when more than one index is used by the program to process the file. For example, if FILEA has an alternative indexed file labeled ALTINDXA, a program can use both files by using the following OCL statements:

```
// FILE NAME-FILEA,DISP-SHRMM  
// FILE NAME-ALTINDXA,DISP-SHRMM
```

Records can be added to or changed in a file accessed by multiple names.

Note: If two names are used to access the same physical file for update from within the same program, a deadlock can occur if the same record is retrieved twice (by two different file names) without first releasing the record.

Programming Considerations

Files and file processing in the System/36 environment provide the user with a System/36-compatible interface to the AS/400 database and data management functions. This section describes some concepts of the AS/400 database and data management function, especially those used by the System/36 environment for the user when System/36 environment applications are started.

Using System/36 Environment Files Library

Unlike System/36 files, all files on the AS/400 system must be stored in a library. In the System/36 environment, all resident (permanent) files are placed in common libraries, called **files libraries**. All job and scratch (nonpermanent) files

are placed in a temporary system-generated library called **QTEMP**.

Note: Every job on an AS/400 system has its own unique QTEMP library, which cannot be shared with another AS/400 job and is deleted when the job ends.

The FILELIB OCL statement creates an interface for users of the System/36 environment to use multiple file libraries. For more information on the FILELIB OCL statement, see the *System/36 Environment Reference* book.

When the library list is not being used for locating files, all System/36 commands and procedures use the current files library and the QTEMP library when working with files. When a System/36 procedure issues an error message saying a file does not exist, the message means that the file could not be found in the files library that was the current files library when the command was entered. When a System/36 procedure issues an error message saying a file already exists, the message means that the file was found in the files library that was the current files library when the command was entered.

Note: If QTEMP is made the current files library or the library list is specified to search for database files and QTEMP is in the library list, the System/36 environment cannot distinguish among permanent files, scratch files, and job files when they share the same file name.

The default name for the files library is QS36F. However, the name of the library to be used as the default files library can be changed using the Change System/36 (CHGS36) CL command or the Change System/36 Environment Attributes (CHGS36A) CL command. For more information on the CHGS36 or CHGS36A CL commands, see Chapter 3, “Configuring the System/36 Environment,” and the *System/36 Environment Reference* book.

Using the Library List Support for Files in the System/36 Environment

Searching the library list for database files rather than restricting the search to the current files library can be accomplished by setting the current

library list search indicator to YES. Do this using one of the following methods:

- Set the default library list search indicator to YES by using the Change System/36 (CHGS36) command or the Change System/36 Environment Attributes (CHGS36A) command.
- Set the session library list search indicator to YES by using the File Library (FLIB) procedure or the File Library (FILELIB) OCL statement.
- Set the current library list search indicator to YES by using the FILELIB OCL statement.

Note: For more information on setting the current library list search indicator, see "Search Order" on page 6-9.

When a batch job is started by either the EVOKE OCL statement or the JOBQ OCL statement, the library list search indicator is automatically passed from the current job to the batch job.

When a MRT job is started, the library list search indicator is automatically passed from the current job to the MRT job.

Note: When routing to an existing MRT job, the library list search indicator of your current job must match the library list search indicator used when the MRT job was started.

The following System/36 environment utilities search the library list if the current library list search indicator is set to YES:

DELETE (\$DELETE)

Uses the library list to locate the file to delete.

RENAME (\$RENAM)

Uses the library list to locate the file to rename.

IDDU (#DSIN)

Uses the library list to locate the file to link.

DFU (#DFMP)

Uses the library list to locate the file to use.

COPYPRT (\$UASC)

Uses the library list to locate the file that contains the copied spool files when used with the NOCOPY option.

TOLIBR (\$MAINT)

Uses the library list to locate the data file from which to copy.

LISTFILE (\$COPY, \$MAINT, \$BICR, \$TCOPY)

Uses the library list to locate files.

The following System/36 environment utilities search the library list in some cases if the library list search indicator is set to YES:

COPYDATA (\$COPY)

Uses the library list for all files except when the output file is new. All new output files are created in the current files library.

SORT (#GSORT)

Uses the library list for all files except when the output file is new. All new output files are created in the current files library.

CATALOG (\$LABEL)

Uses the library list when a catalog of a single file has been requested. If the default to list all files is requested, all the files in the current files library are listed.

FROMLIBR (\$MAINT)

Uses the library list to locate the output file. If the output file is new, the output file is created in the current files library.

TRANSFER (\$BICR)

Uses the library list to locate an existing file to copy to diskette or from diskette. If a new file is to be created from a file on diskette, the new file is placed in the current files library.

RESTORE (\$COPY)

Use to restore a single file or all files:

- Restore a single file.

Uses the library list to search for the files to determine whether the file already exists.

If the file is found and date-differentiated files are not supported or DISP-OLD is not specified, an error is issued.

If the file is found, date-differentiated files are supported and the file does not have the same creation date, the file is restored to the library to which it was previously stored.

If the file is found and DISP-OLD is specified, the file is restored to the library to which it was previously stored.

If the file is not found, the file is restored to the current files library.

- Restore all files.

Only the current files library is searched for files, and all files are restored to the current files library.

SAVE (\$COPY)

Use to save a single file or all files:

- Saving a single file.
Uses the library list to search for the file to be saved.
- Saving all files.
All files in the current files library are saved.

The following utilities always use the current files library regardless of how the library list search indicator is set:

BLDFILE/BLDINDEX (\$FBLD)

All new files are created in the current files library. When creating a logical file with BLDINDEX, the first-level file must also exist in the current files library.

COPYPRT (\$UASF)

If the option to copy from a spool ID is specified, the data file created to store the file is always placed in the current files library.

BLDGRAPH (\$DPGR)

Uses the current files library to create the output file of graphics orders.

The following procedure control expressions and substitution expressions are affected when the library list search indicator is set to YES.

IF DATAF1

Uses the library list to test for the existence of a file or library on diskette.

File size substitution expressions

Uses the library list to locate the file and returns the value of the size of the file.

?FLIB?

Returns the name of the current files library. This expression does not change when the library list is specified.

?SFLIB?

Returns the name of the session files library. This expression does not change when the library list is specified.

Using System/36 Environment Files and AS/400 Files

Data files created in the System/36 environment are AS/400 database physical files. Alternative indexed files created by the System/36 environment are AS/400 database logical files. Both types of files belong to a special class of database files called **program-described files**. Program-described files are database files created without using Data Description Specifications (DDS). Field-level information about the files must be provided by the programs that process the files. For more information about physical files, logical files, or program-described files, see the *DB2 for OS/400 Database Programming* book.

AS/400 database files can be used by programs in the System/36 environment. The only observable differences when using non-System/36 environment files are as follows

- The size of the files will always be in records and will not be rounded up to a block (2560 bytes) boundary.
- For indexed files, the system does not check the key position and length specified in the program to ensure that the key position and length match the keys specified when the file was created.
- For new files intended to be accessed by relative record number (direct files), the user must make sure the file is initialized with a program or with the Initialize Physical File Member (INZPFM) command.

Using non-System/36 environment files can expand user capabilities. For example, the user can use the Create Physical File (CRTPF) command to create a file with a record length greater than 4096 or with more than 3 key fields. See the *CL Reference* book for more information on the Create Physical File (CRTPF) CL command, the Create Logical File (CRTLFL) CL command, and the Initialize Physical File Member (INZPFM) CL command.

A program can also use both program-described and externally described files at the same time. Use the CRTLFL command to create an externally described logical file over a file created using the BLDFILE procedure. Use the BLDINDEX procedure to create an alternative index over an

externally described file created using the CRTPF command.

The BLDINDEX procedure uses field substrings to build its keys and, therefore, is subject to the same restrictions as the CRTLF command when using the SST keyword in DDS. See the *DB2 for OS/400 Database Programming* book for information on field substrings and the *DDS Reference* book for more information on the SST keyword.

AS/400 database data files that do not have a System/36 environment direct, indexed or sequential file organization, can also be used by other System/36 environment utilities. When the \$COPY utility, and the procedures that run it, use such non-System/36 environment files, the output disk file that is created or the file that is saved may be given file attributes which differ from those that you requested or expected. The file attributes that may differ are:

- File organization
- Record length
- Initial allocation size
- Extend size
- Delete capability
- Key position
- Key length
- Duplicate keys

A message that allows you to continue is sent in each of the following cases:

- The output disk file already exists and is externally described, and you have specified DISP-OLD to replace an existing member in it.
- The output disk file already exists and is externally described and keyed, and you are adding a new member to it.
- The output disk file already exists and you have specified DISP-OLD to replace an existing member in it. You have not specified a file organization. The input file is either externally described and keyed, or program described and keyed, but not a System/36 environment indexed file.
- The output disk file already exists and you are adding a new member to it. You have not specified a file organization. The input file is either externally described and keyed or program described and keyed, but not a System/36 environment indexed file.
- The output disk file does not already exist, or you are saving a disk file. You have specified

one of the previously mentioned file attributes, other than file organization. The input file is either externally described or program described, but not a System/36 environment direct, indexed, or sequential file.

In the cases in which the output disk file already exists, the \$COPY utility adds or replaces a member in the file without comparing or changing the indicated file attributes. This is because at least one of the files is a non-System/36 environment file. When only the non-System/36 environment input file exists or is being saved, the \$COPY utility creates a non-System/36 environment output disk file or saved file with the same file attributes. In these situations, the \$COPY utility acts like the AS/400 Copy File (CPYF) command.

To determine whether a file is a non-System/36 environment file, use the CATALOG procedure or the \$LABEL utility. Physical, non-System/36 environment, database data files have a file organization of PHY. Externally described files are indicated by a status value of 9. A key position and key length are identified for keyed files.

Using File Members and Date-Differentiated Files

AS/400 database files can have multiple **members**. A user normally accesses only one member of the file at a time, by identifying the specific member name to be processed. All members of the file must have the same attributes.

On System/36, files did not have multiple members, but multiple versions of the same file existed with different creation dates (date-differentiated files). Date-differentiated files were referred to using the DATE keyword on the FILE OCL statement. Date-differentiated files on System/36 were not required to have the same attributes.

The System/36 environment allows the creation and use of date-differentiated files by making them members within an AS/400 database file. The member names assigned to new files in the System/36 environment are formulated using the

creation date, according to the following arrangement:

yymmdd

where yy is the last 2 digits of the year
mm is the month
dd is the day

When the DATE keyword is coded on a FILE OCL statement in the System/36 environment, the date is used to construct a member name in the previously described format and a member of that name will be used. The actual creation date of the file or member is not used. When the DATE keyword is not coded on the FILE OCL statement, the last (or only) member added to the file as determined by the actual creation date of the member, not the member name, is used.

Notes:

1. On System/36, when no DATE keyword was coded on the FILE OCL statement and date-differentiated files existed, the file with the latest creation date was used.
2. Database file processing outside of the System/36 environment normally uses the first (or oldest) member in a database file when more than one member exists.

Using Override Database File CL Command

The AS/400 Override Database File (OVRDBF) CL command can be used in conjunction with the FILE OCL statement to extend the System/36 environment file processing capabilities. The FILE OCL statement is itself treated like an OVRDBF command by the system. If a file named FILEB exists in the System/36 environment files library (QS36F) with one member named M880601, then the following statement:

```
// FILE NAME-FILEA,LABEL-FILEB
```

will function the same as the following statement:

```
OVRDBF FILE(FILEA) TOFILE(QS36F/FILEB) MBR(M880601)
```

Adding the DBLOCK keyword with a value of 100 to the FILE OCL statement produces the following statement:

```
OVRDBF FILE(FILEA) TOFILE(QS36F/FILEB) MBR(M880601) +  
SEQONLY(*YES 100)
```

Note: If a FILE OCL statement has a name matching a work station or printer file being opened by a program, the system will try to open

a printer or work station file with the name found in the LABEL keyword on the FILE OCL statement. (This occurs because the AS/400 system overrides are independent of file type.) Do not code FILE OCL statements for printer or work station files.

User programs and procedures in the System/36 environment can use the OVRDBF CL command to get to files in libraries other than the Files Library or to use members of files with names other than those having the date format. For example, the following statements tell the system to use a file named MYFILE in a library named MYLIB, and with a member named MYMEMBER, when PGM1 opens a file named FILEA:

```
// LOAD PGM1  
OVRDBF FILE(FILEB) TOFILE(MYLIB/MYFILE) MBR(MYMEMBER)  
// FILE NAME-FILEA,LABEL-FILEB  
// RUN
```

The OVRDBF command must precede the FILE OCL statement it is overriding or an error will be issued. The FILE keyword of the OVRDBF command should match the LABEL keyword of the FILE OCL statement. All keywords on the FILE OCL statement remain valid, except the following:

- The LABEL keyword may be overridden by the TOFILE keyword on the OVRDBF command.
- The DBLOCK keyword may be overridden by the SEQONLY keyword on the OVRDBF command.

See the *Data Management* book for more information on file overrides and override processing. See the *CL Reference* book for more information on the OVRDBF CL command.

Extending Files

The system does not automatically extend files during an access by relative record number. If your application needs to extend files during an access by relative record number, you can do so in your application, using two different access methods on the file at the same time, as follows:

1. Open the file once for sequential output and once for update by relative record number.
2. Add blank or deleted records to the end of your file, using the sequential access when

your program detects that an end of file has occurred during relative record number processing.

- Return to normal relative record number processing for the file.

Note: Refer to the appropriate language reference book for information on checking the return codes for file operations.

The following example shows how extending files might be done.

```

H                                     PGMEXT
F*
F* Sample program to extend a file while accessing by
F* relative record number.
F*
F* The OCL to execute this program would look like:
F* // LOAD PGMEXT
F* // FILE NAME-DFILE,LABEL-MYFILE
F* // FILE NAME-SFILE,LABEL-MYFILE
F* // RUN
F*
FDFFILE UC F      80R      DISK
FSFILE  O F      80      DISK      A
IDFFILE NS
I                                     1  70RECNO
C
C      DO 100
C      ADD 1      CTR      70
C      CTR      CHAINDFILE      98
C 98      EXSR ADDRREC
C 98      CTR      CHAINDFILE      98
C      EXCPTUPDATE
C      END
C      SETON      LR
C*
C* ADD RECORDS AT END OF SEQUENTIAL FILE
C*
C      ADDRREC  BEGSR
C      DO 64
C      EXCPTRECADD
C      END
C      ENDSR
ODFFILE E      UPDATE
O      CTR      7
OSFFILE EADD  RECADD
O      1 ' '

```

The capability to extend a file when it is full is an attribute of the file itself and cannot be overridden on a job basis. When the EXTEND keyword is coded on a FILE OCL statement referencing an existing file, the EXTEND value is ignored.

Note: If DISP-OLD is coded on the FILE OCL statement to rewrite an existing file, EXTEND is not ignored and the attribute of the file is changed according to the value of the EXTEND keyword on the FILE OCL statement.

If the user needs to be able to extend a file that was not created as an extendable file, the CHGPF CL command can be used to change the extend attribute of the file. See the *CL Reference* book for more information on the CHGPF CL command.

Shared Files and System/36 Environment Share Levels

System/36 environment share levels are mapped into AS/400 lock states as defined by the Allocate Object (ALCOBJ) CL command. The following table shows how the mapping is done:

System/36 Environment Share Levels	AS/400 Lock States
SHR	*SHRUPD
SHRMM	*SHRUPD
SHRMR	*EXCLRD
SHRRM	*SHRRD
SHRRR	*SHRNUP
NEW	*EXCL
OLD	*EXCL
No DISP keyword	*EXCL

The Display Job (DSPJOB) command or the Work with Object Lock (WRKOBJLCK) CL command can be used to determine the lock state a job has on a file. See the *CL Reference* book for more information on these commands.

The never-ending program (NEP) attribute does not apply when files are shared in the System/36 environment. For example, when a job in the System/36 environment is forced to wait for a file because another job already has the file allocated with an incompatible share level, no matter what job is holding the file, the results are always the same, as follows:

- If WAIT-NO is coded on the FILE OCL statement, the return code passed back through the OCL expression ?CD? is always 2031.
- If WAIT-YES or no WAIT keyword was specified on the FILE OCL statement, the waiting job will keep waiting until the file is available.

When DISP-NEW is used on a FILE OCL statement in the System/36 environment, no file actually exists at the start of the job. Only the file name is reserved. Only System/36 environment commands and procedures use this reserve of the file name. The reserve of the file name by an

System/36 environment procedure does not prevent an AS/400 command, such as CRTPF, from creating the file. The reserve of the file name will result in the job that reserved the file name being ended later with an error indicating that the file already exists.

Note: The System/36 environment does not check to see whether the application uses a file with a valid share level.

Non-System/36 Environment Programs in the System/36 Environment

You can use non-System/36 environment programs, such as RPG/400, in the System/36 environment and in System/36 environment OCL statements. When you do so, FILE OCL statements can be used, but are not required for files used by the program. If you use FILE OCL statements, you will notice the following differences:

- If the FILE OCL statement is for a new file, no new file is created. If the program attempts to open the file, the program receives a message that the file is not found.
- If the FILE OCL statement uses DISP-OLD and the program attempts to write the file again (open for output), the attributes of the file (such as record length) are not changed to match the attributes specified by the program. The attributes are assumed to match. If the attributes do not match, errors could occur when the program runs.

Shared File Opens within the Same Job

On the AS/400 system, a file can be opened more than once within the same job and designated to share the same **open data path** (ODP). Use of this shared open can improve performance of the overall job in many instances. See the *DB2 for OS/400 Database Programming* book and the *Data Management* book for more specific information about shared opens.

When running procedures in the System/36 environment, shared file open is done automatically for the user whenever possible. When a file is used by a job step in a procedure, the file is kept open

by the system until the next job step starts. If the file is opened by this subsequent job step in a way consistent with the first job step, the program doing the open can be connected with the already existing open data path. If the file is not used by this subsequent job step, the file is closed by the system before beginning this job step. If the same file is opened more than once in the same job step (by coding the same LABEL keyword on more than one FILE OCL statement), the first open of the file is the only one that can be shared by a subsequent job step.

Note: Although the user is normally not aware that this situation is happening, it can be noticeable under certain circumstances. When the system keeps the file open between steps, the system prevents another job from taking the file away from the first job while the system makes the transition to the next job step. The second job requesting the file may have to wait until the current job owning the file reaches a job step that does not use the file or until the job ends. The use of shared file opens can also lead to a file deadlock situation if two jobs, which were sharing a file during concurrent job steps, each attempt to use the file exclusively in their next respective job steps.

The automatic shared open of files is a configurable option that can be turned off using the CHGS36 CL command or the CHGS36A CL command. If turned off, the files do not remain open between job steps. See the "Commands for Configuring the System/36 Environment" on page 3-2 for more information.

If a CL command, issued during a procedure, requires a file that it is processing to be closed, the System/36 environment closes the automatic shared open for the file. If the file (or member) is deleted, renamed, moved, or restored by the CL command, the System/36 environment removes any FILE OCL statements for the file (or member). Any locks that were held through these FILE OCL statements are released. If a job file is renamed, moved, or restored by a CL command, the System/36 environment will not release the disk space occupied by the job file when the job ends. The System/36 environment will not close an automatic shared open of an alternative indexed or logical file which is based on the file being processed by the CL command.

If you want to prevent the automatic shared open from occurring for a particular file in a job step, specify BYPASS-PRF on the FILE OCL statement for the file in that job step.

If you want to prevent the automatic shared open from occurring on a job or session basis, use the IBM-supplied program QEXSHRO. See the appendix on IBM-supplied programs in the *System/36 Environment Reference* book for more information about QEXSHRO.

You should not use the RCLRSC command with LVL(*CALLER) specified to close files with shared opens. This command bypasses the System/36 environment file processing and causes the job to end abnormally when the System/36 environment attempts to use the files closed by the RCLRSC command.

Duplicate Keys and Key Sorting

On System/36, users could bypass duplicate key checking for indexed files designated not to allow duplicate keys, to improve performance. This bypass was done using the BYPASS keyword on the FILE OCL statement. A significant difference in performance quality existed between files where duplicate checking was performed and files where duplicate checking was not performed, primarily because keys within the index where duplicate checking was not performed were not maintained in order. The KEYSORT procedure had to be used periodically to order the keys in the index.

On the AS/400 system, and in the System/36 environment, keys in indexed files are always maintained in order. To maintain database integrity, no duplicate keys are ever allowed to be inserted in a file designated as not allowing duplicate keys. This approach causes the following differences from the System/36:

- The KEYSORT procedure, although provided for System/36 compatibility, does not do any function in the System/36 environment.
- The BYPASS keyword on the FILE OCL statement is ignored, to maintain database integrity. Not doing the duplicate key checking

does not significantly improve performance. If you have an application that is dependent on being able to insert duplicate keys, create the file to allow duplicate keys.

Remote Files

Remote files can be used in the System/36 environment by using the AS/400 distributed data management (DDM) support. See the *Distributed Data Management* book for additional information about DDM.

Note: To use DDM files in the System/36 environment, no changes are required to programs or procedures. DDM files, however, must also reside in the System/36 environment files library to be used by System/36 environment commands and procedures.

Moving from System/36 to the System/36 Environment

The default EXTEND value for the BLDFILE procedure and the FILE OCL statement has been changed from 0 to a default value of 32 767 divided by the record length of the file. Files cannot be automatically extended during an access by relative record number. See “Extending Files” on page 7-35 for more information.

Date-differentiated alternate index files are not supported. See “Multiple Indexes for a File” on page 7-9.

If you have a file named QTEMP, it must be renamed.

All other files migrate. BASIC files that contain 1-byte, 2-byte, 3-byte, and 4-byte centesimal floating-point data need to have the centesimal floating-point data converted to IEEE floating-point data.

No error is given when a System/36 environment program uses an alternative index file and the key length specified in the program does not match the total key length of the alternative index file.

Chapter 8. Folders and Data Dictionaries

This chapter describes:

- What a folder is
- Folders in the System/36 environment
- Folder procedures
- What a data dictionary is
- Data dictionaries in the System/36 environment
- Data dictionary procedures

A **folder** is a named object that is used to manage other objects. Folders are document library objects (DLO) that are directories to documents and other folders. A folder is similar to a library in that it is a directory.

Migration Considerations

Use the System/36 Save Folder (SAVEFLDR) procedure on System/36 to save document folders and interactive data definition utility (IDDU) dictionary folders on diskette or tape. A **data definition** in IDDU is information that describes the contents and characteristics of a field, record format, or file. A data definition can include such things as field names, lengths, and data types. Use the Restore System/36 Folder (RSTS36FLR) command on the AS/400 system to restore the folder. Migrated document folders will exist in folders on the AS/400 system, while migrated IDDU dictionary folders will exist on libraries containing a data dictionary.

Note: You cannot migrate folders from the AS/400 system to System/36.

Using Folders

This section describes characteristics of and activities associated with folders.

Folders and Folder Members

There are several types of folders on System/36:

- Document
- Data dictionary
- Mail
- Mail log

The AS/400 system has only one type of folder, the document type. The document folder is used to store documents created by office and Client Access/400 products.

See the *System/36 Migration Planning* book for more information about the storage of System/36 data dictionary folders on the AS/400 system.

The information stored in mail folders and mail log folders on System/36 is managed by the OfficeVision for OS/400 product. See the *Using OfficeVision/400* book for more information.

Securing Folders

Refer to the *Security – Reference* book for more information on securing folders.

Creating a Folder

Use the TEXTFLDR procedure to create a folder.

Accessing a Folder

Use the TEXTDOC procedure to access folders.

Listing Folder Information

Use the CATALOG procedure to list information about a folder that is on disk, diskette, or tape. You must be in the System Distribution Directory to obtain information on folders.

Deleting a Folder

Use the DELETE procedure to remove a folder. You must be in the System Distribution Directory to do this. You can use the Work with Directory (WRKDIR) CL command to change the System Distribution Directory.

Renaming a Folder

Use the RENAME procedure to rename a folder. You must be in the System Distribution Directory to do this.

Reorganizing a Folder

Use the AS/400 Reorganize Document Library Object (RGZDLO) command to reduce documents and folders to a minimum size. When a document is reorganized, unused storage may be removed.

Saving and Restoring Folders and Folder Members

The following describes how System/36 saves and restores functions are supported in the System/36 environment:

- System/36 SAVEFLDR procedure (for folders) and ARCHIVE procedure (for folder members) are supported by the AS/400 Save Document Library Object (SAVDLO) command.
- System/36 RESTFLDR procedure (for folders) and RETRIEVE procedure (for folder members) are supported by the AS/400 Restore Document Library Object (RSTDLO) command.

Using Data Dictionaries

On the AS/400 system, a data dictionary is located in a library as a data dictionary object and a set of associated database files. A library contains a maximum of one data dictionary, whose name matches the name of the library. As on System/36, the data dictionary contains definitions that describe data fields, record formats, and files. The operations that can be performed on the data dictionary and the definitions the data dictionary contain are described in the following sections.

Working with a Data Dictionary

Use the IDDUDCT procedure to do the following operations:

- Create a data dictionary
- Secure a data dictionary
- Delete a data dictionary
- Print a data dictionary

Working with Data Dictionary Definitions

Use the IDDUDFN procedure to do the following operations:

- Create a definition
- Change a definition
- Copy a definition
- Rename a definition
- Delete a definition
- Print a definition
- View cross-reference information for a definition

Using Data Dictionary Definitions

Use the IDDUDISK procedure to do the following operations:

- Create a database file from a file definition stored in the data dictionary
- Edit the records in a database file that has been created from a file definition

Use the IDDULINK procedure to associate existing program-described database files with a data dictionary file definition that describes the file. Once linked, the file can be queried and the file's records can be edited as though the file had been originally created from a file definition.

Use the IDDUPRT procedure to print definitions in a data dictionary.

Saving and Restoring a Data Dictionary

Since the AS/400 system data dictionary is no longer a folder, you *cannot* use the SAVEFLDR and RESTFLDR procedures to save and restore a data dictionary. Instead, use the SAVLIB and RSTLIB procedures.

Programming Considerations

See the *System/36 Environment Reference* book for additional information on the commands for accessing and maintaining IDDU data dictionaries and folders.

Coexistence Considerations

You can migrate IDDU data dictionaries and document folders from System/36 to the AS/400 systems. You *cannot* migrate IDDU data dictionaries and folders from the AS/400 system to the System/36, since dictionaries are not in folders in the AS/400 system.

Moving from System/36 to the System/36 Environment

Use the SAVEFLDR procedure on System/36 to save DisplayWrite/36* document folders and interactive data definition utility (IDDU) data dictionary folders on diskette or tape. Use the Restore System/36 Folder (RSTS36FLR) command to restore the folder on an AS/400 system.

Chapter 9. Diskette Storage

This chapter contains the following information about the System/36 environment:

- How to use diskette storage
- Diskettes to use with the system
- Formats to store data on diskette
- Programming tips and techniques in processing diskettes

Use diskettes to create backup copies of information. You can also use diskettes for off-line storage of files and libraries, or to transfer information to other systems or devices.

Diskette Types and Storage Capacities

The AS/400 system supports several types of diskettes:

- Diskette 1 is an 8-inch single-sided, single-density diskette.
- Diskette 2D is an 8-inch double-sided, double-density diskette. Double-density means that one side of a diskette 2D can store twice as much information as a diskette 1.
- Diskette 2HC is a 5-1/4 inch double-sided, high-capacity diskette. Diskette 2HC is also referred to as a 96 tpi (tracks per inch) diskette, 2QD (quad-density) diskette, and 2HD (high-density) diskette.

Your needs and your system determine the diskette you use. Depending on the size of your files and libraries, you can place several files on one diskette or a large file on several diskettes.

The system must prepare the diskettes before you can use them. Use the Initialization (INIT) procedure to prepare the diskettes. With the INIT procedure you can specify storage capacity for the diskettes using the FORMAT and FORMAT2 parameters. The INIT procedure is discussed further in "Programming Considerations" on page 9-4.

The following table lists the storage capacities and formats of the different diskettes you can use:

Diskette Type	INIT Procedure Parameter	Number of Bytes per Sector	Number of Sectors	Total Bytes
1	FORMAT	128	1924	246 272
1	FORMAT2	512	592	303 104
2D	FORMAT	256	3848	985 088
2D	FORMAT2	1024	1184	1 212 416
2HC	FORMAT	256	3848	985 088
2HC	FORMAT2	1024	1184	1 212 416

Diskette Exchange Formats

An **exchange format** is a set of rules for the content of the header record and the physical organization of the diskette. Exchange formats are supplied so you can exchange data between systems. The system with which you exchange data determines the diskette format to use. The system supports the following types of exchange formats:

- Basic data exchange (1)
- H-data exchange (2D and 2HC)
- I-data exchange (1, 2D, and 2HC)

For more detailed information about diskette formats, see the *Tape and Diskette Device Programming* book.

For information about the exchange format required to exchange data with another system, see the correct book for that system.

Basic Data Exchange Format

Basic data exchange format files ensure that you can exchange diskettes between systems capable of reading from and writing to diskette 1 diskettes.

Basic data exchange has the following characteristics:

- The diskette sector is 128 bytes (diskette 1).
- All records in the file must be the same length. The maximum record length of the file you can put on diskette is 128 bytes (diskette 1).

- Records are not blocked and cannot span diskette sectors.
- The file name must be 8 characters or less.

Use the TRANSFER procedure to read from and write to diskettes in the basic data exchange format. Use the \$MAINT utility program to read and write library members in basic data exchange format (using record-mode). The *System/36 Environment Reference* book has more information about the TRANSFER procedure and the \$MAINT utility program.

H-Data Exchange Format

H-data exchange format files allow diskettes to be exchanged between systems capable of reading from and writing to diskette 2D and 2HC diskettes.

Note: H-data exchange is also called basic data exchange on System/36.

H-data exchange has the following characteristics:

- The diskette sector is 256 bytes (diskette 2D or 2HC).
- All records in the file must be the same length. The maximum record length of the file you can put on diskette is 256 bytes (diskette 2D or 2HC).
- Records are not blocked and cannot span diskette sectors.
- The file name must be 8 characters or less.

You can use the TRANSFER procedure to read from and write to diskettes in this format.

I-Data Exchange Format

I-data exchange format files have requirements assuring that diskettes may be exchanged between systems capable of reading from and writing to diskette 1, and 2D and 2HC diskettes.

I-data exchange has the following characteristics:

- The diskette sector is 128 bytes or 512 bytes (diskette 1), or 256 bytes or 1024 bytes (diskettes 2D, and 2HC).
- All records in the file must be the same length. The maximum record length of the file you can put on diskette is 4096 bytes.

- Records are blocked and can span diskette sectors. That is, several records and parts of records can be placed in a diskette sector, or a record can extend from one sector to another. However, records cannot span diskette volumes.

- The file name must be 8 characters or less.

You can use the TRANSFER procedure to read from and write to diskettes in this format.

Storing Information on Diskette

When you save a file or library on diskette, the system creates a file that contains the file or library you saved. The diskette volume table-of-contents (VTOC) includes the following information about the diskette file:

- Name of the file
- Type of file created
- Date the file was created
- Size of the file
- Record length of the file
- File expiration date
- Sequence number of the diskette (if the file is contained on more than one diskette)

Types of Diskette Files

The following list shows common types of diskette files you can create with the system:

COPYFILE

Created when you use the Save System/36 File (SAVS36F) command to save a disk file. The format of the diskette file is the same as if the SAVE procedure or the \$COPY utility program had created it on System/36. These files can be exchanged with System/36 (RESTORE procedure) or an AS/400 system using the Restore System/36 File (RSTS36F) command or the RESTORE procedure in the System/36 environment.

EXCHANGE

Created when you use the TRANSFER procedure (or the \$BICR utility program) to copy a disk file in basic data exchange or H-data exchange format.

IFORMAT

Created when you use the TRANSFER procedure (or the \$BICR utility program) to copy a disk file in I-exchange format.

LIBRFILE

Created when you use the Save System/36 Library Member (SAVS36LIBM) command to save procedure and source members from a source file, or the FROMLIBR procedure to create a record-mode file. The format of the diskette file is the same as if the FROMLIBR procedure or the \$MAINT utility program had created it on System/36. These files can be exchanged with System/36 using the TOLIBR procedure or with the AS/400 system using the TOLIBR procedure or the Restore System/36 Library Member (RSTS36LIBM) command.

SAVELIBR

This file cannot be created on the AS/400 system, but can be restored with the RSTS36LIBM command.

SAVEFLDR

This file cannot be created on the AS/400 system, but can be restored with the Restore System/36 Folder (RSTS36FLR) command.

SAVE/RESTORE

The system creates these files when you use the SAVE, SAVELIBR, or FROMLIBR (sector-mode) procedures in the System/36 environment. The format of the diskette file is the same as if you used the Save Object (SAVOBJ) or Save Library (SAVLIB) commands. The files can be exchanged only with another AS/400 system using the System/36 environment RESTORE, RESTLIBR, or TOLIBR procedures, or the Restore Object (RSTOBJ) or Restore Library (RSTLIB) commands.

Diskette Data Compression

Diskette data compression enhances the performance of the SAVE or RESTORE procedure by compressing the duplicate character strings in your data files. To **compress** in the System/36 environment is to replace repetitive characters in a file with control characters so that the file takes up less space when saved to diskette.

If you have files with many duplicate characters, diskette data compression can reduce the number

of diskettes you need and the amount of time needed to process your data.

To run diskette data compression for data files, specify the COMPRESS parameter in the SAVE procedure. The system automatically decompresses data when you restore the compressed files.

You can compress data only on 2D and 2HC diskettes that have been initialized to FORMAT2.

Use the RSTS36F command or the RESTORE procedure in the System/36 environment to accept compressed COPYFILE diskettes in the System/36 format. You can also use the SAVS36F command to create COPYFILE diskettes in the System/36 compression format. This reduces the number of diskettes necessary when files are saved from the AS/400 system to be restored on a System/36. If you elect to compress your files, use 2D or 2HC diskettes with 1024 bytes per sector.

Diskette File Expiration Dates

When you copy a file or library to diskette, one of the parameters you can specify is the number of retention days. This parameter specifies how long the system protects the diskette file. The system uses the value specified for retention days to calculate the expiration date.

The calculation is:

expiration date = program date + retention days

If you specify 999 for the retention days parameter, the file is considered permanent and never automatically expires. The **program date** in the System/36 environment is the date associated with a program (job step).

The system examines the expiration date each time you create a file on diskette. If the date has passed, the system writes over the file. For example, a diskette file named FILE1 has an expiration date of 14 July 1988. If you copy another file to that diskette before 14 July 1988, FILE1 remains on the diskette. However, if you copy a third file to that diskette on or after 14 July 1988, FILE1 is no longer stored on the diskette. The system automatically deletes FILE1 because its retention period has ended.

The FILE OCL statement for diskette files in the *System/36 Environment Reference* book has more information about retaining diskette files.

Programming Considerations

This section describes procedures and techniques you can use to process diskettes.

There is no System/36 environment high-level language support for diskette. For your programs to use information on diskette, you must:

1. Copy the diskette information to a disk file
2. Run your program to use the disk file
3. Copy the disk file back to diskette

Preparing Diskettes

You prepare a diskette for use by the system using the INIT procedure. The INIT procedure determines the format of the diskette (the number of bytes that can be stored on each sector). It also deletes information on the diskette.

You may or may not have to prepare your diskettes. If you do not know the format of your diskettes, use the CATALOG procedure to check the diskettes. See the *System/36 Environment Reference* book for information about the CATALOG procedure.

When you use the INIT procedure, the system allows you to specify identifying information to be placed on the diskette. This identifying information is:

- A 6-character name called the **volume ID**. You specify the volume ID on system procedures to make sure you use the proper diskette. Valid characters for the ID are A through Z and 0 through 9.
- A 14-character name called the **owner ID**. Use the owner ID to determine the owner of a diskette. This ID is not checked by the system procedures, but is displayed by the CATALOG procedure. Valid characters for the ID are A through Z and 0 through 9.

If you specify values for the STARTING LOCATION and ENDING LOCATION parameters, the values are checked for syntax errors. However, the values are ignored when you use

the procedure since the system has only a single-slot diskette drive.

The diskette unit can be secured on the AS/400 system. You must have authority to the device before using the INIT procedure.

The INIT procedure can be used to change the volume ID or the owner ID without affecting the data on the diskette, and to remove all files without changing the volume ID, owner ID, or format.

Note: The AS/400 diskette-related save/restore procedures require double-density diskettes, formatted with FORMAT2 (1024 bytes per sector).

The INIT procedure in the *System/36 Environment Reference* book has more information about preparing diskettes.

Copying, Saving, and Restoring Information

The System/36 environment supplies the following procedures to let you copy, save, or restore information using diskettes. The *System/36 Environment Reference* book has more information about these procedures.

Copying Information: Use the following procedures to copy information to or from diskettes:

COPY11

Copies an entire diskette to one or several diskettes.

FROMLIBR

Copies one or more library members to diskette. If the diskette file created is a sector-mode file, it can only be copied back to the System/36 environment using the TOLIBR procedure. If the file is a record-mode file, it can be exchanged with System/36 using the TOLIBR procedure, or with the AS/400 system using the TOLIBR procedure or the Restore System/36 Library Member (RSTS36LIBM) command.

JOBSTR

Copies a diskette file (that contains one or more procedure members and source members) to a specified library. The diskette

must have been created in the System/36 environment with the FROMLIBR procedure.

TOLIBR

The System/36 environment TOLIBR procedure copies library members from a diskette file to a library. The diskette must have been created with the FROMLIBR procedure on System/36 or in the System/36 environment, or with the Save System/36 Library Member (SAVS36LIBM) command on the AS/400 system.

TRANSFER

Copies disk files to or from diskettes in either basic data exchange or I-data exchange format.

The diskette unit can be secured on the AS/400 system. You must have authority to the device before using the COPYI1, FROMLIBR, JOBSTR, TOLIBR, or TRANSFER procedures.

Saving Information: Use the following procedures to save information to diskettes:

SAVE

Saves disk files on a diskette. The files can be restored only to the AS/400 system.

SAVELIBR

Saves an entire library to a diskette. The library can be restored only to the AS/400 system.

SAVS36F

System command. Saves one or more files on a diskette that can be restored to a System/36. You can compress the files into a System/36-compatible format to reduce the number of diskettes required. You can restore the file or files the AS/400 system by using the Restore System/36 File (RSTS36F) command or the RESTORE procedure in the System/36 environment.

SAVS36LIBM

System command. Saves source or procedure members on a diskette that can be restored to a System/36. The library members can also be restored to the AS/400 system by using the Restore System/36 Library Member (RSTS36LIBM) command.

The diskette unit can be secured on the AS/400 system. You must have authority to the device

before using the SAVE or SAVELIBR procedure or the Save System/36 File (SAVS36F) or Save System/36 Library Member (SAVS36LIBM) commands.

Restoring Information: Use the following procedures to restore information from diskettes:

RESTLIBR

Restores an entire library from a diskette file to disk. You must have created the diskette file with the SAVELIBR procedure on the AS/400 system.

RESTORE

Restores disk files from a diskette to disk. You must have created the diskette files with the SAVE procedure in the System/36 environment or on System/36.

RSTS36F

System command. Restores a file or a group of files saved to diskette on System/36 using the SAVE procedure, or restores a file saved on the AS/400 system using the SAVS36F system command.

RSTS36LIBM

System command. Restores source or procedure members from diskette that were saved on System/36 using the FROMLIBR procedure or the SAVELIBR procedure or that were saved on the AS/400 system using the FROMLIBR (record-mode) procedure or the SAVS36LIBM command.

The diskette unit can be secured on the AS/400 system. You must have authority for the device before using the RESTLIBR or RESTORE procedures or the RSTS36F or RSTS36LIBM commands.

Listing Information from Diskette

The System/36 environment supplies the following procedure to let you list information from a diskette. The *System/36 Environment Reference* book has more information about these procedures.

Use the following procedures to list information from a diskette:

CATALOG

Lists the names of files, saved libraries, and saved folders on a diskette.

LISTDATA

Lists the contents of a saved file from a diskette. You must have created the diskette file with the SAVE procedure in the System/36 environment or on System/36.

LISTFILE

Lists the contents of a basic data exchange file, I-data exchange file, saved file, saved library member, or saved library located on diskette. For saved files, you must have created the diskette files with the SAVE procedure in the System/36 environment or on System/36. For saved library members, you must have created the diskette file with the FROMLIBR procedure in the System/36 environment. For saved libraries, you must have created the diskette file with the SAVELIBR procedure in the System/36 environment.

The diskette unit can be secured on the AS/400 system. You must have authority to the device before using the CATALOG, LISTDATA, or LISTFILE procedures.

Removing Information from Diskette

Use the following procedures to remove information from diskettes:

DELETE

Removes a single file or all files from a diskette.

INIT

Checks for active files before removing all files from a diskette. (See "Preparing Diskettes" on page 9-4.)

Allocating the Diskette Drive to a Job

Use the ALLOCATE OCL statement to dedicate the diskette drive to a job. For example, you have a procedure that saves three files (the diskette has a volume ID of VOL001):

```
SAVE FILE1,,,VOL001
SAVE FILE2,,,VOL001
SAVE FILE3,,,VOL001
```

You may lose control of the diskette drive between SAVE procedures. For example, after FILE1 is saved but before FILE2 is saved, another job on the system could gain control of the diskette drive. If this happens, your job will have to wait until the other job has finished using the diskette drive before your job can save FILE2.

Use the ALLOCATE OCL statement to keep control of the diskette drive throughout the three SAVE procedures:

```
// ALLOCATE UNIT-I1
SAVE FILE1,,,VOL001
SAVE FILE2,,,VOL001
SAVE FILE3,,,VOL001
```

To avoid allocating the diskette drive longer than necessary, use the DEALLOC OCL statement to deallocate the diskette drive. For example, your job might save three files and then run another job step that did not use the diskette drive. You would use the DEALLOC OCL statement to allow other jobs to use the diskette drive, as follows:

```
// ALLOCATE UNIT-I1
SAVE FILE1,,,VOL001
SAVE FILE2,,,VOL001
SAVE FILE3,,,VOL001
// DEALLOC UNIT-I1
*
// LOAD PROG1
// RUN
```

In this example, if you do not specify the DEALLOC OCL statement, your job holds the diskette drive until it ends. While PROG1 is running, other jobs are needlessly prevented from using the diskette drive. See the *System/36 Environment Reference* book for more information about the ALLOCATE and DEALLOC OCL statements.

Coexistence Considerations

This section contains the following coexistence considerations:

- Restoring the AS/400 system to System/36
- Restoring System/36 to the AS/400 system
- Restoring the AS/400 system to the System/36 environment

Restoring the AS/400 System to System/36

You can move both data files and source and procedure library members from an AS/400 system to System/36.

There are two ways to use diskettes to move data files from an AS/400 system to System/36:

- Copy the data to a basic or I-data exchange format diskette using the System/36 environment TRANSFER procedure or the Copy to Diskette (CPYTODKT) command. On System/36, use the TRANSFER procedure to copy the data back to a disk file.
- Save the data files to diskette using the SAVS36F command. On System/36, use the RESTORE procedure to restore the data files.

There are two ways to use diskettes to move source and procedure members from an AS/400 system to System/36:

- Copy the members to a record-mode diskette file using the FROMLIBR procedure in the System/36 environment. On System/36, use the TOLIBR procedure to copy the members from diskette back to a library.
- Save the source and procedure members to a diskette using the SAVS36LIBM command. On System/36, use the TOLIBR procedure to copy the members from diskette back to a library.

Restoring System/36 to the AS/400 System

You can move data files, and source and procedure library members, from System/36 to an AS/400 system.

There are two ways to use diskettes to move data files from System/36 to an AS/400 system:

- Copy the data to a basic or I-exchange format diskette on System/36 using the TRANSFER procedure. On the AS/400 system use the System/36 environment TRANSFER procedure or the Copy from Diskette (CPYFRMDKT) command to copy the data back to a disk file.

- Save the data file to diskette on System/36 using the SAVE procedure. On the AS/400 system, use the System/36 environment RESTORE procedure or the RSTS36F command to restore the data back to a disk file.

There are two ways to use diskettes to move source and procedure members from System/36 to an AS/400 system:

- Save the library that contains the source and procedure members to a diskette on System/36 using the SAVELIBR procedure. On the AS/400 system, use the RSTS36LIBM command to copy the members back to a library.
- Save the source and procedure members to a diskette on System/36 using the FROMLIBR procedure. On the AS/400 system, use the TOLIBR procedure or the RSTS36LIBM command to copy the members from diskette back to a library.

Restoring the AS/400 System to the System/36 Environment

You can move data files, and source and procedure library members, from an AS/400 system to another AS/400 system within the System/36 environment.

There are two ways to use diskettes to move data files from an AS/400 system to another AS/400 system:

- Copy the data to a basic or I-exchange format diskette using the TRANSFER procedure. On the other AS/400 system, use the TRANSFER procedure to copy the data to a disk file.
- Save the data file to diskette using the SAVE procedure. On the other AS/400 system, use the RESTORE procedure to restore the data back to a disk file.

There are two ways to use diskettes to move source and procedure members from an AS/400 system to another AS/400 system:

- Save the library that contains the source and procedure members to a diskette using the SAVELIBR procedure. On the other AS/400 system, use the RESTLIBR procedure to restore the library onto the system.

- Save the source and procedure members to a diskette using the FROMLIBR procedure. On the other AS/400 system, use the TOLIBR procedure to copy the members from diskette back to a library.

Moving from System/36 to the System/36 Environment

You can use diskettes as a migration medium and copy or restore files on diskette to the System/36 environment, depending on the file type. See "Types of Diskette Files" on page 9-2 for more information.

Chapter 10. Magnetic Tape Storage

This chapter describes how to use magnetic tape in the System/36 environment.

You use tapes to create backup copies of information. You can also use tapes to:

- Store files and libraries offline
- Transfer information to other systems

Tapes must be prepared before they can be used. You can use the TAPEINIT procedure to do this. See “Preparing Tapes” on page 10-5.

Tape Drives Supported

For a complete description of the tape drives supported on the AS/400 system, see the *Backup and Recovery – Advanced* book.

Tape Formats

The tape format describes how information is stored on the tape. The format of a tape reel is set when the tape is initialized. The two formats supported are IBM standard label and nonlabeled format.

The system initializes tapes differently depending on your system unit and tape drive.

IBM Standard Label

All supported tape drives use the IBM standard label format. Using IBM standard label tapes allows you to easily determine the names of files stored on tape, and other information. All save and restore operations require IBM standard label tape.

When data is stored in a file on an IBM standard label tape, the record is created in the format shown in Figure 10-1.

The parts in Figure 10-1 are defined as follows:

Tape Volume Label

Written on tape when the tape is initialized. It contains information about the tape volume (reel).

Tape Marks (TM)

Used to separate the various parts of a file. There are two tape marks at the end of the tape.

Tape Header Labels

Written on the tape when the file is created. There are two types of header labels:

- **System header labels** are created by the system and contain information about the tape file. They are required.
- **User header labels** are created by the application and contain user-defined information about the file, library, or member in the tape file. They are optional and are created by the FROMLIBR procedure when creating a record-mode file on the AS/400 system, and the Save System/36 File (SAVS36F) or Save System/36 Library Member (SAVS36LIBM) commands.

Data

Data for the file can be blocked or unblocked.

Tape Trailer Labels

Written on the tape when the file is created. There are two types of trailer labels:

- **System trailer labels** are created by the system and contain information about the tape file. They are required.
- **User trailer labels** are created by the application and contain user-defined information about the file, library, or member in the tape file. They are optional and are created by the FROMLIBR procedure when creating a record-mode file on the AS/400 system, and by the SAVS36F or SAVS36LIBM commands.

Tape Volume Label	T M	FILEA Tape Header Label	T M	FILEA (data)	T M	FILEA Tape Trailer Label	T M	FILEB Tape Header Label	T M	FILEB (data)	T M	FILEB Tape Trailer Label	T M	T M
-------------------------	--------	-------------------------------	--------	-----------------	--------	--------------------------------	--------	-------------------------------	--------	-----------------	--------	--------------------------------	--------	--------

RSLW070-1

Figure 10-1. Layout of Two Files on a Standard Label Tape

System header and trailer labels can include the following information about the tape file:

- Name of the file
- Date the file was created
- Expiration date of the file
- Sequence number (SEQNUM) of the file, either from the beginning of the tape or the beginning of a sequence of tapes
- Volume sequence number of the tape, if the file is on more than one tape (a multivolume file)
- Record length of tape file
- Record format of tape file
- Block length of tape file (system trailer header only)

The user header and trailer labels can contain information about the:

- Type of file created
- Size of file created

Nonlabeled

Use nonlabeled tapes when exchanging data with systems that do not support the IBM standard label tapes.

When you copy a file to a nonlabeled tape, the system does not create volume, header, or trailer labels. You must keep track of the tape information normally stored in these labels. The following figure shows the file format of a nonlabeled tape.

FILEA (data)	T M	FILEB (data)	T M	FILEC (data)	T M	T M
SEQNUM 1		SEQNUM 2		SEQNUM 3		

RSLW071-2

Each file is assigned a sequence number (SEQNUM) and a tape mark (TM).

Note: Some nonlabeled tapes have a leading tape mark. In order to get the first data file, SEQNUM 2 must be specified to read FILEA data.

Tape Files

The types of tape files you can create in the System/36 environment are shown in the following list. Except where noted, the formats of these tape files are unique to this system. The information can be exchanged only with another AS/400 system in the System/36 environment.

COPYFILE

The COPYFILE file type is created when you use the SAVS36F command to save a disk file. The format of the tape file is the same as if the SAVE procedure or the \$COPY utility program had created it on the System/36. The files can be exchanged with another AS/400 system using the RSTS36F command or the RESTORE procedure in the System/36 environment, or exchanged with a System/36 using the RESTORE procedure.

EXCHANGE

The EXCHANGE file type is created when you use the System/36 environment TAPECOPY procedure. The format is the same as created by TAPECOPY procedure on System/36. This type allows information to be exchanged with another system, not necessarily System/36.

LIBRFILE

The LIBRFILE is created when you use the Save System/36 Library Member (SAVS36LIBM) command to save procedure and source members from a source file or the FROMLIBR procedure to create a record-mode file. The format of the tape file is the same as if the FROMLIBR procedure or the \$MAINT utility program had created it on System/36. The files can be exchanged with another System/36 using the TOLIBR procedure or with the AS/400 system using the

TOLIBR procedure or the Restore System/36 Library Member (RSTS36LIBM) command.

SAVELIBR

The SAVELIBR file type cannot be created in the System/36 environment, but can be restored using the System/36 environment RSTS36LIBM command.

SAVEFLDR

The SAVEFLDR file type cannot be created in the System/36 environment, but can be restored using the System/36 environment RSTS36FLR command.

Save/Restore

The system creates these files when you use the SAVE, FROMLIBR, or SAVELIBR (sector-mode) procedure in the System/36 environment. The format of the tape file is the same as if you used the Save Object (SAVOBJ) or Save Library (SAVLIB) commands. The files can only be exchanged with another AS/400 system using the System/36 environment RESTORE, RESTLIBR, or TOLIBR procedures, or the Restore Object (RSTOBJ) or Restore Library (RSTLIB) commands.

Exchanging Tape Files with Other Systems

To exchange tape files with other IBM systems, you should use IBM standard labeled tapes.

If you exchange tape files with another system, it is possible the tape has been written with volume/file security. For more information, see "Tape Security" on page 10-4.

You can specify any of the following processing methods to be used when the system reads tapes. These methods are specified in the TAPECOPY procedure. See the *System/36 Environment Reference* book for more information about TAPECOPY.

- **IBM standard label processing.** Specifies that the header labels are to be used to read the tape.

Note: Files that have only a header 1 label can be processed if the record processing information is supplied.

- **Nonlabeled tape processing.** Specifies that the tapes have no labels and that marks on the tape indicate where the files are stored.

- **Nonstandard label processing.** Specifies that the tapes have labels, but the labels are not IBM standard labels. The system can read only one file from a nonstandard labeled tape.

The system ignores nonstandard labels and reads the tape starting from the first mark on the tape to the second mark on the tape. Thus, only the first file is read.

- **Bypass label processing.** Specifies that the tape has IBM standard labels but that label information is to be ignored. Instead, sequence numbers on the tape are to be used to process the file.

You can use this method when you do not know the volume ID of the tape, or the name of the tape file.

Tape File Expiration Dates

When you copy a file or library to tape, one of the parameters you can specify is the number of retention days. This parameter specifies how long the system should protect the tape file. The system uses the value specified for retention days to calculate the expiration date.

The calculation is:

Expiration date = Program date + Retention days

If you specify 999 for the retention days parameter, the file is considered permanent and does not automatically expire.

When you prepare a tape and you specify date checking, the system examines the expiration date of the first file on an IBM standard labeled tape before initializing the tape. If the file has expired, the system clears the expired file, and any other files that may be on the tape. When the system continues writing a file from one tape to another and files exist on the continued reel, it checks the expiration date of the first file on the continued reel. If the first file has expired, all the files can be written over, even if subsequent files have not expired.

See the FILE OCL statement for tape files in the *System/36 Environment Reference* book for more information about tape file retention.

Tape Security

You can secure access to an entire tape or to specified files using the methods described in the following sections.

Securing Write Access to Tapes

On the reel-to-reel tape drives, you control write access by using the write-enable rings. Write-enable rings are plastic rings fit into the center of the tape reel. To allow information to be written on the tape, insert the write-enable ring. To prevent information from being added to or changed on the tape, remove the write-enable ring from a tape reel. The information on the tape can then only be read. This is the only way to ensure no active tape files are written over.

On the cartridge tape drives, you control write access by using the write-protect device on the cartridge. Set the device to write-enable to allow information to be written onto the tape. Set the device to write-disable to prevent any information from being added to or changed.

Using Tapes Secured by Other Systems

If tapes created on systems other than the AS/400 system were secured using the following methods, they will remain secure in the System/36 environment.

Securing Access to Tapes: Security access on IBM standard label tapes is controlled by a code of X'00', X'40', or X'F0' in the tape volume label (VOL1). If one of these codes is not found, only a security officer can continue using the tape. The system checks volume security when processing standard label or bypass label tapes only.

Securing Read/Write Access to Files:

Security access to files on IBM standard label tapes is controlled by a 0, 1, or 3 in the header label (HDR1) for each file:

- 0 No security access to the file.
- 1 You must be a security officer to read or write to the file.

- 3 You can read the file but you must be a security officer to add to the file.

File security will be checked when processing standard label, or bypass labeled tapes only.

Programming Considerations for Tape Processing

Only IBM-supplied procedures and programs, or procedures you have written that contain a System/36 procedure, can use the tape drive. Because there is no System/36 environment high-level language support for information on tape, you must:

1. Copy the tape information to a disk file.
2. Run your program to use the disk file.
3. Copy the disk file back to tape.

Automatically Advancing to Next Tape Drive

The AUTO parameter on procedures and on the tape FILE statement determine whether or not processing should automatically advance to the next tape drive when processing is complete on the first drive. When using these procedures or the tape FILE statement, the following should be considered:

- The default for the AUTO parameter is to always automatically advance to the next tape drive if the unit is T1 or T2. The AUTO parameter is ignored if the unit is TC.
- If the next tape drive is either not available or not configured and AUTO or AUTO-YES was specified or is the default, the values are ignored and only the first tape drive is allocated. If the next tape drive is available, it is allocated when the first tape drive is allocated.
- If AUTO or AUTO-YES is specified or is the default, the next tape drive must support the same density as the first tape drive. The tape mounted on the next drive must be initialized to the same density as the first tape. If the density of the next tape is not the same as the first tape, an error message is issued.
- When configuring your System/36 environment, you should use caution when assigning unit IDs of T1 or T2 to a cartridge tape drive. Cartridge tape drives and reel-to-reel tape drives do not have the same density. If T1 is

a reel-to-reel tape drive and T2 is a cartridge tape drive, AUTO or AUTO-YES should never be specified or be the default. If this happens, an error message is always issued. The message indicates that the two tape drives are not the same density.

Using REWIND, LEAVE, and UNLOAD Tape Cartridge Processing

Each time a new tape cartridge is inserted into a cartridge tape drive or the latch door is opened and closed, the tape cartridge must be prepared before processing can begin. To prepare the tape, the system must make sure that it is positioned at the load point by winding the tape to the end of the cartridge and then back to the beginning. Consider the following factors before preparing your tapes:

- When you specify REWIND, the tape is rewound to the load point. If the tape cartridge is removed or the batch door is opened, the tape must be completely wound to the end and back to the load point.
- When you specify LEAVE, the tape is left in the position of the last access. If the tape cartridge is removed or the batch door is opened, the tape must be wound to the end and back to the load point. This method may take less time than if you specified REWIND.
- When you specify UNLOAD, the tape is wound to the end. If the tape cartridge is removed or the batch door is opened, the tape must be rewound to the load point. This method may take less time than if you specified REWIND or LEAVE.

Preparing Tapes

You may have to prepare your tapes. If you are unsure about the format of your tapes, use the Catalog (CATALOG) procedure to check the tapes.

To prepare a tape reel for use by the system, use the Tape Initializing (TAPEINIT) procedure, which is described in the *System/36 Environment Reference* book.

When you prepare the tape, you have the option to delete all information on the tape. With the

TAPEINIT procedure, the system allows you to specify identifying information to be placed on the tape. This identifying information includes:

- A 6-character (A – Z, 0 – 9, #, @, or \$) name called the volume ID. You specify the volume ID on system procedures to ensure that you are using the proper tape. It is a good practice to assign unique volume IDs to each tape reel.
- A 14-character (A – Z, 0 – 9, #, @, or \$) name called the owner ID. You can use this name to determine the owner of a tape. The owner ID is not checked by the system procedures, but is displayed by the CATALOG procedure.

To use the TAPEINIT procedure, you must be authorized to the tape device you are accessing.

Allocating the Tape Drive to a Job

Use the ALLOCATE OCL statement to dedicate the tape drive to a job.

For example, you have a procedure that saves three files (the tape has a volume ID of VOL001):

```
SAVE FILE1,,VOL001,T1
SAVE FILE2,,VOL001,T1
SAVE FILE3,,VOL001,T1
```

You do not usually keep control of the tape drive between the SAVE procedures. Another procedure on the system could use the tape drive after FILE1 is saved but before FILE2 is saved. This makes your SAVE FILE2 procedure wait until the other procedure ends.

You can use the ALLOCATE OCL statement to keep control of the tape drive throughout the three SAVE procedures:

```
// ALLOCATE UNIT-T1
SAVE FILE1,,VOL001,T1
SAVE FILE2,,VOL001,T1
SAVE FILE3,,VOL001,T1
```

To avoid allocating the tape drive longer than necessary, use the DEALLOC OCL statement to deallocate the tape drive. For example, if your procedure saves three files and then runs another type of job that does not use the tape drive, use the DEALLOC OCL statement to allow other jobs to use the tape drive:

```
// ALLOCATE UNIT-T1
SAVE FILE1,,VOL001,T1
SAVE FILE2,,VOL001,T1
SAVE FILE3,,VOL001,T1
// DEALLOC UNIT-T1
*
// LOAD PROG1
// RUN
```

In this example, if you do not specify the DEALLOC OCL statement, the job holds the tape drive until the job ends. While PROG1 is running, other jobs are needlessly prevented from using the tape drive.

Note: To fully benefit from the use of the ALLOCATE OCL statement and the LEAVE parameter with the cartridge tape drive, the same tape must be in the drive and the drive door must not have been opened. See the *System/36 Environment Reference* book for more information on the ALLOCATE and DEALLOC OCL statements.

Copying, Saving, Restoring, and Listing Information

The following procedures are supplied with the System/36 environment to allow you to copy, save, restore, and list information on tape.

Listed alphabetically, each procedure is described according to any differences between the System/36 environment and System/36. Except where noted, these procedures require IBM standard label tapes. The *System/36 Environment Reference* book has complete information about these procedures.

Before doing any of these procedures, make sure you are authorized to the tape devices. The AS/400 tape device may be secured.

Copying Information from Tape: Use the following procedures to copy information using tapes:

FROMLIBR

The FROMLIBR procedure copies one or more library members to a tape file. If the tape file created is a sector-mode file, it can only be copied back to the System/36 environment using the TOLIBR procedure. If the file is a record-mode file, it can be exchanged with a System/36 using the TOLIBR procedure, or with the AS/400 system using the

TOLIBR procedure or the Restore System/36 Library Member (RSTS36LIBM) command.

JOBSTR

The JOBSTR procedure copies one or more procedures or source members from a tape file to a library. The tape file must have been created in the System/36 environment with the FROMLIBR procedure.

TOLIBR

The TOLIBR procedure copies library members from a tape to a library. The tape file must have been created with the FROMLIBR procedure on either the System/36 or System/36 environment or with the Save System/36 Library Member (SAVS36LIBM) command.

TAPECOPY

The TAPECOPY procedure copies disk files to tape and tape files to disk in exchange format. This procedure allows you to use both standard label tapes and nonlabeled tapes. You can use standard label processing or nonlabeled tape processing while reading or writing tape files. Nonstandard label processing or bypass label processing can also be used while reading tape files.

Saving Information on Tape: Use the following procedures to save information using tapes:

SAVE

The SAVE procedure saves disk files on a tape. The files can only be restored back to an AS/400 system.

SAVELIBR

The SAVELIBR procedure saves an entire library to tape. The library can only be restored back to an AS/400 system.

SAVS36F

This system command saves a single disk file or multiple disk files to a tape that can be restored to a System/36. The file or files can also be restored to the AS/400 system using the Restore System/36 File (RSTS36F) command or the RESTORE procedure in the System/36 environment.

SAVS36LIBM

This system command saves source or procedure members to a tape that can be restored to a System/36. It can also be restored to the

AS/400 system using the Restore System/36 Library Member (RSTS36LIBM) command or the TOLIBR procedure in the System/36 environment (for record mode).

Restoring Information from Tape: Use the following procedures to restore information using tapes:

RESTLIBR

The Restore Library procedure restores an entire library from a tape to disk. The tape must have been created with the SAVELIBR procedure in the System/36 environment.

RESTORE

The RESTORE procedure restores disk files from tape to disk. The tape files must have been created with the SAVE procedure in the System/36 environment or on System/36.

RSTS36F

This system command restores a file or a group of files saved to tape on System/36 using the SAVE procedure, or restores a file saved on the AS/400 system using the Save System/36 File (SAVS36F) command.

RSTS36LIBM

This system command restores source or procedure members from tape that were saved using the System/36 FROMLIBR procedure, the System/36 SAVELIBR procedure, or the System/36 environment FROMLIBR procedure (record mode only).

Listing Information from Tape: Use the following procedures to list information from tapes:

CATALOG

The CATALOG procedure lists the names of files, saved libraries, and saved folders on tape.

LISTDATA

The LISTDATA procedure lists the contents of a saved file from tape. The tape file must have been created with the SAVE procedure in the System/36 environment or on System/36.

LISTFILE

The LISTFILE procedure lists the contents of an exchange file, saved file, saved library member or saved library from tape files. For

saved files, you must have created the tape file with the SAVE procedure in the System/36 environment or on System/36. For saved library members, you must have created the tape file with the FROMLIBR procedure in the System/36 environment. For saved libraries, you must have created the tape file with the SAVELIBR procedure in the System/36 environment.

Removing Information from Tape

To remove information from a tape, use the TAPEINIT procedure. See "Preparing Tapes" on page 10-5 for information on TAPEINIT.

Note: You cannot remove a specific file from tape.

Using Multiple Tape Drives

If your system has more than one tape drive, each tape drive can be allocated to a different job. For this allocation to work, each job must have the NOAUTO parameter specified in the procedures that use tape and AUTO-NO in the file statement of tape files.

Creating a Sequential Set of Files on Tape

Use the ALLOCATE OCL statement and the LEAVE parameter in a procedure to read or write a sequential set of files on tape.

Note: To fully benefit from the use of the ALLOCATE OCL statement and the LEAVE parameter with the cartridge tape drive, the same tape must be in the drive and the drive door must not have been opened. See the *System/36 Environment Reference* book for more information on the ALLOCATE OCL procedure.

When you save or restore files and libraries from a set of tapes, the system starts with the location you specify in the procedure. In the following example, the files and libraries are saved on tape, but after each save, the tape is rewound to its beginning. This means that when the next job step starts, the tape has to be positioned to the end of the last file before the next file can be saved.

```
* Procedure to save 2 files, and 2 libraries
SAVE FILE1,,VOL001,T1
SAVE FILE2,,VOL001,T1
SAVELIBR LIBR1,,VOL001,,T1
SAVELIBR LIBR2,,VOL001,,T1
```

Use the LEAVE parameter to instruct the system to leave the tape positioned after the end of the file just processed. The next job step using tape may take advantage of the LEAVE state, and the tape is not rewound. In the following example, the system begins with the first file and automatically continues with the other files and libraries.

```
* Procedure to create a sequential set of files
* on tape
// ALLOCATE UNIT-'T1/T2'
SAVE FILE1,,VOL001,T1,AUTO,,LEAVE
SAVE FILE2,,VOL001,T1,AUTO,,LEAVE
SAVE ALL,,GRPB,,VOL001,GRPB,T1,,AUTO,LEAVE
SAVELIBR LIBR1,,VOL001,,AUTO,T1,,LEAVE
SAVELIBR LIBR2,,VOL001,,AUTO,T1,,UNLOAD
// DEALLOC UNIT-'T1/T2'
```

The tape continues processing from the previous job step's ending location. If the tape has auto-advanced to the next tape unit and the LEAVE parameter is used, the system keeps track of the previous job step's ending unit and tape position. Processing continues from there. The last job step causes the system to unload the tape so it can be removed from the drive. On the cartridge tape drive, the tape is wound to the end of the cartridge.

When you are using the LEAVE parameter to read files from IBM standard label tapes, the tape's volume ID, the file name, and (if specified) the file's creation date are examined to ensure that the proper file is being read.

If you are using the LEAVE parameters to write to a standard labeled tape, the system will check to make sure the tape is positioned at the end of the tape before writing to the tape.

If a sequence number is specified, the tape is searched from the current position toward the specified sequence number, checking that the sequence number can be written to the tape. If a tape file exists at that sequence number, it is written over with the new tape file, and all files after the specified sequence numbers are no longer accessible.

Notes:

1. The system issues a diagnostic message if you try to write over files on the ¼-inch tape cartridge.
2. For the ¼-inch tape cartridge, if the sequence number request is less than the current position, the tape rewinds and searches from the beginning of the tape. If the sequence number requested is greater than the current position, the tape searches forward from the current position.

If the next job step is reading a standard labeled tape file in bypass label processing, the tape is searched from its current position to the specified sequence number.

If the next job step is reading or writing to a nonlabeled tape file and no sequence number is specified, the tape is read or written to without any checking. If a sequence number is specified, the tape is rewound and then the sequence number located.

If the next job step is reading a nonstandard labeled tape file, the tape is rewound and the first file's data is read.

The procedure control expressions VOLID and DATAT cause the tape to rewind and then search from the beginning of the tape.

You should also allocate the tape drive to your job when you are using the LEAVE parameter. This step ensures that your use of the tape drive is not interrupted by another job, thus preserving your tape's position. See "Allocating the Tape Drive to a Job" on page 10-5 for information about allocating the tape drive to a job. LEAVE information is maintained by the system from job step to job step, but is not passed from job to job.

Coexistence Considerations

This section describes the following coexistence considerations for restoring data files and source and procedure members:

- Restoring the AS/400 system to System/36
- Restoring System/36 to the AS/400 system
- Restoring the AS/400 system to System/36 within the System/36 environment

Restoring from an AS/400 System to System/36

There are two ways to move data files from the AS/400 system to System/36 using tape:

- Copy the data to an exchange tape on the AS/400 system using the TAPECOPY procedure (System/36 environment) or the Copy to Tape (CPYTOTAP) command. On System/36, use the TAPECOPY procedure to copy the data back to a disk file.
- Save the data files to tape on the AS/400 system using the Save System/36 File (SAVS36F) command. On System/36, use the RESTORE procedure to restore the data files.

There are two ways to move source and procedure members from an AS/400 system to System/36:

- Copy the members to a record-mode tape file using the System/36 environment FROMLIBR procedure or the CPYTOTAP command. On System/36, use the TOLIBR procedure to copy the members from the tape file back to a library.
- Save the source and procedure members to a tape on the AS/400 system using the Save System/36 Library Member (SAVS36LIBM) command. On System/36, use the TOLIBR procedure to copy the members from tape back to a library.

Restoring System/36 Files and Members to the AS/400 System

There are two ways to move data files from System/36 to the AS/400 system using tape:

- Copy the data to an exchange tape file on System/36 using the TAPECOPY procedure. On the AS/400 system use the TAPECOPY procedure (System/36 environment) or the Copy from Tape (CPYFRMTAP) command to copy the data back to a disk file.
- Save the data file to tape on the System/36 using the SAVE procedure. In the System/36 environment on the AS/400 system, use the Restore System/36 File (RSTS36F) command

or the System/36 environment RESTORE procedure to restore the data to a disk file.

There are two ways to use a tape to move source and procedure members from System/36 to the AS/400 system:

- Save the library that contains the source and procedure members to a tape on the System/36 using the SAVELIBR procedure. On the AS/400 system, use the Restore System/36 Library Member (RSTS36LIBM) command to copy the members back to a library.
- Copy the source and procedure members to a tape on System/36 using the FROMLIBR procedure. On the AS/400 system, use the RSTS36LIBM command or TOLIBR procedure to copy the members from the tape back to a library.

Restoring Files and Members from an AS/400 System to the System/36 Environment

There are two ways to move data files from one AS/400 system to another AS/400 system using tape:

- Copy the data to an exchange tape on the AS/400 system using the TAPECOPY procedure from the System/36 environment. On the other AS/400 system, use the TAPECOPY procedure to copy the data back to a disk file.
- Save the data file to tape on the AS/400 system using the SAVE procedure. On the other AS/400 system, use the RESTORE procedure to restore the data to a disk file.

There are two ways to use a tape to move source and procedure members from an AS/400 system to another AS/400 system:

- Save the library that contains the source and procedure members to a tape using the SAVELIBR procedure. On the other AS/400 system, use the RESTLIBR procedure to restore the library onto the system.
- Copy the source and procedure members to a tape using the FROMLIBR procedure. On the other AS/400 system, use the TOLIBR procedure to copy the members from the tape back to a library.

Moving from System/36 to the System/36 Environment

When migrating from System/36 to the System/36 environment, you can use tapes as a migration medium. You can copy or restore some files on tape to the System/36 environment, depending on

the file type. See "Tape Files" on page 10-2 for more information.

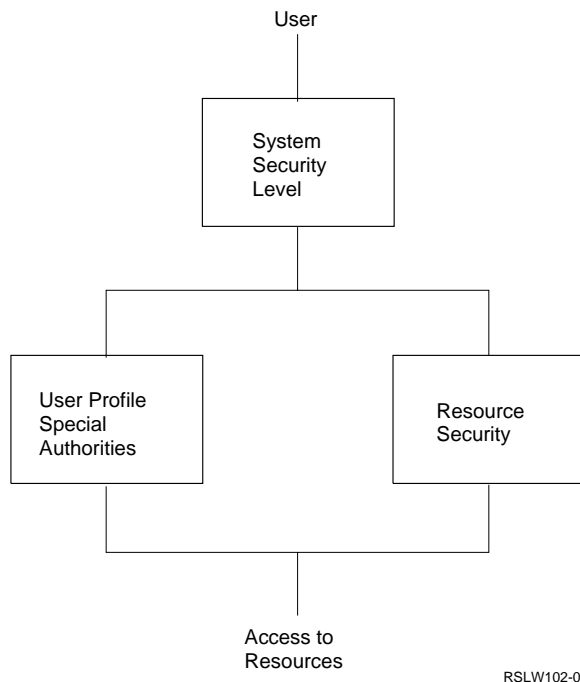
Chapter 11. Security

This chapter introduces security concepts and how to use security in the System/36 environment. The System/36 environment uses the AS/400 security functions. For detailed information about AS/400 system security, see the *Security – Reference* book.

AS/400-provided security helps limit who can use the system by:

- Setting the security level of the system to limit access to system resources
- Identifying users to the system by creating user profiles
- Identifying users to resources by giving the user authority to use those resources

The following figure summarizes system security:



System Security Levels

You can select the level of AS/400 system security for your system by changing the system value QSECURITY and starting the system. Security levels on the AS/400 system include:

- Level 10** Allows anyone to sign on and access any resource.
- Level 20** Password security is active. Resource security is not active.
- Level 30** Password and resource security are both active.
- Level 40** Password and resource security are both active. Programs that try to access objects through unsupported interfaces will fail.
- Level 50** Password and resource security are both active. Programs that try to access objects through unsupported interfaces will fail. Security and integrity of the QTEMP library and user domain (*USRxxx) objects are enforced. Programs that try to pass unsupported parameter values to supported interfaces will fail.

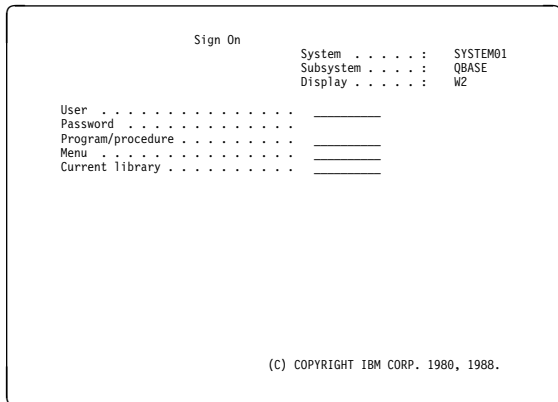
For complete information about security levels, see the *Security – Reference* book.

Sign-On Security

Sign-on security is active when your system security level is 20, 30, 40, or 50. A user must type a password and a user name on the Sign On display to sign on the system (a user profile by that name must exist).

Note: Your system is shipped with physical security active, and user passwords are not required. Anyone can sign on the system by typing any user name from 1 to 10 characters. If a user profile by that name does not exist, the system creates one and the user is given authority for a minimal-security system.

A Sign On display for security levels 20, 30, 40, and 50 follows:



Notes:

1. Passwords do not appear when you type them on this display.
2. The *Password* field does not appear on a level 10 system.

User Profiles and Special User Authority

User profiles identify users on the system. They tell the system who can sign on and what functions can be performed after signing on. Create user profiles with the Create User Profile (CRTUSRPRF) CL command. See the *CL Reference* book for more information.

Throughout this chapter the term *user* also means *remote users* starting communications jobs. Therefore, user profiles must be created for remote users to sign on and allow access to system resources.

User profiles tailor the way a user operates on the system by controlling or assigning:

- User class
- Special authority

- Initial program security
- Menu security
- Limited capability
- Group profile

User Class

The classification you specify in a user profile determines what system control operations and menu options the user can use. The user classes are:

- *SECOFR: Security Officer
- *SECADM: Security Administrator
- *PGMR: Programmer
- *SYSOPR: System Operator
- *USER: General User

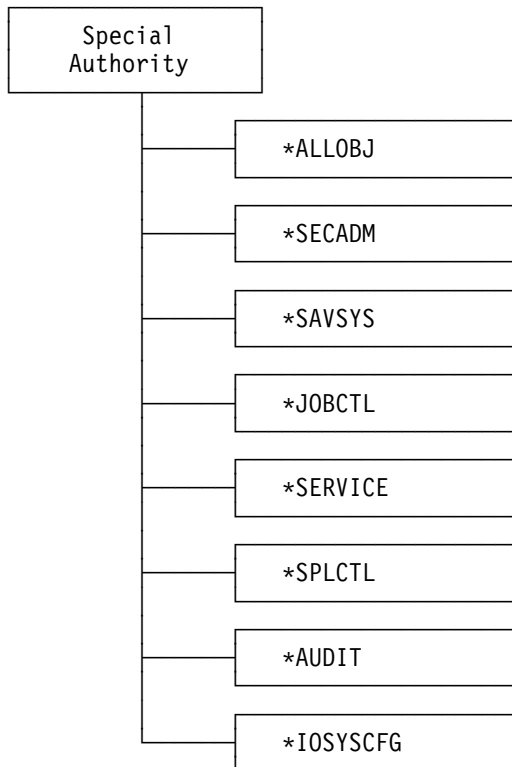
Each user class has a set of special authorities associated with it depending on the system security level you use. See the *Security – Reference* book for information about the special authorities associated with each user class.

See “System/36 User Identification File” on page 11-8 for information about how these user classes compare to System/36 user profile security classifications.

Special Authority

Special authority allows a user to perform system control operations such as saving the system, controlling other users' jobs, using the system service tools, controlling spooled files, and creating user profiles. You assign special authority in the user profile by specifying the class of user (that has the appropriate special authorities associated with it) or by tailoring the special authorities for a specific user environment. Use the Create User Profile (CRTUSRPRF) or the Change User Profile (CHGUSRPRF) CL command to assign special authority.

The following figure summarizes special authorities a user can have:



Special authorities are defined as follows:

- **All Object** (*ALLOBJ) special authority allows a user access to all system resources even if the user has no authority to the resource.

Throughout this chapter, the term *security officer* also means *or someone with all-object special authority*.

Note: Before you give a user all-object special authority, you should consider whether the user really needs this authority. Giving a user this authority may be a security risk because it allows the user to use or delete any user object.

- **Security Administrator** (*SECADM) special authority allows a user to:
 - Add users to the system distribution directory (this includes the right to create and change user profiles for OfficeVision for OS/400 users) using the Work with Directories (WRKDIR) CL command.

You must have security administrator authority to use the CRTUSRPRF and CHGUSRPRF CL commands. This

authority is not required for the Change Profile (CHGPRF) CL command.

- Change or display authority for documents or folders.
- Add and remove access codes on the system.
- Give and remove a user's access code authority.
- Give and remove permission for users to work on another user's behalf.
- Delete documents and folders.
- Delete document lists.
- Change distribution lists created by other users.

Only a user with *SECADM and *ALLOBJ special authority can give another user *SECADM special authority.

- **Save System** (*SAVSYS) special authority allows a user to:
 - Do save and restore operations for all resources on the system.
 - Free storage for all objects on the system.
 - **Job Control** (*JOBCTL) special authority allows a user to:
 - Change, delete, hold, and release all files on output queues if output or job queue is operator controlled (OPRCTL(*YES)).
 - Hold, release, and clear job and output queues if output or job queue is operator controlled (OPRCTL(*YES)).
 - Hold, release, change, and cancel other user's jobs.
 - Start writers to output queues if output or job queue is operator controlled (OPRCTL(*YES)).
 - Change the running attributes of a job, such as the printer for a job.
 - End subsystems.
 - Initial program load (IPL) the system.
 - **Service** (*SERVICE) special authority allows a user to perform the alter service function. The display and dump functions can be performed without this authority.
- Note:** Before you give a user service special authority, you should consider if the user really needs this authority. Giving a user this

authority may be a security risk because it allows the user to access sensitive data.

- **Spool Control** (*SPLCTL) special authority allows a user to control spool functions, such as cancel, delete, display, hold, and release other user's spooled files. This authority allows access to job and output queues that are specified as OPRCTL(*NO).
- **Audit** (*AUDIT) special authority allows a user to perform auditing functions. These include turning auditing on or off for the system and controlling the level of auditing on an object or a user.
- **Input/output (I/O) system configuration** (*IOSYSCFG) special authority allows a user to change system I/O configurations.

Initial Program Security

Initial program security allows you to specify a program/procedure to run when the user signs on the system. For example, if you have a user whose only responsibility is to run an audit program once a month, you can prevent this user from running any other program by specifying the program that is required when the user signs on, and then having the system sign the user off after the program completes.

Menu Security

When all the options listed in a menu match what the user should be allowed to do, use menu security. **Menu security** is a function of the operating system that controls which system resources are available to users. Menu security restricts a user to a single menu or a sequence of menus defined in the user profile. When the user signs on, the specified menu appears, and all other menus remain unavailable. The default value in the user profile for the initial menu is the system Main Menu.

For information about menu security considerations, see the *Security – Reference* book.

Limited Capability

You can impose special limits on a user's capability within those specified in the user profile. Limiting a user's capability is similar to specifying a mandatory menu, program/procedure, or current library on System/36.

Limiting a user's capability is done in the user profile with the Limited Capability (LMTCPB) value. For more information about the limited capability values, see the *Security – Reference* book.

Group Profile

For information about user profiles and how to use the group profile to simplify security, see the *Security – Reference* book.

Resource Security

Resource security allows you to control how a user is able to use a resource by specifying authority for:

- A user to a *resource*
- The *public* (all users not specifically secured) to a resource
- A user to the *library* containing the resource
- A list of users on an *authorization list* to a resource
- A user to an *authority holder* for a file

Notes:

1. When resource security is not active (level 10 or 20), anyone who signs on to the system can use any file, library, or device on the system.
2. When resource security is active, anyone with *ALLOBJ special authority in their user profile can use any file, library, or device on the system.
3. A program can be written to ADOPT the authority of another user profile while running and thus have more authority than the user who actually is running the program.

For more information on these items, see the *Security – Reference* book.

Authority for a User to a Resource

When you specify authority for a user to a resource, you control the user's object authority and data authority. *Object authority* is the capability to perform specific operations on an object, such as move or rename the object, and to control the object's existence. *Data authority* is the capability to perform operations on the data contained in the object. Use the Grant Object Authority (GRTOBJAUT) or Edit Object Authority (EDTOBJAUT) CL commands.

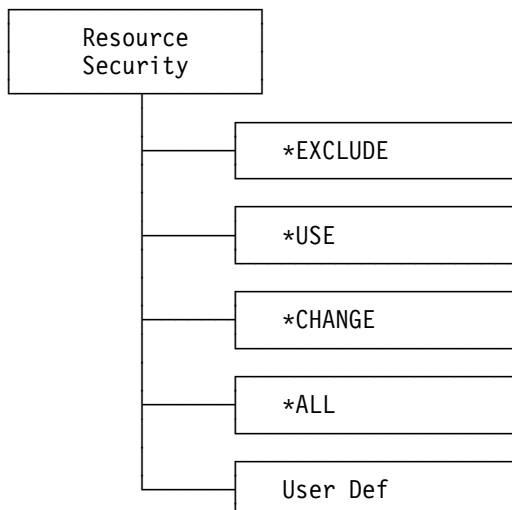
You can define authority by combining one or more object authorities with one or more data authorities (user-defined), or by specifying one of the following system-defined authorities that is a preset combination of object authorities and data authorities:

- *ALL
- *CHANGE
- *USE
- *EXCLUDE

Note: Use the system-defined authorities whenever possible.

You can use the Display Object Authority (DSPOBJAUT) or EDTOBJAUT CL commands to see what authorities have been assigned to a particular resource. On these displays, user-defined authority is shown as USER DEF.

The following figure summarizes authorities that can be specified for a user:



Object Authority: The object authorities listed here give a user the following capabilities:

Object Operational (*OBJOPR)

Look at the description of an object and use the object as determined by the data authorities that the user has to the object. This authority is normally used in combination with one or more data authorities.

Object Management (*OBJMGT)

Specify the security for the object, move or rename the object, and add members to database files.

Object Existence (*OBJEXIST)

Control the object's existence and ownership. This authority is necessary for users who want to delete the object, free storage of the object, perform save and restore operations for the object, or transfer ownership of an object. (If a user has save system (*SAVSYS) special authority, he does not need object existence authority to perform save and restore operations.)

Object Alter (*OBJALTER)

Alter the attributes of an object. A user with object alter authority can add and remove triggers and referential constraints, and change the attributes of database files and SQL packages.

Object Reference (*OBJREF)

Reference an object from another object such that operations on the object can be restricted by the other object. A user with object reference authority on a physical file can add a referential constraint in which the physical file is the parent. This authority is used only for database files.

Authorization List Management (*AUTLMGT)

Add and change users and their authorities on an authorization list. A user with authorization list management authority can add, change, or remove authority only if the user has the authorities being added, changed, or removed.

Data Authority: The data authorities listed below give a user the following capabilities:

Read (*READ)

Display the contents of an object.

Add (*ADD)

Add entries to an object. (For example, adding job entries to a job queue or adding records to a file.)

Update (*UPD)

Change the entries in an object.

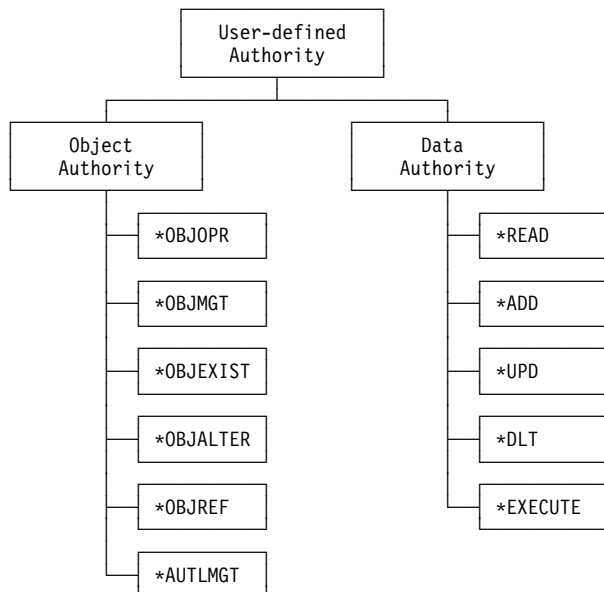
Delete (*DLT)

Remove entries from an object. (For example, removing messages from a message queue or records from a file.)

Execute (*EXECUTE)

Run a program or locate an object in a library or directory.

The following figure lists authorities that can be user-defined.



System-Defined Authority: When you assign a system-defined authority, you select a subset of object authorities and data authorities for an object.

The following table shows the subsets of object authorities associated with each of the system-defined authorities:

Authority	OPR	MGT	EXIST	ALTER	REF
*ALL	X	X	X	X	X
*CHANGE	X				
*USE	X				
*EXCLUDE	No Authority				

The following table shows the subsets of data authorities associated with each of the system-defined authorities:

Authority	READ	ADD	UPD	DLT	EXECUTE
*ALL	X	X	X	X	X
*CHANGE	X	X	X	X	X
*USE	X				X
*EXCLUDE	No Authority				

The system-defined authorities listed here give a user the following capabilities:

All (*ALL)

The user can control the object's existence, specify the security for the object, change the object, and perform basic operations on the object such as run a program or display the object's description and contents. All authority provides all object and data authorities.

Change (*CHANGE)

The user can add, change, and delete entries in an object, or read the contents of an entry in the object. Change authority provides object operational authority and all the data authorities.

Use (*USE)

Perform basic operations on the object, such as run a program or display the object's description or contents. The user is prevented from changing the object. Use authority provides object operational authority and both read and execute authority.

Exclude (*EXCLUDE)

Restrict access to an object. If this authority is specified, you have neither private nor public authority.

Authorization list management authority can be specified for an authorization list and other authorities. The only exception is exclude authority. If *EXCLUDE authority is specified, you have neither private nor public authority.

See "Moving from System/36 to the System/36 Environment" on page 11-8 for a description of how these system-defined authorities map to System/36 access levels.

For more information on resource security, see the *Security – Reference* book.

Public Authority

A user name called *PUBLIC has been defined by the system. The authorities you grant to this user become the default authority (authorities used for any user who does not have any other type of security granted for the resource).

Library-Level Security

Using library security, you can place all sensitive objects in a library and then secure the library to limit use of those objects. For example, if a payroll application is placed in a single library, you can control security to this application by controlling authority to the library. You can limit which users are able to use, add, or delete the library objects. To set library-level security, use the GRTOBJAUT or EDTOBJAUT CL commands and specify the library as the object to be secured.

Using library-level security in the System/36 environment is similar to securing a library on System/36. Authority to the library extends to include the objects within the library, unless specifically changed. When you use any of the CL create commands, use the default *LIBCRTAUT specification on the AUT keyword. The newly created objects assume the security level of the library.

Note: If the CRTAUT value of a library changes, the new value does not affect any existing objects. Objects created after the CRTAUT value changes assume the new authority value.

Just as on the System/36, objects within the library can have different authorities than the library itself. By setting the system value QCRTAUT to *ALL, you can keep each object in a library at level 30 authority. This way, the System/36 environment more closely emulates the security procedure on System/36. You can increase the security level for objects that contain sensitive information and grant authority to those objects using authorization lists.

Authorization Lists

An **authorization list** is a list of users and their authorities. Authorization lists are created with the Create Authority List (CRTAUTL) CL command and are maintained using the Add Authority List Entry (ADDAUTLE), Remove Authority List Entry

(RMVAUTLE), Change Authority List Entry (CHGAUTLE), and Edit Authority List (EDTAUTL) CL commands. You can then use an authorization list to authorize a list of users to an object in a single operation. When you secure an object or a set of objects using the GRTOBJAUT or EDTOBJAUT CL commands, you can specify an authorization list name instead of specifying authority for each user. Only one authorization list can be specified for an object. When an authorization list has been used to secure an object, individual authorities given to the object for a specific user take precedence over the authorization list. Public authority given to the object also takes precedence over the public authority in the authorization list. If you want the public authority for an object to come from the authorization list, specify a value of *AUTL for the public authority of the object.

Authority Holders

Authority holders allow a user to secure a file by name before the file is actually created, or to hold the authorizations for a file as it is deleted and re-created. When an authority holder is created for an existing file, all authorities for the file are transferred to the authority holder. Once the authority holder is created, all authority given to the object is actually given to the authority holder. This approach allows the system to keep authorities for System/36 environment applications that often delete files and then re-create them. If the file is deleted, the authority information is kept with the authority holder to be used again when the file is re-created or restored. Authority holders are created using the Create Authority Holder (CRTAUTHLR) CL command.

Note: Authority holders are valid only for program-described database files. Files created by System/36 environment procedures and commands are normally program-described files. See Chapter 7, "Files" for more information on program-described files.

For more information about authority holders, see the *Security – Reference* book.

Moving from System/36 to the System/36 Environment

During migration from your System/36 to the System/36 environment, security levels and classifications are migrated following the procedures described in this section.

System/36 User Identification File

Each user profile defined in the System/36 user identification file is replaced on the AS/400 system with an AS/400 user profile. The following table shows how the System/36 user profile security classifications are mapped to AS/400 user profile user classes:

System/36 Security Classification	AS/400 System User Class
M — Master Security Officer	Security Officer — *SECOFR
S — Security Officer	Security Administrator — *SECADM
O — System Operator	System Operator — *SYSOPR
C — Subconsole Operator	System Operator — *SYSOPR
D — Display Station User	User — *USER

If you are familiar with System/36 security classifications, use this mapping as a guideline when you are assigning user class to new users.

System/36 Resource Security File

On System/36, the resource security information is contained in the resource security file and applies to files, libraries, folders, subdirectories, folder members, and special resource types such as alternate index files, the system library, and groups of files, libraries, and folders. On the AS/400 system, authority to objects are associated with the object itself, not in a separate file. Also, authority to objects is checked by the System/36 environment whenever an object is referred to, not just when it is used. For example, the FILE OCL statement and the ?F'S,NAME'? procedure substitution expression both require a minimum authority of *USE to the file to be run. On System/36, the FILE OCL statement and ?F'S,NAME' substitution expression could be run without any authority to the file. The following

sections show how information from the resource security file is moved to or done on an AS/400 system.

Access Levels: When the System/36 resource security file is migrated to the System/36 environment, the System/36 access levels are mapped to system-defined authorities on the AS/400 system, as shown in the following figure:

Figure 11-1. Access Levels and AS/400 System-Defined Authorities

System/36 Access Level	AS/400 System-Defined Authority
Owner	*ALL
Change	*CHANGE + *OBJEXIST
Update	*CHANGE
Read	*USE
Run	*USE
None	*EXCLUDE

Resource Ownership: On System/36, a resource can have multiple owners or no owner. On the AS/400 system, a resource must have one and only one owner. When resources are migrated from System/36 to the AS/400 system, ownership is established using the following rules:

- Resources with no owner on System/36 are owned by a system-supplied default owner user profile called QDFTOWN.
- Resources with more than one owner are owned by the first owner found in the resource security file. All other System/36 owners are given all (*ALL) authority to the resource but are not owners.
- Library members, which could not have an owner on System/36, are given the same owner as the library that contained them.

Libraries: When a secured library is migrated from System/36 to the AS/400 system, an authorization list is created with the same name as the library. This authorization list is used to secure the library and all the objects migrated into the library.

Each user record in the resource security file for the library is added to the authorization list with a comparable access level. The public authority for the authorization list comes from the default access level specified for the library in the

resource security file. No public or private authority is given to the library itself.

If the library was not secured on System/36, the library and all the objects in the library are placed on the AS/400 system with the authority specified by the system value of QCRTAUT.

Objects migrated from System/36 libraries to the AS/400 system consist of:

Procedures

A procedure member in a System/36 library is migrated to a member of a file in a library on the AS/400 system. The library name on the AS/400 system is the same as the library name on the System/36. The file name on the AS/400 system is always QS36PRC. The member name in file QS36PRC is the same as the procedure name on the System/36.

If the library was not secured on the System/36, QS36PRC is created with the create authority of the library (CRTAUT).

Source members

A source member in a System/36 library is migrated to a member of a file in a library on the AS/400 system. The library name on the AS/400 system is the same as the library name on the System/36. The file name on the AS/400 system is always QS36SRC. The member name in file QS36SRC is the same as the procedure name on the System/36.

If the library was not secured on the System/36, QS36SRC is created with the create authority of the library (CRTAUT).

Programs

Each program is secured by the create authority for the AS/400 library of the same name as the System/36 library where the program object module is found. This may or may not be the same library as the program source.

If the library is not secured, the program is migrated with the authority specified by the system value of QCRTAUT.

Libraries after Migration: You can add users to or remove users from an authorization list using the ADDAUTLE, RMVAUTLE, CHGAUTLE, and EDTAUTL commands. These changes to the authorization list apply to all objects secured by the authorization list.

If you create a new library and wish to secure it, you can create an authorization list using the CRTAUTL CL command, and then add a private authority to the list for each user you wish to use the library. Use this authorization list to secure the library and all the objects in the library.

Notes:

1. Objects created in a library acquire the authority specified in the creation command. If an authority is not specified in a command, the object gets the create authority of the library by default. Exceptions to this are save files (object type *SAVF), which are created with *EXCLUDE authority.

Objects moved or restored into a library retain their authorities.

When adding new objects to the library (programs, display files), you can secure the objects by the authorization list. New objects placed in the library are not automatically secured by the authorization list.

2. A library may also be included as a part of a GROUP or referred to as a parent or child in the System/36 resource security file. See "Group Names" on page 11-10 for a description of groups and parent/child, and for more information on how libraries are secured in these cases.
3. Migrating UPDATE authority to *CHANGE takes away the ability to add, delete, or change procedures and source members and to delete programs and display files. Adding and changing members in QS36PRC or QS36SRC requires object management (*OBJMGT) authority. Deleting members in QS36PRC or QS36SRC and deleting programs and display files requires object existence (*OBJEXIST) authority.
4. Migrating CHANGE to *CHANGE plus *OBJEXIST authority takes away the ability to add or change procedure and source members. Adding and changing members in QS36PRC or QS36SRC requires *OBJMGT authority.
5. MRT programs receive no special treatment during migration, but do get special treatment when they are run. This special treatment is discussed in "Multiple Requester Terminal (MRT) Programs" on page 11-12.

6. REPLACE (*YES) on compiles preserves authority to the program.

Files: When an entry exists in the System/36 resource security file for a file that does not exist, an authority holder is created on the AS/400 system with the same name as the file. The public authority for the file is assigned to the authority holder using the default access level from the entry in the resource security file. To determine how this authority is mapped to the system-defined authority, see Figure 11-1 on page 11-8. The user authorities for the file are set from each user record specified in the resource security file by applying the same mapping as shown in Figure 11-1 on page 11-8.

Authority holders are created for files that do not exist on System/36 at migration. If the file exists, it must be migrated. If it is not migrated, an authority holder is not created.

Files after Migration: To change the authority for the file, use the GRTOBJAUT, RVKOBJAUT, or EDTOBJAUT CL commands.

To secure a new file, create an authority holder using the CRTAUTHLR command, and specify the default authority needed as the public authority. For each user, grant a private authority using the GRTOBJAUT or EDTOBJAUT CL commands. The authority holder may be created before or after the actual file is created.

Notes:

1. Files can be secured without the use of an authority holder, just like other objects. However, if the file is deleted and recreated, all the authorities previously granted to the file are lost unless an authority holder is defined.
2. DISP-OLD processing of files requires *ALL authority to the file. On the System/36 it required CHANGE authority. When DISP-OLD processing occurs for existing files, the required authority to the file's library may require *CHANGE if it is necessary to change the attributes of the existing file.
3. When a new file is created, and the file already exists with a different date (date-differentiated files), *ALL authority is required to the existing file.

4. System/36 environment files are collected in a files library and therefore additional authorization is required to the files library. *CHANGE to the library is required when creating new files and *USE to the library is required when using existing files.

Folders, Subdirectories, and Folder

Members: The security information for word processing is not migrated with the associated user ID. For more information about folder security, see the *Using OfficeVision/400 Word Processing* book.

Group Names: During migration of the System/36 resource security file, if a GROUP entry is encountered, an authorization list is created with the same name as the group name. Creation of the authorization list may add new functions. All files and libraries containing this group name as part of their name are secured by the authorization list. Each user record for the group in the resource security file is added to the authorization list with the appropriate access level mapped to a system-defined, specific authority. When a library is part of the group, the library and all the objects contained within the library are secured by the GROUP authorization list, not by the library's authorization list.

Group Names after Migration: If you want to add new users to the GROUP, use the ADDAUTLE CL command to add them to the authorization list.

If you want to add your own new GROUP entry, create an authorization list (CRTAUTL command) with the group name as its name and add authority (ADDAUTLE CL command) for each user you want to use the group. Use this authorization list to secure all the objects in the group by specifically granting the authorization list authority for the objects.

When new files are created in the System/36 environment with the GROUP name as part of their name, they are automatically secured with this authorization list. Other objects such as libraries and programs are not.

Note: Automatically adding files to group authorization lists only occurs when files are created using System/36 environment procedures (for example, the BLDFILE or COPYDATA proce-

dures), and does not occur when creating files using AS/400 CL commands (CRTPF, CPYF, and so on).

Parent-Child Concept: The System/36 **parent-child concept** is a way of defining a relationship between a particular file or library and another file or library contained in the same resource security file. The parent contains the resource security information for all of its children. The parent-child concept reduces the number of records needed in the resource security file, when several resources all have the same list of authorized users. Only the parent resource needs user records, since they also apply to each of the children resources.

During migration, an authorization list is created with the name of the parent. User records for the parent result in an authorization list entry being added to the authorization list for the user. The parent resource, and each file or library listed as a child, is secured by the authorization list.

If the parent is a library, the authorization list created for the library is used to secure each child. If the child is a library, the authorization list of the parent is used to secure the library and each of the objects in the library. Authorization lists created for parent-child relationships will be used for files and libraries instead of any GROUP authorization lists that may have been defined.

Parent-Child after Migration: To continue using the parent-child concept to secure resources in the System/36 environment, create an authorization list (CRTAUTL command) and add authority for each user to use the parent (ADDAUTLE command). When creating new *children* objects, use this authorization list to secure them. If the child is a library, each of the objects within the library must have a specific grant to the authorization list.

Note: Library objects that are deleted and recreated without specifying an authority are created with the create authority (CRTAUT) of the library they are created into. Libraries that are deleted and recreated without a specified authority default to the create authority of library QSYS.

System Library: Objects in the system library that have not changed since they were shipped with System/36 are not migrated. They are replaced with the support shipped with the System/36 environment.

Objects that have been changed since they were shipped on System/36 are placed in a library named #LIBRARY on the AS/400 system. They are migrated using the same rules as objects in other libraries. #LIBRARY is created during migration if it did not exist before migration. #LIBRARY is owned by the QSYS user profile and is given a default public authority that is the create authority of library QSYS.

Additional Security Considerations

Beyond the general security concepts, the following circumstances require special consideration.

Entering the System/36 Environment:

When an AS/400 job becomes a System/36 environment job by entering the Start System/36 (STRS36) CL command or Start System/36 Procedure (STRS36PRC) CL command, or because the special environment value (SPCENV(*S36)) is specified in the user profile, the user must have *USE authority to the following libraries:

- The #LIBRARY library
- The QSSP library
- The System/36 environment files library
- The current (*CURLIB) library

If no current library exists when the System/36 environment is entered, #LIBRARY is made the current library.

Command Security: On the AS/400 system, CL commands themselves can be secured, just as resources are. For example, the CL command CRTPF could be secured so that only the system security officer is allowed to create files. This can be done by using the GRTOBJAUT CL command to change the public authority of the file to *EXCLUDE. When this type of security is used on your system, System/36 environment commands and procedures are also affected. For more information about securing commands, see the *Security – Reference* book.

For a summary of command authorities required by System/36 environment commands, procedures, and operation control language (OCL) statements, see Appendix E, "Security Considerations for the System/36 Environment."

Multiple Requester Terminal (MRT)

Programs: MRT programs are migrated from System/36 like other programs, and can be secured like any program. However, when you run MRT programs in the System/36 environment on the AS/400 system, you must consider some additional factors.

Selecting the proper user profile under which to run the MRT program becomes very important. On the AS/400 system every job, including MRT programs, must designate one *user profile* under which the job is to run. This user profile must have the proper authority to all objects referred to by the job, or the job will get an authorization error preventing the MRT from running. Select the user profile under which the MRT programs are to run through a configuration setting that can be changed using the Change System/36 (CHGS36) CL command. This same command controls a configuration setting that determines how files are handled. An example configuration display follows:

```

Change S/36 MRT Security and Performance
S/36 environment . . . . . : #LIBRARY
Type choices, press Enter.
User profile used by MRT program . . . 1      1=First user of MRT
                                           2=Owner of MRT
Check authority of user to files used
by MRT program . . . . . 1      1=All users
                                           2=First user
Seconds to wait before ending a
non-NEP MRT . . . . . 60      0-32767
                                           Bottom
F3=Exit  F5=Refresh  F10=Set to default values  F12=Cancel

```

Only users with a user class of *SECOFR in the user profile are allowed to change MRT security using the CHGS36 command.

Using the CHGS36 command, a user can control:

- Whether all MRT programs should run using the *owner* of the program's user profile, or

whether they should run using the user profile of the person who started the MRT program (the default).

- Whether all users of the MRT program should have their authorization to files checked (the default), or whether authorization is checked for the first user only.

Designating User Profile for MRT Programs:

Selecting the correct user profile is important because authorization to all resources, except files, is checked against this user. If you decide to use the first user's profile (the default), a different user profile could be used each time the MRT program is run. Make sure none of the users have too little or too much authority. If you change MRT security, and use the MRT program owner's user profile, make sure the program owner does not have too little or too much authority.

Defining Files Authorization Checking:

Authorization to files is handled differently than other resources. By default, each user who attempts to use an MRT program will be checked to ensure that they have authority to all of the files currently in use by the MRT program. If having authority to the MRT program itself is sufficient security, you can indicate in your configuration for the System/36 environment to only check authority to files as they are opened by the MRT program, and not to check authority for each new user of the MRT program. Doing so will improve performance of your MRT program.

Note: When a MRT program opens a file, the user profile under which the MRT program is running must always have authorization to that file, or the open request will fail.

Saving and Restoring Authorities

Save authorities using the Save System (SAVSYS) command. Restore authorities using the Restore Authority (RSTAUT) command.

See the *CL Reference* book and the *Security – Reference* book for information about saving and restoring authorities.

Chapter 12. Designing Records

This chapter describes how to design records in the System/36 environment. A **record** is a collection of fields treated as a unit. The way you design records depends on the type and format of the records needed, the type of fields and files they are used with, and how they are used.

Identifying Required Fields

Records must contain all the required input and output fields.

Depending on the application you are using, any of the following fields can be required:

- Keys or relative record numbers
- A status byte to allow for deletions
- A record code to identify the record
- Fields for exception reports (for example, credit limit or minimum stock)
- Fields to help in analysis (for example, number of accesses to this record, date of last update, number of returns, or pending order)
- Fields reserved for expansion

Naming Fields

Assign meaningful and standard field names to make your program and the related application clear and easy to use.

Establish a list of standard abbreviations to use for field names because most high-level languages (such as RPG II or COBOL) limit the number of characters you can use for field names. For example, you can abbreviate the word MASTER as MST, MAST, or MSTR. Choose one abbreviation and use it consistently.

When you use files containing fields that have the same name, assign a prefix to the field name to avoid confusion. For example, use MST as the prefix to all field names in your master file, or TRAN for transaction file, PAY for payroll file, and INV for inventory file.

Standardize the field description entries for your files. Write a field description for each of your

files and store these definitions in separate source members in your libraries. Use \$AUTO / COPY, programming development manager (PDM) or source entry utility (SEU) to copy these field definitions into your program.

This approach has several advantages:

- Data definitions are standardized. Your programmers use identical file definitions with the same field names and descriptions.
- You save time. The field description for each file needs to be written, entered, and debugged only once. All programmers can use these descriptions.
- Maintenance is easier. Programmers are familiar with field definitions in other programs. You change only one source member to change a field definition. The system copies this member into each program that used the old source member.
- You can use the source member for each file to describe the contents of the file. When you build the field description for each file, include comments that describe what each field is used for, what the valid range of values is, and other additional information.

The high-level language you use to write your program determines the maximum length you can specify for your field names. See the correct language book for information about the maximum length allowed for field names in the programming language you use.

Using Numeric Fields

System/36 is a zoned decimal system. The AS/400 system is a packed decimal system. If you have numeric fields, determine whether the field is to be in zoned decimal format, packed decimal format, or binary format. Packed and binary formats reduce the amount of space needed to store numeric information.

Allow the maximum length for amount fields to prevent a high-order position from being lost.

Zoned Decimal Format

A **zoned decimal format** is a format for representing numbers in which the digit is contained in bits 4 through 7 and the sign is contained in bits 0 through 3 of the farthest right byte; bits 0 through 3 of all other bytes contain 1s (hex F). For example, in zoned decimal format, the decimal value of +123 is represented as 1111 0001 1111 0010 1111 0011. For data in zoned decimal format, each byte of storage represents a single character. Each byte of storage is divided into a 4-bit zone portion and a 4-bit digit portion. Figure 12-1 shows zoned decimal format.

In Figure 12-1, the zone portion of the farthest right byte represents a positive sign by hexadecimal F (1111) or C (1100), and a negative sign by hexadecimal D (1101).

Figure 12-2 shows the number 7462 stored in zoned decimal format.

You can specify whether a field in your program is signed or unsigned. A signed field stores the sign with the data. An unsigned field does not store the sign with the data. Define the field as signed to process fields with a negative value. If you do not specify that a field is signed, the system assumes that the data in the field is positive. This can cause unexpected results. For example, if you code your program to do an operation only when the sign of a field is negative, the operation cannot be done if the field is unsigned.

When the zone portion of the farthest right byte is changed to represent the sign, the farthest right byte becomes a character that cannot be printed or displayed.

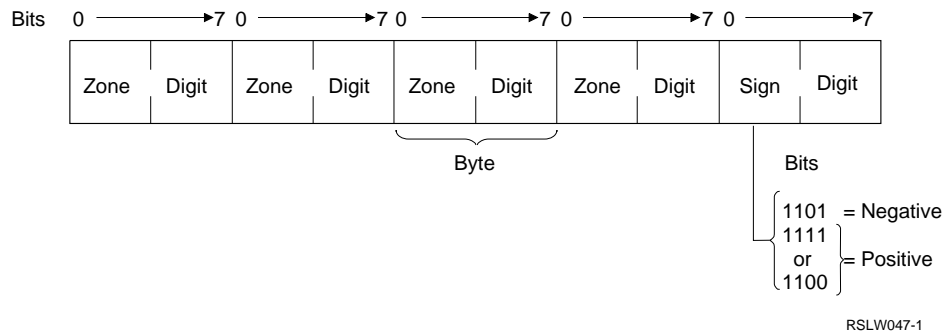


Figure 12-1. Format of Data in Zoned Decimal Format

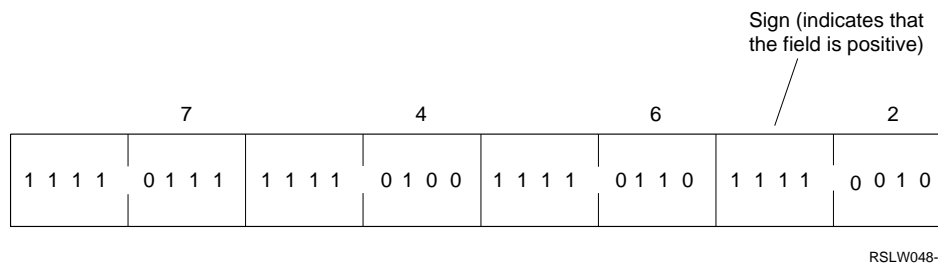


Figure 12-2. Example of the Number 7462 Stored in Zoned Decimal Format

The following table shows how changing the sign affects how the farthest right byte is printed or displayed.

Sign Portion of the Farthest Right Byte	Value of the Farthest Right Byte	How the Farthest Right Byte Is Printed or Displayed
Hex F (positive)	0 through 9	0 through 9
Hex C (positive)	0	Unprintable ¹
Hex C (positive)	1 through 9	A through I
Hex D (negative)	0	Unprintable ¹
Hex D (negative)	1 through 9	J through R

¹ These characters can be printed if your printer has a printer belt that includes additional special characters.

For example, if you place a value of +1234 in a signed zoned decimal field, it is stored as:

F1 F2 F3 F4

If you print this field, it appears as:

1 2 3 4

If you place a value of -1234 in a signed zoned decimal field, it is stored as:

F1 F2 F3 D4

If you print this field, it appears as follows because hex D4 is the EBCDIC value of the letter M:

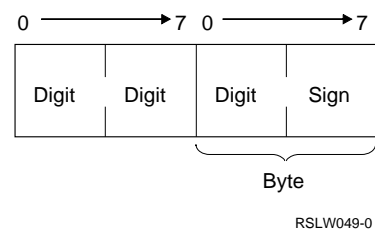
1 2 3 M

Your program must convert the data in the field to properly print the value.

Packed Decimal Format

A **packed decimal format** is a representation of a decimal value in which each byte within a field represents two numeric digits except the farthest right byte, which contains one digit in bits 0 through 3 and the sign in bits 4 through 7. For all other bytes, bits 0 through 3 represent one digit; bits 4 through 7 represent one digit. For example, the decimal value +123 is represented as 0001 0010 0011 1111. Each byte of storage represents 2 decimal digits for data in packed decimal format. You can store almost twice as much data in the same amount of storage by using packed decimal format instead of zoned decimal format.

In packed decimal format, each byte of storage except the farthest right byte is divided into two 4-bit digit portions. The following figure shows packed decimal format:



The farthest right 4 bits of the farthest right byte are reserved for the sign. If you are storing a field with an odd number of digits, the field completely fills the bytes of storage reserved for it, as shown in the figure above. If you are storing a field with an even number of digits, the field is right-adjusted within the bytes of storage reserved for it. The farthest left digit portion of the farthest left byte contains only zeros.

Figure 12-3 shows the number 7462 stored in packed decimal format.

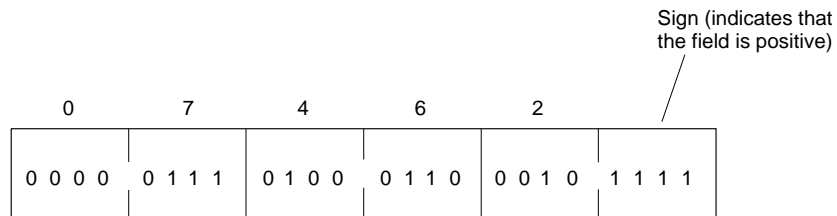


Figure 12-3. Example of the Number 7462 Stored in Packed Decimal Format

In packed decimal format, hex F represents a positive sign and hex D represents a negative sign.

If you define a numeric field with a value of +1234 as an unsigned packed decimal field, it is stored as:

```
01 23 4F
```

If you define a numeric field with a value of -1234 as an unsigned packed decimal field, it is also stored as:

```
01 23 4F
```

However, if you define a numeric field with a value of -1234 as a *signed* packed decimal field, it is stored as:

```
01 23 4D
```

To print or display a packed decimal field, you must translate it into its zoned decimal equivalents.

The maximum length of a packed decimal field is 15 digits (8 bytes of storage). The following list shows the number of bytes of storage required to store zoned decimal and packed decimal fields:

Digits in Number	Zoned Decimal Format	Packed Decimal Format
1	1	1
2	2	2
3	3	2
4	4	3
5	5	3
6	6	4
7	7	4
8	8	5
9	9	5
10	10	6
11	11	6
12	12	7
13	13	7
14	14	8
15	15	8

Note: System/36 allows a program to include nonnumeric data in a numeric field. For example, the number 12 preceded by two blanks would be stored as:

```
40 40 F1 F2
```

The two blanks are not valid numbers and, in a non-System/36 environment, such programs cause a decimal data error. RPG II and System/36 environment COBOL will accept invalid data in zoned numeric fields and convert the invalid data to a valid number.

System/36 would allow the program to store and manipulate a packed decimal format field that had invalid numeric digits, such as the B and C in the following packed decimal format string:

```
04 B1 2C 4F
```

Internally, RPG II works in zoned arithmetic and converts fields from packed to zoned when the data is read from a file. In the conversion process, the invalid digits are changed to zero for packed fields. RPG II in turn converts the field from zoned format back to packed format before writing it to the file. COBOL applications are able to define and manipulate fields in packed decimal format without converting them to zoned. The AS/400 system (including System/36 Environment programs) does not support operations with packed decimal format fields that contain invalid digits. If the program detects a decimal data error, an error message will appear. The user should end the application, correct the data in error, and change the program(s) that introduced the invalid digits.

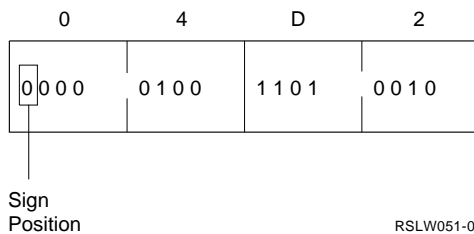
Binary Format

Data in binary format is represented in storage in binary digits (as a number to the base 2). A binary field occupies less storage than a zoned decimal field and sometimes occupies less storage than a packed decimal field. The following list shows how many bytes of storage are occupied by binary fields of various lengths:

Number of Digits in Field	Bytes Required to Represent the Number	Bytes of Storage Occupied
1 to 2	1	2
3 to 4	2	2
5 to 6	3	4
7 to 9	4	4
10 to 11	5	8
12 to 14	6	8
15 to 18	7	8

A binary field normally reserves the farthest left bit in storage as the sign position. If the sign position contains 0, the number is positive. If the sign position contains 1, the number is negative.

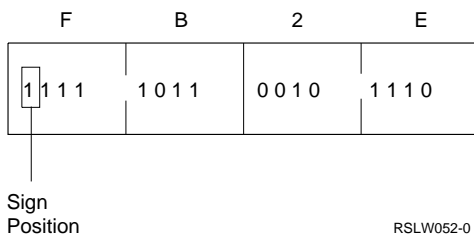
For example, if a binary field has a value of -1234 and you define the field as unsigned, it is represented in storage as hex 04D2, as shown in the following figure:



If you define a binary field as signed, it is stored as a negative number. Negative binary fields are stored in twos complement. To find the twos complement of a binary field, follow these steps:

1. Change the setting of each bit in the binary representation:
 - Change each 0 to 1.
 - Change each 1 to 0.
2. Add 1 to the new binary representation.

For example, in the following figure, the value -1234 is stored as hex FB2E:



Note: To print or display a binary field, you must translate it into its zoned decimal equivalent.

Floating-Point Format

When data is expressed in floating-point format, each character or digit occupies one byte of storage and consists of a decimal number followed by an exponent. The decimal number is called the **mantissa**. A mantissa in the System/36 environment is the decimal number portion of data expressed in floating-point format. The **exponent** specifies a power of 10 used as a

multiplier to indicate the placement of the decimal point.

The value of a floating-point number is the mantissa multiplied by the power of 10 expressed by the exponent. For example, 3E02 is the floating-point representation of 3 times 10 to the 2 power, or 300; while 3E-2 equals 0.03.

Data in floating-point format has the following form:

[±]mantissaE[±]exponent

+ or - signs

A plus (+) or minus (-) sign is optional before the mantissa and before the exponent. If no sign is specified, the system assumes a positive mantissa or exponent. A plus (+) sign indicates positive values and a minus (-) sign represents negative values.

mantissa

The mantissa can contain from 1 to 16 digits. The mantissa must contain one actual or assumed decimal point as a leading, embedded, or trailing symbol.

E The letter E immediately follows the mantissa and indicates the exponent.

exponent

The exponent immediately follows the second optional sign character. It can contain from 1 to 3 digits.

Using Alphanumeric Fields

Alphanumeric data can contain letters, numbers, and special characters. Special characters are those characters that are not alphabetic or numeric characters.

See the correct language book for information about the lengths allowed for alphanumeric fields.

Using Keys

A **key** is one or more characters used to identify the record and establish the record's order within an indexed file. The maximum key length is 120 bytes, although in some cases the limit is lower (for example, DFU, RPG II, and Query/3X restrict the key length to 99 bytes).

A multiple-index file uses a separate key for each index. The key for the indexed physical file must consist of fields that are next to each other in the record. The keys for indexed logical files (for all file types) do not have to be built from fields that are next to each other in the record.

You can use up to three fields that are not touching as the key for an indexed logical file. For example, fields 1, 2, and 3 can be the key for the indexed physical file. Fields 1, 3, and 10 of a record can be the key for an indexed logical file.

Chapter 7, "Files," describes files and indexes.

Allowing for Deletion of Records

When a record is deleted from a delete-capable file, the record remains in the file, but the data is not available. If you may need the data in a deleted record, do not use a delete-capable file. Have your program place a delete code in the record. When the file is processed, your program can check for this code. For example, if a customer record becomes inactive, place a delete code in the record and have the program check for this delete code. When the program finds this delete code, it can bypass the record instead of processing it. Remove the delete code to make the customer record active.

The system does not treat records with a user-specified delete code as deleted records.

To remove deleted records from a file, use the COPYDATA or SAVE procedure. The *System/36 Environment Reference* book has information about these procedures.

Determining Field Size

Field size depends on the data in the field. Each field should be long enough to contain the largest number of characters for that field in any record. For example, the length of each customer's name varies, and while 20 positions is usually long

enough to contain a customer's name, a customer could need more than 25 characters.

Defining Record Length

The lengths of different fields in a record vary, but a field should have the same length in every record, and all records in one file must have the same length. Record length is the sum of the field lengths, plus any extra space reserved for new fields. The maximum record length for a disk file is 4096 positions.

Allowing for New Fields

Make your record length longer than required so that you can add new fields (although this requires extra space and can decrease performance). For example, if a file contains 1000 records, and if 20 positions are set aside in each record for future additions, the file requires an additional 20 000 bytes.

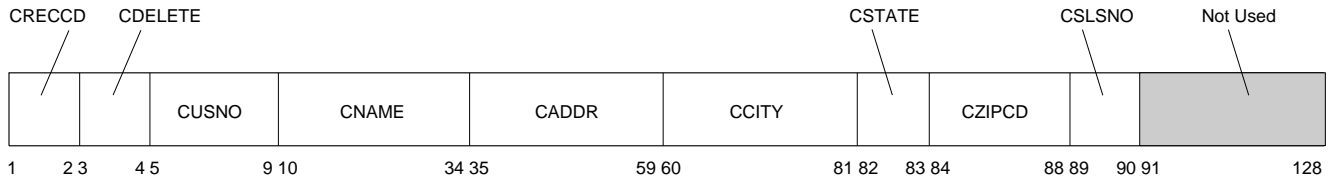
To add new fields as they are needed, you can use a program that reads the file, adds the new fields, and writes the larger file back to disk.

Describing Record Layout

Record layout includes the order of the fields in the record, the length of each field, and the name of each field. Programs are easier to write when you describe your record layouts. Figure 12-4 shows the layout of a master customer record. The field names all begin with the letter C to indicate that the record is from the customer master file.

Figure 12-5 shows one method of describing record layout using a standard form.

Figure 12-6 on page 12-8 is a full-sized sample of a record layout description form. You can make copies of this form to use in describing your record layouts.



RSLW035-0

Figure 12-4. Sample Record Layout of a Customer Record

INPUT/OUTPUT Record Description

Record Name Customer Master System 36 File No. _____ Page 1 of 1

File Name CMAS Date _____

File Organization Indexed Sequence Organized in Key Sequence Prepared by MST

Record Length 128 Key Customer Number Key Length 6

Created by CUSTENT Used by CUSTLIST Updated by CUSTINQ

Values	Field Description	Field Name	Length	Decimal Position	Format	Location	
						From	To
MA	Record Code	CRECCD	2		A	1	2
D or blank	Delete Code	CDELETE	1		A	3	3
	Customer Number	CUSNO	6	0	N	4	9
	Customer Name	CNAME	25		A	10	34
	Customer Address	CADDR	25		A	35	59
	City	CCITY	22		A	60	81
	State	CSTATE	2		A	82	83
	Zip Code	CZIPCD	5		N	84	88
	Salesman Number	CSLSNO	2		N	89	90
	-- Not Used --		38		A	91	128

RSLW061-1

Figure 12-5. Sample Record Layout Description Form

Chapter 13. Communications

The AS/400 system uses data communications to send and receive information from different devices and systems. The system acts as a host system to remote work stations, as a secondary station to a remote host system, or it communicates with another system as a peer. This chapter contains information about:

- Communications configuration on the AS/400 system
- OS/400 subsystem considerations
- Writing communications programs
- Running communications applications in the System/36 environment
- Migration considerations

For additional information about communications on the AS/400 system, see the *Communications Management* book.

Configuring the Communications Environment

You must configure both your local and remote system before communications can be established. This section describes the relationships of the essential elements of System/36 Interactive Communications Feature (SSP-ICF) and their counterparts on the AS/400 system, the OS/400 Intersystem Communications Function (OS/400-ICF), hereafter referred to as ICF.

System/36 Background

On System/36, a primary SSP-ICF relationship exists between the remote location requested by an application and the command used to activate that remote location:

- **Remote location name.** The remote location name is a keyword parameter (LOCATION) on the SESSION OCL statement, or a positional parameter on the MSRJE, PASSTHRU, EM3270, or ES3270 procedure command. A **positional parameter** is a parameter that must appear in a specified location relative to other parameters. It is the name of the spe-

cific information within a library member (called a subsystem member) that defines the communications facilities used by that application. A subsystem member has one or more remote location definitions that must be activated (using the ENABLE procedure) when those remote locations are requested by an application.

Note: The concept of a System/36 SSP-ICF subsystem does not exist on the AS/400 system. See “OS/400 Subsystem Considerations for System/36 Users” on page 13-7 and the *Work Management* book for information on OS/400 subsystems.

The important parameters on the ENABLE command are the subsystem member name (first parameter) and the line number (third parameter). The subsystem member name specifies which subsystem member contains the remote locations being enabled. The specific remote location name (fifth parameter) might be present, but it is often omitted. If the remote location name is omitted, remote locations in the subsystem member defined with ACTIVATE LOCATION AT ENABLE = Y are enabled.

- **Locating the correct subsystem member.** Unless the ENABLE procedure command includes the remote location parameter, no obvious link exists between a remote location name and the subsystem member that contains it. If you do not know which subsystem member contains the remote location you need, examine the procedures used to run your SSP-ICF applications.

If the procedures containing SSP-ICF applications also contain an ENABLE statement, the first parameter of the ENABLE statement should be the name of the subsystem member that contains the remote location definition. Otherwise, you must determine which subsystems are active when the SSP-ICF applications are running and review each (using the CNFIGICF procedure) to determine which has the remote location referred to by the applications.

Communications Procedures Examples

The examples in this section show the mapping between System/36 configuration concepts and AS/400 configuration concepts. They show multiple single statements entered separately. The multiple calls can be replaced by a call to a CL program containing these statements. See the *CL Reference* book for information about writing CL programs. See the *Communications Configuration* book for information about AS/400 communications configuration.

The System/36 environment on the AS/400 system supports the SESSION OCL statement with some exceptions. Figure 13-1 shows the relationships between a System/36 ENABLE statement and the SESSION OCL statement, and the corresponding AS/400 Vary Configuration (VRYCFG) command and SESSION OCL statement.

On System/36, the SESSION statement refers to a remote location name. There is an ENABLE procedure command that precedes the use of the SESSION statement, and that activates the necessary SSP-ICF support. The System/36 subsystem definition activated by the ENABLE statement must contain the remote location definition

referred to by the SESSION statement. The ENABLE statement might refer to the specific remote location name, but in many cases it refers only to the SSP-ICF subsystem member that contains the specific remote location. The ENABLE statement also includes the line number and name of the library containing the System/36 subsystem member. The System/36 subsystem member contains remote location definitions.

Figure 13-2 on page 13-4 through Figure 13-7 on page 13-10 show the relationship between the remote location name on a SESSION statement, the corresponding ENABLE procedure command, the system information used to define the communications environment, and their AS/400 counterparts.

As Figure 13-2 on page 13-4 shows, the AS/400 counterpart for each remote location definition is a **device description**. In some cases, such as for System Network Architecture (SNA) 3270 emulation or SNA upline facility (SNUF), each logical unit supported by a remote location becomes an individual device description. **Systems Network Architecture (SNA)** in IBM networks is the description of the logical structure, formats, protocols, and operational sequences for transmitting information units through and controlling the configuration and operation of networks.

System/36 Example

Remember, while remote location name is required on the SESSION statement, it is not required on the ENABLE statement. If RMTLOC2 had been defined with 'Activate location at ENABLE = Y' then it may be omitted from the ENABLE statement.

```
ENABLE MEMBX,LIBRX,2,,RMTLOC2
.
.
.
.
// LOAD PRGMX
// SESSION SYMID-xx,LOCATION-RMTLOC2,etc.
// RUN
```

-or-

```
ENABLE MEMBX,LIBRX,2,,RMTLOC2
.
.
.
.
MSRJE RMTLOC2,...etc.
```

AS/400 Example

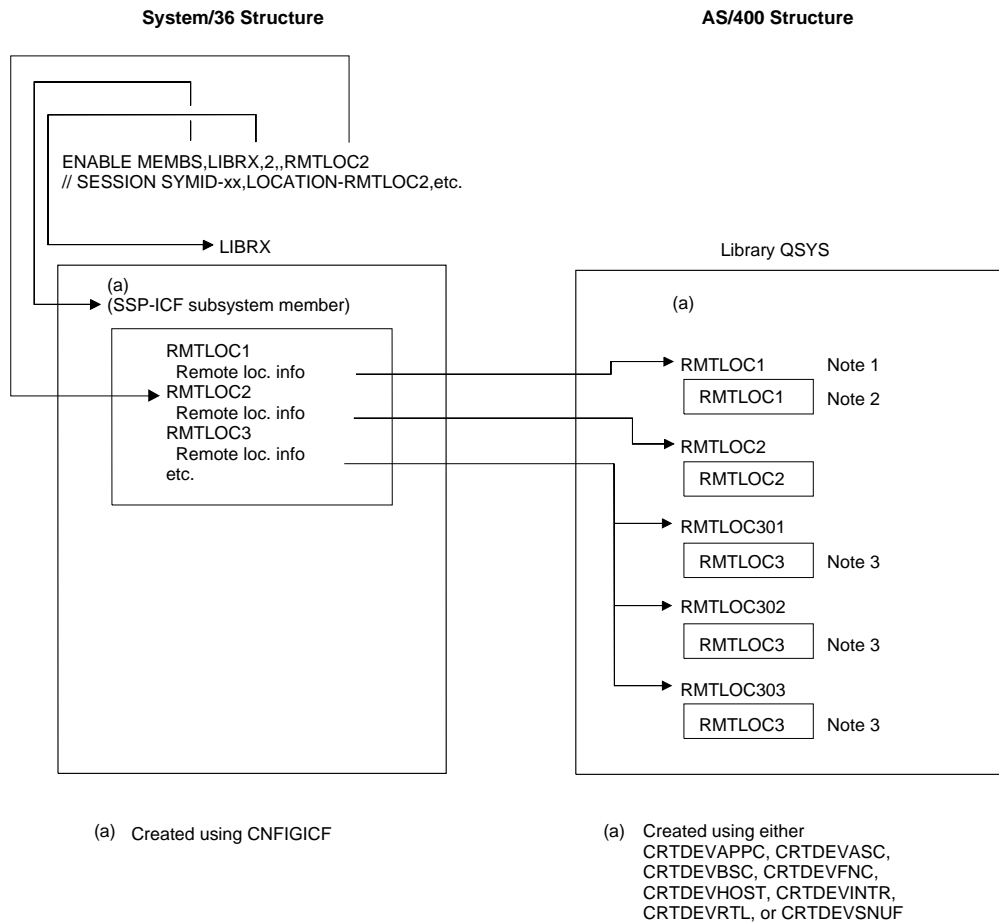
In many cases, multiple VRYCFG statements are required. For example, the line and controller must be varied on before the device.

```
VRYCFG CFGOBJ(RMTLOC2) CFGTYPE(*DEV) STATUS(*ON)
.
.
.
.
// LOAD PRGMX
// SESSION SYMID-xx,LOCATION-RMTLOC2,etc.
// RUN
```

```
VRYCFG CFGOBJ(RMTLOC2) CFGTYPE(*DEV) STATUS(*ON)
.
.
.
.
MSRJE RMTLOC2,...etc.
```

RSLW096-3

Figure 13-1. System/36 ENABLE Statement and AS/400 VRYCFG Command Example



RSLW089-5

Figure 13-2. A Remote Location Resulting in Multiple Device Descriptions

A **logical unit (LU)** is one of three types of network addressable units that serve as a port through which a user accesses the communications network. Because System/36 subsystem members alone contain little pertinent information (they are containers for remote locations), they have no counterpart on the AS/400 system.

Notes:

1. The name shown above each box is the device description name.
2. The name shown within each box is the remote location name (RMTLOCNAME) defined in each device description.
3. Some communications types require a one-to-one relationship between device description and remote location name. Figure 13-2 is an example of one remote location resulting in multiple device descriptions. This occurs if the remote

location defines three 3270 device emulation displays. However, a reference to the original remote location name is within each device description. See the *Remote Work Station Support* book for more information about remote location names.

Figure 13-2 does not show the relationship between System/36 APPC session groups and the AS/400 system counterparts for session groups (mode descriptions). **Advanced program-to-program communications (APPC)** is a data communications support that allows programs on an AS/400 system to communicate with programs on other systems having compatible communications support. APPC is the AS/400 method of using the SNA LU session type 6.2 protocol. **Advanced peer-to-peer networking (APPN*)** is a data communications support that routes data in a network between two or more APPC systems that are not directly attached. See the *APPC Programming* and the *APPN Support* books for infor-

mation about mode descriptions and APPC and APPN configurations.

On System/36, each remote location refers to a line member (the line member is always in the same library as the subsystem member) and a (with some exceptions) remote system definition within that line member. The line member contains general line information as well as remote system definitions. Multiple remote locations, even those in different subsystem members, can refer to a single line member and remote system definition. Referring to a single line member and remote system definition allows multiple subsystems to communicate with a single remote system at one time, as Figure 13-3 on page 13-6 shows.

On the AS/400 system, the remote location name for APPC is a character string up to 8 characters long, of which the first character is uppercase A to Z, \$, #, or @. The remaining characters are uppercase A to Z, \$, #, @, or 0 to 9. This convention is enforced by AS/400 system configuration (that is, you cannot enter any other characters) for the remote location name of all communications types.

The System/36 convention for the remote location name is more lenient for communications types other than APPC. Therefore, remote location names cannot be migrated from System/36 for those cases where the names do not follow AS/400 convention.

The AS/400 counterpart for each remote system definition is called a **controller description**. The AS/400 counterpart for the general line information in a line member is called a **line description**.

In the System/36 SSP-ICF line member, each remote system accessed by a switched line or X.25 switched virtual circuit can refer to a phone- or connection-list member in the same library. (**X.25** defines the interface to an X.25, packet-switching network.) It contains telephone or connection numbers that automatically connect to the remote system.

As Figure 13-4 on page 13-7 shows, there is no AS/400 counterpart for the phone/connection list.

However, the controller description allows for specification of one phone number or connection number (see “Automatic Dial and Telephone Number List Support” on page 13-45).

For users of System/36 BSCEL, it is possible to specify that an application connects to one of many stations. If you choose this option, you can create and maintain a **remote ID list**. The list contains all the valid remote ID’s with which BSCEL applications can connect. The list is contained in a file named #IBSRID.

The information in the System/36 BSCEL remote ID list is specified in the controller description on the AS/400 system.

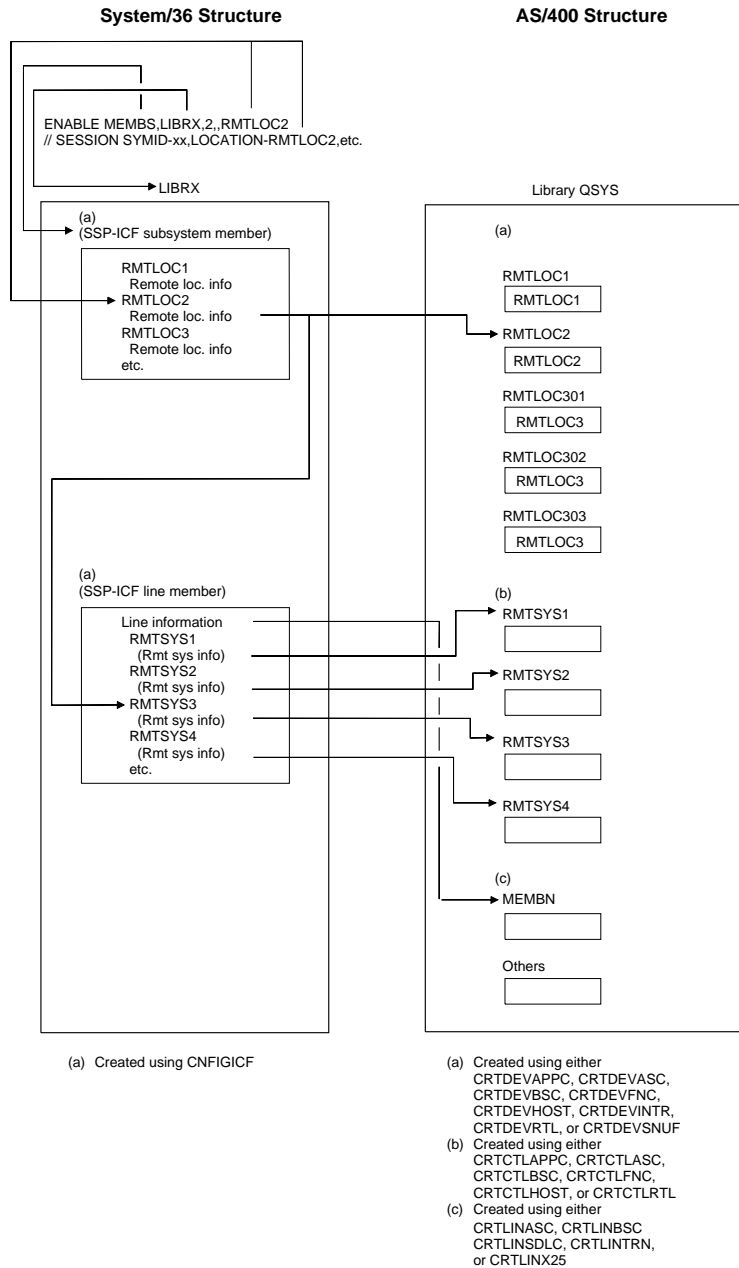
In Figure 13-5 on page 13-8, if the line described by the System/36 line member is an X.25 line, the general information part of the line definition refers to an X.25 virtual circuit configuration member (located in the same library) that supplies specific virtual circuit information for *each* remote system in the line member. A **virtual circuit** is a logical, rather than a physical, connection that is established and controlled by a managing network in a packet-switching communications environment. On the AS/400 system, the specific virtual circuit information associated with a remote system is defined in the controller description that corresponds to the remote system information.

On System/36, as Figure 13-6 on page 13-9 shows, the X.25 virtual circuit member refers to an X.25 member (always in library #X25LIB) that contains X.25 network (PSDN) default values and subsystem information.

On System/36, as Figure 13-7 on page 13-10 shows, for the line number specified on the ENABLE statement, two sets of system configuration information exist for each line:

- A system-wide set of information
- A set of information associated with each work station

The set associated with each work station allows a job started at one work station to change the line characteristics without affecting the characteristics used by jobs started at other work stations.



RSLW090-6

Figure 13-3. Multiple Subsystems Can Communicate with a Single Remote System

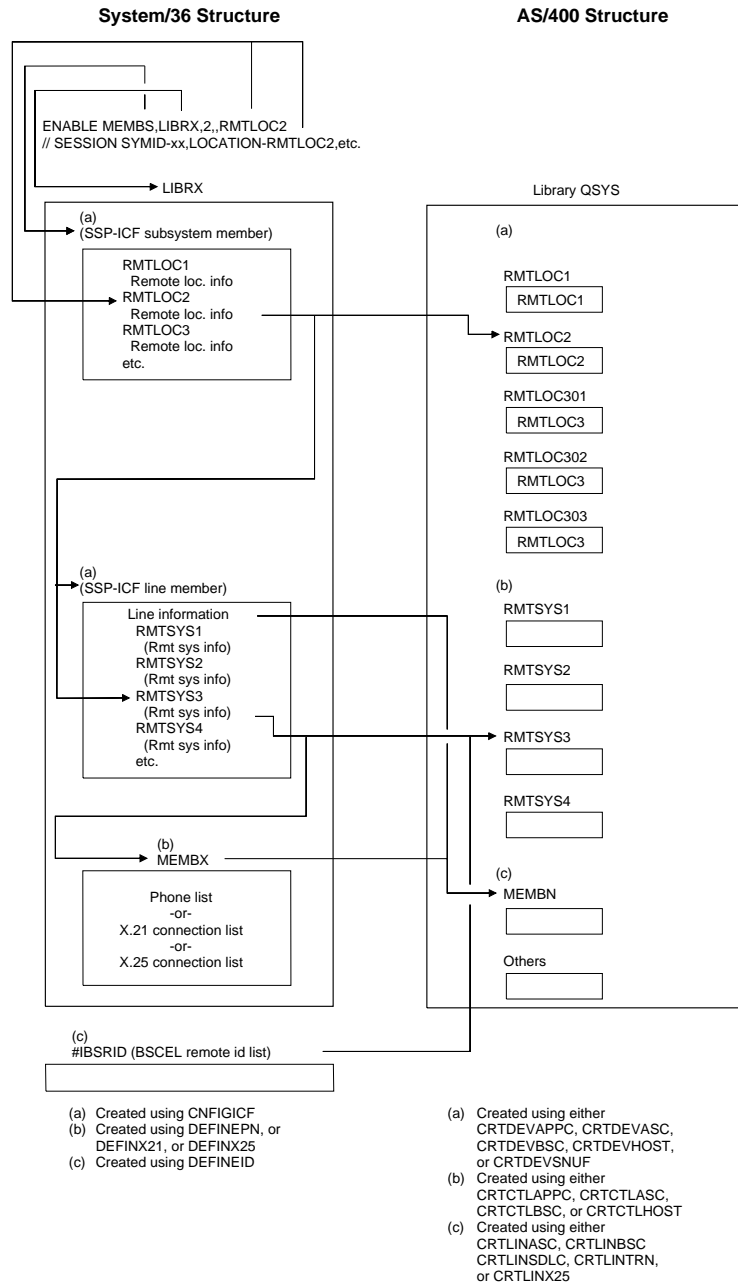
ENABLE and VRYCFG Hierarchy and Examples

On System/36, when a remote location is activated, both the specific remote system and the line are activated.

Note: The physical line used by the application is not specified until the ENABLE procedure is started. Therefore, one subsystem member and one line member can be used on lines with the same characteristics.

On the AS/400 system, the System/36 hierarchy is reversed. When the system activates a line description, network elements that depend on the description can be activated. However, you cannot activate a device description until you activate a controller and line description.

Note: Also, a **resource name** associates the line description with the hardware. Therefore, if a line, controller, and device are to be used on a different line, you must change the resource name in the line description (using a Change Line Description [CHGLINxxx] command) before activating it.



RSLW091-5

Figure 13-4. System/36 Remote ID List Mapped to the AS/400 Controller Description

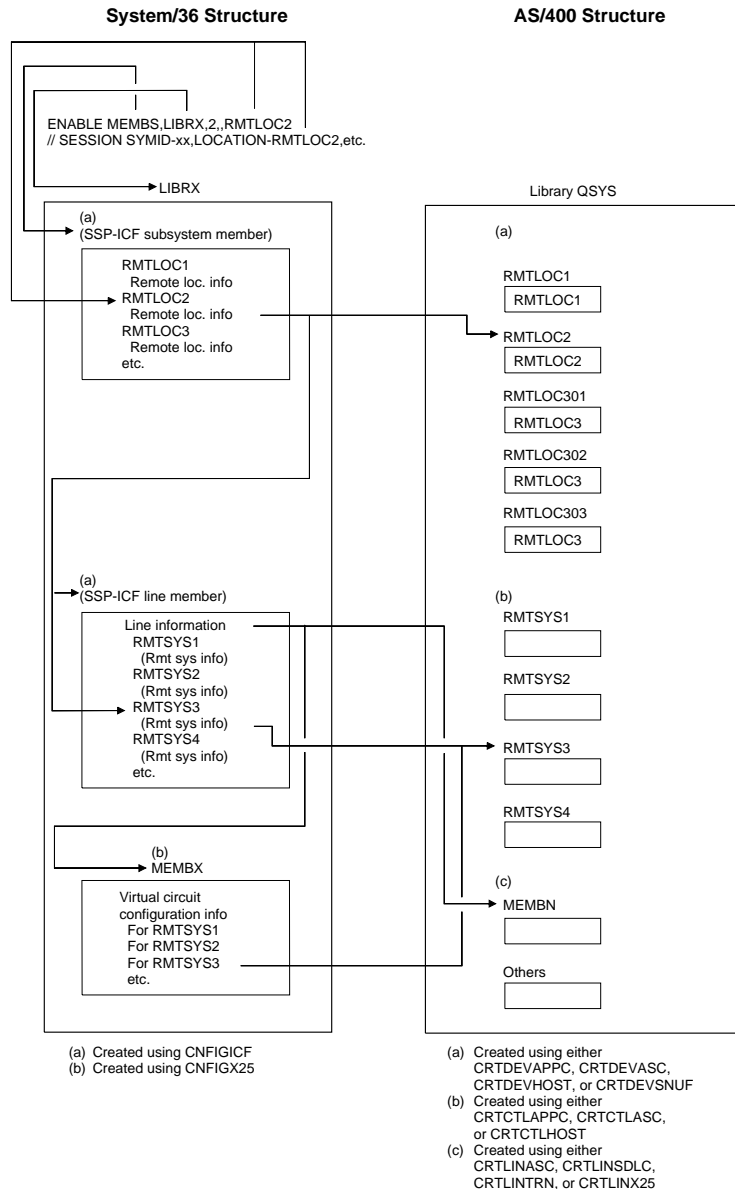
Figure 13-8 on page 13-11 shows an example of the hierarchy on each system.

Note: The VRYCFG command can be replaced by a call to a CL program. See the *CL Programming* book for more information.

OS/400 Subsystem Considerations for System/36 Users

This section provides information on running OS/400 subsystems on the AS/400 system.

Note: The concept of System/36 SSP-ICF subsystems does not exist on the AS/400 system.



RSLW092-5

Figure 13-5. X.25 Virtual Circuit Configuration

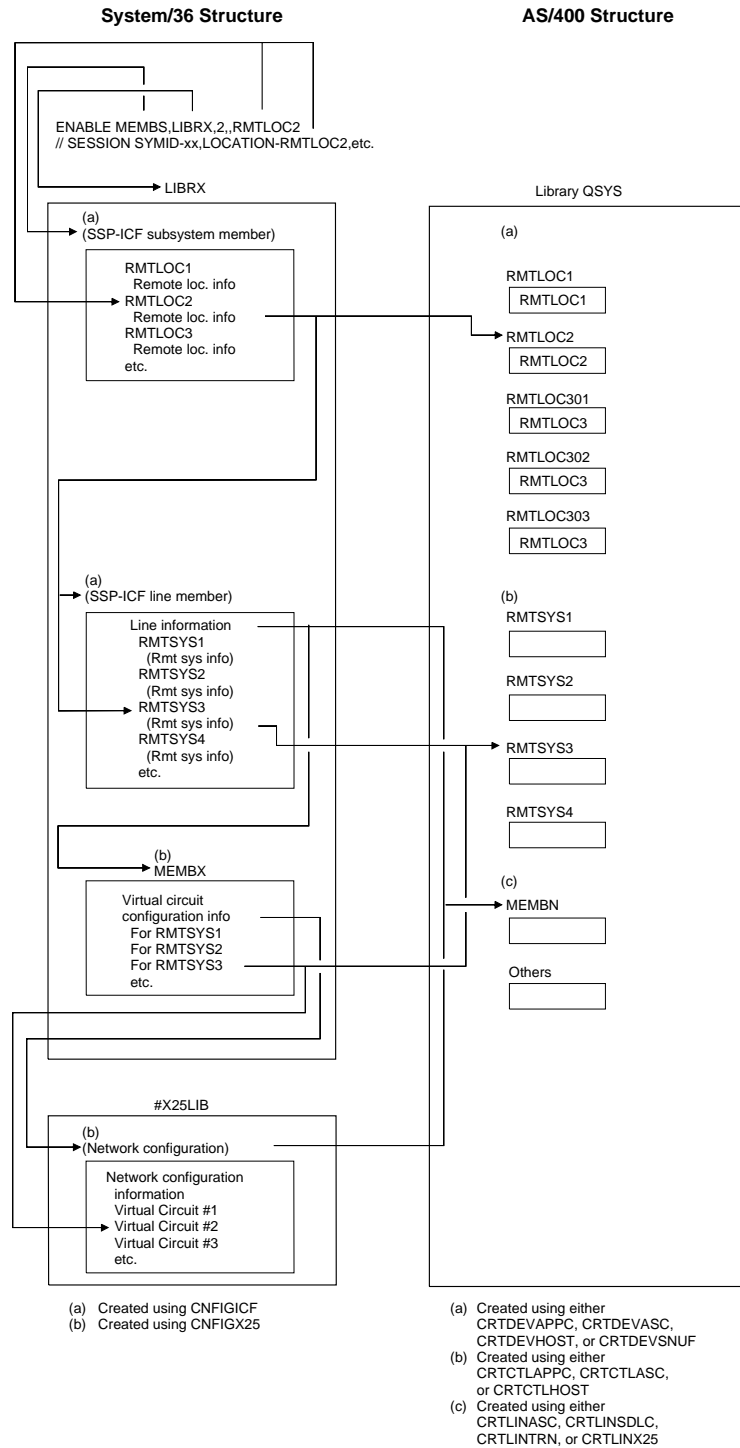
See the *Work Management* book for information about OS/400 subsystems.

Considerations for System/36 Environment Program Start Requests

When an OS/400 subsystem (such as QBASE) receives a program start request, it attempts to determine whether the job should run in the System/36 environment by checking the following:

- A search is made for a prestart job entry matching the program name passed on the program start request. If a match is found, the program start request is attached to a prestart job.

Note: A procedure name cannot be specified on the prestart job entry.
- A search is made within file QS36PRC for a procedure that matches the procedure or program name passed on the program start request. The OS/400 subsystem searches:



RSLW093-5

Figure 13-6. X.25 Subsystem Information Defined in the AS/400 Line Description

- The library specified on the program start request (which becomes the current library for the job)
- Library #LIBRARY

- The System/36 environment library (QSSP)

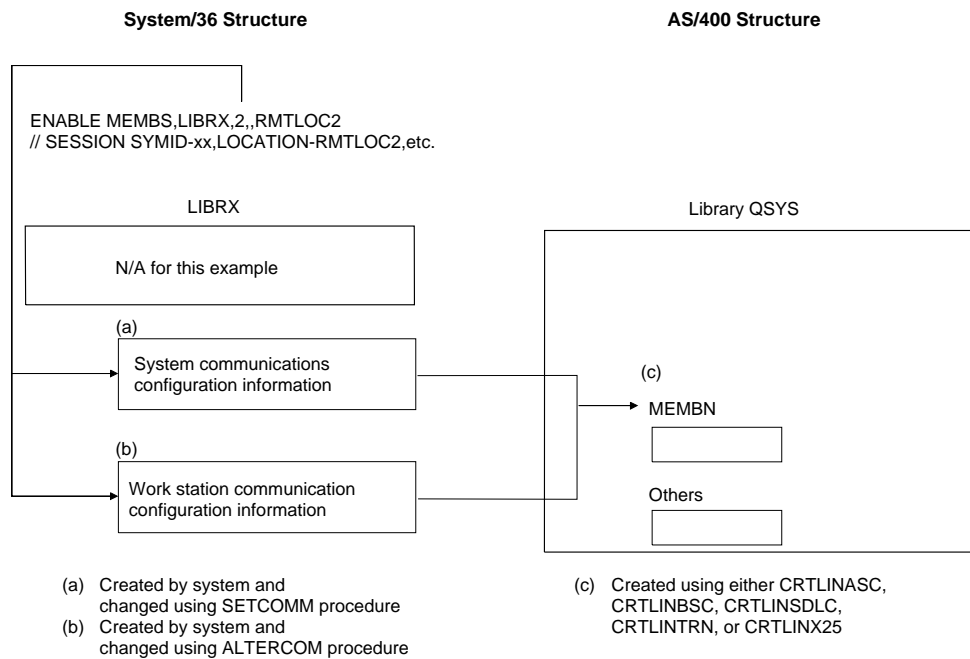
If the subsystem finds a procedure, the job runs in the System/36 environment. A procedure is found if the name follows System/36

defined naming conventions. See “Library Member Names” on page 6-2 for information about library member naming conventions.

If the AS/400 subsystem does not find a procedure, it does the following:

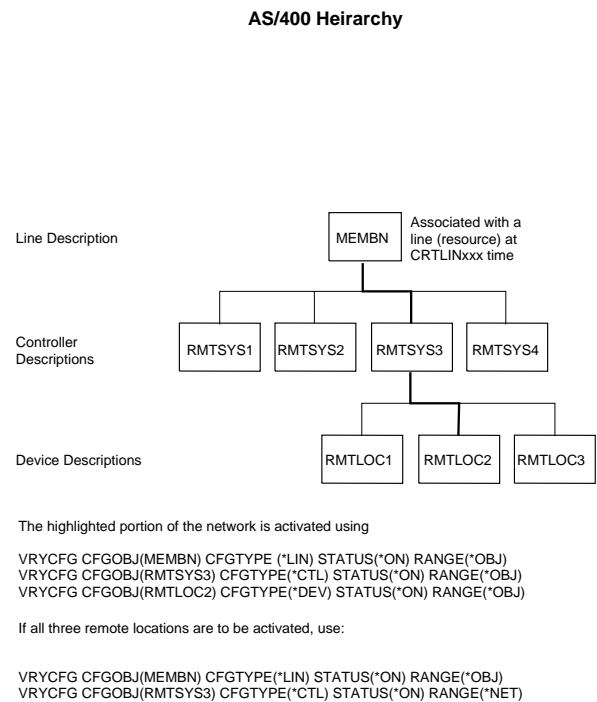
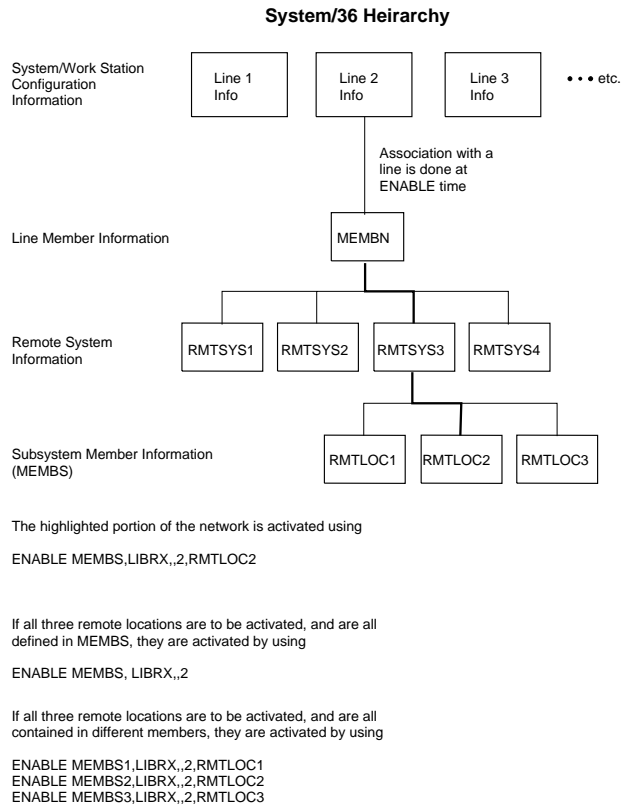
- Determines that the job should not run in the System/36 environment.

- Tries to start the job as an OS/400 job.
- Searches the library specified on the program start request for a program that matches the procedure or program name passed on the program start request. If it finds the program, the job starts in the subsystem as an OS/400 job. If the program is not found, the program start request fails.



RSLW094-4

Figure 13-7. Line Configuration Information Defined in the AS/400 Line Description



RSLW095-4

Figure 13-8. ENABLE and VRYCFG Hierarchy and Examples

Errors on Program Start Requests

If an AS/400 system detects an error when it receives a program start request, the following occurs:

- A notification of the error is sent to the remote system that sent the program start request. See the *ICF Programming* book for information about what happens on the source system when the target system detects an error.
- If the application that initiated the program start request is an OS/400 program, a return code indicating the error is given to the application. Return codes for System/36 environment ICF procedure start failures are the same as the System/36 return codes for SSP-ICF procedure start failures. Some System/36 SSP-ICF subsystems support giving application system messages when the procedure start fails. If the source program issues an input operation to retrieve the message identifying the failure, it receives an

OS/400 message rather than a System/36 message.

- Error message CPF1269 is sent to the QSYSOPR message queue (on the system on which the target program is located). The CPF1269 message has two reason codes. If one of the reason codes in the message is zero and the other is not zero, the nonzero reason code indicates why the procedure start request was rejected.

If both reason codes are nonzero, the OS/400 subsystem cannot determine whether the request was to start a System/36 environment procedure or an OS/400 environment program. One of the reason codes indicates why the program start request was rejected in the System/36 environment. The other reason code indicates why the program start request was rejected by the OS/400 environment. When you receive two reason codes, determine which environment the job was to run in and correct the problem for that environment.

The following table describes the reason codes:

Figure 13-9 (Page 1 of 4). Reason Codes for Rejected Program Start

Reason Code	Reason Description	Communications Types
401	Program start request received to a device that is not allocated to an active subsystem.	All
402	Requested device is currently being held by a Hold Communications Device (HLDCMNDEV) command.	All except APPC
403	User profile is not accessible.	All
404	Job description is not accessible.	All
405	Output queue is not accessible.	All
406	Maximum number of jobs defined by subsystem description are already active.	All
407	Maximum number of jobs defined by communications entry are already active.	All
408	Maximum number of jobs defined by routing entry are already active.	All
409	Library on the library list is exclusively in use by another job.	All
410	Group profile cannot be accessed.	All
411	Insufficient storage in machine pool to start job.	All
412	System job values are not accessible.	All
501	Job description was not found.	All
502	Output queue was not found.	All
503	Class was not found.	All
504	Library on initial library list was not found.	All

Figure 13-9 (Page 1 of 4). Reason Codes for Rejected Program Start

Reason Code	Reason Description	Communications Types
505	Job description or job description library is damaged.	All
506	Library on the library list is destroyed.	All
507	Duplicate libraries were found on library list.	All
508	Storage-pool defined size is zero.	All
602	Transaction <i>program-name value</i> is reserved but not supported.	All
604	Matching routing entry was not found.	All
605	Program was not found.	All
704	Password is not valid.	All except retail
705	User is not authorized to device.	All
706	User is not authorized to subsystem description.	All
707	User is not authorized to job description.	All
708	User is not authorized to output queue.	All
709	User is not authorized to program.	All
710	User is not authorized to class.	All
711	User is not authorized to the library on the library list.	All
712	User is not authorized to group profile.	All
713	User ID is not valid.	All except retail
714	Default user profile is not valid.	All except finance
715	Neither password nor user ID was provided, and no default user profile was specified in the communications entry.	All

Figure 13-9 (Page 2 of 4). Reason Codes for Rejected Program Start

Reason Code	Reason Description	Communications Types
718	No user ID.	All except retail
722	A user ID was provided, but no password was sent.	All except retail
723	No password was associated with the user ID.	All
725	User ID does not follow naming convention.	All except retail
726	User profile has been disabled.	All
801	Program initialization parameters are present but not allowed.	All except retail
802	Program initialization parameter exceeds 2000 bytes for a prestart job.	All except retail
803	Subsystem is ending.	All
804	Prestart job is inactive or is ending.	All
805	WAIT(*NO) was specified on the prestart job entry, and no prestart job was available.	All
806	Maximum number of prestart jobs that can be active on a prestart job entry was exceeded.	All
807	Prestart job ended when a program start request was being received.	All
901	Program initialization parameters are not valid.	All except retail
902	Number of parameters for program is not valid.	All
903	Program initialization parameters required but not present.	All

Figure 13-9 (Page 2 of 4). Reason Codes for Rejected Program Start

Reason Code	Reason Description	Communications Types
1001	System logic error. Function check or unexpected return code encountered.	All
1002	System logic error. Function check or unexpected return code encountered while receiving program initialization parameters.	All
1501	Character in procedure name is not valid.	All
1502	Procedure not found.	All
1503	S/36 environment library not found.	All
1504	Library QSSP not found.	All
1505	File QS36PRC not found in library QSSP.	All
1506	Procedure or library name is longer than 8 characters.	All
1507	Current library not found.	All
1508	Not authorized to current library.	All
1509	Not authorized to QS36PRC in current library.	All
1510	Not authorized to procedure in current library.	All
1511	Not authorized to S/36 environment library.	All
1512	Not authorized to file QS36PRC in S/36 environment library.	All
1513	Not authorized to procedure in S/36 environment library.	All
1514	Not authorized to library QSSP.	All
1515	Not authorized to file QS36PRC in QSSP.	All

Figure 13-9 (Page 3 of 4). Reason Codes for Rejected Program Start

Reason Code	Reason Description	Communications Types
1516	Not authorized to procedure QS36PRC in QSSP.	All
1517	Unexpected return code from S/36 environment support.	All
1518	Problem phase program not found in QSSP.	All
1519	Not authorized to problem phase program in QSSP.	All
1520	Maximum number of target programs started (100 per S/36 environment).	All
1901	Record or block size exceeds maximum size.	BSCEL
1902	ASCII and transparency are mutually exclusive.	BSCEL
1903	Transparent mode and blank compression conflict.	BSCEL
1904	Block length is required with data format.	BSCEL
1905	Blank truncation and ITB mode conflict.	BSCEL
1906	Blank compression and ITB mode conflict.	BSCEL
1907	3740 multiple files and ITB mode conflict.	BSCEL
1908	Record separator mode and transparent modes conflict.	BSCEL
1909	Record separator and ITB mode conflict.	BSCEL
1910	Record length exceeds block length.	BSCEL
1911	Record separator character is not valid.	BSCEL
1912	BLOCK(*USER) and RMTBSCEL(*YES) conflict.	BSCEL

Figure 13-9 (Page 3 of 4). Reason Codes for Rejected Program Start

Reason Code	Reason Description	Communications Types
1913	BLOCK(*NOSEP) and TRUNC(*YES) conflict.	BSCEL
1914	Program name is not valid.	BSCEL
1915	Program start request record was too long.	BSCEL
2001	FMH5 field length is not correct.	APPC
2002	Concatenation code is not valid.	APPC
2003	Function-management-header type is not 5.	APPC
2004	Command code field in function management header is not valid.	APPC
2005	Length for fixed-length fields is not valid.	APPC
2006	Conversation type is not supported.	APPC
2007	Synchronization level is not supported.	APPC
2008	Reconnection is not supported.	APPC
2009	Transaction program name field length is not valid.	APPC
2010	Access code subfield length is not valid.	APPC
2011	UOW-ID subfield length is not valid.	APPC
2012	UOW-ID contents are not valid.	APPC
2013	Requested device is currently being held by a Hold Communications Device (HLDCMDEV) command.	APPC
2014	Transaction program name value is reserved but not supported.	APPC

Figure 13-9 (Page 4 of 4). Reason Codes for Rejected Program Start

Reason Code	Reason Description	Communications Types
2015	LU service request received but the LU service job is not active.	APPC
2016	Pre-verified user ID received, but device description specifies SECURELU(*NO).	APPC
2017	No user ID was provided, but a password was received.	APPC
2018	No user ID was provided, but a user profile was received.	APPC
2019	Remote system indicated it sent a pre-verified user ID, but no user ID was received.	APPC
2020	Remote system sent a pre-verified user ID, but also sent a password.	APPC
2021	Remote system sent a user ID (which it had not verified) and failed to send a password.	APPC
2022	Password received, but this is a nonsecure system.	APPC
2111	Program name missing or not valid.	SNUF
2118	Function management header not valid.	SNUF
2123	End bracket or end chain missing.	SNUF
2501	System logic error. Function check or unexpected return code encountered while processing a program start request.	Intrasytem
2502	Temporarily unable to allocate needed resources for a program start request.	Intrasytem

Figure 13-9 (Page 4 of 4). Reason Codes for Rejected Program Start

Reason Code	Reason Description	Communications Types
2503	No subsystem accepting program start requests for this device.	Intrasytem
2601	Program name missing and device is not configured for display station pass-through.	Retail
2651	*EXEC not specified.	Finance
2652	Blank missing after *EXEC.	Finance
2653	Program name missing.	Finance
2654	Program name longer than 10 characters.	Finance
2655	Library name longer than 10 characters.	Finance

Subsystem Descriptions/Communications Entries

Before an AS/400 system accepts a program start request, a subsystem that supports communications must be started. Two IBM-supplied subsystems accept program start requests for all communications types:

- QBASE
- QCMN

If you use the IBM-supplied subsystems QBASE or QCTL, QINTER, QBATCH, and QCMN, your AS/400 system automatically accepts program start requests.

If you want to create your own subsystem descriptions for handling communications:

- See the *Work Management* book for a description of what a subsystem is and how it works.
- See the *Work Management* and the *CL Reference* books for a description of the following commands used to create a communications subsystem:

- Use the Create Subsystem Description (CRTSBSD) command to create a subsystem description.
- Use the Add Communications Entry (ADDCMNE) command to define the types of communications your subsystem supports.
- Use the Add Routing Entry (ADDRTGE) command to define the program jobs started as a result of a program start request.

Note: See the special communications considerations for the ADDRTGE command in the *Work Management* book.

Subsystem Communications Device Allocation

If you use the IBM-supplied communications subsystems, communications devices are allocated automatically to subsystems.

Note: Do not have subsystems QBASE and QCMN active at the same time.

If you want to create subsystem descriptions, see the *Work Management* book for information about communications device allocation for subsystems.

OS/400 Intersystem Communications Function (ICF)

With the ICF and its underlying support, you can write application programs that use communications lines to communicate with (send data to and receive data from) programs on other systems. You can use the System/36-compatible COBOL and RPG II programming languages in the System/36 environment to write application programs. On System/36, you can also write communications applications in assembler and BASIC. OS/400 BASIC does not support communications, and assembler is not supported on the AS/400 system. Therefore, System/36 communications applications written in either of these two languages must be rewritten.

Communications between application programs use ICF and the underlying support provided by various communications types. ICF provides several communications types so the AS/400 system can communicate with various remote

systems with different communications methods, such as:

- Binary synchronous communications (BSC)
- Systems Network Architecture (SNA)
- Asynchronous communications

The following communications types are supported by ICF:

- Advanced program-to-program communications (APPC)
- Systems Network Architecture upline facility (SNUF)
- Binary synchronous communications equivalence link (BSCSEL)
- Asynchronous communications
- Retail communications
- Finance communications
- Intrasystem communications

The AS/400 system supports the following types of communications lines (all the lines do not have to be the same type):

- Switched point-to-point (manual or automatic answer, manual or automatic call)
- Nonswitched point-to-point
- Nonswitched multipoint
- IBM Token-Ring Network
- X.25 network
- Ethernet

Each communications type, with the exception of intrasystem communications, requires at least one communications line to communicate with a remote system.

See the *ICF Programming* and the *Remote Work Station Support* books for general information about the remote systems and devices supported by each communications type. For detailed information on the support provided, refer to the following books:

- *APPC Programming*
- *APPN Support*
- *Asynchronous Communications Programming*
- *BSC Equivalence Link Programming*
- *Finance Communications Programming*
- *Intrasystem Communications Programming*
- *Retail Communications Programming*
- *SNA Upline Facility Programming*

The AS/400 communications types are equivalent to System/36 System Support

Program—Interactive Communications Feature (SSP-ICF) subsystems. System/36 applications written to a System/36 SSP-ICF subsystem not supported must be rewritten to one of the supported AS/400 system communications types.

ICF Files

On System/36, SSP-ICF application programs use specially defined SSP-ICF operations (such as \$\$EVOK) or interactive data definition utility (IDDU) data dictionary definitions. Programs that use SSP-ICF operations do not have an external file (for example, display files that contain \$SFGR formats). Applications that refer to IDDU data dictionaries use externally described communications file definitions stored in the data dictionary.

On the AS/400 system, device I/O is done through a device file. For communications, an ICF device file is used to send and receive data between two application programs, and describe how to present the data.

The ICF file is comparable in concept and use to a IDDU data dictionary's communications file definition on System/36. The ICF file contains record formats used by the application in I/O applications. Each record format describes the data layout and contains processing keywords that provide functions similar to the SSP-ICF operations and IDDU format definition communications functions. This information is described external to the application through data description specifications (DDS).

QICDMF File: Communications applications running in the System/36 environment that use the specially defined SSP-ICF operations (such as \$\$EVOK) automatically use a system-supplied ICF file called QICDMF in library QSYS. This file is established with characteristics that affect your application. For example:

- The maximum record length of the system-supplied QICDMF file is 4096 bytes. If you modify your System/36 environment applications to use a greater record length, the maximum record length of the file must be increased. If all your applications use a record length less than 4096 bytes, you can also decrease the maximum record length of

the file to avoid the unnecessary use of system resources.

- The maximum number of sessions allowed in a System/36 environment application (using the system-supplied QICDMF) is five. If your applications use more than five sessions (in one application), you must increase the maximum program device value of the QICDMF file.

Note: For more information on ICF files, see the *ICF Programming* book. See the descriptions of the MAXRCDLEN and MAXPGMDEV parameters of the Change Intersystem Communications Function File (CHGICFF) command in the *CL Reference* book for information about changing these characteristics.

Tying the Application to Communications Configurations

Each program that establishes a session includes at least one SESSION statement in the procedure that loads your program. Place the SESSION statement between the LOAD and RUN OCL statements for the program. Use the SESSION statement to specify three things:

- The LOCATION parameter of the SESSION statement identifies the communications configuration with which the program communicates. The LOCATION parameter identifies the remote location (specified as part of the device description) associated with the session.
- On the SYMID parameter, the SESSION statement identifies the symbolic identifier of the session. Your program uses this identifier when it establishes the session and when it issues an operation to the session. The identifier must be 2 characters. The first character must be numeric (0 through 9). The second character must be alphabetic (A to Z, \$, #, or @).
- The SESSION can also specify one or more parameters that change the attributes of the configuration for that session only.

In other words, the SESSION statement identifies the session and remote location with which your program communicates, and indirectly identifies the communications type the application uses.

The application program uses the session identifier (SYMID) on all operations to identify the session in which to direct the I/O.

Following is an example of the use of the SESSION OCL statement in the System/36 environment and its relationship to an application program:

```
YOURPROC
// LOAD YOURPGM
// SESSION LOCATION-CHICAGO,SYMID-1S

RPGPGM
      '1S'      ACQ  WSFILE

CBLPGM
77  ICF-SESSION-1S      PIC XX VALUE '1S'.

      ACQUIRE ICF-SESSION-1S FOR TRANSACTION-FILE.
```

Mapping SESSION OCL Statement to the OVRICFDEVE Command: The System/36 SESSION OCL is supported through use of the OS/400 Override Intersystem Communications Function Device Entry (OVRICFDEVE) command. The OVRICFDEVE command defines the session identifier to any ICF file the application uses.

System/36 applications use file QICDMF or the migrated ICF file. The table in Figure 13-10 on page 13-19 summarizes the mapping of the SESSION OCL statement to the appropriate OVRICFDEVE command parameters.

Figure 13-10. SESSION to OVRICFDEVE Mapping

Session	OVRICFDEVE	Finance	Retail	Intra-system	APPC	SNUF	BSCCEL	Asynchronous
LOCATION	RMTLOCNAME	X	X	X	X	X	X	X
SYMID	PGMDEV	X	X	X	X	X	X	X
GROUP	MODE				X			
APPCNET	(ignored)				X			
LWSID	DEV	X	X			X		
APPLID	APPID		X			X		
HOSTNAME	HOST					X		
RECL	RCDLEN					X	X	
FMHI	HDRPROC					X		
MSGPROT	MSGPTC					X		
BATCH	BATCH		X	X		X		
PARTNER	RMTBSCCEL						X	
SWTYP	INLCNN						X	
PHONE	(ignored)						X	
REFRESH	(ignored)						X	
RESTORE	(ignored)						X	
BLKL	BLKLEN						X	
RECSEP	BLOCK						X	
ITB	BLOCK						X	
BLANK	DTACPR,TRUNC						X	
TRANSP	TRNSPY						X	
LIBRARY	(ignored)							
MAXMSG	(ignored)		X					

Notes:

1. RPG II support for telecommunications uses BSCCEL support.
2. The APPCNET parameter is syntax checked but ignored. The system always assumes the equivalent of APPCNET=YES.
3. The PHONE, REFRESH, RESTORE, MAXMSG, and LIBRARY parameters are syntax checked, but ignored. The AS/400 system automatic dial does not support a telephone list. However, a single number can be used by specifying *YES on the AUTODIAL parameter of the Create Line Binary Synchronous Communications (CRTLINBSC) command and the requested telephone number on the CNNNBR parameter of the Create Controller Binary Synchronous Communications (CRTCTLBSC) command.
4. The LWSID parameter maps to the DEV parameter of the OVRICFDEVE command by converting the LWSID value to its hexadecimal equivalent and concatenating this hexadecimal equivalent with the value specified for the location parameter. For example, a location of CHICAGO with a LWSID of 12 would map to a DEV parameter of CHICAGO0C on the OVRICFDEVE command. For retail and finance communications, this parameter is not required.
5. For finance and retail communications on the AS/400 system, the remote location name must be unique. You may need to develop a naming scheme to ensure your remote location names are unique. You may then need to edit your SESSION OCL statements to conform to the new naming scheme.

If the OVRICFDEVE command is run directly from a System/36 procedure, or from the System/36 Command Entry display, it takes precedence over SESSION OCL statements in that job step, as well as succeeding job steps in the same System/36 job, until a Remove Intersystem Communications Function Device Entry (RMVICFDEVE) command is encountered. If an OVRICFDEVE command is run from a CL program, it applies only while the CL program runs.

See the *ICF Programming* book for more information about the OVRICFDEVE command. See the *System/36 Environment Reference* book for more information on the SESSION OCL statement.

Communications Operations

An application program uses high-level language operations and communications functions to communicate with a remote system through ICF. The high-level language operations you need for communications are introduced in this chapter. See the appropriate language book for details about each operation (its function, syntax, programming considerations, and coding example).

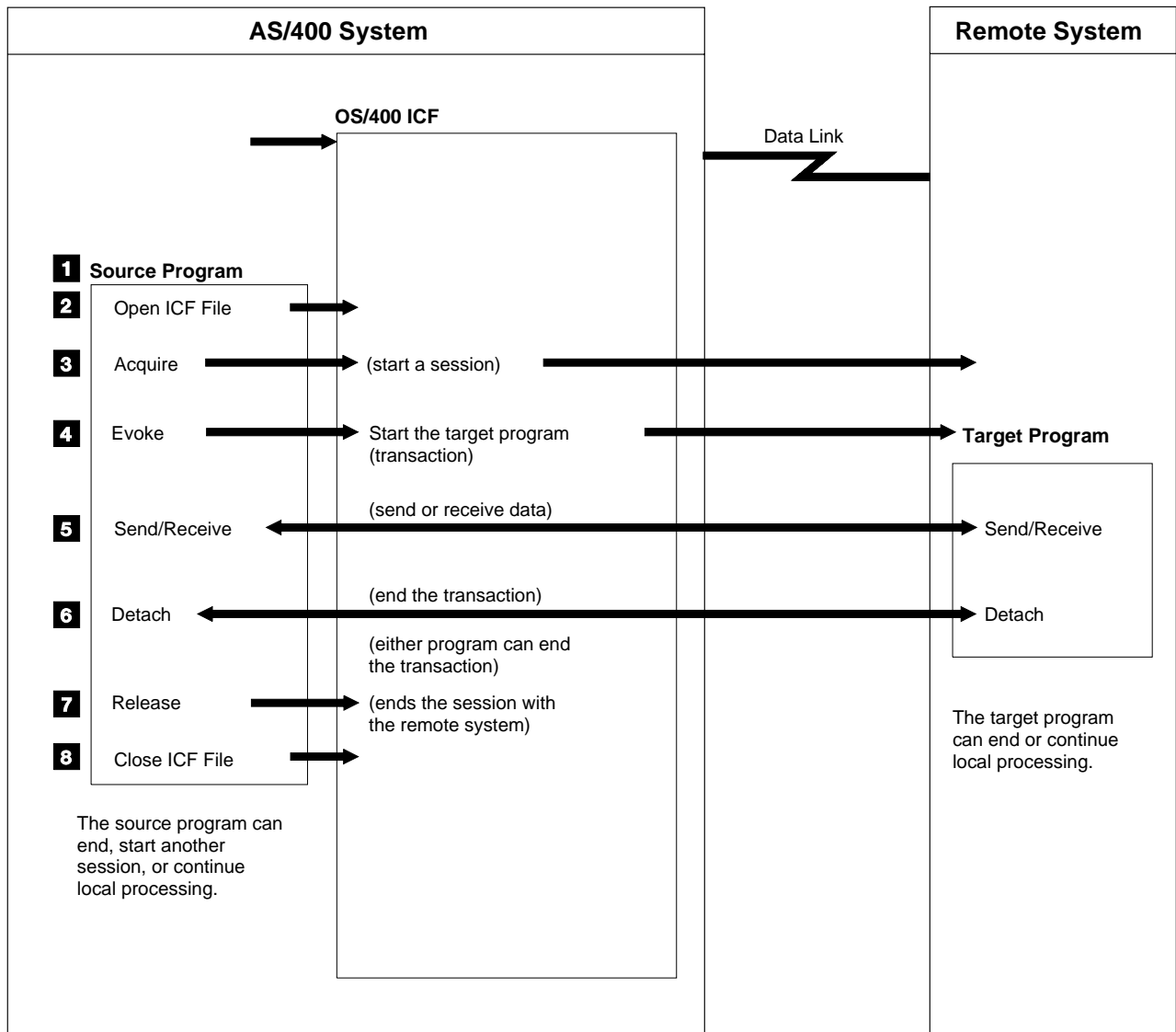
The operations you use in communications are similar to work station operations. You can use noncommunications operations for processing data between your program and the remote program.

System/36 also supports using data definitions in data dictionaries to describe data records and communications functions. The data definitions were defined through the System/36 interactive data definition utility (IDDU). Communications file definitions in IDDU data dictionaries migrate to ICF files. See “IDDU Data Dictionaries” on page 13-27 for more information.

Figure 13-11 on page 13-21 shows how your AS/400 System/36 environment application program starts a session with the remote system. The following list explains the steps:

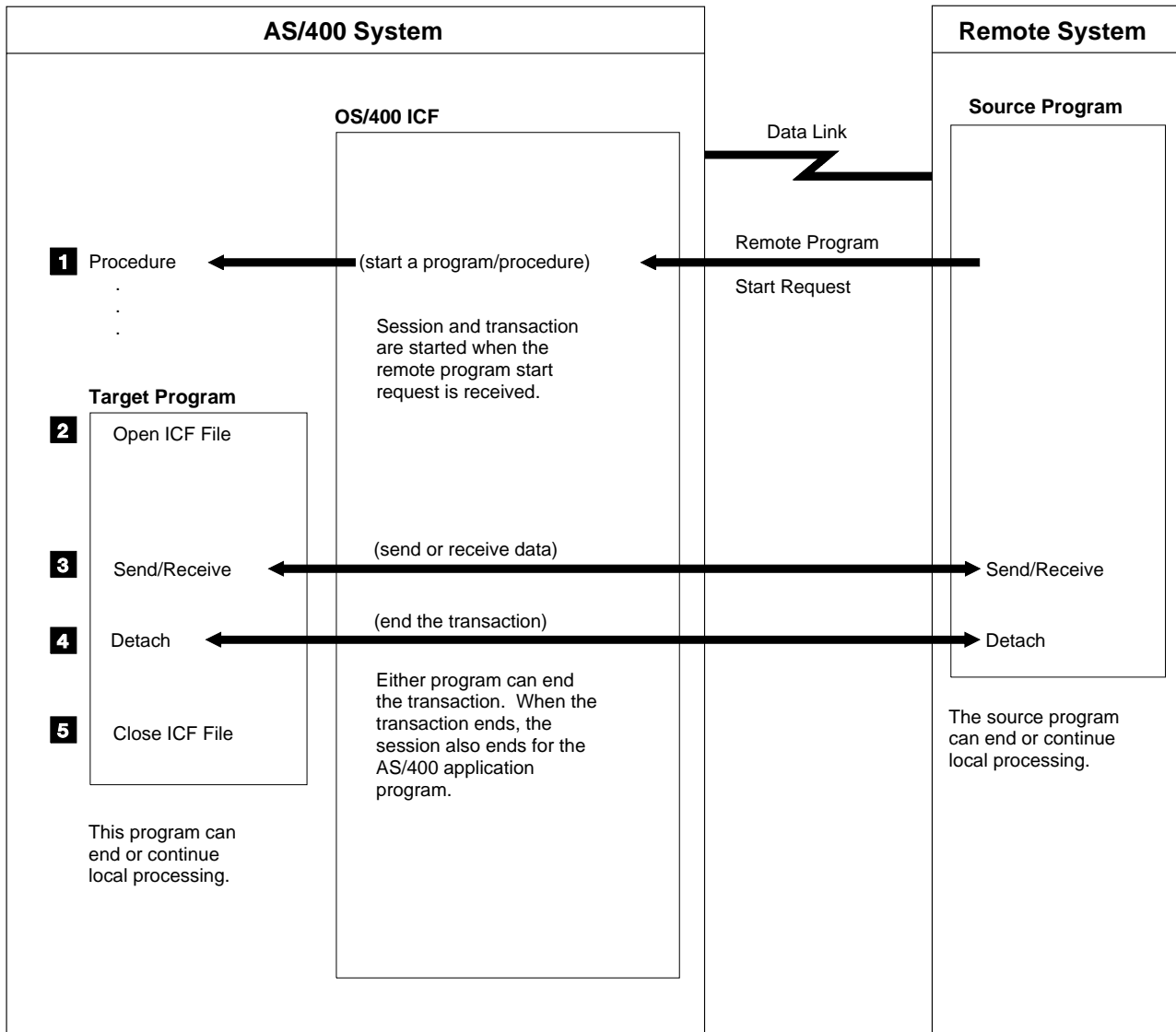
- 1** You must start the application program (source program) that communicates with the program at the remote system.
- 2** The application program must open an ICF file. The QICDMF file is automatically used for applications using the specially defined ICF operations. In an RPG II program, the file is opened implicitly.
- 3** The AS/400 program must start a session with the remote system before communications can begin. Your program starts a session when it runs an acquire operation.

When your program starts (acquires) the session, a SESSION OCL statement (associated with your program) specifies the session ID and the remote location name associated with the session.
- 4** Within each session, you can start (evoke) transactions so your program can communicate with target programs. A transaction starts when your program uses the EVOKE function to start a target program. If the remote system is System/36, a procedure starts as a result of the start (evoke).
- 5** Within each transaction, data can be sent and received between the source and target programs.
- 6** Either program can end the transaction when all data has been sent or received. Your program uses the detach function to end the transaction. When the remote system ends the transaction, your program receives a return code indicating that the transaction has ended.
- 7** When all transactions have ended, your program should end the session. Your program ends the session using the RELEASE operation or the end-of-session function.
- 8** Your program must close the ICF file. In an RPG II program, the file is closed implicitly.



RSLW087-1

Figure 13-11. Source Program



RSLW088-1

Figure 13-12. Program Started by Remote Program Start Request

Figure 13-12 shows how the remote system starts the session by running a remote program start request. The following list explains the steps.

- 1** A procedure or program starts when your system receives the program start request from the remote system. If it is a procedure, the job started is a System/36 environment job (see “Considerations for System/36 Environment Program Start Requests” on page 13-8 for more information). This procedure starts the application program that communicates with the remote system. The session and transaction also start when the program start request is received.

Because the remote system started the session and transaction, the application

program does not issue an ACQUIRE or EVOKE. However, your program can acquire other sessions with the remote system once your program is running.

- 2** The program must open the ICF file. The QICDMF file is automatically used for applications using the specially defined ICF operations. In an RPG II program, the file is opened implicitly.
- 3** You can send two types of information with the program start request:
 - Parameters for the procedure
 - Data for your program

Note: When you are creating a procedure that has parameters sent to it with the

program start request, specify PDATA-NO on the COPY control statement of \$MAINT. If you want data sent to a program, specify PDATA-YES on the COPY control statement.

After the procedure is created, you can change the PDATA value associated with a procedure using the Change System/36 Procedure Attribute (CHGS36PRCA) or Edit System/36 Procedure Attribute (EDTS36PRCA) command.

If data is sent, your program must issue an input operation to receive the data. The first operation should be an input operation. However, if you do not expect data, your program can issue an input or output operation, depending on the communications type you are using and the procedures previously set up with the remote system.

- 4** Either program can end the transaction when all data has been sent or received. Also, the session for your program ends when the transaction ends.
- 5** Your program must close the ICF file. In an RPG II program, the file is closed implicitly.

RPG II: Programs written in RPG II use a WORKSTN file to perform communications operations. RPG II WORKSTN file programs require file description, input, and output specifications. Operations for communications are similar to work station operations. That is, the same input operations are used for communications and work stations, and the output operations are performed with user-defined formats in an ICF file (migrated from an IDDU dictionary) or system-supplied formats (compatible with System/36 SSP-ICF operations).

The file description specifications for a WORKSTN file identify the following:

- The file name assigned to the WORKSTN file
- The maximum length of the data that is read from or written to a remote program

The specifications should contain the same information you would code for a WORKSTN file. Most important, if your System/36 application used IDDU, the file specification identifies the name of the ICF file that contains the record formats used by the RPG II program. The library for the ICF file

is identified by the ICFLIB parameter of the Create System/36 RPG (CRTS36RPG) command or the data dictionary parameter of the RPG procedure.

If you do not specify the ICFLIB parameter or the data dictionary parameter, the system uses the job's current library. If the system does not find the ICF file in the library used (whether from the ICFLIB parameter, data dictionary parameter, or current library), the system searches #LIBRARY and then the job's library list.

The system automatically uses the QICDMF file for communications applications that do not specify an ICF file.

The following WORKSTN operations and RPG cycle input are used with communications:

ACQ

The acquire (ACQ) operation acquires the session specified by the 2-character session identifier. You must specify the 2-character session identifier as a SYMID parameter on the SESSION OCL statement.

REL

The release (REL) operation releases the specified session.

EXCPT

The except (EXCPT) operation does many of the communications operations between two programs once a session starts. The format used determines the type of operation. The value you specify can be any one of the specially-defined ICF functions (system-supplied formats) or the name of a record format in an ICF file that externally describes the operation done.

NEXT

The next operation forces the next input to the program to come from the specified session.

READ

The read operation requests input from an invited session or from a specific session used with the NEXT operation.

Following each operation, a return code that indicates the result of the operation is returned to the application. The *System/36-Compatible RPG II User's Guide and Reference* book contains additional information about operations you can use in a communications program.

With the exception/error processing subroutine (INFSR) and error indicators for the WORKSTN operation codes (REL, ACQ, NEXT, and READ), you can control the program logic if errors occur during WORKSTN file processing. The WORKSTN file information data structure (INFDS) contains status information your program checks to determine what type of exception or error occurred. The INFDS also contains status information for normal conditions. The information is updated after each operation. Using the information in the INFDS, your program can determine which conditions to handle in the INFSR subroutine.

If you specify neither the INFSR subroutine nor the error indicators, the RPG II error handling routine handles an error. When an error is handled, your program halts.

The following table shows the *STATUS values returned in the RPG II INFDS for each major and minor return code. Use the table to determine the ICF major and minor return code or group of codes that correspond to the *STATUS value.

Major Code	Minor Code	RPG II *STATUS Value
00, 01, 02	All (except 10)	00000
00, 02	10	01321
03	00	01311
03	01, 02, 03	01299
03	08	01275
03	10	01331
03	14	01311
03	15	01299
03	1C	01275
04	02, 11, 12	01299
08	00	01285
11	00	00011
28	00	00000
34	01	01201
34	31	01201
80, 81, 83	All	01251
82	All	01281

COBOL Statements Used for Commu-

nications: Programs written in System/36-compatible COBOL use a TRANSACTION file for communications operations. Use the SELECT statement in the FILE-CONTROL paragraph to define the TRANSACTION file. The ASSIGN clause associates the TRANSACTION

file with the ICF file used. The ICFLIB parameter of the Create System/36 COBOL (CRTS36CBL) command, or the data dictionary parameter on the COBOLC procedure, identifies the library for the ICF file.

If you do not specify the ICFLIB or data dictionary parameter, the job's current library is used. If the system does not find the ICF file in the library used (whether from the ICFLIB or data dictionary parameter, or current library), the system searches #LIBRARY and then the job's library list.

The system automatically uses the QICDMF file for communications applications that do not specify an ICF file.

You must also open the TRANSACTION file for I/O in the procedure division.

In COBOL, the communications operations are done by the statements shown in the following list. See the *System/36-Compatible COBOL User's Guide and Reference* for additional information about statements used in a communications program.

ACQUIRE

Use the ACQUIRE statement to specify the session you are acquiring for the specified TRANSACTION file. You must also specify the 2-character session identifier as a SYMID parameter on the SESSION OCL statement.

ACCEPT

Use the ACCEPT statement to get the attributes of a session. You can issue the operation at any time during a session to determine the session's status. The following table shows the fields contained in the status information received by the ACCEPT statement.

Position	Value	Meaning
1	A	Session not yet acquired by the program.
	C	Session initiated by source program.
	R	Session initiated by remote program.
2	N	Input not invited.
	I	Input invited but not available.
	O	Input invited and available.

Position	Value	Meaning
3 thru 10	name	Remote location name (specified during configuration and on the SESSION OCL statement).
11	A	APPC communications type used.
12	0	Synchronization level is NONE.
	1	Synchronization level is CONFIRM.
13	M	Mapped conversation.
	B	Basic conversation.
14 thru 16	blanks	Reserved.
17 thru 33	name	Owner fully qualified LU name.
34 thru 41	name	Partner LU name.
42 thru 58	name	Partner fully qualified LU name.
59 thru 66	name	Mode.
67 thru 74	name	User ID.
75 thru 128	blanks	Reserved.

Note: Only the first 10 bytes of data are given to the application unless a 128-byte data buffer is used.

READ

Use the READ statement to receive data from a remote program. The READ statement requests input from an invited session or a specific session when you specify the TERMINAL option.

WRITE

Use the WRITE statement to perform many of the communications operations between two programs once a session starts. The value you specify for the FORMAT parameter determines the type of operation. The value specified is one of the specially defined ICF functions (system-supplied formats) or the name of a record format in an ICF file that externally describes the operation performed.

DROP

Use the DROP statement to release the specified session.

Following each operation, a return code consisting of a major and minor code is given to your program in the IBM-extended FILE STATUS area. In addition, a COBOL return code is given in the FILE STATUS field identifying the status of the operation. The following table shows the COBOL return code returned in the appropriate FILE STATUS data field and the corresponding ICF return code(s).

OS/400 ICF Return Code	COBOL Return Code
00xx, 03xx, 0800	00
01xx	01
02xx	9A
04xx	9I
1100	10
2800	9E
3401	9G
80xx	30
81xx	92
82xx	9C
83xx	9N or 9K

System-Supplied Formats: Special communications functions are defined to support communications between applications. Functions such as the following are supported:

- Starting a process on the remote system
- Sending data
- Requesting a change in direction

These operations are supported in both System/36-compatible COBOL and RPG II. They are supported as system-supplied formats used on output operations.

Figure 13-13 on page 13-26 defines the system-supplied formats supported for ICF and shows the communications types that support the formats.

Figure 13-13. System-Supplied Format Support

Operation	Retail	Finance	Intra-system	APPC	SNUF	BSCCEL	Asynchronous
\$\$CANL							
Cancel with invite	X		X		X		
\$\$CANLNI							
Cancel	X		X		X		
\$\$CNLINV ¹							
Cancel invite	X	X	X			X	X
\$\$EOS							
End of session	X	X	X	X	X	X	X
\$\$EVOK							
Evoke then invite			X	X	X	X	X
\$\$EVOKET							
Evoke with detach			X	X	X	X	X
\$\$EVOKNI							
Evoke			X	X	X	X	X
\$\$FAIL ²							
Fail			X	X	X	X	X
\$\$NRSP							
Negative response with invite	X		X		X		
\$\$NRSPNI							
Negative response	X		X		X		
\$\$RCD							
Request to write with invite			X	X	X	X	
\$\$SEND							
Put then invite or invite	X	X	X	X	X	X	X
\$\$SENDE							
Put with end of group	X	X	X		X	X	
\$\$SENDET							
Put with detach or detach			X	X	X	X	
\$\$SENDFM							
Put FMH with invite	X	X	X		X		
\$\$SENDNF							
Put FMH	X	X	X		X		X
\$\$SENDNI							
Put	X	X	X	X	X	X	X
\$\$TIMER							
Timer	X	X	X	X	X	X	X

Notes:

1. \$\$CNLINV was not supported to a finance device on System/36, but it is supported in the System/36 environment.
2. \$\$FAIL is not supported by SNUF and BSCCEL on System/36, but it is supported in the System/36 environment.

See the *ICF Programming* book for detailed information about these operations and how they work.

IDDU Data Dictionaries: System/36 IDDU data dictionaries are restored on the AS/400 system using the Restore System/36 Folder (RSTS36FLR) command or System/36 to AS/400 migration aid. A new library is created, the name of which matches the System/36 IDDU data dictionary (unless the name is overridden on the RSTS36FLR command). The library contains an AS/400 data dictionary with definitions similar to those in a System/36 data dictionary except that DDS source is generated for each IDDU communications file definition. The DDS is stored as a member of source file QIDDSSPICF and used to create a corresponding ICF file. Each communications file definition from the System/36 IDDU data dictionary is migrated to an ICF file.

Therefore, when communications file definitions exist in the System/36 data dictionary, the library created to contain the restored dictionary also holds QIDDSSPICF (with one or more members) and one or more ICF files. The names and number of QIDDSSPICF members and ICF files match the names and number of IDDU communications file definitions that exist in the System/36 data dictionary.

You cannot use IDDU on the AS/400 system to change ICF files. Modify the DDS source with a standard source editor (such as SEU) and create the ICF file again.

Consider the following about System/36 environment applications that refer to ICF files:

- The number of System/36 environment applications is determined by the System/36 to AS/400 migration aid, which determines the maximum number of sessions allowed by ICF files created from migrating IDDU communications file definitions. Alternatively, the maximum number of sessions is set at 6 when the data dictionary is restored using the RSTS36FLR command.

When your application attempts to acquire more sessions than the ICF file allows, the application receives an 82A8 return code. If your application receives this code, increase the maximum program device value for the file

using the MAXPGMDEV parameter of the Change Intersystem Communications Function File (CHGICFF) command.

- Because System/36 environment applications maintain separate areas for indicators and data, you must specify the file-level attribute INDARA in the DDS for ICF files used by System/36 environment applications. The System/36 to AS/400 migration aid gives this attribute to ICF files created from IDDU communications file definitions.
- The first record format contained in an ICF file definition determines the input capacity of the file. The record must be large enough to contain data from another record format used by the file. When the record format is not large enough, the application receives a 3401 or 3431 return code while attempting to read the larger format.

Because the largest record length supported on System/36 is 4096, ICF files created from migrating IDDU communications file definitions contain a first record format named DFTRCDFMT which has a length of 4096 records.

The following table shows the relationship between System/36 IDDU format definition communications functions and the corresponding AS/400 DDS keyword.

S/36 IDDU Function	AS/400 ICF Function
Send Fail (FAIL) function Sends a fail indication to the remote system.	FAIL
Send Request to Write (RQSWRT) function Specifies that the program is requesting permission to write.	RQSWRT
Invite (INVITE) function Specifies that the program is inviting input from the remote program.	INVITE
Evoke Process (EVOKE) function Starts a program on the remote system.	EVOKE, SECURITY, and SYNLVL
Confirm (CONFIRM) function Requests that the remote program confirms receiving data.	CONFIRM

S/36 IDDU Function	AS/400 ICF Function
Put End of Transaction (DETACH) function Informs the remote program that the sending program is ending the conversation.	DETACH

DDS keyword support for ICF files is a superset of System/36 IDDU support. See the *DDS Reference* book for more information about the DDS keywords. See the *ICF Programming* book for information about the use of DDS keywords with communications applications, and for more information about ICF files.

For more information on OS/400 support of IDDU, see the *IDDU Use* book.

Return Codes and Messages

When an error occurs that affects an AS/400 application program, the system sends a message to the program message queue of the program. The messages are logged in the job log and you can use them to determine problems. On System/36, comparable messages were given to the work station or system operator with several options, including options 2 and 3. Answering the messages with option 2 or 3 canceled the program regardless of the error handling logic of the program. Because the AS/400 messages given in the job log do not affect the processing of the application, an application that does not handle error recovery properly can loop on error conditions.

Errors are communicated to the program through the language status codes, and a major and minor return code. ICF return codes include System/36 SSP-ICF return codes. New return codes were added. For example:

- 3441** The record format specified on the input operation does not match the record format specified on the read.
- 80EF** User not authorized to file.
- 82F5** RMTFMT format selection not supported.

These return codes do not affect System/36 applications migrated to the System/36 environment

unless the application's function is broader in the System/36 environment.

See the *ICF Programming* book for a summary of the return codes supported by each communications type. Each communications programming book provides detailed information about return code communications types supported and the necessary recovery actions.

System/36 Environment Return Code Considerations:

The following System/36 return codes are unique to System/36 applications running in the System/36 environment of the AS/400 system. They are not documented in the *ICF Programming* book.

- 2800** Return code 2800 indicates a program that previously released its requester session is trying to use the released session.
- 01xx** A major return code of 01 indicates this is a new requester. This return code is used primarily by multiple requester terminal (MRT) applications. The same minor return codes supported with a 00 major return code are supported with the 01 major return code.

A major return code of 02 on System/36 indicates that a STOP SYSTEM or STOP SESSION has been entered, or that communications is ending. A major return code of 02 on the AS/400 system indicates that the job or subsystem is ending. For more information concerning error handling, see "Error-Handling Considerations" on page 19-8.

Testing Communications Applications

The following information will help you when testing your communications applications.

Testing Considerations: Consider the following when testing communications applications:

- **Error messages.** The system issues messages when errors are encountered while a program is running. These messages can be found in the job log, the QSYSOPR message queue, or the history log.
 - **Job log.** A job log is a record of each message the system sent because of a

job. Job log messages include the following:

- Commands you entered from the keyboard.
- Responses to your request.
- Messages about programs the system ran as a result of your request. They are not displayed at your work station.

Look in the job log for messages that note a problem. Serious problems usually produce a function check and escape messages. However, these messages are usually preceded by other messages that describe the problem in detail.

- **QSYSOPR message queue.** The QSYSOPR message queue contains the most recent information on system activity, including the following:
 - Record of device activation (VRYCFG)
 - Device failures
 - Program start request failures

The information available on the QSYSOPR message queue is a subset of the information available in the history log.

- **History log.** A history log is a complete record of system activity including:
 - A record of each job started and finished
 - A record of each device problem
 - A note about each damaged system object
 - Copies of each message sent to the system operator message queue

See the *CL Programming* book for information about displaying messages in the job log, the QSYSOPR message queue, and the history log.

- **Trace ICF.** On the AS/400 system, use the Trace Intersystem Communications Function (TRCICF) command to save information at the job level about the language operations and communications functions directed to an ICF file in the current job or in the job being serviced as a result of the Start Service Job (STRSRVJOB) command. See the *ICF Programming* book for information about accessing the Trace ICF environment.
- **Debug mode.** On the AS/400 system, use the Start Debug (STRDBG) CL command to

enter the debug mode. If you are servicing a job as a result of the STRSRVJOB CL command, then all debug commands affect the job being serviced. This allows you to make use of the associated commands and their functions to get more information about your program status and its variable contents. See the *CL Reference* book for information about accessing the debug mode environment.

The following commands are allowed only in the debug mode:

- With the Add Trace (ADDTRC) command you can record the sequence in which statements in your program are executed and indicate the data to trace.
- With the Display Trace Data (DSPTRCDTA) command you can display the data traced (as a result of the ADDTRC CL command) when you used the ADDTRC command.
- With the Add Breakpoint (ADDBKP) command you can stop a program when you want to look at the status of a variable. Enter the program statement number and the variable you are interested in with the ADDBKP command. When the program reaches this statement, it stops before executing the statement. When the program stops, the variable name you entered with the command appears.
- With the Display Program Variable (DSPPGMVAR) command you can display a variable not included in your ADDBKP command. DSPPGMVAR allows you to look at the variable content at the point where you stopped the program. There are cases when you want to look at the contents of your return code to verify that your operation completed normally. Looking at the contents of the return code allows you to track the progress of your communications transaction.

See the *CL Reference* book for information on how to use these commands in debug mode.

- **Job trace.** Use the Trace Job (TRCJOB) command to trace a job while it is running. You can use the TRCJOB command interactively where it is entered on the command

entry screen or where it is defined in the OCL procedure as another statement. With it you can track the sequence of called programs. Use the TRCJOB command when you are having trouble isolating the program causing the problem. Output is placed on the spooled file. If you specify an output queue (using the CHGJOB OUTQ(xxx) command), use the Work with Output Queue (WRKOUTQ) command. If you do not specify an output queue, use the Work with Job (WRKJOB) command (with option 4), or the Work with Spooled Files (WRKSPLF) CL commands to display the spool files for your job.

See the *CL Reference* book for more information on the Trace Job (TRCJOB) command.

Debugging the Source Program: When you are debugging your source program, the commands described in “Testing Considerations” on page 13-28 can assist you in isolating the source of the problem.

Debugging the Target Program: You can debug your target program by coding the commands described in “Testing Considerations” on page 13-28 in your remote procedure or by setting up a work station for your remote program that you can use interactively.

Normally, trace and debug mode commands are used interactively on your work station where you are running your program. However, when you are isolating a problem in a remote program you started with the EVOKE function in your local program, you must use trace and debug mode commands differently. Therefore, define the trace and debug commands in your remote procedure using the source entry utility (SEU). Your System/36 procedures have the file type OCL36. To prompt on AS/400 commands (ADDBKP, TRCJOB, and so on), change the file type of your procedure to CL. You can change the file type back to OCL36 if you want to. Like a batch program, the remote program does not have a work station associated with it.

You can declare an Add Breakpoint (ADDBKP) command in your remote procedure. The breakpoint information is passed to the breakpoint program. The breakpoint (CL) program is one of the following:

- Associated with another work station (to which it displays its information and from which it receives replies)
- Contains the commands you would have entered interactively

For example, the program can change and display variables, or add and remove breakpoints. When the breakpoint program ends, the program being debugged continues to run.

Additional Considerations: When a program start request received from the remote system fails, error messages that explain the cause of the problem are logged on the QSYSOPR message queue.

In addition to the messages, a reason code that points to a more specific cause of the trouble is provided. The reason code appears with the system message on the message queue for the job. See Figure 13-9 on page 13-11 for reason codes of errors on program start requests.

A **return code** indicates a problem with your source program or the target program. It is associated with an error message sent to your message queue, job log, or history log. Refer to the book that describes the communication type you are using for information about the return code you received.

When the source program receives a **negative response code** from the target program, a system message is sent to the job log. The online help information contains a code that identifies the specific problem. This can be a 2-, 4-, or 8-character SNA negative response code.

File Transfer Subroutines

The file transfer subroutines allow a user application program to retrieve System/36 data files or System/36 library members from a System/36 and send and retrieve database file members with System/36 data file or System/36 library member attributes to an AS/400 system or another System/36.

System/36 data files and System/36 library members reside on the AS/400 system as members in a database file. When a data file is

retrieved from System/36, the data file is stored on the AS/400 system as follows:

- Data files are stored in the System/36 environment files library (the default value is QS36F).
- The file name is the name of the file on System/36. The application program can also specify it.
- The member name is the System/36 creation date preceded by M. The application program can also specify it.

When you are sending a data file to System/36 or an AS/400 system with the System/36 environment support, the system uses the following rules to determine the source AS/400 database file and member:

- The library searched is QS36F.
- The file name specified in the application program determines the file name.
- The member name is the file date specified by the application program, preceded by M.

A library member retrieved from System/36 is stored on the AS/400 system as follows:

- Library members are stored in the library the application program specifies.
- The type of library member received determines the file name:
 - Procedures (PROC) are stored in QS36PRC.
 - Subroutines (SUBR) are stored in QS36SBR.
 - Source members (SOURCE) are stored in QS36SRC.
 - Load members (LOAD) are stored in QS36LOD.
- The member name is the System/36 library member name. (The application program can also specify it.)

When you are sending a library member to System/36 or an AS/400 system with System/36 environment support, the system uses the following rules to determine the AS/400 database file and member:

- The library searched is the library name specified by the program.
- The type of library member sent determines the file name:
 - QS36PRC is used if a procedure is being sent.

- QS36SBR is used if a subroutine is being sent.
- QS36SRC is used if a source member is being sent.
- QS36LOD is used if a load member is being sent.
- The member name is the library member specified in the program.

The file transfer subroutines allow communications using APPC, BSCCEL, and asynchronous communications support. **Asynchronous transmission** in data communication is a method of transmission in which sending and receiving of data is controlled by control characters instead of by a timing sequence.

Two file transfer subroutines exist in the System/36 environment:

- SUBRF1 for System/36-compatible COBOL programs
- SUBRF2 for RPG II programs

These two subroutines are compatible with the file transfer support subroutines supported on System/36. SUBRF1 and SUBRF2 are available in the System/36 environment.

File transfer support is also available to programs running in the OS/400 environment. For information about using file transfer in the OS/400 environment, see the *ICF Programming book*.

File Transfer Subroutine Parameters

The following parameters are passed to the subroutine:

FCODE

This 1-character field contains the file transfer function to be performed. This field is required. Following are valid values:

- | | |
|----------|--|
| S | Send a data file or library member to the remote system. |
| R | Retrieve a data file or library member from the remote system. |

QUAL1 through QUAL6

These six fields tell the system whether you want to transfer a data file or a library member. They also allow you to give the data file or library member a different name on the target system.

The following table shows the meanings of the qualifiers for data files.

Qualifier	Value	Description
QUAL1	File name	The name of the data file (1 to 8 characters long). This field is required. Do not specify a group file name.
QUAL2	File date	The 6-digit decimal field representing the creation date of the file. This field is optional. If you do not specify a file date and more than one file exists with the specified name, the most recent file is used. The format of the file date must be the same as your system date format. If you use this field, make sure the system date format on the remote system is the same as the format on your system.
QUAL3	Blank	This field must be left blank.
QUAL4	Target file name	The name of the data file (1 to 8 characters long) at the target system. This field is optional. <ul style="list-style-type: none"> • If you are replacing a file (REPL=Y), this is the name of the file to be replaced at the target system. • If you are adding a new file, this is the name to be assigned to the new file. • If you do not specify a target file name, the file name from QUAL1 is assumed.

Qualifier	Value	Description
QUAL5	Target file date	The creation date (6 decimal digits long) of the file at the target system. This field is optional. The format of the file date must be the same as your system date format. If you use this field, make sure the system date format on the remote system is the same as the format on your system. <ul style="list-style-type: none"> • If you are replacing a file (REPL=Y), this is the creation date of the file to be replaced at the target system. • If you are adding a new file, this is the date to be assigned to the new file. • If you do not specify a target file date, the file date from QUAL2 is assumed.
QUAL6	Blank	This field must be left blank.

The following table shows the meanings of the qualifiers for library members:

Qualifier	Value	Description
QUAL1	Library name	The name (1 to 8 characters long) of the library in which the member can be found. This field is required.
QUAL2	Library member type	Valid values are SOURCE, PROC (procedure), SUBR (subroutine) and LOAD (load). This field is required.
QUAL3	Library member name	The name of the library member (1 to 8 characters). This field is required.
QUAL4	Target library name	The name (1 to 8 characters long) of the library in which the member is to be stored. This field is optional. If you do not specify a target library name, the value from QUAL1 is assumed.
QUAL5	Blank	This field must be left blank.

Qualifier	Value	Description
QUAL6	Target library member name	<p>The name (1 to 8 characters long) of the library member at the target system. This field is optional.</p> <ul style="list-style-type: none"> If you are replacing a library member (REPL=Y), this is the name of the member to be replaced at the target system. If you are adding a new library member, this is the name to be assigned to it. If you do not specify a target library member name, the value from QUAL3 is assumed.

REPL

This 1-character field tells whether or not you want to replace the data file or library member on the target system. Valid values are as follows:

- Y** Replace an existing data file or library member on the target system.
- N** Do not replace an existing data file or library member; indicate an error condition to the user if the file or member already exists.

The default is N.

LOCNAM

This 8-character name contains the name of the remote location with which you are communicating. This should be the same as the remote location name specified in your configuration (on the RMTLOCNAME parameter of the Create Device xxx [CRTDEVxxx] command).

PWORD

This 4-character field contains the password for signing on the remote system. This field is required only if the remote system has password security active.

RCODE

This 1-character field contains the return code. The subroutine returns this value to the application program to indicate the result of the file transfer. Valid values are as follows:

- 0** Normal completion.
- 1** An error was detected at the local system.
- 2** An error was detected at the remote system.

For return codes 1 and 2, the specific error is logged to the job log file of the program message queue, and the message identifier code is returned to the user in the ERRMIC field. **Message identification** is a field in the display or printout of a message that directs the user to the description of the message in a message guide or a reference book. This field consists of up to four alphabetic characters followed by a hyphen, followed by the message identification code.

ERRMIC

If the value returned in the RCODE field is 1 or 2 (indicating an error), this 8-character field contains the MIC for the specific error.

Although file transfer subroutine message numbers returned to the application are similar to System/36 message numbers, AS/400 system numbers are different. See the *ICF Programming* book for information about file transfer messages.

APPN

This 1-character field is used on System/36 to indicate whether APPN can be used. The AS/400 system assumes the equivalent of APPCNET-YES.

File Transfer Support Considerations

Consider the following points when using file transfer support:

- The AS/400 system requires that all members within a file have the same record length. Files retrieved from System/36 can be stored in any file as long as the record lengths are the same. If the file name only is specified, the defaults of a send from System/36 apply.
- If System/36 sends a data file or library member to an AS/400 system, it can be retrieved from a System/36. If an AS/400 system retrieves a System/36 data file or library member, and the file name on the AS/400 system is not defaulted (QS36F,

QS36PRC, QS36SBR, or QS36SRC), System/36 cannot retrieve the file from the AS/400 system.

- If you try to transfer a null library member to System/36, the library member appears to transfer successfully. However, because System/36 does not support null library members, the library member does not exist on System/36.
- The mode specified in your APPC device description used with file transfer must equal the default mode in your network attributes or a mode of *NETATR.
- If your system is connected to a network by a packet assembler/disassembler (PAD), and you use FTS on an X.25 packet-switching data network (PSDN), you must set network-specific and X.3 parameters to the following values to allow data transparency:
 - No PAD recall using a character
 - No selection of data forwarding characters
 - Only forward on full packets or idle timer
 - PAD must allow 8-bit transparency
 - PAD must allow EBCDIC data
 - No echo
 - No use of XON/XOFF
 - On break signal from start-stop mode of DTE, PAD must send interrupt

Data terminal equipment (DTE) is that part of a data link that sends data, receives data, and provides the data communications control function according to protocols.

Notes:

1. If you use FTS with AS/400 system integrated PAD, X.3 parameters are ignored to achieve data transparency.
2. The network PAD must not perform operations on the file transfer data stream.

See the *Asynchronous Communications Programming* book for more information about X.3 parameters.

Asynchronous Communications

This section summarizes the differences in asynchronous communications support between System/36 and the System/36 environment:

- **Modem initialization support.** The modem initialization command is not supported. You must provide an application program to perform any modem initialization required.
- **Generic location support.** On System/36, you configure generics to accept incoming calls from remote systems whose location names are in the DEFINLOC file. When a remote system calls in, its name is assigned to the generic. At the end of the session, the System/36 program sends a disable command to disable the location assigned to the generic. The generic then goes back to generic status and waits for another incoming call. It does not have to be enabled again before it can be used again. Generic locations cannot be enabled and disabled while they are in generic status.

On the AS/400 system, generic controllers/devices must be enabled before incoming calls can be accepted. If an application disables (varies off) a generic controller/device, it must vary it on before it can be used to accept incoming calls. However, the generic controller/device need not be disabled to break the connection. There is a switched disconnect (SWTDSC) parameter on the controller description. The switched disconnect parameter indicates whether the asynchronous communications support should disconnect the connection at the end of the session. To have generic controllers/devices available for incoming calls after each session, configure SWTDSC(*YES) on the controller description. This configuration causes the connection to be dropped and the generic is again available for incoming calls.

Asynchronous Configuration Considerations

Consider the following points when configuring asynchronous communications:

- **Bits per character plus parity.** System/36 builds a data byte consisting of 8 bits. The

byte consists of 7 data bits plus a parity bit, or 8 data bits with no parity. The 9404 System Unit supports this 8-bit byte.

On a 9406 System Unit, you can set 7 or 8 data bits as well as even, odd, or no parity. Parity causes the system to add a parity bit to the number of data bits.

- **Intercharacter timeout.** Intercharacter timeout causes the I/O adapter to pass input data to the asynchronous communications support if no additional data has been received before the configured timeout value. AS/400 system support allows you to configure the intercharacter timeout value.
- **Remote location names.** On the AS/400 system, you can specify the remote location name *NONE on an asynchronous communications device description. You must specify remote location name (RMTLOCNAME(*NONE)) on the device description and connection number (CNNNBR(*ANY)) on the controller description to configure the generic location. However, you can specify *NONE if you want only programs started from a remote system and not acquired by a local program to use the device.

Asynchronous Programming Considerations

Consider the following when programming asynchronous communications:

- **Packet assembler/disassembler (PAD) operation.** The following modifications have been made to built-in PAD emulation support:
 - **Prompt and service signals.** The text for the prompt and service signals returned by PAD have changed. See the *Asynchronous Communications Programming* book for a list of the messages.
 - **Priority of PAD data.** An application program receives data echoed by the PAD prior to receiving data from the remote system. PAD returns service signals to the application prior to data.
 - **X.3 parameters.** AS/400 asynchronous communications does not support some X.3 parameters. See the *Asynchronous Communications Programming* book for a

list of the X.3 parameters the AS/400 system supports.

- **Ordered X.28 command responses.** Responses to X.28 commands resulting in a list of X.3 parameters and their corresponding values are sorted by parameter (in ascending order) and contain no duplicate parameters.
- **Data buffering and the use of XOFF.** When data arrives faster than a user application receives and processes it, the AS/400 system buffers the data until the application can accept it. Before sending an XOFF character to the remote system, 12K (K equals 1024 bytes) are buffered. The AS/400 system continues to send an XOFF character in response to each buffer received until the amount of data received by the application program reduces the amount of buffered data to less than 4K. When the amount of buffered data is less than 4K, the system sends an XON character. The AS/400 system buffers a total of 24K before it drops the connection with the remote system.
- **New return code 0042.** When the adapter has detected a data loss situation due to a buffer overrun, the loss is reported to the application (with data received prior to the overrun) as return code 0042.
- **Protocol identifier.** The first byte of the call-user data in an X.25 call packet is the protocol identifier. The System/36 set this byte to hex 00. The AS/400 system sets this byte to hex C0. X.25 support accepts both protocol identifiers as non-System Network Architecture (SNA).

The protocol identifier is set to hex 01 when you configure PAD emulation and initiate the call through the PAD.
- **Use of the more data flag.** Data transmitted over an X.25 line utilizes the *more data* flag if your record size exceeds the network packet size. Data received in packets with *more data* flagged are combined and treated as one record.
- **ITF procedure.** When using the ITF procedure, consider the following:
 - **Member send.** The *Remove sequence number and date* field is new. If you type Y in this field, the first 12 bytes are

removed from each record. If you type N in this field, the first 12 bytes from each record are *not* removed from each record.

- **Member receive.** The *Convert to source* field is new. If you type Y in this field, data that is received starts in the thirteenth position. Positions 1 through 12 are used for sequence number and date field information. If you type Y in this field, be sure the receive record size you specified is sufficient to hold the data received. If the receive record size is not large enough, the end of each record will be truncated. If you type N in this field, the data received will start in the first position.
- **Record size on Member Receive if the member does not exist.** On the System/36, the record size was 80, 96, or 120 bytes. On the AS/400 system, the record length is 80, 92, 96, 108, 120, or 132 bytes. The additional values are each 12 bytes more than the System/36 values due to the *Convert to source* field.
- **Deciding if the member exists on Member Receive.** First the member is checked to determine whether it is a source member. If the member is not a source member, it is checked to determine whether it is a procedure member. The System/36 checked to see if it was a source member and did not determine whether it was a procedure member.
- **SUBRA1 subroutine.** The SUBRA1 subroutine allows System/36-compatible COBOL and RPG II programs to retrieve the data length for an ICF file from the last input operation. The call to SUBRA1 must immediately follow the input operation. Following are the SUBRA1 parameters:

WSNAME

The character field that contains the name of the file assigned to the work station. It is required only for COBOL programs.

SYMID

The 2-character field for the session identifier.

DATAL

The 4-character field name that contains the length of data received from the last input operation.

RCODE

The 2-character code that contains the return code. SUBRA1 returns this value to the application program. Following are the valid values:

- 40** Normal completion
- 41** The specified session identifier not found or not associated with the last I/O operation to a work station file
- 42** Not an RPG II or System/36-compatible COBOL program

BSCCEL

This section summarizes the differences in BSCCEL between the System/36 and the System/36 environment on the AS/400 system. For additional information on AS/400 BSCCEL, see the *BSC Equivalence Link Programming* book.

BSCCEL Terminology Considerations

The following list describes System/36 BSCCEL terms that are used differently on the AS/400 system:

- On System/36, the PARTNER parameter specifies the type of session you want with the remote system. NORM and ATTR are the possible choices for this parameter. On the AS/400 system, the RMTBSCCEL parameter specifies the type of session you want with the remote system. The corresponding choices are *YES and *NO. (RMTBSCCEL is a parameter on the ADDICFDEVE, OVRICFDEVE, CHGICFDEVE, CRTDEVBSC, and CHGDEVBSC commands.)
- Many System/36 communications operations are referred to as functions on the AS/400 system.
- System/36 uses end-of-transaction as an operation modifier. The AS/400 system uses detach function to mean the same thing.

BSCEL Configuration Considerations

The following list describes some configuration considerations for BSCEL on the AS/400 system in the System/36 environment:

- On System/36, the default record separator is hexadecimal 00. If you use this default value and you specify blank compression or blank truncation, the system uses hexadecimal 1E as the record separator.

On the AS/400 system, the default record separator is hexadecimal 1E on the CRTDEVBSC command. If your AS/400 system is communicating with a System/36 that is using its default record separator, and you are using blank compression or blank truncation, you must use hexadecimal 1E as your record separator.

If the default record separators are not used, you must ensure that the record separator used on the AS/400 system matches the record separator on the System/36.

- On System/36, you were required to enter a local station address in the CNFIGICF line member when you configured multipoint tributary support. On the AS/400 system, you must enter a local station address in the line description and a local location address in the device description when you configure multipoint tributary support. A host system must use a 4-character POLL/SELECT address to communicate with BSCEL on the AS/400 system when using a multipoint line, instead of the 2-character POLL/SELECT address that was used to communicate with BSCEL on the System/36.
- On System/36, you were allowed to specify multiple remote IDs in the CNFIGICF subsystem member if you specified switched line support. The maximum number of remote IDs you could specify was 55 per System/36. On the AS/400 system, you can specify multiple remote IDs in the controller description if you specify switched line support. The maximum number of remote IDs you can specify is 64 per controller description on the AS/400 system.
- On System/36, you were allowed to specify ITB record blocking and transparency on

receive. This combination is not allowed on the AS/400 system.

- On System/36, you could override the switch type value specified in the line configuration member by specifying the SWTYP parameter on the SESSION OCL statement as either MC, AA, or MA (manual call, auto answer, or manual answer). On the AS/400 system, the SWTYP parameters (MC, AA, MA) are also supported on the SESSION OCL statement, but the BSC line description must also be created or changed with nonconflicting parameters (such as AUTODIAL = *NO if SWTYP = MC). Refer to the SESSION OCL statement in the *System/36 Environment Reference* book for more information.

BSCEL Programming Considerations

The following list describes the programming considerations for BSCEL on the AS/400 system in the System/36 environment:

- AS/400 BSCEL sends the end-of-transaction (*EOX) command when a detach function is used. This only applies when PARTNER-NORM is specified (on the SESSION OCL statement), or the session was started by an *EXEC program start request. On System/36, if you send data with the write operation and you are not using data blocking, BSCEL places the *EOX command at the beginning of the data record before the record is sent. The return code given to the receiving program is 0008. On the AS/400 system, if you send data with the write operation and you are not using data blocking, BSCEL sends the *EOX command in a separate record, after the data record is sent. The return code given to the receiving program for the data record is 0001. The receiving program has to use another input operation in order to detect the end-of-transaction indication. The return code given for this input operation will be 0308.
- The AS/400 system can send positive acknowledgements for received data before the data is checked for BSCEL commands. This can cause a release operation in an **acquired session** (a session started using an acquire operation with an OPEN statement) to time-out and fail if the target program has not

ended its session after the communications transaction has ended. (BSCSEL sends a *REL command to the remote system when a release operation is processed and PARTNER-NORM is specified on the SESSION OCL statement.)

- On System/36, BSCSEL sends online messages with the prefix SYS-. On the AS/400 system, BSCSEL sends online messages with a prefix of either BSCL or CPI. BSCSEL sends an online message (with reason codes) with the prefix BSCL in response to a program start request. BSCSEL sends an online message with the prefix CPI to indicate that a session error has occurred, or for any other error requiring a message. If your System/36 application which communicates with an AS/400 BSCSEL application is dependent on the format of SYS- messages, you must change your System/36 application to recognize the new formats.

- There are three levels of activation required by both System/36 and the AS/400 system. The System/36 ENABLE procedure attempts to establish communications on all three levels. The AS/400 user must vary on (using the VFYCFG command) each line, controller, and device to establish the same state of activation. Once this has completed, a session with a particular device may be initiated.
- The AS/400 system requires that the remote location name (configured in the device description) be unique for finance communications. If you use multiple logical work station IDs for the same location, or multiple subsystem configurations with the same remote location name, you must develop a naming scheme to ensure that your finance remote location names are unique. Once you have developed a naming scheme, you also must edit your finance procedures to ensure the SESSION OCL statements conform to the new naming scheme.

Finance Considerations

This section summarizes the configuration and programming considerations for finance communications. See the *Finance Communications Programming* book for a complete description of the AS/400 system finance support.

Note: The System/36 LOAD3601 diskette image transfer is not supported on the System/36 environment. However, the AS/400 system supports a similar function when the Send Finance Diskette Image (SNDFNCIMG) command is used.

Configuration Considerations

Consider the following when configuring the AS/400 Finance support:

- The System/36 user has to configure an SSP-ICF line member for each line used by System/36 Finance support. The System/36 user also has to configure an SSP-ICF subsystem member, specifying each control unit (remote locations) and characteristics of each control unit, including the number of terminals. The AS/400 system requires the definition of each line, all control units, and all devices connected to each control unit. These definitions are the line descriptions, controller descriptions, and device descriptions.

Programming Considerations

Consider the following when programming for AS/400 Finance support:

- System/36 applications communicate with System/36 Finance support through the SSP-ICF.
- On System/36, a \$\$\$SEND with data while the session is invited, results in an error return code. In the System/36 environment, this is no longer true. The data is sent, and the session remains invited.
- On System/36, an input operation results in an 8322 return code if a chain was started to the controller (\$\$SEND) on a system monitor session and never closed (\$\$SENDE). In the System/36 environment, the input operation results in finance support closing the chain.
- System/36 supports a maximum received data length of 512 bytes. The System/36 environment supports a maximum of 4096 bytes.
- Major return codes 80 and 81 may have different minor return codes in the System/36 environment.
- The AS/400 system buffers data being sent or received by your program for finance devices.

Retail Considerations

This section summarizes the configuration and programming considerations for retail communications. **Retail communications** is the data communications support that allows programs on an AS/400 system to communicate with programs on point-of-sale systems using SNA LU session type 0 protocol. See the *Retail Communications Programming* book for a complete description of AS/400 system retail support.

Configuration Considerations

Consider the following when configuring AS/400 Retail support:

- The System/36 user configures an SSP-ICF line member for each line used by System/36 Retail support. The System/36 user also configures an SSP-ICF subsystem member, specifying each control unit (remote locations) and characteristics of each control unit, including the number of terminals. The AS/400 system requires the definition of each line, all control units, and all devices connected to each control unit. These definitions are the line descriptions, controller descriptions, and device descriptions.
- The MAXMSG parameter is ignored in the System/36 environment. To specify the pacing value on a retail controller, you must configure the PACING keyword on the CRTDEVRTL command.
- There are three levels of activation required by System/36 and the AS/400 system. The System/36 ENABLE procedure establishes communications on all three levels. The AS/400 user must vary on (using the VFYCFG command) each line, controller, and device to establish the same state of activation. Once this is completed, a session with a particular device may be started.
- The AS/400 system requires that the remote location name (configured in the device description) be unique for retail communications. If you use multiple logical work station IDs for the same location, or multiple subsystem configurations with the same remote location name, you must develop a naming scheme to ensure that your retail remote

location names are unique. Once you have developed a naming scheme, you also must edit your retail procedures to ensure the SESSION OCL statements conform to the new naming scheme.

Programming Considerations

Consider the following when programming for the AS/400 Retail support:

- On System/36, a \$\$SEND with data while the session is invited results in an error return code. In the System/36 environment, this is no longer true. The data is sent, and the session remains invited.
- System/36 supports a maximum write operation (\$\$SEND or \$\$SENDE) of 256 bytes. The System/36 environment supports a maximum write operation of 4096 bytes.
- Major return codes 80 and 81 may have different minor return codes in the System/36 environment.
- The AS/400 system buffers data being sent or received by your program for retail devices.

Intrasystem Communications

This section summarizes the differences in support between the System/36 Intra subsystem and the AS/400 intrasystem communications type.

Programming Considerations

Consider the following when programming for the AS/400 intrasystem communications support:

- On the System/36, a \$\$SEND with data while the session is invited results in an error return code. In the System/36 environment, this is no longer true. The data is sent, and the session remains invited.
- System/36 supports a maximum write operation (\$\$SEND or \$\$SENDE) of 256 bytes. The System/36 environment supports a maximum write operation of 4096 bytes.
- The System/36 Intra sent system messages between programs indicating the state of the other programs. This is not supported on the AS/400 intrasystem communications.

- On the AS/400 intrasystem communications, the 831C return code is given following a 0412 return code if a second output operation is done. The 8323 return code never follows a 0412 return code. This may differ slightly from the System/36 Intra handling of output failure.
- On the System/36 Intra, Request Write (RQSWRT) is not allowed if BATCH(NO) is specified. On the AS/400 intrasystem, RQSWRT is allowed for both batch and non-batch modes, as are all other communications types.
- On the AS/400 intrasystem communications, CONFIRM is not ignored in the BATCH(YES) mode. If a detach/confirm function is specified, the system waits for a response to the confirm functions. This differs from the System/36 Intra.
- On the System/36 Intra, return codes may have been one or more operations behind the actual operation for which they were issued, since Intra buffers writes while sending. On the AS/400 system, only one write can be outstanding at a time, so the return codes are never more than one operation behind the actual operation.
- On System/36 Intra, an 0412 or 8323 return code may have been issued on an output condition if the other side canceled an invite and sent data first. On the AS/400 intrasystem communications, a 0412 is always used to indicate an output exception when data is in the buffer.

For information on mixing System/36 and AS/400 programs in the same application, see “Program Control in the System/36 Environment” on page 17-5.

System/36 APPC to AS/400 APPC

APPC support for the AS/400 system differs from System/36 APPC support in the way it handles sessions that send or receive a detach function.

The difference between these systems is the way sessions are released. On the System/36, APPC support holds the session until the user releases it by using the Release (REL) ICF operation. On the AS/400 system, APPC support allows the

session to be available for immediate use after the detach function is sent or received.

Because of this difference, an error can occur when an application program migrates from a System/36 with APPC support to an AS/400 system using APPC support. Most often this error occurs when the application program starts another evoke function, and the session to be acquired is being used by another job.

Both the AS/400 system and the System/36 allow another evoke function to run after the conversation ends. However, on the AS/400 system, the evoke function can fail if the switch line is disconnected or another job has acquired the session. The chances of the evoke function failing depends on the system usage at that time.

To prevent the AS/400 system evoke function from failing, consider the following:

- Use the Change Mode Description (CHGMODD) command to increase the number of sessions available. This only applies to multiple session devices with the single session (SNGSSN(*NO)) parameter specified on the APPC device description.
 - Change the APPC device from a single session device to a multiple session device to increase the number of available sessions. To do this, specify the SNGSSN(*NO) parameter on the APPC device description if the device specifies APPN(*NO). For an APPN(*YES) device, use the Change Configuration List (CHGCFGL) command to a specify single session (*NO) in the *Single session* field, and then delete the device description using the Delete Device Description (DLTDEVD) command.
- Note:** You must change the source and target system configurations in order to change from a single session device to a multiple session device.
- Use the Release (REL) ICF operation to release the session after you send or receive a detach function.

Before you try to use the evoke function again, you must acquire a new session. This causes the switch line to be dialed if disconnected. Therefore, if a session is not available, you receive the same return code on an AS/400 system as you get on a System/36.

- Use the WAITFILE parameter on the Create ICF File (CRTICFF) command to allow more time for a session to become available.
- Use the CHGDEVAPP or the CHGCFGL command to set the Number of conversations value to 1 when the SNGSSN(*YES) parameter is specified. This prevents another job from acquiring the session.

System/36 Peer to AS/400 Advanced Program-to-Program Communications (APPC)

You can change a System/36 program that uses the System/36 Peer subsystem so it can use the AS/400 ICF APPC support.

AS/400 APPC does not support the system-supplied send-end-of-chain (\$\$SENDE) format or the return code indicating reception of an end of chain.

An AS/400 APPC application can receive a return code indicating an end of transaction was received with the data record. When you are using the System/36 Peer subsystem, an end-of-transaction indication is not returned to the application with the data record.

On AS/400 APPC, you are not notified of an EVOKE failure until a subsequent operation. An evoke operation in the Peer subsystem does not complete until it receives acknowledgement from the remote Peer subsystem. In AS/400 APPC, a CONFIRM function can be included with the EVOKE function to verify completion of the EVOKE function.

Some minor codes returned by the System/36 Peer subsystem are different than those returned by AS/400 APPC. If the program checks minor return codes, changes are required to handle the AS/400 APPC minor return codes. For example, System/36 Peer subsystem return codes for failure errors (remote program issues a FAIL) are different than AS/400 APPC return codes for failure errors. See the *APPC Programming* book for a list of AS/400 APPC return codes.

See the *APPC Programming* book for information on security considerations.

System/36 BSC/CICS to AS/400 SNA Upline Facility

You can change a System/36 program that uses the System/36 BSC/CICS* subsystem so it can use the AS/400 SNUF support. Generally, only minimal changes are needed in the application program. Changes are also needed in the configuration and external interfaces (for example, lines and the remote host support).

Remote Host Support Considerations

To support a change to AS/400 SNUF, change the VTAM/NCP* generation to include an AS/400 system as an SNA device. Also change the CICS terminal definitions from BSC to SNA. Consult with your remote host system programmer for additional considerations.

SNUF Programming Considerations

The **SNA upline facility (SNUF)** is the communications support that allows the AS/400 system to communicate with Customer Information Control System for Virtual Storage (CICS/VS) application programs on a host system. You can run System/36 programs that use the BSC/CICS subsystem, with little change, on the AS/400 SNUF support if no minor return codes are checked. The minor return codes issued by System/36 BSC/CICS and AS/400 SNUF differ somewhat, especially in the way an end of a transaction is indicated. Therefore, coding changes are required if minor return codes are checked. There are also some differences in the generation of major return codes that indicate a permanent error (80 and 81). However, in general, these should not affect your application.

Data buffering on the AS/400 system is different than data buffering on System/36. This can affect when an application program is notified (by a major/minor return code) of an error condition. Some examples of this are:

- The timing of when an application receives notification of an SNA error varies from System/36 to the AS/400 system, based on various factors, such as frame sizes.

- On System/36, when an application issued a put operation, SSP-ICF data management sent the data to the SSP-ICF subsystem and then returned control to the application without waiting for the operation to complete. Therefore, if the subsystem determined that the operation was not valid based on the state of the session, the application was not notified until the next operation. On the AS/400 system, the application is notified as soon as possible of the state error that occurred.

Another difference between System/36 BSC/CICS and AS/400 SNUF is that BSC/CICS allows only the *EXEX program start request by CICS, whereas SNUF allows either *EXEX or *EXEC. This does not cause any changes unless it is decided to use the additional facility.

On the AS/400 system, uninvited data is data received in a session that has no active AS/400 program associated with it and the data does not begin with *EXEC or *EXEX. (Data records that begin with *EXEC or *EXEX are program start requests, and are not considered uninvited data.) An AS/400 program must be specified for program-start-capable devices (during configuration with the DFTPGM parameter of the CRTDEVSNUF command) so that the program begins running when uninvited data is received. When data is given to the program, the program can process the data, save it in a file, or discard it. If the program is using the AS/400 SNUF support, the program must receive 256-byte records and should be prepared to handle function management headers.

The System/36 BSC/CICS subsystem was always one record ahead of the user. A read for change direction or EOT was done in background processing so the line was free, in case the user did not get back to read the EOT. Some application programs may have taken advantage of this action, and issued an output operation, instead of a read, when the user knew that the data was complete. This function is supported for SNUF in the System/36 environment. However, if you choose to migrate your application to the AS/400 environment, this support is not available. The user program must continue to read until change direction is indicated.

System/36 BSC/IMS to AS/400 SNA Upline Facility

You can change a System/36 program that uses the System/36 BSC/IMS subsystem so it can use the AS/400 SNUF support. Changes are needed in the application program, in configuration, and external interfaces (for example, lines and remote host support).

Remote Host Support Considerations

To support a change to AS/400 SNUF, change the VTAM/NCP generation to include an AS/400 system as an SNA device. Also change the IMS terminal definitions from BSC to SNA. Consult with your remote host system programmer for additional considerations.

SNUF Programming Considerations

Consider the following items when writing interactive communications programs that communicate with IMS/VS:

- Uninvited data
- Detach (end of transaction) functions
- BATCH parameter on the SESSION statement
- Return code processing
- Change direction/EOT processing

You must consider these items whether you use AS/400 SNUF or the System/36 BSC/IMS subsystem. The following sections compare the AS/400 SNUF and the System/36 BSC/IMS subsystem's implementations with respect to these items.

Uninvited Data: On the AS/400 system and System/36, uninvited data is data (or messages) received in a session that has no active AS/400 program associated with it and the data does not begin with *EXEC or *EXEX. (Data records that begin with *EXEC or *EXEX are program start requests and are not considered uninvited data.) An AS/400 program must be specified for program-start-capable devices (during configuration with the DFTPGM parameter of the CRTDEVSNUF command), so that the program begins running when uninvited data is received.

When data is given to the program, the program can process the data, save it in a file, or discard it.

Uninvited data on an AS/400 system or System/36 can be received for any of the following reasons:

- A message switch or broadcast message is received.
- An IMS/VS application sends data (any data not beginning with *EXEC or *EXEX) to an output queue associated with a program-start-capable device.
- An AS/400 or System/36 program abnormally ends before all data has been sent by IMS.
- An IMS/VS status message is sent to a session.

Because IMS/VS cannot accept input from the same session used to start a remotely-started (target) program, the AS/400 or System/36 program that receives the uninvited data cannot perform any output operations in the session. If the program is using the AS/400 SNUF support, the program must receive 256-byte records and should be prepared to handle function management headers.

Detach (End of Transaction)

Functions: Detach functions (for example, write with detach or evoke with detach) are treated differently by the System/36 BSC/IMS subsystem than they are by AS/400 SNUF.

When an application issues a detach function, the System/36 BSC/IMS subsystem indicates to the host that the message has ended.

When a detach function is issued by an AS/400 SNUF application, SNUF sends an end bracket (if the secondary is allowed to send end brackets), or a change direction indication. If a change direction indication is sent, SNUF expects to receive a null record from IMS/VS that terminates the transaction. If anything other than a null record is received, the session is terminated abnormally, and an error return code is sent to the application program. Something other than a null record can be received in any of the following cases:

- The IMS/VS remote program generates output to be sent to this session.
- An output message is placed on the IMS/VS output queue by an IMS/VS program other than the one evoked by the AS/400 program.

- A message switch or broadcast message is placed on the IMS/VS output queue for this session.

A detach function used with AS/400 SNUF fails when output remains on the IMS/VS output queue and the secondary is not allowed to send an end bracket for this session. (The detach works if the secondary can send an end bracket.) The detach will not fail when using the System/36 BSC/IMS subsystem.

Because of these situations, detach functions are not recommended for AS/400 SNUF sessions that communicate with IMS/VS unless a logmode entry that allows the secondary to send end brackets is used.

BATCH Parameter on the SESSION

Statement: You must consider the differences in BATCH processing when converting the System/36 BSC/IMS SESSION OCL to the AS/400 SNUF SESSION OCL supported in the System/36 environment. The BATCH parameter on the SESSION statement determines how the input and output operations are handled. The following guidelines should be used in determining the appropriate BATCH parameter to use if your application is operating in forced terminal response mode:

- If more than one output operation is performed consecutively, the BATCH-YES parameter must be specified on the SESSION statement.
- If an input operation is used (one that is not combined with an evoke or output operation), the BATCH-YES parameter must be specified on the SESSION statement.

System/36 BSC/IMS: When BATCH-NO is specified on the SESSION statement, the System/36 BSC/IMS subsystem accumulates records as they are received from the System/36 program. The records are then sent as one message when the program indicates the message is complete (the program issues either an end-of-transaction operation or an input operation). The total length of all the records submitted within a message cannot be greater than the specified maximum record length.

When BATCH-YES is specified on the SESSION statement, the System/36 BSC/IMS subsystem sends each record as it is received from the

System/36 program. The subsystem sends each record as a segment of a message until it receives an end-of-transaction operation or an input operation from the program. A session specified as BATCH-YES cannot be acquired while another session is active because of the delays in line turnaround that may occur. BATCH-YES should be specified when large amounts of data must be sent without intervening responses from the host system.

AS/400 SNUF: When BATCH-NO is specified on the SESSION statement, SNUF handles each record from the application program as a complete chain. If you expect to receive data from an IMS/VS application, and you are operating in forced terminal response mode, IMS/VS requires that a change direction indication must also be sent when a chain ends. This means that the application program cannot issue multiple output operations consecutively. To assure that the change direction indication is sent, each output or evoke function must be accompanied by an invite or input operation (such as a write with invite). A modifier of detach also causes a change direction if the secondary is not allowed to send an end-bracket indicator. Note that a standard evoke function followed by either an invite or input operation is not acceptable, because the change direction indication will not accompany the evoke, and for that reason IMS/VS will reject it.

When BATCH-YES is specified on the SESSION statement, and you are operating in forced terminal response mode, SNUF sends each record from the application program as an element of a chain. Sending records in this manner allows the program to perform consecutive output operations. However, if you expect to receive data from an IMS/VS application, and you are operating in the forced terminal response mode, a change of direction must still be indicated at the end of each chain. Therefore, do not terminate a chain with an end-of-group function. The last output operation should include an invite, input, or detach modifier, or the output operation should be followed by an invite or input operation.

Return Codes: Return codes for permanent errors (major return code 80, 81, or 82 or *STATUS values greater than 99) returned by AS/400 SNUF differ from those returned by the System/36 BSC/IMS subsystem. Data buffering on the AS/400 system is different than data buffering on System/36. This can affect when an application program is notified (by a major/minor return code) of an error condition. Some examples of this are:

- The timing of when an application will get notification of an SNA error will vary from System/36 to the AS/400 system, based on various factors, such as frame sizes.
- On the System/36 when an application issued a put operation, SSP-ICF data management sent the data to the SSP-ICF subsystem and then returned control to the application without waiting for the operation to complete. Therefore, if the subsystem determined that the operation was not valid based on the state of the session, the application was not notified until the next operation. On the AS/400 system, the application is notified as soon as possible of the state error that occurred.

Change Direction/EOT Processing: The System/36 BSC/IMS subsystem was always one record ahead of the user. A read for change direction or EOT was done in background processing so the line was free, in case the user did not get back to read the EOT. Some application programs may have taken advantage of this action, and issued an output operation, instead of a read, when the user knew that the data was complete. However, if you choose to migrate your application to the AS/400 environment, this support is not available. The user program must continue to read until change direction or EOT is indicated.

Using CL Override Commands

The Override Intersystem Communications Function File (OVRICFF) command may be used to override the name or attributes of the file specified in a System/36 application program. If you specify OVRICFF in a System/36 procedure or from the System/36 Command Entry display in a System/36 environment job, it affects that job step and succeeding job steps in that System/36 job

until you specify a Delete Override (DLTOVR) command. If you specify OVRICFF from a CL program called by a System/36 procedure, it affects only the CL program running.

See “Mapping SESSION OCL Statement to the OVRICFDEVE Command” on page 13-18 for a description of the OVRICFDEVE command. See the *ICF Programming* book for more information about the OVRICFF and OVRICFDEVE commands.

General Programming Considerations

Consider the following points for all communications applications running in the AS/400 System/36 environment:

- System/36 allows a maximum of 360 active sessions at a time. The same restriction applies for the System/36 environment on the AS/400 system.
- An end-of-session (EOS) operation always completes successfully (with a '0000' return code) for applications running in the System/36 environment. An EOS is successful for a job running in the AS/400 environment if a session exists, but it fails (with an '830B' return code) if the session does not exist.
- Data buffering on the AS/400 system is different than data buffering on System/36. This can affect when an application program is notified (by a major/minor return code) of an error condition.
- A maximum of 4075 bytes (4096 for APPC and intrasystem communications) is supported on an input or output operation on System/36. The maximum has increased for each AS/400 communications type. You can send a larger value if you change the maximum record length on the system-supplied QICDMF file using the MAXRCDLEN parameter on the Change Intersystem Communications Function File (CHGICFF) command. However, the maximum value specified must not exceed the maximum supported by the other system. If the remote program is an AS/400 program running in a System/36 environment, the maximum record length also must be changed on the QICDMF file on that system.

- Although shared opens are a commonly used technique for performance improvements of databases in the System/36 environment, this technique is not recommended for performance improvements of communications in the System/36 environment. This is because when a System/36 application closes an ICF file, all acquired sessions are released regardless of whether the file is shared.
- On the AS/400 system, the \$\$TIMER value accepts only valid zoned decimal digits.

Migration Considerations

Consider the following information when migrating from System/36 to the AS/400 system.

Automatic Dial and Telephone Number List Support

Unlike autodial on System/36, the AS/400 system does not support a telephone list. Instead, the telephone number is stored in the controller description and the AS/400 automatically dials the telephone number when the first I/O operation is issued from an application using the configuration. See the *Communications Configuration* book for information about configuring the line and controller descriptions for autodial. See the *ICF Programming* book for information about associating the application with the configuration.

To automatically dial more than one location or telephone number on the AS/400 system, change the connection number (C>NNNBR) in the controller description in one of the following ways:

- Do the following for each telephone number to be dialed:
 - Use the CHGCTLxxx (such as CHGCTLASC, CHGCTLAPPC) command to change the telephone number (C>NNNBR) of the controller description your communications session uses.
 - Run your communications program.
- Write a program to process a list of telephone numbers from a database file. For each phone number:
 1. Call a CL program that uses the CHGCTLxxx command to change the C>NNNBR of the controller description.

2. Call the associated communications application that uses the controller description.

Note: The foregoing example is best used for asynchronous (AS/400) and BSCEL communications. The AS/400 system has the same restriction the System/36 telephone list support has for SNA communications (APPC, SNUF, Retail, and Finance). That restriction is that you should not change the telephone number in the controller description because you cannot change the XID field to get you to a different remote controller. Therefore, for AS/400 SNA applications doing automatic dial to more than one remote controller, create a separate description for each remote controller and write the application to process the separate configurations.

If automatic dial is unsuccessful because the remote system's telephone is busy or not answered, a CPA5712 inquiry message is sent to the system message queue, QSYSOPR.

Note: For X.21 lines, inquiry messages CPA57B1, CPA57B8, or CPA57B9 may be sent. Use the Display Message Description (DSPMSGD) CL command to display more information about these messages. You can respond to the message manually through the console, or you can specify automatic response to the message. There are two methods of specifying an automatic response:

- Because C is the default response to the CPA5712 message, you can place the entire QSYSOPR queue in default (*DFT) mode (CHGMS6Q MSGQ(QSYSOPR) DLVRY(*DFT)).
- You can use the selected message reply function, which is the better way to handle the message. This does not require that the entire message queue be placed in the default mode, and also allows you to respond automatically to messages based on the controller description name. Use the Add Reply List Entry (ADDRPYLE) command to add an entry to the reply list and specify the following:

Comparison Data	Controller Description Name
Message data start	16
Reply	C
Dump	*NO

On the open or acquire operation of your program, check the feedback area for an 82 major code returned for the autodial failure.

X.21

The AS/400 system does not support binary synchronous communications (BSC) on an X.21 facility or the dynamic facility registration function REQUESTX. The AS/400 system supports the short-hold mode of operation with the following enhancements:

- APPC
- Multiple port sharing
- Finance controller communications

System/36 environment applications running over an X.21 network may take advantage of these enhancements by configuring appropriately.

System Network Architecture Distribution Services (SNADS)

System/36 SNADS is most familiar to users of the Personal Services/36 product and ODF/36 PRPQ. **Programming Request for Price Quotation (PRPQ)** is a customer request for a price quotation for a licensed program to be designed especially for a particular group of customers or an application. Documentation for the program is provided only to those customers who order the PRPQ. For the AS/400 system, SNADS is packaged as part of the base operating system and supports the System/36 environment. See the *SNA Distribution Services* book for information about the AS/400 SNADS function and configuration.

The following table lists the Personal Services/36 procedures often used by System/36 SNADS users with their equivalents for the System/36 environment.

Personal Services/36	System/36 Environment
OFCCANCL procedure	Use ENDSBS QSNADS.
OFCCOMM procedure	Use CFGDSTSRV for OFCCOMM QUEUES and OFCCOMM ROUTES equivalents.
OFCDIR procedure	Use WRKDIR command.

Personal Services/36	System/36 Environment
OFCMAINT procedure	Use CFGDSTSRV for OFCMAINT QUEUES and OFCMAINT ROUTES.
OFCQ procedure	Use WRKDSTQ for OFCQ COMM.
OFSTART procedure	Use STRSBS QSNADS.

Note: For AS/400 SNADS, System/36 communications queues are called distribution queues.

Object Distribution

The following table lists the equivalent System/36 environment commands for the frequently used Object Distribution Facility/36 procedures.

Object Distribution Facility/36 Procedures	Equivalent System/36 Environment Commands
LISTOBJ	WRKNETF
LISTRSCS	WRKDSTQ
ODF	GO CMDNETF
ODFCANCL	ENDSBS QSNADS
ODFDFLT	GO CMDNET
ODFPROF	GO CMDNET
OFSTART	STRSBS QSNADS
RECVFILE	RCVNETF1
RECVFLDR	RCVNETF1
RECVLIBR	RCVNETF1
RECVPRT	RCVNETF1
SENDFILE	SNDNETF1
SENDFLDR	SNDNETF1
SENDJOB	SBMNETJOB
SENDLIBR	SNDNETF1
SENDPRT	SNDNETSPLF

¹ Some send and receive operations require that the objects be saved or restored as a step in the send or receive process. The following sections provide more information.

Sending AS/400 Objects

To send AS/400 objects, follow these steps:

1. Use the Save Object (SAVOBJ) or Save Library (SAVLIB) commands to save the objects to be sent in a save file.
2. Use the Send Network File (SNDNETF) command to send the save file to the remote system.

For example, to send folder FOLDER1 in library LIBRARY1 to USER2 SYSTEM2, first enter the following to save the objects to be sent in a save file:

```
SAVOBJ OBJ(FOLDER1) LIB(LIBRARY1) DEV(*SAVF)
SAVF(LIBRARY1/SAVEFILE1) DTACPR(*YES)
```

Note: The name of the save file to contain the folder is not significant. The save file must already exist.

To send the save file to the remote system, enter the following:

```
SNDNETF FILE(LIBRARY1/SAVEFILE1) TOUSRID((USER2 SYSTEM2))
```

Receiving Objects as AS/400 Objects

To receive objects as AS/400 objects, follow these steps:

1. Use the Work with Network File (WRKNETF) or Receive Network File (RCVNETF) commands to receive the save file on the remote system.
2. Use the Restore Object (RSTOBJ) or Restore Library (RSTLIB) commands to restore the objects on the remote system.

For example, to receive FOLDER1 on the remote system, enter the following:

```
RCVNETF FROMFILE(SAVEFILE1) TOFILE(LIBRARY2/SAVEFILE2)
```

Note: The name of the save file to receive is not significant. The save file must already exist.

To restore the objects on the remote system, enter the following:

```
RSTOBJ OBJ(FOLDER1) SAVLIB(LIBRARY1) DEV(*SAVF)
SAVF(LIBRARY2/SAVEFILE2)
```

Sending AS/400 System/36 Environment Objects

To send AS/400 System/36 environment objects, follow these steps:

1. Use the Save System/36 File (SAVS36F) or Save System/36 Library Member (SAVS36LIBM) commands to save the AS/400 System/36 environment objects in an AS/400 physical file.

2. Use the Send Network File (SNDNETF) command to send the physical file to the remote system.

For example, to send System/36 environment file S36FILE1 to USER2 SYSTEM2, first enter the following to save the AS/400 System/36 environment objects in an AS/400 physical file:

```
SAVS36F FROMFILE(S36FILE1) FROMLIB(QS36F) DEV(*PHYFILE)
PHYFILE(LIBRARY1/PHYFILE1)
```

Note: The name of the physical file to contain the System/36 environment file is not significant. The record length of the physical file must be 256 bytes. If the physical file does not exist, it is created.

To send the physical file to the remote system, enter the following:

```
SNDNETF FILE(LIBRARY1/PHYFILE1) TOUSRID(USER2 SYSTEM2)
```

Note: If you try to transfer a null library member to a System/36, the library member appears to transfer successfully. However, because System/36 does not support null library members, the library member will not exist on the System/36.

Receiving Objects in the AS/400 System/36 Environment

To receive objects in the AS/400 System/36 environment, follow these steps:

1. Use the Work with Network File (WRKNETF) or the Receive Network File (RCVNETF) com-

mands to receive the network file in a physical file.

2. Use the Restore System/36 File (RSTS36F), Restore System/36 Library Member (RSTS36LIBM), or Restore System/36 Folder (RSTS36FLR) commands to restore the objects in the AS/400 System/36 environment.

For example, to receive file FILE1 in the AS/400 System/36 environment, first enter the following to receive the network file in a physical file:

```
RCVNETF FROMFILE(LIBRARY1) FROMMBR(PHYFILE1)
TOFILE(LIBRARY2/PHYFILE2)
```

Note: The name of the physical file to receive is not significant. The record length of the physical file must be 256 bytes. The physical file must already exist.

To restore the objects in the AS/400 System/36 environment, enter the following:

```
RSTS36F TOFILE(S36FILE1) TOLIB(QS36F) DEV(*PHYFILE)
PHYFILE(LIBRARY2/PHYFILE2)
```

See the *SNA Distribution Services* book for information on the AS/400 System/36 environment commands. See the *Using the Object Distribution Facility/36 PRPQ* book for information on exchanging objects between the AS/400 system and System/36.

Chapter 14. Menus and Displays

This chapter describes:

- How to design and create menus and displays
- The differences between System/36 user-defined menus and displays and those you use in the System/36 environment

Menus

A menu is a displayed list of options. Each option has a number and a brief description of a task. When the user types the number for a menu option, the system runs the task associated with that option number.

The following screen is an example of the menu INVINF. A warehouse manager uses it to see information in the files used by the inventory management application.

```

COMMAND                                MENU: INVINF                            W1
Inventory Management: File Information Menu

Select one of the following:
1. Display item master
2. Display item balance detail (warehouse)
3. Display item balance detail (manufacturing)
4. Display open orders
5. Display item availability
6. Display item balance history

Help key - Display help information for this menu and its options
F12 - Cancel
Home key - Display sign-on menu
    
```

The menu shows:

- The name of the menu
- A descriptive title of the menu
- The option numbers
- A brief description of what each option does
- Prompts describing other functions that can be done using this menu

Menus simplify the work of an application user and reduce errors. The user does not need to know operation control language (OCL) statements, procedures, or control commands needed to run a job.

You can use menus to group jobs by application. For example, all accounts receivable jobs are

listed on one menu to allow users to run several related jobs consecutively.

Notes:

1. See the *ADTS/400: Screen Design Aid for the System/36 Environment* for more information about using menus.
2. Normally, the Exit (F3) and Cancel (F12) keys return you to the previous application. If you accessed the current application by running the Start System/36 (STRS36) CL command, you must use the End System/36 (ENDS36) CL command to return to the previous application.

Function Key Differences

Some of the function keys you use to work from menus in the System/36 environment are different than on System/36.

System/36 Environment Function Key

Processing for Menus: The following table compares various System/36 function keys to their corresponding actions in a System/36 environment.

Action	System/36 Environment Result on AS/400 System
HELP <Procedure or Menu Name>	<ul style="list-style-type: none"> • If you specify a System/36 environment procedure name, the \$HELP prompt for the procedure appears. • If you specify a System/36 system menu name, the equivalent AS/400 menu appears.
// HELP <Procedure or Menu Name>	<ul style="list-style-type: none"> • If you specify a System/36 environment procedure name, the \$HELP prompt for the procedure appears. • If you specify a System/36 environment system menu name, the equivalent AS/400 menu appears when the outermost procedure ends.
<CL command> <Help-key>	Displays information about the specified system CL command.

Action	System/36 Environment Result on AS/400 System
<Option-number> <Help-key>	Displays menu help for the specified option number if supplied in the menu definition. See "Creating and Displaying Online Help Information for Menus" on page 14-7.
<Procedure or CL command> Cmd 4	<ul style="list-style-type: none"> If you specify a System/36 environment procedure name, the \$HELP prompt for the procedure appears. If you specify an AS/400 CL command, the prompt for the CL command appears.
Cmd 3	Exit.
Cmd 9	Retrieve previous requests.
Cmd 12	Cancel.
Cmd 13	Display user support menu.
Cmd 16	Display system main menu.
Cmd 23	Change menu name in user profile.
Home key	Display <i>Home</i> menu.
Dup key	Retrieve previous requests.

Note: The HELP procedure supports the System/36 tutorials PCE (list of procedure control expressions) and OCL (list of OCL statements and parameters).

Differences from System/36: Following are the key processing differences in the System/36 environment:

- Online help information for System/36 environment operator control commands is not supported. Refer to the *System/36 Environment Reference* book for information on these commands.
- The Cmd 1 function to resume an interrupted job is not supported. Use the SIGNOFF CL command or the OFF operator control command.
- The Cmd 3 function to display the previous menu is now F12.
- The Cmd 5 function to display the system menu is now F16.
- The Cmd 6 function to display the user beginning help menu is not supported. Use the

MENU operator control command to see a specific menu.

- The Cmd 7 function to end the displaying of system menus is not supported. Use F12 to see the previous menu.
- The Cmd 11 function to display menu actions is not supported. This function is not replaced.
- The Cmd 12 function to display user support and education information is now F13.
- The Cmd 23 function to change the user menu name in the user profile is supported. A single name replaces the user and help menu names.
- The Cmd 24 function to change the system menu name is now F23. A single name replaces the user and help menu names.
- Use Cmd 9, instead of the DUP key, to display the last command you ran from the command line, and any parameters you selected. If you press this key once, you see the last command you ran. If you press this key twice, you see the next-to-last command you ran, and so on.

User Menus

The user can display a menu by:

- Typing the name of a menu in the menu field on the Sign-On display
- Leaving the menu field blank during sign-on to display the first menu defined in the user profile
- Requesting a menu from the main system menu
- Selecting a menu from another menu
- Running a procedure that displays a menu
- Using the MENU control command

The user can respond to a menu by:

- Typing an option number
- Typing a control command, a procedure, or OCL statement
- Pressing the Help key to request online help information for the menu or its options
- Pressing the Home key to return to the menu named during sign-on or to return to the default menu
- Pressing F12 to return to the previous menu
- Pressing F13 to display the system help menu

The command lines on the AS/400 system appear as the bottom three lines of user menus.

User Menu Differences: Following are the differences between displaying user menus in the System/36 environment and on System/36:

- The system-defined command line covers the user command line.
- IBM-supplied menus do not supply the display station ID in the upper right corner.
- The 0 option to display the command display is not supported. Request the System/36 environment command entry display by typing an asterisk (*). Request the AS/400 command entry display by typing CALL QCMD.
- System/36 supports user and system menus. The AS/400 system does not distinguish between user menus and IBM-supplied system menus. User profiles support a single menu value only.
- You can use the MENU operator control command to display AS/400 menus. The MENU command accepts a 10-character menu name.

Help Menu Differences: The support for displaying system menus in the System/36 environment maps a System/36 help menu to an AS/400 menu.

Type the following to use the HELP procedure to display a menu:

HELP menu-name

Note: The System/36 help menu MANDMENU for mandatory menu users is not supported.

Menu Option Logging

When the System/36 environment is not active, statements that result from selecting a menu option are not logged to the job log. When the System/36 environment is active, statements that result from selecting a menu option are logged to the job log, except in the following cases:

1. Option from system menus that are handled by the menu driver directly are not logged. For example, an option to go to another menu would not be logged.

2. If a menu is displayed from within a System/36 environment job (for example, by using a GO command in a procedure), menu options from that menu are not logged. Because the System/36 environment is already active, it is not called again to process statements from that menu, so the menu options are not logged. Menu options will be logged again when the procedure ends.

Menu Security

Use the system-supplied menu security to limit the jobs a user can run. You can assign the user's default menu as mandatory. A mandatory default menu restricts the user to making selections from that menu or from a menu displayed by an option on the default menu, and to using a few control commands. The user is limited to doing those jobs controlled by the mandatory menu.

For more information about menu security, default menus, and mandatory menus, see the *Security – Reference* book.

Menu Formats

You can create two types of menus:

- Fixed-form
- Free-form

You must display menus in 24 x 80 format *only* (24 lines of 80 characters). If you specify 27 x 132 for a menu format (for a 3180 or Monochrome 3197 Display), the menu may not display correctly.

Fixed-Form Menus: A fixed-form menu contains two columns of menu option numbers, 1 through 24, with 12 options in each column. When you create a fixed-form menu, you do the following:

1. Tell the system the name of the menu and which option numbers to use.
2. Give the descriptive text for each option.
3. Supply the procedure to be run for each option.

In the following screen, the programmer supplied the name of the menu, and the text and procedures for option numbers 1 through 9 only.

```

COMMAND                                MENU: INVFIX                            W1
Select one of the following:
1. Print stock status                  13.
2. Print open order status             14.
3. Print financial stock analysis      15.
4. Print stock movement analysis      16.
5. Print reorder report                17.
6. Print control totals                18.
7. Print item price list               19.
8. Print item balance list             20.
9. Print item master list              21.
10.                                    22.
11.                                    23.
12.                                    24.

Ready for option number or command

```

```

COMMAND                                MENU: INVFRE                            W1
Inventory Management: Reports Menu
Select one of the following:
1. Print stock status
2. Print open order status
3. Print financial stock analysis
4. Print stock movement analysis
5. Print reorder report
6. Print control tables
7. Print item price list
8. Print item balance list
9. Print item master list

Help key - Display additional information about this menu and its options
Home key - Display Home
F12 - Cancel

Ready for option number or command

```

Although all 24 option numbers appear on a fixed-form menu, the user can select only the option numbers that have procedures or commands defined for them.

Free-Form Menus: A **free-form menu** is a menu for which the programmer defines the format of lines 3 through 20. A free-form menu contains only the option numbers you want to use. You determine how the menu appears. A large portion of the display is available to supply descriptive text.

When you create a free-form menu you can:

- Define up to 24 option numbers and descriptions.
- Define the procedure, control commands or OCL statements associated with each option number.
- Decide the placement of the option numbers and their descriptions, and all other text.

The following screen is an example of a free-form menu. The options on this menu are the same as the options on the fixed-form menu shown in the last screen, but the programmer provides the numbers and their descriptions and supplies a descriptive title and information about online help information and function keys.

Designing Menus

A well-organized and descriptive menu increases user productivity. Consider the following when you design menus:

- The menu's use
- The types of jobs you want the menu to control
- The level of experience or responsibility of the user

Use the following guidelines when you design your menus:

- Use free-form menus as much as possible. Free-form menus do not have unused option numbers.
- Avoid mixing free-form and fixed-form menus within the same application.
- Results cannot be predicted if you do the following:
 - Specify the 27 x 132 character attribute for menus.
 - Put any fields with user-supplied data on a menu.
 - Move the input field on a menu.
- Write your menus in uppercase and lowercase letters. Text written only in uppercase letters is difficult to read.
- Number your options beginning with 1.
- Place frequently selected options near the top of the menu, place the options in the sequence they are to be selected, or arrange the options alphabetically.
- Make the menu title and option descriptions meaningful and descriptive. For example, *Release orders* is a more meaningful option

description than *Reload*, which might be the name of the program that releases orders.

- Use a word that suggests action, such as list or print, for the first word in the option description.
- If the same task (sign-off, for example) is done on several menus, use the same option number for that task on each of the menus.
- Avoid using abbreviations.
- Supply online help information for menus.

Sample menu ORDENT, shown in Figure 14-1 on page 14-5, is the main menu for an order entry and invoicing application. It shows good design techniques.

Menu chaining helps organize a user's work by guiding the user to the displays needed to do a particular job. Menu chaining uses a main menu that lists other menus from which a user can select a job. For example, menu ORDENT, shown in Figure 14-1 on page 14-5, is the main menu for an order entry and invoicing application.

Figure 14-2 on page 14-6 shows the options on ORDENT and the menus chained to ORDENT.

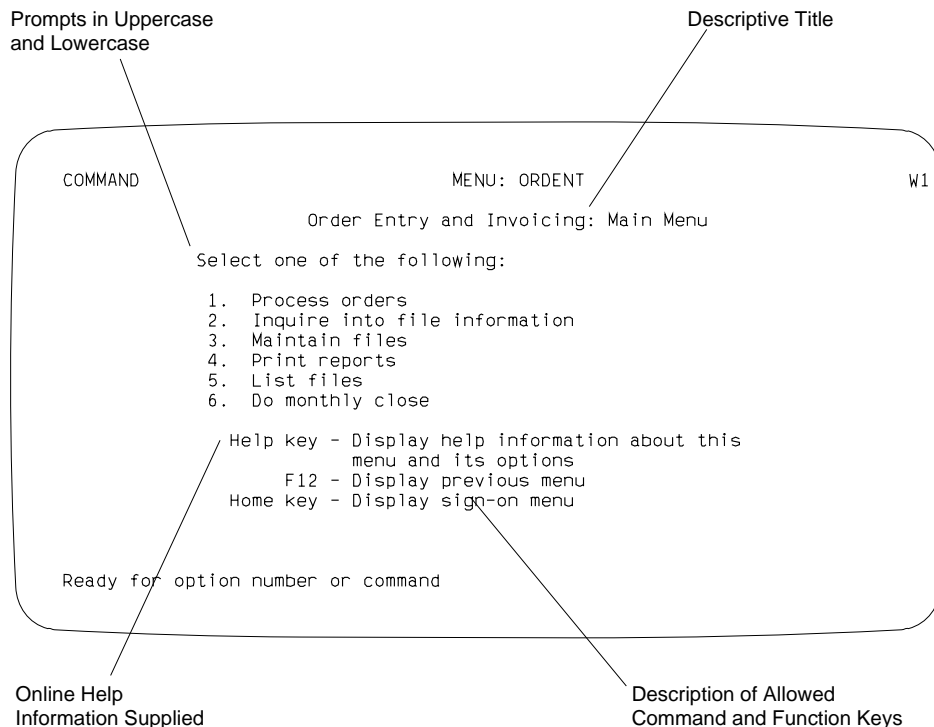
When a user selects option 1 from the main menu, the MENU control command is run and the Orders Menu appears. When the user selects an option from the Orders Menu, either another menu appears, if there are additional order processing categories to select, or a display appears, on which the user can begin a job.

The MENU OCL statement and the MENU control command are useful when you build a menu chain. The *System/36 Environment Reference* book has more information about the MENU OCL statement and the MENU command.

When you chain menus, you can allow the user to display the main menu again by:

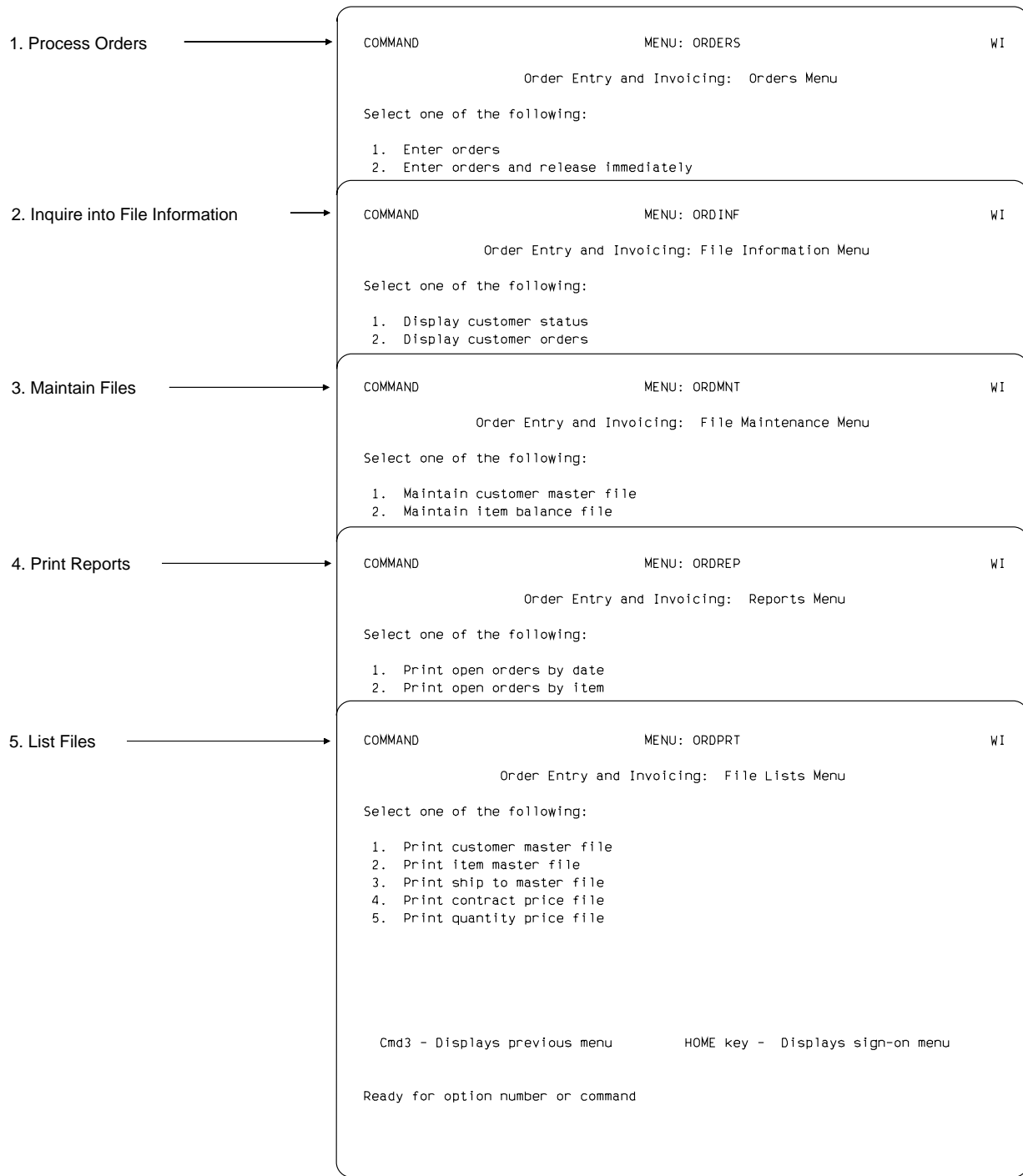
- Using an option on the menu that displays after the main menu
- Reminding the user F12 causes the previous menu to display

For example, on the Orders Menu, the user can return to the main order entry and invoicing menu (ORDENT) by pressing F12. You can allow experienced users to bypass the menu chains and directly begin their jobs by, for example, typing a procedure from the menu.



RSLW069-3

Figure 14-1. Menu Design Techniques



RSLW076-0

Figure 14-2. Menu Chaining

Creating and Changing Menus

A menu contains two library members:

- **Menu display file.** The menu display file tells the user what each option number does. It includes any descriptive text associated with an option number, the placement of the option numbers, and the name and title of the menu.
- **Command text message file.** The command text message file tells the system what to do for a selected option number. The command text message file describes what procedures, commands, or statements are used to run a job when the user selects an option number.

You can create and change your menus using the following:

- Screen design aid (SDA)
- Build Menu (BLDMENU) procedure

Using SDA to Create a Menu: The screen design aid (SDA), which is part of the AS/400 Application Development Tools (Appl Dev Tools) product, leads you through the steps to create and change a menu. Consider the following benefits of using SDA to create a menu:

- You can design your menu at the display station and see immediately how your menu looks.
- SDA automatically creates the source member and files required to display a menu. You supply only the menu text and the command text for the menu.
- You can use SDA to create and change online help information for each menu option or for the entire menu. For more information about online help information for menus, see “Creating and Displaying Online Help Information for Menus.”

The *ADTS/400: Screen Design Aid for the System/36 Environment* book describes how to use SDA to create and update menus and online help information for menus.

Using the BLDMENU Procedure to

Create a Menu: The BLDMENU procedure, which runs the \$BMENU utility program, is part of the System/36 environment. If you use the BLDMENU procedure instead of SDA, you must do the following:

- Determine the design of your menu on paper or on a preprinted form. Use the programming development manager (PDM), the source entry utility (SEU), or the \$MAINT utility program to enter and create the menu text and command text source members.
- Run the BLDMENU procedure to convert the source members into the load members used by the system.

Running the BLDMENU procedure compiles faster than SDA. For minor changes to your menu source members, use Source Entry Utility (SEU).

The *ADTS/400: Screen Design Aid for the System/36 Environment* book describes how to use the BLDMENU procedure to create and update menus.

Creating and Displaying Online Help Information for Menus

Although each menu option has a description of the job that runs when the user chooses a particular menu option, a simple job description may not be enough. For example, you may want to supply the following information for a menu used to run an application:

- A summary of the application, such as an explanation of when to select a particular menu option or job
- A description of the input forms necessary to do the job
- The work the user is expected to do while the job is running
- An estimate of how long a job takes to run
- A description of the output produced by the job, and an explanation of what to do with that output

You can supply this kind of additional information through online help information.

Creating Online Help Information for Menus:

SDA allows you to create online help information for menus. After you create your menu, you can add online help information for the entire menu, for a single option on that menu, or for a range of menu options. For example, if a menu has six menu options, you can create a display of summary online help information for the entire menu and a display of online help information describing the menu options in more detail. After you specify the range of option numbers for which you want to create online help information for menus, SDA shows you a display on which you can design the online help information.

After you create the online help information for your menu, you can change it just as you would the menu it describes. Specify the range of option numbers the online help information addresses, and SDA shows you the online help information. You can change the online help information.

Two types of online help information exist for display files:

- Cursor-sensitive online help information (using H-specifications)
- Menu option online help information (not using SFGR H-specifications)

Cursor-Sensitive Online Help Information:

When you use cursor-sensitive online help information, the user defines the help formats. Help (H) specifications that specify what cursor positions are supplied with online help information and what help formats appear for those positions refer to the help formats. You can define a general help menu that supplies online help information for all the parts of the display that have specific online help information. Cursor-sensitive online help information does not have to reside in the same display file as the display format.

The help format that appears depends on the current cursor position and the positions handled by the SFGR H-specifications. Work station data management handles the Help key and shows the proper help format.

The system-supplied menu processor does not support cursor-sensitive online help information.

You can use it only on menus shown by user-supplied applications.

Menu Option Online Help Information: When you use menu option online help information, the names of the help formats are predetermined by the system-supplied menu processor. The help formats must be in the same display file as the menu format. The names of the help formats are as follows:

- #H0000: General menu help.
- #HXXYY: xx is the starting option number and yy is the ending number for the menu online help information. For example, format #H0309 is used to display online help information for menu options 3 through 9.

The help format that appears depends on what option number the user selects. If the user does not select an option number, the general help menu appears. The system-supplied menu processor handles the Help key and shows the proper help format. The system calls the menu processor when you use the MENU OCL statement or Operator Control command.

The *ADTS/400: Screen Design Aid for the System/36 Environment* book has more information about creating and updating online help information for menus.

Displaying Online Help Information for Menus:

The user can display online help information for menus by doing one of the following:

- Pressing the Help key when the menu appears
- Typing an option number on that menu and pressing the Help key

General menu text (#H0000) appears if the user presses the Help key without typing an option number. To see the other help formats the user can use the Page Up and Page Down keys. For example, the user can press the Help key from the menu for jobs (that displays information about the files used by the inventory management application), to view the online help information for menus summarizing the jobs run by the menu, as shown in the following screen.

```

COMMAND  ONLINE HELP INFORMATION FOR MENU OPTIONS: 00 - 24      W1
Help information for the menu INVINP, which is used to display information
about the files used by the inventory management application.

Option 1 displays information about the item master file.
Option 2 displays item balance detail for goods located at the warehouse.
Option 3 displays item balance detail for goods located on the
manufacturing floor.
Option 4 displays status information about open orders.
Option 5 displays information about the availability of your inventory.
Option 6 displays history about the balance of your inventory.
Roll keys - Display additional information  Cmd3 or Enter key - Display INVINP

```

If the user types an option number and press the Help key, more detailed menu online help information about the menu option appears. For example, if the user selects option 1 on the inventory management application menu and presses the Help key, the information in the following screen appears:

```

COMMAND  ONLINE HELP INFORMATION FOR MENU OPTIONS: 01 - 01      W1
If you select option 1 of INVINP, you should know the following information
before you begin:
REQUIRED FIELDS:  Item number
OPTIONAL FIELDS:  None.
COMMAND/FUNCTION KEYS:  Cmd7 ends the program.  The data you type on the
display is ignored, and INVINP reappears.
NOTES:  When you press the Enter key to continue, the Item Master File
Record display appears, showing information about the file number
you entered.
POSSIBLE MESSAGES:  0101 XXXXXX - File record is missing
                    4520 Item master record not found
                    4675 Severe error - end program and rerun

```

Once the online help information appears, the user must return to the menu to select an option. Once online help information for menus appears, the user cannot use the Help key to obtain more online help information. However, if the user views a portion of the online help information supplied for a menu, the user can view the remaining online help information by pressing the Page Down or Page Up key. The user can return to the original menu by pressing F12.

Using Color or Highlighting on Menus

Color, used properly, is more pleasing to the eye than monochrome data. Use color to draw attention to fields that need user attention, such as a request for user input or a response to an error condition.

You can use a menu or display designed for a monochrome display station for a color display station.

Some display stations, depending on the model, show the following colors:

- | | |
|-------|-----------|
| Green | Turquoise |
| Red | Pink |
| White | Yellow |
| Blue | |

When you design a menu or display that uses color, consider the following:

- You must use the Design Display Format option in SDA to add highlighting or color to a menu. You cannot use the Menu option to add highlighting or color to a menu. Do not add, move, or delete fields on the menu. Once you have used the display format option to add color or highlighting, you can use only this option to update the menu.
- Use each color for a particular purpose. Use a color in the same way on every display. For example, use white to highlight important output fields or error messages throughout the displays that you design.
- Use a limited number of colors. The fewer colors used, the more effective each color. Too many colors on a display confuse the user.
- Group colors. If colors are grouped in a recognizable and consistent manner, the user can easily organize and follow information.

The *ADTS/400: Screen Design Aid* book has more information on using color effectively.

Even if you do not have color display stations, you can use SDA to highlight fields on your menus and displays. You can display a particular field in reverse image, in high intensity, or with column separators, blink or underline the field, or display

the field with a combination of these field attributes. Highlighting on a noncolor display makes the menu more interesting and easier to use.

Displays

You use a display station to communicate with the system and to run application programs.

The information shown on the display station is called a **display**. You define that information in a **display format**. Using the display defined in a display format, a program can prompt the user to type information or show requested information to the user.

The following screen is a sample display.

```
                DISPLAY ITEM MASTER FILE

                To display information in the item master file,
                type an item number and press the Enter key.

Item number:    50011230
Item description: Storage Cabinet with Doors

Item type:      E           Cost per unit:    250.00
Item class:     50          Selling price per unit: 325.00
Warehouse location: H1
Unit weight:    115

Date record last maintained: 12/15/88

Press the Enter key to see the next record in the file
Cmd1 to change the information in the record that is displayed
Cmd7 to end this program and return to the previous menu
```

The preceding display contains 24 lines of 80 characters each. For a 3180 Display Station or a Monochrome 3197 Display Station, the display can contain 27 lines of 132 characters each.

Displays are useful for the following reasons:

- They make it easier for the user to type and get data.
- They improve productivity.
- They are defined separately from the program or procedure that uses them, therefore:
 - One set of display formats can be used by different procedures or programs.
 - You can usually change display formats without recompiling your programs.

- You can have multiple language versions of one program.

Note: The *ADTS/400: Screen Design Aid for the System/36 Environment* book has more information about displays.

The following sections describe how your programs, and the system, use displays and display formats.

Display Data Management

Programs that communicate with display stations are called interactive programs. Interactive programs use a system function called **display data management** to write data to and read data from a display station. Display data management communicates with the interactive programs by using display formats to write data to and get data from the display.

A program must be able to do the following to use display formats:

- The program must specify the display file or files containing the display formats to use. You must define and open the display file and identify and describe it by a specification, statement, or subroutine.
- The program must be able to select the specific display formats to use and must be able to send data to the display. That data is the output needed by the user running the program. The sending of data to the display is often called an **output** or write operation.
- The program must be able to accept data typed on the display. That data is the input needed by the program. The accepting of data typed by the user is often called an **input** read, invite, or read operation.

In addition to the operations listed above, the program must be able to control other functions such as indicators and the command and function keys.

Figure 14-3 on page 14-11 shows examples of output, input, and input/output fields as described in the following sections.

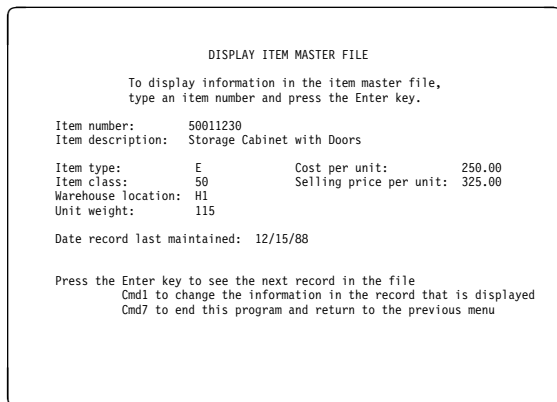


Figure 14-3. Sample Output, Input, and Input/Output Fields

Output Operations and Output Fields:

For output operations, display data management prepares a data stream to transmit to the display station by merging data supplied by the program with the display format.

Output fields contain information that the user cannot change on the display. The contents of output fields are not returned to the program. Output fields can contain data supplied by the program, or they can be **prompts** or **constants**, defined by the display format. A prompt is a request for information or action from the user. A prompt can tell the user what type of information to type, the form in which to type that information, and the options or values allowed as input for that data field.

Figure 14-3 shows examples of output fields. In this figure, all the output fields are prompts defined by the display format. An example of an output field on the display is the prompt *Item number*.

Input Operations and Input Fields:

Input fields are fields in which the user can type data on the display. When a program shows a display, input fields are blank or contain a default value. The user can type data into the input field. The contents of the input field are sent to the program when the user presses the Enter key. This data is used by the program to do an operation, such as a calculation or a file update.

An example of an input field on Figure 14-3, is the field in which the user typed the item number 50011230.

Input/Output Fields: Input/output fields allow the user to type data, and allow data to appear to the user. The user can type new data in a blank input/output field, or change existing data in an input/output field. Data displayed to the user can be supplied by the program or specified by the display format. Data contained in an input/output field returns to the program when the user presses the Enter key.

For example, when the user typed the item number 50011230 on Figure 14-3 and pressed the Enter key, the item description appeared. *Item description* is an input/output field because the user can type over the description.

Data Types

If you define a particular field as an input or input/output field, you must also define the type of data the user can type. For example, you can specify that an input field is to accept numeric data only (0 through 9, commas, decimal points, plus signs (+), and minus signs (-)). This definition is useful if a field requires information such as inventory amounts or account balances.

You can specify that a field is to accept alphanumeric data only (characters A through Z, special characters, and any numeric data). This definition is useful for a field that requires alphabetic characters and numeric digits, such as a customer's address.

Other data type definitions include:

- **Alphabetic data.** Only the letters A through Z, and certain special characters, are allowed.
- **Digits.** Only the numbers 0 through 9 are allowed.
- **Signed numeric.** Only the numbers 0 through 9 are allowed. The sign is determined by the key pressed, Field+ or Field-, after entry of the numbers.
- **Right-to-left field.** If the correct national language PRPQs are installed, the cursor moves from right to left within this field as the user types in data.

- **Katakana data.** This field can contain Katakana characters.
- **Double-byte character set (DBCS) data.** For the DBCS version of the System/36 environment, you can type and display DBCS characters.
- **Numeric shift fields.** On data entry keyboards, the keyboard automatically shifts to numeric shift when the cursor is in this field. On keyboards that are not data entry keyboards, the field defaults to an alphanumeric field.

Attributes

In addition to defining the field and the data type, you define the physical characteristics, or **attributes**, of a field. The attributes that you can specify include:

- **High intensity.** Data shown in high intensity is brighter than data shown in normal intensity. Your data looks as if it is displayed in bold faced characters. Use high intensity to draw attention to important information, such as display titles or column headings.
- **Blink field.** Data displayed in a blink field blinks. A blinking field is easy to see and draws attention to important information (but blink fields can be difficult to read if you use them too often).
- **Nondisplay.** If data is typed or sent to a nondisplay field, it does not appear on the display. A nondisplay field is useful for information needed by the program but not by the user, such as a display or record ID, or for confidential information, such as a password or security code.
- **Reverse image.** Data on the display normally appears as light characters on a dark background. If a field has the reverse image attribute, the data in that field appears as dark characters on a light background. Use reverse image to show the user the location of a field or to draw attention to an error message or to fields in which the user has typed incorrect data.
- **Underline.** Use underlined fields to emphasize information or to show the length of an input field. Showing the user the length of an input field is important because a keyboard

error message results if the user tries to type data outside the input field.

- **Column separators.** Column separators are useful for showing the number of positions in an input field. Column separators appear as dots or vertical lines (depending on the type of display station) on either side of each character position within the field. Column separators do not require character positions of their own. An input field with 5 character positions, for example, looks like this:

```
.i.n.p.u.t.
```

You can use combinations of field attributes. For example, you can specify that:

- A field appears in high intensity and reverse image.
- A field attribute is always used.
- A field attribute is controlled by the program.

The program can use a switch called an **indicator** to turn an attribute on or off.

Display Data Management Operations

When you define a display format, you can specify the following special operations:

- Erase input fields
- Override fields
- Suppress input

These special operations improve the performance of your display stations, display formats, and programs. You can perform these operations each time you use a display format, or as needed by setting on an indicator.

Erase Input Fields Operation: For an erase input fields operation, display station data management blanks out the contents of unprotected input and input/output fields on the display and invites input from the display. This operation sends the display format from the disk and sends the control characters required to remove the contents of the input field.

Request the erase input fields operation when an application using remote display stations requires a user to type information in the same fields time after time. Specify an indicator to control the erase input fields operation. The first time the

program shows the display format, that indicator must be off. The program should turn the indicator on for the next and following times it shows the display. Each time the display is issued with the indicator on, the input fields are blanked out, and the user can again type data in them.

This operation is important when a program communicates with a remote display station because the amount of information transmitted to a remote display station directly affects the performance of the jobs using the communications line.

See “Designing Displays for Remote Display Stations” on page 14-15 for remote display station considerations.

Override Fields Operation: For an override fields operation, display station data management:

- Transmits the contents of conditional output fields (output fields for which an indicator is specified for the output data attribute) if an output indicator is on.
- Retransmits the attribute for all field attributes controlled by indicators.

This operation is valuable when a program communicates with a remote display station, because it reduces the amount of information transmitted over the communications line.

See “Designing Displays for Remote Display Stations” on page 14-15 for remote display station considerations.

Suppress Input Operation: For a suppress input operation, display station data management does *not* invite input from the display station after transmitting the format to the display station. The user can type information into input fields on the display if the keyboard is not locked. However, the system does not send the typed information to the program until the program requests work station data management to read input from the display station.

This operation is usually used with display formats that contain only output fields. Use the suppress input operation if multiple display formats appear before input returns to the program. When mul-

iple formats are sent, specify the suppress input operation on all but the last format displayed.

When an input format is to be processed by a MRT or as part of Read-under format (RUF) processing, input must be invited. You can suppress input for a menu that is not used by a MRT or as part of RUF processing.

See “Designing Displays for Remote Display Stations” on page 14-15 for remote display station considerations.

Designing Displays

Design input displays for ease of data entry. Design output displays for ease of reading. Design displays that show output and allow input for ease of data entry and ease of reading.

Displays must be clear, complete, and understandable. A well-organized and descriptive display improves user productivity. When you design a display, consider the following factors:

- How the display is used
- What kind of information you want the display to process
- The source documents used as input for the display
- The level of experience or responsibility of the user
- The size of the display (24 lines of 80 characters, or 27 lines of 132 characters for the 3180 or Monochrome 3197 Display Station)

When you design your display, you should:

- Make the user feel productive.
- Identify the displays and supply meaningful headings.
- Design easy-to-read displays.
- Show a small amount of information at one time, or supply one idea for each display.
- Be consistent among displays.
- Keep user responses short.
- Respond to user input.
- Make error correction easy.
- Supply online help information.
- Describe your displays.

The *ADTS/400: Screen Design Aid for the System/36 Environment* book has more detailed information about these design guidelines.

Types of Displays

The following are general types of displays:

- Fixed-form
- Adjacent form
- Free-form
- Menu form
- Code-link form

Choose the correct form for your application.

Fixed-Form Displays: Using a fixed-form display, the user supplies input in response to prompts on the display. You design a fixed-form display to resemble the source documents that contain the data to enter into the system. Arrange prompts and input fields on a fixed-form display as fields are arranged on the source document.

The following screen is an example of a fixed-form display.

```

                                COMPANY CAR REGISTRATION INFORMATION
Motor ser. no.: 3TX0123   Purchase price: 7500   Dealer: Bob's Pontiac
Mfg.: PON   Year: 80   Model: FIREB   Style: ESP   Gross wt.: 3500
Ins.: Aetna
Name: Frank Fredora
Address: 1500 Rampart St.
City: Raleigh           State: NC   Zip: 27609   County: Wake
Fees
Document:      1.00
Title:         5.00
Registration:  13.00
Total:        19.00
Tag no.: LAG535
Enter key - Display next record
Cmd1 - Change this record
Cmd7 - End program
```

Adjacent-Form Displays: In an adjacent-form display, data is arranged and typed in columns. The first column contains prompts for the kind of information to type or display. The second column contains the fields that receive the input data or show the output data. You can use a third column to allow the user to change information displayed in the second column.

The following screen is an example of an adjacent-form display.

```

                                COMPANY CAR REGISTRATION INFORMATION
Motor ser. no.: 377X0123
Purchase price: 7500
Dealer: Bob's Pontiac
Mfg.: PON
Year: 1980
Model: FIREB
Body style: ESP
Gross wt.: 3500
Ins. co.: Aetna
Name: Frank Fredora
Address: 1500 Rampart St.
City: Raleigh
State: NC
Zip: 27609
County: Wake
Fees
Document: 1.00
Title: 5.00
Registration: 13.00
Total: 19.00
Tag no.: LAG535
```

Free-Form Displays: On a free-form display, you type data into long unprotected fields. The user types data in a string of characters. Individual fields or records are separated by a special predetermined character. For example, you can type a customer's name and address as follows, with the separating character being a semicolon (;):

```
last name;first name;initial;address;city;state;zip code
```

Free-form displays are useful for more experienced users and all rapid data entry. One line of constant header information is usually all that is needed to help the user in remembering the position of individual fields or records.

Your program must be able to interpret the format of that data. Special input array processing can be required.

Menu-Form Displays: One of the first displays a user sees within an application is a menu used to select a particular job. Menu formats can be applied to the displays that your programs use. Many applications have several predefined transactions for which the same kind of data or actions are used repeatedly. To end unnecessary typing of data, you can use a menu-form display on which the user types in a selection from a displayed list of options. Data is displayed or processed based on that selection.

The following screen shows a menu-form display.

```

COMMAND                                MENU:  INVINF                            W1

                Inventory Management:  File Information Menu

Select one of the following:

1.  Display item master
2.  Display item balance detail (warehouse)
3.  Display item balance detail (manufacturing)
4.  Display open orders
5.  Display item availability
6.  Display item balance history

Ready for option number or command

```

Code-Link Form Displays: The code-link form display is an extension of both the free-form and menu-form displays. In a code-link form display, the user selects a code number from the menu and types a value associated with that selection in free form. Based on the selected code number, the program must interpret and process the typed data (input array processing can be required).

The following screen shows a sample code-link form display.

```

Stock number: 1234
Inventory: 1 on hand      2 on order
Sales:

   3 Jan.  4 Feb.  5 Mar.  6 Apr.
   7 May   8 Jun.  9 Jul.  10 Aug.
  11 Sep. 12 Oct. 13 Nov. 14 Dec.

Type the inventory code and the sales code and press Enter.

Inventory code:
Sales code:

Press Cmd7 to end program.

```

Using Color or Highlighting on Displays: “Using Color or Highlighting on Menus” on page 14-9 describes how to use color to highlight data.

If you use SDA to create and update displays, you can select colors for each field from the Color Attributes for Field display. The *ADTS/400: Screen Design Aid* and the *ADTS/400: Screen Design Aid for the System/36 Environment* books have more information about highlighting data with color on your display.

Designing Multiple Formats: Information from several display formats can appear on the display at one time. This is useful if some information on the display is not to change and other information is to be replaced. If you use multiple formats, the following are true:

- Response time is improved because no unnecessary information is sent to the display station.
- Less coding is required because the specifications for each format are generally simpler.
- Because fewer specifications are required for each format, less disk space is needed and data is not duplicated.

When using multiple formats, be careful not to clear or replace any information that should remain on the display.

When multiple formats appear before information is read from the display, the system reads only the input fields from the last format displayed that had input fields. When all or a portion of a display format is replaced by a display format with input fields, the input fields from the previous display cannot be read if the input fields on the new display format are at different locations on the display.

If the display formats do not define any input fields, specify the suppress input operation for *all* but the last display format. The last format *must* define at least one input field if *all* of the following are true:

- A value other than 24, 27, or blanks is specified as the number of lines to clear.
- The format replaces or clears a line that contains an input field created by a previous format displayed with input fields defined.
- Suppress input is not specified (the user can enter data on the display).

Designing Displays for Remote Display Stations: Remote display stations communicate with the system at a slower rate than do local display stations and can decrease overall system activity. Use one of the following techniques to improve performance when using remote display stations:

- Reduce the amount of data transmitted over the communications line.
- Increase the line speed.

Reducing the amount of data transmitted is directly controlled by programming techniques. You can reduce the amount of data transmitted by following these suggestions when coding your display formats:

- Send only the data that the user needs to efficiently use the application. Consider creating help display formats for the user to request additional information.
- Do not show the same data and prompts again.
- Specify N (no) for the return input operation as described in “Input Operations and Input Fields” on page 14-11. This reduces the amount of data transmitted over the communications line and also reduces the response time.

If you have an identification field on a display format that a program is reading, specify Y (yes) for return input operation.

- Use an erase input fields operation, as described in “Erase Input Fields Operation” on page 14-12, to remove the contents of an input field rather than displaying the entire format again.
- If there is no need to display a format to get input from the user, then do not use the read-under format technique described in “Using the Read-Under-Format Technique” on page 14-20 just to pass information from one job step to another. Use the local data area to pass the information from one job step to another.
- Use an override fields operation, as described in “Override Fields Operation” on page 14-13, to display error messages. Display the error message and the fields in error only to avoid transmitting unnecessary data when errors occur.
- In the display format specifications, define the fields in the same left-to-right, top-to-bottom order that they appear on the display. Additional control characters must be transmitted for any out-of-sequence fields, thus increasing the response time.

- If you do use multiple formats, specify the suppress input operation, as described in “Suppress Input Operation” on page 14-13, on all but the last format. Specifying suppress input reduces the turnaround time before each new format displays. If input is not suppressed data is transmitted and received several times after each format displays.

When designing and coding display formats for remote display stations, use the proper coding techniques.

You can improve response by reducing data transmission. For example, in an application with five display formats in which input data is read and output data is displayed, the total characters transmitted over the communications line can number 2700. By properly using the return input, erase input, override fields, and suppress input operations, you can reduce the total number of characters transmitted over the line to 2100 or fewer.

Using Message Members with Your Display Formats:

Display formats allow you to use message members to display constant information. Message members are especially useful for fields that show error messages or conditional instructions. For example, if a program detects an input error, the program can select the correct error message from a message member and display that message. The program only has to specify the message identification code (MIC), a four-digit number that identifies a record in a message member, for the particular message to display. Use the following codes to indicate in which message file the message is located:

- M1 (##MSG1: first-level text)
- M2 (##MSG4: second-level text)
- U1 (User message file: first-level text)
- U2 (User help message file: second-level text)
- P1 (Program message file: first level-text)
- P2 (Program help message file: second-level text)

U1 corresponds to USER1, U2 corresponds to USER2, P1 corresponds to PROGRAM1, and P2 corresponds to PROGRAM2 in the // MEMBER OCL statement. See the // MEMBER OCL statement for more information.

The program or the display format does not have to define the text of the error message. Therefore, you can define one message member for your application and have all the programs within the application use the same messages in that member. This definition makes the application consistent and frees you from coding message text within your programs or display formats.

The message to display can be identified by the display format or specified by the program using the display format. If you specify a MIC number and a message member identifier for an output field in your display format, the corresponding message displays in the output field. If you do not specify a MIC number and a message member identifier for the output field, the program supplies that information in the output record area. When a write operation is sent to the display format, the message corresponding to the supplied MIC number and message member identifier displays in the output field.

For more information about creating and using message members, see Chapter 15, “Messages and Message Members.” Also, the *ADTS/400: Screen Design Aid for the System/36 Environment* book has more information about the specific entries made in the display format for displaying messages from a message member.

Using Self-Check Digits: Self-checking digits give some protection against data entry errors and fraud. Self-checking supplies a method of verifying the contents of an input field at the same time it is entered. This method is especially useful if an application requires the entry of numeric data such as account numbers.

Note: You must have the extended function feature installed on remote 5251 or 5294 display stations to use self-check digits. If this feature is not installed, a program error results when you specify self-check and the operator tries to leave the field.

The system offers two methods of self-checking:

- Modulus 10
- Modulus 11

If you specify a self-checking method for an input field:

- The system determines a self-check digit for the field's contents using the specified self-check method.
- That self-check digit is compared to the farthest right position of the input field.
- If the self-check digit matches the farthest right position of the input field, the contents of the input field are allowed, and the user can continue.

If those numbers do not match, the contents of the input field are not allowed, and a keyboard error displays. The user must type an allowed number before continuing.

The *ADTS/400: Screen Design Aid for the System/36 Environment* book gives more detailed information about how the system checks the contents of an input field according to the correct self-check method.

Creating Display Formats

After you design your display, you must create the library member the system uses to show the display. Every display is defined by a display format stored in a **display file**.

The display file can contain up to 255 display formats. Each display format is made up of **specifications**. See Appendix B, “\$SFGR Specification Forms” for more information about specifications. These specifications define information about:

- The entire display format. This information is defined in the S-specification.
- Individual fields in the display format. This information is defined in the field definition (D) specifications.
- Optionally, online help information available for the display. This information is defined in the H-specifications.

The system provides two ways for you to create display formats:

- Screen design aid (SDA) utility
- FORMAT procedure

Using SDA to Create a Display

Format: SDA leads you through the steps used to create a display format. SDA has several advantages over using the FORMAT procedure for creating display formats:

- You can design display formats at the display station and see immediately how the display looks.
- You can use SDA to test your displays. By controlling which indicators are on or off and by specifying the order in which a series of displays is shown, you can see how your displays work when the application is run.
- SDA does most of the work. You need only supply certain control information for the display, and the location and characteristics of the fields to display.
- SDA has some additional options to help you create your displays and code and describe your application programs. Using SDA, you can call the SEU full-screen editor to create or update source and procedure members. You can also use SDA to create RPG II program specifications for your display formats.

The *ADTS/400: Screen Design Aid for the System/36 Environment* book describes how to use SDA to create display formats.

Using the FORMAT Procedure to

Create a Display Format: If you use the FORMAT procedure, which runs the \$SFGR utility program, instead of SDA, you should:

1. Design your display on paper, or on a pre-printed form.
2. Use SEU or the \$MAINT utility program to enter and create the display format source member.
3. Run the FORMAT procedure to create a load member defined by the specifications in the source member.

After the FORMAT procedure creates the load member, the system prints information about the display format you defined. Use this information

to describe your displays and correct problems with your display formats and programs.

Using SEU to make minor changes to your display format source members and running the FORMAT procedure to compile them is faster than using SDA to make changes.

The *ADTS/400: Screen Design Aid* book describes how to use the FORMAT procedure to create display formats.

Creating Online Help Information for Your Displays

Use SDA, the FORMAT procedure, or the word processing function of OfficeVision for OS/400 to define online help information for your displays. You can use online help information to explain all or a portion of a display shown by the application program. To supply online help information to your application users, you define help areas on the display used by your application program. Each help area is defined by a specification called the help definition specification, or H-specification.

The word processing function of OfficeVision for OS/400 lets you define online help information and store it as a document in a folder. The *Getting Started with OfficeVision/400* book has information about defining online help information using the word processing function of OfficeVision for OS/400.

For SDA or the FORMAT procedure, the help area and its H-specification correspond to online help information on a type of display format called a **help format**. To use help formats, you should disable the Help key. The system detects whether the Help key is pressed, rather than allowing the program to detect it. If the cursor is within a help area when the user presses the Help key, the corresponding help format appears. Using the Page Down and Page Up keys, the user can page through other help formats you have defined for the display. Once the user has viewed the help formats, the user can return to the original display by pressing the Enter key or a function key.

Note: H-specifications are ignored by the system-supplied menu processor.

Using SDA Application Help Support to Create Online Help Information:

Application help is a part of SDA that allows you to define and delete H-specifications within your \$SFGR format members. Application help simplifies creating online help information for your applications. You can use SDA to do the following with your H-specifications:

Add	Display all
Change	Delete
Display	View

Help areas and help formats supply user instructions on the system, separate from the application they describe. You can add online help information to an application by making a few changes to the existing display formats and by creating the help format or formats that contain the online help information. Generally, you do not have to rewrite or recompile your programs to support the new online help information.

The *ADTS/400: Screen Design Aid for the System/36 Environment* book has more detailed information about how help areas and help formats are designed and created.

Using Display Formats with the Programming Languages

Each of the programming languages can use display formats. In addition, a procedure can use a PROMPT OCL statement to show a display. The display formats you create allow a program or procedure to use the display station as an input or output device.

The following sections describe how programming languages use display formats and how the display format display file is identified and described.

Using Display Formats with RPG II:

Programs written in RPG II use a work station (WORKSTN) file to use display formats. RPG II WORKSTN file programs require file description, input, and output specifications. To code these specifications correctly, you must use the information printed by the \$SFGR utility program after it has created the display format display file.

Display formats that an RPG II program uses *must* be designed, coded, and created *before* you write the RPG II program.

The file description specifications for a WORKSTN file identify:

- The file name assigned to the WORKSTN file.
- An indication that the WORKSTN file is a combined file. A combined file is capable of being both an input *and* an output file.
- The maximum length of the data that is read from or written to the display format.

The file description specifications identify the display format display file that contains the formats used by the RPG II program.

Because a WORKSTN file is a combined file, the data read from the display format must be described on the input specifications.

The request for a particular display format and the data to display are identified in the output specifications. The output record contains the program-supplied data to be sent to the display format.

The *System/36-Compatible RPG II User's Guide and Reference* book has more information about the use of RPG II WORKSTN files.

Using Display Formats with COBOL:

Programs written in COBOL use a TRANSACTION file to read from and write to display stations. The TRANSACTION file associated with the display station must be identified by the FILE-CONTROL paragraph of the CONFIGURATION section of the ENVIRONMENT division. The ASSIGN clause of the FILE-CONTROL paragraph associates the TRANSACTION file with a display format display file to be used by the COBOL program.

A WRITE statement, used in the PROCEDURE division of the COBOL program, identifies the specific format that displays. In addition, the WRITE statement sends program-supplied data to the display.

A READ statement, also used in the PROCEDURE division of the COBOL program, accepts data typed on the display format.

The *System/36-Compatible COBOL User's Guide and Reference* has more information about using COBOL TRANSACTION files.

Using Display Formats within a Procedure

If you want a procedure to show a display that prompts for input data, use the PROMPT OCL statement. The PROMPT OCL statement allows you to:

- Prompt for up to 64 procedure parameters (or a total of 1024 characters) by using one or more display formats.
- Define each parameter for the user.
- Assign default parameters.
- Control various display format functions.
- Show the display format to be read on the first read operation in a program (the read-under format technique described in "Using the Read-Under-Format Technique").

When you show a display using the PROMPT OCL statement, any parameters that have a value cause the corresponding display format indicator to be set on. For example, if parameters 1 through 5 and 7 have values, display format indicators 01 through 05 and 07 are set on.

The PROMPT OCL statement in the *System/36 Environment Reference* book has information about showing display formats and using parameters to set indicators on or off.

You can use this feature to:

- Display defaults for the parameters.
- Highlight a specific field. You can highlight a field when a parameter is typed wrong, to allow the user to identify the field in error.
- Position the cursor to a specific field. You can position the cursor to a field when a parameter is typed wrong, to allow the user to type the parameter again.

Using the Read-Under-Format Technique

The **read-under-format (RUF)** technique allows you to type information on a display while the program that uses the display is being loaded. When you use the RUF technique, a program or procedure displays a format and invites input, and

the next program called reads it. This format is first displayed by a PROMPT OCL statement with PDATA-YES specified, or by a program. While the next program starts, the user can type information on the display. When the user types the data, the information is sent to the next program that sends an input operation to the display station. If an output operation is issued before the next input operation, the RUF data is not processed.

This means that RUF can occur between:

- Two programs running with the same single request terminal (SRT) job.
- A PROMPT OCL statement and a program running in the same SRT job.
- A SRT program and a MRT program. RUF can occur in either direction, for example:
 - A MRT program can read in data from a format displayed by a SRT program that ran before the SRT called the MRT procedure.
 - After a SRT regains control from a MRT, a SRT program can read data from a format displayed by the MRT program before it released the device.
- A PROMPT OCL statement in a SRT procedure and a MRT program.
- A PROMPT OCL statement in a MRT procedure and the MRT program run from the MRT procedure.

Note: Program data from an OCL PROMPT statement takes priority over RUF being used by an application program when the PROMPT statement was interpreted.

The RUF from an application program and program data from the OCL PROMPT statement take priority over program data specified when a procedure is called.

You can use the RUF technique with all types of applications. The RUF technique can decrease the size of a program because it requires fewer read and write operations. Although the RUF technique increases response time because of the extra work the system does while starting and ending a program, overall performance is improved because two tasks are done at once. For example, while the second program starts, the user is already typing data for the first input operation in that program.

Using Data Description Specifications (DDS) and Screen Format Generator (\$SFGR)

The following sections describe:

- How screen format generator (\$SFGR) utility changes System/36 SFGR to AS/400 data description specifications (DDS)
- Considerations for using \$SFGR and DDS
- Differences between System/36 SFGR and AS/400 DDS

Using \$SFGR to Change SFGR to DDS

\$SFGR creates AS/400 display files from System/36 SFGR source containing S (display control), D (field definition), and H (help definition) specifications. For more information on completing the specification forms, see Appendix B, “\$SFGR Specification Forms.”

You usually use \$SFGR from System/36 through the FORMAT and BLDMENU procedures. When using a System/36 interface, the SFGR source member must be in the AS/400 source file QS36SRC. The CRTS36DSPF command is installed in QSYS when library QSSP is installed. The CRTS36DSPF command can also be used when you are not in the System/36 environment. The System/36 migration product can use CRTS36DSPF to compile the display format source into an AS/400 display file. The CRTS36DSPF command can check SFGR syntax, generate AS/400 DDS from the System/36 SFGR source, and save the DDS in the user’s source file. See the *CL Reference* book for more information.

SFGR Printed Output

\$SFGR produces a compile printout called QPUTSFGR. The DDS compiler produces a printout with the same name as the display file being created. The DDS compile printout contains the data description and error information. The flag parameter specifies the minimum severity of DDS error messages that are printed.

\$SFGR calls SFGR syntax-checking modules to syntax check the SFGR source and produce an SFGR compile printout. This printout includes

warning or error messages from the syntax checking. If warning messages are printed but there are no errors, a System/36 message is issued and the compile continues. \$SFGR uses default values that change the results. You should change the original SFGR source so the SFGR source is correct. \$SFGR calls conversion modules to create AS/400 DDS source. The conversion modules assume that the SFGR source contains no terminal errors. You cannot predict the results if the SFGR source contains terminal errors. If the SFGR conversion is successful, \$SFGR runs the AS/400 Create Display File (CRTDSPF) command to compile the DDS.

When you use the FORMAT or any other procedure running within the System/36 environment, you can control the SFGR compile printout through the System List (SYSLIST) procedure or OCL statement. The SYSLIST options you choose can cause the printout to be:

- Shown at the requester’s display
- Canceled
- Directed to a specific printer device

\$SFGR produces the output list using an AS/400 print file. When run from within the System/36 environment, the SYSLIST CRT function is simulated by using the AS/400 display spooled file (DSPSPLF) command. The SYSLIST options do not apply to the DDS compile listing produced by the CRTDSPF command.

The \$SFGR print options are mapped to the following OPTION keyword values and the FLAG parameters on the AS/400 CRTDSPF command:

PRINT-YES	==> OPTION(*SRC *LIST)	FLAG (00)
PRINT-PARTIAL	==> OPTION(*NOSRC *NOLIST)	FLAG (20)
PRINT-NO	==> OPTION(*NOSRC *NOLIST)	FLAG (20)

Note: For the \$SFGR print options (print-partial and print-no), the DDS compile printout contains only error messages that are equal to or greater than 20.

Creating, Adding, Changing, or Deleting Display File Formats

You can add new formats to the display file, delete whole formats from the display file, or update a format in the display file. When you create a display file, \$SFGR saves the DDS source in DDS source file QS36DDSSRC, in the same library as the display file. If there is no

QS36DDSSRC source file, \$SFGR creates one with a record length of 92. The first user who creates a display file in a library owns QS36DDSSRC. All other users have the create authority defined for the library. If the DDS source file QS36DDSSRC already exists, the \$SFGR utility does not change the authorities to the file. This DDS source should only be changed by \$SFGR. The DDS source member always has the same name as the display file. When you create additional displays in this library, \$SFGR adds corresponding DDS source members to the source file.

The following System/36 SFGR functions affect the DDS source member in QS36DDSSRC as follows:

Create

Creates DDS from the System/36 SFGR source and copies the DDS source into the DDS source member. The source member is replaced if it already exists. \$SFGR copies file-level DDS only when you use the create function.

Delete

Deletes the DDS format records from the QS36DDSSRC source member.

Add

Creates new DDS from the System/36 SFGR source. Appends the new DDS source to the QS36DDSSRC source member.

Update

Creates new DDS from the System/36 SFGR source. Deletes the old DDS format records from the QS36DDSSRC member. Appends the new DDS source to the QS36DDSSRC source member.

You can specify up to 32 requests (32 creates, or a combination of adds, deletes, and updates) at once with the System/36 \$SFGR utility, but the System/36 FORMAT procedure allows only one request at a time. You must write your own procedure to specify more than one request at a time. Processing several source members at a time is more efficient than updating one after the other. Performance improves more when all of the SFGR source is in a single source member.

Creating New Files Using DDS Instead

of \$SFGR: You must use certain DDS keywords when you want to create new files in the System/36 environment using DDS instead of SFGR:

- **Separate indicator areas.** On System/36, the formats and programs are created with separate indicator areas. DDS allows users to specify the indicator area (INDARA) keyword in a display file to get a separate indicator area.

The System/36 environment supports only display files and ICF files with separate indicator areas. You can satisfy the System/36 requirement by specifying the INDARA keyword in the DDS to get a separate indicator area. The System/36 environment imitates the System/36 method of using indicator areas while using RUF between display files or using RUF between ICF files when the files use different indicator conventions.

Requiring a file to have a separate indicator area allows the application to migrate between the AS/400 system and System/36. If the System/36 environment application uses an indicator convention other than the separate indicator area, the application has to be changed before returning to a System/36.

- **INVITE keyword.** You must add the INVITE keyword for the following reasons:
 - The display must be invited after a write operation to use RUF. If the display was not left invited while waiting for RUF to complete, the system issues an exception instead of waiting for the RUF to complete.
 - You must specify the INVITE keyword to read from a file with multiple devices.

The System/36 environment RUF supports only display files and ICF files with INVITE keywords.

Replacing a System/36 Load Member with an AS/400 Display File

When you use the REPLACE-YES option to create a display file, \$SFGR copies the original DDS source to library QTEMP and uses this copy for any add, delete, or update requests. If the

compile is successful, \$SFGR copies the new DDS source back to the DDS source member in QS36DDSSRC, replacing the original DDS. The original display file is deleted, and the new display file is moved from QTEMP to the target library. When replacing a display file, \$SFGR does not allow others to use the file during the compile. If the display file is already being used, the compile fails and an error message appears. If an SFGR compile fails, the original display file and its DDS source file remain unchanged in their original library. However, if the target library is QTEMP, the original display file may no longer exist after an SFGR compile fails.

Display files, programs, and message files (*FILE, *PGM, and *MSGF) are separate object types on the AS/400 system. On System/36, programs, message files, and display files are called load members. If you create a display file specifying REPLACE=YES, you can also replace any program or message load members with the same name.

The following file types exist on the AS/400 system:

- Database file
- Source file
- Printer file
- Bisynchronous communication (BSC) file
- Systems Network Architecture (SNA) communications file

Only one type of file can be in a library at a time with the same name.

Consider the following system actions to avoid accidentally deleting files:

- \$SFGR replaces only a *FILE of subtype DSP. It does not replace any other type of AS/400 *FILE object.
- The System/36 environment puts all database files in one library (usually called QS36F). The only source members that you can refer to directly through the System/36 environment OCL are in source files named QS36SRC or QS36PRC. Do not place display files in library QS36F and do not use names such as QS36SRC or QS36PRC for System/36 load members.
- Do not create display files in library QTEMP because temporary (RETAIN-S or RETAIN-J)

database files are there. \$SFGR always creates a display file in QTEMP before moving it to the target library.

Maximum Number of Display Devices

When you use MRT programs or programs that acquire multiple devices for the same display file, you must use the Maximum Device (MAXDEV) attribute within the display file to define the maximum number of devices you can use with the display file. On System/36, you determine and set the maximum number of devices that a display file can handle at a time at program start. When you use \$SFGR on the AS/400 system, you must know how many devices a display file can handle at a time during SFGR compilation. You can set the MAXDEV attribute on the SFGR source member to tell \$SFGR how many devices to allow. If you do not set the MAXDEV source attribute, the system uses a default value of 5. When a display file has already been created, use the AS/400 Change Display File (CHGDSPF) command to increase or decrease this value. If you set the MAXDEV value too high, it requires more storage and causes slower performance. Use the Change System/36 Source Attributes (CHGS36SRCA) or Edit System/36 Source Attributes (EDTS36SRCA) command to set the MAXDEV attribute in each SFGR source member, so you do not have to change this value each time the display file is changed or re-created. See the *CL Reference* book for more information.

Public Authority to Use SFGR Display Files

The system creates all objects with the create authority (CRTAUT) of the library they are created into, unless another authority is specified. The \$SFGR and other System/36 environment utilities provide an SSP keyword which allows you to create a library member with *USE authority. If SSP=YES is specified, the library member can be deleted only by a user with security officer authority, and users without security officer authority are not allowed to change the display formats. If SSP=NO is specified, the library member is created with the create authority (CRTAUT) of the library the member is created into.

FORMAT Procedure Parameters

The following sections describe using the FORMAT procedure HALT and NUMBER parameters with \$SFGR.

The HALT Parameter: The AS/400 system uses the NOHALT parameter on the FORMAT procedure to prevent \$SFGR error messages from being sent to your display. For NOHALT, \$SFGR sets the System/36 environment return code and the AS/400 CL return code. \$SFGR initializes the return codes to zero, and sets them to 1008 if it detects an error.

The NUMBER Parameter: System/36 uses the NUMBER parameter value to calculate the size of a work space that it must allocate. The \$SFGR on an AS/400 system does not use a work space and therefore does not need this number. The AS/400 system accepts a valid NUMBER parameter and passes it to \$SFGR. \$SFGR checks it for proper syntax and valid range of values. An error message appears if the value is invalid.

Differences between System/36 SFGR and AS/400 DDS

The following sections describe differences between System/36 SFGR and AS/400 DDS.

Format Names: On System/36, the SFGR for each format (S-specification columns 7 through 14) assigns the names of the format being defined. The first character of the display format name must be alphabetic: characters A through Z, or the characters #, \$, or @. The other characters can be any combination of characters except commas (,), single quotation marks ('), and blanks. There must be 8 or fewer characters in the display format name, and it must be left-justified in the field.

Names for DDS formats on the AS/400 system must begin with an alphabetic character (A through Z, @, \$, and #). All other characters can be alphanumeric (A through Z, 0 through 9, @, \$, #, and _). Embedded blanks are not allowed.

\$SFGR substitutes a valid DDS format name in place of an SFGR format name that does not

meet DDS restrictions. It uses the invalid SFGR format name as a DDS alternate name (DDS ALTNAME keyword), which is less restrictive. An application can refer to the same format with either name. AS/400 programs can refer to the format using the DDS format name, and System/36 programs can continue to use the alternate System/36 format name.

A HLPSEQ (help sequence) keyword is generated for all format names greater than seven characters in length that conform to the AS/400 HELP format naming convention. This convention is described in "Help Definition (H) Specifications" on page B-8.

When a HELP format name contains unsupported characters, use of either name applies to System/36 HELP format names used by System/36 as well. If the format name referred to in the H-specification (columns 7 through 14) is defined in the display file being created (columns 16 through 23 of the H-specification are blank), the corresponding DDS that is built refers to the valid DDS format name created by \$SFGR.

If the HELP format name referred to contains characters not allowed by DDS, and columns 16 through 23 of the H-specification are not blank (indicating the HELP format is in some other display file), the valid DDS format name created in the other display file is not known to \$SFGR, so that H-specification is ignored. To use this H-specification, you must change the HELP format name in that H-specification and the corresponding format name in the other display file, to a valid DDS format name.

Right-to-Left Cursor between Input

Fields: On System/36, column 40 of the SFGR for each format (S-specification) defines whether the cursor moves from one input field to another in a right-to-left direction. On the AS/400 system, the Check Right-to-Left (CHECK(RLTB)) keyword is a file-level keyword that applies to all formats in the display file. There can be only one such keyword.

\$SFGR takes the right-to-left definition from the first format (S-specification) encountered during the create function and uses that definition as the default for all other formats in the display file for the AS/400 system.

If System/36 SFGR specifies right-to-left differently depending on which format is being displayed, the results on the AS/400 system differ from the results on System/36. Place the right-to-left definition you prefer in the first S-specification. You must create separate display files if you want some to be left-to-right and some to be right-to-left.

You cannot change the right-to-left definition with the add, update, or delete functions, even if that request changes the first S-specification. Only a create function assigns the proper right-to-left function.

Data Types for Input/Output Fields: On System/36, data types E, F, O, and X (D-specification column 27) are related to DBCS-capable fields. DDS supports DBCS-capable fields only when you use a DBCS version of the OS/400 program. If you are compiling SFGR specifications without a DBCS version of the operating system installed, these data types are mapped to the System/36 Data Type B that allows alphanumeric data (data type A in DDS). If you use the DBCS version of the OS/400 program later, recompile the display file with the original SFGR specifications to obtain full DBCS support for these input/output fields.

Moving from System/36 to the System/36 Environment

Consider the following when you migrate help specifications created with the \$SFGR utility:

- The system always assumes null fill.
- Help displays on System/36 have input fields. The AS/400 system ignores or discards input entered on a System/36 environment help display.
- The RESTORE parameter of the H-specification is ignored when data is returned and a function key is passed.

System/36 Display File Enhancements

This section describes enhancements in display file handling since Release 1.0.

Using the Print Key: On System/36, the SFGR for each format (S-specification) defines whether the Print key is handled by the system or by the application program.

In Release 1.0, the PRINT keyword in the DDS was a file-level keyword that applied to all formats in the display file, and \$SFGR used the Print key definition from the first format (S-specification) as the default for the entire file.

In Release 2.0, the Print key definition is handled at the format level as it was on System/36, but the display file created and the DDS are not compatible with Release 1.0. To obtain the full format-level print function, the SFGR source for Release 1.0 display files can be compiled again.

To create a display file that will be used on an AS/400 system operating with a previous release, compile the SFGR source using the CRTS36DSPF command and specify TGTRLS(*PRV).

Displaying Messages: In your SFGR source, you can define a field whose displayed value comes from a message. The message is specified either by the SFGR source or by the application program, and can be selected by an output indicator. In Release 2.0, improvements have been made, but the created display file and the DDS are not compatible with Release 1.0. To obtain the full System/36 message display function, the SFGR source for Release 1.0 display files can be compiled again.

If a display file is to be used on an AS/400 system operating with a previous release, use the CRTS36DSPF command and specify TGTRLS(*PRV) to compile the SFGR source.

Return Input = No: Return Input = No (SFGR S-spec column 22 = N) is fully supported.

Position Cursor: On the AS/400 system, Position Cursor (SFGR D-specification, columns 32 and 33, where an indicator is specified) is fully supported. Display files from Release 1.0 may optionally be compiled again to take advantage of the added function.

***DDS in SFGR Source Comments:**

User-supplied DDS statements can be supplied in the SFGR source for insertion into the created DDS. This is recommended only for inserting the FRCDTA keyword after a format definition (S-specification) to override DFRWRT(*YES) for that format. **Defer write (DFRWRT)** is an attribute used, when creating a System/36 display file, to control sending formats to the display. See Appendix B, “\$SFGR Specification Forms,” for more information about *DDS in SFGR comments, and “DFRWRT Attribute” for more information about DFRWRT.

Optimizing Performance of Display Files

You can use several coding techniques to improve an application’s performance when doing input/output (I/O) with a display file. Some of the coding techniques are dependent on the application’s use of one or more of the following display data management operations:

- Erase input fields
- Override fields
- Suppress input
- Use multiple formats

These display data management operations are selected when the application sets an indicator on for corresponding operations in the SFGR specifications of the display file.

For further information on these topics, see “Display Data Management Operations” on page 14-12, “Designing Multiple Formats” on page 14-15, and “Designing Displays for Remote Display Stations” on page 14-15.

Other techniques are not dependent on the application program, but depend only on the options used when creating the display file, as discussed in the following sections.

MAXDEV Attribute: If a display file is created to handle a large number of devices, more storage is required, and it takes longer for the system to load the display file into main memory. A MAXDEV value greater than 1 is needed only when an application that uses the display file is a MRT application or an application that acquires

additional work stations. The sum of the MRTMAX and the number of acquired devices should equal the MAXDEV value. A MAXDEV value larger than this sum degrades the application’s performance slightly. You can adjust the MAXDEV value for a given display file after the display file is created by using the CHGDSPF command. The default MAXDEV value, used when the display file is created, can be changed by using the CHGS36SRCA or EDTS36SRCA command to set the MAXDEV attribute for the source member containing the SFGR specifications that define the display file.

Multiple Display Format Sizes: Avoid mixing 132-column and non-132-column formats in the same display file. Column 39 in the SFGR S-specification specifies the size (80 column or 132 column) of the format. When a display file contains some formats for 132-column devices and other formats for 80-column devices, the system may have to rebuild the data stream at run time to handle the I/O request. Rebuilding the data stream requires extra time and causes slower performance of your application.

DFRWRT Attribute: You can specify the DFRWRT attribute on the CRTS36DSPF command when creating your System/36 display file. The attribute in the source member containing the SFGR source that defines the display file can also be set using the CHGS36SRCA or EDTS36SRCA commands. If this source attribute has not been set, DFRWRT(*YES) is the default.

The DFRWRT attribute allows you to specify how the system is to handle write operations. If you specify DFRWRT (*NO), the write operation is sent to the display and control is returned to the application when the write operation is displayed on the display station.

If you specify DFRWRT(*YES), the write operation is placed in a buffer (not sent to the display station) and control is immediately returned to the application. If an application is issuing multiple write operations, specifying DFRWRT (*YES) results in significantly improved performance. The deferred data is sent to the display when:

- The application issues a read request to the display station.
- The application releases the display station.
- The application closes the display file.

- The buffer used to hold the deferred data is full.
- A write operation is done to a format that has specified the DDS FRCDTA keyword.

For example, a format uses a variable starting line and defines fields for a single line on the display. The application builds the final display image by writing the format once for each line on the display, while increasing the starting line number. If DFRWRT (*NO) is specified, the application waits for each line to be displayed on the display station. If there are 24 lines that are displayed, the application must wait 24 times. If DFRWRT (*YES) is specified, the application only waits when the system buffer (containing all the lines of output data) is displayed instead of waiting for each line.

If an application writes information that must be displayed immediately and does not require any user input, the application should specify the FRCDTA keyword in the format or specify DFRWRT (*NO) for the display file. If the data is allowed to be buffered by the system, the data is not immediately displayed by the system.

Common examples of these applications are:

- Continuously update a time-of-day display
- Send status messages, such as:
 - Running application
 - Please wait
 - Starting final pass

Some display files can contain a mixture of DFRWRT(*YES) and DFRWRT(*NO) formats. You can use each format when appropriate in

applications, without making any programming changes. To do this identify those formats in the DFRWRT(*NO) category, and locate their S-specifications. Immediately after the S-specification for each of those formats insert a *DDS comment similar to the following:

```
1234 SFMT01
*DDS S*                                FRCDTA
1234 S*.1....+....2....+....3....+....4....+....5....+....6
```

The third line is a comment to show the column numbers for the previous lines. In the preceding example, format FMT01 requires DFRWRT(*NO), so a *DDS comment was inserted after its S-specifications. A *DDS comment contains the characters *DDS in columns 1 through 5, and an asterisk (*) in column 7. The remainder of the line after column 7 must contain a valid DDS keyword in the proper DDS column.

The FRCDTA keyword temporarily overrides the DFRWRT(*YES) attribute whenever this format is written. The FRCDTA keyword starts in column 45. Since a *DDS comment is an SFGR comment, the SFGR source can still be installed and compiled on System/36. See Appendix B, "\$SFGR Specification Forms," for more information about *DDS comments.

Recompile the display file specifying DFRWRT(*YES). Specify DFRWRT(*YES) on the CRTS36DSPF command or in the source attribute in the member containing the SFGR source. The resulting display file provides improved performance for those formats without the FRCDTA keyword, while those formats with the FRCDTA keyword are displayed when required by the application.

Chapter 15. Messages and Message Members

This chapter describes:

- Message types
- Message concepts
- How to create and change messages
- How to use and display messages

Types of Messages

The types of messages that appear in the System/36 environment are as follows:

- **Error messages.** These messages indicate an error has occurred and the system is waiting for a response.

Some IBM-supplied error messages have automatic responses (default responses on the AS/400 system) and severity levels. The system uses the severity level of the message, and the severity level at which the application program runs, to determine when to respond to the error message automatically rather than having the operator enter the response.

- **Informational messages.** These messages provide a descriptive statement about a current or completed system action. They do not require a response. For example:

LISTLIBR procedure is running

or

Payroll program running

- **Prompting messages.** These messages ask a user to enter information. For example:
Enter the library member name.

Messages can be displayed or printed.

Displayed messages also allow a user to communicate with other display stations and users.

IBM-supplied printed messages are used to show errors or information about source members, such as programs, display formats, menus, or message load members you have compiled. For example, use printed messages for report headings.

Message Concepts

The messages that appear in the System/36 environment are created by and structured according to the System/36 message load member concept or the AS/400 system message file concept. For some uses you may decide to use message members, but for others you may want to use the message file concept.

Message Member Concept

System/36 messages consist of a message identification code (MIC) and a first-level message or a second-level message. A **first-level message** is a message that is issued immediately when an error occurs. A **second-level message** is a message that supplies additional information about an error condition when the Help key is pressed for a first-level message. A message source member is a source member in a QS36SRC source file (source type is MSGF36) that defines the parts of the message (MIC and text).

System/36 message members allow messages from 1 to 75 characters of first-level text. All programming languages and procedure control expressions use messages. On System/36, message help can contain from 1 to 225 characters of second-level text.

Uses of Message Members: You can use message members to define messages for programs, display formats, and procedures. For example:

- Titles of displays or computer printouts. For example: Stock Status Report
- Operator information shown on displays. For example: Press the Enter key to continue
- Printed or displayed application error messages. For example: Customer number must be entered
- Procedure substitution using the ?Mmic? expression. See the *System/36 Environment Reference* book for information on the ?Mmic? substitution expression.

Using message members allows you to store your messages in one place and refer to them by MIC

number instead of coding the text each time you want to display or print them. Using message members also avoids having the same text worded several ways.

You can create messages that allow data to be inserted when a procedure is running. See “Inserting Variable Data into Displayed Messages” on page 15-3 for more information.

For the double-byte character set (DBCS) languages, the system allows you to create message members with the message in two languages. The *System/36 Environment Reference* book has more information about creating these types of message members. Chapter 20, “System/36 Environment National Language Support,” describes DBCS characters.

For your programs and procedures to use these messages on the AS/400 system, you must use the AS/400 Create System/36 Message File (CRTS36MSGF) command or the System/36 environment CREATE procedure to create an AS/400 message file. See the *System/36 Environment Reference* or the *CL Reference* for more information.

Message Members and Application

Design: When you design your applications, you can:

- Create a single message member (AS/400 message file) in which to put all application messages.
- Create several message members and group messages by:
 - Program type. For example, place the order-entry messages into a member named ORDERMSG.
 - How they are used. For example, place the displayed messages into a member named MSGDISP, and the printed messages into a member named MSGPRINT.

Example Message Source Member:

The following is an example of a message source member:

```
MSGSAMPL,1
0001 Enter your name
0002 Enter yesterday's date
0003 ACCOUNTS PAYABLE APPLICATION
```

The example indicates:

- The name assigned to the message member (AS/400 message file) is MSGSAMPL. The 1 following MSGSAMPL indicates that the member contains first-level messages.
- The numbers (0001, 0002, and 0003) are MICs. These numbers identify the message. For example, to display or print the message
Enter your name
use MIC 0001.

Some messages to which you can respond allow the following responses: 0, 1, 2, 3, D (dump), F (formatted dump), and H (help).

Message Files and the System/36 Environment

To run on the AS/400 system, messages are always stored in AS/400 message files. Each message becomes a message description in the message file. Once messages are in message files, they can be altered using AS/400 commands. See the Change Message Description (CHGMSGD) and the Add Message Description (ADDMSGD) commands in the *CL Reference* book for more information.

Creating Messages from Message

Source Member: When a message is created from a source member you need to understand the following changes:

- The MIC is changed to a 7-character message identifier.
- The last 4 characters of the message identifier are the MIC. The first 3 characters of your user messages are set to USR except for DBCS messages.
- For DBCS messages (messages defined following MIC A000 in the System/36 message source member), the first 3 characters of the message are changed to USZ to distinguish it from the USR version. A USZ message may exist without a corresponding USR version.

Rules for Changing Message Prefix:

You can change the first 3 characters of the message identifier using the following rules:

- Message prefixes must be the same for the entire message file (with the exception of a file containing DBCS and non-DBCS messages).
- When a user message is sent, the system gets the first message from the message file and uses the first 3 characters of the message ID as the first 3 characters of the message identifier (the 7-character AS/400 number) for the message you are requesting.

For example, when you display message 2345 and the first message in the message file is PAY0002, the system looks for message identifier (the 7-character AS/400 number) PAY2345.

The third prefix character in the message identifier (the 7-character AS/400 number) for your DBCS messages is always a Z. In this example, the system looks for message PAZ2345 when a DBCS version of the message is found.

Note: If you are converting the first 3 characters of the message to a different prefix, do not use any numbers as the third character, because those values are used for keeping automatic response information.

- The prefix for all IBM product message files is predetermined and cannot be changed. If you alter the message prefixes for IBM product files, the system may not be able to find the messages.

Inserting Variable Data into Displayed Messages

You can insert variable data into a message using the ERR procedure:

1. Code the message with a # sign for each character you want to insert.
2. Use the ERR procedure and enter the data to be inserted in the appropriate parameter.

For example, if MIC 0003 in message file DISPMSG is:

Procedure ##### on the job queue
and the following statements are in a procedure:

```
// MEMBER USER1-DISPMSG,LIBRARY-INVLIB  
ERR 0003,0,INVJOB
```

the following message appears:

```
USR0003 Options (0)  
Procedure INVJOB is on the job queue
```

Because one character from the ERR procedure is substituted for each # sign, your messages can have more than one field of insert data. For example, if MIC 0005 in message file DISPMSG is:

```
Job ##### changed file #####
```

and the following statements are in a procedure:

```
// MEMBER USER1-DISPMSG,LIBRARY-INVLIB  
ERR 0005,0,'INVJOB CUSTMST'
```

the following message appears:

```
USR0005 Options (0)  
Job INVJOB changed file CUSTMST
```

Notes:

1. The variable fields in the message must be long enough to handle the data to insert.
2. You must insert blanks if the data is shorter than the variable field in the message.

You can insert the entire message from the ERR procedure by creating one message that contains 75 # signs. For example, if MIC 0004 in message file DISPMSG is:

```
##### ... #####  
75 # signs.
```

and the following statements are in a procedure:

```
// MEMBER USER1-DISPMSG,LIBRARY-INVLIB  
ERR 0004,13,'A REQUIRED FILE CANNOT BE ACCESSED.'
```

the following message appears:

```
USR0004 Options ( 13)  
A REQUIRED FILE CANNOT BE ACCESSED.
```

See the *System/36 Environment Reference* book for complete information about the ERR procedure.

Converting Message Text

When message text is converted from System/36 message member source to AS/400 message files, variable data fields follow the AS/400 message description format. This means converting the strings of consecutive # signs to &n substitution text descriptors (n is a number). The format parameters for the subsequent text descriptors are defined to be of type *CHAR, and the same length as the consecutive number of # signs. The substitution text descriptors are in sequential order for compatibility. For example, if MIC 0005 in message member DISPMMSG is:

```
Job ##### changed file #####
```

following conversion the message text will appear as follows:

```
Job &1 changed file &2
```

where the format of &1 is type *CHAR of length 8 and the format of &2 is type *CHAR of length 8. \$MGBLD converts System/36 message source to AS/400 message CL source and maps System/36 insertion text fields (# strings) to AS/400 insertion text fields (&n). The insertion text field conversion is done selectively so that normal text containing a # character (such as ##MSG1 or #LIBRARY) is not confused with true insertion text fields. The selection requires a delimiter on both sides of the # string to qualify the # string for conversion to an AS/400 insertion text field. The delimiter that precedes the # string can differ from the delimiter that follows the # string.

The following table shows a list of valid message source delimiters:

Hex	EBCDIC	Description
40		Blank (space)
41		Special blank
4B	.	Period
4C	<	Less than
4D	(Left parenthesis
4E	+	Plus
50	&	Ampersand
5C	*	Asterisk
5D)	Right parenthesis
5E	;	Semicolon
60	-	Hyphen
61	/	Slash
6B	,	Comma
6E	>	Greater than
6F	?	Question mark

Hex	EBCDIC	Description
7A	:	Colon
7D	'	Apostrophe (quote)
7E	=	Equal
7F	"	Double quote

If the # string occurs at the beginning or end of the message text, the beginning or end condition is treated as if it were a delimiter.

Occasionally, you may find it necessary to force a blank or hyphen symbol to be treated as if it were not a delimiter. If you must do this, you can use (depending on your language and type of work station) the required blank special character (HEX 41) in place of a blank or the syllabic minus sign special character (HEX CA) in place of a hyphen on either side of a # string to prevent the # string from being treated as a substitution field, without changing the visual representation of the message text. To substitute these characters in your System/36 message source, and for information on how to enter characters in hexadecimal form, see the reference book for your display station.

In some languages, the # symbol (HEX 7B) is treated as an alphabetic character used in text. If no messages allow insertion text fields, you can use the CRTS36MSGF command specifying SUBST(*NO) rather than the CREATE procedure to prevent replacement of # characters with substitution fields. Also, you can use the AS/400 Change Message Description (CHGMSGD) CL, Delete Message Description (DLTMSGD), or Add Message Description (ADDMSGD) command to change or replace the message description.

For more information on AS/400 substitution data and format parameters, see the *CL Reference* book.

If there are no coexistence requirements (same messages used on both the System/36 and the AS/400 system), you may wish to take advantage of the flexibility of AS/400 message definitions, such as:

- Length of second-level text is no longer limited to 225 bytes.
- Insertion of data into messages is no longer restricted to character format or the order in which the data is passed.

Note: The pointer type format is not supported and may lead to unexpected results if used.

Supplying Default Responses for Messages

You can have the system automatically respond to displayed system and application messages using the RESPONSE procedure or the CHGMSGD command in conjunction with the NOHALT procedure. When the NOHALT severity is such that the severity of the message being sent is smaller, and a valid response exists in the message description, the system responds automatically to the message, rather than having the operator enter a response.

Note: For a message to have a default response, it must be in a message file. See “Unattended System Operation” on page 18-12 for more considerations when using default responses.

Some IBM-supplied displayed messages have default responses and severity levels assigned. You can use either the RESPONSE procedure or the CHGMSGD command to change these values. For your application’s displayed messages, assign your own default responses and severity levels.

The RESPONSE procedure (also called automatic response function) allows the N option so you can set the current response back to its initial value. This option can be used for either IBM-supplied messages, or your messages, but you should be aware that on System/36, N is ignored for your application messages.

When you use the RESPONSE procedure to define or change a default response for a message, you specify:

- **The MIC of the message to be responded to.** If the message is one of your own application messages, you must specify the message member (message file) containing the messages. If the message is an IBM-supplied message, the system automatically determines the message file.
- **The response to use.** The default response you choose is valid only if the message allows that response when it is sent. For example, a message is sent allowing options 2 and 3. A

default response of 1 is not valid and results in requiring the operator to respond. You cannot specify the H (Help) option as a default response.

- **The severity level for the response.** See “Severity Levels.”

Default Response Process

When you create or change the default response for a message, the following actions take place:

- **Initial default responses.** The message is checked to determine if it already has an initial response value. If not, the value you set will become the initial default response. Later, if this value is changed, you can set it back to this initial response value by specifying a response of N.
- **Default responses and subsequent changes to default response.** For AS/400 message files, a special message description is created when an initial response is first changed. The description retains the initial default response. The message identifier is the same identifier as the message except that the third character of the identifier is replaced by a 0. For example, if the message is USR5412, the initial default response is stored in message US05412. You can display or change this initial default response message using the CHGMSGD CL command. When you delete a message containing the initial default response (such as US05412), the current response (in USR5412) becomes the initial default response. When you use the response procedure and the message file contains DBCS and non-DBCS messages, the initial response value saved in the special description is the response value of the non-DBCS message. Changing the response value updates both message descriptions.

Severity Levels

When you assign a default response to your messages, you must also specify a severity level. The system uses these severity levels to determine if messages are to be automatically responded to. For example, you can allow a default response for messages with a severity of 3 or lower, but require a manual response for messages with severity of 4 and 5.

On System/36 and in the System/36 environment, severities range from 0 to 5. The following list suggests severity levels for different categories of messages. IBM uses these severity levels for system messages.

When assigning default response for System/36 messages, use the following information as a guideline:

Severity Level	Explanation
0	No severity level.
1	Informational messages that require a response (option 0 only).
2	Messages with one option, such as a warning message. Messages with two or more options in which one of the options is to try the function again.
3	Program error messages. These messages usually have more than one option for the operator to choose.
4	Messages for severe errors, such as device errors or permanent input/output errors.
5	No default response can be defined for the message (requires operator intervention).

How to Assign Severity Levels: When you use the AS/400 Create System/36 Message File (CRTS36MSGF) command to create a message file, the severity level default is 0. For System/36 messages, use the following operation control language (OCL) statements to assign severity level:

```
// NOHALT 2,JOB
// MEMBER USER1-MSGDISP,LIBRARY-INVLIB
// LOAD INVPROG,INVLIB
// RUN
```

The NOHALT OCL statement specifies a severity level of 2 for the job. The MEMBER OCL statement specifies the message member to use (MSGDISP) and the library containing the member (INVLIB). If the program (INVPROG) displays a message with a default response that matches one of the displayed options, and if the severity level of that message is 1 or 2, the system automatically responds to the message.

Note: In this example, any system message with a severity level of 1 or 2 also receives the default response.

See the *System/36 Environment Reference* book for complete information about using the RESPONSE procedure to assign default responses and severity levels, and about using the NOHALT procedure or (OCL) statements to specify a severity level for a job, a session, or the system.

Mapping Severity Codes from the System/36 Environment to the AS/400 System:

The **severity code** is a number that indicates how important a message is. The higher the number, the more serious the condition. When severities are mapped from the System/36 environment to the AS/400 system, they are changed from 0 through 5 to 00 through 99. The severity field in the message description is used to keep the severity level. Severity 99 identifies messages as requiring operator intervention. The following table shows how System/36 environment severity levels are mapped to AS/400 severity levels:

System/36 Environment	AS/400 System
0	00
1	10
2	20
3	30
4	40
5	99

Mapping Default Responses and Severity Levels from the AS/400 System to the System/36 Environment:

If a message has a default reply that is not allowed by the System/36 environment (valid replies are 0, 1, 2, or 3), and this message is issued through the System/36 environment, the System/36 environment will not automatically respond to the message. Instead, the System/36 environment displays the message and waits for the user's response to the message.

The OS/400 system supports 100 message severity levels (0 through 99), and the System/36 environment supports six severity levels (0 through 5). The following table shows how OS/400 message severity levels are mapped to System/36 environment severity levels:

AS/400 System	System/36 Environment
00 to 09	0
10 to 19	1
20 to 29	2
30 to 39	3
40 to 98	4
99	5

Considerations for Default Responses to Messages

Consider the following factors when using default responses:

- The message and the default response of the system are written to the job log. The message is not displayed at the display station.
- Do not create a default response to system messages indicating that the system is trying an operation again. For example, if invalid data is found on a diskette, a user can select an option to try reading the diskette again. If you specify the retry option as a default response, the system repeatedly tries to read the diskette, without giving the user a chance to stop.
- Use caution when creating a default response to error messages that are informational (such as a PAUSE message). If you want an operator to see the message, the message should require a response.
- Do not create a default response to messages that require the user to do something before responding to the message. For example, if you specify a default response to the message that indicates that the forms are to be aligned, the user does not have an opportunity to align the forms.
- Do not create a default response to messages that indicate serious problems with the system, such as device errors and messages indicating that a service representative should be called.

The *System/36 Environment Reference* book lists additional considerations for using the RESPONSE procedure to set up default responses.

Displaying Response Messages

Work stations display System/36 environment messages on the System/36 Program Messages display. The following shows the format of the System/36 Program Messages display, with three information messages followed by an error response:

```

S/36 Program Messages

HELP CATALOG
CATALOG ALL,F2,,NAME
CATALOG procedure is running

SYS1690 Options ( 3H)
Unit parameter must be F1, I1, T1, T2, TC or null.
-

```

You can respond on this display, or request additional message information by pressing the Enter key or the Help key.

AS/400 Message Help: The AS/400 system handles response messages much like System/36. The AS/400 system supplies help information message text (second-level text) for OS/400 and System/36 environment messages. The message help displayed in the System/36 environment, is formatted similarly to the System/36 message help. The following display is an example of a System/36 environment Message Help display:

```

Additional Message Information

SYS1690 Options ( 3H)
Unit parameter must be F1, I1, T1, T2, TC or null.

Cause . . . . . : An incorrect unit parameter was specified in the
CATALOG procedure statement. Enter option 3 and give the programmer the
message ID(SYS--1690).
Possible choices for replying to message . . . . . :
3 -- The job is canceled. Data created by previous steps in this job
is saved, but data created by this step is lost.
D -- This option is available whenever option 3 appears on the display;
however, option D never appears on the display. When you enter option
D, a Job Process Dump will be taken. The system actions described for
option 3 occur.
H -- This option is available whenever option 3 appears on the display
and you are entering information from a Help prompt. When you enter
More...

The allowed options are listed in parentheses.
Option . . . . . : _

F10=Display Job Log

```

On the first Additional Message Information display, the error message is repeated on lines 3

and 4. Lines 6 through 18 contain the beginning of message help text.

You can use F10 from the System/36 environment Message Help display to view the job log for your job. The job log is a list of procedures and commands you have run along with any messages issued by these procedures or commands. From the Job Log display you can do the following:

- Press F10 to display low-level messages. Low-level messages are normally used to qualify other error messages.
- Position the cursor on a message and press the Help key to display AS/400 message help text.
- Press the Enter key to return to the System/36 environment Message Help display.

You can press the Page Down key from the System/36 environment Additional Message Information display to see the next page of message help:

```

Additional Message Information
SYS1690 Options ( 3H)
displayed again and the cursor is positioned at the field where the
error occurred. You can then correct the error and continue
your job.

The allowed options are listed in parentheses.
Option . . . . . : _
F10=Display job Log
Bottom

```

To return to the first page of message help, press the Page Up key.

After you respond to the message, the System/36 Program Messages display may appear again, but the error message does not appear.

```

S/36 Program Messages

HELP CATALOG
CATALOG ALL,F2,,,NAME
CATALOG procedure is running

```

Displaying Informational and Prompting Messages

The following display shows an example of two informational messages followed by a prompting message, generated by a procedure named TESTMSG:

```

S/36 Program Messages

TESTMSG
This is the first informational message
This is the second informational message
Enter required parameter to continue
-

```

If the user enters ABCD in response to *Enter required parameter to continue*, the system continues to run the procedure, and displays an additional informational message, as shown in the following display:

```

S/36 Program Messages

TESTMSG
This is the first informational message
This is the second informational message
Enter required parameter to continue
ABCD
This is the third informational message

```

Note: When you respond to a prompting message, the message and response are rolled up the display along with the informational messages. Informational and response messages roll off the top of the display. Error messages disappear after a valid response has been entered.

Formatting Messages with Control Characters

System and user messages are formatted through control characters in the message description. Use the ADDMSGD or CHGMSGD commands to enter the format control characters in the help information message text. See the *CL Reference* book for information about these commands.

The format control characters are as follows:

&N &N, followed by one space, forces the text to a new line. If the text is longer than one line, the next lines are indented two characters until the system finds another format character or the text ends.

&P &P, followed by one space, forces the text to a new line indented by four characters. If the text is longer than one line, the next lines are only indented two characters until the system finds another format character or the text ends.

&B &B, followed by one space, forces the text to a new line and indents it by two characters. If the text is longer than one line, the next lines are indented by two more characters until the system finds another format character or the text ends.

System Operator Displays

Operator intervention messages and messages from noninteractive jobs in the System/36 environment are sent to the system operator message queue (QSYSOPR).

User Authority to the System Operator Message Queue

The system operator message queue (QSYSOPR) is created on your system with public (*PUBLIC) authority defined as *OPR, *READ, and *ADD. Consider the following when changing the public authority for the system operator message queue:

- Users must have at least *OPR and *ADD authority to the message queue for their interactive and batch jobs to send messages to it. Do not change *PUBLIC authority to anything less than this.
- Users must have at least *OPR, *READ, and *ADD authority to the message queue to read and respond to messages on it.

See “Resource Security” on page 11-4 for more information about securing resources.

Sending System/36 Environment Messages

System/36 environment messages may be sent to the system operator message queue (QSYSOPR). See the *CL Programming* book for more information on the system operator message queue.

Following are examples of displays with a System/36 error message (message SYS1631) that has been sent to the system operator and is waiting for the operator to respond. The Work with Messages display initially appears when the Basic Assistance level is being used:

```

Work with Messages                               System:  RCHSX33
Messages in:  QSYSOPR
Type options below, then press Enter.
4=Remove    5=Display details and reply

Opt  Message
-----
-   SYS1631 Options ( 123)
    Empty slot or picker failure during DELETE.
-   Controller CTL02 contact not successful. Probable remote station
    problem. (C R)

Messages needing a reply
-   Subsystem QSNADS ended.
-   Subsystem QSNADS ending in progress.
-   Device DSP25 no longer communicating.
-   Device DSP12 no longer communicating.
-   Controller CTL02 failed. Automatic recovery started.
More...
F1=Help    F3=Exit    F5=Refresh    F16=Remove messages not needing a reply
F17=Top    F10=Bottom  F24=More keys

```

To view the Additional Message Information display and reply to the message shown on the Work with Messages display, type a 5 in the *Opt* column and press the Enter key. The following

display appears when the Basic Assistance level is being used:

```

Additional Message Information
Message ID . . . . . : SSP0337
Date sent . . . . . : 11/14/91      Time sent . . . . . : 14:29:14
Message . . . . . : SYS1631 Options ( 123)
                    Empty slot or picker failure during DELETE.
Cause . . . . . : You were running the DELETE procedure or the $DELETE
                    utility to delete information from a diskette when one of the following
                    conditions occurred:
                    -- The diskette drive was empty.
                    -- The diskette was not properly inserted into the diskette drive.
                    -- A hardware problem exists.
Recovery . . . . . : Do one of the following:
                    -- Enter option 0, and continue the operation with the next diskette slot
                    through the specified end location.
Type reply below, then press Enter.
Reply . . . . .
F1=Help  F3=Exit  F6=Print  F9=Display message details  F12=Cancel
F21=Select assistance level

```

The Display Messages display initially appears when the Intermediate Assistance level is being used:

```

Display Messages
Queue . . . . . : QSYSOPR      System: RCHSX33
Library . . . . : QSYS        Program . . . : +DSPMSG
Severity . . . . : 90         Delivery . . . : *HOLD
Type reply (if required), press Enter.
Writer 055228/QSPJOB/PRT01 started.
Device PRT02 no longer communicating.
Controller CTL02 no longer communicating. Automatic recovery started.
Device DSP12 no longer communicating.
Device DSP25 no longer communicating.
Subsystem QSNADS ending in progress.
Subsystem QSNADS ended.
Controller CTL02 contact not successful. Probable remote station
problem. (C R)
Reply . . . . .
SYS1631 Options ( 123)
Empty slot or picker failure during DELETE.
Reply . . . . .
Bottom
F3=Exit      F11=Remove a message      F12=Cancel
F13=Remove all  F16=Remove all except unanswered  F24=More keys

```

To view the Additional Message Information display for this display, position the cursor at a message and press the Help key. The following display appears when the Intermediate Assistance level is being used:

```

Additional Message Information
Message ID . . . . . : SSP0337      Severity . . . . . : 99
Message type . . . . : Inquiry
Date sent . . . . . : 11/14/91      Time sent . . . . . : 14:29:14
Message . . . . . : SYS1631 Options ( 123)
                    Empty slot or picker failure during DELETE.
Cause . . . . . : You were running the DELETE procedure or the $DELETE
                    utility to delete information from a diskette when one of the following
                    conditions occurred:
                    -- The diskette drive was empty.
                    -- The diskette was not properly inserted into the diskette drive.
                    -- A hardware problem exists.
Recovery . . . . . : Do one of the following:
                    -- Enter option 0, and continue the operation with the next diskette slot
                    through the specified end location.
Type reply below, then press Enter.
Reply . . . . .
F3=Exit  F6=Print  F9=Display message details  F12=Cancel
F21=Select assistance level

```

When you press F9 on either of the Additional Message Information displays, the following display appears, providing more information:

```

Display Message Details
Message ID . . . . . : SSP0337      Severity . . . . . : 99
Date sent . . . . . : 11/14/91      Time sent . . . . . : 14:29:14
Message type . . . . : Inquiry
CCSID . . . . . : 65535
From job . . . . . : DSP04
User . . . . . : USER7
Number . . . . . : 010118
From program . . . . . : QEXCLSG
To message queue . . . . . : QSYSOPR
Library . . . . . : QSYS
Press Enter to continue.
Bottom
F1=Help  F3=Exit  F12=Cancel

```

When you press F16 on either the Work with Messages display or the Display Messages display, all messages on the message queue are removed except unanswered inquiry messages. The displays appear again with the unanswered inquiry messages.

When no unanswered messages are on the message queue and you press F16, all messages are removed and the displays appear with the phrase No messages available.

When you press F13 (Remove all) on the Display Messages display, the Display Messages display appears again with the phrase No messages available.

Embedding Messages: In the preceding example, the message actually sent to the QSYSOPR message queue is SSP0335 not the SYS1631 message. This is because messages sent to the system operator from the System/36 environment are sent as replacement text in other messages in the QSSPMMSG file. This concept is called **message embedding**, and the messages used are:

- SSP0303 and SSP0305 for informational messages
- SSP0311 through SSP0313 for prompting messages
- SSP0335 through SSP0357 for error messages

Embedding of messages is done for special response handling and verification. Reply list entries for System/36 environment messages must

specify these special message identifiers as the message being sent and use the System/36 environment error message identifier (the first 7 bytes of the message text) as the comparison data. For example, if you want to automatically respond to the message shown in the previous example with a 3 response, enter following command to set up the reply list:

```
ADDRPYLE SEQNBR(50) MSGID(SSP0335) CMPDTA(SYS1631 1) RPY(3)
```

The message sent to the system operator is SSP0335, but the System/36 environment message identifier is SYS1631. In this case SEQNBR is set to 50. This is the relative position of the entry in the reply list.

Notes:

1. The System/36 environment message identifier is provided as the first 7 characters for SSP0335-SSP0349. The text displayed on the screen for SSP0311-SSP0313 starts in the first character text position of the message and can be used to define comparison values for reply list entries for the prompting messages. If you send messages that are both DBCS and non-DBCS from the same message file, you may see both message identifiers. For example, the double-byte version of USR1234 is USZ1234.
2. IBM DBCS and non-DBCS messages are not stored in the same message file, so the message identifiers for IBM DBCS and non-DBCS messages are both SYS1631.
3. Auto response for System/36 environment messages is also valid for messages sent to QSYSOPR. When the RESPONSE procedure is used to set up auto response values, use the original message identifier (for example, SYS1631).

When System/36 environment messages are embedded in other messages, the actual text of the message is retrieved from the message file before the message is sent. When the operator displays the message, the System/36 environment messages are displayed in the language of the sender of the message and not the language of the person requesting to display the messages. Other AS/400 messages stored in message files appear in the language of the user displaying the system operator message queue.

See “System/36 Environment Double-Byte Character Support” on page 20-3 for more information on DBCS messages.

Handling Defaults for System Operator Messages

Defaults for messages SSP0335-SSP0349 are preset to the highest possible response (to avoid repeating retry requests). You may change this setting as you feel appropriate, but they are reset each time you install a new release. If you set the value to an invalid response and the system operator message queue is in default mode, the system still responds to the message, with the highest response value.

Sending Messages

The System/36 environment supports the MSG operator control command, which allows you to send a message to the operator, another user, a work station, a personal computer location, or through Systems Network Architecture Distribution Services (SNADS) to a user or group of users in a network. Also, if you specify the command with no parameters, any waiting messages for you or your work station appear. The system places messages sent with this command in AS/400 message queues.

All AS/400 work stations and personal computer locations have a message queue. All users have a message queue defined in their user profile. The System/36 environment sends user-directed messages to the message queue associated with the user's profile. If the queue does not exist, the system tells the sender why the message is not delivered. For example, Message queue not found.

When you ask to see waiting messages, the System/36 environment uses AS/400 support to display the user and work station message queues.

Message Handling Considerations

If the message handler encounters an unexpected error, it signals exceptions to its caller so that messages may not be displayed. If it encounters an input/output error, it makes two attempts to

display the message. If the message still cannot be displayed, the message handler closes the System/36 environment system display files and places an SSP0309 message in the job log. It then goes to the end of the job. If the job continues and messages can again be displayed, any prior informational messages and prompting messages that were on the display no longer appear. For more information about the message handler, see the *CL Programming* book.

Error Messages Not Displayed

In some situations, when you attempt to display an error message, the message cannot be displayed. One of the following conditions occurs:

- **Original message not found.** If a requested message cannot be found, the message identifier, name of the message file, and message file library name are displayed. The message not found implies that the library was not found, the message file in the library was not found, or the message ID was not found in the message file. It is possible to request the display of a user or product message before specifying the user or message file on a member statement. When this happens, the message file name and message file library name output is *N/*N.
- **Not authorized to the message file.** If you do not have sufficient authority to read the message from the message file, a message appears that gives the message identifier and the name of the message file that you are not authorized to. A security officer or an authorized user must grant you the authority to display the message. After you have been authorized to the message file, you can use the DSPMSGD CL command to display the message.
- **Message is not accessible.** If an unexpected error occurs while attempting to obtain the message, a message appears, stating that the system encountered a problem while attempting to display the message. This message identifies the message file and the message identifier. You can attempt to display the message using AS/400 commands.

To determine what the original error was:

- Request additional message text by pressing the Enter key.
- Request to see the job log by pressing F10.
- From the Display Job Log display, page until you see the message that appeared on your display.

The messages immediately preceding this message identify the original cause of the access error.

- **The message text does not appear (only the identifier and message file name appear on display).** If an error occurs and the IBM supplied message for the error condition cannot be found, or there is an authorization or access problem, you will see only the message identifier and message file name and library name on the screen for the message text. The message line that normally identifies the options will not display the word *Options* or the parentheses. The option values and the message identifier are displayed.

Problems with the Operator Messages

When a message is sent to the System Operator message queue (QSYSOPR), it is possible that the System/36 environment message cannot be located or accessed. When this condition occurs, an AS/400 message is sent to the System Operator message queue, which explains that the message could not be used to display the message. The embedded System/36 environment message does not appear.

To see the System/36 environment message and your reply options, you must first determine the identity of the job that sent the message. If you are using the Basic Assistance level, type a 5 in the *Opt* column for the message on the Work with Messages display and press the Enter key. The Additional Message Information display appears. If you are using the Intermediate Assistance level, position the cursor on the message located on the Display Messages display and press the Help key. The Additional Message Information display appears. Press F11 to display message details and copy the job, user, and number from the Display Message Details display.

Use the job, user, and number with the DSPJOBLOG CL command to display the job log of the job which sent the message to the System Operator message queue. The error message you are looking for should be near the end of this job log. It will have an Options line followed by the text of the message, but it will not show a response. Immediately following this error message will be the AS/400 message that appeared in the System Operator message queue.

Console (System) Operator Messages

System/36 messages are sent to the console, a subconsole, or a work station. In the System/36 environment, the concept of a console has been replaced with the QSYSOPR message queue. There is no direct support for subconsoles. The System/36 environment uses message queues associated with each printer instead of subconsoles.

See Chapter 2, "Operating in the System/36 Environment," for more information.

Automatic Reply Handling When QSYSOPR Is in Default Mode

On the AS/400 system, you can place the operator queue in default mode (*DFT). In default mode, all messages sent to the operator receive an automatic response, even if no valid response exists for the message. When no valid response exists, the AS/400 system returns *N to the requesting program. For System/36, the special set of operator messages (SSP0335-SSP0349) used for sending error messages has the highest response allowed as the default reply values for that message. Even if the response is deleted from these messages, the System/36 environment handles *N as though these replies were still defined in the message. If the deletion of the response is not acceptable, use one of the following options:

- Do not set the operator queue in default mode (this is the safest option).
- Change the replies on these messages.
- Ensure that default responses are defined for any critical message.
- Use the AS/400 message reply list function.

Dual-Routed Messages

The System/36 environment uses AS/400 support for dual routing of messages. When an inquiry message is sent to a message queue from an interactive or multiple requester terminal (MRT) job, a notification message is sent to the requester (and all MRT displays for MRT jobs). This notification tells you that your job is waiting on a message and tells you the queue to which the inquiry message has been sent.

The messages display may be automatically presented, or the waiting light and audible alarm may be activated to prompt you to display your device message queue. By placing the cursor on the notification message and pressing the Help key, the Additional Message Information display appears. This display contains the actual text of the inquiry message.

You may also choose to respond to the inquiry message. By displaying the notification message, you can determine which queue the inquiry message was sent. Press the System Request key and enter a 4 followed by a space and the queue name. The requested message queue is displayed and you can respond to the message.

Enhancements and Restrictions

The following sections describe some enhancements and restrictions regarding message handling.

User, Device and Operator Message Enhancements: On System/36, user and device messages sent to the message file and operator were limited to no more than 25 messages. This restriction no longer applies. In the System/36 environment you also have the option of holding messages (this is the default). Paging a message off the screen no longer automatically deletes it.

Because the System/36 environment uses AS/400 message queues, you can also use AS/400 message commands to send messages to a user or device queue.

You do not have to wait until the end of a job to see messages that have been sent, or interrupt the job to display messages that have been sent. Use the AS/400 CHGMSGQ CL command to place the message queue in break mode. When

a message of correct severity is sent, the message is automatically displayed.

User, Device, and Operator Message

Restrictions: This section discusses the restrictions on user, device, and operator messages.

Message Notification

If you sign on the system twice, the user queue is placed in notify mode for the first sign-on only. A queue can be in notify mode in only one job at a time.

Although a message queue is in notify mode, you may not always be notified when a message arrives on the queue because the queue has a severity threshold. You are notified only for messages that exceed the threshold. AS/400 commands control the threshold values. See the description of the CHGMSGD and CHGMSGQ CL commands in the *CL Reference* book for more information.

External Message Queue

If you are in the System/36 environment, you may see a new display: the AS/400 Display Program Message display. Every job has an external message queue. When it is used to output a message it interrupts any other display. Messages may or may not require a response. For example, run-time messages sent from RPG III appear on this display. Corresponding RPG II messages are presented on the System/36 Program Messages display, so recompiling an RPG II module to RPG III can change the display used to display messages.

Programming Guidelines

This section describes how you can create, change, and use message members (AS/400 message files).

Creating or Changing Message Source Members

Use the programming development manager (PDM) or source entry utility (SEU) to create the new source members.

If you want to change or add messages:

1. Use PDM to change the message source member.
2. Use the CREATE procedure or the Create System/36 Message File (CRTS36MSGF) CL command to create or replace a message file.

Note: To update a message file, use either the Create Message File (CRTMSGF), Add Message Description (ADDMSGD), or Change Message Description (CHGMSGD) CL commands. See the *CL Reference* book for more information on these commands.

Assigning Default Responses and Severity Levels

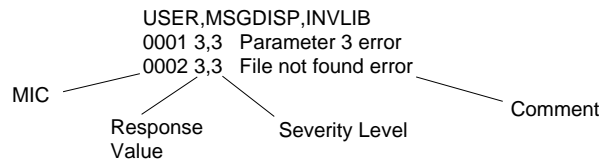
To assign default responses to messages in a message file, use the System/36 environment RESPONSE procedure (use PDM to create response source member).

Figure 15-1 shows a message member and Figure 15-2 shows its corresponding response source member for a sample application. The library INVLIB contains the message member MSGDISP. The response source member specifies default responses for two messages, MIC 0001 and MIC 0002, and it specifies that the MICs are in message member MSGDISP in library INVLIB.

```
MSGDISP,1
0001 3,3 Parameter 3 must be SALES or CREDIT
0002 3,3 File INVMST is not on the disk
```

RSLW072-0

Figure 15-1. Message Member MSGDISP



RSLW067-2

Figure 15-2. Response Source Member for MSGDISP

After the response source member is created, use the RESPONSE procedure to assign the default responses and severity levels to the message member. You must run the NOHALT procedure or OCL before the RESPONSE procedure is effective.

If you want to change a response or severity level:

- Use PDM or SEU to change the response source member.
- Use the RESPONSE procedure to assign the new values.

Notes:

1. If you recompile the message source member (by using the CREATE procedure or CRTS36MSG CL command), the default responses no longer apply. You must run the RESPONSE procedure again to set the default responses and severity levels.
2. You can also use the Change Message Description (CHGMSGD) CL command to assign default responses for messages. See the *CL Reference* book for more information on the CHGMSGD CL command.

Specifying a Message Member to Be Used within a Procedure

The MEMBER OCL statement assigns message members to a job. The messages in the assigned member can be used in the procedure, by programs run by the procedure, and by display formats. For example, to assign message member DISPMSG from library INVLIB to the program PROG1, enter:

```
// MEMBER USER1-DISPMSG,LIBRARY-INVLIB
// LOAD PROG1
// RUN
```

Message Member and Message File Considerations

Consider the following points when using message members and files in the System/36 environment:

- The text for a message will be retrieved from a message file (message member). The message file can be specified in several ways depending on how the message is to be displayed or retrieved:
 - If the message is to be retrieved by a program, it will always be retrieved from the USER1 message member. The message file name must be specified on a MEMBER OCL statement on the USER1 parameter.

- If the message is to be displayed using a display file and the display file was created using SFGR specifications, the message member is specified when the display file is defined or passed to the display file by a program or a PROMPT OCL statement. The values that can be specified or passed are:

U1	User-1 message member
U2	User-2 message member
P1	Program-1 message member
P2	Program-2 message member
M1	SSP level-1 message member (##MSG1)
M2	SSP level-2 message member (##MSG1)

- If the message is to be displayed using a display file and the display file was created using Data Definition Specifications (DDS), the message member is specified when the display file is defined or passed to the display file by a program or a PROMPT OCL statement. The values that can be specified or passed are:

U1	User-1 message member
U2	User-2 message member
P1	Program-1 message member
P2	Program-2 message member
M1	SSP level-1 message member (##MSG1)
M2	SSP level-2 message member (##MSG1)

*USR1	User-1 message member
*USR2	User-2 message member
*PGM1	Program-1 message member
*PGM2	Program-2 message member
*SYS1	SSP level-1 message member (##MSG1)
*SYS2	SSP level-2 message member (##MSG1)

- If a PROMPT OCL statement is used to display the display file, the message file name must be specified in a MEMBER OCL statement on one of the following

parameters: USER1, USER2, PROGRAM1, or PROGRAM2.

- If a program is used to display the display file, and the program was given control with the // LOAD OCL statement, the message file name must be specified on a MEMBER OCL statement on one of the following parameters: USER1, USER2, PROGRAM1, or PROGRAM2.
 - If a program is used to display the display file, and the program was given control with a CALL CL command, the message file name and library must be specified on an Override Message File (OVRMSGF) command. The message file being overridden must be: USR1, USR2, PGM1, PGM2, SYS1, or SYS2.
- IBM system message and second-level text message members are contained in a single AS/400 message file.
 - You can define first and second level text for your messages as separate AS/400 message files. If you decide to merge the members into a single message file, update the MEMBER OCL statement accordingly.
 - IBM system message members containing DBCS messages have been split into DBCS and non-DBCS versions of the message file (placed in different libraries).
 - You can split message members (containing DBCS messages) into DBCS and non-DBCS versions of the message file (placed in different libraries), or have DBCS and non-DBCS messages in the same files.
 - When you use the CREATE procedure to add a message to an AS/400 message file (a System/36 message member), the AS/400 message ID prefix uses the USR default value. For example, a MIC of 0123 becomes message ID USR0123.
Note: You can successfully specify a different set of 3 characters for the message ID prefix if all messages in the message file have the same prefix.
 - If you add DBCS messages to an AS/400 message file, the third character of the AS/400 message ID must be changed to a Z. For example, a MIC of 0123 becomes the AS/400 message ID USZ0123.

- When you are using a DBCS-capable display station and a user MIC of 0123 is being displayed during a DBCS session, the system uses the message ID USZ0123 to try to retrieve the text. If it does not find this message ID, the system tries again using message ID USR0123.
- When the device is neither DBCS-capable nor a DBCS session, and a user MIC of 0123 is being displayed, the system does not alter the message ID prefix when retrieving the message text (USR0123 is the only message for which the system searches).
- See Chapter 20, “System/36 Environment National Language Support,” for information on national language support for messages.

Displaying Messages from Procedures

You can use procedure control expressions and procedures to display messages from procedures.

// * (Informational Message) Statement: The informational message statement displays a message from a procedure to the operator running the procedure. For example, the following statement:

```
// * 'Enter today''s date:'
```

displays the message:

```
Enter today's date:
```

// ** (Console Message) Statement: The console message statement sends a message from a procedure to the QSYSOPR message queue. For example, the following statement:

```
// ** 'Procedure PROC1 is running'
```

displays the message:

```
Procedure PROC1 is running
```

Displaying Your Messages in the Same Format as System Messages:

The ERR procedure displays your error messages in the same format as system-displayed messages. For example, if message 0001 in message member DISPMSG in library INVLIB is Parameter 3 is invalid, and the following statements are issued:

```
// MEMBER USER1-DISPMMSG,LIBRARY-INVLIB  
ERR 0001,0123
```

the ERR procedure displays the following message:

```
USR0001 (0123)  
Parameter 3 is invalid
```

For options 0, 1, and 2, the system sets a return code that you can test using the ?CD? substitution expression. For option 3, the job is immediately canceled. The *System/36 Environment Reference* book describes the ?CD? substitution expressions.

Checking Entries for Required Parameters

The ?nR'mic'? substitution expression displays a message when the nth parameter does not have a value. Use this expression when you are checking the parameters of a procedure to make sure that the operator has entered a required parameter. In the following example, the EVALUATE statement processes the substitution expression:

```
// MEMBER USER1-DISPMMSG,LIBRARY-INVLIB  
// EVALUATE ?1R'0004'?  
:
```

If the first parameter was not entered when the operator started the procedure, message 0004 from message member DISPMMSG would appear.

The operator receives a prompt to enter the parameter.

Using Messages with Programs

User-written programs can issue System/36 environment messages. The text for the message can be defined in the program or in a message member. See the correct language manual for more information.

Using Messages with Displays

User display formats can display System/36 environment error messages. The text for the message can be defined in a program, procedure, or message member. See "Displays" on page 14-10 for more information.

Moving from System/36 to the System/36 Environment

If you have only a System/36 message load member, you can use the MSG2SRC migration tool to create message source files on System/36. The MSG2SRC tool converts message load members in a library back to a source members. The source member can then be migrated to the System/36 environment on the AS/400 system. For more information, see the *System/36 Migration Assistant* book.

Chapter 16. Programs and Procedures

This chapter describes the types of programs and procedures you can design in the System/36 environment.

Designing Programs and Procedures

The types of programs and procedures you design depend on the following factors:

- Structure of the application
- Use of the program
- Number of users
- Numbers and types of displays required
- Backup and recovery methods
- Type of job to run

In the System/36 environment, there are special considerations that affect how you apply these factors and design programs and procedures.

Programs

This section describes the types of programs you can design, the reasons for choosing each type, and suggestions for designing applications.

Batch and Interactive Programs

A **batch program** processes records without operator interaction. A batch program processes a group of related transactions that have accumulated over a given period of time. For example, a batch program is one that prints invoices at the end of the day, rather than when the order is entered.

An **interactive program** is one that receives requests from one or more display stations or ICF sessions and may respond to each request as it is received. The program processes individual records or transactions at the time the request is received, rather than processing requests that accumulate over a period of time. An example of

an interactive program is one that processes orders and prints an invoice at the time each order is entered.

Interactive programs use a display file to communicate with a user or an ICF file to communicate with a remote system. A display station file or ICF file is called different names in each programming language. For example:

Programming Language	Display File or ICF File
COBOL 74	TRANSACTION file
System/36-Compatible RPG II	WORKSTN file

For more information on using a display file or ICF file with a particular programming language, refer to an appropriate language reference or user guide. For more information on ICF, refer to Chapter 13, "Communications."

Most applications include both interactive and batch programs. The following list shows typical uses of interactive and batch programs for order entry, accounts receivable, and inventory control applications:

Order Entry Applications	Program Type
Order entry	Interactive
Open order inquiry	Interactive
Inventory allocation	Interactive or batch
Print invoices	Interactive or batch

Accounts Receivable Applications	Program Type
Cash receipts	Interactive
Account status inquiry	Interactive
Open items	Interactive or batch
Monthly statements	Batch

Inventory Control Applications	Program Type
Receipts/adjustments	Interactive
Status inquiry	Interactive
Vendor code changes	Interactive or batch
Parts requisition	Batch

Program Characteristics

When you use interactive programs, the size of the program, the number of non-requesting users that can communicate with the program (acquired display stations or ICF session), and the number of users that can request the program (requesters) affect your program and application design.

Number of Users: Programs can communicate with any number of display station users, requesters, acquired display stations or ICF sessions. An acquired display station or ICF session cannot call the program.

In System/36-Compatible RPG II, you can limit the number of users to one by specifying a value of 1 for the NUM continuation-line option on the WORKSTN file description specification. In other programming languages, you cannot limit the number of users.

The system supports the following types of programs that restrict the number of users:

- One-user programs
- Multiple-user programs
- No-user programs

One-User Programs: A one-user program is an interactive program that communicates with only one user at a time. A one-user program has a display file that is limited to one display station or an ICF file for a single ICF session.

When a one-user program is a SRT program, the requester is the only display station or ICF session with which the program can communicate.

Multiple-User Programs: A multiple-user program is an interactive program that can communicate with more than one user at a time. A multiple-user program, like a one-user program, may have a display station file or an ICF file or both. However, the display station file and ICF file for a multiple-user program allow two or more users. Those users can be requesting display stations, ICF sessions, acquired display stations, or acquired ICF sessions.

No-User Programs: A no-user program is a batch program because it does not have a display file or ICF file. An example of a no-user program is a program that prints a disk file.

You also can specify the number of devices attached to display files.

MAXDEV Value: The MAXDEV value for display files specifies the maximum number of devices that can be attached to a particular display file. MAXDEV applies only to display files. ICF files have a corresponding MAXPGMDEV parameter.

Use the Change Display File (CHGDSPF) CL command to change the MAXDEV value of a display file. Use the Change ICF (CHGICFF) CL command to change the value for an ICF file.

Program Types

Program type is determined by the number of users who can request a program. Users who call the program are called **requesters**. Users who are acquired by the program are called **acquired display stations** or **acquired ICF sessions**.

The number of requesters affects the application design because it affects the operation control language (OCL) statements and procedures you use to call the programs in the application. If a program has more than one requester, you must use a MRT procedure to call that program. The system supports the following types of programs:

- SRT programs
- MRT programs
- Nonrequester terminal (NRT) programs

Single Requester Terminal Programs:

A **single requester terminal (SRT) program** is a program that can process requests from only one display station or ICF session from each copy of the program. A SRT program can interact with only one requesting user. If multiple users request the same SRT program, the system uses the same copy of the program for all users but creates a separate area for variables used by the program for each user.

Specifying a SRT Program: A program not called as a NRT program, or specified as a MRT program, is a SRT program. No coding is required to specify that a program is a SRT.

Difference between a SRT Program and a One-User Program: In a SRT program, the user is always a requester. In a one-user program, the

user can be a requester, an acquired display station, or an acquired ICF session. A one-user program is usually a SRT program, but it could be a NRT program with one acquired display station or ICF session.

Multiple Requester Terminal

Programs: A MRT program is an interactive program that processes requests from more than one user at the same time. Similar to SRT programs, in a MRT program all users share the same copy of the MRT program. The MRT program is assigned an area of storage to hold variables for each of the display stations the MRT program communicates with. A MRT program uses a display station file or an ICF file.

Difference between MRT and Multiple-User

Programs: The difference between a MRT program and a multiple-user program lies in whether the users are requesters or are acquired.

A MRT program can have as many users as permitted by the display station file definition, the ICF file definition, the MRTMAX value for the program, and the NUM continuation-line option in RPG II WORKSTN file definitions.

A multiple-user program that is not a MRT can also have any number of users. However, no more than one of those users can be a requester. The other users must be acquired.

Specifying a MRT Program: Both the program and the procedure that calls the program must be specified as MRTs. You specify the program as a MRT by assigning a MRTMAX value when the program is compiled. You specify the procedure as a MRT using the Edit System/36 Procedure Attributes (EDTS36PRCA) or Change System/36 Procedure Attribute (CHGS36PRCA) control language (CL) commands, or the MRT parameter of the \$MAINT utility program. For information about MRT procedures, refer to “Procedures” on page 16-21.

MRTMAX Value: Use the MRTMAX value to specify the maximum number of requesters that can be active at once for a MRT program. When the number of requesters equals the MRTMAX value, a subsequent requester must wait until a current requester is finished using the program.

You can decrease the MRTMAX value when you run the program using the following:

- The Change System/36 Program Attributes (CHGS36PGMA) command
- The Edit System/36 Program Attributes (EDTS36PGMA) command
- The Work with System/36 Program Attributes (WRKS36PGMA) command
- The ATTR OCL statement

Recompile the program to increase the MRTMAX value.

A program must be able to handle any number of requesters, up to the MRTMAX value specified by the programmer.

Notes:

1. The MAXDEV attribute of a display file used by a MRT program must be set to the maximum number of display stations that are used by the MRT program. See the Create System/36 Display File (CRTS36DSPF), Change System/36 Source Attributes (CHGS36SRCA), and Change Display File (CHGDSPF) CL commands for additional information.
2. The MAXPGMDEV attribute of an ICF file used by a MRT program must be set to the maximum number of ICF sessions used by the MRT program. See the Create ICF File (CRTICFF) and Change ICF File (CHGICFF) CL commands in the *CL Reference* book for additional information.

NUM Value: The NUM value specifies the sum of the MRTMAX value plus the number of acquired display stations that can communicate with an RPG program. A System/36-Compatible RPG II programmer can control the number of users and requesters. For example, if a NUM value of 3 is specified, and the program has two requesters, no more than one display station can be acquired.

The NUM value is defined on the continuation line for the WORKSTN file on an RPG file description specification. In other programming languages, you must code the logic in your source program if you want to check the number of users.

Nonrequester Terminal Programs: A nonrequesting terminal (NRT) program in the System/36 environment, is a program that is not associated with a requesting display station. A NRT program has no requesters. A program becomes an NRT program when the requester purposely separates the program from the display station using a command or OCL statement. For example, if an EVOKE OCL statement calls a program, the system immediately separates the program from the requester. The user is free to do other work, and the program has no requesters.

A NRT program can also be a program that specified // ATTR RELEASE=YES before a // LOAD-//RUN pair. This job step is now a NRT and is released from a procedure.

Comparison of Program Types

The following table compares the characteristics of six types of programs:

Type of Program	Characteristics
One-user program	Designed to use only one display station or ICF session, which can be a requester or acquired. Can be a SRT, NRT, or MRT (MRTMAX = 1).
Single requester terminal program	When you do not specify program type, a program is a SRT. Can be a one-user program, a no-user program, or a multiple-user program.
Multiple user program	Users can be any combination of requesters, acquired display stations, or acquired ICF sessions. Can be a NRT, SRT, or MRT.
Multiple requester terminal program	Typically designed to communicate with more than one user at the same time (MRTMAX = 2 or more), but can be restricted to one user (MRTMAX = 1).
No-user program	Has no users and cannot acquire any. Can have a requester but cannot communicate with it. Can be a NRT or a SRT.
Nonrequester terminal program	Has no requesters but can acquire any number of users. Can be a one-user, multiple-user, or no-user program.

Summary Table of Users and Requesters

The following table lists the program types you would use based on the combinations of requesters, acquired display stations, and acquired ICF sessions:

Number of Users	Number of Requesters	Type of Program	Description
1	1	SRT	Most common situation.
1	1	MRT	MRTMAX = 1. No acquired display stations or ICF sessions.
1	0	NRT	Can acquire one display station or ICF session.
More than 1	1	SRT	In System/36-Compatible RPG II, can acquire up to NUM - 1 display station or ICF session. In other languages, can acquire any number of display stations or ICF sessions.
More than 1	More than 1	MRT	In System/36-Compatible RPG II, requesters + acquired display stations or ICF sessions ≤ NUM. In other languages, the total can be any number. Common MRT situation.
More than 1	0	NRT	In System/36-Compatible RPG II, can acquire up to NUM display stations or ICF sessions. In other languages, can acquire any number.
0	0	SRT	Common situation for batch programs. Cannot communicate with users.
0	0	MRT	Meaningless combination.
0	0	NRT	Cannot communicate with users.

Designing Applications

The following sections present information to help you select the correct program types.

Differences between Batch and Interactive Programs: With interactive programs you can produce immediate results, but it is difficult to determine whether the last transaction update actually occurred for each user. For example, you may want to update your inventory file as orders are entered. However, this interactive updating makes it difficult to recover an inventory master file if the system ends prematurely. If you used a batch program to update your files, recovery is easier. A batch program does not normally share files, so you know whether the last update was made.

Application Structure: On the AS/400 system, when a program is running, one copy of the program is kept in main storage. Each call of the program is allocated a separate area, which contains variables used by the program. The program and variables are divided into storage units called **pages**. When references are made to the pages of a program, the pages used by the program are copied from disk to main storage. When the pages in main storage are needed for another function, the pages of variables are copied from main storage to disk. The pages of a program are not copied from main storage to disk because the copy of the program on disk is the same as the copy of the program in main storage, and the pages containing the variables are kept separate from the program instructions.

When deciding if you should write a single large program or many small programs, consider the following factors:

- A small program is easier to design and maintain than a large program.
- A small program compiles faster than a large program.
- A significant amount of system overhead is used to end one program and load a different program using the // LOAD and // RUN OCL statements. The more programs called, the larger the system overhead to start and end the programs.

When deciding if you should write a program that processes one display or processes many display stations, consider the following:

- If a program processes many display stations or ICF sessions, the program can only process requests from one display station at a time. If a request is made from a display station when the program is already processing a request, the first request must be completed by the program before the second request can be started. This requirement may affect the response time of the requests. The response time will tend to increase as the number of display stations or ICF sessions processed by the program increases.
- If a program processes only one display station or ICF session and the program is running from multiple jobs, delays may occur if both programs need to allocate the same resources and the program cannot share the resources. For example, if the program requires exclusive use of a file, the second call of the program must wait until the first call is done with the file.

Differences between SRT and MRT

Programs: SRT programs are easier to write and maintain than MRT programs. If multiple SRT programs result in a considerable contention for resources, the application should be written as a single MRT program.

If you can write your MRT program as a **never-ending program** (NEP), there is much less system overhead when a display station or ICF session is attached to an already-running MRT program than when a new program is started. A MRT program that is a NEP program will usually perform better than multiple SRT programs or a MRT program that is not a NEP program.

Differences between Requesters and Acquired Display Stations:

If you decide that more than one user should be able to interact with the program, you must also decide how many users can request the program. Normally, all users are requesters. However, acquiring display stations is sometimes an advantage because acquiring them allows the application to control where its users are.

Program Attributes: A program can have the following user-assigned attributes:

- Never-ending
- Inquiry
- MRTMAX

Never-Ending Programs (NEP): When a MRT program is a NEP, it is called a MRT-NEP. A MRT-NEP is not instructed to go to the end of the program when the last device is released. The NEP is expected to run for a long time and does not know when it temporarily has no requesters.

Because the NEP attribute keeps a MRT program from ending, the MRT starts and ends only once, rather than each time someone requests the MRT program. If your MRT application does not have a requirement preventing it from being a NEP program, specify NEP to reduce the demands on the system. You can use the ATTR OCL statement to override the NEP attribute of a program.

When a MRT program is not a NEP and releases its last attached device, the system checks the MRTDLY attribute of the MRT procedure to determine whether the MRT should end immediately or delay ending for a short period of time in case a new requester attaches to the MRT.

To end a NEP program you can use the End Job (ENDJOB) or the End Subsystem (ENDSBS) CL commands.

If the MRTDLY attribute indicates that the MRT should delay ending, the MRT waits for the number of seconds specified for the System/36 environment before issuing a return code instructing the program to go to the end of the program.

If the MRTDLY attribute indicates that the MRT should not delay ending, the program immediately receives a return code instructing it to go to the end of the program. When the program ends, the MRT is no longer an active MRT. If another device requests the MRT, a new MRT is started.

You can change the attributes of the MRT procedure by using the Change System/36 Procedure Attributes (CHGS36PRCA) command, the Edit System/36 Procedure Attributes (EDTS36PRCA) command, or the Work with System/36 Procedure

Attributes (WRKS36PRCA) command. The attributes of the MRT procedure can be displayed with the EDTS36PRCA or WRKS36PRCA commands. You can display the number of seconds that the MRT waits before ending by using the Display System/36 (DSPS36) command, and change it by using the Change System/36 (CHGS36) command. You can also use the Work with System/36 Environment Configuration (WRKS36) command to display and change the number of seconds the MRI waits before ending.

Inquiry: The inquiry attribute allows you to specify whether the program can be interrupted so another program can be run from the same display station. The inquiry attribute specifies whether System Request menu option 1, which creates a new session, is permitted.

You can use the ATTR OCL statement to specify the inquiry attribute. See “Preventing Users from Ending Jobs” on page 18-13 and “Preventing Interrupted Jobs” on page 18-13 for more information. In System/36-Compatible RPG II, the inquiry attribute can also be specified in column 37 of the control specification. The inquiry attribute cannot be specified in the other high-level languages.

MRTMAX: See “MRTMAX Value” on page 16-3 for information about the MRTMAX value.

Programming Considerations

This section contains considerations to help you code more efficient programs.

Programming Considerations for All Programs:

The following sections contain information applicable to all programs.

Acquiring a Display Station or ICF Session: A display station must be active (powered on and varied on) and signed off to be acquired. A program can acquire a display station by:

- Using the WORKSTN OCL statement with REQD=YES and a display station identification specified
- Using the programming language statement that acquires a display station

The following table lists the appropriate statement for each programming language:

Programming Language	Statement for Acquiring Display Station or ICF Session
COBOL 74	ACQUIRE statement
System/36-Compatible RPG II	ACQ operation code

For more information, refer to the appropriate language manual.

Releasing a Display Station or ICF Session:

Display stations should be released from a program when the program is no longer using them. A SRT program can release its requesting program device for the remainder of the job step, but it is acquired again automatically when the next job step opens a display file or ICF file. A MRT program can release requesters, acquired display stations, and acquired ICF sessions. The following table lists the statement for releasing a display station or ICF session in each programming language:

Programming Language	Statement for Releasing Display Station or ICF Session
COBOL 74	DROP statement
RPG II	REL operation code in calculation specifications, or R in column 16 of output specifications

For more information, refer to the appropriate language manual.

File Sharing: When a file is shared, you must prevent two problems:

- **Loss of data.** You can lose data when two or more users update the same record of a file shared within a single copy of a program. When the program writes the second updated record back to the file, it writes that record over the first updated record, losing the first update.
- **File deadlock.** Files deadlock when two or more programs try to update records in two or more files at the same time. Take the following steps to prevent file deadlock:
 - Do not own more than one record at a time. Each time you read a record from a file shared for update, write the record back to the file before you read a record from another file shared for update.

However, this technique significantly affects how you design your application because you may have to divide the application into several small programs.

- Design your application so each shared file is processed by only one program. This technique also requires you to write an updated record back to the file before you read another record.

This technique prevents loss of data because one update cannot be partially completed while another update of the same record occurs. For example, if two display station operators enter orders for the same inventory item at the same time, the second update may replace the first update.

To prevent loss of data and file deadlock, treat each display station input as a separate transaction. Do not assume that values valid at the previous display station input are still valid for the current display station input.

For more information about file deadlock, see “File Deadlock Conditions” on page 7-29.

Transaction File Design: A transaction file contains data (such as customer orders) used to update a master file.

Note: This transaction file is not the same as a COBOL TRANSACTION file, which is an input/output file used to communicate with display stations and ICF sessions.

You can design your transaction file by:

- Using a single file for all transactions
- Using a separate file for each device

If you use a single file for all your transactions, you can put all the transactions in the file in the order in which the transactions arrive, or you can put the transactions for each device in a separate part of the file. For example, you allocate 3000 records for the file, partitioned as follows:

Record 1	Control record for W1 (display station 1)
Records 2-1000	Data records for W1
Record 1001	Control record for W2 (display station 2)
Records 1002-2000	Data records for W2

Record 2001 Control record for W3
(display station 3)

Records 2002-3000 Data records for W3

You can assign a separate file to each device. For the preceding example, you can assign a separate file, each containing 1000 records, to each device. Because the FILE OCL statement is read only for the first requester of a MRT program, this approach requires a separate FILE statement for each user.

The method you choose depends on how the transaction file will be used and how you plan to recover from errors. For information about error recovery, see Chapter 19, "Error Prevention, Detection, and Recovery." For more information about transaction files, see "Transaction File" on page 7-19.

Memo Updating: Memo updating is a technique that allows interactive updates to your master files and provides batch processing to check that the updates have been applied correctly.

An advantage of an interactive environment is that operators always have access to up-to-date information in the master files. For example, an operator enters a transaction that reduces the quantity of an item on hand in an inventory master file. When another operator inquires for the quantity of that item on hand, the value shown reflects previous changes made to it.

Note: Do interactive updates to files carefully because recovery from a system or program failure is difficult if you do not know which updates are reflected in the file and which updates need repeating.

Master file records must allow duplicate fields for those fields that can be updated interactively. For example, in Figure 16-1 the field named *MBAL* (memo balance) could reflect interactive updates, and the field named *BAL* (balance) could be used for batch processing.

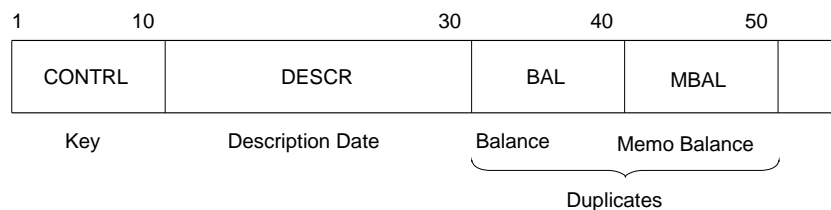
At the beginning of the day, these two fields shown in Figure 16-1 should be equal. The transactions made during the day are applied only to the memo balance field.

The RPG input and output specifications in Figure 16-2 on page 16-9 are used for the master file by interactive data entry and inquiry programs.

Note: These specifications ignore the balance field. The memo balance field should always reflect the current balance.

At the end of the day, a batch edit program processes the transaction file. The transactions are posted to the balance fields in the master file by a batch update program, as the segment of the program in Figure 16-3 on page 16-10 shows.

You must save the transaction files periodically, usually each day. You can save the master files frequently if the files are constantly being updated. You can recover by reloading the master files and processing all subsequent transactions. (For information about file recovery, see Chapter 19, "Error Prevention, Detection, and Recovery.") To bring the memo balance field to its current value, run a program that updates the memo balance field with the transactions. After the memo balance field has been updated, all current activity has been accounted for and normal operations can continue.



RSLW038-0

Figure 16-1. Fields Used in Memo Balancing

Line	Form Type	Filename or Record Name	Data Structure Name	Sequence		Number (UN) E	Option (O) US	Record Identifying Indicator, or DG	External Field Name									Field Location		RPG Field Name	Control Level (L1 L9)	Matching Fields or Changing Fields	Field Record Relation	Field Indicators		
				A	R				Record Identification Codes			From	To	Occurs n Times	Length	Decimal Positions	Plus	Minus	Zero or Blank							
									1	2	3															
01	I	TRAINING		NS	01										1	10	CONTRL	M1								
02	I														1	1	60AMT									
03	I	MASTIER		NS	02										1	10	CONTRL	M1								
04	I														3	1	400BAL									

Line	Form Type	Control Level (L0-L9, LR, CR, ANOR)	Indicators				Factor 1	Operation	Factor 2	Result Field		Resulting Indicators	Comments
			And	And	Not	Not				Name	Length		
01	C		O:2	MR		A:MT		A:D:D	B:A:L	B:A:L			
02	C												
03	C												

Line	Form Type	Filename or Record Name	Type (H/D/I/E)		Skip		Output Indicators			Field Name or EXCPT Name	Edit Codes	End Position in Output Record	P/B/L/R	Commas				Zero Balances to Print				No Sign				CR				-				x = Remove Plus Sign				y = Date				z = Zero Suppress				5 - 9 = User Defined			
			A	R	Before	After	Not	Not	Not					Not	Not	Not	Yes	No	Yes	No	1	2	3	4	A	B	C	D	J	K	L	M																	
01	O	MASTIER	D				O:2	MR																																									
02	O									B:A:L		4	0																																				
03	O									B:A:L		5	0																																				
04	O																																																

Note that these balances are set equal to one another.

RSLW078-0

Figure 16-3. Transaction File Processed by Batch Edit Program

Printed Output: If the application does not have exclusive use of the printer, you must decide whether the output can wait until the application is complete or should be printed as soon as it is available.

If you want the printed output as soon as possible, without having exclusive use of the printer, you must decide whether to use the PRINTER OCL statement to delay printing. For example, if you are printing invoices, and you specify DEFER-NO on the PRINTER OCL statement, you can make the printer unavailable to anyone else. If you specify DEFER-YES, you cannot print until your job is complete.

To have invoices printed as soon as they are available and have the printer available for other jobs, design the application so the print program runs as required. Use the EVOKE OCL statement to call the print step, which is therefore a NRT program.

The advantage of this technique is that it runs only when the output is ready to print. Also, the operator can continue processing because, as soon as the NRT starts, the application continues with the next step. If the print step is not a NRT, the operator has to wait until the program start, allocation, and end, as well as the printing, are complete.

The disadvantage of making the print program a NRT program is that the system can become overloaded with numerous small entries.

System Request Menu Options: When you press the System Request key, the System Request menu appears. Some options from that menu differ, depending on whether your program is a SRT or a MRT as described below:

For a SRT program:

- Option 2 ends the job and closes files.
- Option 20 is used by high-level languages to do special processing functions. See the correct language manual for more information.

For a MRT program:

- Option 2 releases the requester from the MRT program and sends the requester to the next step in the input stream. If RUF is in progress, the data from it will not be processed. You can use ?CD?=3721 on a proce-

sure control expression to test whether the MRT job step was ended by option 2.

- Option 20 is not allowed.

In a MRT program, processing of the system request key is delayed until the MRT has an input operation outstanding for the display station from which the system request key was used.

Disabling System Request Menu Options: A SRT program can disable System Request menu option 1 (Display sign-on for alternative interactive job) using any of the following methods:

- Include an ATTR OCL statement specifying INQUIRY - NO in a SRT procedure.
- Enter an ATTR OCL statement specifying INQUIRY - NO from the terminal before running a SRT procedure.
- Use the RPG II noninquirable program attribute.

A MRT program can disable System Request menu option 1 using any of the following methods:

- Include an ATTR OCL statement specifying INQUIRY - NO in a SRT procedure that calls a MRT procedure.
- Enter an ATTR OCL statement specifying INQUIRY - NO from the terminal before running a MRT procedure.
- Enter an ATTR OCL statement specifying INQUIRY - NO from the terminal before running a SRT procedure that calls a MRT procedure.
- Include an ATTR OCL statement specifying INQUIRY - NO in a MRT procedure.
- Use the RPG II noninquirable program attribute.

A SRT program can disable System Request menu option 2 using any of the following methods:

- Include an ATTR OCL statement specifying CANCEL - NO in a SRT procedure.
- Enter an ATTR OCL statement specifying CANCEL - NO from the terminal before running a SRT procedure.

A MRT program can disable System Request menu option 2 (Release display station from MRT) using any of the following methods:

- Include an ATTR OCL statement specifying CANCEL - NO in a SRT procedure that calls a MRT procedure.
- Enter an ATTR OCL statement specifying CANCEL - NO from the terminal before running a MRT procedure.
- Enter an ATTR OCL statement specifying CANCEL - NO from the terminal before running an SRT procedure that calls a MRT procedure.
- Include an ATTR OCL statement specifying CANCEL - NO in a MRT procedure.

Calling the Program: An operator can call a program in several ways:

- Enter the OCL statements at the display station.
- Enter the procedure name at the display station.
- Select a menu option at the display station.

A MRT procedure can call a MRT program. The system checks the procedure name to see whether the MRT is already active.

When you create a procedure, you can specify that the data following a procedure name is data for the program or parameters for substitution in the OCL statements. For MRT procedures, the procedure can be called only with data, not with parameters for substitution. Anything following the MRT procedure name is saved until the MRT program does its first input operation. See the appropriate language manual for information about calling the procedure with data. Use the Edit System/36 Procedure Attributes (EDTS36PRCA), the Change System/36 Procedure Attributes (CHGS36PRCA), or the Work With System/36 Procedure Attributes (WRKS36PRCA) CL commands to change the PGMDTA attribute. The PGMDTA attribute is described in "Procedure Attributes" on page 16-21.

Read-Under Format: The RUF technique allows an operator to enter information on a display while the program that uses the display is starting. When you use RUF, a program or procedure displays the format, and the program called next in the procedure reads it. The format is displayed by a program or a PROMPT OCL statement with PDATA=YES specified. While the next program is being started, the operator enters information for

the display. When the operator presses the Enter key, the input from the display is sent to the second program.

For more information about RUF, see "Using the Read-Under-Format Technique" on page 14-20.

External Switches: Eight external indicators (switches) are available for each requester. You can set or change these switches using the SWITCH OCL statement or the SWITCH procedure. The UPSI=YES parameter of the PROMPT OCL statement allows the current value of the eight external switches to be mapped to indicators 91 through 98. You can affect how the system processes your job by setting these switches. For example, if a certain error condition occurs, you can set a switch on and bypass those job steps that are in error.

The following table shows how to access these switches in various high-level languages:

Programming Language	How to Access External Switches
COBOL 74	IF and SET statements
RPG II	U1-U8 or SUBR 20

The following example uses the SWITCH OCL statement to affect how the system processes an input stream. An **input stream** is a group of records submitted as a batch job that contains CL commands for one or more jobs and data from one or more inline data files. In the example, the SWITCH1 statement sets switch 1 to 1 (on). Switch 1 determines whether the program is a daily or weekly run. In the procedure, a FILE OCL statement for the daily run differs from the FILE statement for the weekly run.

```
// IF SWITCH1-1 FILE NAME-A
// ELSE FILE NAME-A,LABEL-B
```

When switch 1 is on, file A is used. When it is off, file B is used.

For a MRT program, a separate copy of the switch settings is kept for each requester's input stream, and control is passed to the next job step when the requester is released. Also, the switch settings are normally the first requester's settings. Some high-level languages allow you to access a specific requester's switch settings.

The *System/36 Environment Reference* book has more information about using the external switches.

Local Data Areas: A local data area (LDA) consists of 512 bytes that you can use to receive data from previous job steps or to pass data to later job steps. You can use the LOCAL OCL statement to change the LDA. Local data can be substituted in OCL statements and procedure control expressions. A separate LDA is automatically created for each job.

For a MRT program, a separate copy of the LDA is kept for each requester's input stream, and control is passed to the next job step when the requester is released. Also, the LDA for the MRT is a copy of the first requester's LDA.

Whenever a program is placed on the job queue, called by the EVOKE OCL statement, released by the ATTR OCL statement, or called during inquiry, the external switches and the LDA have the value that was in effect at the time.

Each high-level language has a separate way of accessing the LDA. The following table shows how to access the LDA in COBOL 74 and RPG II:

Programming Language	How to Access LDA
COBOL 74	ACCEPT and DISPLAY statements
RPG II	SUBR 21 or special data structure

Note: Unlike System/36, acquired display stations do not have LDAs.

Programming Considerations for

Multiple-User Programs: The following sections describe programming considerations for multiple-user programs.

Creating a Table of Separate Variables:

Because a multiple-user program can handle more than one transaction at a time, you may require separate variables and work areas for each active display station. For example, you may need separate copies of:

- Indicators, flags, or switches
- Current record identification for each disk file
- Current display format

Design a multiple-user program so an entire transaction is completed between two successive display station input operations.

- Indicators, Flags, or Switches

Because a multiple-user program can process transactions that require several display station input operations, it may be necessary for the program to switch from one display station to another before completing a transaction. Therefore, the program may have to store the status of each display station whenever the program returns to the input operation for the display station file. Usually, you can code an array or table in which each element consists of the display station identification and the appropriate status fields.

- **Current Record Identification for Each File.**

The status fields in the table elements can include the key or record number for the last record processed in each disk file. When the program is reading duplicate keys sequentially and not all of the keys have been processed, the first of the series may have to be saved.

When the program reads from a display station file, it searches the table for an element that matches the display station identification. If it finds a match, it uses the element for that identification. If it does not find a match, it should allocate a new element. The program re-starts the identification field again when it releases the display station.

Another method for maintaining multiple current record IDs (one for each display station) is to define multiple logical files. One logical file could be defined for each display station using the file. For information about how to define multiple logical files, see "Using Multiple Names to Access a Single File" on page 7-30.

When a disk file is shared, the table of variables should not include current record identification because the only valid file information is that which you read after the most recent display station input. For more information, see "Sharing Files" on page 7-26.

- **Current Display Format.** Your program can store the last operation for each display station. The operation includes the display format name.

This table element could also include fields for such variables as local data area, external switches, and separate totals.

Note: Keeping track of these variables is complicated, because completion of a transaction can require several input operations from a display station, and those input operations may not be consecutive. Therefore, if possible, you should try to avoid relying on tables of variables between display station input operations. To avoid using tables, try to obtain all the necessary variables from the preceding display format.

For example, a part number or customer number is on the preceding display format. To use these fields, you have to change them from output-only to input/output fields. You have to write as protected or nondisplayed fields information that otherwise would be stored in the table of variables. This method also requires you to read the master file again for each display station input operation.

Sequential Processing of Multiple Records

with Duplicate Keys: When you use the generalized processing method to process multiple records with duplicate keys in one file shared among users of the same program, you may need to keep track of which record you last processed for each user so that you can continue processing from that point. Each time you process the file for a given key, you process the first record with that key value. If other records have the same key, continue reading the file sequentially until you find the record you want. If you must interrupt the sequence of read operations to read from the display station file, you might not be able to find the last record you processed unless you create a table to store the last record processed for each file by each display station. For information about the generalized processing method, see "Generalized Processing Method" on page 7-17. For information about duplicate keys, see "Using Duplicate Keys" on page 7-11.

Changing a One-User Program to a Multiple-

User Program: To change a one-user program to a multiple-user program, you may have to change the program logic. In a one-user program, there is no need to keep separate copies of variables and work areas for each device. A table or array is usually necessary in a multiple-user program.

If you always complete a transaction before receiving input for the next transaction (such as a read-only inquiry program) it probably requires no more logic as a multiple-user program than as a one-user program, except to release the requesters.

If you are changing a one-user program to a multiple-user program, then usually you are converting a SRT to a MRT, and the logic is changed to account for multiple transactions at the same time. To convert a SRT to a MRT, you must specify a nonzero value for the maximum number of requesters when the program is compiled. Also, use the CHGDSPF or CHGICFF command to change the maximum number of devices (MAXDEV value for display files, MAXPGMDEV for ICF files) allowed by the display file or ICF file.

The general steps to change a one-user program to a multiple-user program are:

1. Create a data structure to save unique information required by individual display stations from one cycle to another if necessary. The data structure search argument should be the display station identification.
2. Always reread the disk record to be updated if a display station input operation occurred after the previous disk read.
3. Make sure that the program logic can handle the maximum number of users.

Response Time: If a multiple-user program has considerable input/output or processing, the average response times for the display station operators increase if there are many users at the same time. For this reason, a technique often used to ensure reasonable response time is to group the display stations. This multiple-user program is usually a MRT program, so each group has a separate copy of the MRT program. This technique reduces the number of display stations trying to use the program.

You can group the display stations using a MRT program by using the NEWNAME parameter on the LIBRLIBR utility. This will create a new copy of the MRT procedure within the same library. You can decrease run-time response time for a MRT by using the MRTMAX parameter when compiling the MRT program to limit the number of display stations using each copy.

Programming Considerations for MRT Programs:

The following sections describe programming considerations for MRT programs.

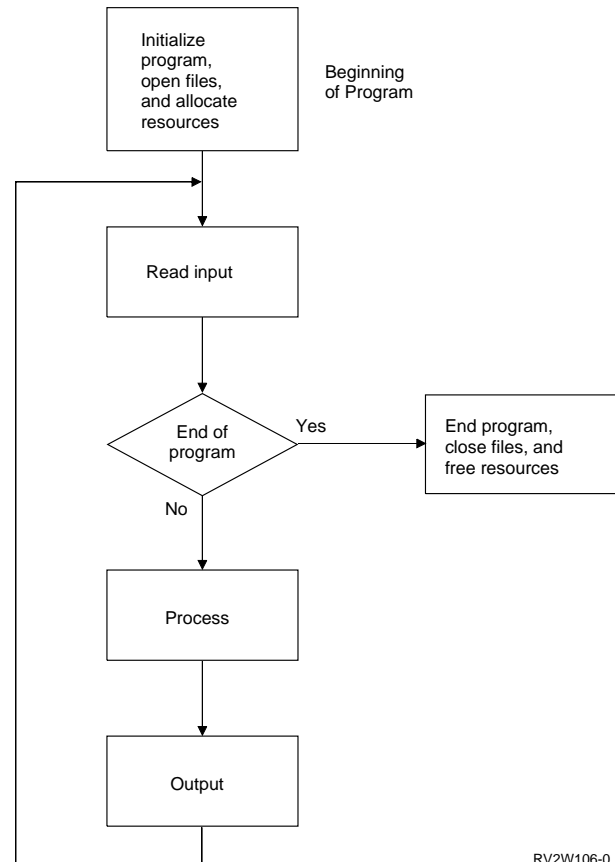
Security Verification: To run a SRT program or route into a MRT program, you must be authorized to the program and the library in which the program resides.

Resources used by a program require the correct authorization. During System/36 environment configuration, you can specify one of the following methods:

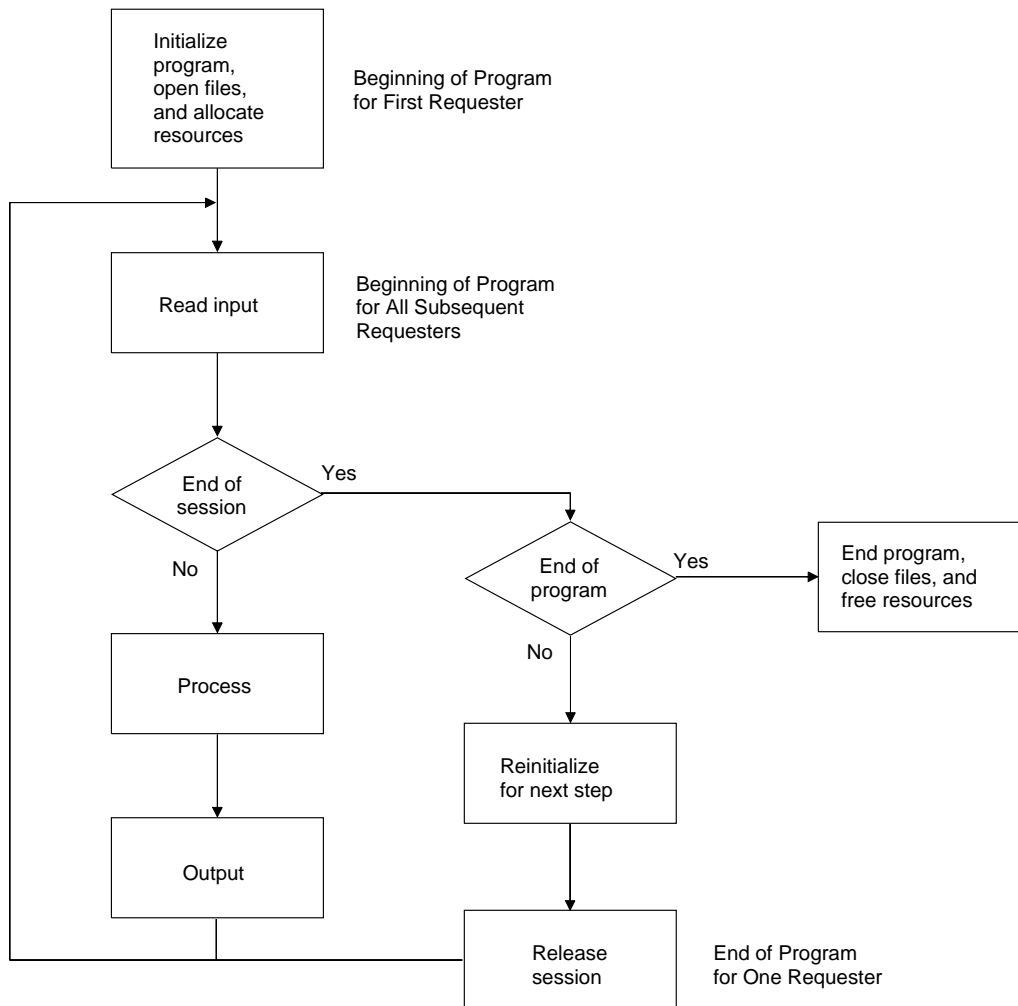
- Check the MRT initiator and all subsequent requesters of the program for the correct authorization to the files the MRT is using.
- Check only the MRT initiator for the correct authorization to the files.

An advantage of checking only the starting MRT is that overall system response time is shorter because less authorization checking is performed. Specifically, response time is quicker for the second and subsequent requester of a MRT NEP. Checking only the MRT initiator also allows you to enroll new users more easily because there are fewer authorizations to set up for them. MRT security considerations are described in more detail in Chapter 11, "Security."

Input Stream: The following figure illustrates a normal program flow:



RV2W106-0



RSLW059-0

Figure 16-4. Input Stream Flow of a MRT Program

As shown in Figure 16-4, a MRT program has the same flow, except:

- All requesters start by reading input.
- All requesters end by being released from the application program.

SRTs can interact with more than one display station or ICF session, but only one can be the requester. There is a separate input stream for each display station attached to a MRT (see

Figure 16-5 on page 16-17). The input stream consists of a series of steps, each step including LOAD and RUN OCL statements, typically FILE statements, and possibly other OCL statements. When a step is a MRT, the MRT is normally already active, so the system attaches to the MRT program rather than going through the program start-up required for a non-MRT. When the step is complete for that input stream, the MRT releases the requester rather than going through a time-consuming program end.

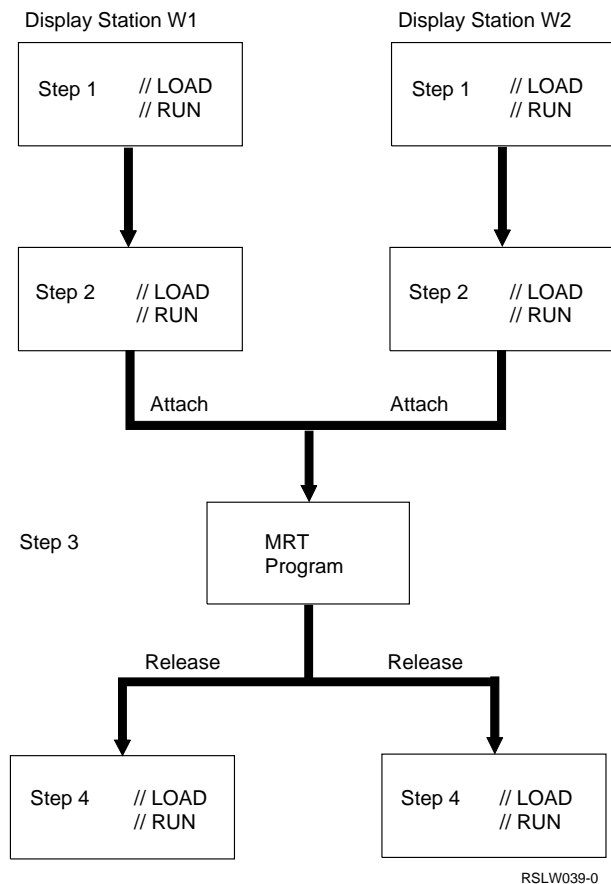


Figure 16-5. Input Stream for a MRT Program that Interacts with Multiple Display Stations

For additional information about jobs and job processing, see Chapter 18, “Jobs and Job Processing.” For additional information about procedures, see “Procedures” on page 16-21.

Modular Applications: If you divide a large application into a set of small, simple programs, the performance decreases if the programs are SRTs because of the time for start and end. The performance is probably better if the programs are NEP-MRTs.

In Figure 16-6 on page 16-18, an order-entry application is divided into four steps:

1. Read the customer master file (CUSTNO program).
2. Enter the order (DETAIL program).
3. Update the inventory file (INVENTORY program).
4. Print the invoice and packing slip (ORDPRINT program).

This modular design is easier to maintain than one large, complex program. Because the devices

attach to and release from already-active programs, the MRT programs have acceptable response times.

You can code the procedure for this example as follows:

```

// TAG TAGB           Start of procedure.
// PROMPT format name First order entry format.
CUSTNO               Procedure to read customer file.
DETAIL               Procedure to enter order.
INVENTORY            Procedure to update inventory file.
// IF ?L'1,4'?/0 GOTO TAGA Test for successfully completed
*                    order. First 4 bytes of LDA are
*                    set to nonzero value when order
*                    is canceled.
// GOTO TAGB         Start next order.
// TAG TAGA         Normal processing continues here.
// ATTR RELEASE=YES Create NRT program to print invoice
// LOAD ORDPRINT    and packing slip while beginning
// RUN              to process next order.
// GOTO TAGB         Process next order.
  
```

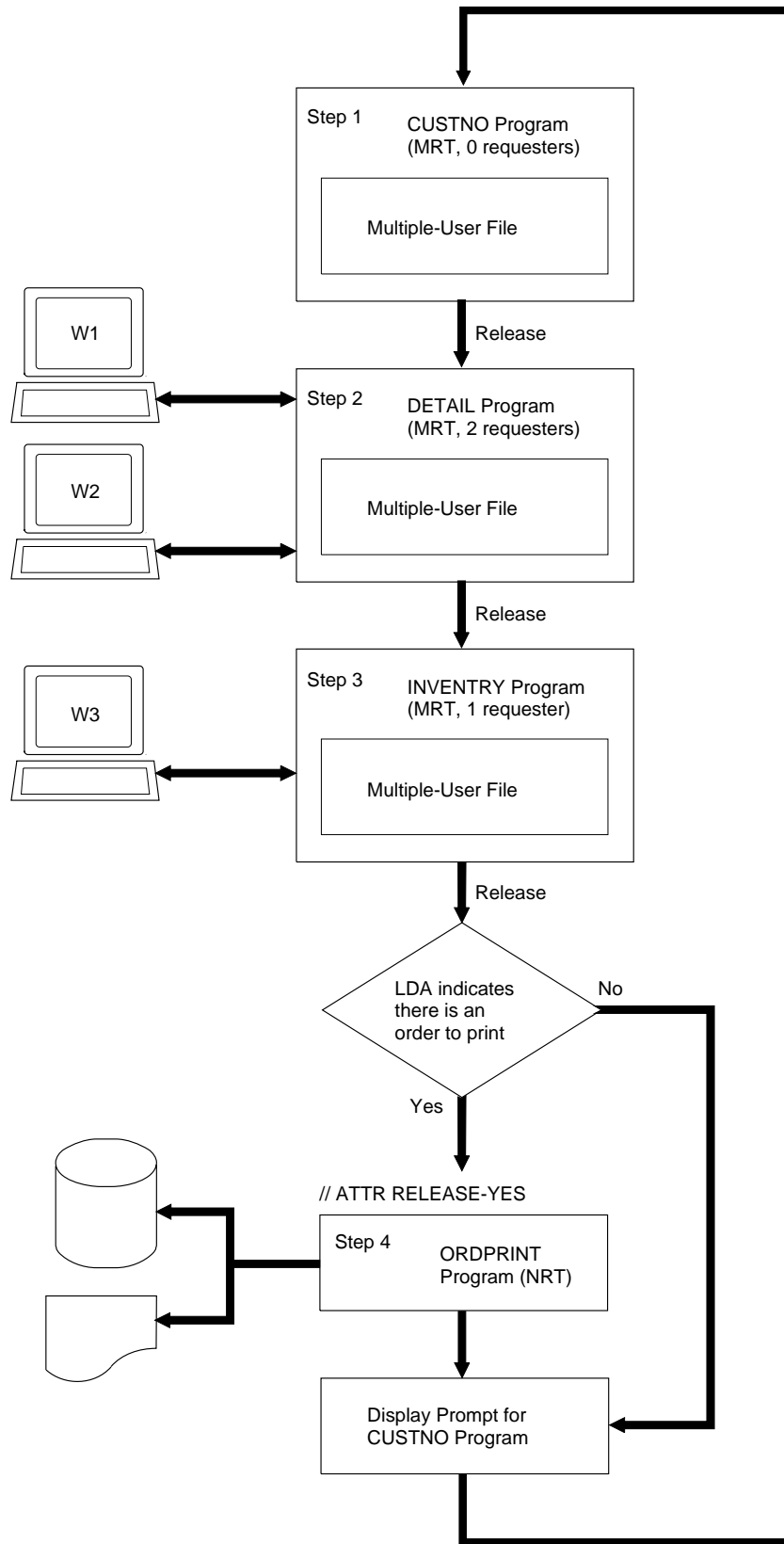
In this example, the print job step is shown as a NRT program to ensure the best performance for the procedure and maximum availability of the printer.

Testing MRTMAX: A MRT can be a one-user or multiple-user program. You specify the MRTMAX value when you compile your program. The value can be decreased on the ATTR OCL statement or with the Change System/36 Program Attributes (CHGS36PGMA), the Edit System/36 Program Attributes (EDTS36PGMA), or the Work With System/36 Program Attributes (WRKS36PRC) CL commands.

If a MRT is a one-user program, it must have a MRTMAX value of 1. A MRTMAX value of 1 can ensure that only one requester is attached to the MRT a given time. This technique is sometimes used to control resource sharing.

If your program has the maximum number of requesters, and another display station or communications program start request attempts to attach to it, the new requester must wait until a previous requester is released. An application can avoid unnecessary waiting for a MRT that has the maximum number of requesters either by testing if at MRTMAX or by using the MRTWAIT-NO parameter on the ATTR OCL statement.

Using IF MRTMAX: Use the IF MRTMAX procedure control expression to test whether the MRTMAX value has been reached. Use the IF MRTMAX procedure control expression to issue a prompting message. This allows the IF test to be



RSLW040-2

Figure 16-6. Modular Application Using MRT Programs

tried again before calling the MRT. If the problem persists, raise the MRTMAX value or use separate copies of the program. The *System/36 Environment Reference* book has information about procedure control expressions.

Using MRTWAIT-NO: Use the MRTWAIT parameter on the ATTR OCL statement to test if the MRT is at MRTMAX.

If you specify MRTWAIT-YES, you wait until a requester is released from the MRT before you attach.

If you specify MRTWAIT-NO, and the MRT is at MRTMAX, you regain control and return code 2045 is set. This return code can be tested using the ?CD? substitution expression. You cannot enter the MRTWAIT-NO parameter from the keyboard.

The //ATTR statement with MRTWAIT-NO is required before each start of the MRT procedure to use this function.

The following is an example of MRTWAIT-NO specified in a procedure.

```
// ATTR MRTWAIT-NO
MRTPROC
// IF ?CD?=2045 GOTO PROC2 /*TRY PROC2*/
// RETURN
// TAG PROC2
// ATTR MRTWAIT-NO
// PROC2
// IF ?CD?=2045 GOTO PROC3 /*TRY PROC3*/
// RETURN
// TAG PROC3
.
.
.
```

Limiting the Number of Users: Usually, the number of users of a MRT program means the number of requesters. To limit the number of requesters, use the MRTMAX value at compile time. At run time, you can use the ATTR OCL statement to reduce the MRTMAX value.

Program users can also include acquired display stations or ICF sessions, which can be acquired using statements in the high-level language program. Display sessions can also be acquired using the WORKSTN OCL statement. Except in

System/36-Compatible RPG II, you cannot control the number of display stations or ICF sessions acquired at run time. In RPG II, you can use the NUM keyword to limit the total number of requesters plus acquired display stations or ICF sessions.

In addition to the MRTMAX value, the number of requesters and acquired workstations is limited by the MAXDEV value in the display file. The number of requesters and acquired ICF sessions is limited by the MAXPGMDEV value in the ICF file. If a new requester attempts to attach to a MRT program, but the number of active workstations or ICF sessions are already attached to the MRT program, the new requester will wait until a device is released by the MRT program, unless MRTWAIT-NO was specified on an ATTR OCL statement before the MRT program was requested. The IF MRTMAX procedure control expression does not detect the situation where a device cannot attach to a MRT program because the MAXDEV or MAXPGMDEV values are less than the MRTMAX value. For further information, see "MAXDEV Value" on page 16-2.

Other Limitations: The same display station cannot have more than one active session in the same MRT. If you use the system request key to suspend processing in a MRT, your display station may route to a different MRT.

It is possible to code a MRT program that causes all other requesters to wait indefinitely. To avoid having all other requesters wait indefinitely, do one of the following:

- When a MRT program writes more than once to a display station before allowing input for each output (except the last), you should specify suppress input in columns 35 and 36 of the S specification for the \$SFGR utility.
- When the displays are created with DDS, for each output, except the last, you should not specify the INVITE keyword on the DDS).

Note: If you do not specify suppress input on the SFGR (or have the INVITE keyword on the DDS), the program and all other devices requesting it may wait indefinitely if the operator presses the System Request key between the output operations to the same display station.

First-Requester Considerations: The following considerations apply only to the first requester of a program:

- The MRT procedure is interpreted only for the MRT initiator. Other requesters are attached directly to the MRT program.
- Many run-time variables of the MRT program (such as the parameters on the FILE OCL statement) are specified by the first requester. You may not know if you are the first requester or a subsequent requester. If you are a subsequent requester and you specify a file name, it will be ignored.
- Although a MRT is logically a separate job, you can use it as a step in any other job. Whenever possible, variables that start the job status come from the System/36 environment configuration (use the Change System/36 Environment (CHGS36) or the Work with System/36 Environment Configuration (WRKS36) CL command to change these) or from the process of loading the system. These variables include date, date format, forms number, and lines per page. Other variables must come from the first requester. These variables include priority, NEP, MRTMAX, log, external switches, current library, procedure library, and local data area.

Summary of MRT Program Considerations:

Unlike a SRT program, a MRT program cannot write to a requesting display station before reading from the display station. You can call a MRT program from a MRT procedure.

The system checks by procedure name whether the MRT is already active. Because it takes more time, the system does not check by program name or by procedure name qualified by library name. Therefore, you should have only one copy of a MRT procedure in the system.

When a MRT program that is not a NEP releases its last attached device, the system checks the MRTDLY attribute of the MRT procedure to determine if the MRT should end immediately or delay termination for a short time, in case a new requester would like to attach to the MRT.

If the MRTDLY attribute indicates that the MRT should delay termination, the MRT will wait for the number of seconds specified for the System/36

environment before giving the program a return code instructing it to go to the end of the program.

If the MRTDLY attribute indicates that the MRT should not delay termination, the program will immediately be given a return code instructing it to go to the end of the program. When the program ends, the MRT is no longer an active MRT. If another device requests the MRT, a new MRT is started.

The attributes of the MRT procedure can be changed using the Change System/36 Procedure Attributes (CHGS36PRCA) command, the Edit System/36 Procedure Attributes (EDTS36PRCA) command, or the Work with System/36 Procedure Attributes (WRKS36PRCA) command. The attributes of the MRT procedure can be displayed using the EDTS36PRCA or WRKS36PRCA commands. You can display the number of seconds that the MRT will wait before ending by using the Display System/36 (DSPS36) command and you can change it by using the Change System/36 (CHGS36) command. You can also use the Work with System/36 Environment Configuration (WRKS36) command to display and change the number of seconds that the MRT will wait before ending.

When starting non-MRT procedures, you can specify that the data following a procedure name is data for the program or parameters for substitution in the OCL statements. When starting MRT programs, you can call the procedure only with data, not with parameters for substitution. Anything following the MRT procedure name is saved until the MRT program does its first input operation.

If a MRT program reads from a specific display station (rather than reading from any display station that has input ready), all other users wait until the specified display station input operation is complete. For example, an RPG II program processes a NEXT operation for display station W2. No other display station input is processed until the operator at display station W2 presses the Enter key to input the data.

When you specify RELEASE-YES with an ATTR OCL statement for a NEP MRT program, the MRT program becomes a MRT program with zero requesters, not a NRT program.

You cannot use an EVOKE OCL statement to start a MRT procedure, but you can use an OS/400 ICF EVOKE operation code to start a MRT procedure. See “ICF Files” on page 13-17 for more information about evoked ICF communications jobs.

For a SRT ICF program, system messages go to the job, job log and/or system operator message queue. When a SRT display program runs, system messages go to the job, job log, and/or the requesting display station. When a MRT or NRT program runs, all system messages go the job, job log and/or system operator message queue.

Procedures

A **procedure** is a collection of statements that causes one or more programs to run. Use procedures to start jobs. With System/36 environment procedures you can create and copy data files, and create libraries. The procedures supplied with the languages (such as RPGC and COBOLC) allow you to compile and run the programs you code. These procedures are described in the *System/36 Environment Reference* book, and the programming language manuals.

You can also create your own procedures.

Procedure Attributes

Procedure attributes are indicators that determine the procedure’s characteristics. They are set by the migration utility for procedures migrated from System/36. You use the Edit System/36 Procedure Attributes (EDTS36PRCA), the Work with System/36 Procedure Attributes (WRKS36PRCA) command, or Change System/36 Procedure Attributes (CHGS36PRCA) CL commands to change the following procedure attributes:

MRT

Specifies if the procedure is a MRT procedure.

LOG

Specifies if OCL statements are logged to the job log. See the *System/36 Environment Reference* book for information on how to prevent the procedure’s OCL statements from being logged to the job log when the procedure

runs. The procedure command that started a procedure is always logged to the job log. Normally, the procedure command and the OCL statements for your procedures are all logged to the job log. This is done to help you debug your procedures.

PGMDTA

Specifies whether data passed to the procedure should be handled as parameters or as program data. For information on using the PGMDTA, see “Calling Procedures” on page 16-22. This is the same as the PDATA attribute on System/36.

RCDLEN

Specifies the logical record length of the statements in the procedure member (used by Save System/36 Library Member (SAVS36LIBM) command).

REFNBR

Specifies the reference number assigned to the procedure member.

MRTDLY

Specifies if the termination of the MRT should delay a configured number of seconds after the last attached device is released from the MRT. This attribute is only used by the system if this procedure is used to start a MRT that is not a NEP. The delay value can be displayed using the Display System/36 (DSPS36) command, or changed using the Change System/36 (CHGS36) command. You can use the Work with System/36 Environment Configuration (WRKS36) command to display and change the delay value.

Parts of a Procedure

Procedures can contain the following types of statements:

OCL statements

Used to load and run programs. OCL statements indicate how the System/36 environment runs the program and uses input and output devices. Examples of OCL statements are LOAD, FILE, and RUN.

Procedure control expressions

Control how the procedure is processed.

Procedure commands

Cause other procedures to be run. Examples of these procedure commands are

COPYDATA and SAVE. You can use procedure commands to run your own procedures.

Utility control statements

Used with System/36 environment utility programs, to pass information to other utility programs.

AS/400 CL commands

In the System/36 environment, you can use AS/400 CL commands. CL commands can be used in procedures to request AS/400 functions.

Procedures cannot contain operator control commands.

See the *System/36 Environment Reference* book for complete information on these statements, including a list of those System/36 statements restricted or not supported in the System/36 environment.

Using Procedures

Use procedures to avoid entering several statements each time a job is run. For example, OCL statements can be included in a procedure. The collection of statements is stored in a library member called a **procedure member**. A procedure member is a member of source physical file QS36PRC. See Chapter 6, "Libraries," for information about accessing procedure members.

Functions of Procedures: When you use procedures to run jobs, you can:

- Run several job steps by entering one procedure command. This method eliminates the repeated entering of OCL statements each time a job runs.
- Start your jobs from the menu by entering an option number when you code the procedure command in a menu.
- Prompt for and pass variables (parameters) to your jobs.
- Check that the proper values were entered for the parameters, or make decisions based on the values entered.
- Code a procedure so it passes data to a program called by the procedure.

- Change the user-programmable status indicator (UPSI) switches and the local data area (LDA).

A **user program status indicator (UPSI) switch** in the System/36 environment is one of a set of eight switches that can be set by and passed between application programs and procedures.

Calling Procedures: Use one of the following methods to call a procedure and specify data:

- Procedure-name,library-name data
- // procedure-name,library-name data
- // INCLUDE procedure-name,library-name data
- A procedure start request sent over a communications line

Depending on the PGMDTA and MRT attributes of the procedure, the data specified on the request:

- Is sent as parameters to the procedure. This method is used when the procedure is a SRT procedure with a PGMDTA-NO attribute.
If the procedure is a MRT procedure, only program data (not parameters) is sent, regardless of the PGMDTA value.
- Returns as program data on the first input operation to the requesting display station that is issued from the next program run by the procedure. The data is used as program data whenever the procedure is a SRT procedure with a PGMDTA-YES attribute, or whenever the procedure is a MRT procedure.

Procedure Parameters: You can define parameters in your procedures. Parameters allow information and variables to be passed to the procedure. A procedure can have up to 64 parameters, and each parameter can have up to 128 characters.

Parameters passed to procedures are called **positional parameters**. Parameters are separated by commas (.). When a parameter appears in a procedure command, it must appear in the same position in relation to other parameters in the procedure command. That is, each parameter is assigned a place, such as the first parameter or the second parameter. If a parameter is omitted, a comma must still be used to indicate the *posi-*

tion of the omitted parameter. In the following example, the second parameter is omitted:

```
PROCA PARM1, ,PARM3
```

The first and third parameters are separated by two commas, which indicates that the second parameter is omitted.

Use substitution expressions to define parameters in your own procedures. The *System/36 Environment Reference* book has more information about using parameters with procedures.

The following example shows how two parameters are entered and substituted in a procedure. Procedure PROCA contains the following statements:

```
// LOAD PROGRAM1
// FILE NAME-INPUT,LABEL-?1?
// FILE NAME-OUTPUT,LABEL-?2?
// RUN
```

When you enter the procedure command PROCA FILE1,FILE2 to start PROCA, the first parameter (FILE1) and the second parameter (FILE2) are substituted for the expressions ?1? and ?2? as shown in the following example:

```
// LOAD PROGRAM1
// FILE NAME-INPUT,LABEL-FILE1
// FILE NAME-OUTPUT,LABEL-FILE2
// RUN
```

The substitution is done when the initiator function processes the procedure. The initiator function is described in Chapter 18, “Jobs and Job Processing.”

Procedures with Menus

Application users generally do not need to know the procedure’s name or parameters because you supply menus for them. The user selects an option from a menu and the system runs the procedure you assigned to the option. The procedure then loads and runs the program to do the task that the application user specified.

Calling a Procedure from Another Procedure

One procedure can call another procedure. A procedure called by another procedure is a **nested procedure**. Use nesting when the same procedure is called several times in a job. The procedure can be entered and stored only once, and then called as often as necessary.

Deleting a file your program creates is a good example of using a procedure to call another procedure. Your procedure can include the DELETE procedure command. For example:

```
* Delete work file FILE1, if it exists
// IF DATAF1-FILE1 DELETE FILE1,F1
* Run program
// LOAD PROGRAM1
// FILE NAME-FILE1,RECORDS-250
// RUN
```

Considerations for Multiple Requester Terminal Procedures

You can attach to a MRT program with MRT procedures. A MRT program allows several requesting display stations or requesting ICF sessions to attach to one copy of a program at a time.

You specify a MRT procedure using the Change System/36 Procedure Attributes (CHGS36PRCA), the Edit System/36 Procedure Attributes (EDTS36PRCA), or the Work with System/36 Procedure Attributes (WRKS36PRCA) CL commands, or by specifying the MRT parameter of the \$MAINT utility.

If you enter a MRT procedure and the MRT is not already active, the system processes the OCL statements in the MRT procedure, including the LOAD and RUN of the MRT program, as part of starting the MRT. If you enter a MRT procedure with the MRT already active, and the number of requesters using the program is less than the maximum number of devices that can be attached to the MRT (MRTMAX), the requester is attached directly to the MRT program. If the MRT is active and has reached the maximum number of requesters, the requester waits for the MRT to release one of the other requesters.

Consider the following factors when using MRT procedures:

- Only one LOAD and RUN OCL statement pair is allowed in a MRT procedure. Also, any statements that follow the RUN OCL statement are ignored. However, there may be several MRT job steps in the same job.
- A MRT procedure can be contained in another procedure, but a MRT procedure cannot contain another procedure.
- The MRT procedure is interpreted only for the first requester of the MRT. Once the MRT program is running, other display stations or ICF sessions that request to use the MRT program are attached directly to the MRT program. Therefore, the OCL statements in a MRT procedure are not processed for other requesters.
- The INCLUDE OCL statement, the procedure command, or the communications procedure start request that starts a MRT procedure, can pass data to the MRT program. Also, the procedure name can be followed by a comma and the library name. The data to be passed to the program starts with the first nonblank character following the blank after the procedure name and ends with the last nonblank character in the statement. The system passes the data to the program on the first input operation for the first requester.
- DATE, FORMS, or MEMBER OCL statements used in a previous job step do not affect a job step that runs a MRT program. Instead, the MRT program uses values specified during system configuration, at IPL, or in the MRT procedure.
- Any PRINTER or SYSLIST OCL statement used in a previous job step has no effect on a job step that runs a MRT program. Instead, the MRT program uses the configured system printer for both program and system list output.

You may experience problems with RUF started by a MRT program when break messages appear on your display before you press the Enter key. A break message appears when:

- You change a message queue's delivery mode to *BREAK using the CHGMSGQ command.

- The SNDBRKMSG command is used by another user.
- The work station is attached to a SRT job. If a break message arrives while the work station is attached to a MRT program, the message does not appear until the work station is released by the MRT program.

When you return from the break message display, a blank display appears instead of the RUF display. You are unable to continue. If a MRT is attempting to read from the display, cancel either the MRT or the user's job. If a SRT is attempting to read from the display, either press the System Request key and select option 2, or cancel the user's job.

To prevent break messages from appearing on your work station, use the CHGJOB command, as follows:

```
CHGJOB BRKMSG(*HOLD)
```

Break messages that arrive are then saved until you enter CHGJOB BRKMSG(*NORMAL).

After you enter this command, all saved break messages appear. Any new break messages are handled in the usual manner.

For example, the following procedure performs RUF from a MRT to a SRT:

```
* The following MRT writes to and releases
* the display
// MRTPROC
* The following program reads from the display
// LOAD SRTPROC
// RUN
```

To stop the system from displaying break messages, add the following commands:

```
* Break messages will be saved
CHGJOB BRKMSG(*HOLD)
* The following MRT writes to and releases
* the display
// MRTPROC
* The following program reads from the display
// LOAD SRTPROC
// RUN
* Break messages will be shown
CHGJOB BRKMSG(*NORMAL)
```

Note: The System/36 environment MSG command does not cause your session to be interrupted unless you change your work station

message queue's delivery mode to *BREAK using the CHGMSGQ command.

Delaying MRT Termination

When a MRT program that is not a NEP releases its last attached device, the system checks the MRTDLY attribute of the MRT procedure to determine if the MRT should end immediately or delay termination for a short time, in case a new requester would like to attach to the MRT.

If the MRTDLY attribute indicates that the MRT should delay termination, the MRT will wait for the number of seconds specified for the System/36 environment before giving the program a return code instructing the program to go to the end of the program.

If the MRTDLY attribute indicates that the MRT should not delay termination, the program will immediately be given a return code instructing it to go to the end of the program. When the program ends, the MRT is no longer an active MRT. If another device requests the MRT, a new MRT is started.

Initiating and terminating MRTs uses much more of the system resources than attaching to a MRT that is already active. Therefore, you may wish to delay the termination of your often-used MRTs. A new requester that requests the MRT during the delay period will attach to the active MRT instead of starting a new MRT.

A non-NEP MRT with a MRTDLY procedure attribute indicating that the MRT should wait for a new requester before terminating will stay active longer than a non-NEP MRT with a MRTDLY procedure attribute indicating that the MRT should terminate immediately after releasing its last device.

MRT jobs continue to own system resources, such as files, during the MRT delay period, which affects other applications waiting for those resources.

If you do not want a certain non-NEP MRT to delay termination after releasing its last device, use the Change System/36 Procedure Attributes (CHGS36PRCA) command, the Work with System/36 Procedure Attributes (WRKS36PRCA) command, or the Edit System/36 Procedure Attri-

butes (EDTS36PRCA) command to change the MRTDLY attribute for the MRT procedure. If you do not want any of your non-NEP MRTs to delay ending, use the Change System/36 (CHGS36) command or the Work with System/36 Environment Configuration (WRKS36) command to change the System/36 environment MRT delay value to zero seconds.

Changing the configured MRT Delay value will affect the termination of all your non-NEP MRTs. Therefore, you must have security officer authorization to change this value.

Internal Processing of MRT Jobs

When the System/36 environment starts a MRT job for the first requester of a MRT procedure, the System/36 environment submits a batch job to job queue QS36MRT in library QGPL using the job description QS36MRT in library QGPL. Once the batch job gets started, the allocation of the requesting display station or ICF session is moved from the job that requested to run the MRT procedure to the MRT job. This allows the MRT job to issue I/O operations to the display station or ICF session.

If the MRT job is already running when a request to run the MRT procedure is received, the System/36 environment only needs to move the allocation of the display station or ICF session from the job that requested to run the MRT procedure to the MRT job.

You can use the Work with Active Jobs (WRKACTJOB) CL command to determine what MRT jobs are currently running on the system. MRT jobs are identified by the value MRT in the *Type* field of the WRKACTJOB display. For MRT jobs, the following conditions apply:

- The *Subsystem/Job* field contains the name of the MRT procedure.
- The *Function* field identifies the MRT as a NEP MRT program or non-NEP MRT. If the *Function* field contains NEP, the MRT is a NEP MRT. If the *Function* field does not contain NEP, the MRT is not a NEP MRT.

The *Function* field also displays the maximum number of requesters (MRTMAX) and the current number of requesters. For example, if the *Function* field is MRT-2/ 5, this indicates

there are currently 2 requesters of the MRT, and the MRTMAX value is 5.

The WRKACTJOB CL command also identifies the jobs (either interactive or ICF) that have been attached to a MRT. These jobs are identified by MRT-procedure name in the *Function* field of the WRKACTJOB display. For example, if the *Function* field is MRT-PAYMRT, the job has moved the allocation of the display station or ICF session to the MRT procedure PAYMRT. For jobs that have been attached to a MRT, the following conditions apply:

- If the *Type* field is INT, the *Subsystem/Job* field contains the display station name of the display station that is running a MRT procedure.
- If the *Type* field is EVK, the *Subsystem/Job* field contains the ICF communication device name of the device that is running a MRT procedure.

When a MRT job is started the AS/400 Work Management function sets up the attributes of the job based on the information in the QS36MRT job description in library QGPL. Information such as output priority, output queue, printer device, and message logging levels are initialized for the MRT job, based on the information in the QS36MRT job description. You can change the QS36MRT job description with the Change Job Description (CHGJOB) CL command to change the attributes of all MRT jobs. You can use the Change Job (CHGJOB) CL command in a MRT procedure to change the attributes of a single MRT job. For example, you can specify LOG(4 0 *SECLVL) for the keyword on the CHGJOB or CHGJOB command to create a job log for a MRT job. However, the following values are not taken from the QS36MRT job description:

Job queue

Job queue QS36MRT in library QGPL will always be used for MRT jobs.

Routing data

Routing data of QS36MRT will always be specified for MRT jobs.

Designing Procedures

The following sections introduce several topics you should consider when you are creating procedures.

Naming a Procedure

Assign meaningful names to your procedures to make them easier to remember and use. For example, if your accounting procedures begin with ACC or ACCT, use ACCTPAY for your accounts payable application and ACCTREC for your accounts receivable application.

Naming Conventions: A procedure name can be up to 8 characters long, and must begin with an alphabetic character (uppercase A through Z, #, \$, or @). The remaining characters can be any combination of alphanumeric and special characters, except DIR, SYSTEM, NEW, or ALL.

Avoid using the following characters in member names since they have special meanings in procedures: comma (,), apostrophe ('), question mark (?), slash (/), greater than sign (>), equals sign (=), plus sign (+), period (.), and hyphen (-). Do not use names that begin with Q because the system creates procedures with these names.

Procedure Performance and Coding Techniques

This section describes techniques you can use to help improve the performance of your procedures. The *System/36 Environment Reference* book has more information.

- Use GOTO and TAG statements rather than several IF expressions. **IF expressions** in the System/36 environment are expressions within a procedure that are used to test for a condition. Use one IF expression and a GOTO expression to reduce the time needed to evaluate several IF expressions. The statements skipped by the GOTO and TAG expressions are not processed.
- Use ELSE statements if you have more than one IF expression and only one of the expressions can be true. All ELSE statements are skipped after a true IF or a false IFF expression.

- Combine IF expressions when possible. The remainder of a statement is not processed after a false condition.
- Do not use the informational message (`// *`) statement to display prompting messages (such as `ENTER MEMBER NAME` or `ENTER LIBRARY NAME`). Use the `PROMPT OCL` statement and a display format instead. More information can be displayed, thus requiring fewer I/O operations.
- After you have tested your procedures, stop the logging of OCL statements to the job log. You may need to have the OCL statements only logged when you are creating and testing your procedure.
- If you have many comments in your procedure, you should put a `RETURN` statement at the end of the procedure and put your comments after the `RETURN`. That way the system processes the `RETURN` statement and your comments are not processed (thus saving the amount of time the system would otherwise use to read the comments).
- Use your own libraries for your applications. Run procedures and programs from a library other than the system library (`#LIBRARY`).
The OS/400 program always searches the current library first, and if the member is not found, it then searches `#LIBRARY`.
- Use IF conditional expressions to avoid having the system operator respond to an informational message when a procedure is sent to the job queue when the procedure is started by the `EVOKE OCL` statement, or an OS/400-ICF procedure start request.

Programming Considerations for Procedures

The following sections describe the procedures that create, change, list, control, and debug procedures.

Creating or Changing Procedures:

When you create procedure library members, the source physical file containing the procedure must be `QS36PRC`.

Enter procedures into a library using the programming development manager (PDM) or source

entry utility (SEU). PDM is described in the *ADTS/400: Programming Development Manager* book. SEU is described in the *ADTS/400: Source Entry Utility* book.

Also, use the `$MAINT` utility to create and copy procedures into a library. The `$MAINT` utility is described in the *System/36 Environment Reference* book.

PDM and SEU allow you to change lines in a procedure and store those changes. The `$MAINT` utility allows you to create procedure members only. If you want to change a line in the procedure, you must enter the entire procedure again.

The `$MAINT` utility, the Change System/36 Procedure Attributes (`CHGS36PRCA`) CL command, the Work with System/36 Procedure Attributes (`WRKS36PRCA`) CL command, or the Edit System/36 Procedure Attributes (`EDTS36PRCA`) CL commands allow you to specify the procedure attributes, described in "Procedure Attributes" on page 16-21.

Listing Procedures: Use the `LISTLIBR` procedure, PDM, or SEU to list a procedure. The `LISTLIBR` procedure is described in the *System/36 Environment Reference* book.

Controlling How a Procedure Runs:

You control how a procedure runs by using **procedure control expressions**. These expressions allow you to do the following:

- Create variables in procedures using substitution expressions including the following:
 - Value entered for a parameter
 - A return code set by system programs called by your procedure
 - Current or session library
 - Date and time
 - Information in the local data area
 - Session printer
 - System list device
 - Requesting display station's ID
- Test a value and process a statement using the IF statement. For example, you can test the following:
 - The value of a parameter for equal to or greater than
 - Whether a procedure is currently running

- Whether a file or library is on disk
 - Whether a file is on diskette
 - Whether a job is being run from the job queue or is evoked
 - Whether the maximum number of display stations is using a MRT program
 - Whether a library member exists in a library
 - The security classification of an operator
 - The volume ID of a diskette
- Display messages to the application users using the // * (informational message) statement or to the system operator using the // ** (console message) statement.
 - End an entire procedure using the CANCEL statement, or end a procedure level using the RETURN statement.
 - Use the EVALUATE statement to do the following:
 - Assign values to parameters.
 - Add, subtract, multiply, and divide numbers.
 - Determine the value of substitution expressions.
 - Set the job return code.
 - Branch to statements in a procedure using the GOTO and TAG statements.
 - Temporarily stop the running of a procedure and display a message using the PAUSE statement.
 - Start another procedure, or start the same procedure again using the RESET statement.

The procedure control expressions are described in the *System/36 Environment Reference* book.

Debugging Procedures: The following paragraphs describe OCL statements and CL commands that can help you debug your procedures:

DEBUG OCL Statement: With the DEBUG OCL statement, you can trace the logic flow of your procedures. It shows each level of substitution expression evaluation. The output is listed on the system list device and logged to the job log. Also, you can temporarily stop the running of a procedure between job steps.

LOG OCL Statement: With the LOG OCL statement, you can have the statements that are processed in your procedures logged to the job log.

You can then display or print the job log to determine the statements that were run.

DSPJOBLOG CL Command: The Display Job Log (DSPJOBLOG) CL command displays the contents of the job log.

Job Log and Procedure Processing:

The system automatically logs each statement processed in a procedure to the job log unless:

- You specified that statements were not to be logged with the Change System/36 Procedure Attributes (CHGS36PRCA), Work with System/36 Procedure Attributes (WRKS36PRCA), or the Edit System/36 Procedure attributes (EDTS36PRCA) CL commands.
- You use the LOG OCL statement to turn off statement logging. The LOG OCL statement controls whether the statements in a procedure are logged to the job log. The LOG OCL statement overrides the logging indicator specified in the procedure member.

The LOG statement affects only the logging of OCL statements to the job log. Other items, such as messages and job information are not affected by using the LOG OCL statement.
- The procedure is IBM-supplied.

Note: All IBM procedures have a default of NOLOG. You can log IBM-supplied procedures to the job log the way user procedures are logged.

Logging requires that system-processed OCL statements are written to the job log. This technique can increase the number of disk write operations and affect performance. You should log OCL to job log only when you first create your procedure members and are testing them to see if they are working correctly.

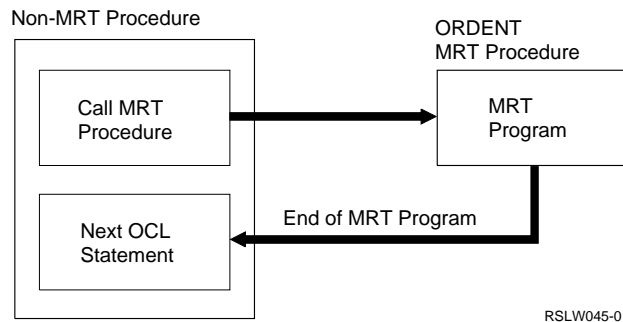
In addition to the LOG OCL statement and the procedure attributes, there is a setting for the entire job or session which controls how much information is written to the job log and whether the job log is produced when the job or session ends. This setting is taken from the LOG value of the job description that is used for the job. It is changed with the LOG parameter of the CHGJOB CL command for a job that is currently running.

Calling Multiple Requester Terminal Procedures:

When a non-MRT procedure calls a MRT procedure, you can have the non-MRT procedure check whether the maximum number of requesters is already attached to a MRT program. For example, the following non-MRT procedure uses the IF procedure control expression and the MRTMAX conditional expression:

```
* Test for the maximum number
// IF MRTMAX-ORDENT GOTO TOOMANY
*
* Number of requesters is less than the maximum,
* call ORDENT (which is a MRT procedure)
ORDENT
// RETURN          (End this procedure)
*
// TAG TOOMANY     (Maximum requesters already using ORDENT)
// * 'Too many people are running order entry.'
// * 'Canceling the procedure, try again later.'
// PAUSE
// RETURN          (End this procedure)
```

The following figure shows how the procedure ORDENT is called:



Moving from System/36 to the System/36 Environment

The migration utility creates display and ICF files with a default value for the maximum number of devices that can use the files. The value for the maximum number of devices must be the largest number of devices used by any program that uses this file. This maximum number of devices should include requester and acquired devices. You may need to change this value for your display and ICF files based on the number of display stations and ICF sessions that use the files.

When you create a display file with the FORMAT procedure or Create System/36 Display File (CRTS36DSPF) CL command, the system uses

the MAXDEV attribute of the source member. When multiple source members are used to create a display file, the system uses the MAXDEV attribute of the first source member. If the MAXDEV attribute is not set, or if a SFGR source member is not used, the system assumes a value of five. For more information about the MAXDEV parameter, see "MAXDEV Value" on page 16-2.

After migration, you can use the Change Display File (CHGDSPF) command to change the MAXDEV value of the display file. Use the Change System/36 Source Attributes (CHGS36SRCA) command, Work with System/36 Source Attributes (WRKS36SRCA) command, or the Edit System/36 Source Attributes (EDTS36SRCA) command to set the MAXDEV value in the source member containing the SFGR source used to create a display file. You can then create or recreate the display file, using the specified MAXDEV attribute saved in the source member, and do not have to use the CHGDSPF command whenever the file is created from the SFGR source, when the SFGR source member has the correct MAXDEV value for its source attributes. For ICF files, you can use the Change ICF File (CHGICFF) command to change the MAXPGMDEV value.

Notes:

1. Do not set MAXDEV to a number larger than necessary because this adversely affects storage and performance.
2. On System/36, MRT and SRT programs could access the local data area (LDA) of almost any device on the system. In the System/36 environment, MRT and SRT programs can only access the LDA of devices attached to the program.

Be aware of the following differences between System/36 and the AS/400 system when using the Dup key:

- On System/36:
 - Pressing the Dup key in an alphanumeric field returns a X'1C' to the application program.
 - Pressing the Dup key in a numeric field returns a X'FC' to the application program.
- On the AS/400 system:

- Pressing the Dup key in an alphanumeric field returns a X'1C' to the application program (as is done on System/36).
- Pressing the Dup key in a numeric field returns a X'F0' to the application program (this is different from System/36).
- If an application needs to process a Dup key in a numeric field, the application program needs to be changed to:
 - Define a character field over the numeric field.
 - Check the character view of the field for X'1C'.

Chapter 17. Mixing System/36 Environment and AS/400 Functions

An application that is migrated from a System/36 to the System/36 environment of the AS/400 system is called a **System/36 application**. System/36 applications use some of the following System/36 functions:

- Procedures that are made up of operation control language (OCL) statements and procedure control expressions (PCE)
- References to the System/36 utilities, such as \$COPY, \$DELETE, and so on
- High-level language programs
- References to System/36 utility programs such as SORT, DFU, and so on
- Message load members
- Screen Format Generator (SFGR) display formats
- Program-defined files

You can change these migrated applications to use AS/400 functions called **mixed mode applications**. The following is partial list of AS/400 functions:

- Control language (CL) commands
- CL programs
- AS/400 high-level language programs
- AS/400 licensed programs such as Query/400 and DFU
- Message files
- Data description specifications (DDS) display files
- Externally described files
- Database functions such as data recovery, transaction recovery, and access path recovery

You may change System/36 applications to use AS/400 functions for one or more of the following reasons:

- To support System/36 assembler subroutines that must be rewritten in an AS/400 high-level language. For more information on rewriting System/36 assembler subroutines, refer to the *System/36 Assembler Conversion Newsletter*.
- To add new functions that are available only as AS/400 functions.

- To enhance the application itself.
- To improve maintainability.
- To convert a System/36 application to an AS/400 application.

You can gradually change a System/36 application to a mixed mode application, and eventually to an AS/400 application. The changed application can contain a mixture of System/36 environment functions and AS/400 functions all running within the System/36 environment of the AS/400 system. An application that has been changed to use AS/400 functions can no longer run on a System/36.

When you start mixing System/36 environment functions and AS/400 functions within the same application, there is a set of rules you must follow. These rules, including using AS/400 CL commands, program control, and file processing in the System/36 environment, are discussed in the following sections.

Using AS/400 Architectural Features in System/36 Programs

System/36 applications have access to many of the architectural features of the AS/400 system that do not exist on System/36. For example:

- A program written in one language can call and pass parameters to a program written in the same language or a different language on the AS/400 system. There is no practical limit to the number of such programs that may be called.
- AS/400 CL commands may be intermixed among OCL statements in System/36 procedures.

In general, System/36 environment users have access to the best of both architectures. However, some of the architectural restrictions of System/36 remain. For example, System/36 does not allow recursive System/36 jobs, and this restriction has been retained within the System/36 environment of the AS/400 system. Two exam-

ples of the effect of this restriction are the following:

- On System/36 the system does not display a command line to the user once a procedure is running. Because of this, users cannot be running a procedure, request a command line, and run a second procedure. On the AS/400 system, the System/36 environment allows command lines to be presented to the user while a procedure is running. The System/36 environment allows AS/400 functions to be requested from this command line but does not allow System/36 environment functions (for example, procedures, OCL statements, operator control commands, and so on) to be entered.
- The AS/400 system provides support for a CL command (STRS36PRC) that starts a System/36 procedure. The System/36 environment provides support for running AS/400 CL commands from a System/36 procedure, either directly or from a CL program that may be called either by a CL CALL command or by a LOAD or RUN OCL statement. However, the STRS36PRC command may not be run from within a procedure, because this would attempt to start a System/36 job from within a System/36 job.

Using AS/400 CL Commands in the System/36 Environment

AS/400 CL commands can be processed in the System/36 environment. You can enter CL commands on the command line of a menu, from the System/36 Command Entry display, or from a procedure. A CL command can be used in the System/36 environment if the command is valid in the environment in which it is running (batch or interactive). Commands that are allowed only in a CL program (DCL, MONMSG, or RTVJOBA) and commands that are valid only in a batch job stream (BCHJOB, DATA, or ENDBCHJOB) cannot be processed in the System/36 environment.

The System/36 environment first checks a statement to determine whether it is a valid System/36 OCL statement or command. If not, the statement is processed as a CL command. If you want to process a CL command that has the same name as a System/36 procedure or command, you must

enter a command label or a library qualifier on the command line to prevent the command from being processed as a System/36 statement.

The following examples show how a statement that uses the name CMD1 is processed when you have a System/36 procedure and a CL command that are named CMD1:

CMD1 [parameters]

Process the System/36 procedure.

CMD1,USERLIB [parameters]

Process the System/36 procedure from library USERLIB.

A:CMD1 [parameters]

Process the CL command.

?CMD1 [parameters]

Process the CL command with prompting.

***LIBL/CMD1 [parameters]**

Process the CL command.

USERLIB/CMD1 [parameters]

Process the CL command from library USERLIB.

Note: If you type the *LIBL/CMD1 CL command, make sure the asterisk is not in column 1. An asterisk in column 1 is interpreted as a comment by the System/36 environment. Therefore, the statement is ignored.

Entering AS/400 CL Commands Interactively

You can type a CL command on the command line of a menu or on the System/36 Command Entry display. To prompt for the command, press PF4 or type a question mark in front of the command name. When you are in the System/36 environment, statements typed interactively are in uppercase. If you type lowercase characters on the prompt display and enclose the characters in quotation marks, they are processed as lowercase characters.

You can type CL commands either before or mixed with the OCL statements for a job step. CL commands cannot be entered when source or utility control statements are expected. For example, if you type the following OCL statements:

```
// LOAD $MAINT
// RUN
```

you can type only utility control statements that are valid for the \$MAINT utility program until the job step is either canceled or ended with a // END statement.

Handling Errors on CL Commands: If a CL command entered interactively in the System/36 environment ends with an exception, it is handled differently depending on whether a System/36 job step is active. A System/36 job step becomes active when you start entering OCL statements and remains active until the job step is canceled or the program specified on the // LOAD statements ends.

If you enter a CL command when a System/36 job step is not active, the job ends in an error, an error message is displayed at the bottom of the display, and the command remains on the command line.

If you enter a CL command when a System/36 job step is active, and the job ends in an error, a System/36 halt is issued. You can select option 0 to ignore the error and continue entering statements for the job step, or option 3 to cancel the job step.

Adding AS/400 CL Commands to System/36 Procedures

You can use CL commands in procedures, and generally wherever an OCL statement is valid. AS/400 commands are not allowed where a utility control statement or source statement is required (after a // RUN and before a // END or /*).

Note: If a CL command, issued during a procedure, requires a file it is processing to be closed, the System/36 environment closes the automatic shared open for the file. If the file (or member) is deleted, renamed, moved or restored by the CL command, the System/36 environment removes any // FILE statements for the file (or member). Any locks that were held through these // FILE statements are released. If a job file is renamed, moved, or restored by a CL command, the System/36 environment will not release the disk space occupied by the job file when the job ends. The System/36 environment will not close an automatic shared open of an alternative indexed or logical file which is based on the file being processed by the CL command. See “Shared File

Opens within the Same Job” on page 7-37 for more information.

Syntax of CL Commands in Procedures: CL commands in a procedure must have a valid syntax after any procedure control expressions or substitution expressions are processed. CL command syntax is described in the *CL Reference* book.

You can enter a question mark on a CL command to cause the command to be prompted, but only in an interactive job. An error occurs if prompting is requested for a command running in a batch job.

If a CL command requires more than one line in the procedure, it can be continued with a plus sign (+). The minus sign (-) is not valid as a continuation character for commands entered in a procedure.

Unlike OCL statements, it is not possible to condition parts of a CL command by placing each part on a separate line with a comma continuation character. If you want to condition part of a CL command, you can do so by setting a procedure parameter to either a null value or to the value to be included on the command.

The System/36 environment allows a number of special characters in names. The AS/400 system requires that names that contain special characters be enclosed in quotation marks ("). When you specify a name that contains special characters on a System/36 statement, you do not specify the quotation marks. When you specify a name that contains special characters on a CL command, you must specify the quotation marks. You may also specify quotation marks for a name that does not contain special characters; in this case, the quotation marks are ignored. The AS/400 naming rules are described in the *CL Reference* book.

This is shown in the examples below. The following two statements set the current library to be a library named "AB%CD". The quotation marks are not specified on the // LIBRARY OCL statement, but are added by the System/36 environment to create a correct AS/400 name.

```
// LIBRARY NAME-AB%CD
CHGCURLIB CURLIB("AB%CD")
```

The following three statements all delete a file named ABC. Because ABC is a valid name, the quotation marks are ignored in the third statement.

```
DELETE ABC,F1
DLTF FILE(ABC)
DLTF FILE("ABC")
```

If your installation uses special characters in names, it is recommended that you put quotation marks around a substitution expression used as a name on a CL command. This ensures that the name is valid if the substitution expression contains special characters. Quotation marks are not included in the value of a substitution expression that returns the name of an object. For example, if the current library is named "AB%CD", the ?CLIB? substitution expression returns the value AB%CD.

In the following example, an error occurs if either the files library name or parameter number 1 contains special characters:

```
CLRPFM FILE(?FLIB?/"F?1?") MBR(*LAST)
```

The next example shows how this statement can be changed to work correctly if either value contains special characters.

```
CLRPFM FILE("?FLIB?"/"F?1?") MBR(*LAST)
```

Quotation marks are not removed from substitution expressions that return the value of a positional parameter or retrieve a value from a message member or the local data area. In the following example, positional parameter 1 is set to a value with quotation marks. This value is then used on a CL command to set the current library name.

```
// EVALUATE P1="AB%CD"
CHGCURLIB CURLIB(?1?)      Sets current library to "AB%CD"
SLIB ?1?                   Error because quoted name not
                           allowed on System/36 statement.
```

Substitution Expressions on CL Commands: To facilitate mixing OCL statements and CL commands in a procedure, you can use System/36 substitution expressions on any part of a CL command, including the command name and keyword names. In addition, new substitution expressions have been added to the System/36 environment to assist the programmer who wants to add CL commands to a procedure. The substitution expressions allow a programmer to:

- Determine the name of the System/36 environment files library (?FLIB? substitution expression).
- Determine whether a message was issued by a CL command (?MSGID? substitution expression).
- Determine the AS/400 10-character device name of a System/36 environment 2-character device name (?DEV'unit'? substitution expression).

Use the following statement in a System/36 environment procedure to store the AS/400 10-character device name used for the System/36 environment 2-character device name for printer P1 in the local data area:

```
// LOCAL OFFSET-1,DATA-'?DEV'P1?',AREA-USER
```

For more information on the ?DEV'unit'? substitution statement, see the *System/36 Environment Reference* book. This book also has information on using substitution expressions and CL commands in procedures.

All substitution expressions on a CL command are processed before the command is processed. This means that it is not possible to use a CL command to modify the value of a positional parameter. For example, the following statement can be used to call a program and pass the value of positional parameter 1 to the program. However, if the program changes the value of the parameter passed to it, the change is not reflected in the value of the positional parameter.

```
CALL PGM(PGM1) PARM('?1?')
```

Use the IBM-supplied program QEXCVTDV to convert device names, as described in the *System/36 Environment Reference* book.

Handling Errors on CL Commands in Procedures: Handling errors on CL commands in a procedure is controlled by the default message action set in the System/36 environment configuration and by the CHGS36MSGL command. The initial setting of the default message action is to issue a halt message with options 0 and 3 allowed. You can use the CHGS36MSGL command to change the default message action for that job. The *System/36 Environment Reference* book has more information on using the CHGS36MSGL command.

When you use a CL command in a procedure, you should determine which action should be performed for any escape message sent by the command and ensure that errors are not ignored when they should be handled. You can use the CHGS36MSGCL command to control processing for errors, or test the ?MSGID? substitution expression to determine whether an error occurred on a command.

Program Control in the System/36 Environment

This section discusses giving control to both System/36 programs and AS/400 programs that are to run in the System/36 environment.

In general, each System/36 and AS/400 program follows the rules defined for the language in which it is written.

For example, the default data types are not always the same. The default for numeric data in RPG II is zoned decimal, but in RPG III the default is packed decimal. When a System/36 program and an AS/400 program are passing parameters to each other or accessing the same data, you must ensure that the programs process the data correctly according to their respective rules.

// LOAD and // RUN OCL Statements

As on System/36, a job step in the System/36 environment begins when the // LOAD OCL statement is run and ends when the program specified on the // LOAD OCL statement completes. The program that is running, as well as any other programs called either directly or indirectly from that program, are all considered to be running under the control of a // RUN OCL statement. All System/36 and AS/400 programs that run under the control of a // RUN OCL statement are considered part of a System/36 application. Any programs given control in the System/36 environment that do not run under the control of a // RUN OCL statement are considered part of an AS/400 application running in a System/36 environment.

For specific information on a particular language, refer to the manual for that language.

If you plan to mix System/36 programs and AS/400 programs within the same application, you need to understand how the programs run and interact with each other, as well as how they interact with System/36 environment and AS/400 system functions, in terms of using System/36 rules and AS/400 rules. These rules are discussed throughout this chapter.

The initial program of a System/36 environment job step, which may be either a System/36 program or an AS/400 program, should be called by a // LOAD and // RUN pair of OCL statements. The // LOAD and // RUN pair allows the System/36 environment to recognize the program as running in a System/36 job step. The System/36 environment allows System/36-like processing to occur for functions other than the program itself, such as OCL statements that take effect when either the // LOAD or // RUN is processed. The program itself runs according to the rules of the language in which it is written. For example, a System/36 program may begin with an input operation to the display station from which the user signed on, whereas an AS/400 program cannot issue an input operation until either the same program or another program sharing the same file has issued an output operation to the device. If the top-level program is not called by a // LOAD and // RUN pair, then it is considered to be an AS/400 application running in a System/36 environment job, and none of the special processing takes place.

Other programs needed in the System/36 environment application job step should be given control through the use of the high-level languages' external call instructions.

// LOAD Time Processing: When a // LOAD OCL statement is processed by the System/36 environment, the following actions occur:

1. A search is done for the program that is referred to on the // LOAD OCL statement.

The libraries that are searched are controlled by the following:

- If a library was specified on the // LOAD statement, that library is searched first, then #LIBRARY, and finally the libraries in

the user portion of the library list are searched.

- If a library was not specified on the // LOAD statement, the current library is searched first, then #LIBRARY, and finally the libraries in the user portion of the library list are searched.

If the program is not found, processing is ended and a message is issued stating the program on the // LOAD statement was not found.

If the program is found, the address of the program is saved and processing continues.

2. The program attributes are read and saved. These attributes indicate to the System/36 environment how this program should be run. Some of these attributes are:
 - Whether the program is a multiple requester terminal (MRT) program
 - Whether the program is a never-ending program (NEP)
 - Whether the program handles inline data

If a program is given control by another program that is running under the control of the // RUN statement (a call statement), the program attributes are searched for and read as if this program was on a // LOAD statement.

The program specified on the // LOAD statement, or the program that is given control by a program specified on a // LOAD statement, can be a System/36 program or an AS/400 program.

Note: A CALL CL command should never be used to give control to a System/36 program,

because none of the extra System/36 processing is performed when the // LOAD statement is run. A CALL CL command can be used to give control to an AS/400 program, but this should be avoided when running in the System/36 environment. The program that is called is considered part of an AS/400 application, and none of the special System/36 processing is done.

// RUN Time Processing: When a // RUN OCL statement is processed by the System/36 environment, the following actions occur:

1. Existing database files referred to by // FILE OCL statements within the job step are allocated (locked).
2. The work station referred to by // WORKSTN OCL statements within the job step is allocated (locked).
3. Performs override function.

Information gathered from OCL statements previously processed within the job step is used to build the override commands. The override commands insure that information provided on the OCL statements is used by the system when the file or device is opened. The following tables list OCL statements and parameters, and the corresponding override commands and parameters:

OCL Statement	Override CL Command
// FILE for disk	Override Database File (OVRDBF) command
NAME-	FILE()
LABEL-	TOFILE() MBR()

OCL Statement	Override CL Command
// SESSION	Override Intersystem Communications Function Device Entry (OVRICFDEVE) command
LOCATION-	RMTLOCNAME()
SYMID-	PGMDEV()
LWSID-	DEV()
GROUP-	MODE()
APPLID-	APPID()
BATCH-	BATCH()
HOSTNAME-	HOST()
FMHI-YES	HDRPROC(*USER)
FMHI-NO	HDRPROC(*SYS)
MSGPROT-	MSGPTC()
RECSEP-	BLOCK()
BLKL-	BLKLEN()
RECL-	RCDLEN()
TRANSP-	TRNSPY()
PARTNER-NORM	RMTBSCSEL(*YES)
PARTNER-ATTR	RMTBSCSEL(*NO)
BLANK-C	DTACPR(*YES)
BLANK-T	TRUNC(*YES)
SWTYP-AC	INLCNN(*DIAL)
SWTYP-MC	INLCNN(*DIAL)
SWTYP-AA	INLCNN(*ANS)
SWTYP-MA	INLCNN(*ANS)

OCL Statement	Override CL Command
// PRINTER	Override Printer File (OVRPRTF) command
NAME-	FILE()
DEVICE-	DEV()
LINES-	PAGESIZE(length *N)
LPI-	LPI()
CPI-	CPI()
FORMS-	FORMTYPE()
ALIGN-YES	ALIGN(*YES)
ALIGN-NO	ALIGN(*NO)
SPOOL-YES	SPOOL(*YES)
SPOOL-NO	SPOOL(*NO) DFRWRT(*NO)
COPIES-	COPIES()
CONTINUE-	SHARE()
PRIORITY-0	HOLD(*YES) OUTPTY(7)
DEFER-YES	SCHEDULE(*FILEEND)
HOLD-YES	SAVE(*YES)
HOLD-NO	SAVE(*NO)
IGCCPI-5	IGCCPI(5)
IGCCPI-6.7	IGCCPI(*CONDENSED)
SOSI-NORMAL	IGCSOSI(*YES)
SOSI-SHIFT	IGCSOSI(*RIGHT)
SOSI-DROP	IGCSOSI(*NO)
TYPE-IGC	IGCDTA(*YES)
EXTN-OFF	IGCEXNCHR(*NO)
JUSTIFY-	JUSTIFY()
FONT- (HEX)	FONT((DECIMAL) *NONE)
DRAWER-	DRAWER()
DRAWER-3	DRAWER(*E1)
ROTATE-	PAGRRT()
TEXT-YES	PRTQLTY(*NLQ)
TEXT-NO	PRTQLTY(*DRAFT)
EOFMSG-YES	FORMFEED(*CONT)
EOFMSG-NO	FORMFEED(*CUT)

OC L Statement	Override CL Command
// SYSLIST for printer	Override Printer File (OVRPRTF) command
PRINTER/printer id	DEV()
NOEXTN	IGCEXNCHR(*NO)

OC L Statement	Override CL Command
// WORKSTN	Override Display File (OVRDSPF) command
UNIT-	DEV()
EXTN-ON	IGCEXNCHR(*YES)
EXTN-OFF	IGCEXNCHR(*NO)

Note: The override of a printer file is always run regardless of whether a // PRINTER OCL statement is specified for the job step. If a // PRINTER OCL statement is not specified, the information for the session printer is used for the override.

4. The program specified on the // LOAD statements is called to begin running.
5. After the program specified on the // LOAD statement has completed, cleanup for this job step is done. This may consist of deleting files with RETAIN-S specified in a // FILE OCL statement or deallocating (unlocking) files that were allocated earlier.

Note: If a program running in the System/36 environment was given control with a CALL CL command rather than a // LOAD and // RUN pair of OCL statements, none of the special System/36 processing is done.

High-Level Language CALL Statement

The CALL statement in high-level languages is used to transfer control from one program to an other program. The program that transfers control is a **calling program**. The program that receives control from the calling program is a **called program**.

A program written in one language can call and pass parameters to a program written in the same language or in a different language. The programs can be System/36 programs, AS/400 pro-

grams, or a combination of both types of programs. Each program follows the rules of the language in which it is written. For language-specific information, refer to the specific language manual.

If the called program is going to do any input/output (I/O) operations, and that called program is a System/36 program, the file must be referred to on a // FILE OCL statement when the initial program in the job step is loaded.

If the called program doing the I/O is an AS/400 program and the name of the file in the program and on the disk are the same, a // FILE OCL statement is not necessary to process the file. If the name in the program and on the disk are not the same, a // FILE OCL statement is required when the initial program in the job step is loaded.

The search order for a called program in the System/36 environment is the same as the program that was initially loaded.

- If a library was specified on the // LOAD statement, that library is searched first, then #LIBRARY, and finally the libraries in the user portion of the library list are searched.
- If a library was not specified on the // LOAD statement, the current library is searched first, then #LIBRARY, and finally the libraries in the user portion of the library list are searched.

File Processing in System/36 Environment

This section discusses how the various files used in an application are processed when that application is running in the System/36 environment. Database files, printer files, display files, communication files, and other device files are covered.

Database Files

System/36 programs (for example, RPG II) must have a // FILE statement for each database file used within the program, and the entire // FILE statement is used to provide database support for the program. For AS/400 programs within the application (for example, RPG III), the // FILE statement is optional. If a // FILE statement is specified, all of the parameters are used, except

for dynamic file creation (DISP-NEW) and load-to-old (DISP-OLD).

For more information about using System/36 environment and AS/400 files, see Chapter 7, "Files."

File Processing

Database support in a System/36 program followed by an AS/400 program: For database support, the following conditions apply.

Note: Throughout this section, unless specifically noted otherwise, the term **open options** refers to the open options of input, output, and update, as well as the attributes of key feedback, arrival sequence, file-dependent I/O, and sequential-only processing.

- A // FILE statement is needed for each database file referred to by the System/36 program.
- The database files opened in the System/36 program are always shared when opened, except for files with DISP-OLD and all database files opened in a MRT program, for which the determination as to whether they are opened shared is based on AS/400 rules. See the *DB2 for OS/400 Database Programming* book for more information on sharing database files in the same job. If there are two separate job steps, these files are kept open until the System/36 environment determines whether there is a // FILE statement that refers to them for the next job step.
- A // FILE statement is optional for each database file referred to by the AS/400 program. If there are two separate job steps, any database file that does not have a // FILE statement applicable for the AS/400 job step is closed.

If there are two separate job steps using a database file from the System/36 program that is kept open for the AS/400 job step:

- If the AS/400 program opens the shared file and all of the open options are the same, the AS/400 program starts processing at the beginning of the file.
- If the AS/400 program opens a file that is not shared, or is shared but with different options, then the AS/400 program gets a new open of the file, and if the open

options are in conflict with each other, the file left open from the System/36 program is closed.

In the example with the System/36 program calling the AS/400 program, and both programs opening the same database file:

- If the AS/400 program opens the shared file, the AS/400 program starts processing the file at the point the files share (for example, the two programs share the cursor), regardless of whether the open options are the same.

Note: If the System/36 program is a MRT program or opens the database file as DISP-OLD, then the System/36 program does not necessarily do a shared open, because the determination as to whether the open is shared is based on the AS/400 rules. If the file in the System/36 program is opened shared, then the information above applies. If the file in the System/36 program is not opened shared, then the AS/400 program processes the file according to AS/400 rules.

- If the AS/400 program opens a file that is not shared, then the AS/400 program receives a new open of the file, and that file is not kept open for succeeding job steps.

Database support in an AS/400 program followed by a System/36 program: For database support, the following conditions apply:

- A // FILE statement is optional for each database file referred to by the AS/400 program.
- Database files opened in the AS/400 program are not kept open across job steps. Therefore, if there are two separate job steps, database files opened in the AS/400 program are not open when the System/36 program is called.
- A // FILE statement is required for each database file referred to by the System/36 program. Each database file opened by the System/36 program is shared when opened, except for those with DISP-OLD and all database files in a MRT program, for which the determination as to whether they are opened shared is based on AS/400 rules.

- When the AS/400 program calls the System/36 program, the System/36 program starts processing the file at the point where the AS/400 program left off (as if AS/400 operations had occurred in the System/36 program) if a database file in the System/36 program is the same as a database file that was opened shared in the AS/400 program, and both programs opened the file with the same options.

Database support in a System/36 program followed by a System/36 program: Both System/36 programs open all shared database files, and the effect is the same as if both System/36 programs are part of a single System/36 program. The only exception to this is for database files opened DISP-OLD and for all database files opened in a MRT program, which uses AS/400 rules to determine whether the file is opened shared.

Database support for an AS/400 application in a System/36 environment job: All database files are opened according to AS/400 rules, and any // FILE statements are ignored by the AS/400 programs.

Any database files opened in the System/36 program are opened shared and are kept open until the System/36 environment determines that they are not needed in the next System/36 job step. Since the CALL CL statement is not recognized by the System/36 environment as a job step, all database files opened in the System/36 program are still open when the AS/400 program is called. Consider the following:

- If the AS/400 program opens the file that was kept open from the System/36 program, and it is opened shared with the same options, the AS/400 program starts processing the file at the same point in the file that the System/36 was at when it ended.
- If the AS/400 program opens the file that was kept open from the System/36 program, and it is opened shared with different options, then any conflicting options are ignored, a corresponding diagnostic message is issued, and errors may occur.
- If the AS/400 program opens the file that was kept open from the System/36 program, but it is not opened shared, then the AS/400 program receives a new open of the file.

Note: If a CL command, issued during a procedure, requires a file it is processing to be closed, the System/36 environment closes the automatic shared open for the file. If the file (or member) is deleted, renamed, moved or restored by the CL command, the System/36 environment removes any // FILE statements for the file (or member). Any locks that were held through these // FILE statements are released. If a job file is renamed, moved, or restored by a CL command, the System/36 environment will not release the disk space occupied by the job file when the job ends. The System/36 environment will not close an automatic shared open of an alternative indexed or logical file which is based on the file being processed by the CL command. See “Shared File Opens within the Same Job” on page 7-37 for more information.

Externally-Described Files: Some file types have a detailed description of the file stored in the file itself. These detailed descriptions are called field-level descriptions, and these files are called **externally-described files**. That is, the description of the data related to the file is not represented by variable declarations you have coded in the program, but the description of the data is external to the program.

When a System/36 program processes an externally-described file there is no level checking done to see if the file description has changed since the program was created.

For more information on externally-described files and on level checking, see the *Data Management* book.

Program-Described Files: Some file types do not allow a detailed description to be used because field-level descriptions are not supported. These files are referred to as program-described files. That is, the description of the data related to the file is represented by variable declarations you have coded in the program. These are the types of files created by all System/36 environment functions, such as BLDFILE procedure and the #GSORT utility, or a file created by an RPG II program.

AS/400 programs can process these program-described files, which are created by System/36 environment functions. This is done by declaring

in the AS/400 program the variables that make up the records in the file.

For a more information on program-described files, see the *Data Management* book.

Naming Conventions: All database files follow the AS/400 object naming convention, even if those files were created in the System/36 environment. For more information on file naming conventions, see “Naming a Physical File” on page 7-1 and “Using File Members and Date-Differentiated Files” on page 7-34.

Printer Files

In the System/36 environment, printer file support is implemented in three parts:

1. Printer file attributes processing
2. CONTINUE-YES processing
3. Printer file open processing

Together, these parts produce printed output consistent with that produced on the System/36.

Attribute Processing: When a System/36 program opens a printer file, the attributes of that printer file are derived in this order:

1. File definition in the program
2. Any printer file overrides
3. The printer file opened

Two types of special printer overrides are generated whenever the // RUN OCL statement is processed:

1. Session printer (*PRTF) override
 - Parameters are specified on the // FORMS OCL statement or PRINT procedure.
 - Parameters are specified with a // PRINTER statement outside of a // LOAD // RUN, and CONTINUE-YES is specified.
2. File name specific override
 - Parameters are specified on any // PRINTER statement on which a NAME parameter is specified.

These overrides stay in effect until the program running under the // RUN OCL statement ends.

CONTINUE-YES Processing:

CONTINUE-YES processing is the one function that is unique to System/36 environment printer support. CONTINUE-YES processing combines the printed output from different job steps within one job into a single spooled file. It is enabled when a // PRINTER statement is specified with the CONTINUE-YES parameter. This function is very similar to SHARE(*YES) printer file processing using the OS/400 operating system in the AS/400 environment, but includes the ability to change the page size, overflow, and record length between job steps.

AS/400 programs and CONTINUE-YES: AS/400 programs may share the file if they open the file with SHARE(*YES) specified.

Attribute changing rules:

- Attributes are changed only between System/36 environment job steps.
- Attributes are changed only by System/36 programs.

Open Time Processing: When a System/36 program opens a printer file, special processing occurs, including:

- Printer file resolution
- Program-specific attribute processing
 - Page size
 - Overflow
 - Record length
- Special CONTINUE-YES processing

Printer file resolution

The OS/400 operating system requires the existence of a printer file to generate printed output. The System/36 does not. In order to allow System/36 applications to run in the System/36 environment, printer files named the same as the System/36 environment printer names are created in #LIBRARY.

At open time, a search is made for a printer file using the name specified in the program or in the override that may have changed the name. If the search cannot find the printer file, the printer file with the same name as the System/36 environment printer name is used instead.

Program-specific attribute processing

On the System/36, the program attributes take precedence over any overrides. On the AS/400 system, the overrides take precedence over the program. For attributes to be processed as they would be on the System/36, special processing at the printer file open must occur.

These attributes are the record length specified in the program as well as the page length and overflow, if specified.

During the open, the program's file description is analyzed and a special override is created reflecting the program's attributes. To ensure that the program attributes are honored, this special override cannot be overridden.

CONTINUE-YES processing

When the first printer file is opened while CONTINUE-YES is active, the file description is copied from the System/36 program into the System/36 environment work area. This description is then opened SHARE(*YES) and held open by the System/36 environment until a CONTINUE-NO parameter is specified on the // PRINTER OCL statement or until the outermost procedure in the job ends.

To accurately reflect all overrides, the changeable attributes from the initial program are not used when the file is opened with CONTINUE-YES active. This ensures that any file overrides are used as the default. (Not all programs may have page size or overflow specified.)

Naming Conventions: Within the System/36 environment configuration, AS/400 printer device names are mapped to System/36 printer device names. For example, PRT01 in the AS/400 system is mapped to P1 in the System/36 environment. AS/400 names must be used when using AS/400 functions. System/36 environment names must be used when using System/36 environment functions.

The System/36 environment provides an application interface program, called QEXCVTDV, in library QSSP that can convert AS/400 names to System/36 environment names or System/36 envi-

ronment names to AS/400 names. See the *System/36 Environment Reference* book for further details on QEXCVTDV.

Display and Communications Files

Many of the same rules apply to both display and intersystem communication function (ICF) files opened in the System/36 environment.

When a file is opened, the following actions occur:

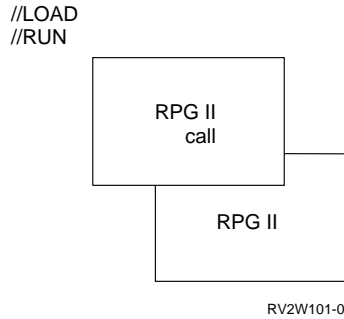
- The job's requester device is automatically acquired by the file. For display devices, the program device name is the same as the System/36 environment work station ID for the device. This can be changed by using a WORKSTN OCL statement and specifying the SYMID parameter. For ICF devices, the program device name ranges from 01 to 99 and is automatically generated when the job begins. The device name is unique among all the jobs currently running on the system.
- If the file is a display file and the user has coded one or more WORKSTN OCL statements with the REQD-YES keyword, the specified work stations are automatically acquired.

Read-Under-Format Processing: Read-under format (RUF) is the System/36 programming technique by which a program issues an output operation and a subsequent program issues the corresponding input operation.

The System/36 environment supports RUF operations by maintaining files that share their ODP with the most recently opened display or ICF file. This allows a program to close its file without ending an outstanding I/O operation.

Read-Under Format Considerations

with External Calls: The ability of one program to call another program on the AS/400 system allows the user to perform RUF operations that are not possible on the System/36. For example, consider the case where one RPG II program calls another RPG II program, as in the following figure.



In this scenario, the first program can issue a write operation to the requester, and the second program can issue the read.

Consider the following items when coding applications in which only System/36 programs are used:

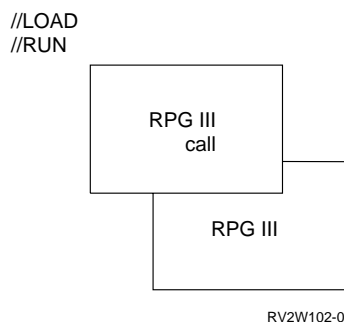
- A RUF operation may be started using the most recently opened file.

In the example above, when control returns to the first program, RUF may not be initiated unless the considerations outlined in “Read-Under Format Considerations with AS/400 Programs” are followed.

- A blank record or program data is returned only if the first I/O operation issued in a job step is a read and RUF is not in progress. This is done only once per job step.
- If a device is released by single requester terminal (SRT) program, it is not available until another display file or ICF file is opened.

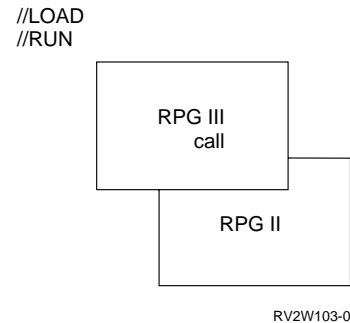
Read-Under Format Considerations

with AS/400 Programs: AS/400 programs may perform RUF-type operations through the use of shared opens. For example, consider the case where one AS/400 program calls another AS/400 program, as shown in the following figure:



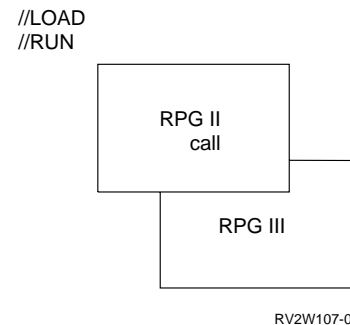
In this scenario, the first program can issue a write operation to the requester, and the second program can issue the read, but only if both programs use the same file and specify a shared open.

This technique can also be used to perform RUF-type operations between System/36 and AS/400 programs. However, both programs must use the same file, as shown in the following figure:



In the preceding figure, the RPG III program writes to the requester and the RPG II program reads from the requester.

AS/400 programs can also make use of the shared open performed by the System/36 environment. For example:



If both programs use the same file, and the RPG III program performs a shared open, then the RPG III program can complete a RUF operation started by the RPG II program.

Special Considerations: The following list contains other guidelines for display and ICF files:

- If a device is passed to or from a MRT job, no I/O should be performed with that device until a file is opened for that device by a System/36 program.

- If a device is released by a System/36 program, no I/O should be performed with that device until a file is opened for that device by another System/36 program.
- If a file is opened by both an AS/400 system and a System/36 program, the AS/400 program should not release the device.

Other Device Files

System/36 applications cannot use diskettes or tapes as I/O devices. These devices can be used only by the System/36 utilities (such as \$COPY and \$MAINT) as save and restore devices or to process exchange files. AS/400 applications, as well as CL commands, can use diskettes and tapes as I/O devices. The proper device naming convention must be considered when a diskette or tape device is to be used by a System/36 utility, an AS/400 application, or a CL command.

Naming Conventions: The following are the 2-character System/36 unit IDs for diskette and tape:

- I1 for diskette device
- T1 for tape drive 1
- T2 for tape drive 2
- TC for tape cartridge

During System/36 environment configuration, these 2-character System/36 unit IDs are mapped to AS/400 system device names for the diskette and tape devices supported on the system.

To indicate which device should be used by a System/36 utility, the appropriate 2-character System/36 unit ID must be specified on the FILE OCL statement, the procedure command, or the utility control statement.

To indicate which device should be used by an AS/400 application or CL command, the appropriate AS/400 system device name must be specified in the application or command. Mapping 2-character System/36 unit IDs to AS/400 system device names does not affect any AS/400 applications or CL commands that use the diskette or tape devices.

Chapter 18. Jobs and Job Processing

This chapter describes jobs and job processing on System/36 and on the AS/400 system in the System/36 environment.

Using Jobs and Job Processing

In the System/36 environment, a job is a unit of work done by the system. It is composed of one or more programs. A job step is a unit of work done by one program. A job that runs two programs has two job steps.

The concept of an interactive job is different from the System/36 concept of an interactive job. On System/36, when you sign on to the system, you are *not* running a job. When you start a procedure, you are running a job. On the AS/400 system, as soon as you sign on to the system, you are running a job. Because of this difference, commands that end jobs, hold jobs, and so on affect the interactive session. For example, if you end an interactive job, the user is signed off the system. For more information about the AS/400 system, see the *Work Management* book.

Note: In this chapter, the term *job* refers to the System/36 concept of a job and not the AS/400 concept of a job.

You start a job from your display station. The system names and processes a job. You can manage how your jobs use system storage, and you can schedule the order in which they run.

Jobs and Job Steps

A job is a unit of work done by the system. For example, an order-entry job runs one program to process orders and another program to print reports about the orders.

A job step usually begins with a LOAD OCL statement and ends with a RUN OCL statement. The following procedure contains one job step because only one program is loaded and run:

```
// LOAD PROG1
// RUN
```

The following example has two job steps because two programs are loaded and run:

```
// LOAD PROG1
// RUN
// LOAD PROG2
// RUN
```

The statements in a procedure control the files, display stations, printers, and other resources used by a program. For example:

```
// LOAD PROG3
// FILE NAME-CUSTOMER
// RUN
```

The statements in the preceding example have the following meaning:

- | | |
|-------------|---|
| LOAD | The program run is named PROG3. |
| FILE | PROG3 uses a disk file named CUSTOMER. |
| RUN | The program is run. This statement also indicates the end of the operation control language (OCL) statements for this job step. |

Starting and Ending Jobs

This section describes how jobs are started and ended.

Starting Jobs

You can start a job by doing any of the following:

- Entering OCL at a display station.
- Entering procedures at a display station.
- Entering menu options that run procedures.
- Using the JOBQ control command, or OCL statement, to place a procedure on the job queue. The job queue is a list of jobs waiting for the system to process them. The user can place batch jobs, which require no interaction with a user, on the job queue.
- Using the EVOKE OCL statement to start a procedure.
- Using the SBMJOB CL command.

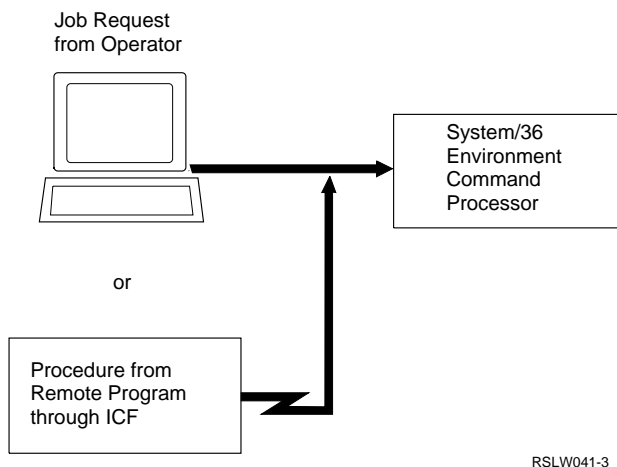
Note: There may be subtle differences from jobs submitted by EVOKE or JOBQ.

- Using the intersystem communications function (ICF) to have a remote program start a job.

An AS/400 job is active from the time you sign on until you sign off. A System/36 environment job is active from the time an OCL statement is recognized until you are returned to the menu where the OCL statement was entered.

Running Jobs

Start a job by selecting an item from a menu, or by typing an OCL statement or a procedure command. A remote program can also request that a job be run. As the following figure shows, when you request a job, a function called the **System/36 environment command processor** processes the request. The **command processor** in the System/36 environment is the part of the system that processes control commands and that passes procedure commands and operation control language statements to the initiator.



The System/36 environment command processor does one of the following:

- Passes control to the appropriate System/36 environment program for operator control commands
- Passes control to the System/36 environment initiator function

The System/36 environment command processor processes control commands.

As Figure 18-1 on page 18-3 shows, the System/36 environment initiator function reads and processes:

- Procedures (including multiple requester terminal (MRT) procedures that start MRT programs)
- OCL statements

When the initiator function processes a RUN OCL statement, the initiator function loads and passes control to the program, which begins running, as Figure 18-2 on page 18-3 shows.

When the program ends, the System/36 environment ending function ends the job step by freeing system resources the program uses. If more job steps follow, the OS/400 program returns control to the System/36 environment initiator function.

Figure 18-3 on page 18-4 shows that if no other job steps follow in the job, the OS/400 program ends the job and does one of the following:

- Returns control to the System/36 environment command processor for local jobs
- Ends the ICF session for remote jobs

The following information describes jobs and job processing in detail.

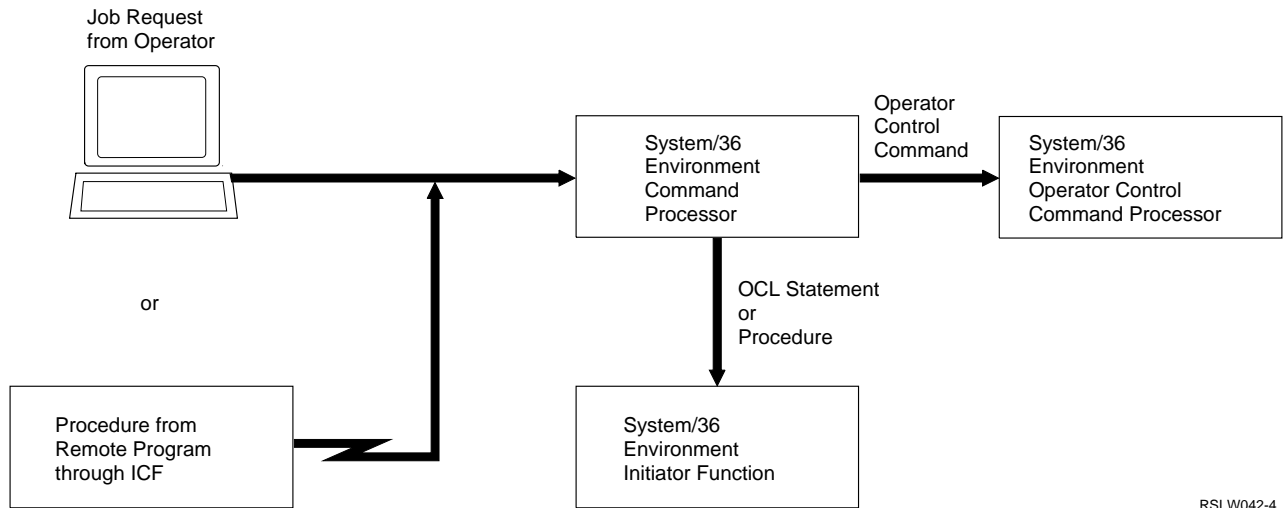
Using the System/36 Environment Command Processor

The System/36 environment command processor is the function that first processes information you enter. When you enter a command or select a menu item, or when a remote program sends a program start request using ICF, the System/36 environment command processor checks the command, or the command associated with the menu item, to determine whether it should start a job.

If the entry or menu item is a control command, the System/36 environment command processor does not start a new job. Instead, the command processor passes control to a System/36 environment routine that immediately processes the control command.

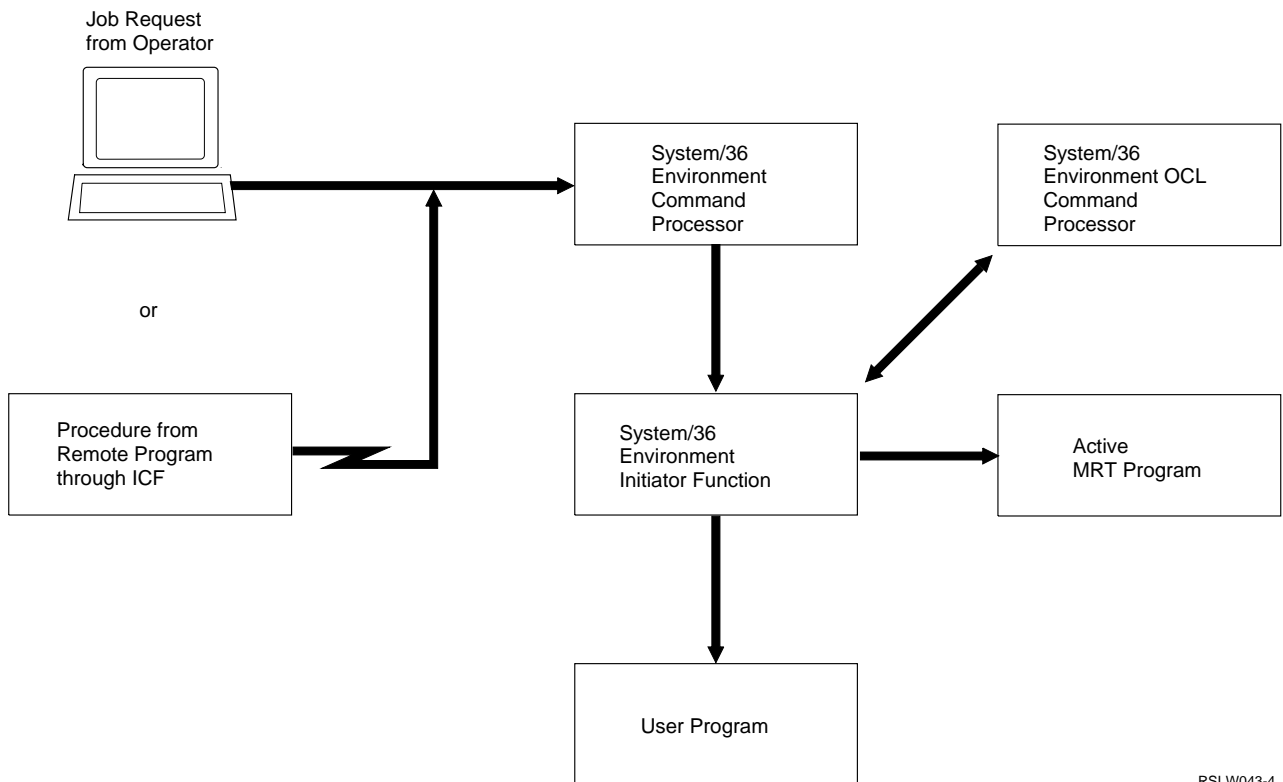
If the entry or menu item is a procedure, the System/36 environment command processor passes the procedure to the System/36 environment initiator function.

If the entry or menu item is an OCL statement, the System/36 environment command processor passes the statement to the System/36 environment initiator function.



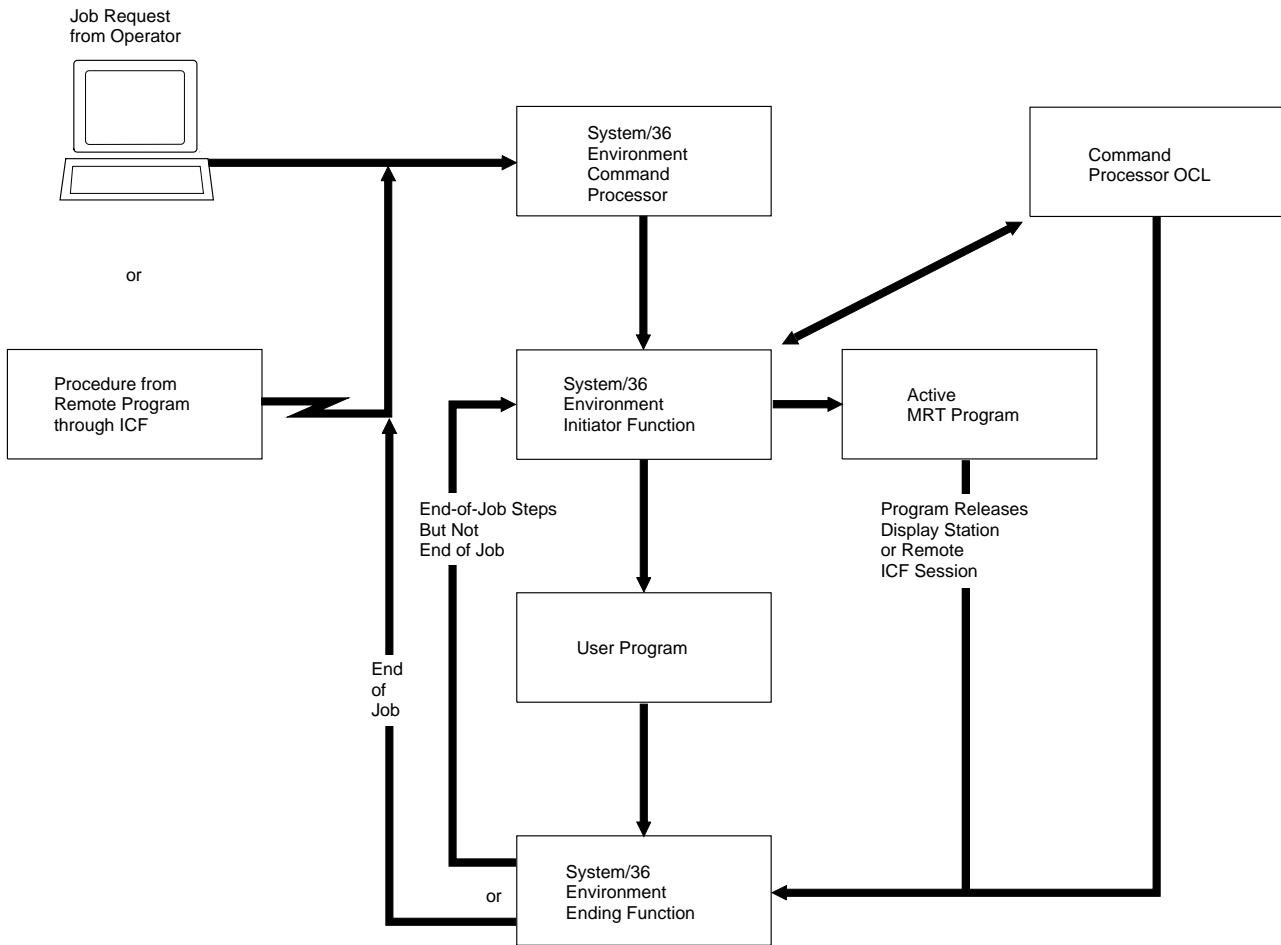
RSLW042-4

Figure 18-1. System/36 Environment Command Processor and Starting Function



RSLW043-4

Figure 18-2. System/36 Environment Command Processor and User Program



RSLW044-4

Figure 18-3. Command Processor and Ending Function

Using the Initiator Function

The System/36 environment initiator function finds and loads the programs, and passes control to the programs in a job. In addition, the System/36 environment initiator function:

- Processes procedure control expressions (substitution expressions and conditional tests)
- Processes OCL statements
- Ensures that required load members exist
- Ensures that the files the program needs are at the specified share level
- Acquires display stations for which you have specified REQD=YES on the WORKSTN OCL statement
- Releases requesting display stations if you specify RELEASE=YES on the ATTR OCL statement

Processing OCL Statements and Procedure Control Expressions

A System/36 environment function called system input processes statements entered from a display station or from a procedure member. After reading the statement, system input processes all the substitutions and the functions specified by the statements.

The procedure control expressions control system input processing. The *System/36 Environment Reference* book has more information about these expressions.

After processing a statement, system input returns the processed statement to the calling function. During job starting, the calling function is the System/36 environment initiator function. Therefore, the system returns to the System/36 environment initiator function all statements up to and including the RUN OCL statement. After a job

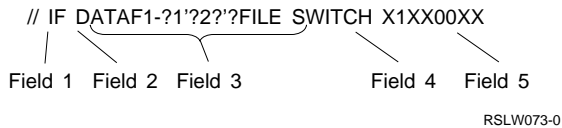
starts, the system returns the statements to the program that requested system input processing. \$COPY is an example of a system utility program that requires system input to process utility control statements.

Following are the fundamental rules of system input processing:

- System input processes a statement one field at a time, from left to right. Fields are delimited by blanks.
- Each time the system evaluates a substitution expression, system input goes back to the beginning of the field and begins processing again (allowing for nested substitution expressions).
- After the system does the substitutions, the length of the created statement must not exceed 512 characters (including spaces). The actual length of the statement before substitution can be up to 512 characters (including spaces).

System Input Processing Example:

The example in this section shows how system input works. In this example, system input processes the following statement:

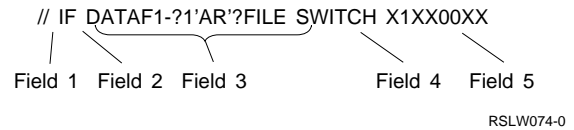


When the system reads the statement, parameter 1 does not have a value and parameter 2 has a value of AR. A file named ARFILE exists on disk.

The system input function does the following:

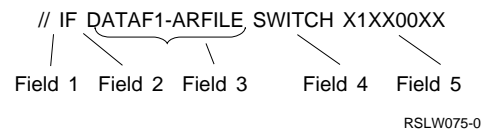
1. Identifies the first field as //.
2. Identifies the second field as IF, a valid procedure control expression.
3. Examines the third field and determines that the field contains a nested substitution expression. The innermost substitution is evaluated first. Therefore, system input substitutes the value of parameter 2, AR, in the

field. After the substitution, the statement looks like the following:



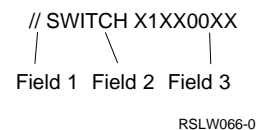
Because the system made a substitution, system input goes back to the beginning of field 3 and starts processing it again.

4. Examines the third field and determines that the field contains a substitution expression. System input does the substitution. In this case, parameter 1 does not have a value, and the value AR is substituted. The statement now looks like the following:



Because the system made another substitution, system input goes back to the beginning of field 3 and starts processing it again.

5. Examines the third field and determines that the field is an existence test for a file.
6. Evaluates the conditional expression formed by fields 2 and 3. The file ARFILE exists on disk, so the test is true. Because the test is true, system input discards the IF test (fields 2 and 3). Now the statement looks like the following:



After checking each field and determining that the statement requires no further substitution or system input processing, the system passes the statement back to the requester.

If the file ARFILE does not exist on disk (that is, the conditional expression in the original third field is false), system input discards the remainder of the statement and processes the next statement.

Ending Jobs

A job ends when:

- The last step in a job ends.
- A MRT program that is not a NEP releases its last attached device and the MRTDLY procedure attribute of the MRT procedure indicates that the MRT should end immediately.
- A MRT program that is not a NEP, with a MRTDLY procedure attribute indicating that the termination of the MRT should delay, waits the configured period of time but does not receive any new requesters before the delay period expires.
- A user selects option 3 in response to an error message.
- The job is ended, using the End Job (ENDJOB) or End Subsystem (ENDSBS) CL command. Either command signs the user off the system.

When a job or job step ends, the OS/400 program performs system actions necessary to end the job or job step. If more job steps need to be processed in the job, the System/36 environment returns control to the System/36 environment initiator function. If the job step is the last in the job, or if a MRT program releases its last requester, the OS/400 program ends the job, returns control to the System/36 environment command processor, and ends any active ICF sessions.

Normal Ending: When a job step ends, or when you select option 2 (Cancel job step and close files; new data is saved) in response to an error message, the System/36 environment does the following:

- Saves newly created resident files on disk
- Initializes work areas used by the next job step
- Deletes scratch files (RETAIN-S) used by the job

In addition, if the job step is the last of the job, the System/36 environment does the following:

- Deletes job files (RETAIN-J) used by the job
- Releases the requesting display station (if it is still attached to the job) and returns control to the command processor so you can request another job

- Ends the requesting ICF session if the program was requested by a remote program and the remote session is still active
- Frees any system resources the job used

Abnormal Ending: Abnormal ending of a program happens when any of the following conditions occur:

- A user selects option 3 in response to an error message.
- The user interrupts the program and selects option 2 (Cancel job and close files; new data is saved) from the System Request menu for all programs except MRT programs.

For MRT programs, option 2 (Cancel job and close files; new data is saved) releases the display station from the MRT program and continues with the next job step.
- The system detects an error condition during normal ending and cannot let the job normally end.
- Either the ENDJOB CL or the ENDSBS CL commands are processed.
- The // CANCEL procedure control expression is processed.

When one of the conditions described above occurs, the ending function is run automatically by the System/36 environment. Any job steps following the job step in error are not performed. When an abnormal end occurs:

- Files contain all updates done before the abnormal end.
- Additions made to shared and nonshared files remain in the files.
- If RETAIN-S was specified for a resident file in this job step, the resident file is not deleted.
- New files created by the current job step are deleted.

When the System/36 environment ending function is completed, the SSP0010 escape message displays. For interactive jobs, the System/36 environment ending function returns control back to the System/36 environment command processor. For noninteractive jobs, the jobs are ended. The escape message has the following implications:

- In an interactive job, the command remains on the command line, and an error message is displayed at the bottom of the menu.
- If the System/36 job was started by the STRS36PRC command in a CL program, the CL program should use the MONMSG command to determine if the System/36 procedure ended abnormally.
- If the System/36 job is running in a batch job that was started by a SBMJOB command or in a batch job stream, the job ending severity (ENDSEV keyword) is used to determine whether the batch job ended abnormally. If the batch job ends abnormally, no more requests are to be processed in the batch job, and an error message is sent to the submitter if a completion message was requested.

Managing and Scheduling Jobs

The system allows you to manage and schedule your jobs. For example, you can determine:

- How your programs use main storage with different run priorities
- The order in which the system runs your jobs using different job queue priorities

Note: The scheduling of jobs based on the job's priority is handled by the OS/400 program. For more information about job scheduling, see the *Work Management* book.

Job Priorities

Priority is the relative order of importance of items. For example, a job with high run priority should run before a job with medium or low priority.

You can specify the processing priority for a job or job step with the ATTR OCL statement. To have a job processed at the same priority each time it is run, specify the ATTR OCL statement in the job's procedure. Following are the processing priorities:

- High or Yes
- Medium
- Low or No

Note: See Figure 18-4 for information about the mapping of System/36 environment priorities to AS/400 priorities.

If your job does not specify a run priority, the system assigns it normal priority. The *System/36 Environment Reference* book describes the ATTR OCL statement.

The initiator function considers the following items regarding run priorities of jobs:

- Each job step (or program) can have its own priority, and the priority becomes effective as soon as the initiator function checks the OCL statements.
- If a job issues the //ATTR OCL statement to change the priority of the job, jobs created by this job will be started with the priority specified on the //ATTR OCL statement.
- The priority of a MRT program is not related to current requesters of the MRT. The following items determine priority of MRT programs:
 - The priority of the MRT procedure
 - The priority specified when you first requested the MRT procedure
 - The priority specified in the single requester terminal (SRT) procedure (when appropriate)

Job Priority Considerations: Use the run and job queue priorities to establish groups of jobs with shared characteristics. For example, you can run your testing jobs with one run priority and your production programs with another run priority.

You can assign run priority to programs based on the following values:

- Assign high priority to jobs that require fast response time at a display station, or quick system throughput.
- Assign low priority to jobs that do not use display stations, or that run for a long time.

Assigning a high priority to more than one job reduces the response time of all high priority jobs. Do not assign high-priority to jobs not debugged or tested.

The following table shows how System/36 environment priorities are mapped to AS/400 priorities:

Figure 18-4. Mapping of System/36 Environment Priorities to AS/400 Priorities

System/36 Environment Priority	AS/400 Priority
High/Yes	10
Medium	20
Low	50
No	20 (interactive, MRT and ICF evoked jobs)
No	50 (batch jobs)

Using Batch Job Immediate Support

The batch job immediate support is the default for starting batch jobs within the System/36 environment for // EVOKE, MRT jobs, and job steps started with // ATTR RELEASE-YES.

The batch job immediate support starts a batch job without using a job queue. The batch job runs in the same subsystem as the submitting job and is started with the attributes of the submitting job. This reduces the amount of time the batch job takes to get started.

Configuration options are provided by the CHGS36 and WRKS36 command to set:

- Method of job initiation (*IMMED) for this support
- Default job priority of the batch job
- Storage pool to be used, (*BASE is the default)

See the sections on “S/36 Environment Values” on page 3-9 and “S/36 MRT Security and Performance” on page 3-11 in this book for more details.

Using the Job Queue

A **job queue** is an object that contains a list of batch jobs waiting to be processed by the system. The system-recognized identifier for the object type is *JOBQ. Use the job queue to run batch jobs while you continue to use your display station for other work.

If job initialization is JOBQ, the System/36 environment submits batch jobs to the following job queues:

Job queue jobs

When you specify the Job Queue (JOBQ) command or the // JOBQ OCL statement, the system submits jobs to job queue QBATCH in library QGPL.

Evoke jobs

When you specify the // EVOKE OCL statement, the system submits jobs to job queue QS36EVOKE in library QGPL.

Note: Procedures placed on the QS36EVOKE job queue by the // EVOKE OCL statement will end abnormally if the job is held on the job queue, an IPL is performed, and the job is released after the IPL.

SBMJOB jobs

When you use the SBMJOB CL statement to submit System/36 procedures, the procedure must be entered using the Request Data (RQSDTA) keyword.

Nonrequester terminal (NRT) jobs

When you specify the // ATTR RELEASE-YES OCL statement for a SRT job, the system submits jobs to job queue QS36EVOKE in library QGPL. Do not end NRT jobs on this job queue.

Multiple requester terminal (MRT) jobs

MRT jobs are submitted to job queue QS36MRT in library QGPL. Do not end MRT jobs on this job queue.

When the AS/400 system is installed, there is no maximum for the number of job queue jobs that can be active at one time on the system.

Changing the maximum activity levels can cause problems for MRT and NRT jobs. When an SRT job starts a NRT or MRT job, the system suspends the SRT job until the NRT or MRT job starts.

Job Queue Priority Levels: You can specify six job queue priority levels, 0 through 5. Job queue priority 5 is the highest level, and 0 is the lowest level.

If you do not assign a priority level to a job using the job queue, the system assigns the job a level-3 priority. For example, assign level 5 to important batch portions of a printing application (such as the printing of payroll checks), but assign level 1 to program compilations.

The following table shows how System/36 environment JOBQ priority values are mapped to the AS/400 system job queue priority values:

System/36 Environment Job Queue Priority	AS/400 System Job Queue Priority
5	3
4	4
3	5
2	6
1	7
0	8

You must use System/36 environment priorities for System/36 environment commands and AS/400 priorities for AS/400 CL commands.

Changing Job Queue Priority Levels: Use the CHANGE JOBS command to redefine the active job values for jobq QBATCH in System/36 environment. Use the CHANGE JOBS, JOBQ command to set the number of job queue jobs you want active at one time in the system. For example, if you want three job queue jobs running at one time, enter CHANGE JOBS, JOBQ, 3. You can also use the Change Job Queue (CHGJOBQE) CL command to change the maximum number of active jobs allowed from all priorities of a job queue.

Use the CHANGE JOBS, PRIORITY command to set the number of job queue jobs of the priority you want active in the system. For example, if you want to run two priority-5 jobs at once, enter CHANGE JOBS, 5, 2. Also, you can use the CHGJOBQE CL command to change the maximum number of active jobs for a priority (you must use System/36 environment priority mapping).

The STOP JOBQ command sets the maximum number of jobs for job queue QBATCH to 0. For example, STOP JOBQ, 3 sets the maximum number of active jobs for priority level 3 to 0. Another example would be STOP JOBQ, ALL which sets the maximum number of active jobs to 0 for all System/36 environment job queue priorities. The START JOBQ, ALL command specifies that all jobs running at a priority that maps to a System/36 environment priority on the job queue can run. If no parameter is specified, ALL is assumed. All sets the maximum number of active jobs to 1 for all System/36 environment job queue priorities. This parameter value resets any values

set by a CHANGE JOB command. For example, START JOBQ, 3 sets the maximum number of active jobs for priority level 3 to 1. Another example would be START JOBQ, ALL which sets the maximum number of active jobs to 1 for the queue and 1 for all priorities. The maximum number of job queue jobs for the specified priority in effect before the STOP JOBQ command is *not* retained.

You can increase the number of jobs to run from the job queue by typing CHANGE JOBS, JOBQ, # OF JOBS. Using this command allows you to have multiple job queue jobs and sequential job processing.

To have 5 job queue jobs run at the same time, but also have job queue priority-3 jobs process sequentially, type CHANGE JOBS, 3, 1, and then type CHANGE JOBS, JOBQ, 5.

You specify the job queue priority of a job when you put it on the job queue using the JOBQ control command or OCL statement. For example, JOBQ 4, PAYLIB, PAYROLL puts the PAYROLL procedure (from the PAYLIB library) on the job queue with a level-4 priority. The *System/36 Environment Reference* book describes the JOBQ control command and OCL statement.

The System/36 environment commands that control the job queue affect only the QBATCH job queue in the library QGPL. The CHANGE, START, and STOP commands initially attempt to affect the QBATCH job queue entry in the subsystem description under which the job is running. If such a job queue entry does not exist, the commands attempt to affect the QBATCH job queue entry in the QBATCH subsystem description in libraries QGPL and QSYS. The CHGJOBQE CL command must be used to control the job queue if the QBATCH job queue entry is moved to a different subsystem description than those specified, or if the QBATCH subsystem description, under which the QBATCH job queue is running, is moved to a library other than QSYS or QGPL.

Job Queue Processing Priorities: Processing priority is different from job queue priority. Processing priority is the order in which jobs *already running* on the system are assigned system resources. You can assign jobs on the job queue a processing priority, but it becomes meaningful only when the job queue job starts running.

The following table shows the order in which the system considers jobs to be run (based on the order the jobs were placed on the job queue), and the job queue priority specified:

Job Queue Order	System/36 Environment Priority	Order System Runs the Job
Job 1	3	Job 6
Job 2	4	Job 2
Job 3	2	Job 1
Job 4	3	Job 4
Job 5	1	Job 3
Job 6	5	Job 5

You can start and stop individual jobs within each priority level. For example, if you do not want job 4 with priority level 3 to run, you use the HOLD JOBQ control command so the system selects jobs in the order shown in the following table:

Job Queue Order	System/36 Environment Priority	Order System Runs the Job
Job 1	3	Job 6
Job 2	4	Job 2
Job 3	2	Job 1
Job 4	3	Job 3
Job 5	1	Job 5
Job 6	5	

To run job 4, use the RELEASE JOBQ control command so job 4 is selected from the job queue.

The system does not run a job being held until it is released. The system runs other jobs that are not held at the same priority level.

You can prevent the system from running an entire priority level. For example, if you do not want to run jobs from priority level 3, use the STOP JOBQ control command. The program presents jobs to the system in the order shown in the following table:

Job Queue Order	System/36 Environment Priority	Order System Runs the Job
Job 6	5	Job 6
Job 2	4	Job 2
Job 1	3	Job 3
Job 4	3	Job 5
Job 3	2	
Job 5	1	

Use the START JOBQ command to have the system run the priority-3 jobs.

Priority Level 0: Priority level 0 is not automatically stopped at Initial Program Load (IPL) on the AS/400 system. To set up priority level 0 to run similar to priority level 0 on System/36, you must add the Change Job Queue Entry (CHGJOBQE) command to your start up procedure (#STRUP1 or #STRUP2). See "Running Jobs during Initial Program Load (IPL)" on page 18-13 for more information on start up procedures.

An example of the command is as follows:

```
CHGJOBQE SBSD(QGPL/QBATC)
        JOBQ(QGPL/QBATC) MAXPTY8(0)
```

This example sets the maximum number of jobs active for System/36 environment priority level 0 to 0 and stops priority level 0 jobs from running.

Assign priority level 0 to jobs that do not need to be run soon. For example, assign jobs priority level 0 if you want them run later in the day or overnight.

You can submit jobs with priority level 0 during the day, and start priority level 0 at night to run overnight.

Use priority level 0 with the system's automatic message response capability to submit jobs to be run overnight when the processing load of the system is not too great, and the system can run unattended. For more information about the system's automatic response capability, see Chapter 15, "Messages and Message Members."

Run Priority of Jobs on the Job

Queue: Each job on the job queue can have a different run priority. The **job queue priority** level determines which jobs are presented to the system for processing. The **run priority** determines the order in which the system runs jobs.

The run priority of a job placed in the job queue is **medium** unless you use the ATTR OCL statement to specify a different run priority. (See Figure 18-4 on page 18-7 for mapping of System/36 environment priorities to AS/400 priorities.) For example, the following // ATTR and JOBQ commands put the PAYROLL procedure (from the library PAYLIB) on the job queue with a job queue priority of 4, and a run priority of high:

```
// ATTR HIGH
JOBQ 4,PAYLIB,PAYROLL
```

If you use a procedure to place a job in the job queue, your job has the same run priority as the procedure.

Use the Change Job (CHGJOB) CL command to change the run priority of a job running or a job queue.

Evoking Other Jobs

Use the EVOKE OCL statement to start a new job from a job that is running. For example, a procedure with one or more job steps is similar to the job shown below:

```
* Start order entry program
// LOAD ORDENT
// FILE NAME-ORDFILE
// RUN
* Start order summary listing program
// LOAD PRTLST
// FILE NAME-ORDFILE
// RUN
```

The first program, ORDENT, uses the display station to enter new orders into the system. The orders are stored in a disk file. The second program, PRTLST, reads a disk file and prints the information on an order form. Because the PRTLST program does not require a display station, you could use the display to do other work while PRTLST runs. However, the display station remains attached to the procedure while the PRTLST job step is running, thus preventing the display station from doing other work.

If you create two procedures, and you use the EVOKE OCL statement to start the second procedure, your display station can do additional work while the PRTLST program runs. For example:

```
* Start order entry program
// LOAD ORDENT
// FILE NAME-ORDFILE
// RUN
* Start order summary listing procedure
// EVOKE PRTLST
```

The following PRTLST procedure is started:

```
* Start order summary listing program
// LOAD PRTLST
// FILE NAME-ORDFILE
// RUN
```

The *System/36 Environment Reference* book describes the EVOKE OCL statement.

Submitting Jobs to Run Later

You can submit a job to start at a later time using:

- The job queue
- The WAIT OCL statement

Using Job Queue to Run Jobs Later

See “Using the Job Queue” on page 18-8 for more information about using job queues. Use the JOBQ control command, or the JOBQ OCL statement to place a job on the job queue. For more information, see the *System/36 Environment Reference* book.

The JOBQ OCL statement allows you to place a job on the job queue from within a job that is already running. Use the HOLD and RELEASE commands to hold and release jobs on the job queue. Use the START and STOP commands to start and stop the processing of jobs from the job queue, or to start and stop the processing of job queue priority levels. To immediately start a job on the job queue, use the CHGJOB command and specify JOBQ(QGPL/QS36EVOKE).

Note: Procedures placed on the QS36EVOKE job queue by the // EVOKE OCL statement complete abnormally if the job is held on the job queue. When this occurs, an IPL is performed and the job is released after the IPL.

WAIT OCL Statement

The WAIT OCL statement causes a job to wait until a specified time, or until a specified period of time has passed. Once the system processes a WAIT OCL statement, the job does not resume processing until it meets the specified condition. The *System/36 Environment Reference* book describes the WAIT OCL statement.

If you use the WAIT OCL statement, consider the following points:

- You can use the ?TIME? substitution expression, along with the WAIT OCL statement, to see whether the job has waited until

a specific time. Processing can begin based on how long the job has waited.

- Use the STATUS control command with the USERS parameter to see if the job is waiting. The function field on the display will show DLY if the job is waiting.
- You can end a job even if it is waiting.
- If you place a job containing a WAIT OCL statement on the job queue, that job can prevent other jobs on the queue from processing. Processing can be delayed quite a bit, depending on the length of time the job specified uses the WAIT OCL statement.

The following example shows how to make a procedure wait until 4 p.m. (a time of 160000):

```
// IFF ?TIME?>160000 WAIT TIME-160000  
// EVOKE PROC1
```

If it is after 4 p.m., the system does not process the WAIT statement. For example, if the system ran the procedure at 5 p.m., PROC1 runs immediately. If you do not do the ?TIME? test, the PROC1 procedure does not run until 4 p.m. the next day.

Unattended System Operation: You can use default responses to have the system process programs without operators. For example, place certain programs in priority level 0 of the job queue during the day and start the job queue in the evening to have the system process these programs during the night. Any messages created by the system or the programs should have default responses created for them. See “Supplying Default Responses for Messages” on page 15-5 for complete information about creating default responses.

If you are planning to run programs while the system is unattended, you should consider the following points:

- Printed output:
 - Put the correct forms in the printer.
 - Put enough forms in the printer to print all jobs.
 - Align the forms correctly in the printer.

You can have reports write to the output queue and not print until an operator is

present. Do this by specifying PRIORITY-0 on the PRINTER OCL statement.

You can also have the system print your output while no operators are present and have one copy stored in the output queue (System/36 spooled file). Do this by specifying HOLD-YES on the PRINTER OCL statement. Therefore, you can print your output, but if something goes wrong with the paper or the printer, you can print the output that is saved in the output queue.

- Diskette processing:
 - Put a diskette in the diskette drive.
 - Do not run any jobs that require removal or insertion of diskettes in the diskette drive.
- Tape processing:
 - Place the tapes in the correct tape drives.
 - Do not run any jobs that require removal or insertion of tapes in the tape drive.
- Program processing:
 - Run programs that do not use a display station.
 - Run jobs that have been tested and are working correctly.
 - Have some method of starting your program again, in the event of program errors.

Submitting Jobs by Security Classification

You can specify that users with a particular security classification run certain security jobs. Following is a list of the security levels you can assign. They are listed from the highest to the lowest level of security:

1. Master security officer (M)
2. Security officer (S)
3. System operator (O)
4. Alternative console operator (C)
5. Display station operator (D)

The mapping of System/36 environment security levels to AS/400 user classes is shown in the following table:

System/36 Security Classification	AS/400 User Classification
Master Security Officer	*SECOFR
Security Officer	*SECADM
System Operator	*SYSOPR
Subconsole Operator	*SYSOPR
Display Station Operator	*USER

For more information about user classes, see the *Security – Reference* book.

When you submit a job, use the SECURITY conditional expression to determine whether an operator has the required security level.

The following example uses the SECURITY conditional expression to determine whether the user who submitted the job has a security classification of system operator or higher:

```
// IFF SECURITY-O CANCEL
// LOAD PROGRAM
// RUN
```

The SECURITY-O expression tests for the system operator security classification. If the user does not have system operator classification or higher, the IFF statement ends the job.

Also, you can use the SECURITY conditional expression to determine whether password security is active.

The *System/36 Environment Reference* book has more information about the SECURITY conditional expression.

Preventing Users from Ending Jobs

Specify CANCEL-NO on the ATTR OCL statement to prevent a job from being ended from the System Request menu.

Preventing Interrupted Jobs

Specify INQUIRY-NO on the ATTR OCL statement of a job to prevent another job from being started from the display station. The *System/36 Environment Reference* book has more information about the ATTR OCL statement.

If you are running MRT programs, place the ATTR OCL statement in the MRT procedure.

Preventing Informational Messages from Appearing

Most IBM-supplied procedures display informational messages at the display station from which you run them. When the job is composed of programs and IBM-supplied procedures that display informational messages (like DELETE), the messages can confuse a user who does not know the system's processing steps.

Messages sent to remote display stations cause the system to:

1. Store the display on disk.
2. Show the informational message.
3. Show the previous display again.

Sending messages to remote display stations decreases the remote station's performance.

Use the INFOMSG control command or OCL statement to select whether informational messages from procedures should appear. The *System/36 Environment Reference* book has more information.

Running Jobs during Initial Program Load (IPL)

When the system is initial program loaded, it automatically runs the #STRTUP1 and #STRTUP2 procedures if the following are true:

- #STRTUP1 and #STRTUP2 are on the system.
- The IPL is in attended mode.
- The operator is a System/36 environment user who has selected #STRTUP1 and #STRTUP2 to run on the IPL overrides display.

Other users can sign on to and use System/36 environment while these procedures are running.

You can use the Start System/36 Procedure (STRS36PRC) command in the AS/400 start-up program to specify that the #STRTUP1 and #STRTUP2 procedures run at each IPL (attended or nonattended). The start-up program is defined by system value QSTRUPPGM. For more information on the start-up program, see the *Work Management* book. The *System/36 Environment Reference* book has more information about the #STRTUP1 and #STRTUP2 procedures.

Running Jobs without Operators

You can run jobs on the system while no system operators are present. When you schedule jobs to run without operator supervision, consider the following:

- Use tested programs that work correctly. The programs should not require display stations.
- Use the automatic response function to respond to error messages. Automatic response is discussed in Chapter 15, “Messages and Message Members.”
- Use the HOLD-YES parameter of the PRINTER OCL statement if you are printing. The HOLD-YES parameter holds one copy of the output on the output queue. Use it to reprint the output queue entry without having to run the job again that created the printed output.
- Ending the print writers prevents output from printing. However, your programs run and the output is stored in the output queue.
- If you print the output from programs, your printers must have enough forms or paper to complete the jobs and the forms must be aligned properly.

End-of-Day Processing

Use the WAIT and Power Down System (PWRDWN SYS) CL commands together to automatically start end-of-day processing and turn the power off.

Following is an example of a nightly save procedure:

```
* Nightly Save Procedure
*
* Wait until 6 p.m. before beginning save
* and power-down sequence.
// WAIT TIME-180000
*
* Allocate the tape drive
// ALLOCATE UNIT-T1
*
* Save the master files on tape
SAVE CUSTMAST,,,VOL001,T1
SAVE ITEMMAST,,,VOL001,T1
SAVE ACCTMAST,,,VOL001,T1
SAVE SHIPMAST,,,VOL001,T1
*
* Power down the system
* in 30 minutes
PWRDWN SYS OPTION(*CNTRLD) DELAY (1800)
```

This procedure requires a tape in drive T1. It must be run before 6 p.m.

Note: The PWRDWN SYS CL command turns off the system even if there is system activity.

Job Date and Date Format

The System/36 environment supports a session date and a program date. The AS/400 system supports the job date. The **session date** in the System/36 environment is the date associated with a session. The **job date** is the date associated with a job. The job date usually assumes the system date, but can be changed by the user. The CHGJOB CL command changes the System/36 dates in the same way the DATE OCL statement does. A CHGJOB command between a LOAD OCL statement and a RUN OCL statement changes the job date only for the duration of the job step. After the job step ends, the job date is restored to the System/36 session date.

The CHGJOB CL command also affects the System/36 date format, as does the System/36 utility \$SETCF. The System/36 environment does not support the Julian date format (yyddd). If you use the CHGJOB command to change the job date format to anything other than a Julian date format, a corresponding change is made in the System/36 date format. If you specify Julian for the job date format, the System/36 date format is set to yymmdd (year-month-day), and the two date formats will not be the same.

When entering the System/36 environment, the System/36 session date format is always set to the AS/400 *job* date format if the AS/400 job date format is not Julian. If the job date format is Julian, both formats are set to yymmdd format. When the System/36 environment is exited, the

AS/400 date format is set back to Julian, unless the date format is changed while in the System/36 environment. Use the \$SETCF utility, the SET procedure, or the CHGJOB CL command to change the date format.

Chapter 19. Error Prevention, Detection, and Recovery

This chapter describes how to prevent, detect, and recover from System/36 environment failures and errors. It helps you anticipate and prevent the unexpected.

Once you understand the errors that can occur, and define the type of backup and recovery your system requires, you can make your own backup and recovery plan.

Types of Failures and Errors

The probability of a failure or error depends on the type of problem. This section describes the most common errors or failures and gives a general description of the required recovery.

System Failures

A system failure can damage a file or library. To recover from such a failure, restore a copy of the file or library and reapply all changes to the file or library since the last recorded save operation. Keeping up-to-date copies of your files and libraries makes recovery easier.

Disk Device Failures

A disk device failure can cause the loss of data stored on disks. If you do not use disk recovery tools, you cannot identify which data has been lost because the system automatically stores objects on disk. Disk recovery tools include:

- **Auxiliary storage pools (ASP)** that isolate storage of certain objects on certain disk devices. This protects against losing data on other devices and different ASPs.
- **Check sum protection** that takes data residing on several disk devices and combines it on another device. If the device fails, its contents can be recovered by recombining the data on remaining devices.
- **Mirroring** keeps two copies of the data on separate disk devices in an ASP. If the

device fails, its contents can be recovered from the other device.

If you do not use these recovery tools, you must recover from a disk device failure by reloading the entire system from backup media.

Power Failures

Power failures can damage programs and files in use at the time of the failure. To recover from a power failure, you must find which programs and files were in use when the power failure occurred. You must find the last successful checkpoint, the last point at which data in the files correctly and accurately reflected the corresponding data in other files. Changes made to the files after the last successful checkpoint should be removed. Changes from that checkpoint should be reapplied.

Equipment Failures

An equipment failure could be a modem failure, a problem with a communications line, a problem with external disks, or a display station problem. Often an error message is displayed when the system detects an equipment failure. You can cancel the job or continue the job. If you continue the job, a code may be returned to your program indicating the error type. Your program can check the status of the data you were processing, or make the necessary corrections.

Programming Errors

When a program unsuccessfully tries to use a display station, or when you find inaccurate information in a printout, a programming error has become apparent. Handle this type of error as you would a power failure (also debug the program producing the error). Other programming errors can be detected only after a long period of time, such as when the error causes inaccurate monthly or yearly summaries. This kind of error might be unrecoverable. To prevent errors, test your programs extensively before using them.

System Operator Errors

System operator errors are much more frequent than programming errors. For example, if the system operator inadvertently cancels an important job, data in your files might not accurately reflect the corresponding data in other files. It is important to detect and correct such a situation as soon as possible. As with a power failure, look for the last successful checkpoint and reapply transactions made since that checkpoint. You will have fewer transactions to reapply if you do checkpoints more often.

User Errors

Because users of applications are not always aware of how their system works, they may not know when an error occurs. As a result, users may be unfamiliar with the consequences of the error.

Common user errors include the following:

- Inadvertently powering off a display station
- Entering inaccurate data

To reduce the occurrence of errors, restrict the user to running options from an assigned menu. If a problem occurs, allow the program to detect the problem and stop the user from going further. You or the program (not the user) should handle the recovery. Recovery from user errors should be treated in the same way as recovery from system operator errors.

Error Prevention

This section describes how to prevent some failures and errors.

Using the Automatic Response Function

Use the RESPONSE and NOHALT procedures to automatically respond to system messages. When you use these procedures, the system automatically responds instead of displaying an error message. So, you can define a specific response that will be used automatically, rather than having an operator enter a response.

See “Supplying Default Responses for Messages” on page 15-5 for more information.

Preventing Unscheduled Ending of Jobs

One of the major causes of system operator or user error is the unscheduled ending of jobs.

Using the CANCEL-NO parameter of the ATTR OCL statement, you can prevent the system request cancel option (option 2 on the System Request menu) from being displayed. See “Preventing Users from Ending Jobs” on page 18-13 for more information.

Testing and Debugging Programs

Thoroughly test and debug your programs to minimize programming errors.

As you test your programs and procedures, use the LOG OCL statement or the LOG procedure to ensure that all OCL statements in your procedures are recorded to the job log. Logging OCL statements to the job log allows you to trace the activity of your procedures and programs.

The DEBUG OCL statement allows you to follow the logic of your procedures as you run them. Using the DEBUG OCL statement, you can specify whether procedure control expressions and OCL statements in your procedures should be listed as they are evaluated, and whether the procedures should stop after each job step. Like the LOG statement or procedure, the DEBUG statement enables you to evaluate your procedures and programs as you test them. The *System/36 Environment Reference* book has more information about the LOG OCL statement, the LOG procedure, and the DEBUG OCL statement.

Many programming languages allow you to debug operations during compile time. Some programming languages allow you to override debugging operations, preventing the debugging information from being added to your source code. However, the debug option is a good way to detect program errors before they occur. For more information about the problem analysis and program debugging capabilities of the programming language you use, see the corresponding language manual.

Using the WAIT and FILE OCL Statements

The WAIT OCL statement causes a job to wait until a certain time, or until a certain period of time has passed. Once the system processes a WAIT OCL statement, the job does not continue until a specified condition is met.

If a program needs to use a file or resource, and that file or resource is being used by another program or procedure, an error can occur. The WAIT parameter of the FILE OCL statement can prevent other programs from waiting for the resources owned by another job. The *System/36 Environment Reference* book has more information about the WAIT and FILE OCL statements.

See “Extendable Files” on page 7-23 for information about correctly extending files.

Allocating the Diskette or Tape Drive to a Job

You can use the ALLOCATE OCL statement to dedicate the diskette or a tape drive to a job. For example, you normally do not keep control of the drives between SAVE procedures. After FILE1 is saved, but before FILE2 is saved, another procedure on the system uses the drive. Because of this, your SAVE FILE2 procedure must wait until the other procedure is done.

To avoid allocating the drive longer than necessary, use the DEALLOC OCL statement to deallocate the drive. For example, your procedure saves three files and runs another type of job that did not use the drive. You could use the DEALLOC OCL statement to allow other jobs on the system to use the drive. The *System/36 Environment Reference* book has more information about these statements.

Error Detection

Errors can occur in the system functions or the input data that your program uses. Try to anticipate these problems by putting code in your programs to handle them. If such code is not in your programs, output data and files can contain errors you are not aware of.

Error Detection Subroutines

Special error detection subroutines are supplied by some programming languages, such as System/36-Compatible RPG II. If your programming language does not supply error-detection capabilities, include error-handling code in the program.

Your error-handling code performs functions such as correcting problems and stopping the program. A message should alert your users to the situation. You can define the message, severity levels, and various procedures the display station operator should perform when an error occurs.

When a routine detects an error, the program must determine what to do. Consider the following:

- Try the display station operation again. If your program cannot read a display format, the program can continue trying to read the display format.
- Save the data the program uses and end the program. If you do this, you must be able to use the saved data and start the program again.
- End the program without saving the data, or end the program and discard previously-saved data. You must load the program and enter the data again.
- Release the display station from a multiple requesting terminal (MRT) program.

For more information, refer to “Error-Handling Considerations” on page 19-8.

Program Language Error Detection

Each programming language supplies ways to detect errors as they occur. The following sections describe some of these error detection capabilities. For more information about the error detection capabilities and the error return codes for programming languages, see the appropriate language manual.

System/36-Compatible COBOL Language

Use the EXCEPTION/ERROR declarative to specify how the program handles input or output errors.

System/36-Compatible RPG II Language

The WORKSTN file information data structure (INFDS) contains status information the program checks to determine the type of error. Using the information in the INFDS data structure, the program determines which error conditions the INSEFR subroutine can process.

User-Coded Error Detection Routines

You can create your own detection routines for errors unique to your computer system. The design and purpose of such coding depends on your particular application.

Checking Return Codes in Procedures

Using the ?CD? substitution expression, your procedures can check a return code that indicates abnormal or problem situations (such as disk failures). Procedure logic flow can be based on the substituted return code, and the procedure can branch to recovery routines if a particular return code is found. Refer to the *System/36 Environment Reference* book for an explanation of how to use the ?CD? substitution expression and the return codes that you can use to control program logic.

Referring to the Job Log

Use the LOG OCL statement to log OCL statements to the job log. The system logs each procedure that contains a LOG OCL statement to the job log, regardless of the OCL statement job-logging indicator in that procedure. When you refer to the job log, you can create your procedures with the job logging-indicator set to off (for

better performance), but still have the OCL statements logged to the job log when it is necessary to debug the procedure. To look at the job log, use the Display Job Log (DSPJOBLOG) command. The system logs error messages and responses. For complete instructions on using the LOG OCL statement, see *System/36 Environment Reference* book.

Backup and Recovery

A **backup** is an alternative copy used as a substitute if the original is lost or destroyed. System backup plans and procedures should minimize the effect of any breakdown in the system and allow recovery from the breakdown as quickly and economically as possible.

Most backup procedures used for batch systems also apply to interactive systems. However, there are additional backup considerations in interactive systems. In interactive applications, the entire business can become dependent on the system. Therefore, the system must have reliable backup and recovery procedures.

Recovery is a series of steps you follow, or procedures you run, to restore data on the system. Recovery procedures can require removing all or some master files, restoring backed up master files, and running procedures that updated files to post transactions in the order in which they were originally processed.

The following information describes backup and recovery procedures for your system.

Equipment Backup

Mutual backup plans between users with similar equipment are generally not practical in an interactive environment. Even if an identically configured system could be located, the delays and difficulty of establishing the necessary data links prevent you from relying on this approach to equipment backup.

Equipment backup plans can vary according to the business requirements and the cost of using the plans. Equipment backup can range from having a spare display station to completely duplicating the central system. Determine the level of equipment backup required by comparing the cost

against the possible business loss if the backup is not there when it is needed.

Data Backup and Recovery

Because data can be damaged or destroyed by incorrect modification, a system failure, operator errors, or a natural disaster, keep backup copies of vital information. Backup procedures involve copying the vital information kept on the system, and then storing the copy in a safe location. For example, copy files and libraries to diskettes or tape, date and label them, and store them in a fireproof safe or at another location.

Establish and use a standard, well-documented backup procedure. Save master files and all files related to the master files at the same time. After you enter and edit new data (like batches of transaction records), copy it to disk, diskette, or tape. You can use these saved transactions during recovery procedures to make the master files current.

The level of data protection can vary greatly without a significant difference in the cost of implementation. Developing a backup plan and testing that plan can supply a return far greater than the time and effort involved. Developing a backup plan includes analyzing the effects of a system failure on each step of an application, defining the correct recovery procedures, and testing those procedures. It is important to test the procedures.

Before designing a backup and recovery plan, consider each type of data that you use:

- Historical data
- Master files
- Data processed but not distributed
- Data logged but not processed
- Data received but not logged

Historical Data: Because you do not use historical or permanently stored data in day-to-day processing, save it in a secure location, preferably off site. Any test of the backup plans should include trying to reconstruct current files from these historical files. The test should verify considerations such as:

- Are the files always available?
- Are the storage media compatible with the present hardware?

- Are the media protected against modification or deterioration?
- Are copies of the programs that process the data protected in the off-site location? Is the program documentation available?
- How current are master files the system reconstructs directly from these historical files? What would be the cost of making the reconstructed files more current?
- Are operating procedures and run books available?
- Is the stored data the correct data?

Master Files: Master files are used in day-to-day operations. Keep backup copies of all master files on site and update them on a daily basis. So that you can lose no more than one day's processing, and so the transactions for that day are logged in the transaction log file, keep the transaction log file until the daily update run has been successfully completed.

Data Processed but Not Distributed: If the system fails during the day's processing, you must consider data that has been processed but not distributed. Master files have been updated, but the output of the application is stored within the system and is not recoverable. You cannot run the transactions again because the master files would reflect double activity. Application programs that process the transaction file can flag the header record of all orders processed. The recovery plan would be designed to:

- Begin processing at the last order printed or distributed to a display station
- Process all the following transaction records, but not actually update master files or memo fields on flagged orders.

Data Logged but Not Processed: When the recovery program has processed the flagged orders, the records in the transaction file that have been received from the display stations, but have not been processed, must be recovered. If the system fails, users must not enter these records again. Users must be notified as quickly as possible not to enter more data because a recovery is in process. The recovery program must scan the transaction file and identify for the operator the last record correctly entered in the transaction file. If transactions are linked together by display

station within the transaction file, the logic of the recovery program can be more direct.

Data Received but Not Logged: Data received but not logged must be reentered by users. Because this data was in main storage at the time of the power or system failure, it is lost.

Loss of Transaction File Data: The transaction file is important in a successful recovery from a system failure. Users must reenter transaction file data that is lost or unusable. Transaction file data can be made less critical if master files are backed up each day. If the master files are backed up twice a day, a maximum of one-half day's input has to be entered again. Even orders that have been processed and printed have to be entered again so that the master files can be correctly updated. The printing of the output can be bypassed for the duplicates to prevent wasting forms.

Whichever method you use, compare the time required to enter records again to the time required to save files.

Backup and Recovery Methods

Programs and procedures can be designed to restore and recover all files, inform the operators about the last items correctly processed, and allow operations to continue from that point. This effort might involve using additional fields in records and using additional calculations in programs. Also, new files, programs, and procedures might be needed, particularly for recovery in an interactive environment. Businesses that depend most on their data processing system require the shortest recovery times, and should develop the most elaborate backup and recovery procedures. Regardless of their complexity, backup and recovery procedures should be well described so operators use them correctly.

Note: See the *System/36 Environment Reference* book for information about using the System/36 environment SAVE and RESTORE procedures for backing up and restoring.

There are three methods of backup and recovery:

- The first method requires the least design and programming effort, but requires the longest recovery time because transaction batches

are not saved. The operator must periodically save master files and files that the application updates to establish a point from which to recover (start the application again). For example, after all transactions have been posted, the operator might run a procedure that contains SAVE procedures to save all master files and their related files on diskette or tape.

Operators should keep a log of the work they do on the system. This manually kept log must be accurate if it is to be relied on during recovery. One way to keeping a log is to use a run sheet, as shown in Figure 19-1 on page 19-7.

Another way to obtain a log of work done on the system is to print the job log. This requires the system operator to do the following:

- Delete the files from disk.
- Restore the backup copies from diskettes or tapes to establish a point from which to recover.
- Reprocess all transactions that have been entered since the last backup was done.

Work done since the last backup must be done again. Transaction batches must be entered again because they are not saved. This method is adequate for a business that processes low volumes of data and that frequently backs up its data.

- The second method requires more planning and programming, but reduces the amount of recovery time required because it is not necessary to reenter the transactions. The operator must do the following:
 - Periodically save the master files and their related files.
 - Save batches of transactions at logical breakpoints in the application.

For example, at the end of each day after all transactions have been posted, the operator runs a procedure that contains SAVE procedures to back up master files and their related files on diskettes or tapes. As part of the transaction-posting procedure run during normal processing, a batch of transactions is saved on diskette or tape and deleted from disk. The operator labels the diskettes or tapes that contain the transactions to describe

The recovery method requires the operator to run the common recovery procedure. It lists the control file, and restores the files. The operator uses the log to run the application's procedures again in their original order. The common recovery procedure prompts the operator to insert the proper backup diskettes or tapes in the correct sequence.

This recovery method uses a program-created control log that is more accurate than a manually kept log. Because unnecessary procedures such as reprinting statements or reports could be skipped during recovery, this method supplies the quickest recovery of the three methods. This method is similar to the backup and recovery procedures used in some IBM-licensed application programs.

Service Aid Procedures

You may need to run service aid procedures to do problem analysis and correction. See the *System/36 Environment Reference* book for information about service aids procedures.

Error-Handling Considerations

This section describes error-handling considerations for:

- Disk storage full
- Display stations
- Printers
- ICF
- Databases

Disk Storage Full

On System/36, the operating system allocates all of the space requested by the user when a file, library, or folder is created (and allows them to be extended). System/36 requires that space for a file or library be contiguous. If sufficient space is not available, the request does not complete successfully.

If secondary storage (disk space) becomes filled on System/36, the operator can end the job that encounters the secondary storage full condition. For example, if a file is being restored from diskette, or a program is being compiled, and there is not enough secondary storage to hold the

file or the compiled program, the operator would end the restore or the compile. Other jobs active in the system will continue. The System/36 operator can then:

- Save files to diskette or tape
- Delete files to make more secondary storage available
- Condense space in a library, to eliminate the need to extend it

The AS/400 system is designed for single-level storage, where main storage and secondary storage are managed as a single large address space. The AS/400 system does not allocate all of the space for a file, for example, when it is created. The OS/400 program obtains space and extends a file as it is needed. A file or library does not need to occupy contiguous storage.

Because the OS/400 program manages secondary storage as a single large area, when it fills to capacity the entire system stops (abnormally ends). As space utilization approaches capacity, messages are sent to the system operator message queue, informing the operator that secondary storage use exceeds a user preset value (default 90% of secondary storage available).

Note: Do *not* let secondary storage fill to capacity. Re-IPLing and recovery can be a long process.

For more information on secondary storage management and recovery, refer to the Auxiliary Storage Pool (ASP) threshold description in the disk recovery chapter of the *Backup and Recovery – Advanced* book.

Display Station Device Error Considerations

When a display station device error occurs on System/36, the System/36 operating system generally displays a message at the system console informing the operator of the error. The operator can select either to have the system or application program retry the failed operation, or to have the system end the job and sign the user off.

If the operator returns control to the System/36 program to retry the operation, the general System/36 programming action is to reissue the failed operation. If the operation completes successfully the program continues. If the operation

does not complete successfully, a new message is sent to the console and the operator again can choose to retry the failed operation, or end the job.

When the System/36 programs are migrated to the System/36 environment, you may need to modify System/36 programs that are designed to recover from display station errors by reissuing the last operation.

The OS/400 program generally returns control to the application program when a device error occurs, and allows the program to determine the appropriate recovery action. In some cases the device error return code the program receives can be corrected by reissuing the operation in error. The system ends the program if the program keeps reissuing an operation to a device with a permanent error (ends program after 6 attempts). Because a message is not necessarily sent to the system operator, the error recovery routine in an application program should prevent it from getting into an infinite loop. An infinite loop could occur when the program releases and reacquires a display station, and the device error is permanent.

The following list describes some of the possible error return codes:

2800 Error Indication: Your program has issued a release operation for the requester device, and is now attempting to use it. Because that device was released from your program, this operation was not performed, and any further attempts to use the requester display station will result in another 2800 return code.

Used in the System/36 environment only.

Recovery Action: Continue local processing, end your program, or reacquire the requester device. Your program may be in error; you should correct it so that the release operation is issued after all use of the requesting device has been completed.

Message:

CPF4761 (Escape)

3200 Error Indication: Your program has attempted to acquire a display device for which it is not authorized. The acquire was not successful and any attempts to use that display will fail.

Used in the System/36 environment only.

Recovery Action: If the operation was an open, close the file, correct the problem, then reissue the open. If the operation was an acquire, correct the problem and reissue the acquire. For authority errors, obtain authority to the device from your security officer or the device owner.

Message:

CPF5279 (Escape)

3800 Error Indication: The acquire operation was not successful because the display device you are acquiring is in use in another process or is not varied on. The session was not started.

Used in the System/36 environment only.

Recovery Action: Vary on the device or wait for the display station to become available, then reissue the acquire operation. Otherwise, you may continue other processing or end the program. The WRKCFGSTS command may be used to determine whether the device is in use or not varied on.

Consider increasing the WAITFILE parameter with the CHGDSPF or OVRDSPF command to allow more time for the device to become available.

Messages:

CPF4282 (Escape)

CPF4285 (Escape)

CPF5332 (Escape)

CPF5333 (Escape)

Refer to the Display Station File chapter in the *Application Display Programming* book for more information regarding additional coding considerations for display station error handling.

If the display station is a remotely attached device, communications line and controller errors can also occur. You can find additional information about remote display station error processing in the *Remote Work Station Support* book.

When coding for display station device error recovery, consider the following:

- Releasing a display station associated with a MRT returns the requester to the job that the

MRT was called from. The MRT will be unable to reacquire the display station. You may need to:

- Prevent a partial transaction from being entered by the device that was released
- Prevent control from being prematurely returned to the job that called the MRT program (for example, before a local data area used to communicate between the jobs is set up).

Other display stations associated with the MRT could be interrupted for a period of time, while specific operations are issued to the display station in error.

- For some errors the recovery action is to close the file and to reopen it. System/36-Compatible RPG II performs an implicit OPEN and CLOSE function for the user when the program begins and ends. Because the OPEN and CLOSE operations are not available to the System/36-Compatible RPG II programmer, the program should terminate.

If a System/36 environment program is not designed to handle display station device errors, the operating system or high-level language generally terminates the program.

Display Station Device Error Recovery

On the AS/400 system, you can choose a device error recovery action for a particular session, or for the entire system. Enter the Display Job command (DSPJOB CL) and select option 2 from the Display Job menu to display the current device recovery action for the session. Use the Change Job command (CHGJOB CL) to change the device recovery action for the session (DEVRCYACN parameter).

There are several possible values for the device recovery action job attribute and each results in a different device recovery action in the System/36 environment:

*MSG

The application program requesting the input/output operation receives an error message indicating the input/output operation

failed. This is the default error recovery action.

*DSCMSG

The job is automatically disconnected when the input/output operation is detected. When the job is reconnected, the system can communicate with the device again.

*DSCENDRQS

The job is automatically disconnected when the input/output operation is detected. When the job is reconnected, the system ends the current job and the system can communicate with the device again. For interactive sessions, your menu is displayed.

*ENDJOB

The job and the session are ended. A job log is produced for the session.

*ENDJOBNO LIST

The job and the session are ended. A job log is not produced for the session.

For more information about the DSPJOB CL command, CHGJOB CL command, or device recovery action, see the *Work Management* book.

Printer Device Error Considerations

As on System/36, most printed output is spooled to secondary storage, rather than sending the output directly to the printer from the application program. Printer device error recovery and restarting of a device after an error is handled by the OS/400 spool support. Additional information on spool error handling and printer considerations can be found in the *Printer Device Programming* book.

Programs which use print files that go directly to the device, instead of being spooled, should close the file and reopen it when an error occurs.

ICF Error Considerations

On the System/36, if a permanent communication error occurs when two programs are communicating using a System Support Program-Interactive Communications Function (SSP-ICF) session, the session is released by SSP-ICF data management.

On the AS/400 system, the ICF session is automatically released by the OS/400 ICF support. If the ICF session is attached to a multiple requester terminal (MRT) program, the session is released from the MRT program. The MRT can continue to support other ICF sessions or display stations that are attached to the MRT program. When the session is released by the OS/400 ICF support due to a permanent error, the source program can attempt to acquire a new session with the target system. If the acquire is successful, the source program can continue sending and receiving data. If the acquire is unsuccessful, the source program can continue to support other ICF sessions or display stations associated with the application program. If no other ICF sessions are active, the source program can end.

In addition, the AS/400 system has support to handle program loops that occur because of permanent device errors. When an ICF session encounters a permanent device error, the application program receiving the error usually goes into

a retry loop if the program is not written to handle the error. The system limits this loop to only five iterations before ending the job with a reduced priority, thus reducing the overhead associated with these errors.

You can find additional information about processing communications errors in the *ICF Programming* book.

Database File Error Conditions

The AS/400 system attempts to recover from any secondary storage device errors that occur. If control is returned to the program, indicating a device error has occurred, the program should close the file and end the program. The *DB2 for OS/400 Database Programming* book has a chapter that describes handling database file errors in a program. The *Backup and Recovery – Advanced* book also discusses the procedures an operator could use to recover a file that becomes unusable.

Chapter 20. System/36 Environment National Language Support

This chapter describes the national language support for the System/36 environment

System/36 Environment Multiple Language Support

Multiple language support deals with entering data and displaying data in different national languages on the same system. This data can be broken down into two distinct groups:

- IBM-supplied data
- User-supplied data

Multiple Language Support for IBM-Supplied Data

IBM-supplied data, such as help prompts, help text, menus, and messages, is part of the System/36 environment. This data is translated by IBM and shipped to you in various languages. The System/36 environment can display this data in various national languages on the same system. This allows information in one national language to be presented to one user while information in a different national language is presented to another user. The presentation of this data is based on the current setting of the library list for the job.

The System/36 environment is installed with the IBM-supplied data in library QSSP in the primary national language of the system. For example, if the primary national language of the system is English, library QSSP contains the English version of the data. Secondary national languages are installed in separate national language libraries. For example, French is contained in library QSYS2928, and German is contained in library QSYS2929.

If you want the System/36 environment data presented in a language different from the primary national language, change the library list so the desired national language library is before the primary national language library (QSSP) in the library list. For example, if you want the French version of data, type CHGSYSLIBL QSYS2928 to

place the French version of the data at the top of the library list.

If you usually want a national language different from the primary national language, you can set up the initial library list for the job. To do this, define an initial program (in the user profile) that uses the Change System Library List (CHGSYSLIBL) command to add the desired national language library to the front of the library list.

The use of the CHGSYSLIBL command is restricted when shipped by IBM. Use one of the following methods to make the CHGSYSLIBL command available to all users:

- Use the Edit Object Authority (EDTOBJAUT) command so all users are authorized to run the CHGSYSLIBL command.
- Write a CL program containing the CHGSYSLIBL command. A security officer should write and own the CL program, and the program should adopt the security officers profile.

For information on security, see the *Security – Basic* book. For information on writing CL programs and commands, see the *CL Programming* book.

Multiple Language Support User-Supplied Data

User-supplied data, such as database files, display files, menus, and help text, is part of the user application. This data is not necessarily translated when it is created.

Maintaining data integrity when character data is passed from system to system or user to user is critical. The System/36 environment can maintain the data integrity for database or source files on all SAA* systems. Tagging the character data in these files with a Coded Character Set Identifier (CCSID) ensures this integrity. The CCSID identifies the code points used to represent the character data. The CCSID value of the file is then used to encode the character data as needed to

preserve its meaning. Each job that is running on the system also has a CCSID associated with it. These two CCSIDs (file CCSID and job CCSID) are used to maintain the integrity of the data that is passed between the job and the file.

If data is being read from a database or source file and the CCSID of the file is the same as the job CCSID, no encoding of the data must be done. If the CCSID of the file and the job CCSID are different, the data is encoded to match the CCSID of the job.

If data is being written to a database or source file and the CCSID of the file is the same as the job CCSID, no encoding of the data must be done. If the CCSID of the file and the job CCSID are different, the data is encoded to match the CCSID of the file.

The job CCSID value is the same as the system CCSID value unless it is changed through the user profile or a Change Job (CHGJOB) command.

The file CCSID for all database files created by the System/36 environment functions have a CCSID of 65535 with the following exceptions:

- A file restored using the RESTORE procedure that was saved from another AS/400 system. This file has the same CCSID as it did on the system it was saved from.
- A file copied using the COPYDATA procedure and the input file is an externally described file. If the output file is externally described, the CCSID of the new output file is the same as the input file.

The CCSID for all source files created by the System/36 environment functions is set to the job CCSID with the following exceptions:

- The source files in a library that were restored to the system using the RESTLIBR procedure. The CCSID of these source files will be the same as the CCSID of the saved file.
- A QS36PRC source file that contains IBM-supplied procedures has a CCSID of 65535.

Multilingual System Environment

A multilingual system environment is where you are running jobs that have a job CCSID that is different from the system CCSID. If this is true for your system the following should be considered:

- The CCSID of the QS36PRC source file that contains the procedures you run should have a CCSID of 65535. This must be done so any variant characters (such as \$, # and @) in the procedure do not get converted. For example, if one of the procedures has a // LOAD \$COPY in it and the CCSID of the source file is 273 for Germany, the \$ has a value of hexadecimal '5B'. The CCSID of the job that is to run that procedure is 277 for Denmark. The \$ will be converted to a hexadecimal '67' when it is read from the source file. The \$COPY utility program would not be found because the \$ in the name in the library is a hexadecimal '5B' and an error message would be issued. If the CCSID of the QS36PRC source file was 65535, no conversion would take place and the \$ would remain a hexadecimal '5B'.
- The CCSID of the QS36PRC source files should also be looked at. If any of the source in these files contain variant characters, a CCSID of 65535 should be considered.
- The sort specifications used by the Disk Sort utility should also be looked at. If the input file for the sort was created by the System/36 environment it has a CCSID of 65535, indicating no conversion of character data should take place. If the sort specifications that are to be used for the sort are read from a source file where conversion does take place, some of the include, omit, and record characters may not be the same as those in the file. In this case, the CCSID of the source file should be changed to 65535.

If the input to the sort is from a file whose CCSID is not 65535, the CCSID of the source file with the sort specifications should be the same as the CCSID of the input file.

More information on CCSIDs can be found in the *National Language Support* book.

System/36 Environment Double-Byte Character Support

Characters defined by 1 byte of data (alphanumeric characters) are called **single-byte** characters and are part of the **single-byte character set** (SBCS).

Because some national languages include thousands of characters, a single byte cannot uniquely represent each character in the system. In the double-byte version of a national language, each character is represented by 2 bytes of data. Characters defined by 2 bytes of data are called **double-byte** characters and are part of the **double-byte character set** (DBCS).

The System/36 environment can process data for double-byte national languages. Following are examples of double-byte national languages:

- Japanese Kanji
- Traditional Chinese
- Simplified Chinese
- Korean Hanguel

You must have special display stations and printers to display, enter, and print double-byte characters.

AS/400 Double-Byte Character Set System Value

The AS/400 system maintains a DBCS system value (QIGC) that indicates whether or not you have installed the DBCS version of the operating system.

The AS/400 DBCS system value is used by the System/36 environment to control what action is performed for the following functions:

- **Creating or changing System/36 print files**

This can be done when a new printer is configured on the system or the Change S/36 Configuration (CHGS36) command is run.

If the DBCS system value indicates the system is capable of handling DBCS data, the description of the printer is checked to see if it is DBCS capable. If it is, the print file is created or changed to allow DBCS data. If it is not, the print file is created or changed not to allow DBCS data.

If the DBCS system value indicates the system is not capable of handling DBCS data, the description of the printer is not checked, and the print file is created not to allow DBCS data. Any existing print files are not changed.

- **Processing the PRINTER OCL statement**

If the DBCS system value indicates the system is capable of handling DBCS data and any of the following parameters were specified on the PRINTER statement: IGCCPI, SOSI, TYPE-IGC or EXTN, an Override Printer File (OVRPRTF) command is run with the corresponding keywords.

If the DBCS system value indicates the system is not capable of handling DBCS data and any of the following parameters were specified on the PRINTER statement: IGCCPI, SOSI, TYPE-IGC or EXTN, the values are ignored.

- **Librarian Functions**

If the DBCS system value indicates the system is capable of handling DBCS data, shift-out (SO) and shift-in (SI) characters are scanned for when copying data from the READER.

AS/400 Double-Byte Character Set Job Attribute

The AS/400 system maintains a DBCS job attribute for every job running on the system. The job attribute indicates if the job is capable of handling double-byte character data. The job attribute is initially set when you sign on to a work station or when a batch job is started. The job attribute is based on the job CCSID value and the DBCS system value.

If the job CCSID is mixed and the DBCS system value indicates the DBCS operating system is installed, the job attribute indicates the job is capable of handling DBCS data.

If the job CCSID is SBCS and the DBCS system value indicates the DBCS operating system is installed, the job attribute indicates the job is not capable of handling DBCS data.

If the job CCSID is SBCS and the DBCS system value indicates the DBCS operating system is not

installed, the job attribute indicates the job is not capable of handling DBCS data.

Note: The job CCSID cannot be mixed if the DBCS system value indicates the DBCS operating system is not installed.

The AS/400 job attribute is used in the System/36 environment to control what action is performed for the following functions:

- **Creating a System/36 menu**

This can be done by running the BLDMENU procedure or loading the \$BMENU utility program.

If the DBCS job attribute indicates the job is capable of handling DBCS data, the option and command line of the menu are created with the open field attribute. The source file and display file created for the menu allows DBCS data.

If the DBCS job attribute indicates the job is not capable of handling DBCS data, the option and command line of the menu are created with the alphameric field attribute. The source file and the display file created for the menu does not allow DBCS data.

- **Creating System/36 messages**

This can be done by running the CREATE procedure or loading the \$MGBLD utility program.

If the DBCS job attribute indicates the job is capable of handling DBCS data and a Message ID Character (MIC) of A000 appeared in the source member before the messages, the following actions take place:

- The third character of the message prefix is changed to a Z (that is, USZ).
- Shift-out (SO) and shift-in (SI) characters are inserted when a IGC string is continued on another line.
- The source file created for the Create System/36 Message File (CRTS36MSGF) command allows DBCS data.

If the DBCS job attribute indicates the job is not capable of handling DBCS data, the following actions will take place:

- The third character of the message prefix is not changed to a Z (that is, USR).
- The scanning of shift-out (SO) and shift-in (SI) characters is not done.

- The source file created for the Create S/36 Message File (CRTS36MSGF) command does not allow DBCS data.

- **Creating System/36 display files**

This can be done by running the FORMAT procedure or loading the \$SFGR utility program.

If the DBCS job attribute indicates the job is capable of handling DBCS data, the following actions take place:

- All input fields are created with the open field attribute.
- Syntax checking allows IGC constants.
- Shift-out (SO) and shift-in (SI) characters are inserted when a IGC constant is continued on another line.
- The source file and display file created will allow DBCS data.

If the DBCS job attribute indicates the job is not capable of handling DBCS data, the following actions will take place:

- All input fields are created with the alphameric field attribute.
- Syntax checking does not allow IGC constants.
- The scanning of shift-out (SO) and shift-in (SI) characters is not done.
- The source file and display file created do not allow DBCS data.

- **Creating System/36 data files**

This can be done by running any System/36 procedure or utility program that creates a System/36 data file or any user application program that creates a System/36 data file.

If the DBCS job attribute indicates the job is capable of handling DBCS data, the file is created allowing DBCS data.

If the DBCS job attribute indicates the job is not capable of handling DBCS data, the file is created not allowing DBCS data.

- **Updating the Auto Response value for messages**

This can be done by running the RESPONSE procedure or loading the \$ARSP utility program.

If the DBCS job attribute indicates the job is capable of handling DBCS data, the DBCS messages are updated. An example would be

those messages that have an X as the third character in the prefix, such as USX.

If the DBCS job attribute indicates the job is not capable of handling DBCS data, the SBCS messages are updated. An example would be those messages that do not have an X as the third character in the prefix, such as USR.

- **IGC-related help prompts shown for OCL and PCE**

This can be done by running the HELP procedure or pressing F4.

If the DBCS job attribute indicates the job is capable of handling DBCS data, the help prompts will have IGC-related parameters.

If the DBCS job attribute indicates the job is not capable of handling DBCS data, the help prompts will not have IGC-related parameters.

- **Librarian functions**

This can be done by running the BLDLIBR, TOLIBR, RESTLIBR, JOBSTR, and LIBRLIBR procedures or loading the \$MAINT utility program.

If the DBCS job attribute value indicates the job is capable of handling DBCS data, the QS36PRC (procedure members) and QS36SRC (source members) source files are created allowing DBCS data.

- **Data file copy functions**

This can be done by running the SAVE, RESTORE, COPYDATA, or LISTDATA procedures or loading the \$COPY utility program.

If the DBCS job attribute indicates the job is capable of handling DBCS data, any intermediate files created during the operation allow DBCS data.

If the DBCS job attribute indicates the job is not capable of handling DBCS data, any intermediate files created during the operation do not allow DBCS data.

- **Force a program dump**

This can be done by taking a D option to any System/36 environment message.

If the DBCS job attribute indicates the job is capable of handling DBCS data, the dump is taken allowing DBCS data.

If the DBCS job attribute indicates the job is not capable of handling DBCS data, the dump is taken not allowing DBCS data.

- **Display System/36 messages**

This can be done whenever an error message is displayed to a user by a System/36 function in the System/36 environment.

If the DBCS job attribute indicates the job is capable of handling DBCS data, an attempt is made to display a DBCS message. The prefix of a DBCS message ends with a Z (such as, USZ). If the DBCS version cannot be found, the SBCS version is used.

If the DBCS job attribute indicates the job is not capable of handling DBCS data, an attempt is made to display a SBCS message. The prefix of a SBCS message does not end with a Z (such as, USR). If the SBCS message is not found, an error is issued.

- **Output to SYSLIST device**

This is done by one of the System/36 environment utilities that is used to display data such as CATALOG, LISTLIBR, or FORMAT.

If the DBCS job attribute indicates the job is capable of handling DBCS data, the printer or display station is opened to allow DBCS data.

If the DBCS job attribute indicates the job is not capable of handling DBCS data, the printer or display station is opened not to allow DBCS data.

- **Process the MSG command or MSG OCL statement**

If the DBCS job attribute indicates the job is capable of handling DBCS data, shift-out (SO) and shift-in (SI) characters are inserted when a IGC string is continued on another line.

If the DBCS job attribute indicates the job is not capable of handling DBCS data, and if there are shift-out (SO) and shift-in (SI) characters in the message text, characters are replaced with periods (“.”).

System/36 Environment Double-Byte Character Job Attribute

The System/36 environment maintains a job attribute for every job running in the System/36 environment. This job attribute indicates if the job is capable of handling DBCS data.

IGC Procedure

The System/36 environment uses the IGC procedure to set or reset the System/36 environment DBCS job attribute for both interactive and batch jobs. The IGC procedure support is similar to the IGC SESSION prompt support on the System/36 Sign-on display. The following figure shows the format of the IGC procedure:

```
IGC      [ YES ]
         [ NO  ]
         RSLW100-1
```

Note: A parameter must be entered. There is no default value for it.

For more information about the IGC procedure, see the *System/36 Environment Reference* book.

When a DBCS-capable display station starts an interactive System/36 environment job (either from the Sign-on display or the Start System/36 (STRS36) command), the job automatically becomes a System/36 environment DBCS job. Therefore, you do not need to run the IGC procedure to set the System/36 environment DBCS job attribute. If you are using a DBCS-capable display station, but you do not want the job to be a System/36 environment DBCS job, you can use the IGC procedure to change the System/36 environment DBCS job attribute.

If you run the IGC procedure from a display station that is not capable of handling double-byte characters, the procedure does not set the System/36 environment DBCS job attribute. The procedure ignores the request to change the System/36 environment DBCS job attribute because the display station is not capable of handling double-byte characters.

Also, you can set or reset the System/36 environment DBCS job attribute in a batch job using the IGC procedure. When you create a batch job with System/36 environment functions like // EVOKE or // JOBQ, the batch job is the same as the setting of the System/36 DBCS job attribute. For example, if the current job is a System/36 environment DBCS job and // EVOKE is used to create a batch job, the batch job created is a System/36 environment DBCS job.

Because MRT procedures allow only a single // LOAD and // RUN pair, System/36 environment provides a special interface that allows a MRT procedure to set the System/36 environment DBCS job attribute. The interface is a call to a System/36 environment program that sets or resets the System/36 environment DBCS job attribute based on the parameters passed to the program.

Following is an example of a MRT procedure that sets the System/36 environment DBCS job attribute:

```
* MRT Procedure to set on the S/36 environment
*   double-byte character job attribute
CALL PGM(QSSP/QEXIGCP) PARM(*YES)
// LOAD MRTPGM
// RUN
```

The following example of a MRT procedure resets the System/36 environment DBCS job attribute:

```
* MRT Procedure to set off the S/36 environment
*   double-byte character job attribute
CALL PGM(QSSP/QEXIGCP) PARM(*NO)
// LOAD MRTPGM
// RUN
```

Note: The initial setting of the System/36 environment DBCS job attribute is off when a MRT procedure is started.

The System/36 environment DBCS job attribute controls the following System/36 environment functions:

- **// IF DSPLY-IGC procedure control expression**

If the System/36 environment DBCS job attribute is not on, the // IF DSPLY-IGC procedure control expression is false. If the System/36 environment DBCS job attribute is on, the // IF DSPLY-IGC procedure control expression is true. If you use the // IF DSPLY-IGC proce-

duration control expression in a batch job, the system displays an error message.

- **Retrieve DBCS messages**

The System/36 environment supports both single- and double-byte versions of messages for IBM-supplied programs and for user-written System/36 environment applications.

Messages displayed by IBM-supplied programs use the library list to determine whether to send the single- or double-byte version of a message. See “Setting the Library List for DBCS Session” for information on setting the library list to display single- or double-byte versions of messages.

Messages retrieved and sent by user-written System/36 environment applications use the System/36 environment DBCS job attribute. If the job attribute is on, an attempt is made to retrieve a DBCS version of the message. The prefix of a DBCS message ends with a Z (such as, USZ). If the DBCS version cannot be found, the SBCS version is used. If the System/36 environment DBCS job attribute is off, the SBCS version of the message is retrieved for the application.

For more information on creating SBCS and DBCS versions of messages, see the description of the CREATE procedure in the *System/36 Environment Reference* book.

- **SYSLIST interpretation of shift-out (SO) and shift-in (SI)**

If the System/36 environment DBCS job attribute and AS/400 job attribute indicate the job is capable of handling DBCS data, the SO/SI characters are left in the data that is sent to the SYSLIST device.

If either the System/36 environment DBCS job attribute or the AS/400 job attribute indicate the job is not capable of handling DBCS data, the SO/SI characters and the string they surround are replaced with periods (“.”).

- **SYSLOG interpretation of shift-out (SO) and shift-in (SI)**

If the System/36 environment DBCS job attribute and AS/400 job attribute indicate the job is capable of handling DBCS data, the SO/SI characters are left in the data when the message is sent. Also, SO/SI characters are

inserted when a DBCS string is continued on another line.

If either the System/36 environment DBCS job attribute or the AS/400 job attribute indicate the job is not capable of handling DBCS data, the SO/SI characters and the string they surround are replaced with periods (“.”).

The STATUS SESSION (D S) operator control command can be used to display the current setting of the System/36 environment DBCS job attribute.

Notes:

1. Messages sent to the console from a System/36 environment application are not affected by the current setting of the System/36 environment job attribute. The double-byte version of System/36 environment messages are sent to the console message queue (QSYSOPR) if the AS/400 job attribute indicates the job is capable of handling DBCS data and the message member contains a DBCS version of the message.
2. If an interactive job attempts to display DBCS characters on a S/36 Program Message display that is not capable of displaying DBCS data, the system replaces the DBCS characters with periods (“.”).

Setting the Library List for DBCS Session

The DBCS support for IBM-supplied data is similar to support for displaying multiple national languages. See “System/36 Environment Multiple Language Support” on page 20-1 for information about System/36 environment multiple language support. How IBM-supplied data is presented is based on the setting of the job’s library list.

When you install the System/36 environment, the IBM-supplied data in library QSSP is in the primary national language of the system. For example, if the primary national language of the system is a single-byte national language, library QSSP contains the single-byte version of the IBM-supplied data. If the primary national language of the system is a double-byte national language, QSSP contains the double-byte version of the IBM-supplied data.

Secondary national languages are installed in separate national language libraries. For example, Kanji is contained in library QSYS2962.

If you want the System/36 environment IBM-supplied data presented in the primary national language of the system (for example, double-byte Kanji), simply sign on to the System/36 environment. The System/36 environment national language default is the primary national language of the system.

If you want the System/36 environment IBM-supplied data presented in a national language other than the system's primary national language, you must change the library list so the national language library you want is above the primary national language library (QSSP) in the library list. For example, if your system's primary national language is double-byte Traditional Chinese, but you want the Kanji version of IBM-supplied data, type CHGSYSLIBL QSYS2962 to place the Kanji version at the top of the library list.

If the system contains only DBCS-capable display stations and all users want the IBM-supplied data presented in a national language different from the primary national language, you can change the system value QSYSLIBL so the national language library you want is placed at the front of the system portion of the library list.

Use the Display System Value (DSPSYSVAL) command to display the value of QSYSLIBL, and the Change System Value (CHGSYSVAL) command to change the value of QSYSLIBL for all users. For example, if you specify CHGSYSVAL SYSVAL(QSYSLIBL) VALUE('QSYS2962 QSYS QUSRSYS QHLPSYS'), the system uses the Kanji version of the IBM-supplied data for all jobs.

If the system contains a mixture of display stations that can handle DBCS data and some that cannot, and you want the national language library set based on *display station type*, use work station entries and routing entries to set the library list. Use the work station entries to select a routing entry based on the type of work station signing on to the subsystem. You can use the routing entries to select the program called when signing on to the subsystem.

The default program for IBM-supplied subsystems is QCMD. The routing entry should specify a user-written program that adds the national language library you want to the front of the system portion of the library list (using the CHGSYSLIBL command). Then the program should transfer control from the user-written program to the program QCMD (using the Transfer Control (TFRCTL) command).

For information on subsystems, work station entries, and routing entries, see the *Work Management* book. For information on writing CL programs, see the *CL Programming* book. For information on CL commands, see the *CL Reference* book.

If the system contains a mixture of display stations that can handle DBCS data and some that cannot, and you want the national language library set based on *display station type* and *the user running at the display station*, you can use an initial procedure and some System/36 environment OCL (Operation Control Language) functions to set the library list.

The initial procedure defined for the user (initial program attribute in that user's profile) should contain the following statements:

```
* Initial procedure to put library QSYS2962 at the top of
* the system portion of the library list if the current
* display station is DBCS-capable
// IF DSPLY-IGC CHGSYSLIBL QSYS2962 /* If this is a */
* /* DBCS device, */
* /* add QSYS2962 */
* /* to lib list */
```

If the job is a System/36 environment DBCS job, the // IF DSPLY-IGC procedure control expression is true. Therefore, the CHGSYSLIBL QSYS2962 command runs. If the job is not a System/36 environment DBCS job, the // IF DSPLY-IGC procedure control expression is false. Therefore, the CHGSYSLIBL QSYS2962 command does not run.

When you create a batch job with a System/36 environment function like // EVOKE or // JOBQ, the system initializes the library list of the batch job based on the setting of the library list of the job that creates the batch job. For example, if you use // EVOKE to create a batch job, the library list of the batch job is the same as the library list of the job that created the batch job. Therefore, the batch job can display messages in the same

national language as the job that created the batch job.

When you start a MRT job, the library list is set up as defined in the QS36MRT job description in library QGPL. The elements in the following list make up the default setting for this job description:

- The system portion of the library list is set to the libraries contained in system value QSYSLIBL.
- The current library is set to the current library of the job that started the MRT procedure.
- The user portion of the library list is set to the libraries contained in system value QUSRLIBL.

For information on job descriptions, see the *Work Management* book.

System/36 Environment DBCS Printer Support

For every printer defined in the System/36 environment, the system creates a print file with the same name as the System/36 environment printer. If P1 is a System/36 environment printer ID, library #LIBRARY contains a print file named P1. When a System/36 environment application creates printed output, the System/36 environment support uses these print files to define some of the attributes of the spooled file.

If the printer can print double-byte characters and the AS/400 system value indicates the job is capable of handling DBCS data, the system creates the print file in #LIBRARY with the IGCDTA parameter specified as *YES. This allows DBCS data to be printed.

If the printer cannot print double-byte characters or the AS/400 system value indicates the job is not capable of handling DBCS data, the IGCDTA parameter is not specified when the print file is created. Thus, DBCS data cannot be printed and DBCS characters sent to the printer are interpreted as single-byte characters.

If the system created a spooled file with the IGCDTA parameter of *YES and you move this file to a printer that cannot print DBCS data, the system sends a message to the message queue

for the printer informing the operator that the spooled file may not print correctly on the printer. If the operator selects the option to print the spooled file on the non DBCS-capable printer, the system does not send advanced printer functions to the printer.

If you change the printer's device description to be DBCS capable from not being DBCS-capable, the System/36 environment print file associated with this printer does not match the current printer description. You can perform the following steps to update the System/36 environment print files to match the printer device descriptions:

1. Enter the Change System/36 environment (CHGS36) CL command.
2. Type 2 in the *S/36 printer IDs* field of the Change S/36 Environment Configuration display.
3. Press the Enter key when the Change S/36 Printer IDs display appears.
4. When the Change S/36 Environment Configuration display appears again, press the Enter key to update the System/36 environment print files.

The CHGS36 command updates the System/36 environment print files to match the IGC attribute in the printer device descriptions. If you cannot change a System/36 environment print file, an error message appears. The message describes the reason you cannot change the print file and the action you need to take to correct the problem. After you correct the problem, run the CHGS36 command again to update the print files.

Writing Applications for Translating Considerations

Consider the following guidelines if you are writing applications that will be translated into several languages:

- Information the user sees should be separated from the programming statements. Messages a user sees should be in a message member rather than literals in programming language statements.
- Each translation of IBM-supplied data should be in a separate language library.

- Use the library list to select the national language in which the application shows IBM-supplied data.

Appendix A. Access Algorithms for Direct Files

This appendix describes some access algorithms you can use with direct files. With them you can design your direct files more efficiently.

To design and use a direct file define an access algorithm that satisfies the processing requirements for the file while preserving the advantages of direct files.

Assign relative record numbers sequentially. The first record placed in the file has relative record number 1, the second record has relative record number 2, and so on.

Use a control field in each record as its relative record number. For example, use loan number 3456 without change as relative record number 3456. Use a control field directly as a relative record only if a large number of unused values are not within the range of values for the control field. If there are many unused values and record positions, define an algorithm to reduce the size of the file.

Choosing an Access Algorithm

An **access algorithm** is the method you use to determine the position each record will occupy. The algorithm must supply a positive, whole number as a relative record number.

Use a formula as an algorithm to determine the record number. For example, if loan numbers start with 1001, loan number 3456 could be relative record number 2456 (3456 minus 1000). The formula can be as complex as you need to make it. Refer to "Examples of Access Algorithms" on page A-2 for more information and examples.

Another method is to use a control field that contains alphanumeric data. An algorithm would convert the alphanumeric data to a relative record number. Refer to "Handling Synonym Records" for an example of using a customer name as the control field.

The choice of an access algorithm and the decision about whether to use a direct file is usually based on how well synonym records can be handled. A **synonym record** is a record in a

direct file whose control field supplies the same relative record number as another control field. The first record with a relative record number is called the **home record**. If the handling of synonyms requires a significant number of additional disk accesses, you lose one of the important advantages of the direct file. If the algorithm and synonym handling are changed, you may need to rebuild files and change all the programs that use those files, if the access algorithm and the synonym code must reside in each program that uses a direct file.

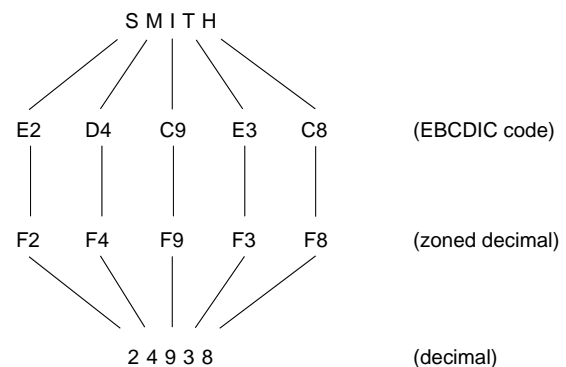
Handling Synonym Records

You can handle synonyms in many ways. Following are two common ways:

- Place synonyms in a separate part of the file.
- Place synonyms in the next available blank location.

The record must contain a pointer to the synonym record. If two or more synonyms exist, the first synonym contains a pointer to the second synonym, and so on.

For example, the control field for a file is the first five characters of the customer's name. The file contains space for 40 000 records and allowance for three synonyms for each home record. The customer's name is converted to a decimal value, as shown in the following figure:



RSLW046-1

The decimal value is then divided by 9999:
 $24938 / 9999 = 2.4940$

Ignoring the whole number of the quotient, you would calculate the location as follows:

$$(4940 \times 4) + 1 = 19761$$

Because many customers can have the same name, the program may have to read records 19761, 19762, 19763, and 19764 to find the correct Smith. If extra synonyms are required, the third synonym could point to the next available space in the file (possibly an unused synonym location for the next home record). Also, to reduce the number of synonyms, you can accept six or more characters from the customer name.

Examples of Access Algorithms

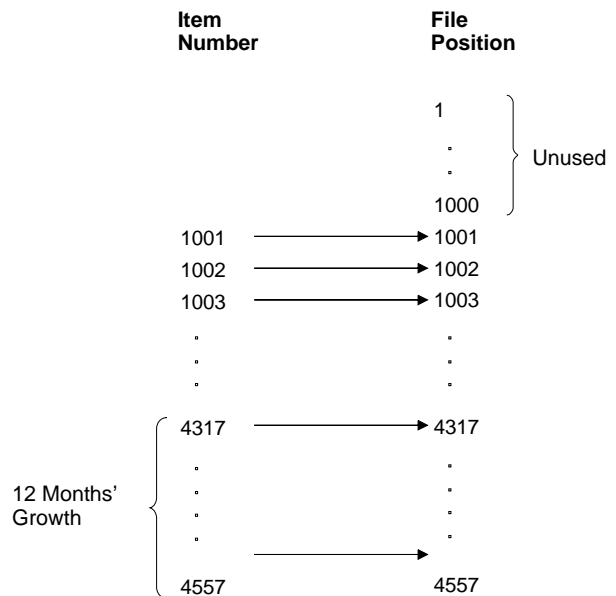
The following example illustrates approaches to designing access algorithms for direct files. In the example, the major goals are to build a file in which:

- The records can be accessed with an average of one disk access
- The disk space used for the file should contain little unused space
- The file should easily accommodate new records

Defining the Algorithm

An indexed item file is to be converted to a direct file for an interactive order entry application. The key field is a five-digit item number; four digits are assigned by the user, and the fifth digit is a check digit. The four digits start with 1001, and the user assigns the next sequential number to new items. Deleted item numbers are not reused until item 9999 has been taken. Approximately 20 new items are added per month, and four items are deleted. The highest number is currently 4317, but the file contains only 2812 items.

The algorithm could state that the direct file position for each record is equal to the four-digit item number. Assume that the new record is a few bytes larger than the old record and that the file also accommodates 12 months of growth before reorganization. The algorithm requires a file containing 4557 record positions. The items are related to direct file positions, as shown in the following figure:

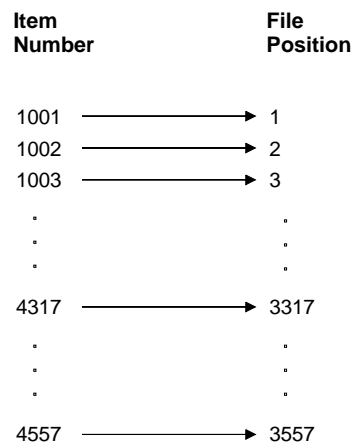


RSLW053-0

The approach shown in the preceding figure supplies no synonyms. It uses only two-thirds of the record positions, and most of the unused space is at the beginning of the file.

You can change the algorithm to state that the direct file position for each record is equal to the four-digit item number minus 1000. The file now requires 3557 positions.

The following figure shows the relationships:



RSLW054-0

The approach shown in the preceding figure also supplies no synonyms, but uses 85% of the record positions. The unused portions are embedded randomly within the file where items have been

deleted. Although each record requires only one disk access, the file size is 15% larger than the data portion of the indexed file it is to replace.

Further change the algorithm to state that the direct file position for each is found by subtracting 1000 from the four-digit item number, multiplying the difference by 0.85, and half-adjusting the result. The file then occupies 3023 positions with the relationships shown in the following figure:

Item Number	→	File Position
1001	→	1
1002	→	2
1003	→	3
.		.
.		.
.		.
4317	→	2819
.		.
.		.
.		.
4557	→	3023

RSLW055-0

The approach shown in the preceding figure uses 99% of the record positions, and the file size is 1% larger than an indexed file. It has, however, introduced the possibility of synonym records. For example, if item 1004 exists, it is assigned to direct file record position number 3 (same as item 1003). Similarly, items 4316 and 4317 conflict, as do items 4556 and 4557. Thus, the refinement of the algorithm to meet the second major goal, minimum file space, may now affect the first goal,

minimum disk accesses, because synonym records take a minimum of two accesses.

Handling Synonyms

Your method for handling synonyms must minimize accesses and file space. Define (program) how a record is placed in an alternative position when its home location is filled.

Further analysis of the item file in this example can offer some suggestions for synonym handling. In this example, a synonym occurs about once in seven records.

The previous algorithm caused the mapping shown in Figure A-1 (asterisks identify synonyms).

In Figure A-1, approximately one in seven item numbers is unused because of deleted items and because the file is only 86% full. There is an unused position in the direct file about as frequently as the synonyms occur.

For example, the method of handling synonyms can state that a synonym record is placed in the next higher numbered position that is unused. Because the file uses only 85% of the range of numbers, 15% of the numbers are not used because they are deleted. However, the deleted numbers are randomly distributed throughout the range of numbers. Thus, some positions are available in the file for synonym records. About every seventh number is a synonym. Assume that of the first 40 item numbers, items 1007, 1008, 1015, 1017, 1020, and 1039 are among those deleted numbers, as shown in Figure A-2 on page A-4.

Item Number	→	File Position	Item Number	→	File Position	Item Number	→	File Position
1001	→	1	1009	→	8	1017	→	14*
1002	→	2	1010	→	9*	1018	→	15
1003	→	3*	1011	→	9*	1019	→	16
1004	→	3*	1012	→	10	1020	→	17
1005	→	4	1013	→	11	1021	→	18
1006	→	5	1014	→	12	1022	→	19
1007	→	6	1015	→	13	1023	→	20*
1008	→	7	1016	→	14*	1024	→	20*

RSLW056-0

Figure A-1. File Position Mapping When Using Synonym Handling

Item Number	File Position	Item Number	File Position	Item Number	File Position
1001	→ 1	1016	→ 14	1030	→ 26
1002	→ 2	1018	→ 15	1031	→ **
1003	→ 3	1019	→ 16	1032	→ 27
1004	→ 6	1021	→ 18	1033	→ 28
1005	→ 4	1022	→ 19	1034	→ 29
1006	→ 5	1023	→ 20	1035	→ 30
1009	→ 8	1024	→ 33	1036	→ 31
1010	→ 9	1025	→ 21	1037	→ **
1011	→ 13	1026	→ 22	1038	→ 32
1012	→ 10	1027	→ 23	1040	→ 34
1013	→ 11	1028	→ 24		
1014	→ 12	1029	→ 25		

RSLW057-0

Figure A-2. Items Deleted When Using Synonym Handling

Consider the following:

- Item 1031 is placed after position 34.
- Item 1037 occupies a higher-numbered position than item 1031.
- File positions 7 and 17 are unused.
- After accessing a record, the program has to verify that the record is the one requested. If it is not, the program must access a synonym.
- No more than two items have the same relative record number. Thus, most records require no more than two disk accesses.

Note: In this example, records are loaded into locations before synonym records are loaded in a second run, and few records are added. If records are added after the synonyms are loaded, the locations for the added records can be occupied by synonyms. Thus, the added record becomes a **pseudosynonym**. If many records are added, most have to be handled as synonyms. In this situation, the technique described here can be less useful because performance is degraded as records are added.

In this synonym-handling technique, the average synonym should be close to the first position searched. Thus, a second access is necessary approximately 15% of the time, and this access should find the record not too distant from the original location.

The file should be loaded, and the synonyms added in a second run. As the synonyms are added in the next available higher-numbered position, a synonym pointer in the record has to be updated to point to the synonym record position.

Indexed File with Keys

In this example, a customer master file contains three types of records (A, B, and C) for three types of customers. These records are in an indexed file with keys. Type A records have customer numbers from 10000 to 49999. Type B records are numbered from 60000 to 79999. Type C records are numbered from 90000 to 99999. Each type of record is arranged alphabetically by customer name.

The file was first loaded with approximately 500 alphabetized type C records, followed by 1000 alphabetized type B records, and finally by about 3000 alphabetized type A records.

Records were added at the end of the file as follows:

- The added record type is determined (A, B, or C).
- The added record is assigned an unused customer number that corresponds to the alphabetic sequence of the customer name according to a printout of the file.

The following figure shows the contents of the file when first loaded:

Record Number	Customer Number	
0001	90000	} Type C (alphabetical by customer name)
0002	90020	
0003	90040	
.	.	
.	.	
.	.	
0467	60020	} Type B (alphabetical by customer name)
0468	60040	
0479	60060	
.	.	
.	.	
.	.	
1592	10000	} Type A (alphabetical by customer name)
1593	10013	
1594	10026	
1595	10039	

RSLW058-0

The file originally contained 4725 records. Space was allowed for 6000 records. Now, 18 months later, the file contains 5638 records.

An analysis of the file indicates the following:

- The file expands at the rate of about 12% per year and should probably be planned for about 6600 records to meet one year's requirements.
- Eight percent of customer numbers 10000 through 50000 are used, and 5% of the other numbers are used.
- You should keep synonym records as close as possible to the expected location.
- The best file design solution is to use more than one file and more than one type of file organization.
- If all the customer numbers are in one file, an algorithm must take into account the necessity of loading type C customers at the front of the file, followed by types B and A.
- The ratio of A to B to C types is about 6 to 2 to 1.

A trial algorithm may try to accomplish the mapping shown in the following table:

Customer Number	Type	File Record Number
90000 through 99999	C	0001 through 0733 (1/9 x 6600 = 733)
60000 through 79999	B	0734 through 2200 (2/9 x 6600 = 1467)
10000 through 49999	A	2201 through 6600 (6/9 x 6600 = 4400)

To accomplish the mapping, the algorithm must:

- Convert customer numbers 90000 through 99999 into a set of relative record numbers from 1 through 733
- Convert customer numbers 60000 through 79999 into a set of relative record numbers from 734 through 2200
- Convert customer numbers 10000 through 49999 into a set of relative record numbers from 2201 through 6600

Following is a method for doing these conversions:

- If the customer number is greater than 89999, subtract 89999 from it, multiply the difference by 0.0733 (the ratio of 733 positions to 10000 numbers), and use the half-adjusted product as the record position.
- If the customer number is less than 50000, subtract 9999 from it, multiply the difference by 0.11 (the ratio of 4400 record positions to 40000 record numbers), add the half-adjusted product to 2200, and use the sum as the record position.
- For all other customer names (60000 to 79999), subtract 59999 from the number, multiply the difference by 0.0733 (the ratio of 1467 record positions to 20 000 numbers), add the half-adjusted product to 733, and use the sum as the record position.

The synonym-handling technique can be the same as in "Defining the Algorithm" on page A-2. You should test the synonym-handling technique by loading the file. Then another program that attempts to retrieve all records, and counts the number of necessary accesses, can measure the technique's effectiveness. The results of the second program indicate whether changes are necessary or desirable. To further test the synonym-handling technique, run a sample program in an interactive environment to see

whether response time at the display stations is acceptable.

Randomizing Techniques

This example uses randomizing techniques. These techniques use part of the data to determine the record position. Regardless of which randomizing technique you use, describe the concept and approach in each program that uses the technique.

Some master files have different uses and use different techniques. For example, a rate file in a telephone revenue accounting application has one record for every *from-to* location in the United States. A call made from number (123) 555-1234 to (456) 555-4567 requires the retrieval of a rate record from the master file with a key of 123555456555.

To convert such a number to relative record position on a direct file, develop an algorithm that multiplies the numbers 123555 and 456555 and uses the second, fourth, sixth, eighth, and tenth digits of the product as the relative record position. This technique can produce a random distribution across a file for approximately 100,000 records.

Another approach is to use an algorithm that takes the second, fourth, sixth, eighth, and tenth digits from the 12-digit key. Thus, the first algorithm can locate the rate record in relative position 20632 ($123555 \times 456555 = 22109653025$); the second algorithm might place the same record in position 25555.

Some records for a specific billing location are more active than the majority of the records. You can put these very active records into a separate file which may or may not be direct.

Appendix B. \$SFGR Specification Forms

The following sections tell you how to complete each column of the S-, H-, and D-specifications forms.

Display Control (S) Specifications

The first record coded for each display format is the display control (S) specification. The S-specification supplies information about the entire display format, not about individual fields. One display control specification is required and must be the first record in the specifications for each display format.

Use the form shown in Figure B-1 on page B-2 to code the S-specification.

The following sections describe the entries on the S-specification section of the Display Format Specifications form:

Note: If the entries for a field are shorter than the length of the form, you must left-justify them.

Sequence Number (Columns 1 through 5): Columns 1 through 5 do not require an entry. These columns contain record sequence information to number the records in a display format. The FORMAT procedure does not process record sequence information.

A special sequence number value of *DDS when column 7 contains an asterisk (*) indicates that the record contains user-supplied data definition statements (DDS), which is inserted at this position in the DDS generated by the system format generator utility (\$SFGR). In the generated DDS, the asterisk in column 7 is replaced by a blank and the character in column 6 is replaced by an A. All of the remaining text remains unchanged, including the *DDS in the sequence number. Only the first 80 characters of text are used.

When a *DDS comment is encountered, warning message SYX5328 is issued in the SFGR compile listing.

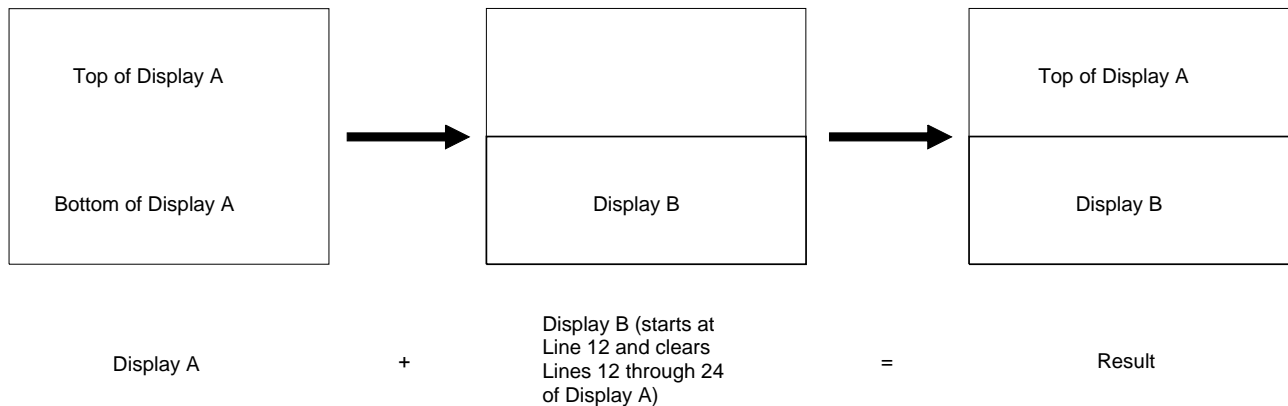
Warning: System/36 treats warning message SYX5328 as a comment and the DDS is ignored, but this loss of function could affect the display or application if ported back to System/36. The function specified by the *DDS statement might not be supported on a previous release of the OS/400 program. If the SFGR or DDS source is ported to an AS/400 system where an earlier release is installed, the display might not compile successfully. Avoid using Screen Design Aid (SDA) to modify this source member. SDA may move the *DDS comment to another location, causing a difference in function or causing the DDS compile to fail. The *DDS comment is not recommended for general use. It is recommended for inserting the FRCDTA keyword into the generated DDS at column 45, when placed immediately after an S-specification defining a format which must be immediately written to the display, even though the display file is created with the DFRWRT(*YES) attribute. (The FRCDTA keyword is supported in OS/400 Release 1.0.) See “DFRWRT Attribute” on page 14-26 for details related to DFRWRT. See the CRTS36DSPF and CRTDSPF commands and the *DDS Reference* book.

Because other types of DDS may not be compatible for use by System/36 environment applications, results cannot be predicted for all other uses of this function. If the resulting DDS fails to compile, you must remove or correct all *DDS comments in the SFGR source.

Specification Type (Column 6): The character S identifies this record as the display control specification. An entry in column 6 is required and is preprinted on the S-specification form.

Display Format Name (Columns 7 through 14): Enter a display format name in columns 7 through 14 that meets the following requirements:

- The name cannot include commas, single quotation marks ('), or embedded blanks.
- The name can have a maximum of eight characters with the first an alphabetic character (A through Z), or a special character including #, \$, or @.



RSLW097-0

Figure B-2. Use of Start Line Number and the Lines to Clear Entries

in column 29 of the D-specification for any input field in the display format, the operator must type data in that field. An error message appears if the operator presses a function or command key or the Enter key and does not type data in the mandatory entry field. The operator must press the Error Reset key and type data in the mandatory entry field.

- N** The contents of all input fields return to the application program only if the operator types in data. If no data is entered in any input field, RPG II program record input specification indicators dependent on display constants do not turn on because the input fields do not return to the application program.

Note: You can improve response time for display formats shown at a remote display station if you specify N (No) for return input.

Sound Alarm (Columns 25 and 26):

Select one of the following entries for columns 25 and 26:

N (or blank)

If you specify N (No) or leave these columns blank, the alarm does not sound when the operator looks at this display format.

- Y** If you specify Y (Yes), the alarm sounds when the operator looks at this display format.

01 through 99

The alarm sounds if the specified indicator is on when this format is displayed.

Enable Function Keys (Column 27):

Use column 27 to enable function keys. When a function key is enabled, the key performs a special function and must be handled by the application program. When a function key is disabled, the key either has no special function in the program or is handled by the system. Specify one of the following values in column 27:

blank

All function keys are enabled. All numbers listed in the key mask are ignored.

- Y** The function keys identified by numbers listed in the key mask are enabled, while all others not listed are disabled. If the key mask contains no numbers, all the function keys are disabled.

- N** The function keys identified by numbers listed in the key mask are disabled, while all others not listed are enabled. If the key mask contains no numbers, all the function keys are enabled.

R (retain)

The function keys previously enabled remain enabled, and those previously disabled remain disabled. All numbers in the key mask are ignored.

Notes:

1. An error message appears if the operator presses an inactive function key. The operator can then press the Error Reset key, followed by the correct function key. You identify the function keys to be turned on or off by numbers in the key mask (columns 64 through 79).

2. System/36 environment compilers may disable some function keys at run time. For more details, see the *System/36-Compatible COBOL User's Guide and Reference* and the *System/36-Compatible RPG II User's Guide and Reference*.

Enable Command Keys (Column 28):

Use column 28 to enable command keys. When a command key is enabled, the key is allowed and must be handled by the application program. When a command key is disabled, the key is not allowed. Specify one of the following values in column 28:

blank

All command keys are enabled. All alphabetic characters listed in the key mask are ignored.

- Y** The command keys identified by alphabetic characters listed in the key mask are enabled, while all others not listed are disabled. If the key mask contains no alphabetic characters, all the command keys are disabled.

- N** The command keys identified by alphabetic characters listed in the key mask are disabled, while all others not listed are enabled. If the key mask contains no alphabetic characters, all the command keys are enabled.

R (retain)

The command keys previously enabled remain enabled, and those disabled remain disabled. All alphabetic characters in the key mask are ignored.

Note: An error message appears if the operator presses an inactive command key. The operator can then press the Error Reset key, followed by the correct command key. You identify the command keys to be turned on or off by alphabetic characters in the key mask (columns 64 through 79).

Blink Cursor (Columns 29 and 30):

Select one of the following entries for columns 29 and 30:

N (or blank)

The cursor does not blink when this display format appears.

- Y** The cursor blinks when this display format appears.

01 through 99

The cursor blinks if the specified indicator is on when this display format appears.

Erase Input Fields (Columns 31 and 32):

Select one of the following entries for columns 31 and 32:

N (or blank)

Data is not erased from the input and input/output fields when the display format appears.

- Y** The input and input/output fields are erased each time the display format appears. You should usually not specify Y because the application program ignores all entries on the D-specifications when this display format appears. If you want to erase the input fields, specify an indicator in column 31, and turn that indicator on when this display format appears.

01 through 99

The application program erases the data from the input and input/output fields and resets the keyboard if the specified indicator is on when the display format appears.

Note: If you specify an override operation in columns 33 and 34 of the S-specification, the display format is sent to the display station. Both the erase input and override operations are processed if you specify both operations.

Use the erase-input-fields entry when you use one display format for repeated input operations. The indicator should be off the first time the display format appears. Each time the display format appears and the indicator is on, the input fields are blank and the operator can type data.

A display station error occurs if you request that the system erase input fields when there are no input fields currently on the display format. Use the erase-input-fields entry only after a display format with input or input/output fields has appeared without erase input.

Considerations for the Erase-Input-Fields

Entry: The following operations are affected after you do an erase-input-fields operation:

Return input

Data contained in all input fields returns to the application program if the operator types data

in an input field. If the operator does not type data in any input field, input data does not return to the application program.

Mandatory entry

The system does not require the operator to type data in mandatory entry fields if the operator does not type data in any input field.

The following occur after you do another erase-input-fields operation:

- If the operator has pressed the Enter key without typing in data, data contained in the input fields is *not* erased, including mandatory entry input fields.
- If the operator has typed data in *any* input field, then only those input fields in which the operator has typed in data are erased. If the operator types data in any of the input fields, the operator must type data in all mandatory entry input fields; data in the mandatory entry fields is also erased.

The first erase-input-fields operation erases data from all input fields. The second erase-input-fields operation erases data from only those input fields in which data has been typed since the first erase-input-fields operation, if the following occur:

1. A display format with return input specified appears.
2. An erase-input-fields operation is done.
3. Another erase-input-fields operation is done.

Override Fields (Columns 33 and 34):

An override operation allows you to change (override) fields in a display format without showing the same display format again. If the application program determines the operator typed incorrect data into a field, turn this indicator on to override the entry and display the same display format again. If the indicator specified in the output data entry for a field in columns 23 and 24 of the D-specification is off, the incorrect data does not change. Once the operator corrects the error, turn the indicators off and display the display format again using an override operation to remove any highlighted fields or error messages.

Field attributes may be changed if the D-specification contains one or more of the following:

- An output indicator (columns 23 and 24)
- A high intensity field attribute (columns 39 and 40)
- A blink field attribute (columns 41 and 42)
- A nondisplay field attribute (columns 43 and 44)
- A reverse image field attribute (columns 45 and 46)
- An underline field attribute (columns 47 and 48)

Note: If an indicator specified in the protect-field entry (columns 37 and 38 of the D-specification) is on, that indicator is ignored during the override operation.

Select one of the following entries for columns 33 and 34:

N (or blank)

The application program does not do an override operation for this display format. The system shows a normal output operation.

01 through 99

An override operation does not occur if this indicator is off when the display format appears.

The following occur during an override operation when the indicator in columns 33 and 34 is on:

- The data does not change in a field for which you specified an indicator for the output data entry in columns 23 and 24 of the D-specification, and that indicator is off. Data the operator typed is unchanged. Any field with a Y (Yes), an N (No), or a blank specified for the output data entry does not change.
- The program displays the field with data supplied by the application program if the indicator for the output data entry is on.

Y If Y is specified in column 33, an override operation is performed every time this format is displayed. Generally, Y should not be specified. If an override operation must be performed for this format, an indicator should be used.

Figure B-3 on page B-7 summarizes the effects of indicators on output data during an override operation.

Considerations for the Override Fields Entry:

The following operation is affected after you do an override fields operation:

Return Input

If you specify N (No) for return input when the override fields indicator is on, the contents of the input field return to the application program even if the operator does not enter any data.

Suppress Input (Columns 35 and 36):

Specify suppress input from the keyboard when several display formats appear before the application program needs input. Specify suppress input on all but the last display format, even if the display formats do not contain input fields, when you send multiple display formats. If you show a display format over a portion of another display format, data can be typed and read only from the input fields defined by the last display format.

Select one of the following entries for columns 35 and 36:

N (or blank)

The input fields are read and returned to the application program when the operator presses the Enter key.

- Y** The keyboard locks and the input fields are not read or returned to the application program until the following occur:
- A display format for which you specified N (No) appears, or for which the specified indicator is off appears.
 - The operator presses the Enter key at the display format.

01 through 99

If the specified indicator is on, the keyboard locks and the input fields are not read or returned to the application program until the following occur:

- A display format for which you specified N (No) appears, or for which the specified indicator is off appears.
- The operator presses Enter at the display format.

Note: If you specify a Y (Yes) on the S-specification for a help format, it is processed as if you had specified N (No) when the display

format appears. See *ADTS/400: Screen Design Aid for the System/36 Environment* book for more information. Always specify N (or blank) for any format used by a MRT or RUF. For menus not displayed by a MRT, always specify Y. See "DFRWRT Attribute" on page 14-26 for related information.

Null Fill (Columns 37 and 38): If you specify null fill for a display format, any remaining character positions in the input or input/output fields on the display fill with null characters.

Columns 37 and 38 are syntax-checked, but have no effect on the creation of the display on the AS/400 system. On the AS/400 system, all character fields have their trailing blanks replaced by null characters on the display. This approach allows easier use of the Insert key to enter data.

Null characters (unfilled character positions) in input fields become blanks when the application program reads the fields.

Select one of the following entries for columns 37 and 38:

N (or blank)

Blanks do not change to null characters when the display format appears.

- Y** The following occur when the display format appears:
- All blanks in the first constant input field change to null characters.
 - All blanks in application-program-supplied data sent to input fields change to null characters.

01 through 99

The following occur if you specify an indicator:

- All blanks in the first constant input field change to null characters.
- If the specified indicator is on when the display format appears, any blanks in program-supplied data sent to input fields change to null characters.
- If the specified indicator is off when the display format appears, blanks in program-supplied data sent to input fields do not change to null characters.

		Indicator Specified in the Override Fields Entry	
		OFF	ON
Indicator Specified in the Output Data Entry for the Field	OFF	Output data is constant data specified for the field.	No change occurs to output data on the display.
	ON	Output data comes from the program.	Output data comes from the program.
		Normal Output Operation	Override Operation

RSLW099-0

Figure B-3. Effects of Indicators during an Override Operation

132-Column Format (Column 39):

Select one of the following entries for column 39 to specify 132-column format :

N (or blank)

The display format is 80 columns.

Y The display format is 132 columns and can appear only on display stations which support 132 columns.

Note: Mixing both 80-column display formats and 132-column display formats in the same display file can result in poor performance.

Right-to-Left Display (Column 40): You must use source entry utility (SEU) to specify right-to-left display.

The entry you make in column 40 of the S-specification controls cursor movement from input field to input field for the entire display. Use column 27 of the D-specification to control the cursor movement in individual fields on the display. If necessary, the operator can use the Reverse key to reverse the direction of the cursor in an input field.

Select one of the following entries for column 40:

N (or blank)

The cursor moves in a left-to-right direction from input field to input field. The application program processes input fields in a left-to-right sequence.

Y The cursor moves in a right-to-left direction from input field to input field. The application program processes input fields in a right-to-left sequence.

Notes:

1. The right-to-left option is ignored by display stations that do not handle this feature.
2. The first format in the source member controls the right-to-left option for the entire display file, including all other formats. If the right-to-left option is specified in formats after the first format, they are ignored on the AS/400 system.
3. The right-to-left option is used only during a \$SFGR CREATE operation, and is ignored during a \$SFGR ADD or UPDATE operation.

Reserved (Columns 41 through 63):

The FORMAT procedure does not use columns 41 through 63. Leave them blank.

Key Mask (Columns 64 through 79):

The key mask is a string of numbers or letters that identify the function and command keys to be turned on (allowed) or turned off (not allowed) when the display format appears. You can specify the function and command key numbers and letters in any order. The key mask cannot contain any embedded blanks.

Function Keys: The function keys are identified in the key mask by a number, as shown in the following table:

Function Key	Key Mask Entry
Print	1 (see "Print Key Exception" below)
Page Down (Roll Up)	2
Page Up (Roll Down)	3
Clear	4
Help	5 (see "Help Key Exception" below)
Home (when the cursor is in the home position)	6

Note: When you specify Y (Yes) for the enable function keys entry (column 27), the numbers in the key mask identify function keys to be allowed. If you specify Y (Yes) for the enable function keys entry and the key mask does not contain any numbers, all function keys are disallowed.

When you specify N (No) for the enable function keys entry (column 27), the numbers in the key mask identify function keys to be disallowed. If you specify N (No) for the enable function keys entry and the key mask does not contain any numbers, all function keys are allowed.

Print Key Exception: When a display format enables the Print key, the operator can use it like any of the other function keys to control program operations. When the Print key is disabled, using it causes the contents of the display to print. For further information about the Print key, see "Using the Print Key" on page 14-25. For more information about assigning the Print key, see the *System/36 Environment Reference* book.

Help Key Exception: When a display format enables the Help key, the operator can use it just like any of the other function keys to control program operations. When the Help key is disabled and H-specifications are not defined for this format, or the display station is showing an error message, the Help key functions normally (for example, it displays help information for a keyboard error). *Do not* enable the Help key if you want help formats to display.

Command Keys: Command keys are identified in the key mask by alphabetic characters, as shown in the following table:

Alphabetic Character	Command Key	Alphabetic Character	Command Key
A	1	M	13
B	2	N	14
C	3	P	15
D	4	Q	16
E	5	R	17
F	6	S	18
G	7	T	19
H	8	U	20
I	9	V	21
J	10	W	22
K	11	X	23
L	12	Y	24

Note: When you specify Y (Yes) for the enable command keys entry (column 28), the letters in the key mask identify command keys that are allowed. If the key mask does not contain any letters, all command keys are disallowed.

When you specify N (No) for the enable command keys entry (column 28), the letters in the key mask identify command keys that are disallowed. If you specify N (No) for the enable command keys entry and the key mask does not contain any letters, all command keys are allowed. For example, to enable the Page Up, Page Down, and Home keys and all the command keys except 1 and 15, you would specify the following options:

- In column 27, Y for enable function keys
- In column 28, N for enable function keys
- 236AP in columns 64 through 68 for the key mask, which identify the following keys:
 - 2 This identifies the Page Up key.
 - 3 This identifies the Page Down key.
 - 6 This identifies the Home key.
 - A This identifies command key 1.
 - P This identifies command key 15.

Help Definition (H) Specifications

The help definition (H) specifications follow the S-specification and precede the first D-specification. They are optional. You can add the H-specifications to the display formats without any changes to your application.

Each H-specification specifies an area on the display format for which you have defined online help information or have supplied input to online information. A display containing help information

appears when the operator presses the Help key while the cursor is in a help area. A help document appears when the operator presses the Help key while the cursor is in an area for which you specified an online document.

For information about help areas and how they are used, see the *ADTS/400: Screen Design Aid* book.

Figure B-4 on page B-10 shows the portion of the Display Format Specifications form used for coding the help definition specifications.

The following sections describe the entries in the H-specification section of the Display Format Specifications form:

Sequence Number (Columns 1 through 5):

Columns 1 through 5 do not require an entry. These columns contain record sequence information to number the records in a display format. The FORMAT procedure does not process record sequence information. A special sequence number value of *DDS when column 7 contains an asterisk (*) indicates that the record contains user-supplied DDS, which is inserted at this position in the DDS generated by \$SFGR. See "Sequence Number (Columns 1 through 5)" on page B-1 for a more complete description and warnings.

Specification Type (Column 6): The character H identifies this record as a help definition specification. This entry is required and is preprinted on the H-specification form.

Help Format Name (Columns 7 through 14):

The help format name specifies the name of the first format displayed if the operator presses the Help key while the cursor is in the help area defined in columns 34 through 42.

Enter a help format name in columns 7 through 14 exactly 8 characters long, in the form **axxxxnn**, where:

- a** An alphabetic character (one of the characters A through Z).

xxxxx Five alphanumeric characters. Name the help formats with an abbreviation of the application or information they describe. Avoid using special characters. If special characters are included and an external display file name is specified in columns 16 through 23, the H-specification is ignored.

nn Two numeric digits in the range 00 through 99.

You can define up to 100 help formats for one help area. All help formats for a help area must be in the display file specified in columns 16 through 23 of the H-specification. These help formats must have the same first 6 characters as specified in columns 7 through 12. The last 2 characters must be different and must vary in the range 00 through 99. The system uses the last 2 digits to select the sequence in which the help formats appear.

Note: See "Format Names" on page 14-24 for additional considerations.

Help Text Label (Columns 7 through 14):

Use the help text label if you specify online document in column 53 of the H-specification. The help text label identifies the location in the help document that appears when the operator presses the Help key with the cursor in this help area. This label represents the label typed on a help text instruction in the help document.

Enter a help text label that meets the following requirements:

- Must be from 1 to 8 characters long.
- Must begin with an alphabetic character (A through Z), #, \$, or @.
- Must not contain a comma (,), single quotation mark ('), or blank. Avoid using special characters.

If the help text label is fewer than 8 characters long, you must left-justify it in this field (it must start in column 7). An asterisk (*) in column 7 identifies this record as a comment statement.

Reserved (Columns 15, 24, 33, 38, 43, 46, 49, 52, 54–80): The FORMAT procedure does not use these columns. Leave them blank.

specified document name in this folder when the operator presses the Help key.

Type a folder name that meets the following requirements:

- Must be from 1 to 8 characters long.
- Must not contain a question mark (?), single quotation mark ('), slash (/), period (.), hyphen (-), equal sign (=), greater than sign (>), comma (,), asterisk (*), or blank. Procedures and the operation control language (OCL) use these characters as delimiters.
- Cannot be ALL, #LIBRARY, F1, READER, PRINT, or DISK.

You must left-justify the folder name (starting in column 25) in this field if it is fewer than 8 characters long.

Upper Left Row and Column (Columns 34 through 37):

Use this field to specify the location, in the form rrcc, of the upper left corner of the rectangular help area. The row number, rr, must be a number from 1 through 24 for an 80-column display format, or 1 through 27 for a 132-column display format. Leading zeros are not required, but you must right-justify the value in columns 34 and 35. The column number, cc, is a 2-digit number (if below 100), or alphanumeric (100 through 132) character that represents columns from 1 through 80 for an 80-column display and from 1 through 132 for a 132-column display.

Columns 100 through 132 are represented by alphanumerics, as shown in the following table:

Figure B-5. Alphanumeric Row and Column Numbers for 100 through 132

100=A0	110=B0	120=C0	130=D0
101=A1	111=B1	121=C1	131=D1
102=A2	112=B2	122=C2	132=D2
103=A3	113=B3	123=C3	
104=A4	114=B4	124=C4	
105=A5	115=B5	125=C5	
106=A6	116=B6	126=C6	
107=A7	117=B7	127=C7	
108=A8	118=B8	128=C8	
109=A9	119=B9	129=C9	

Note: Values beyond D2 are not permitted.

For example, if the upper left corner of the help area is located at row 5, column 10, you would type:

```
| 5 | 1 | 0 |
```

If you specify a start line number or V (variable start line) in columns 17 and 18 of the S-specification of this display format, the row numbers in columns 34 and 35 of the H-specification are adjusted using the following equation:

$$\begin{array}{r}
 \text{Start line number specified in columns 17 and} \\
 \text{18 of display control specification} \\
 + \\
 \text{Row number specified in columns 34 and 35} \\
 \text{of help definition specification} \\
 - \\
 1 \\
 = \\
 \text{Actual row number}
 \end{array}$$

For example, if you specify an upper left row number of 5 in columns 34 and 35 of the H-specification, and a start line number of 10 in columns 17 and 18 of the S-specification, the actual row number would be 14, as shown by the following calculation:

$$10 + 5 - 1 = 14$$

If the resulting upper left row number is greater than the number of lines on the display (24 for an 80-column display station or 27 for a 132-column display station), one of the following occurs:

- If you specify a numeric value for the start line number, the system reports a terminal error when you compile the display format.
- If you specify V for the start line number and the resulting row number is greater than 24 for an 80-column display station or 27 for a 132-column display station, the application program ignores this H-specification when the display format appears.

If you do not specify the upper left row and column numbers, you must also leave the lower right row and column numbers entry blank in columns 39 through 42 of the H-specification.

You can define a null help area by omitting all row and column entries. An operator cannot see the help formats of a null help area by pressing the

Help key. The operator can use help formats of a null help area by using Page Down and Page Up.

Lower Right Row and Column

(Columns 39 through 42): You can specify the location, in the form rrcc, of the lower right corner of the rectangular help area. The row number, rr, must be a decimal number from 1 through 24 for an 80-column display, or from 1 through 27 for a 132-column display. Leading zeros are not required, but the value must be right-justified in columns 39 and 40. The column number is a 2-digit number, which represents numbers from 1 through 80 for an 80-column display and from 1 through 132 for a 132-column display. See Figure B-5 for information on how to represent columns 100 through 132. Leading zeros are not required, but the value must be right-justified in columns 41 and 42.

The system adjusts the row number in columns 39 and 40 of the H-specification if you specify a start line number or V (variable start line) in columns 17 and 18 of the S-specification of this display format. See "Upper Left Row and Column (Columns 34 through 37)" on page B-11 for additional explanation.

If you do not specify lower right row and column numbers, you must also leave the upper left row and column numbers entry in columns 34 through 37 of the H-specification blank.

Suppress Selection Indicator (Columns 44 and 45): Select one of the following entries for columns 44 and 45:

blank

The system uses this H-specification to create a help area when the display format appears. The application program displays the help formats for this help area when the operator presses the Help key.

01 through 99

If the specified indicator is on when the display format appears, the system ignores this H-specification and does not create the help area it defines. The application program does not display the help formats for the help area when the operator presses the Help key.

If the specified indicator is off when the display format appears, the system uses this H-specification to create a help area.

Do not specify Y or N in columns 44 or 45. If you do, an error message appears when the FORMAT procedure processes this display format.

Restore Display Format (Columns 47 and 48): Select one of the following entries for columns 47 and 48:

Y (or blank)

When the operator presses an active function key to leave a help format, the display format, and any data typed in, appears.

N If the operator stops looking at a help format by pressing an active function key, the display format and any input data do not appear. The system sends the function key back to the application program, along with any data the operator typed in the help format before pressing the function key. Data the operator typed into the display format before pressing the Help key is lost. The application program must clear the display to remove the help format and must display the display format again.

01 through 99

If the specified indicator is on when the display format appears, and the operator presses an active function key to stop looking at a help format, the system restores the display format and any input data the operator typed in. The system sends the function key back to the application program, along with any input data the operator typed in the display format before pressing the Help key.

If the specified indicator is off when the display format appears and the operator presses an active function key to stop seeing a help format, the system does not restore the display format or any input data. The system sends the function key back to the application program, along with any input data the operator typed in the help format before pressing the function key. Input data the operator typed in the display format before pressing the Help key is lost. The application program must clear the display to remove the help format and must display the display format again.

Notes:

1. This function is not supported on the AS/400 system. It is syntax-checked and ignored by the \$SFGR compiler.
2. This field must be blank on an online document H-specification. The FORMAT procedure ignores any entry in this field and produces a \$SFGR warning error.

Boundary Indicator (Columns 50 and 51):

The operator uses Page Down and Page Up to look at all the help formats for a help area. To restrict the amount and type of online help information the operator can see, use the boundary indicator to specify that the H-specification for a help area is the end point for using Page Up and Page Down. Select one of the following entries for columns 50 and 51:

N (or blank)

This H-specification does not act as a boundary for using Page Up and Page Down.

Y Use this H-specification as the boundary for using Page Up and Page Down.

01 through 99

If the specified indicator is on, the application program uses this H-specification as a boundary for using Page Up and Page Down when it shows the display format. If the specified indicator is off, the application program does not use this H-specification as a boundary for using Page Up and Page Down when the display format appears.

Note: This field must be blank on a document H-specification. The FORMAT procedure ignores any entry in this field and produces a \$SFGR warning error.

Online Document (Column 53): Use the online document field to indicate that this is an online document H-specification. If you specify Y (Yes) in column 53, the system processes this H-specification as a document H-specification. When the operator presses the Help key with the cursor in this help area, the application program opens the document described by the document name field (columns 16 through 23) and folder name field (columns 25 through 32). The application program shows the document starting with the location identified by the help information label field (columns 7 through 14).

If you omit this field (column 53 is blank) or N [No] is specified, the system processes this H-specification as a display format H-specification.

See the *ADTS/400: Screen Design Aid* book and the *Using OfficeVision/400 Word Processing* book for additional information.

Field Definition (D) Specifications

The records that follow the S-specification and the optional H-specifications are field definition (D) specifications. Each D-specification completely describes a single field on the display. You must use a D-specification to specify each field on the display.

Code the D-specifications in a left-to-right and top-to-bottom pattern from the fields on the layout sheet. For each field on the display, the D-specification can identify the following:

- Name of the field
- Length of the data field or constant
- Starting location (line and column number) of the field
- Field type (input or output field, or both)
- Cursor position when the display first appears
- Any field attributes such as reverse image or high intensity
- Data requirements of the field
- Type of data the operator can type in the field
- What data appears in the field

Use the field definition specification portion of the Display Format Specifications form shown in Figure B-6 on page B-14 to code the D-specifications.

The following sections describe the entries on a D-specification section of the Display Format Specification form:

Sequence Number (Columns 1 through 5):

Columns 1 through 5 do not require an entry. These columns contain record sequence information to number the records in a display format. The FORMAT procedure does not process record sequence information.

A special sequence number value of *DDS when column 7 contains an asterisk (*) indicates that the record contains user supplied DDS, which is

Horizontal Position (Columns 21 and 22): The horizontal position entry specifies the column number of the farthest left position in a field.

The column number is a 2-digit number, representing numbers from 1 through 80 for an 80-column display and from 1 through 132 for a 132-column display. For display formats to appear on an 80-column display, the row number can be from 1 through 24 and the column number from 1 through 80. For display formats to appear on a 132-column display, the row number can be from 1 through 27 and the column number from 1 through 132.

Column numbers 100 through 132 are represented by alphanumerics, as shown in the following table:

100=A0	110=B0	120=C0	130=D0
101=A1	111=B1	121=C1	131=D1
102=A2	112=B2	122=C2	132=D2
103=A3	113=B3	123=C3	
104=A4	114=B4	124=C4	
105=A5	115=B5	125=C5	
106=A6	116=B6	126=C6	
107=A7	117=B7	127=C7	
108=A8	118=B8	128=C8	
109=A9	119=B9	129=C9	

You must right-justify the horizontal position entry. A leading zero is not required.

The system reserves and uses the first column of the display (row 1, column 1) for control information. A field cannot begin in row 1, column 1 (and also row 1, column 2 for SDA). Each field is preceded on the left by one nondisplay control character that defines the attributes of the field. The character supplied by the system does not appear on the display. You must allow at least one space between fields to leave room for this control character. If a field begins in column 1 of a row, the control character is in column 80 of the previous

row for an 80-column display format and in column 132 for a 132-column display format.

The horizontal column must be an even number if both of the following conditions exist:

- You are using the DBCS version of the OS/400 operating system.
- The horizontal column plus the field length (columns 15 through 18 on the D-specification) is greater than 81 and the data type is E, F, or O.

Output Data (Columns 23 and 24): The output data entry allows you to make this field an output field. The field is an input/output field when you use it with the input-allowed entry in column 26. When you use it with other D-specification entries, you can specify the type of data to appear in this field. You must reserve space for the field in the application program output data area if the field is an output field. If you specify M for the constant type in column 56 of the D-specification, you need to reserve only 6 bytes for the field in the program.

Select one of the following entries for columns 23 and 24:

N (or blank)

This field is not an output field.

Y The type of information or data that appears in the output field depends on what you specify in the other columns of the D-specification for the field.

Figure B-7 shows the result of the output field, depending on the entries you make.

01 through 99

If you specify an indicator, the type of data the application program displays in the output field depends on the entries you make in the other columns of the D-specification for the field.

Figure B-8 on page B-16 shows the result of the output field, depending on the entries you make.

Figure B-7. Output Field Result When You Specify Y for Output Data

Value in Columns 23 and 24 (Output Data)	Value in Column 56 (Constant Type)	Contents of Columns 57 through 79 (Constant Data)	Result (Contents of Output Field)
Y	Blank	Blank	Output data is supplied by program.
		Data	Data in columns 57 through 79 appears.
	C	Blank	Blanks appear.
		Data	Data in columns 57 through 79 appears.
	M	Blank	Message identified by program appears.
		Message identification code (MIC) number and Message member identifier	Message identified in columns 57 through 62 appears.

Figure B-8. Output Field Result for Numeric Specification

Value in Columns 23 and 24 (Output Data)	Value in Column 56 (Constant Type)	Contents of Columns 57 through 79 (Constant Data)	Result (Contents of Output Field)
Indicator 01 through 99	Blank	Blank	If specified indicator is on, output data is supplied by program. If specified indicator is off, blanks appear.
		Data	If specified indicator is on, output data is supplied by program. If specified indicator is off, data in columns 57 through 79 appears.
	C1	Blank	Blanks appear.
		Data	Data in columns 57 through 79 appears.
	M	Blank	If specified indicator is on, a message identified by the program is displayed. If specified indicator is off, blanks appear.
		MIC number and Message member identifier	If specified indicator on, message identified by program is displayed. If specified indicator is off, message identified in columns 57 through 62 appears.

¹ When you specify a valid indicator in columns 23 and 24 (output data) and you specify C in column 56 (constant type), the application program issues a warning message and assumes Yes for the output entry. It ignores the indicator for this specification.

If the application program does an override operation and the specified indicator is on, data supplied by the application program or the message identified by the application program appears in

this field. See “Override Fields (Columns 33 and 34)” on page B-5 for information about override operations.

This field does not change if the application program does an override operation and the specified indicator is off.

If you specify a field as an output-only field, the operator cannot change data in the field. If you define a field as an input/output field, the operator can change data in the field.

For more information about specifying constant type, constant data, or messages and message members, see “Constant Type (Column 56)” on page B-24 and “Constant Data (Columns 57 through 79)” on page B-24.

Input Allowed (Column 26): Select one of the following entries for column 26:

N (or blank)

This field is not an input field.

Y This field is an input field.

The type of data the operator can type in the input field is identified by the data type entry described in “Data Type (Column 27).”

Data Type (Column 27): The data type entry specifies the type of data the operator can type in an input field. You cannot specify data type for an output field, except data type O when running the DBCS version of the OS/400 operating system. Entries allowed for the data type are as follows:

B (or blank)

This field can contain any type of character data. This definition is useful for input fields that require values made up of both alphabetic characters and numbers, such as addresses or part numbers.

A This field can contain only alphabetic characters (the characters A through Z) and some special characters. This definition is useful for input fields that require customer names or cities.

M This field can contain all alphabetic data, but on data entry keyboards, the keyboard will automatically shift to numeric shift for this field when manual shift is not active.

N This field can contain only numeric data, blanks, commas, periods, plus signs, or minus signs. This definition is useful for input fields

that are made up strictly of numbers, such as account balances or inventory amounts.

For some programming languages (such as RPG II or COBOL), when special characters (commas, plus signs, or minus signs) are entered in an N-type field, the program might not be able to use these characters. In that case, an error recovery routine should be included in the program that uses this format.

S This field can contain only signed numeric data; the last position of the field is reserved for a sign. Only the decimal digits (0 through 9) can be entered in the field, and a control field exit key must be used to exit from the field. The Field + key and the Field Exit key can be used to enter a positive value. The Field – key can be used to enter a negative value.

An input or input/output field with a data type of S (signed numeric) can be from 2 through 16 characters long. A signed numeric field requires one byte less in the program input or output data area than the field length.

R The field contains data to be read from the magnetic stripe reader. The field can have a maximum of 128 characters. If you specify data type R, nondisplay must also be specified on the SDA Field Attributes display or in columns 43 and 44 of the D specifications.

A data type of R cannot be specified in a display format used by a WSU program.

Z This entry specifies that the cursor moves from right to left within this input field. For more information about right-to-left display processing, see the description of column 40 of the S specification.

The SDA Field Attributes display does not allow a Z entry. You must use DSU or SEU to make this entry to the display format source specifications. A data type of Z cannot be specified in a display format used by a WSU program.

D Input data typed in this field can contain only the numeric characters 0 through 9. The Field – key cannot be used in this type of field; all other function keys can be used. A data type of D can be used on remote systems only if the system is attached to a 5294 work station controller. Remote screens attached to a 5251 Model 12 Display Station

will allow alphanumeric characters to be entered into a D type field.

- K** This field can contain Katakana characters.
Note: When column 40 of the S-specification (right-to left display) is a Y, the K data type is treated like the Z data type. When column 40 of the S-specification is not a Y, it is an error to have K data types and Z data types in the same display file.
- E** This field can contain alphanumeric (A/N) and Katakana characters or double-byte characters, but not both. The field is initially filled with binary zeros and the display station is set to enter alphanumeric and Katakana characters. The cursor blinks when it is in the first position of the field to indicate that the operator can switch modes and enter double-byte characters.
Note: The AS/400 system does not support the F data type. When data type F is specified, it is treated like the E data type.
- O** This field can contain any combination of alphanumeric, Katakana, and double-byte characters.
- X** This field can contain only double-byte data.

Notes:

1. E, F, O, and X are used only for the double-byte version of the OS/400 operating system. E, F, X, and O cannot be specified for a display format that is to be used by a remote work station attached through a 5251 Model 2 or 12.
2. If an E, F, or O field is an input/output field (Y in column 23 of the D specification, or columns 23 and 24 of the D specification specify an indicator that is on when the field is displayed), the output data, whether from the format or from the program, will overlay the alphanumeric or double-byte nulls and override the coded input attributes of the E, F, and O fields.
3. If an X field is an input/output field, the output data from either the format or from the program should be double-byte data.

On the AS/400 system, when \$SFGR converts the SFGR source to DDS, the data types are mapped as follows:

SFGR Data Type	DDS Data Type
blank	blank
B	A
A	X
M	N
N	M
S	S
R	I
Z	A
D	D
K	W
E	E ¹
F	E ¹
O	O ¹
X	J ¹

¹ On a nondouble-byte character set version of the system, these data types are mapped to the A data type in DDS.

Mandatory Fill (Column 28): Select one of the following entries for column 28:

N (or blank)

The operator does not have to fill this input field.

Y If the operator typed at least 1 character in the field, all positions in the field must be filled with characters the operator can type from the keyboard. You cannot specify adjust/fill in column 31 if you specify mandatory fill for the same field. The operator does not have to fill this field if you specify mandatory fill for an input/output field.

Mandatory Entry (Column 29): Mandatory entry means the operator must type at least 1 character in this input field. The system can return input from the display station to the user program.

Note: Input data does not return to the application program if a mandatory entry is bypassed.

If you specify mandatory entry, the operator can bypass a mandatory entry field if the return input entry is N (No), and the operator does not type data in any of the fields.

The operator can use the cursor keys to exit from a mandatory entry field. However, if the operator uses the cursor keys and adjust/fill is specified, no adjusting occurs.

Select one of the following entries for column 29:

N (or blank)

The operator does not have to type data in the field.

- Y** The operator must type at least 1 character in the field before input from the display station can return to the application program.

Self-Check (Column 30): If you specify **modulus 10** or **modulus 11** self-checking (formulas used to calculate the check digit for a self-check field), the farthest right position of the input field (the self-check digit) is checked by the correct check algorithm after the operator enters the field. Self-check fields cannot be longer than 32 positions.

Select one of the following entries for column 30:

blank

This input field is not a self-check field.

- T** This input field is a modulus 10 self-check field.

- E** This input field is a modulus 11 self-check field.

You can specify self-check for a right-to-left field (see column 40 of the S-specification and column 27 of the D-specification), but the position of the self-check digit remains at the right end of the input field. In a right-to-left self-check field, the first digit entered is the self-check digit. The self-check entry is ignored for fields having a data type of E, F, O, or X.

Adjust/Fill (Column 31): Select one of the following entries for column 31:

blank

If you leave column 31 blank, the system assumes blank fill on input for signed numeric input fields. The system does not adjust or fill for all other input fields.

- Z** The system right-justifies information entered in this field and fills unused positions with zeros before sending the field to the application program. A Z entry is ignored for fields that have a data type of E, F, O, or X.

- B** The system right-justifies information typed in this field and fills unused positions with blanks before sending the field to the application program. The system replaces leading zeros

with blanks for output to a signed numeric field (S specified for column 27).

Note: If you specify adjust/fill, the system assumes controlled field exit. The operator must press one of the field exit keys before the cursor leaves this input field. For more information, see “Considerations for Using the Adjust/Fill Entry” below and “Controlled Field Exit (Column 35)” on page B-20.

Considerations for Using the Adjust/Fill Entry:

Consider the following factors when you use the adjust/fill entry:

- You cannot specify adjust/fill and mandatory fill (in column 28) for the same field.
- To enter adjust/fill fields, the operator must press the Field+ key for numeric or signed numeric fields, the Field– key for signed numeric fields, or the Field Exit key. The operator can use the Field– key to enter a numeric-only field from a remote display station only.

The operator can press the Field Advance key to enter an adjust/fill field, but the adjust/fill does not occur. You should supply information explaining which keys you want the operator to use.

- When you specify adjust/fill for a right-to-left input field, no adjusting occurs when the operator uses the Field Exit key and the cursor direction is from right to left. An adjust/fill action can cause data typed to the right of the cursor to be lost when the cursor direction is from left to right.

Position Cursor (Columns 32 and 33):

Use the position-cursor entry to move the cursor to the first position of an input field if the keyboard is going from an inactive to an active state.

If the keyboard is active when this display format appears, the home position of the cursor changes, but the cursor position itself does not change.

See “Protect Field (Columns 37 and 38)” on page B-21 for considerations regarding the position-cursor entry when the input field is protected.

Select one of the following entries for columns 32 and 33:

N (or blank)

The cursor appears in the first position of this field if all the following conditions are present:

- This field is the first unprotected input field on the display.
- You have not specified a Y (Yes) in any field for the position-cursor entry.
- You have not specified an indicator that is on for any other fields for the position-cursor entry.

Y The cursor appears at the first position in this input field unless you have specified indicators for other fields in columns 32 and 33 and one or more of these indicators are on when the display format appears. In this case, the cursor appears in the first position of the first field on this display format specifying a cursor position. You can specify Y for only one field in the display.

01 through 99

If the indicator is on when display format appears, the cursor appears in the first position of this input field. If more than one field has the cursor on, the cursor appears in the first position of the first field on the display with a cursor position specified.

Note: On the AS/400 system, a cursor position specified with an indicator does not necessarily override a cursor position specified with a Y. The system uses the first active cursor position encountered.

Enable Dup Key (Column 34): When the Dup (duplicate) key is enabled and the operator presses the Dup key, the system fills the position of the cursor and the remainder of the field with a character called the **duplicate character**. The duplicate character is displayed as an overscored asterisk, as follows:

([¯])

RSLW106-0

This duplicate character has a hexadecimal value of X'1C'. The operator can type data into the first record and press the Dup key for following records to contain the same data. For each following record, the program removes the duplicate characters and replaces them with the actual data.

Select one of the following entries for column 34:

N (or blank)

The field does not allow Dup. An error appears if the operator presses the Dup key while the cursor is in this input field.

Y The field allows Dup. The operator can press it while the cursor is in this input field.

Controlled Field Exit (Column 35): If you specify controlled field exit, the operator must press one of the field exit keys before the cursor leaves this input field. The following are field exit keys:

- Field Advance
- Field Exit
- Field+
- Field–
- Field Backspace
- Home
- Erase Input
- Duplicate

Select one of the following entries for column 35:

N (or blank)

The cursor automatically exits from the field when it is filled. If you specify adjust/fill in column 31 of the D-specification, the system ignores an N or blank entry.

Y The operator must press one of the field exit keys before the cursor leaves this input field.

Note: The operator can use the cursor keys to exit from a controlled exit field. No adjusting occurs.

Automatic Record Advance (Column 36): Select one of the following entries for column 36:

N (or blank)

The system does not return the data typed in the input fields on this display to the application program until the operator enters the entire display.

Y When the operator types data into the input fields on the display, the system automatically returns to the application program when one of the following occurs:

- You type the last character in this input field and specify N (No) or blank for controlled field exit.

- The cursor is in this input field and the operator presses the Enter key, the Field Adv key, the Field Exit key, the Field+ key, the Field– key (if the field is a signed numeric field), or the Dup key.

Protect Field (Columns 37 and 38):

Use the protect-field entry to prevent an operator from entering or changing data in an input field.

N (or blank)

The operator can type data in this field.

- Y** The operator cannot type data into this field. You can specify Y (Yes) for input fields only.

01 through 99

If you specify an indicator, the operator cannot type data into this input field when the indicator is on.

Considerations for Using the Protect-Field

Entry: Protect field can be specified for input fields only. If you specify Y (Yes) or an indicator for the protect-field entry, some or all of the following considerations may apply:

- If you use an override operation, the system ignores the protect field indicator.
- The cursor might still appear in a protect field if all the following conditions exist:
 - The field is protected by an indicator that is on when the display format appears.
 - The field is the first input field (that does not specify a Y in column 37) defined by a D specification in the format.
 - The cursor is not positioned by an indicator to any field not protected.

Note: Although the cursor may appear in a protect field if all of these conditions exist, the operator cannot type input data into the field.

- If a field is defined as either nondisplay or protected, or both, and column separators are requested, the column separators are displayed on a 5251 Display Station and on a 5291 Display Station; the column separators are *not* displayed on a 5292 Color Display Station.
- If an indicator controls either the nondisplay or protect field attributes, or both, and column separators are requested, the separators are

not displayed if the indicator is on when the field is displayed.

High Intensity (Columns 39 and 40):

Select one of the following entries for columns 39 and 40:

N (or blank)

The application program shows characters in this field with normal intensity.

- Y** The program shows characters in this field with high intensity.

01 through 99

The application program shows the application characters in this field with high intensity if the specified indicator is on, and with normal intensity if the specified indicator is off.

If the display format appears on a 5292 Color Display Station and you specified high intensity, the field appears with white characters. The color may be different if you also specify other field attributes. See Figure B-9 on page B-23 for the result of specific attribute combinations.

You cannot specify high intensity, reverse image, and underline for the same field at the same time. If the operator tries to display the field with all three attributes requested, the application program shows the field as a nondisplay field.

Blink Field (Columns 41 and 42): Select one of the following entries for columns 41 and 42:

N (or blank)

The characters in this field do not blink when the display format appears.

- Y** The characters in this field blink when the display format appears.

01 through 99

The characters in this field blink if the specified indicator is ON when the display format appears. The characters in this field do not blink if the specified indicator is off when the display format appears.

If the display format appears on a 5292 Color Display Station and you specified blink field, the application program shows the field with red characters and without blink. Specify high intensity to cause the characters in the field to blink. For

more information, see Figure B-9 on page B-23 for the result of specific attribute combinations.

Nondisplay (Columns 43 and 44):

When you specify nondisplay for an output or input field, information sent in the field is not visible on the display.

Select one of the following entries for columns 43 and 44:

N (or blank)

The information in this field displays.

Y Information in this field does not display.

01 through 99

Information in this field does not appear if the specified indicator is on.

Consider the following factors if you specify Y (Yes) or an indicator in the nondisplay entry:

- If you specify nondisplay and high intensity, reverse image, or underline, the system defines the field as nondisplay only.
- If you specified the nondisplay attribute, protect-field attribute, or both, and you requested column separators, the column separators appear on a 5251 Display Station and on a 5291 Display Station. They do not appear on a 5292 Color Display Station.
- The system does not display column separators if you specified Y (Yes) for one of the nondisplay or protect-field entries and the other is controlled by an indicator, or if indicators control both the nondisplay and protect-field attributes.

Reverse Image (Columns 45 and 46):

Select one of the following entries for columns 45 and 46:

N (or blank)

The characters in this field appear as light characters on a dark background.

Y The characters in this field appear as dark characters on a light background.

01 through 99

The characters in this field appear as dark characters on a light background if the indicator is on when the display format appears.

You cannot specify reverse image, high intensity, and underline for one field at the same time. If the operator tries to see a field with all three attributes requested, the application program shows the field as nondisplay.

Underline (Columns 47 and 48): You can use an underline to show the location or length of an input field, or to emphasize information displayed in an output field.

Select one of the following entries for columns 47 and 48:

N (or blank)

The field is not underlined.

Y The field is underlined.

01 through 99

The field is underlined if the indicator is on when the display format appears.

If this display format appears on a 5292 Color Display Station and you specify underline, the field appears with a blue line under the character positions in the field. The color of the characters displayed in the field depends on the other field attributes you specified. See Figure B-9 on page B-23 for the result of specific attribute combinations.

You cannot specify underline, reverse image, and high intensity for the same field at the same time. If the operator tries to display the field with all three attributes requested, the application program shows the field as nondisplay.

Column Separators (Column 49):

Column separators are vertical lines or dots that precede and follow each character position in a field. They do not require additional character positions.

Select one of the following entries for column 49:

N (or blank)

Column separators do not appear in this field.

Y Each character position in this field is preceded and followed by column separators.

Note: If this display format appears on a 5292 Color Display Station, the column separators appear as blue dots at the bottom corners of each character position in the field. If you also specify blink field, the column separators do not appear

on the display. The color of the characters displayed in the field depends on the other field attributes you specify. See Figure B-9 for the results of specific attribute combinations.

If an indicator controls the nondisplay attribute, protected field attribute, or both, and you request column separators, the separators do not appear if the indicator is on when the field appears.

The first dot of a column separator appears in the character cell of the field starting attribute. The last dot of a column marked *Field* appears in the character cell of the field ending attribute. The fol-

lowing occurs when a field starts in the farthest left column of the display:

- A column mark appears at the end of the row preceding the fields (that is, at the right of the field starting attribute).
- The first column mark in the field is dimmer than the other column marks because it is a single dot only.

A similar situation occurs when the field ends in the farthest right column of the display.

Figure B-9 shows the results of specific attribute combinations.

Figure B-9. Using Field Attributes to Control Color

Attributes Specified:	Column Separators ¹	Blink Field	High Intensity	Reverse Image	Underline Can Also Be Specified
Green				X	X
Green, reverse image					X
White			X	X	X
White, reverse image			X		
Red		X ²	X	X	X
Red, reverse image		X ²	X	X	X
Red, blink		X			X
Red, reverse image, blink		X			
Turquoise, column separators	X			X	X
Turquoise, reverse image, column separators	X				
Pink	X ³	X ²		X	X
Pink, reverse image	X ³	X ²			X
Yellow, column separators	X		X	X	X
Yellow, reverse image, column separators	X		X		
Blue	X ³	X ²	X	X	X
Blue, reverse image	X ³	X ²	X		
Data in fields with these combinations of attributes does not appear.	X	X	X	X	X
	X	X	X	X	X
			X	X	X
			X	X	X

1 Column separators do not appear when you use reduced line spacing.

2 Field does not blink.

3 Column separators do not appear.

Notes:

1. Underlines and column separators are always blue.
2. Underlines do not blink if you also specify blink field.
3. Column separators do not appear if you also specify blink field.
4. Use the limit color select option of the 5292 Color Display Station to see how a display format designed for color appears on a single-color display.

Lowercase (Column 51): If you specify lowercase, the system displays and sends all alphabetic characters typed into this field to the application program in lowercase (if the operator does not press the Shift key when typing a character) or in uppercase (if the operator presses the Shift key when typing a character).

The D-specification entry for lowercase always takes precedence over the S-specification entry for lowercase.

You can use the lowercase entry for input fields only. Lowercase data cannot be specified for numeric fields. The lowercase entry is ignored for fields with the X data type.

Use Figure B-10 on page B-25 to determine what the operator can type in the input field depending on what you specify in the display format specifications.

Columns 50, 52 through 55: Leave columns 52 through 55 blank. The system does not use them.

Constant Type (Column 56): You can define information as a constant on the SDA Attribute Work display or in columns 57 through 79 of the D-specification, or you can indicate that a message is displayed by specifying a MIC and a message member identifier.

If you specify Y (Yes) for output data columns 23 and 24, you can select the following entries for column 56:

blank

If column 56 is blank, and you define constant information for this field in columns 57 through 79 of the D-specification, that information

appears in this field. If you leave column 56 blank and no constant information is defined, information sent from the application program appears.

C If you specify C, the constant information defined in columns 57 through 79 of the D-specification appears in this field. C is required only if the constant information is all blanks, and you want to show all blanks in the field. For more information on possible combinations, refer to "Output Data (Columns 23 and 24)" on page B-15.

M A message appears in this output field. The user identifies the message by an MIC and a message member identifier in columns 57 through 62 of the D-specification. If columns 57 through 62 are blank, the application program or PROMPT OCL statement must specify the MIC and message load member identifier.

The \$SFGR compiler assumes that the message ID has a prefix of USR. On the AS/400 system, a message has the format PPPmmmm, where PPP is the message ID prefix and mmmm is the MIC. Your application program should not supply the prefix.

For more information, see "Output Data (Columns 23 and 24)" on page B-15.

Constant Data (Columns 57 through

79): The constant data entry specifies the information to be included in an output or input/output field when the display format appears.

If you specify Y (Yes) for output data columns 23 and 24, you can select the following entries for columns 57 through 79:

- If you specify C or blank in column 56 (constant type), columns 57 through 79 contain the information that will appear in the field. If you leave columns 57 through 79 blank, blanks or data supplied by the application program appear in the field.
- If you specify M in column 56 (constant type), the application program shows the message identified by a 4-digit MIC in columns 57 through 60 and a 2-character message member identifier in columns 61 and 62. The MIC number identifies the message containing the information to appear in the field.

Entry in Column 21 of the S Specification
or on the SDA Format Attributes Display

		Blank	N	Y
Entry in Column 51 of the D Specification	Blank	uppercase only	uppercase only	lowercase or uppercase
	N	uppercase only	uppercase only	uppercase only
	Y	lowercase or uppercase	lowercase or uppercase	lowercase or uppercase

RSLW098-1

Figure B-10. Entries to Specify What the Operator Can Use in the Input Field

Depending on the length of the output field, the system fills the message text with blanks or shortens it to the length of the field when the display format appears.

The message member identifier identifies the message member that contains the message. The message member identifier can be one of the following:

U1 or blank

User-1 message member

U2 User-2 message member

P1 Program-1 message member

P2 Program-2 message member

M1 System/36 environment level-1 message member (##MSG1)

M2 System/36 environment level-2 message member (##MSG1)

If you specify a MIC, but do not specify a message member identifier, the system assumes U1 as the message member. A user or procedure can redirect the system to the proper message member and library by using the //MEMBER OCL statement, or by using the Override Message File (OVRMSGF) CL command. The //MEMBER OCL statement takes precedence over the OVRMSGF CL command.

If you leave columns 57 through 79 blank, the following happens:

- If an application program is used to show the display and the program passes a MIC number, the message text for that number is shown on the line of the display file.
- If an application program is used to show the display and the program does not pass a MIC number, the line on the display is left blank.

- If the PROMPT OCL statement is used to display a screen, the system looks at the value of the parameter that corresponds with the output field of the display file. If that value is a MIC number, the message text for that MIC number is shown on that line of the display file. If the value is blank, the line on the display file is left blank.

For example, if you specify M in column 56 of the D specification for the third output field on a display file, leave columns 57 through 79 blank, and the third parameter of the procedure that contains the PROMPT statement is 0001, the message text for MIC number 0001 is displayed in the third output field. If the value for the third parameter of the procedure is not given, the third output field is left blank.

If the MIC is not found, or if the message member is not found, then the MIC, followed by ?? is shown on the display file where the message text would normally be shown. For example, if you specify MIC 1234, but this MIC has not been defined in the message member, 1234?? would be shown on the display file. The job log for the job would also contain diagnostic message CPF2419, with more information about the MIC and the message file. If the message member could not be found, escape message CPF2407 would be in the job log with the message file and library name identified.

If the MIC is not numeric or not between A and F, then the MIC followed by the message member identifier is shown on the display file where the message text would normally be shown. For example, if

you specify MIC 123G and a message member identifier of U1, 123G*USR1 would be shown on the display file.

For information about assigning user-1, user-2, program-1, and program-2 message members, see the description of the MEMBER OCL statement in the *System/36 Environment Reference* book.

For more information about what is displayed in the field, see "Output Data (Columns 23 and 24)" on page B-15.

Considerations for Using the Constant Data

Entry: Consider the following factors when you use the constant data entry:

- Information from the program output data area appears if you do not specify C in column 56 and you leave columns 57 through 79 blank, and if the field is an output field (Y (Yes) specified in column 23).
- If you specify an MIC in columns 57 through 79, you need to reserve only 6 bytes for the field in the program output data area.
- M1 and M2 are valid only if you specify SSP-YES on the LOADMBR utility control statement of the \$SFGR utility program. The \$SFGR utility program is run by the FORMAT procedure. See the description of the \$SFGR utility control statements in the *System/36 Environment Reference* book for more information.

Continuation (Column 80): The system requires an entry to this column only if you are using the D-specification coding form and the output-constant information is greater than the space in columns 57 through 79.

If you use more than 23 positions of constant data

(specified in columns 57 through 79), you must also make an entry in column 80. You must specify an X in column 80 to indicate that the next record (or D-specification) is a continuation of this record. Positions 7 through 79 of the next D-specification contain the continued constant data.

A comment, indicated by an asterisk (*) in column 7, cannot follow a D-specification with an entry made in column 80.

The constant for the DBCS version of the OS/400 operating system can contain DBCS characters. The shift-out (SO) and shift-in (SI) characters are counted as part of the data. If the constant output consists of more than 10 DBCS characters, you must code an alphanumeric X in column 80. The DBCS data continues on the next D-specification.

If a Shift-in (SI) character is in column 78 or 79 and the constant continues with a Shift-out (SO) character in column 7 of the next D-specification, the system deletes the SI/SO pair. When column 78 contains a SI character, any character in column 79 is omitted when the constant is joined and displayed.

When column 78 contains an SO character, and column 79 contains an SI character, the system deletes the SO/SI pair regardless of what character is in column 7 of the next D-specification.

If you are using extended DBCS characters, put an SO character in column 57 to show the DBCS characters correctly. You can also specify Open Data Type (O) for the output field. This specification causes the application program to show the DBCS character correctly.

Appendix C. Merging Graphics and Text

This appendix describes how to print a graphics file alone or with other output.

System/36 supports the intelligent printer data stream (IPDS) graphics/text merge functions with the following PRPQs:

- Program Number PRPQ P84096 (5799-CGJ) for the 5360/5362 System Unit
- Program Number PRPQ P84097 (5799-CGP) for the 5363/5364 System Unit

When using graphics/text merge functions, you must use an IPDS printer.

Printing a Graphics File Only

This function allows you to print graphic files, such as Business Graphics Utility (BGU) graphs or charts, separate from other print files.

The procedure for printing graphics alone is:

```
PRTGRAPH prtId,FILE,filename,width
```

The parameters are described as follows:

prtId

Is the ID of the IPDS printer to print the graphics file. The default printer is the session printer.

FILE

Indicates that the data is in a disk file.

filename

Is the name of the graph object file to be printed. The file with the latest date is used. This function also supports remote files.

width

Is the width of the graphics area in inches. The width should not exceed the width of the paper. The default is 13.2 inches.

This parameter cannot contain more than 5 characters, specified in decimal numbers of any combination of xx.xx, where x is any number from 0 through 9. The specified value cannot be more than 13.2 inches.

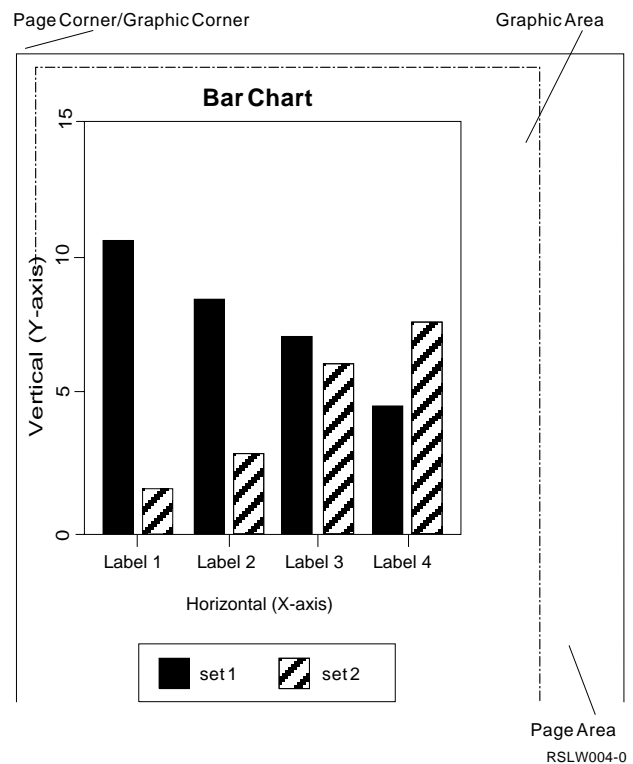
The graphics area is placed as follows:

- The upper left corner of the graphics area is the same as the upper left corner of the page.
- The length of the graphics area is the same as the length of the page.

Note: The printer will scale the graph to fit a square area. The dimensions of the square are determined by the value for either the width or the length of the graphics area, whichever is smaller.

Graphics File Printout Example

The following figure is an example of a graphic printout.



The printing of a graphics file function generates no error messages. However, if an error occurs, the #\$\$\$INCLGRPH control record prints in place of the graph.

Printing a Graphics File Along with Other Output

This function allows a user to include a graph anywhere in the data that is printed by any program (for example, AS/400 word processing, COBOL, or RPG). You can create the graphics file by using such utilities as the Business Graphics Utility (BGU) or the IPDS Advanced Functions.

For more information about the BGU, see the *BGU User's Guide and Reference* book. For more information about word processing, see the *Getting Started with OfficeVision/400* book or the *Using OfficeVision/400 Word Processing* book. For more information about the IPDS advanced functions, see Appendix D, "Intelligent Printer Data Stream (IPDS) Advanced Function Support."

To include graphics with other program output, use a special control record. The format for this control record is:

```
#$@INCLGRPH filename,x,y,w,l
```

The parameters are described as follows:

filename

Is the name of the graph object file to include. The file with the latest date will be used. This function also supports remote files.

- x** Is the distance in inches from the left edge of the page to the left edge of the graphics area on the page. The default is 0.
- y** Is the distance in inches from the top of the page to the top edge of the graphics area on the page. The default is 0.
- w** Is the width of the graphics area in inches. The default is the width of the current page you are using.

- l** Is the length of the graphics area in inches. The default is the length of the current page you are using.

Consider the following when using the #\$\$@INCLGRPH control record:

- Parameters x, y, w, and l define the area on the page where the graphics file will print. Parameters x and y define the upper left corner of the graphics area, and parameters w and l define the size of the graphics area.
- There must be only *one* space between the command word and the parameters.
- The #\$\$@INCLGRPH control record should be in a print record by itself, because any other data along with it may be considered as parameters.
- Actual parameters should immediately follow one another, separated by commas, using no blanks.
- The specified values for width (w) and length (l) should be equal, since the printer will scale the graphic to fit a square area.
- Parameters x, y, w, and l can be specified in decimal numbers of any combination of xx.xx, where x is any number from 0 through 9. The specified value cannot be more than 22.75 and, if a 0 value is specified, the result is the default value for that parameter.
- The characters #, \$, and @ must be given as defined in code page 500. That is, they must be X'7B', X'5B', and X'7C' respectively.

If there is anything wrong with the control record or an error occurs while processing the graphics file, the control record will print as normal text data.

Example of an Included Graphics File

The following figure is an example of a graphics file that is included with other output:

Sample of Graphic/Text Merge Page

This is a sample page of text produced by using the word processing function of the AS/400 Office program product with a bar graph merged onto the page. The graph/text merger was done by inserting an include graphics control record in the text of the page on which the graph was to appear.

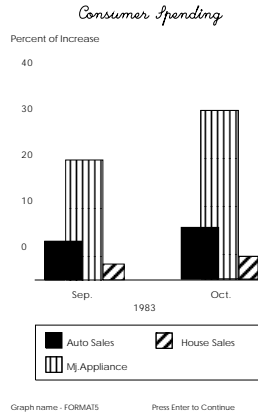
The graphic control record for making this sample graphic/text merge is:

```
#$$@INCLGRPHBARGRAPH,3.75,2.5,4,4
```

where BARGRAPH is the name of the graphic object file to be printed; 3.75, 2.5 is the distance in inches from the upper left corner of the page to the upper left corner of the graphic area on the page; and 4, 4 is the width and length, respectively, of the graphic area on the page.

This control record can be added anywhere on the displayed page of text. You must, however, edit your page to provide adequate blank space for your graphic or else the graphic will overlay the text.

The graphic area will always be a square (all sides are equal) of the size of the smallest specified value for w (width) and l (length). If the size of the graphic area is too small or the graphic too complex, the graphic may not be legible. You can enhance legibility by making the graphic area larger or by editing the graphic to reduce the amount of detail shown and building a new graphic object file. You can also choose a different character style and character size in the graphic to enhance legibility.



RV2W108-1

Programming Considerations

This section includes considerations for printing a graphics file along with other input using #\$\$@INCLGRPH. The following considerations refer to Figure C-1:

- A** If you use the #\$\$@INCLGRPH control record in an application program, *do not* place the entire word (#\$\$@INCLGRPH) all together in the source program.
- B** If #\$\$@INCLGRPH is placed all together, and the compiler output prints on an IPDS printer, the result is that the graphics file prints with the compiler output as well as in the program output.
- C** To prevent printing the graphics file with the compiler output, divide #\$\$@INCLGRPH into two parts and have the program concatenate the parts when the program runs.

Printer Storage Limitations: All of the included graphic files for a print file are downloaded to the printer storage, separate from the rest of the print file. There are two conditions you must consider:

- The included graphic files may overflow the printer storage and cause an error condition. To prevent this, include fewer graphs or make the included graphs less complicated. For example, choose a simpler character type or smaller character size, remove extraneous text and details, and so forth.
- All included graphic files in printer storage are deleted from storage at the start of a new page.

A WORKING-STORAGE SECTION.
 01 INCLUDE-RECORD.
 02 FILLER PIC X(28) VALUE '#\$@INCLGRPH FORM, 0, 0, 8, 5, 11'.
 02 FILLER PIC X(52) VALUE SPACES.

Entire Word

B

STNO -A . . . B . . . C O B O L S O U R C E S T A T E M E N T . . . IDENTFCN SEQ/NOS

Wholesale
Tire & Supply
Company

10% Discount
On All Orders Over
\$500.00

AccountNumber

1 IDENTIFICATION DIVISION
 2 PROGRAM-ID IS TIRE DRIVE
 3 AUTHOR. LL HIRSH. CUSTOMER FOR DEPT 48X.
 4 INSTALLATION ROCHESTER.
 5 DATE-WRITTEN
 6 DATE-COMPILED. 85/11/20.
 7 SECURITY.
 COBOL TEST PROGRAM - D10105 Bill To:

THIS PROGRAM IS USED FOR TESTING THE GRAPHICS/TEXT MERGE USING THE #@\$@INCLGRPH CONTROL RECORD. IN THIS TEST CASE, THE CONTROL RECORD STRING APPEARS TOGETHER ON THE COMPILED OUTPUT; SO THAT THE GRAPH WILL BE MERGED AT THE TIME OF THE COMPILER OUTPUT AS WELL AS ON THE PROGRAM OUTPUT.

Item	Description	Qty	Unit Price	Total
8	ENVIRONMENT DIVISION.			
9	CONFIGURATION SECTION.			
10	SOURCE-COMPUTER IS IBM-S36			
11	OBJECT-COMPUTER, IBM-S36.			
12	SPECIAL NAMES.			
13	C01 IS TO-NEW-PAGE.			
14	UPSI-01S			
15	LOCAL-DATA IS LOCAL-DATA-AREA.			
16	INPUT-OUTPUT SECTION.			
17	FILE-CONTROL.			
18	SELECT PRINT-FILE ASSIGN TO PRINTER-CBLPRT.			
19	SELECT CUST-FILE ASSIGN TO DISK-CUSTFILE.			
20	SELECT PART-FILE ASSIGN TO DISK-PARTFILE.			
21	DATA DIVISION.			
22	FILE SECTION.			
23	FD PRINT-FILE			
	LABEL RECORDS ARE OMITTED.			
	RECORD CONTAINS 80 CHARACTERS.			
	DATA RECORD IS PRINT-RECORD.			
24	01 PRINT-RECORD PIC X(80).			
25	FD CUST-FILE			
	LABEL RECORDS ARE STANDARD.			
	RECORD CONTAINS 90 CHARACTERS.			
	DATA RECORD IS CUST-RECORD.			
	01 CUST-RECORD.			
26	02 CUST-NAME-IN			
27	02 STREET-IN		Subtotal	\$
28	02 CITY-IN PIC X(28).		Less Discount	\$
29	02 STATE-IN PIC X(20).		Total	\$
30	02 ZIP-IN PIC X(15).			
31	02 ACCT-NUM-IN PIC X(2).			
32	02 FILLER PIC X(5).			
33	PIC X(11).			
	PIC X(9).			

Graphic Overlays
Compiler Output

C WORKING-STORAGE SECTION.
 01 INCLUDE-RECORD.
 02 FILLER PIC X(3) VALUE '#\$@'.
 02 FILLER PIC X(28) VALUE 'INCLGRPH FORM, 0, 0, 8, 5, 11'.
 02 FILLER PAC X(53) VALUE SPACES.

Divided Word

RSWL000-1

Figure C-1. Output Example when the Control Word (#@\$@INCLGRPH) in the Source Program is Not Divided

Appendix D. Intelligent Printer Data Stream (IPDS) Advanced Function Support

This appendix describes the following functions of the intelligent printer data stream (IPDS) advanced function support and tells how to use them. The **intelligent printer data stream (IPDS)** is an all-points-addressable data stream that allows users to position text, images, and graphics at any defined point on a printed page.

- Calling the IPDS advanced function subroutines
- Using the IPDS advanced function support, including:
 - Setting printer options
 - Printing graphics using RPG II and COBOL subroutines
 - Printing forms and graphs
 - Printing bar codes

“Sample Form” on page D-14 describes how to print a sample form to demonstrate the use of the forms generation utility.

System/36 supports the IPDS advanced functions with the following PRPQs:

- Program Number PRPQ P84094 (5799-CGK) for the 5360/5362 System Unit
- Program Number PRPQ P84095 (5799-CGL) for the 5363/5364 System Unit

When using IPDS advanced functions you must use an IPDS printer. The printer file must also specify an *SCS data type and to spool the output. If you do not spool the data, the AS/400 will override the *SCS data type, and may generate incorrect output.

Calling the Subroutines

This section includes information on calling the subroutines when using IPDS advanced functions.

COBOL Subroutines

The following are COBOL subroutines:

PRTAPI Sets printer options
PRTGRC Prints graphics
PRTBAR Prints bar codes

The following coding example shows how to call the PRTAPI subroutine. The same coding format is used for the PRTGRC and PRTBAR subroutines.

```
CALL 'PRTAPI' USING FNAME,OPTION,PARM,RTCODE
```

FNAME is the name of the printer file. For example, if the name of your printer file is COBPRT, enter the following:

```
CALL 'PRTAPI' USING COBPRT,OPTION,PARM,RTCODE
```

The parameters to be passed to the subroutine are described in “RPG II and COBOL Printer Parameters.”

RPG II Subroutines

The following are RPG II subroutines:

SUBR50 Sets printer options
SUBR51 Prints graphics
SUBR52 Prints bar codes

Figure D-1 on page D-2 is a RPG II coding example calling the SUBR50 subroutine. The same coding format is used for SUBR51 and SUBR52.

The parameters to be passed to the subroutine are described in “RPG II and COBOL Printer Parameters.”

RPG II and COBOL Printer Parameters

The following are the printer parameters passed to COBOL and RPG II subroutines. All printer parameters must be left-justified.

FNAME

Is an 8-character field that contains the name of the printer file.

OPTION

Is an 8-character field that tells the system which printer option to specify. See “Setting Printer Options” on page D-2 for descriptions of the options.



RPG CONTROL AND FILE DESCRIPTION SPECIFICATIONS

Program	Keying Instruction	Graphic	Card Electro Number
Programmer	Date	Key	

Page 1 2 of ___ Program Identification 75 76 77 78 79 80

Line	Form Type	Control List (C, L, LR, OR, AND, OR)	Indicators			Factor 1	Operation	Factor 2	Result Field		Resulting Indicators			Comments
			And	And					Name	Length	Plus	Minus	Zero	
			Not	Not	Not						High	Low	Equal	
01	C						E: X: I: T: S: U: B: R: 5: 0:							
02	C						R: L: A: B: L		F: N: A: M: E:	8				
03	C						R: L: A: B: L		O: P: T: I: O: N:	8				
04	C						R: L: A: B: L		P: A: R: M:	8: 0				
05	C						R: L: A: B: L		R: T: C: O: D: E:	2				

RSLW001-1

Figure D-1. RPG II Subroutine Coding Example

PARM

Is an 80-character field that contains the value for the OPTION specified.

- Printing forms and graphs
- Printing bar codes

RTCODE

Is a 2-character field that contains the return code. The subroutine returns this value to the application program to indicate the result of the printer operation.

Valid values are:

Value	Description
40	Normal completion.
41	The specified OPTION is not valid.
42	The specified PARM is not valid.
43	An I/O error was detected by printer data management.
44	The specified FNAME is not valid.
45	The specified OPTION is not valid in this sequence.
46	The specified OPTION is not supported by this printer.

Setting Printer Options

The advanced functions support allows a user application program to specify the printer options, using the following subroutines:

- PRTAPI for COBOL programs
- SUBR50 for RPG II programs

The options you specify are used until you change them or until the start of the next print job. Then the printer default values are reset (except when otherwise noted later in this manual).

These subroutines control the following printer options:

- Characters per inch (cpi)
- Character style (font)
- Color
- Drawer selection
- Emphasis (highlight, make bold)
- Forms characters (for building boxes)
- IPDS transparent options (IHTRANS, IPTRANS)
- Lines per inch (lpi)
- Print quality
- Page rotation
- Transparent

Using the IPDS Advanced Function Support

This section describes how to use the advanced functions listed below:

- Setting printer options
- Printing graphics using RPG II and COBOL subroutines

Printer	Trans- parent	CPI	LPI	Fonts	Text	Drawer	Forms	Em- pha- sis	Print Quality	Color	IPDS Trans- parent	Rota- tion
3812	yes	10, 12 15	3, 4 6, 8	yes	yes	1, 2	no	no	yes	no	no	yes
4214	yes	5, 10 12, 15 16.7	3, 4 6, 8	no	yes	1, 2, 3	yes	no	yes	no	no	no
4245	yes	no	no	no	no	no	no	no	no	no	no	no
5219	yes	10, 12 15	6, 8	yes	yes	1, 2	no	no	yes	no	no	no
5224	yes	10, 15	3, 4 6, 8	no	no	no	yes	no	no	no	no	no
5225	yes	10, 15	3, 4 6, 8	no	no	no	yes	no	no	no	no	no
5256	yes	no	no	no	no	no	no	no	no	no	no	no
5262	yes	no	6, 8	no	no	no	no	no	no	no	no	no
6262	yes	no	no	no	no	no	no	no	no	no	no	no
IPDS	yes	no	4, 6, 8	yes	yes	no	no	yes	yes	yes	yes	no

RSLW109-0

Figure D-2. IPDS Advanced Function Support AS/400 Printer Options

IPDS Advanced Function Support Printer Options and the AS/400

Printers: Not every printer option can be used by all AS/400 printers. Figure D-2 shows the options that you can use with each AS/400 printer or its equivalent.

Note: In this figure, IPDS means intelligent printer data stream printers.

You must be sure that the options you specify are valid for the printer that you use. Otherwise, an error occurs. If you specify printer options in a spool file, the file could be sent to a printer that does not support these options, and an error could occur.

Characters per Inch Option: The characters per inch (cpi) option lets you specify the number of characters to be printed per inch. Valid values are 5, 10, 12, 15, and 16.7. There is no default for this option. A parameter value must be specified.

Consider the following when specifying characters per inch:

- When you specify 5 cpi for the 4214 Printer, the printed characters are double-wide.

- When the record length of the printer file is greater than 132, the printer OCL statement must set characters per inch to 15, even though the IPDS advanced function support may set characters per inch to 12, 15, or 16.7 within the program.
- When you change the characters per inch option for the 5219 Printer, the font ID that normally corresponds with this characters per inch value display on the 5219. Press the Start key on the 5219 to start printing. You do not need to change the print wheel to the font that is displayed.

Character Style Option: You can specify any character style (FONT) ID of up to 4 hexadecimal characters from 01 to FFFE.

You must specify a parameter value because there is no default for this option.

Consider the following when specifying character style:

- Selection of an F0 through F9 character style results in double-wide characters on IPDS and 3812 printers.

- On IPDS printers, print quality and character style selections interact with one another. To get the best results, print quality should be set first, then the character style. See “Print Quality Options” on page D-6 for information on setting print quality.

Printing OCR Characters: To print optical character recognition (OCR) characters on the 3812 printer, you must set the code page to 340 (hex 0154), using the transparent (TRANS) option. For information on the TRANS option, see “Transparent Option” on page D-5. The parameter for the TRANS option is:

2BD1060100000154

After setting the code page, select either of the character styles OCR-A or OCR-B, using the character style option (FONT). The parameters are:

Hex Value	Character Style
0013	OCR-A
0003	OCR-B

Before changing back to another non-OCR character style, use the TRANS option to set the code page to your default value.

Printing Mathematical Symbols: To print mathematical symbols on the 3812 printer, perform the same steps as above for printing OCR characters, except use the math symbols font code (hex 001E) for the character style option parameter and code page 259 (hex 0103) for the transparent option parameter. At the start of the next job, the code page is set to the value selected at the printer operator panel.

Color Option: The color (COLOR) option is only valid for the color models of IPDS printers. This option sets the color for the printed output. The parameters are:

BLACK	RED
BLUE	TURQ (turquoise, also cyan)
BROWN	YELLOW

GREEN WHITE (no color)
PINK (magenta)

The default is BLACK.

Consider the following when specifying the color option:

- If the specified color cannot be printed by the ribbon currently on the printer, black is printed.
- The color option WHITE does not print a character, but allows the printer to advance (print a blank) for each character printed.

Drawer Selection Option: Select the printer source drawer for paper using the drawer selection (DRAWER) option. Valid values are 1, 2, and 3. You must specify a parameter value since there is no default.

When you specify the DRAWER option, you automatically start a new page of printing. If you are not currently printing a page, a blank page may be fed from the current drawer.

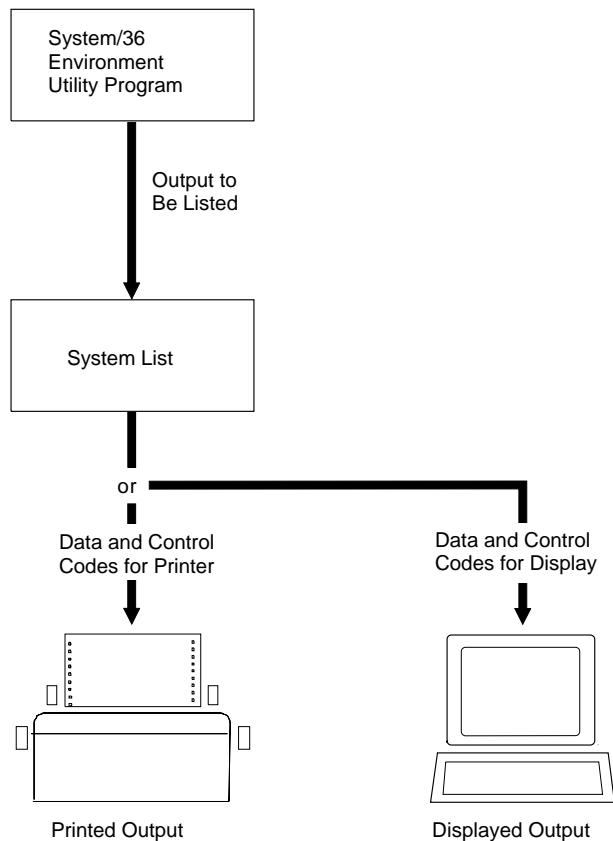
Emphasis Option: The emphasis (EMPHASIS) option causes bold characters to print on IPDS printers by using a double-strike character style (FONT). The parameters are:

Parameter	Description
ON	Begin emphasis printing
OFF	End emphasis printing

You must specify a parameter value for this option because there is no default for this option.

Forms Characters Option: The forms characters (FORMS) option downloads to the printer, a set of 11 characters used for drawing boxes. Valid values for this option are a blank, 22 hexadecimal characters, or RESET.

You can use a set of default values for these characters by leaving the PARM field blank. The following figure shows the default values for the box characters:



RSLW008-3

Note: The default character values in the preceding figure are for the U.S. character set. If you use a character set other than U.S., the default hexadecimal values remain the same, but the default characters that they represent may be different.

When you use the FORMS values to print box characters, the box or forms characters replace the characters that would normally be printed. The printer documentation contains a list of hexadecimal code point values for normal operation.

If you do not want to use the default value for one or more of the box characters, you can specify a different hexadecimal value for that character. This value should be greater than hex 40. You must specify a hexadecimal value for *each* box character, even if only one of the hexadecimal values is different from the default values. For example, if you do not want the less than (<) character to be used as a box character, you could specify hex 6C or the percent sign (%), for that

character. In this case, the value for PARM would be 6C4A6ED06AC05A5E7C4FA1.

The print speed is slower while the printer is printing forms characters than it is during normal printing.

To return to the printer's default character set, call the printer subroutine again and specify FORMS RESET.

For the 4214 Printer consider the following:

- The FORMS characters print at 10 or 15 cpi. If you specify 5 or 12 cpi, then the FORMS characters print at 10 cpi. If you specify 16.7 cpi, the FORMS characters print at 15 cpi.
- The FORMS characters print at draft quality only. If you specify TEXT YES, the default characters print rather than the FORMS box characters. If you specify TEXT NO later in the program, the box characters print again.

Transparent Option: You can use the transparent (TRANS) option to pass non-IPDS commands to the printer from RPG II and COBOL programs. Valid values for this option are 2-byte hexadecimal codes for printer commands.

Consider the following when using the transparent option:

- The data must be left-justified in the parameter field, otherwise you will get return code 42. (The specified parameter is not valid.)
- The parameter must contain an even number of characters.
- The data in the parameter field must be in EBCDIC.
- A blank (hex 40) anywhere in the parameter field indicates the end of the data stream.
- Do not use the TRANS option for vertical positioning.

IPDS Transparent Options: Two transparent options, IPTRANS and IHTRANS send an IPDS data stream to a printer.

The IPTRANS option sends page state commands to the printer. The IHTRANS option sends home state commands to the printer.

The only parameter consists of a data stream of EBCDIC-coded, 2-character hexadecimal

numbers, followed by a blank (hex 40) to denote the end of the data stream.

Consider the following when specifying transparent options:

- The data must be left-justified in the parameter field; otherwise, you will get return code 42. (The specified parameter is not valid.)
- The parameter must contain an even number of characters.
- The data in the parameter field must be in EBCDIC.
- A blank (hex 40) anywhere in the parameter field indicates the end of the data stream.
- Do not request an acknowledgment from the printer, because there is no facility provided to handle such acknowledgement, and the results are unpredictable.

Lines per Inch Option: The lines per inch (lpi) option lets you set the lines per inch to print. Valid values are 3, 4, 6, or 8.

You must specify a parameter value for this option, since there is no default.

If you are using cut forms on the 5219 or 4214 Printer, the lines per page set on the printer OCL statement in a procedure should correspond with the lines per inch value. For example, if the form length is 11 inches and the lines per inch is 8, then the lines per page should be 88.

For the best printing results, do not change the lines per inch setting more than once within a page.

For a 5262 printer, the actual number of lines per inch depends on which was set last: the lines per inch switch on the printer, or the lines per inch value in this subroutine. For example, if you set lines per inch to 6 in this subroutine but change the lines per inch switch to 8, the line spacing will be 8 lpi. The default lines per inch value is not reset for a 5262 printer at the start of the next print job. You must reset it at the end of your program.

Print Quality Options: There are two ways to set print quality on a printer, one is to use the TEXT option and the other is to use the QUALITY option.

Text Option: You can set the print quality on 4214, 5219, and IPDS printers by using the text (TEXT) option. Valid values are YES or NO. If you specify TEXT YES, the printer always prints in final quality. If you specify TEXT NO, the printer setting determines print quality.

For example, suppose that the 4214 printer is set for draft quality (using the operator control panel). If you specify TEXT YES, the printer prints in best quality. If you specify TEXT NO, however, the printer setting determines print quality, so it prints in draft quality.

For a 4214 printer that is set to 5, 15, or 16.7 cpi, the printer always prints draft quality, even if you have specified TEXT YES. If you want best quality, you must specify 10 or 12 cpi.

TEXT NO is not supported on 3812 and 5219 printers.

Quality Option: The quality (QUALITY) option is similar to the text option, except that it always sets the print quality to the specified value. It does not use the quality setting on the printer operator panel. It is used with printers that have two or more quality levels. The parameters are:

Parameter	Description
1	Best quality
2	Good quality
3	Draft quality

Page Rotation Option: The page rotation (ROTATE) option, used with the 3812 printer, rotates the printed output on the page, or prints a computer output reduction (COR), which means printing 11 by 14 inch computer output on an 8.5 by 11 inch page, rotated 90 degrees. The parameters are:

Parameter	Description
0	No rotation
90	90 degree clockwise rotation
180	180 degree clockwise rotation
270	270 degree clockwise rotation
COR	Computer output reduction

The ROTATE option overrides the rotation selection at the operator's panel on the printer. The default for this option is the printer default rotation at the printer control panel.

Printing Graphics Using Subroutines

There are two subroutines that allow you to print graphics with your RPG II or COBOL program. SUBR51 allows you to print graphics with RPG II and PRTGRC allows you to print graphics with COBOL.

Note: These subroutines can only be used with an IPDS printer.

The parameters for these options are specified in 80-byte fields, left-justified. The unused bytes should be blank. When specifying more than one parameter, separate the parameters with a comma or one or more blanks. Figure D-3 lists the available options with short descriptions. For more detailed information on the options, see the sections beginning with "Begin Filled Area Option" on page D-8.

Figure D-3. SUBR51 and PRTGRC Subroutine Options

Option	Description
BEGAREA	Begin filled area
BEGSEG	Begin graphics area (segment); must be the first option used
CHARORI	Set character orientation
CHARSIZE	Set character size
CIRCLE	Draw a circle
COLOR	Set color
ENDAREA	End filled area
ENDSEG	End graphics area (segment); must be the last option used
IGTRANS	Graphics transparent option
LINE	Draw a line
LINETYPE	Set line type
LINEWIDTH	Set line width
MARKER	Print a marker
MARKTYPE	Set marker type
PATTYPE	Set fill pattern
POSITION	Set current position
TEXT	Print option

In some options, you must specify parameters, such as x (horizontal), y (vertical), w (width), and l (length), for positioning the graphic area and the current position from which a graphics is drawn. Figure D-4 and Figure D-5 show the orientation for specifying the required measurements. The measurements are specified in *inches*.

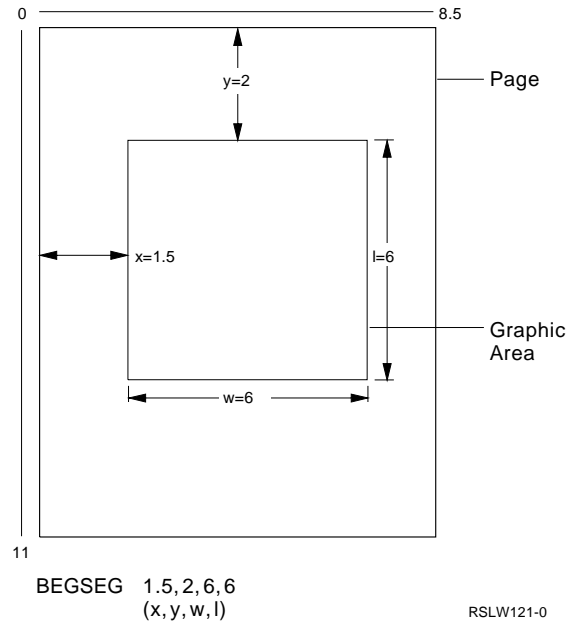


Figure D-4. Orientation for Specifying Measurements, Part 1

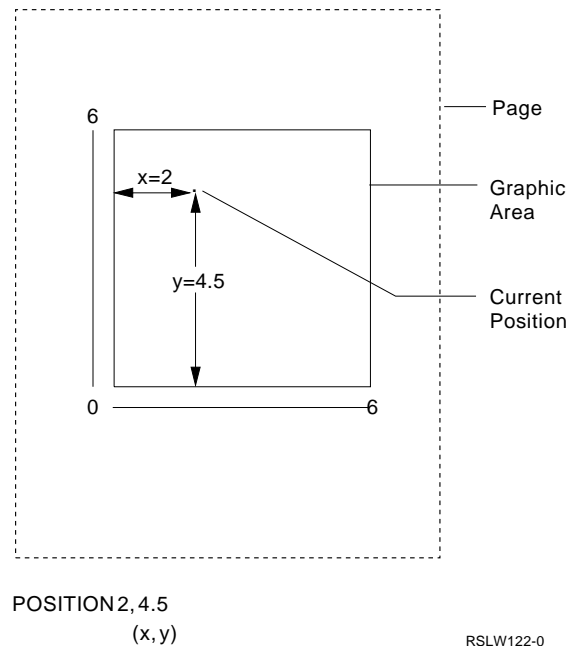


Figure D-5. Orientation for Specifying Measurements, Part 2

Some options define graphic characteristics, such as color. Once such an option is defined, it remains in effect until changed or until you use an ENDSEG option. For example, if you want several lines drawn each of the same color, you only have to specify the color option once.

The graphics area is located on the page by using the Begin graphics area (BEGSEG) option. The

current position for drawing the graphic is located by using the Position (POSITION) option.

Once you begin to define a graphic area with the BEGSEG option, only the options listed in Figure D-3 on page D-7 should be used until you have closed the current graphic area with the ENDSEG option. The BEGSEG option sets the printer for printing graphics only. Any attempt to print something else during the time between BEGSEG and ENDSEG causes a printer error. The graphics area definition is ended by the ENDSEG option.

The ENDSEG option causes all the other options to return to their default value.

Begin Filled Area Option: The begin filled area (BEGAREA) option defines an area that can be filled with the current shading function, or **fill pattern**. The filled area itself must be drawn by using the line and circle options.

There are no parameters for this option.

You can draw approximately 36 lines in the filled area specified by BEGAREA and ENDAREA (end filled area). You can use a maximum of 221 bytes to define the filled area specified by this option and ENDAREA. If you exceed the 221-byte maximum while defining the area, or if an ENDAREA or ENDSEG (End graphics area) option is missing, an error message issues.

Do not use the following options between the BEGAREA and ENDAREA. If you use these options within a filled area, you will get an error.

- Character orientation (CHARORI)
- Character size (CHARSIZE)
- Marker (MARKER)
- Marker type (MARKTYPE)
- Fill pattern type (PATTYPE) (See note)
- Text (TEXT)

Note: You must use the PATTYPE option just prior to the BEGAREA option for the fill area being specified.

Begin Graphics Area (Segment)

Option: The begin graphics area (BEGSEG) option defines the size and location of the area containing the graphic output. *It must be the first option used.* The parameters are:

Parameter Description

x	The distance in inches from the left edge of the page to the left edge of the graphics area on the page. The default is 0.
y	The distance in inches from the top of the page to the top edge of the graphics area on the page. The default is 0.
w	The width of the graphics area in inches. The default is 8.5 inches.
l	The length of the graphics area in inches. The default is 11 inches.

Consider the following when using the begin graphics area option:

- Parameters x, y, w, and l can be specified in decimal numbers of any combination of xx.xx, where x is any number from 0 through 9.
- All xx.xx values should be less than 22.75.
- If the size of the graphic is larger than the size of the page, an error message issues.
- The printer may not print graphics that are less than 0.1 inch from the edge of the graphic area or that extend beyond the graphic area. No error message issues.

Character Orientation Option: The set character orientation (CHARORI) option defines the angle on the page that the characters defined by the text option are printed. The parameters are:

Parameter Description

0	Characters print as normal, across the page. This is the default.
90	Characters are rotated 90 degrees counterclockwise from normal.
180	Characters print upside down across the page.
270	Characters are rotated 270 degrees counterclockwise from normal.

Character Size Option: The character size (CHARSIZE) option defines the size of the characters printed by the text option. The parameters are:

Parameter Description

- w** The width of the character in inches. The default is 0.14.
- h** The height of the character in inches. The default is 0.13.

You can specify parameters w and h in decimal numbers of any combination of xx.xx, where x is any number from 0 through 9.

Circle Option: Use the circle (CIRCLE) option to draw a circle with the current position as its center. See “Position Option” on page D-10, for information on the position option. The only parameter is r, which defines the radius of the circle in inches. The default is 1.

Consider the following when using the circle option:

- The characteristics for the circle are set by the color, line type, and line width options.
- You can specify parameter r in decimal numbers of any combination of xx.xx, where x is any number from 0 through 9.
- If the circle is too big for the BEGSEG option, only the part within the graphics area will print.

Color Option: The color (COLOR) option is used to select the color for the graphic to be drawn. The parameter is one of the following:

- BLACK
- RED
- BLUE
- TURQ (turquoise, also cyan)
- BROWN
- WHITE (no color; see note)
- GREEN
- YELLOW
- PINK (magenta)

The default is BLACK.

Consider the following when using the color option:

- WHITE will only appear if printed over a non-white graphics background.
- If the specified color cannot be printed by the ribbon currently on the printer, black is printed.

End Filled Area Option: Use the end filled area (ENDAREA) option to identify the end of the filled area as defined by the begin filled area option. The area to be filled should be closed. If an area is not closed, the printer will attempt to draw a line to close it. There are no parameters for this option.

End Graphics Area (Segment) Option: Use the end graphics area (ENDSEG) option to identify the end of the graphic defined by the begin graphics area option. There are no parameters for this option. *It must be the last option used* after all the graphics have been specified for this area.

Graphic Transparent Option: Use the graphic transparent (IGTRANS) option to send a graphic drawing order that is not supported by the other options to a printer. The parameter is a drawing order of EBCDIC coded, 2-character hexadecimal numbers, followed by a blank (hex 40) to denote the end of the drawing order.

Consider the following when using the graphic transparent option:

- The data must be left-justified in the parameter field. Otherwise, an error issues.
- The parameter must contain an even number of characters.
- The data in the parameter field must be in EBCDIC.
- A blank (hex 40) anywhere in the parameter field indicates the end of the drawing orders.








Line Option: Use the line (LINE) option to draw a line from the current position to a specified position. See “Position Option” on page D-10 for more information on the position option. The specified position is defined by the parameters x and y, which define the coordinate, in inches, where the line is to be drawn. The horizontal measurement is x, and the vertical measurement is y. See Figure D-4 on page D-7 and Figure D-5 on page D-7 for an example of how parameters x and y are measured. The default is 0,0.

Note: The line option updates the current position to the coordinates for the end of the line.

You can specify parameters x and y in decimal numbers of any combination of xx.xx, where x is any number from 0 through 9.

The characteristics for the line are set by the color, line type, and line width options.

Line Type Option: The line type (LINETYPE) option defines the line type or style for the line and circle options. The parameter is one of the parameters in the following figure. The default is SOLID.

Parameter
 SOLID
 DOT
 SDASH (Short Dash)
 DASHDOT (Dash Dot)
 LDASH (Long Dash)
 DASHDD (Dash Double Dot)
 DBLDOT (Double Dot)
INVIS (Invisible)











RSLW123-0

Line Width Option: Use the line width (LINEWDTH) option to set the width of the line or circle to be drawn. The parameter is either NORMAL or THICK. The default is NORMAL.

Marker Option: Use the marker (MARKER) option to print a marker at the current position. See "Position Option" for more information on the position option. There are no parameters for this option.

The characteristics for the marker are set by the color and marker type options.

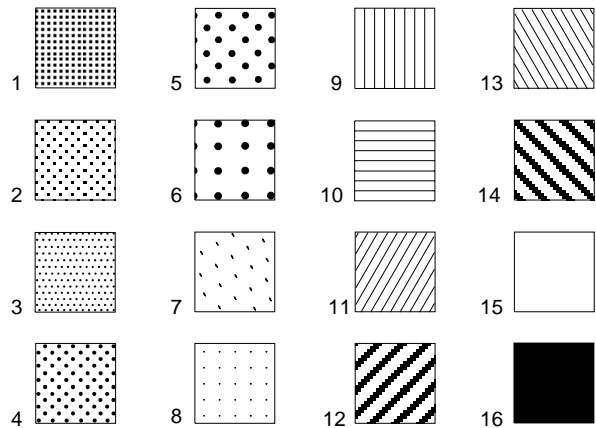
Marker Type Option: The marker type (MARKTYPE) option defines the marker type or style for the marker option. The parameter is one of the parameters in the following figure. The default is CROSS.

Parameter
 CROSS
 PLUS
 DIAMOND
 SQUARE
 6STAR (6 Pointed Star)
 8STAR (8 Pointed Star)
 FDIAMOND (Filled Diamond)
 FSQUARE (Filled SQUARE)
 DOT
 CIRCLE

RSLW124-0

Pattern Type Option: Use the pattern type (PATTYPE) option to select the fill pattern for filling the space defined by circle and line options between a BEGAREA and ENDAREA.

The fill patterns are shown in the following figure. Select a parameter value from 1 to 16. The default is 16 (solid).



RSLW003-0

Position Option: Use the position (POSITION) option to set the current position for the circle, line, marker, and text options. The current position is set by the parameters x and y, which are defined in inches. The horizontal measurement is x, and the vertical measurement is y. The default is 0,0. You must specify a parameter value. See Figure D-4 on page D-7 and

Figure D-5 on page D-7 for an example of how parameters x and y are measured.

Parameters x and y can be specified in decimal numbers of any combination of xx.xx, where x is any number from 0 through 9.

Text Option: The text (TEXT) option is used to print the text defined in the parameter field at the current position. See “Position Option” on page D-10 for information on the position option. The characteristics for the text are set by the color, character orientation, and character size options. There are no options provided to change print quality or character style (font).

Printing Forms and Graphs

The forms generation utility reads the options and parameters as if they were the options used in SUBR51 and PRTGRC subroutines. However, the options and parameters are in a library source member. The utility either prints the graphics output on a printer or saves the information in a graphics object file.

The utility reads all of the records from the source member until it encounters the ENDSEG option. Everything in the source member after the ENDSEG option is ignored.

Note: You can use this function only with an IPDS printer.

The utility provides two procedures: one for printing graphics (PRTGRAPH) and one for building a graphics object file (BLDGRAPH).

Print Graphic Procedure: Use the print graphic (PRTGRAPH) procedure to print the form or graphics data by itself.

To print the graphics data from a disk file, the format is:

```
PRTGRAPH prtId,FILE,graphics file name,width
```

prtId

Is the ID of the IPDS printer to print the form or graphics file. The default is the current session printer.

FILE

Specifies that the graphics data is in a disk file.

graphics file name

Specifies the name of the graphics object file to be printed.

width

Specifies the width, in inches, of the graphics area to be printed. This parameter cannot contain more than 5 characters specified in decimal numbers. The specified value cannot be more than 45.50. For example, if the area to be printed is 13 inches, you can specify either 13 or 13.0 for this parameter. If you do not specify this parameter, 13.2 is assumed.

To print the graphics data from a library source member, the format is:

```
PRTGRAPH prtId,SOURCE,membername,libraryname
```

prtId

Is the ID of the IPDS printer to print the form or graphics file. The default is the current session printer.

SOURCE

Indicates that the data is in a library source member.

membername

Is the name of the library source member that contains the graphic options and parameters. Each record in the source member must be in the following format:

```
Columns 1 through 8, the option name  
Column 9, leave blank  
Columns 10 through 89, the parameters  
for the option
```

Note: This utility supports comment records. Place an asterisk (*) in column 1 of the source records containing the comment.

libraryname

Is the library name that contains the source member. If not specified, only the current library is searched.

Build Graphics Object File Procedure:

Use the build graphics object file (BLDGRAPH) procedure to build a graphics object file containing the graphic options and parameters for printing with other data print files.

The format for this procedure is:

```
BLDGRAPH membername,libraryname,filename
```

membername

Is the name of the library source member that contains the graphic options and parameters. Each record in the source member must be in the following format:

Columns 1 through 8, the option name
 Column 9, leave blank
 Columns 10 through 89, the parameters for the option

libraryname

Is the library name that contains the source member. If not specified, only the current library is searched.

filename

Is the name of the graphics object file to be created.

Note: This utility supports comment records. Place an asterisk (*) in column 1 of the source records containing the comment.

See section "Sample Form" on page D-14 for a sample source member.

Printing Bar Codes

This bar code function provides two subroutines for printing bar codes: SUBR52 for RPG II programs and PRTBAR for COBOL programs.

Note: You can use this function only with an IPDS printer.

The options and parameters valid for this function are described in Figure D-6.

Figure D-6 (Page 1 of 2). Valid Options and Parameters for the Bar Code Function

Option	Description				
BARCODE	<p>This option causes the data specified in the parameter field to print as a bar code. The characteristics are set by the BARSIZE, BARTYPE, HRI, and POSITION options. The following parameter is available:</p> <table border="1"> <thead> <tr> <th>Parameter</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>Bar code data</td> <td>The length of the data is from the beginning of the parameter field to the last nonblank character. There is no default. Data must be specified in the parameter field. The BARTYPE option specifies the type of data allowed.</td> </tr> </tbody> </table>	Parameter	Description	Bar code data	The length of the data is from the beginning of the parameter field to the last nonblank character. There is no default. Data must be specified in the parameter field. The BARTYPE option specifies the type of data allowed.
Parameter	Description				
Bar code data	The length of the data is from the beginning of the parameter field to the last nonblank character. There is no default. Data must be specified in the parameter field. The BARTYPE option specifies the type of data allowed.				
BARSIZE	<p>Set bar code size. This option is used to specify the height of the bar code in inches. The following parameter is available:</p> <table border="1"> <thead> <tr> <th>Parameter</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>A</td> <td>The height of the bar code in inches. The default is 1 inch. Specify parameter A in decimal numbers of any combination of xx.xx, where x is any number from 0 through 9.</td> </tr> </tbody> </table>	Parameter	Description	A	The height of the bar code in inches. The default is 1 inch. Specify parameter A in decimal numbers of any combination of xx.xx, where x is any number from 0 through 9.
Parameter	Description				
A	The height of the bar code in inches. The default is 1 inch. Specify parameter A in decimal numbers of any combination of xx.xx, where x is any number from 0 through 9.				

Figure D-6 (Page 2 of 2). Valid Options and Parameters for the Bar Code Function

Option	Description																												
BARTYPE	<p>Bar code type. This option sets the type of bar code printed by a bar code option. The following parameters are available:</p> <table border="1"> <thead> <tr> <th>Parameter</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>MSI</td> <td>Modified Plessey, up to 50 digits per bar code. The range of characters allowed is 0 through 9. Two modulus 10 check digits are generated.</td> </tr> <tr> <td>UPCA</td> <td>Eleven digits per bar code. The range of characters allowed is 0 through 9. One check digit is generated.</td> </tr> <tr> <td>UPCE</td> <td>Ten digits per bar code. Only six digits are encoded into bars. The range of characters allowed is 0 through 9. One check digit is generated.</td> </tr> <tr> <td>UPC2</td> <td>If a UPC two-digit add on is used, it must follow a UPCA or UPCE bar code. There can only be two digits per bar code. The range of characters allowed is 0 through 9. No check digit is generated.</td> </tr> <tr> <td>UPC5</td> <td>If a UPC five-digit add-on is used, it must follow a UPCA or UPCE bar code. There can only be five digits per bar code. The range of characters allowed is 0 through 9. No check digit is generated.</td> </tr> <tr> <td>EAN8</td> <td>Seven digits per bar code. The range of characters allowed is 0 through 9. One check digit is generated.</td> </tr> <tr> <td>EAN13</td> <td>Twelve digits per bar code. The range of characters allowed is 0 through 9. One check digit is generated.</td> </tr> <tr> <td>EAN2</td> <td>If a EAN two-digit add-on is used, it must follow an EAN8 or EAN13 bar code. There can only be two digits per bar code. The range of characters allowed is 0 through 9. No check digit is generated.</td> </tr> <tr> <td>EAN5</td> <td>If a EAN five-digit add-on is used, it must follow an EAN8 or EAN13 bar code. There can only be five digits per bar code. The range of characters allowed is 0 through 9. No check digit is generated.</td> </tr> <tr> <td>25INDUS</td> <td>A 2 of 5 industrial code of up to 50 digits per bar code. The range of characters allowed is 0 through 9. One check digit is generated.</td> </tr> <tr> <td>25MATRIX</td> <td>A 2 of 5 matrix code of up to 50 digits per bar code. The range of characters allowed is 0 through 9. One check digit is generated.</td> </tr> <tr> <td>25INTRL</td> <td>A 2 of 5 interleaved code of up to 50 digits per bar code. The range of characters allowed is 0 through 9. One check digit is generated.</td> </tr> <tr> <td>CODE39</td> <td>A 3 of 9 code of up to 50 characters per bar code. The range of characters allowed is 0 through 9, A through Z, -.\$/= % and (blank). No check digit is generated.</td> </tr> </tbody> </table> <p>The default is UPCA.</p>	Parameter	Description	MSI	Modified Plessey, up to 50 digits per bar code. The range of characters allowed is 0 through 9. Two modulus 10 check digits are generated.	UPCA	Eleven digits per bar code. The range of characters allowed is 0 through 9. One check digit is generated.	UPCE	Ten digits per bar code. Only six digits are encoded into bars. The range of characters allowed is 0 through 9. One check digit is generated.	UPC2	If a UPC two-digit add on is used, it must follow a UPCA or UPCE bar code. There can only be two digits per bar code. The range of characters allowed is 0 through 9. No check digit is generated.	UPC5	If a UPC five-digit add-on is used, it must follow a UPCA or UPCE bar code. There can only be five digits per bar code. The range of characters allowed is 0 through 9. No check digit is generated.	EAN8	Seven digits per bar code. The range of characters allowed is 0 through 9. One check digit is generated.	EAN13	Twelve digits per bar code. The range of characters allowed is 0 through 9. One check digit is generated.	EAN2	If a EAN two-digit add-on is used, it must follow an EAN8 or EAN13 bar code. There can only be two digits per bar code. The range of characters allowed is 0 through 9. No check digit is generated.	EAN5	If a EAN five-digit add-on is used, it must follow an EAN8 or EAN13 bar code. There can only be five digits per bar code. The range of characters allowed is 0 through 9. No check digit is generated.	25INDUS	A 2 of 5 industrial code of up to 50 digits per bar code. The range of characters allowed is 0 through 9. One check digit is generated.	25MATRIX	A 2 of 5 matrix code of up to 50 digits per bar code. The range of characters allowed is 0 through 9. One check digit is generated.	25INTRL	A 2 of 5 interleaved code of up to 50 digits per bar code. The range of characters allowed is 0 through 9. One check digit is generated.	CODE39	A 3 of 9 code of up to 50 characters per bar code. The range of characters allowed is 0 through 9, A through Z, -.\$/= % and (blank). No check digit is generated.
Parameter	Description																												
MSI	Modified Plessey, up to 50 digits per bar code. The range of characters allowed is 0 through 9. Two modulus 10 check digits are generated.																												
UPCA	Eleven digits per bar code. The range of characters allowed is 0 through 9. One check digit is generated.																												
UPCE	Ten digits per bar code. Only six digits are encoded into bars. The range of characters allowed is 0 through 9. One check digit is generated.																												
UPC2	If a UPC two-digit add on is used, it must follow a UPCA or UPCE bar code. There can only be two digits per bar code. The range of characters allowed is 0 through 9. No check digit is generated.																												
UPC5	If a UPC five-digit add-on is used, it must follow a UPCA or UPCE bar code. There can only be five digits per bar code. The range of characters allowed is 0 through 9. No check digit is generated.																												
EAN8	Seven digits per bar code. The range of characters allowed is 0 through 9. One check digit is generated.																												
EAN13	Twelve digits per bar code. The range of characters allowed is 0 through 9. One check digit is generated.																												
EAN2	If a EAN two-digit add-on is used, it must follow an EAN8 or EAN13 bar code. There can only be two digits per bar code. The range of characters allowed is 0 through 9. No check digit is generated.																												
EAN5	If a EAN five-digit add-on is used, it must follow an EAN8 or EAN13 bar code. There can only be five digits per bar code. The range of characters allowed is 0 through 9. No check digit is generated.																												
25INDUS	A 2 of 5 industrial code of up to 50 digits per bar code. The range of characters allowed is 0 through 9. One check digit is generated.																												
25MATRIX	A 2 of 5 matrix code of up to 50 digits per bar code. The range of characters allowed is 0 through 9. One check digit is generated.																												
25INTRL	A 2 of 5 interleaved code of up to 50 digits per bar code. The range of characters allowed is 0 through 9. One check digit is generated.																												
CODE39	A 3 of 9 code of up to 50 characters per bar code. The range of characters allowed is 0 through 9, A through Z, -.\$/= % and (blank). No check digit is generated.																												
HRI	<p>Set human readable interpretation (HRI). This option is used to specify whether or not the HRI will print. The following parameters are available:</p> <table border="1"> <thead> <tr> <th>Parameter</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>YES</td> <td>The HRI is printed.</td> </tr> <tr> <td>NO</td> <td>The HRI is not printed.</td> </tr> </tbody> </table> <p>The default is NO.</p>	Parameter	Description	YES	The HRI is printed.	NO	The HRI is not printed.																						
Parameter	Description																												
YES	The HRI is printed.																												
NO	The HRI is not printed.																												
POSITION	<p>The coordinate in inches from the upper left corner of the page to the upper left corner of the bar code. There is no default for this option. A nonzero parameter value must be specified. The following parameter is available:</p> <table border="1"> <thead> <tr> <th>Parameter</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>x,y</td> <td>Specify parameters x and y in decimal numbers of any combination of xx.xx, where x is any number from 0 through 9.</td> </tr> </tbody> </table>	Parameter	Description	x,y	Specify parameters x and y in decimal numbers of any combination of xx.xx, where x is any number from 0 through 9.																								
Parameter	Description																												
x,y	Specify parameters x and y in decimal numbers of any combination of xx.xx, where x is any number from 0 through 9.																												

Sample Form

The following is a sample source member that you can use with the PRTGRAPH procedure to create a form:

```
BEGSEG 0,0,8.5,11
CHARSIZE .10,.10
*
* DRAW BIG OUTSIDE SQUARE
*
POSITION .5,1.5
LINE 8,1.5
LINE 8,10.5
LINE .5,10.5
LINE .5,1.5
*
* DRAW 2 SQUARES IN UPPER RIGHT CORNER
*
PATTYPE 7
BEGAREA
POSITION 5,9.5
LINE 7,9.5
LINE 7,10.25
LINE 5,10.25
LINE 5,9.5
ENDAREA
POSITION 5,8.5
LINE 7,8.5
LINE 7,9.25
LINE 5,9.25
LINE 5,8.5
POSITION 5,9
LINE 7,9
*
* DRAW ALL THICK HORIZONTAL LINES
*
LINEWIDTH THICK
POSITION .5,6.3
LINE 7,6.3
POSITION .5,5.72
LINE 7,5.72
POSITION .5,3
LINE 7,3
POSITION 6,2.75
LINE 7,2.75
POSITION 4.5,2
LINE 7,2
*
* DRAW REST OF HORIZONTAL LINES
*
LINEWIDTH NORMAL
POSITION .5,5.55
LINE 7,5.55
POSITION .5,5.38
LINE 7,5.38
POSITION .5,5.21
LINE 7,5.21
POSITION .5,5.04
LINE 7,5.04
POSITION .5,4.87
LINE 7,4.87
POSITION .5,4.7
LINE 7,4.7
POSITION .5,4.53
LINE 7,4.53
POSITION .5,4.36
LINE 7,4.36
POSITION .5,4.19
LINE 7,4.19
POSITION .5,4.02
LINE 7,4.02
POSITION .5,3.85
LINE 7,3.85
POSITION .5,3.68
LINE 7,3.68
POSITION .5,3.51
LINE 7,3.51
POSITION .5,3.34
LINE 7,3.34
POSITION .5,3.17
LINE 7,3.17
POSITION 4.5,2.75
LINE 6,2.75
POSITION 4.5,2.44
LINE 7,2.44
*
* DRAW REST OF VERTICAL LINES
*
POSITION 4.5,6.3
LINE 4.5,2
POSITION 5,6.3
LINE 5,3
POSITION 6,6.3
LINE 6,2
POSITION 7,6.3
LINE 7,2
POSITION 5.80,5.72
LINE 5.80,3
POSITION 6.80,5.72
LINE 6.80,2
*
* ALL TEXT FOR THE FORM FOLLOWS
*
CHARSIZE .20,.20
POSITION 1.5,9.5
TEXT Wholesale
POSITION 1.1,9.35
TEXT Tire & Supply
POSITION 1.65,9.2
TEXT Company
CHARSIZE .1,.1
POSITION 1.4,8.7
TEXT 175 R14 Tire Drive
POSITION 1.65,8.55
```



```

TEXT      Radial City, MI
POSITION  5.4,9.95
TEXT      10% Discount
POSITION  5.1,9.8
TEXT      On All Orders Over
POSITION  5.6,9.65
TEXT      $500.00
POSITION  5.3,9
TEXT      Account Number
POSITION  4.25,7.9
TEXT      Bill To:
POSITION  1.88,5.9
TEXT      Item Description
POSITION  4.63,5.9
TEXT      Qty
POSITION  5.00,5.9
TEXT      Unit Price
POSITION  6.19,5.9
TEXT      Total
POSITION  4.52,2.82
TEXT      Subtotal
POSITION  4.52,2.54
TEXT      Less Discount
POSITION  4.52,2.15
TEXT      Total
POSITION  6.02,2.82
TEXT      $

```

```

POSITION  6.02,2.54
TEXT      $
POSITION  6.02,2.15
TEXT      $
ENDSEG

```

Use the following steps to print the sample graph:

1. The form generated by this source member requires a page that is 11 inches long. To ensure the proper page length, enter the following procedure:

```
PRINT ,66,6
```

2. Print the form by entering the following procedure:

```
PRTGRAPH prtId,SOURCE,DEMOFORM,#LIBRARY
```

where prtId is the ID for an IPDS printer.

You may want to study this listing and compare it to the form it produced. You can use this source member to learn how to make your own form.

Figure D-7 on page D-16 shows the sample form created by the sample source member using the PRTGRAPH procedure.

Appendix E. Security Considerations for the System/36 Environment

This appendix lists System/36 procedures, commands, and OCL statements, along with any special authorities to AS/400 objects or commands needed to support them. Authorities are listed in terms of the generic *ALL, *CHANGE, and *USE, and not by the specific object authorities.

- The utility control statements (if any) used to pass information to the utility programs
- The OS/400 commands run by the utility programs to perform the requested function
- The authorities to objects used to perform the request

System/36 Procedures

Figure E-1 describes authorities required for System/36 procedures on the AS/400 system, as follows:

- System/36 procedures and the utility programs called by the procedures

In addition to the special authorities listed, you need *USE authority for the following:

- Utility programs called by System/36 procedures
- OS/400 commands run by the utility programs
- Library QSSP, including the QS36PRC source file in library QSSP
- The ##MSG1 and ##MSG2 message files in library QSSP

Figure E-1 (Page 1 of 19). Authorities Required for System/36 Procedures on the AS/400 System

System/36 Procedure	Utilities Called	OS/400 Commands	Additional Authorities Required
#ERR	\$CPPE/#ERR		*USE, to the message files specified on the USER1 and USER2 keywords of the MEMBER OCL statement. *USE, to the library specified on the LIBRARY keyword on the MEMBER OCL statement.
#FORMAT	\$SFGR/LOADMBR \$SFGR/INOUT \$SFGR/DELETE \$SFGR/END	CRTDSPF, CPYF and RMVM	*ALL, to the display file if it exists and REPLACE is specified. *CHANGE, to the library to contain the display file. *USE, to the library containing the QS36SRC source file. *USE, to the source file QS36SRC.
AUTO	\$UASF/SPOOL \$UASC QEXSETLB #RPDD #AUTO	CHGS36MSGL	*USE, to library #RPGLIB and all objects therein. *USE, for message file QRP2MSGE in library QSSP. *USE, to message file #RP#CPL1 and #RP#CPL2 in library #RPGLIB. *USE, to the library containing the source program. *USE, to the source file QS36SRC in the library containing the source program. *CHANGE, to the library to contain the compiled program.

Figure E-1 (Page 2 of 19). Authorities Required for System/36 Procedures on the AS/400 System

System/36 Procedure	Utilities Called	OS/400 Commands	Additional Authorities Required
BLDFILE	\$FBLD/FILE	CRTPF	*CHANGE, to the files library. *CHANGE, to the existing file for date-differentiated files.
BLDINDEX	\$FBLD/FILE	CRTL F	*CHANGE, to the files library. *ALL, to the parent file.
BLDGRAPH	\$DPGR \$DPGR2		*USE, to the printer device description and device file. *USE, to the physical file.
BLDLIBR	\$MAINT/ALLOCATE	CRTL B	
BLDMENU	\$MAINT/COPY \$MAINT/DELETE \$MGBLD/MGBLD \$SFGR/LOADMBR \$SFGR/INOUT \$SFRR/CREATE \$SFGR/END \$BMENU	CRTMSGF ADDMSGD CRTDSPF CRTSRCPF CPYF RMVM	*USE, to source file QS36SRC. *USE, to source library. *ALL, to the display file if it exists. *ALL, to the message file if it exists. *CHANGE, to the library to contain the display file.
CATALOG	\$LABEL/DISPLAY	CHKDKT DSPDKT CHKTAP DSPTAP DSPLIB DSPOBJD DSPFD	*USE, to the device description and the device file for the diskette or tape drive, if specified. *USE, to the device description and print file for the output printer. *USE, to the System/36 files library. *USE, to the file being cataloged. Note: You must be enrolled in the system distribution directory to list any folders.
CHNGEMEM	\$MAINT/CHANGE	RNM OBJ RNMM CHGS36SRCA CHGS36PRCA CHGS36PGMA	*ALL, to the QS36PRC or QS36SRC files if procedure or source members are to be renamed. *ALL, to any other object to be renamed. *CHANGE, to the library that contains the object to be renamed. *CHANGE, to the QS36PRC or QS36SRC files and *USE, to the library, if the subtype or reference number of a procedure or source member is to be changed.

Figure E-1 (Page 3 of 19). Authorities Required for System/36 Procedures on the AS/400 System

System/36 Procedure	Utilities Called	OS/400 Commands	Additional Authorities Required
CGU	QCGPGM QEXSETLB	CHGS36MSGL	<p>*USE, for one of the following:</p> <ul style="list-style-type: none"> *IGCTBL object QIGC2424 *IGCTBL object QIGC2424K *IGCTBL object QIGC2424C *IGCTBL object QIGC24242S <p>If any of the following exist, you must have at least *USE authority to one of the following:</p> <ul style="list-style-type: none"> *IGCTBL object QIGC3232 *IGCSRT object QCGMSTR *IGCSRT object QCGACTV *IGCSRT object QCGACTVK *IGCSRT object QCGMSTRC *IGCSRT object QCGACTVC <p>Certain functions from the CGU Main Menu require the following authorities:</p> <ul style="list-style-type: none"> *CHANGE, to *IGCTBL objects. *ALL, to *IGCSRT objects. <p>*USE, to the libraries #CGULIB and QPDA.</p>
CLRPF		CLRPFM CHGS36MSGL	*ALL, to the DB file
COBOL	QEXSETLB #CB00	CHGS36MSGL	<p>*USE, for library #COBLIB.</p> <p>*USE, for message files #CB#M1, #CB#M2 and QSBLMSG in library #COBLIB.</p> <p>*USE, for programs in #COBLIB.</p> <p>*USE, to the library containing the source program.</p> <p>*USE, to the source file QS36SRC in the library containing the source program.</p> <p>*CHANGE, to the library to contain the compiled program.</p>
COBOLC	QEXSETLB \$UASF/SPOOL #CB00 \$UASC	CHGS36MSGL	See the COBOL procedure.
COBSDA	QEXSETLB	CHGS36MSGL	<p>*USE, for message file #CB#M1 in library #COBLIB.</p> <p>See the SDA procedure for additional authorities.</p>
COBSEU		CHGS36MSGL	<p>*USE, for message file #CB#M1 in library #COBLIB.</p> <p>See the SEU procedure for additional authorities.</p>

Figure E-1 (Page 4 of 19). Authorities Required for System/36 Procedures on the AS/400 System

System/36 Procedure	Utilities Called	OS/400 Commands	Additional Authorities Required
COPYDATA	\$COPY/COPYFILE	CPYF	<p>*USE, to the DB input file.</p> <p>*CHANGE, to the DB output file.</p> <p>*CHANGE, to the System/36 files library.</p> <p>Note: Load-to-old processing using OCL and utility control statements requires *ALL authority to the DB output file.</p>
COPYI1	\$DUPRD/COPYI1	DUPDKT	<p>*USE, to the device description and device file for the diskette drive.</p>
COPYPRT	\$UASF/SPOOL	CRTPF CPYSPLF DLTSPFL RLSSPLF	<p>*JOBCTL or *SPLCTL special authority to use the SYSTEM keyword.</p> <p>*USE, to the output queue that contains the spool file to be copied.</p> <p>*CHANGE, to the files library.</p> <p>*CHANGE, to the output queue that contains the spool file to be canceled or released. If you created the spool file, or you have *JOBCTL or *SPLCTL special authority and the output queue was created with operator control right, you do not need *CHANGE authority.</p>
COPYPRT	\$UASC	CPYF DSPSPLF	<p>*USE, to the data file that contains the copied spool files.</p> <p>*USE, to the files library.</p> <p>*USE, to the printer device description and print file if the print option is selected.</p>
CREATE	\$MGBLD/MGBLD	CRTMSGF ADDMSGD	<p>*USE, to source file containing source member.</p> <p>*USE, to library containing source member.</p> <p>*ALL, to the message file named in the source.</p> <p>*CHANGE, to the message file library.</p>
DEFSUBD		CRTFLR DLTDL0 WRKFLR CHGS36MSGL	<p>*USE, for the ##MSG1 message file in library #LIBRARY.</p>
DELETE	\$DELET/SCRATCH \$DELET/REMOVE	DLTTF DLTLIB RMVM INZDKT CLRDKT DLTDKTLBL	<p>*ALL, to the file, library or member.</p> <p>Security officer authority is required for LABEL-ALL, UNIT-F1.</p> <p>*USE, to the device description and device file of the diskette drive, if specified.</p> <p>Note: You must be enrolled in the system distribution directory to delete a folder.</p>

Figure E-1 (Page 5 of 19). Authorities Required for System/36 Procedures on the AS/400 System

System/36 Procedure	Utilities Called	OS/400 Commands	Additional Authorities Required
DFU	#DFMP QEXSETLB	STRDFU CHGS36MSGL	*USE, to library QPDA. *USE, to library QSYS. *USE, to library QSSP. *USE, to library #DFULIB and all objects therein. *USE, to all objects having a name starting with QDF or QDL in libraries QPDA, QSYS, and QSSP. *USE, to message file #DF#MG in library #DFULIB. *USE, to library #DSULIB. *USE, to message file #ED#M1, #ED#M2, and #ED#M3 in library #DSULIB.
DISPLAY	\$COPY/COPYFILE	CPYF DSPPFM	*USE, to the DB input file. *USE, to the System/36 files library. *USE, to the device description and print file for the output printer.
DSPLOCK		WRKOBJLCK CHGS36MSGL	
DSPSYS		WRKSYSSTS CHGS36MSGL	
DSU	QEXSETLB #EDLIS	CHGS36MSGL	*USE, to library #DSULIB *USE, to message file #ED#ML, #ED#M2 and #EDM3 in library #DSULIB
EM3270	#EMFP #EMAD #EM9D	STREML3270 STRPRTEML ENDPRTEML SBMJOB	Authority to the emulation location (user-specified), the print file (QPMPRTF), and the emulation devices associated with the location. *USE, to #EM#M1 System/36 messages.
ENTER			See the DFU procedure.
ENTER#			See the DFU procedure.
EP3270	#ESFP #ESEP	STREML3270 STRPRTEML ENDPRTEML EJTEMLOUT SBMJOB	Authority to the emulation location (user-specified), and the emulation devices associated with the location. *USE, to #ES#M1 System/36 messages.
ERR			See the #ERR procedure.
ES3270	#ESFP #ESAD #ESPI	STREML3270 STRPRTEML ENDPRTEML EJTEMLOUT SBMJOB	Authority to the emulation location (user-specified), the print file (QPMPRTF), and the emulation devices associated with the location. *USE, to #ES#M1 System/36 messages. *USE, to program #ESFP.

Figure E-1 (Page 6 of 19). Authorities Required for System/36 Procedures on the AS/400 System

System/36 Procedure	Utilities Called	OS/400 Commands	Additional Authorities Required
EXTRACT	\$COPY/COPYFILE	CPYF DSPPFM	*USE, to the DB input file. *CHANGE, to the DB output file, if any. *USE, to the System/36 files library for the display/print operation. *CHANGE, to the System/36 files library for the copy operation. *USE, to the device description and print file for the output printer, for display/print operation.
FLIB		ADDLIBLE	*USE, for the specified library.
FORMAT	\$SFGR/LOADMBR \$SFGR/CREATE \$SFGR/ADD \$SFGR/UPDATE \$SFGR/END	CRTDSPF CRTSRCPF CPYF RMVM	*USE, to source file QS36SRC. *USE, to the library containing the QS36SRC source file. *ALL, to the display file if it exists and REPLACE is specified. CHANGE, to the library to contain the display file.
FROMLIBR	\$MAINT/COPY	CRTSAVF SAVS36LIBM SAVOBJ	*USE, to the diskette or tape device descriptions and device files, if I1, T1, T2 or TC are specified. *CHANGE, to the files library, if F1 is specified. *USE, to the QS36PRC and QS36SRC files, if procedures or source members are to be copied. *USE, to any other object to be copied. *USE, to the library from which the members are to be copied. Note: If you have save system (*SAVSYS) special authority, you can copy the object.

Figure E-1 (Page 7 of 19). Authorities Required for System/36 Procedures on the AS/400 System

System/36 Procedure	Utilities Called	OS/400 Commands	Additional Authorities Required
IDDUDCT	#DSIN	WRKDTADCT	<p>*USE to message file #DS#1 in library QSSP.</p> <p>*USE, to display file QDIDDU in library QSYS for procedures IDDUDCT, IDDUDFN, and IDDULINK.</p> <p>Depending on the options and function keys requested for IDDUDCT, IDDUDFN, and IDDUDISK, there may be additional authorities required during the interactive session for the following commands:</p> <p style="padding-left: 40px;">UPDDTA WRKDTADCT WRKDTADFN WRKDBFIDD EDTOBJAUT DSPDTADCT CRTDTADCT DLTDTADCT CRTLIB CRTPF</p> <p>You must have appropriate authority for the files, libraries, and data dictionaries referenced for all of these procedures.</p>
IDDUDFN		WRKDTADFN	See the IDDUDCT procedure.
IDDUDISK		WRKDBFIDD	See the IDDUDCT procedure.
IDDULINK		LNKDTADFN	See the IDDUDCT procedure.
IDDUPRT		DSPDTADCT	See the IDDUDCT procedure.
IGC	QEXIGCP		
INIT	\$INIT/UIN \$INIT/VOL	CLRDKT INZDKT RNMDKT CHKDKT	*USE, to the device file and device description for the diskette drive.
INQUIRY			See the DFU procedure.
INQUIRY#			See the DFU procedure.
ITF	QY1ITF36	STRITF CHGS36MSGL	<p>*USE, to the files library.</p> <p>*ALL, to files QS36SRC and QS36PRC.</p> <p>*USE, for folders and documents.</p> <p>*USE, to message file QSSPMSG in library QSSP.</p>
JOBSTR	\$MAINT/COPY \$MAINT/DELETE	CRTSRCPF RSTS36LIBM RSTOBJ CPYF RNMM SBMJOB	<p>*USE, to the procedure library.</p> <p>*CHANGE, to the QS36PRC source file if procedure is to be saved; *ALL if procedure is not to be saved.</p> <p>*USE, to the diskette or tape device descriptions and device files.</p> <p>*CHANGE, to the job queue QBATCH in library QGPL.</p>

Figure E-1 (Page 8 of 19). Authorities Required for System/36 Procedures on the AS/400 System

System/36 Procedure	Utilities Called	OS/400 Commands	Additional Authorities Required
KEYSORT	\$DDST/KEYSORT		*USE, to files library. *USE, to the file.
LIBRLIBR	\$MAINT/COPY	CPYF CRTDUPOBJ	*CHANGE, to the QS36PRC and QS36SRC files in the library to which procedure or source members are to be copied. If members are to be replaced, *ALL authority is required. *USE, to the QS36PRC and QS36SRC files in the library from which procedure or source members are to be copied. *ALL, to any other objects in the TO library that are to be copied. *USE, to any other objects in the FROM library that are to be copied. *CHANGE, to the library into which the members are to be copied. *USE, to the library from which the members are to be copied.
LIST			See the DFU procedure.
LIST#			See the DFU procedure.
LISTDATA	\$COPY/COPYFILE	CPYF DSPPFM RSTOBJ	*USE, to the DB input file. *USE, to the diskette or tape input file. *USE, to the System/36 files library, if displaying or printing the DB input file. *USE, to the device description and device file for the specified diskette or tape drive, if displaying or printing a diskette or tape input file. *CHANGE, to the user profile for the diskette or tape input file being displayed or printed. *USE, to the device description and print file for the output printer.
LISTFILE	\$BICR/DISPLAY	DSPPFM CHKDKT CPYF	*USE, to the diskette drive device file and device description.
LISTFILE	\$COPY/COPYFILE		See the LISTDATA procedure.
LISTFILE	\$MAINT/COPY	DSPDKT DSPTAP DPSAVF	*USE, to the diskette or tape device descriptions and device files if I1, T1, T2 or TC are specified. *USE, to the save file (SAVF) and to the files library if F1 is specified. *USE, to the printer device description and print file on which the output is to print.
LISTFILE	\$TCOPY/DISPLAY	CPYF DSPPFM CHKTAP OVRTAPF	*USE, to the tape device description and device file. *USE, to the printer device description and print file assigned to the system list device.

Figure E-1 (Page 9 of 19). Authorities Required for System/36 Procedures on the AS/400 System

System/36 Procedure	Utilities Called	OS/400 Commands	Additional Authorities Required
LISTLIBR	\$MAINT/COPY		*USE, to the library that is listed. *USE, to the printer device description and print file assigned to the system list device.
MSDOWNL		SNDEMLIGC CHGS36MSGL	*CHANGE, to virtual diskette file QAPIVDKT.
MSRJE		STRRJESSN STRRJECSL SBMRJEJOB	*USE, to library QRJE. *USE, to the session description (*SSND). *USE, to the forms control table (*FCT) (this is optional and depends on the RJE configuration).
MS3270	#MEFP #MEEP	STREML3270	Authority to the emulation location (user-specified) and emulation devices associated with the location. *USE, to message file #ES#M1 in library QSSP.
OFCCAL	QOFEELIB	CHGS36MSGL STROFC ADDLIBLE	
OFCDFLT	QOFEELIB	CHGS36MSGL STROFC ADDLIBLE	
OFCDIR		WRKDIR CHGS36MSGL	
OFCFILE	QOFEELIB	CHGS36MSGL WRKDOC ADDLIBLE	
OFCDGRP		WRKDSTL CHGS36MSGL	
OFCLDF	QOFEELIB	CHGS36MSGL STROFC ADDLIBLE	
OFMAIL	QOFEELIB QOFMAIL	CHGS36MSGL STROFC SNDDOC ADDLIBLE	*USE, to message file QOFMSG in library QSYS.
OFMSG	QOFEELIB	CHGS36MSGL STROFC ADDLIBLE	
OFCSRCH	QOFEELIB	CHGS36MSGL STROFC ADDLIBLE	
OFSTAT	QOFEELIB	CHGS36MSGL STROFC ADDLIBLE	
OFUSER	QOFEELIB	CHGS36MSGL STROFC ADDLIBLE	

Figure E-1 (Page 10 of 19). Authorities Required for System/36 Procedures on the AS/400 System

System/36 Procedure	Utilities Called	OS/400 Commands	Additional Authorities Required
ORGANIZE	\$COPY/COPYFILE	CPYF SAVOBJ	<p>*USE, to the DB input file, if copying data to a DB output file.</p> <p>*ALL, to the DB input file, if saving a file to diskette.</p> <p>*CHANGE, to the DB output file, if any.</p> <p>*CHANGE, to the System/36 files library, if copying data to a DB output file.</p> <p>*USE, to the System/36 files library, if saving a file to diskette.</p> <p>*USE, to the device description and device file for the specified diskette drive, if the output device is diskette.</p> <p>Note: If you have *SAVSYS special authority, no other authority is required to DB input file when saving file to diskette.</p>
PASSWORD	\$PRPWD	CHGPWD	
PCEXEC	#ORXT QEXSETLB	CHGS36MSGL	<p>*USE, to QORCMD command help.</p> <p>*USE, to QDORDSP DDS display file.</p>
PCOPROF	#ORPR QEXSETLB	CHGPCOPRF CHGS36MSGL	<p>*USE, to message file QIWSMSG in library QIWS.</p> <p>*USE, to QGORPROF panel group.</p> <p>*USE, to QHORPROF panel group.</p>
PCOFRPC			See the PCEXEC procedure.
PCOTOPC			See the PCEXEC procedure.
PCOVPRT			See the PCEXEC procedure.
PCU		CVTTOFLR CHGS36MSGL	*USE, to menu PCSHOST.
PRINTKEY	\$SETCF		
PRTGRAPH	\$DPGP \$DPGR		*USE, to file QSYSPRT.
QRY	#QUDA		*USE, to message file QQRYMSG in library QSSP.
QRYDE	\$DSIN	UPDDTA	See the IDDUDCT procedure.
QRYRUN			See the QRY procedure.
READINFO		DSPHLPDOC CHGS36MSGL	*USE, to message file QOFCMSG in library QSYS.
REBLD	\$COPY/COPYFILE	CPYF	<p>*USE, to the DB input file.</p> <p>*CHANGE, to the DB output file.</p> <p>*CHANGE, to the System/36 file library.</p>

Figure E-1 (Page 11 of 19). Authorities Required for System/36 Procedures on the AS/400 System

System/36 Procedure	Utilities Called	OS/400 Commands	Additional Authorities Required
REMOVE	\$MAINT/DELETE	RMVMM DLTCLS DLTCMD DLTDTAARA DLTDTAQ DLTF DLTFCT DLTJOBQ DLTJRN DLTJRNRV DLTMSGF DLTMSGQ DLTOUTQ DLTPRTIMG DLTPGM DLTQRY DLTSSND DLTSPADCT DLTTBL	*ALL, to the QS36PRC and QS36SRC files if procedure or source members are to be removed. *ALL, to any other members that are to be removed. *USE, to the library from which the members are to be removed. *ALL, to any other object being deleted. This includes job queues, journals, journal receivers, message files, message queues, output queues, print images, programs, session descriptions, spelling aid dictionaries, and tables.
RENAME	\$RENAM/RENAME	RNMOBJ	*USE, to files library or folder. *ALL, to object being renamed. Note: You must be enrolled in the system distribution directory to rename a folder.
RESPONSE	\$ARSP/RESPONSE	ADDMSGD CHGMSGD	*CHANGE, to the message file for which responses are being altered. *USE, to the library in which the message file resides. *USE, to the source file QS36SRC containing the response data. *USE, to the library in which the source file resides.
RESTLIBR	\$MAINT/COPYLIBR	RSTLIB	*USE, to the diskette or tape device descriptions and device files if I1, T1, T2 or TC are specified. *CHANGE, to the library to be restored. *ALL, to every object in the library to be restored. Note: If you have *SAVSYS special authority, you can restore any object.

Figure E-1 (Page 12 of 19). Authorities Required for System/36 Procedures on the AS/400 System

System/36 Procedure	Utilities Called	OS/400 Commands	Additional Authorities Required
RESTORE (ALL files)	\$COPY/COPYALL	RSTOBJ RSTS36F	*ALL, to diskette or tape input files. *ALL, to DB output files being created. *CHANGE, to the System/36 files library. *USE, to the device description and device file for the specified diskette or tape drive. *ADD, to the owner's user profile for the DB output files being created. Note: If you have *SAVSYS special authority, no other authority is required for diskette or tape input files, or DB output files.

Figure E-1 (Page 13 of 19). Authorities Required for System/36 Procedures on the AS/400 System

System/36 Procedure	Utilities Called	OS/400 Commands	Additional Authorities Required
RESTORE (Single file)	\$COPY/COPYFILE	RSTOBJ CPYF RSTS36F	<p>For any restore operation, one of the following authorities is required:</p> <ul style="list-style-type: none"> *SAVSYS special authority or *CHANGE authority to the System/36 files library. <p>In addition to either of these, you must have:</p> <ul style="list-style-type: none"> *USE authority to the device description, and *USE authority to the device file for the diskette or tape device being used. <p>If an existing disk file has the same label as the diskette file to be restored, and the diskette file is to be restored without changes, and a load-to-old operation is not performed, then one of the following authorities is required:</p> <ul style="list-style-type: none"> *SAVSYS special authority or *ALL authority to the existing disk file. <p>If an existing disk file has the same label as the disk file you create by restoring the diskette file, and the restored disk file is to have a file label or file attributes that differ from the diskette file, or record selection is to be performed, or you do a load-to-old operation, then one of the following authorities is required:</p> <ul style="list-style-type: none"> *SAVSYS special authority or *CHANGE authority to the user profile of the owner of the diskette file being restored. <p>In addition to either of these, you must have:</p> <ul style="list-style-type: none"> *ALL authority to the existing disk file, and *USE authority to the diskette file being restored. <p>If an authority holder has the same label as the diskette file to be restored, but no disk file exists with that label, and the diskette file is to be restored without changes, then one of the following authorities is required:</p> <ul style="list-style-type: none"> *SAVSYS special authority or *CHANGE authority to the user profile of the owner of the diskette file being restored. <p>In addition to either of these, you must have:</p> <ul style="list-style-type: none"> *ALL authority to the authority holder.

Figure E-1 (Page 14 of 19). Authorities Required for System/36 Procedures on the AS/400 System

System/36 Procedure	Utilities Called	OS/400 Commands	Additional Authorities Required
RESTORE (Single file) <i>(continued)</i>	\$COPY/COPYFILE	RSTOBJ CPYF RSTS36F	<p>If an authority holder has the same label as the disk file you create by restoring the diskette file, but no disk file exists with that label, and the restored disk file is to have a file label or file attributes that differ from the diskette file, or record selection is to be performed, then one of the following authorities is required:</p> <ul style="list-style-type: none"> *SAVSYS special authority or *CHANGE authority to the user profile of the owner of the diskette file being restored. <p>In addition to either of these, you must have:</p> <ul style="list-style-type: none"> *ALL authority to the authority holder, and *ALL authority to the diskette file being restored if the file is a logical file, or *USE authority to the diskette file being restored if the file is a physical file. <p>If no disk file or authority holder exists with the same label as the diskette file to be restored, and the diskette file is to be restored without changes, then one of the following authorities is required:</p> <ul style="list-style-type: none"> *SAVSYS special authority or *CHANGE authority to the user profile of the owner of the diskette file being restored. <p>If no disk file or authority holder has the same label as the disk file you create by restoring the diskette file, and the restored disk file is to have a file label or file attributes that differ from the diskette file, or record selection is to be performed, then one of the following authorities is required:</p> <ul style="list-style-type: none"> *SAVSYS special authority or *CHANGE authority to the user profile of the owner of the diskette file being restored. <p>In addition to either of these, you must have one of the following:</p> <ul style="list-style-type: none"> *ALL authority to the diskette file being restored if the file is a logical file, or *USE authority to the diskette file being restored if the diskette file is a physical file.
RGZFILE		CPYF CHGPF RGZPFM	*ALL, to the DB file

Figure E-1 (Page 15 of 19). Authorities Required for System/36 Procedures on the AS/400 System

System/36 Procedure	Utilities Called	OS/400 Commands	Additional Authorities Required
RJFILE		CVTRJEDTA	*USE, to the library QRJE. *USE, to the session description *SSND. *USE, to the forms control table *FCT (optional). *USE, to message file QRJEMSG in library QRJE.
RJTABLE		CRTFCT DLTFCT ADDFCTE CHGFCT CHGFCTE WRKFCT CMDFCT GO	*ALL, to the forms control table *FCT. *USE, to the forms control table and the library QRJE. *USE, to the CMDFCT menu.
RPGC	QEXSETLB \$UASF/SPOOL \$UASC #RPDD #RPG	CHGS36MSGL	*USE, to the library #RPGLIB and all objects therein. *USE, to the message file QRPG2MSGE in library QSSP. *USE, to message file #RP#CPL1 and #RP#CPL2 in library #RPGLLIB. *USE, to the library containing the source file. *USE, to the source file QS36SRC in the library containing the source program. *CHANGE, to the library to contain the compiled program.
RPGR	QEXSETLB \$MAINT/COPY \$SFGR/LOADMBR \$SFGR/INOUT \$SFGR/CREATE \$MAINT/DELETE #RPGEN	CHGS36MSGL	*USE, to the library #RPGLIB and all objects therein. *USE, to message file #RP#CPL1 and #RPCPL2 in library #RPGLIB. *USE, to the library containing the source program. *USE, to the source file QS36SRC in the library containing the source program. *CHANGE, to the library to contain format load member.
RPGSDA	QEXSETLB	CHGS36MSGL	See the SDA procedure. *USE, to message file #RP#CPL1 in library #RPGLIB.
RPGSEU	QEXSETLB	CHGS36MSGL	See the SEU procedure. *USE, to message file #RP#CPL1 and #RP#CPL2 in library #RPGLIB.

Figure E-1 (Page 16 of 19). Authorities Required for System/36 Procedures on the AS/400 System

System/36 Procedure	Utilities Called	OS/400 Commands	Additional Authorities Required
RPGX		CRTS36AUTO CHGS36MSGL	*USE, to #RPGLIB and all objects within. *USE, to message file #RP#CPL1 and #RP#CPL2 in library #RPGLIB. *USE, to the library containing the source program. *USE, to the source file QS36SRC in the library containing the source program.
SAVE (ADD)	\$COPY/COPYADD	SAVOBJ	*ALL, to the DB input file. *USE, to the System/36 files library. *USE, to the device description and device file for the specified diskette or tape drive. Note: If you have *SAVSYS special authority, the DB input files require no other authority.
SAVE (ALL)	\$COPY/COPYALL	SAVOBJ	*ALL, to the DB input files. *CHANGE, to the System/36 files library. *USE, to the device description and device file for the specified diskette or tape drive. Note: If you have *SAVSYS special authority, DB input files require no other authority.
SAVE (Single file)	\$COPY/COPYFILE	CPYF SAVOBJ	*ALL, to the DB input file. *USE, to the System/36 files library. *USE, to the device description and device file for the specified diskette or tape drive. Note: If you have *SAVSYS special authority, the DB input file requires no other authority.
SAVELIBR	\$MAINT/COPYLIBR	SAVLIB	*USE, to the diskette or tape device descriptions and device files if I1, T1, T2 or TC are specified. *CHANGE, to the library to be restored. *ALL, to every object in the library to be restored. Note: If you have *SAVESYS authority, you can save any object.
SDA	QSDBIN in QPDA	CHGS36MSGL CRTSRCPF CRTS36DSPF CRTS36MNU	To use all SDA functions, *CHANGE for the selected library. *ALL for the source file QS36SRC in the selected library, and *USE for the display file to be viewed from the selected library. *USE, to library #SDALIB and all objects within.
SET	\$SETCF	CHGDEVDS	*CHANGE, to device description of the display.

Figure E-1 (Page 17 of 19). Authorities Required for System/36 Procedures on the AS/400 System

System/36 Procedure	Utilities Called	OS/400 Commands	Additional Authorities Required
SEU	QEXSETLB #SEU	STRSEU CHGS36MSGL	*ALL, to the file being edited, either QS36SRC or QS36PRC. *CHANGE, to the library containing these files. *USE, to library #SEULIB and all objects within.
SORT	#GSORT/SOURCE	FMTDTA	*USE, to the source file QS36SRC. *USE, to the source library. *ALL, to the input files. *CHANGE, to the output file. *USE, to message file #GS#MM in library QSSP.
SRTX	#KASRT/SOURCE	FMTDTA	*USE, to source library. *USE, to the source file QS36SRC. *ALL, to the input files. *CHANGE, to the output files. *USE, to message file #GS#MM in library QSSP.
SRTXBLD		STRCGU	*ALL, to *IGCSRT, object QCGMSTR and object QCGACTV. *USE, to library #CGULIB and all objects within.
STARTPCO	QEXSETLB #ORTS	CHGS36MSGL	*USE, to message file QIWSMSG in library QIWS. *USE, to library QIWS and all objects within.
STOPPCO		SIGNOFF CHGS36MSGL	
TAPECOPY	\$TCOPY/TRANSFER	OVRTAPF CHKTAP CPYTOTAP CPYFRMTAP	*USE, to the tape device description and device file. *USE, to the disk file to be copied to tape and to the files library. *CHANGE, to the files library if a tape file is to be copied to a disk file. *CHANGE, to the disk file if a tape file is to be added to a disk file.
TAPEINIT	\$TINIT/VOL	CHKTAP INZTAP	*USE, to the device file and device description for the tape drive.
TEXTDCT	#TUPH		*ALL, to file QADDENDA in QUSR library.
TEXTDOC (blank)	#TUPH	WRKDOC	*USE, to the folder. *USE, to message file QOFCMSG in library QSYS.
TEXTDOC (copy)	#TUPH	CPYDOC	*CHANGE, to the specified documents and folders. *USE, to message file QOFCMSG in library QSYS.

Figure E-1 (Page 18 of 19). Authorities Required for System/36 Procedures on the AS/400 System

System/36 Procedure	Utilities Called	OS/400 Commands	Additional Authorities Required
TEXTDOC (create)	#TUPH	CRTDOC	*CHANGE, to the specified folder. *USE, to message file QOFCMSG in library QSYS.
TEXTDOC (delete)	#TUPH	DLTDLO	*ALL, to the specified document. *USE, to the folder. USE, to message file QOFCMSG in library QSYS.
TEXTDOC (merge)	#TUPH	MRGDOC	*CHANGE, to the specified documents and folders. *USE, to message file QOFCMSG in library QSYS.
TEXTDOC (paginate)	#TUPH	PAGDOC	*CHANGE, to the specified document. *USE, to message file QOFCMSG in library QSYS.
TEXTDOC (print)	#TUPH	PRTDOC	*USE, to the specified document. If *ALL is specified for the document, *USE authority is required for all documents in the specified folder. *USE, to message file QOFCMSG in library QSYS.
TEXTDOC (prtfile)	#TUPH	PRTDOC	*CHANGE, to the System/36 files library. *CHANGE, to the specified database file, if it is already created. See the PRINT option for additional authorities. *USE, to message file QOFCMSG in library QSYS.
TEXTDOC (rename)	#TUPH	RNMDLO	*ALL, to the specified document. *USE, to message file QOFCMSG in library QSYS.
TEXTDOC (revise)	#TUPH	EDTDOC	*CHANGE, to the specified document. *USE, to message file QOFCMSG in library QSYS.
TEXTDOC (spell)	#TUPH	CHKDOC	*CHANGE, to the specified document. *USE, to message file QOFCMSG in library QSYS.
TEXTDOC (view)	#TUPH	DSPDOC	*USE, to the specified document. *USE, to message file QOFCMSG in library QSYS.
TEXTFLDR	#TUPH	WRKFLR	*USE, to the specified folder.
TEXTOBJ	#TUPH	WRKDOC	*USE, to the specified folder.
TEXTPROF	#TUPH	WRKTXTPRF	
TEXTPRTQ	#TUPH	WRKDOCPRTQ	

Figure E-1 (Page 19 of 19). Authorities Required for System/36 Procedures on the AS/400 System

System/36 Procedure	Utilities Called	OS/400 Commands	Additional Authorities Required
TIMER		CHGSYSVAL DSPSYSVAL	*ALLOBJ and *SECADM special authorities, if attempting to enable or to disable any of the three unattended IPL operations.
TOLIBR	\$MAINT/COPY	RSTS36LIBM RSTOBJ CPYF	*USE, to the diskette or tape device descriptions and device files if I1, T1, T2 or TC are specified. *USE, to the save file (SAVF) if F1 is specified. *USE, to the files library if F1 is specified. *CHANGE, to the QS36PRC and QS36SRC files if the procedures or source are to be copied. If members are to be replaced, *ALL authority is required. *ALL, to any other object to be copied. *CHANGE, to the library the members are to be copied to. Note: If you have *SAVESYS special authority, you can copy any object.
TRANSFER	\$BICR/TRANSFER	ADDPFM CHKDKT CPYFRMDKT CPYTODKT	*USE, to the diskette drive device file and device description. *USE, to the file, if transferring to diskette and *ALL, to the file, if transferring from diskette. *CHANGE, to the file if doing a load-to-old.
UPDATE			See the DFU procedure.
UPDATE#			See the DFU procedure.
WRKSPL		WRKSPLF CHGS36MSGL	
WRKUSER		WRKUSRJOB CHGS36MSGL	

System/36 Operator Control Commands

The following table lists System/36 commands with the special authorities needed to run them:

Figure E-2 (Page 1 of 3). Authorities Required for System/36 Commands on the AS/400 System

System/36 Commands	OS/400 Commands	Authorities Required
ASSIGN		Not supported
CANCEL jobname JOBQ,ALL PRT	WRKUSRJOB CLRJOBQ DLTSPLF	*JOBCTL, if ALL is specified. Owner of the spool file. If printer id, ALL, FORMS, or USER are specified, one of the following authorities: <ul style="list-style-type: none"> *SPLCTL *JOBCTL and output queue containing spool file is defined OPRCTL (*YES). *READ, *ADD and *DTL authority to the output queue containing the spool file and the queue is defined AUTCHK(*DTAAUT). Owner of the output queue containing the spool file and the queue is defined AUTCHK(*OWNER).
SESSION	WRKUSRJOB	
CHANGE COPIES DEFER FORMS ID JOBQ JOBS PRT PRTY SEP	CHGSPLFA CHGSPLFA CHGSPLFA CHGSPLFA WRKJOBQ CHGJOBQE CHGSPLFA CHGJOB CHGWTR	See CANCEL PRT command. See CANCEL PRT command. See CANCEL PRT command. *JOBCTL *JOBCTL *JOBCTL *JOBCTL *JOBCTL See CANCEL PRT command.
CONSOLE	CHGMSGQ	*USE and *DELETE, to the message queue *READ, to the library containing the message queue.
HOLD JOBQ PRT	WRKJOBQ HLDSPLF	*JOBCTL and output queue containing spool file entry is defined OPRCTL (*YES), if parameter ALL is specified.
INFOMSG		
JOBQ	SBMJOB	*USE, to job queue QBATCH in library QGPL.
MENU	GO	*USE, to menu. *USE, to library, if specified.
MODE		Not supported.

Figure E-2 (Page 2 of 3). Authorities Required for System/36 Commands on the AS/400 System

System/36 Commands	OS/400 Commands	Authorities Required
MSG		
workstn-id	SNDNETMSG	*CHANGE, to the associated message queue.
user-id	SNDNETMSG	*CHANGE, to the associated message queue.
PC location	SNDNETMSG	*CHANGE, to the associated message queue.
user-defined	SNDNETMSG	Must be enrolled in the distribution directory.
group name	SNDNETMSG	*CHANGE, to the associated message queue.
ALL	SNDBRKMSG	*CHANGE, to the associated message queue.
OFF	SIGNOFF	
POWER OFF	PWRDWN SYS	*JOBCTL
PRTY	WRKUSRJOB	
RELEASE		
JOBQ	WRKJOBQ	*JCBCTL, if the job being released is not your own.
PRT	RLSSPLF	*JOBCTL and output queue containing spool file entry is defined OPRCTL (*YES), if parameter ALL is specified. See CANCEL PRT command.
REPLY	DSPMSG	*USE, to the message queue and library that contains the message queue.
RESTART	CHGSPLFA	See CANCEL PRT command.
START		
JOB	WRKUSRJOB	*JCBCTL
JOBQ,ALL	CHGJOBQE	*JCBCTL, *ALL (to the QINTER and QBATCH subsystem descriptions)
JOBQ,priority	CHGJOBQE	*JCBCTL, *ALL (to the QINTER and QBATCH subsystem descriptions)
JOBQ,jobname	CHGJOB	*JCBCTL
PRT	STRPRTWTR	*JCBCTL
SERVICE		Not supported.
SESSION		Not supported.
SYSTEM	STRSBS	*OBJOPR to subsystem and *READ to the library.
WRKSTN	WRKCFGSTS	*OBJOPR to device description.
STATUS		
ALERT	DSPNETA	
APPC		Not supported.
COMM	WRKCFGSTS	
COMCNFIG	WRKCFGSTS	
JOBQ	WRKJOBQ	
LINE	WRKCFGSTS	
MESSAGE	DSPMSGQ	
MSRJE	WRKRJESSN	
PRT	WRKSPLF	
SESSION	DSPJOB	
SUBSESS		Not supported.
SUBSYS		Not supported.
SYSTASK		Not supported.
USERS	WRKUSRJOB	
WORKSTN	WRKCFGSTS	
WRT	WRKWTR	
STATUSF		
JOBQ	WRKJOBQ	
PRT	WRKSPLF	
WORKSTN	WRKCFGSTS	
USER	WRKUSRJOB	

Figure E-2 (Page 3 of 3). Authorities Required for System/36 Commands on the AS/400 System

System/36 Commands	OS/400 Commands	Authorities Required
STOP		
JOB	WRKUSRJOB	*JOBCTL
JOBQ	CHGJOBQE	*JOBCTL, *ALL (to the QBATCH and QINTER subsystem descriptions)
PRT	ENDWTR	See CANCEL PRT command.
SERVICE		Not supported.
SESSION		Not supported.
SYSTEM	ENDSBS	*JCBCTL
WORKSTN	WRKCFGSTS	*OBJOPR, to the device description.
TIME		
VARY	VRYCFG	*JOBCTL

System/36 OCL Statements

The following table lists System/36 OCL statements with the special authorities needed to run them:

Figure E-3 (Page 1 of 3). Authorities Required for System/36 OCL Statements on the AS/400 System

OCL Statements	OS/400 Commands	Authorities Required
ABEND		
ALLOCATE	ALCOBJ	*USE, for the device description of the allocated device.
ASSIGN		Supported for compatibility only.
ATTR RELEASE-YES		*CHANGE, for job queue QS36EVOKE in library QGPL.
CANCEL	DLTSPLF	Owner of spool file. If printer id, ALL, FORMS or USER are specified, one of the following authorities: <ol style="list-style-type: none"> *SPLCTL *JOBCTL and output queue containing spool file is defined OPRCTL(*YES). *READ, *ADD and *DLT authority to the output queue containing the spool file and the queue is defined AUTCHK(*DTAAUT). Owner of the output queue containing the spool file and the queue is defined AUTCHK(*OWNER).
CHANGE COPIES FORMS ID	CHGSPLFA CHGSPLFA CHGSPLFA	Owner of the spool file. If printer id, ALL, FORMS or USER are specified, one of the following authorities: <ol style="list-style-type: none"> *SPLCTL *JOBCTL and output queue containing spool file is defined OPRCTL(*YES). *READ, *ADD and *DLT authority to the output queue containing the spool file and the queue is defined AUTCHK(*DTAAUT). Owner of the output queue containing the spool file and the queue is defined AUTCHK(*OWNER).
COMM		Not supported.

Figure E-3 (Page 2 of 3). Authorities Required for System/36 OCL Statements on the AS/400 System

OCL Statements	OS/400 Commands	Authorities Required
COMPILE		*USE, for the library specified. *USE, to the library containing the source program. *USE, to the source file QS36SRC in the library containing the source program. *CHANGE, to the library to contain the compiled program.
DATE		
DEALLOC	DLCOBJ	*USE, for the device description of the allocated device.
DEBUG		
EVOKE	SBMJOB	*CHANGE, for job queue QS36EVOKE in library QGPL.
FILE (disk)		*USE, for the System/36 files library.
FILE (diskette)		*USE, for the diskette device description.
FILE (tape)		*USE, for the tape device description.
FILELIB	ADDLIBLE	*USE, for the library specified.
FORMS		
IMAGE		Supported for compatibility only.
INCLUDE	SBMJOB	*USE, for the procedure library and the procedures file QS36PRC in that library. Note: The SMBJOB command applies only to MRT procedures.
INFOMSG		
JOBQ	SBMJOB	*CHANGE for job queue QBATCH in library QGPL.
LIBRARY	CHGCURLIB	*USE, for the library specified.
LOAD		*USE, for the program being loaded.
LOCAL		
LOG		
MEMBER		*USE, for the library containing the message member.
MENU	GO	*USE, for the menu specified. *USE, for library, if specified.
MSG	SNDNETMSG	*CHANGE, to the associated message queue.
workstn-id	SNDNETMSG	*CHANGE, to the associated message queue.
user-id	SNDNETMSG	*CHANGE, to the associated message queue.
PC location	SNDNETMSG	Must be enrolled in the distribution directory.
user-defined	SNDNETMSG	*CHANGE, to the associated message queue.
group name	SNDNRKMSG	*CHANGE, to the associated message queue.
ALL		
NOHALT		*JOBCTL, if SYSTEM is specified.
OFF	SIGNOFF	
POWER	PWRDWN SYS	*JCBCTL
PRINTER		
PROMPT		*USE, for the display format and its library.
REGION		Supported for compatibility only.
RESERVE		Supported for compatibility only.
RUN	OVRICFDEVE ALCOBJ	*USE, for the program specified in the LOAD statement.

Figure E-3 (Page 3 of 3). Authorities Required for System/36 OCL Statements on the AS/400 System

OCL Statements	OS/400 Commands	Authorities Required
SESSION		
SETDEV		*USE, to library specified *USE, to device descriptions
START	STRPRTWTR	See CANCEL OCL statement.
STOP	ENDWTR	See CANCEL OCL statement.
SWITCH		
SYSLIST		
TIMERSET	CHGSYSVAL DSPSYSVAL	*ALLOBJ and *SECADM special authorities, if attempting to enable or to disable any of the three unattended IPL operations.
VARY	VRYCFG	*JOBCTL
WAIT	DLYJOB	
WORKSTN		

System/36 Procedure Control Statements

The following table lists System/36 procedure control statements with the special authorities needed to run them. Procedure control statements not listed do not require any special authorities.

Procedure Control Statements	Authorities Required
IF LOAD	*USE, for the library.
IF SOURCE	*USE, for the library.
IF PROC	*USE, for the library.
IF SUBR	*USE, for the library.

OS/400 System/36 Commands

The following table lists OS/400 commands that pertain to the System/36 environment, along with the special authorities needed to run them. Besides these special authorities, you need *USE authority for library QSSP.

Figure E-4 (Page 1 of 3). Authorities Required for OS/400 Commands in the System/36 Environment

OS/400 Command	Referenced Object	Authorities Required to Referenced Object
CHGS36	Configuration object QS36ENV Library #LIBRARY	*CHANGE, to configuration object QS36ENV *USE, to Library #LIBRARY
CHGS36A	Configuration object QS36ENV Library #LIBRARY	*CHANGE, to configuration object QS36ENV *USE, to Library #LIBRARY
CHGS36PGMA	Program Library	*OBJMGT, to the program *USE, to the library
CHGS36PRCA	Source file QS36PRC Source file library	*CHANGE, to source file QS36PRC *USE, to source file library
CHGS36SRCA	Source file QS36SRC Source file library	*CHANGE, to source file QS36SRC *USE, to source file library
CRTMSGFMNU	Message file with command messages Message file library Message file with option messages Message file library Source file QS36DDSSRC Source file library Source file if TOMBR not *NONE Source file library Display file if it already exists Display file library CRTDSPF command	*USE, to message file with command messages *USE, to message file library *USE, to message file with option messages *USE, to message file library *ALL, to source file QS36DDSSRC *USE, to source file library *ALL, to source file if TOMBR is not *NONE *CHANGE, to source file library *ALL, to display file if it already exists *CHANGE, to display file library *USE, to CRTDSPF command
CRTS36CBL	Library #COBLIB and its contents Source file with COBOL source Source file library Program if it already exists Program library	*USE, to library #COBLIB and its contents *USE, to source file with COBOL source *USE, to source file library *ALL, to the program if it already exists *CHANGE, to the program library
CRTS36DSPF	Source file with SFGR source Source file library Source file QS36DDSSRC Source file library Source file if TOMBR not *NONE Source file library Display file if it already exists Display file library CRTDSPF command	*USE, to source file with SFGR source *USE, to source file library *ALL, to source file QS36DDSSRC *USE, to source file library *ALL, to source file *CHANGE, to source file library *ALL, to the display file if it already exists *CHANGE, to display file library *USE, to CRTDSPF command
CRTS36MNU	Source file with command text Source file library Source file with option text Source file QS36DDSSRC Source file if TOMBR not *NONE Source file library Display file if it already exists Display file library Message file if it already exists Message file library CRTDSPF command CRTMSGF command ADDMSGD command	*USE, to source file with command text *USE, to source file library *USE, to source file with option text *ALL, to source file QS36DDSSRC *ALL, to source file *CHANGE, to source file library *ALL, to display file if it already exists *CHANGE, to display file library *ALL, to message file if it already exists *CHANGE, to message file library *USE, to CRTDSPF command *USE, to CRTMSGF command *USE, to ADDMSGD command

Figure E-4 (Page 2 of 3). Authorities Required for OS/400 Commands in the System/36 Environment

OS/400 Command	Referenced Object	Authorities Required to Referenced Object
CRTS36MSGF	Source file with message source Source file library Source file if TOMBR not *NONE Source file library Message file if it already exists and OPTION(*CREATE) Message file if it already exists and OPTION(*ADD) or (*CHANGE) Message file library CRTMSGF command ADDMSGD command CHGMSGD command	*USE, to source file with message source *USE, to source file library *ALL, to source file *CHANGE, to source file library *ALL, to message file if it already exists and OPTION(*CREATE) *ALL, to message file if it already exists and OPTION(*ADD) or (*CHANGE) *CHANGE, to message file library *USE, to CRTMSGF command *USE, to ADDMSGD command *USE, to CHGMSGD command
CRTS36RPG	Library #RPGLIB and its contents Message file QRP2MSG2 in library QSYS Source file with RPG source Source file library Program if it already exists Program library	*USE, to library #RPGLIB and its contents *USE, to message file QRP2MSG2 in library QSYS *USE, to source file with RPG source *USE, to source file library *ALL, to program if it already exists *CHANGE, to program library
CRTS36RPGR	Library #RPGLIB and its contents Source file RPG source Source file library Display file library Source file to receive formats Source file library Source file to receive DDS Source file library	*USE, to library #RPGLIB and its contents *USE, to source file RPG source *USE, to source file library *CHANGE, to display file library *ALL, to source file to receive formats *CHANGE, to source file library *ALL, to source file to receive DDS *CHANGE, to source file library
CRTS36RPT	Library #RPGLIB Source file with AUTO source Source file library Program if it already exists Program library	*USE, to library #RPGLIB and its contents *USE, to source file with AUTO source *USE, to source file library *ALL, to the program if it already exists *CHANGE, to program library
DSPS36	Configuration object QS36ENV Library #LIBRARY	*USE, to configuration object QS36ENV *USE, to library #LIBRARY
EDTS36PGMA	Program Library	*OBJMGT, to the program *USE, to the library
EDTS36PRCA	Source file QS36PRC Source file library	*CHANGE, to source file QS36PRC *USE, to source file library
EDTS36SRCA	Source file QS36PRC Source file library	*CHANGE, to source file QS36PRC *USE, to source file library
ENDS36		
RSTS36F	Phyfile if *PHYFILE Phyfile library if *PHYFILE To-file if it already exists To-file library Device file and device description	*USE, to phyfile if *PHYFILE *USE, to phyfile library if *PHYFILE *ALL, to to-file if it already exists *CHANGE, to to-file library *USE, to device file and device description
RSTS36FLR	Phyfile if *PHYFILE Phyfile library if *PHYFILE To-folder if it already exists and replacing the contents To-folder if it already exists and adding new data To-folder library Device file and device description Note: Must be enrolled in Office if document folder	*USE, to phyfile if *PHYFILE *USE, to phyfile library if *PHYFILE *ALL, to to-file if it already exists *OBJOPR, *ADD, *DLT, *READ, and *UPD, to to-folder if it already exists and adding new data *CHANGE, to to-folder library *USE, to device file and device description
RSTS36LIBM	Phyfile if *PHYFILE Phyfile library if *PHYFILE To-file To-file library Device file and device description	*USE, to phyfile if *PHYFILE *USE, to phyfile library if *PHYFILE *CHANGE, to to-file *CHANGE, to to-file library *USE, to device file and device description
RTVS36A	Configuration object QS36ENV Library #LIBRARY	*USE, to configuration object QS36ENV *USE, to library #LIBRARY

Figure E-4 (Page 3 of 3). Authorities Required for OS/400 Commands in the System/36 Environment

OS/400 Command	Referenced Object	Authorities Required to Referenced Object
SAVS36F	From-file From-file library Phyfile if *PHYFILE and it already exists Phyfile library if *PHYFILE Device file and device description	*USE, to from-file *USE, to from-file library *ALL, to phyfile if *PHYFILE and it already exists *CHANGE, to phyfile library if *PHYFILE *USE, to device file and device description
SAVS36LIBM	From-file From-file library Phyfile if *PHYFILE and it already exists Phyfile library if *PHYFILE Device file and device description	*USE, to from-file *USE, to from-file library *ALL, to phyfile if *PHYFILE and it already exists *CHANGE, to phyfile library if *PHYFILE *USE, to device file and device description
STRS36	Library #LIBRARY S/36 environment configuration object (QS36ENV) Library specified as current library S/36 environment files library Initial menu, if specified Menu library, if menu specified Source file QS36PRC, if initial procedure specified Source file library, if initial procedure specified Initial program, if initial program specified Initial program library, if initial program specified	*USE, to library #LIBRARY *USE, to S/36 environment configuration object (QS36ENV) *USE, to library specified as current library *USE, to S/36 environment files library *USE, to initial menu, if specified *USE, to menu library, if menu specified *USE, to source file QS36PRC, if initial procedure specified *USE, to source file library, if initial procedure specified *USE, to initial program, if initial program specified *USE, to initial program library, if initial program specified
STRS36PRC	Library #LIBRARY S/36 environment configuration object (QS36ENV) Library specified as current library S/36 environment files library Source file QS36PRC contain the procedure Source file library	*USE, to library \$LIBRARY *USE, to S/36 environment configuration object (QS36ENV) *USE, to library specified as current library *USE, to S/36 environment files library *USE, to source file QS36PRC containing the procedure *USE, to source file library
WRKS36	Configuration object QS36ENV Library #LIBRARY	*USE, to configuration object QS36ENV *USE, to library #LIBRARY
WRKS36PGMA	Program (if attributes change) Library	*USE and *OBJMGT, to program if attributes change *USE, to library
WRKS36PRCA	Source file QS36PRC (if attributes changed) Source file library	*USE and *CHANGE, to source file QS36PRC if attributes change *USE, to source file library
WRKS36SRCA	Source file QS36SRC (if attributes changed) Source file library	*USE and *CHANGE, to source file QS36SRC if attributes change *USE, to source file library

Bibliography

The following books contain information you may need. Except where otherwise indicated, each is an AS/400 system book.

General AS/400-Related Books

- *Backup and Recovery – Advanced*, SC41-4305, provides programmers with information about the different media available to save and protect system data, as well as a description of how to record changes made to database files and how that information can be used for system recovery and activity report information.
- *DDS Reference*, SC41-3712, provides application programmers with detailed descriptions of the entries and keywords needed to describe both logical and physical database files and certain device files (for displays and printers) external to the user's programs.
- *Data Management*, SC41-4710, provides application programmers with information about using files in application programs.
- *DB2 for OS/400 Database Programming*, SC41-4701, provides application programmers and programmers with a detailed discussion of the AS/400 database structure, including information on how to create, describe, and manipulate database files on the system.
- *Application Display Programming*, SC41-4715, provides application programmers with information about defining screens for display on display devices. This includes information about display device descriptions, display device files, and how they are used to define and display screens on display devices.
- *Printer Device Programming*, SC41-3713, provides application and system programmers with information on how to control and understand printing: printing elements and concepts, printer file support, print spooling support, printer connectivity, advanced function printing, and printing with personal computers.
- *Tape and Diskette Device Programming*, SC41-4716, provides application programmers with information about the tape and diskette media supported by the AS/400 system, configuration descriptions and device files for tape or diskette, and information about using tape device files or diskette device files in high-level language programs.
- *Client Access/400 for DOS and OS/2 Technical Reference*, SC41-3563, provides application programmers with technical information needed to do advanced configuration or tailoring of Client Access/400 for a special operating environment.
- *Client Access/400 for DOS with Extended Memory User Guide*, SC41-3501, provides Client Access/400 users with concepts and examples of how to use the Client Access functions in DOS.
- *Client Access/400 for OS/2 User Guide*, SC41-3521, provides users with personal computers attached to an AS/400 system with concepts and examples of how to use the Client Access/400 functions in the OS/2* environment.
- *CL Programming*, SC41-4721, provides application programmers and programmers with information on AS/400 programming topics, including a general discussion on objects and libraries, control language (CL) programming, controlling flow and communicating between programs, working with objects in CL programs, and creating CL programs, predefined and impromptu messages and message handling, and defining and creating user-defined commands and menus.
- *CL Reference*, SC41-4722, provides application and system programmers with a description of the AS/400 control language (CL) and its commands. Each command is defined, including its syntax diagram, parameters, default values, and keywords.
- *Performance Tools/400*, SC41-4340, provides programmers with information about what AS/400 Performance Tools are, gives an overview of the tools, and tells how the tools can be used to help manage system performance.
- *System/36 Environment Reference*, SC41-4731, provides application programmers, system programmers, and system operators with information about the procedure control expressions, procedures, operation control language (OCL) statements, control commands, and utilities supported in the System/36 environment.
- *Work Management*, SC41-4306, provides programmers with information about how to create an initial work management environment and how to change it.
- *Query/400 Use*, SC41-4210, provides business professionals and programmers with detailed information about how to use Query/400 to get data from any database file.
- *Security – Basic*, SC41-3301, and *Security – Reference*, SC41-4302, explain why security is necessary, define major concepts, and provide information on planning, implementing, and monitoring basic security on the AS/400 system.

- *Getting Started with OfficeVision/400*, SH21-0732, provides OfficeVision for OS/400 users with information on learning how to use the word processing functions of OfficeVision for OS/400.
- *Using OfficeVision/400*, SH21-0697, provides OfficeVision for OS/400 users with detailed information on how to use OfficeVision for OS/400, including information on handling mail and calendars.
- *Using OfficeVision/400 Word Processing*, SH21-0701, provides OfficeVision for OS/400 users with information on using all the word processing functions of OfficeVision for OS/400.

Programming Language and Utility Books

- *ADTS/400: Character Generator Utility*, SC09-1769, provides application programmers and programmers with information about using the Application Development Tools character generator utility (CGU) to create and maintain a double-byte character set (DBCS) on the AS/400 system.
- *ADTS/400: Data File Utility*, SC09-1773, provides application programmers, programmers, and help desk personnel with information about using the Application Development Tools data file utility (DFU) to create programs to enter data into files, update files, inquire into files, and run DFU programs.
- *ADTS/400: Programming Development Manager*, SC09-1771, provides application programmers, system programmers, and help desk personnel with information about using the Application Development Tools programming development manager (PDM) utility to work with lists of libraries, objects, members, and user-defined options to easily perform such operations as copy, delete, and print.
- *ADTS/400: Screen Design Aid*, SC09-1768, provides application programmers with information about using the Application Development Tools screen design aid (SDA) to design, create, and maintain display formats and menus on the AS/400 system in the System/38 environment.
- *ADTS/400: Screen Design Aid for the System/36 Environment*, SC09-1893, provides application programmers and system operators with information on how to use the screen design aid (SDA) for developing displays, menus, and online information in the System/36 environment of the AS/400 system.
- *ADTS/400: Source Entry Utility*, SC09-1774, provides application programmers and help desk personnel with information about using the Application Development Tools source entry utility (SEU) to create and edit source members.
- *BGU User's Guide and Reference*, SC09-1408, provides application programmers, programmers, system administrators, and business and technical professionals with information about using AS/400 Business Graphics Utility (BGU) to create various types of charts.
- *System/36-Compatible COBOL User's Guide and Reference*, SC09-1815, provides application programmers with information about using COBOL in the System/36 environment on the AS/400 system.
- *System/36-Compatible RPG II User's Guide and Reference*, SC09-1818, provides application programmers with information on how to design, code, enter, compile, test, and run RPG II programs. In addition, differences between compiling in the System/36 environment and the AS/400 system are described and explained.
- *Software Installation*, SC41-4120, provides system operators and system administrators with information on new programs and releases and how to install them.
- *System/36 Assembler Conversion Newsletter*, GC21-8160, provides information about rewriting System/36 assembler subroutines.
- *DFU User's Guide and Reference*, SC09-1362, provides application programmers with information to create programs that list files, change an existing list program, and run a program against a given file to produce a listing in the System/36 environment.
- *IDDU Use*, SC41-3704, provides administrative secretaries, business professionals, and programmers with detailed information on how to use the OS/400 interactive data definition utility (IDDU) to describe data dictionaries, files, and records to the system.
- *Using the Object Distribution Facility/36 PRPQ*, SC12-9800, provides information about using the Object Distribution Facility/36 PRPQ.

Communications Books

- *APPN Support*, SC41-3407, provides programmers with information for defining or using OS/400 advanced peer-to-peer networking (APPN).
- *APPC Programming*, SC41-3443, provides application programmers with information for developing application programs that use OS/400 advanced program-to-program communications (APPC).
- *Asynchronous Communications Programming*, SC41-3444, provides application programmers with information which includes a description of asynchronous communications, configuration requirements, commands used to start a communications session, and programming considerations for the AS/400 system.
- *BSC Equivalence Link Programming*, SC41-3445, provides application and system programmers with the information needed to write programs using

- OS/400 binary synchronous communications equivalence link (BSC/EL) with the AS/400 system to communicate with a remote system.
- *SNA Distribution Services*, SC41-3410, provides system operators and system administrators with information about administering data communications applications on an AS/400 system and may also be useful to a programmer who works with data communications functions on the AS/400 system.
 - *Finance Communications Programming*, SC41-3449, provides application programmers, system operators, and system administrators with information on the OS/400 finance support program. It describes how finance support communicates with a controller and how to set up finance support.
 - *ICF Programming*, SC41-3442, provides application programmers with the information needed to write application programs that use AS/400 communications and the ICF file.
 - *Intrasystem Communications Programming*, SC41-3447, provides application programmers with information for defining or using intrasystem communications support to develop communications between two application programs on the same system.
 - *Communications Management*, SC41-3406, provides application programmers with information on how to start, stop, verify, and test communications, handle communications errors, and work with communications status.
 - *Communications Configuration*, SC41-3401, provides system operators, system programmers, and service personnel with general configuration information, including detailed descriptions of network interface, line, controller, device, mode, and class-of-service descriptions, configuration lists and connection lists.
 - *Remote Work Station Support*, SC41-3402, provides system operators, application and system programmers, and service personnel with information on using the following supports: DHCF, Display Station Pass-Through, 3270 remote attachment, remote work station configuration support, and 5394 on an SNA backbone.
 - *Retail Communications Programming*, SC41-3448, provides application programmers and system administrators with information on using the OS/400 retail support program. It describes how retail support communicates with a controller and how to set up retail support.
 - *SNA Upline Facility Programming*, SC41-3446, provides application and system programmers with programming information for using the OS/400 Systems Network Architecture (SNA) upline facility with the AS/400 system, describing how to set up the SNA upline facility, how to write application programs for the SNA upline facility, and the return codes the SNA upline facility can send to a program.
 - *Local Device Configuration*, SC41-4121, provides system operators and system administrators with information on how to do an initial configuration and how to change that configuration. It also contains conceptual information about device configuration.

Migration Books

- *System/36 Migration Planning*, SC41-4152, provides application programmers, system administrators, and data processing managers with information to help migration of products and applications with the System/36 Migration Aid.
- *System/36 Migration Assistant*, SC41-4151, provides system operators, application programmers, programmers, and data processing managers with information about using the System/36 Migration Aid to move System/36 items to the AS/400 system using menus and displays, or commands.

Index

Special Characters

// IF LOAD 6-9
// IF SUBR 6-9
\$SFGR
 See screen format generator
***NONE** 2-1
***S36** 2-1, 3-2
***SYSVAL** 2-1
#\$@INCLGRPH C-1
#LIBRARY 5-4, 6-1

Numerics

3270 device emulation 3-2

A

abnormal ending 18-6
ACCEPT statement 13-24
access algorithms
 choosing A-1
 defining A-2
 direct files A-1
 examples A-2
 handling synonym records A-1
 indexed file with keys A-4
 randomizing techniques A-6
accessing folders 8-1
ACQUIRE statement 13-24
acquired session 13-37
Add Authority List Entry (ADDAUTLE)
 command 11-7
ADDAUTLE
 See Add Authority List Entry command
addressing 5-1
advanced peer-to-peer networking (APPN) 13-4
advanced program-to-program communication (APPC)
 definition 13-4
 ICF support 13-16
 mapping 13-19
 output support 13-45
 procedures 13-4
 programming considerations 13-45
 reason codes 13-11
 System/36 to AS/400 13-40, 13-41
 using SESSION OCL statement 13-19
all object special authority 11-3
ALLOCATE OCL statement 9-6, 10-5
alphanumeric fields 12-5
alternative indexed files 7-9, 7-12

APPC

 See advanced program-to-program communication

APPN

 See advanced peer-to-peer networking

ARCHIVE procedure 8-2

AS/400 CL commands in the System/36 environment

 error handling 17-3, 17-4
 in procedures
 special characters 17-3
 substitution expressions 17-4
 syntax 17-3
 used interactively 17-2

AS/400 system

 Client Access/400 1-5
 device identification 3-3
 general environment values 3-9
 ICF files 13-17
 library QSSP 5-3
 objects 5-1
 office
 tasks 1-4
 word processing 1-4
 program start request errors 13-11
 programming languages 1-5
 securing files 7-3
 subsystem considerations 13-7
 System library (QSYS) 6-1

ASP

 See auxiliary storage pool

assigning libraries 6-3

asynchronous communications 13-34

attention key 2-5

ATTR OCL statement 2-4

attributes

 special environment (SPCENV) 2-1
 values 2-1

audit (AUDIT) 11-4

authority

 all object (*ALLOBJ) 11-3
 assigning 11-2
 audit (AUDIT) 11-4
 data 11-5
 defined 11-3
 files 7-3
 holders 11-7
 input/output system configuration (IOSYSCFG) 11-4
 job control (*JOBCTL) 11-3
 lists 11-7
 object 11-5
 public 11-7
 save system (*SAVSYS) 11-3
 security administrator 11-3

authority (*continued*)
 service (*SERVICE) 11-3
 special 11-2
 specifying for libraries 6-4
 spool control (SPLCTL) 11-4
 system-defined
 All (*ALL) 11-6
 Change (*CHANGE) 11-6
 Exclude (*EXCLUDE) 11-6
 Use (*USE) 11-6
 user profiles 11-2
autodial support 13-45
automatic advance to next tape drive 10-4
auxiliary storage pool (ASP)
 references 6-10
 user information 6-10

B

backup and recovery 1-4
bar codes, printing D-12
BARCODE D-12
BARSIZE D-12
BARTYPE D-12
basic data exchange format 9-1
batch jobs 2-2, 6-8
batch programs, definition 16-1
begin filled area (BEGAREA) option D-8
begin graphics area (BEGSEG) option D-8
BEGSEG
 See begin graphics area option
BGU
 See business graphics utility
binary format 12-4
BLDFILE
 See Build File procedure
BLDGRAPH
 See Build Graphics Object File procedure
BLDINDEX
 See Build Index (BLDINDEX) procedure
BLDLIBR
 See Build Library procedure
BLDMENU
 See Build Menu procedure
block, definition 1-2
blocking records 7-25
bold printing D-4
box characters D-4
BSCCL
 BSC/CICS to AS/400 system 13-41
 BSC/IMS to AS/400 system 13-42
 configuration considerations 13-37
 programming considerations 13-37
 terminology considerations 13-36
Build File (BLDFILE) procedure 7-1, 7-23

Build Graphics Object File (BLDGRAPH)
 procedure D-11
Build Index (BLDINDEX) procedure 7-9
Build Library (BLDLIBR) procedure 6-6
Build Menu (BLDMENU) procedure 1-3, 14-7
business graphics utility (BGU) 1-4, C-1

C

cache 3-9, 7-4
CALL statement 17-8
called program, definition 17-8
calling program, definition 17-8
CANCEL PRT command 4-6
CATALOG procedure 5-5, 6-7, 8-1, 10-5
CGU
 See character generator utility
chaining
 definition 7-25
 menus 14-5
Change Authority List Entry (CHGAUTLE)
 command 11-7
CHANGE COPIES command 4-6
CHANGE DEFER command 4-6
Change Device Printer (CHGDEVPRT)
 command 2-3
CHANGE FORMS command 4-6
CHANGE ID command 4-6
Change Job (CHGJOB) command 4-4
Change Member (CHNGEMEM) procedure 6-7
CHANGE PRT command 4-6
CHANGE PRTY command 4-6
CHANGE SEP command 4-6
Change System Value (CHGSYSVAL)
 command 2-1, 4-3, 4-5
Change System/36 (CHGS36) command 3-4
Change System/36 Environment Attributes (CHGS36A) 2-5, 3-2
Change System/36 Message List (CHGS36MSGL)
 command 2-5
Change System/36 Procedure Attributes (CHGS36PRCA) command 2-5
Change System/36 Program Attributes (CHGS36PGMA) command 2-5
Change System/36 Source Attributes (CHGS36SRCA) command 2-5, 14-23
Change User Profile (CHGUSRPRF) command 2-1, 11-2
character generator utility (CGU) 1-4
character orientation (CHARORI) option D-8
character size (CHARSIZE) option D-9
character style options D-3
characters per inch (cpi) D-3
CHARORI
 See character orientation option

CHARSIZE

See character size option

CHGAUTLE

See Change Authority List Entry command

CHGDEVPR

See Change Device Printer command

CHGJOB

See Change Job (CHGJOB) command

CHGS36

See Change System/36 command

CHGS36A

See Change System/36 Environment Attributes command

CHGS36MSGL

See Change System/36 Message List command

CHGS36PGMA

See Change System/36 Program Attributes command

CHGS36PRCA

See Change System/36 Procedure Attributes command

CHGS36SRCA

See Change System/36 Source Attributes command

CHGSYSVAL

See Change System Value command

CHGUSRPRF

See Change User Profile command

CHNGEMEM

See Change Member procedure

CIRCLE option D-9**CL commands**

See commands

Client Access/400 1-5**COBOL statements 13-24****code-link form displays 14-15****COLOR option D-4, D-9****command processor, definition 18-2****command security 11-11****commands**

ADDAUTLE (Add Authority List Entry) 11-7
 CANCEL PRT 4-6
 CHANGE COPIES 4-6
 CHANGE DEFER 4-6
 CHANGE FORMS 4-6
 CHANGE ID 4-6
 CHANGE PRT 4-6
 CHANGE PRY 4-6
 CHANGE SEP 4-6
 CHGAUTLE (Change Authority List Entry) 11-7
 CHGDEVPR (Change Device Printer) 2-3
 CHGJOB (Change Job) 4-4
 CHGS36 (Change System/36) 3-4
 CHGS36A (Change System/36 Environmental Attributes) 2-5
 CHGS36A (Change System/36 Attributes) 3-12
 CHGS36MSGL (Change System/36 Message List) 2-5

commands (continued)

CHGS36PGMA (Change System/36 Program Attributes) 2-5
 CHGS36PRCA (Change System/36 Procedure Attributes) 2-5
 CHGS36SRCA (Change System/36 Source Attributes) 2-5, 14-23
 CHGSYSVAL (Change System Value) 2-1, 4-3, 4-5
 CHGUSRPRF (Change User Profile) 2-1, 11-2
 control language 6-9
 CPYFRMDKT (Copy from Diskette) 9-7
 CPYTODKT (Copy to Diskette) 9-7
 CRTAUTL (Create Authority List) 11-7
 CRTDEVPR (Create Device Printer) 2-3
 CRTMSGFMNU (Create System/36 Message File Menu) 2-5
 CRTS36CBL (Create System/36 COBOL Program) 2-5
 CRTS36DSPF (Create System/36 Display File) 2-5, 14-21, 14-25
 CRTS36MNU (Create System/36 Menu) 2-5
 CRTS36MSGF (Create System/36 Message File) 2-5
 CRTS36RPG (Create System/36 RPG II Program) 2-6
 CRTUSRPRF (Create User Profile) 2-1, 11-2
 DLTOVR (Delete Override) 13-44
 DSPJOBLOG (Display Job Log) 16-28
 DSPMSG (Display Message) 2-4
 DSPOBJAUT (Display Object Authority) 11-5
 DSPS36 (Display System/36) 3-13
 DSPSYSVAL (Display System Value) 2-1
 EDTOBJAUT (Edit Object Authority) 11-5
 EDTS36PGMA (Edit System/36 Program Attributes) 2-6
 EDTS36PRCA (Edit System/36 Procedure Attributes) 2-6
 EDTS36SRCA (Edit System/36 Source Attributes) 2-6
 ENDS36 (End System/36) 2-2, 2-6
 ENDSBS (End Subsystem) 6-4
 GRTOBJAUT (Grant Object Authority) 11-5
 HOLD PRT 4-6
 in System/36 environment 17-2
 overview 2-2
 OVRDBF (Override Database File) 7-35
 OVRICFDEVE (Override ICF Device Entry) 13-18
 OVRICFF (Override ICF File) 13-44
 procedures 2-5
 RCLSTG (Reclaim Storage) 6-4
 RELEASE PRT 4-6
 RESTART PRT 4-6
 RGZDLO (Reorganize Document Library Object) 8-2
 RMVAUTLE (Remove Authority List Entry) 11-7
 RSTDLO (Restore Document Library Object) 8-2

commands *(continued)*

RSTS36F (Restore System/36 File) 2-6
RSTS36FLR (Restore System/36 Folder) 2-6, 8-3
RSTS36LIBM (Restore System/36 Library Member) 2-6, 6-10
RTVS36A (Retrieve System/36 Environment Attributes) 2-6
SAVDLO (Save Document Library Object) 8-2
SAVLIB (Save Library) 6-3
SAVOBJ (Save Object) 6-3
SAVS36F (Save System/36 File) 2-6, 9-7
SAVS36LIBM (Save System/36 Library Member) 2-6
SBMJOB (Submit Job) 2-3
SETATNPGM (Set Attention Program) 2-5
SIGNOFF 2-4
START PRT 4-6, 4-7
STATUS 4-5
STATUS SESSION 4-2
STATUSF 4-5
STOP PRT 4-6
STRS36 (Start System/36) 2-1, 2-2, 2-6, 11-11
STRS36PRC (Start System/36 Procedure) 2-2, 2-6, 11-11
STRSEU (Start Source Entry Utility) 6-2
WRKOBJLCK (Work Object Lock) 6-5
WRKS36 (Work with System/36 Configuration) 2-6
WRKS36 (Work with System/36 Environment command) 3-13
WRKS36PGMA (Work with System/36 Program Attributes) 2-6
WRKS36PRCA (Work with System/36 Procedure Attributes) 2-6
WRKS36SRCA (Work with System/36 Source Attributes) 2-7

communications

APPC (advanced program-to-program communications) 13-16, 13-40
asynchronous 13-16, 13-31, 13-34
autodial and telephone number list support 13-45
BSC/CICS 13-41
BSC/IMS 13-42
BSC/CEL 13-36, 13-37
configuring environment 13-1
controller description 13-5
debugging 13-30
ENABLE hierarchy/example 13-6
file transfer 13-30, 13-31, 13-33
files 17-12
finance communications 13-16, 13-38
ICF
 COBOL 13-24
 files 13-17
 IDDU data dictionaries 13-27
 operations 13-20
 RPG II 13-23
 system-supplied formats 13-25

communications *(continued)*

intrasystem communications 13-16, 13-39
line description 13-5
override commands 13-44
OVRICFDEVE command 13-18
procedures examples 13-2
program start request errors 13-11
QICDMF file 13-17
receiving objects 13-47, 13-48
resource name 13-6
retail 13-39
return codes 13-28, 13-30
sending objects 13-47
SESSION OCL statement 13-18
SNA distribution services (SNADS) 13-46
SNA upline facility (SNUF) 13-16, 13-41, 13-42
subsystems 13-15
testing applications 13-28
VRYCFG hierarchy/example 13-6

compress, definition 9-3

computer output reduction (COR) D-6

configuration menus 3-4

configuring

 asynchronous considerations 13-34

 communications environment 13-1

consecutive processing method 7-14

console, definition 2-3

CONTINUE parameter 4-8, 17-11, 17-12

control command, definition 1-1

control language commands

 See commands

control language, definition 1-1

controlling print spooling 4-4

Copy from Diskette (CPYFRMDKT) command 9-7

Copy to Diskette (CPYTODKT) command 9-7

COPYDATA procedure 7-1, 7-3, 7-24

copying files 7-3

copying information 9-4

copying libraries 6-7

COPYPRT procedure 4-6, 4-7

cpi

 See characters per inch

CPYFRMDKT

 See Copy from Diskette command

CPYTODKT

 See Copy to Diskette command

Create Authority List (CRTAUTL) command 11-7

Create Device Printer (CRTDEVPRT) command 2-3

Create System/36 COBOL Program (CRTS36CBL) command 2-5

Create System/36 Display File (CRTS36DSPF) command 2-5

Create System/36 Menu (CRTS36MNU) command 2-5

Create System/36 Message File CRTS36MSGF command 2-5

Create System/36 Message File Menu (CRTMSGFMNU) command 2-5
Create System/36 RPG II Program (CRTS36RPG) command 2-6
Create User Profile (CRTUSRPRF) command 2-1, 11-2
creation date, definition 7-2
CRTAUTL
 See Create Authority List command
CRTDEVPRT
 See Create Device Printer command
CRTS36CBL
 See Create System/36 COBOL Program command
CRTS36DSPF
 See Create System/36 Display File command
CRTS36MNU
 See Create System/36 Menu command
CRTS36MNUF
 See Create System/36 Message File Menu command
CRTS36MSGF
 See Create System/36 Message File command
CRTS36RPG
 See Create System/36 RPG II Program command
CRTUSRPRF
 See Create User Profile command
current position D-10

D

D-Spec

adjust/fill B-19
 automatic record advance B-20
 blink field B-21
 column separators B-22
 constant data B-24
 constant type B-24
 continuation B-26
 controlled field exit B-20
 data type B-17
 DBCS B-26
 enable dup key B-20
 field length B-14
 high intensity B-21
 horizontal position B-15
 input allowed B-17
 line number B-14
 lowercase B-24
 mandatory entry B-18
 mandatory fill B-18
 nondisplay B-22
 output data B-15
 position cursor B-19
 protect field B-21
 reverse image B-22
 self-check B-19

D-Spec (continued)

shift-in B-26
 shift-out B-26
 underline B-22

data

authority
 Add (*ADD) 11-6
 Delete (*DLT) 11-6
 Execute (*EXECUTE) 11-6
 Read (*READ) 11-5
 Update (*UPD) 11-6
 communications 1-3
 compression 9-3
 definition 8-1
 dictionaries 8-2
 transfer 1-5
 types 14-11

Data Description Specifications (DDS)

creating new files 14-22
 creating, adding, changing, or deleting a display file 14-21

data file utility (DFU) 1-4, 7-1

data terminal equipment (DTE), definition 13-34

database files

deleted files stored in cache 7-4
 in System/36 application job steps 17-8
 naming conventions 17-11

date-differentiated files 7-2

DBCS

See double-byte character set

DDS

See Data Description Specifications

DEALLOC OCL statement 9-6, 10-5

DEBUG statement 16-28

DEFER parameter 4-8

defer write (DFRWRT) attribute 14-26, B-1

Delete Override (DLTOVR) command 13-44

DELETE procedure 7-2, 8-1

device description 13-2

device identification 3-3

DFRWRT

See defer write attribute

DFU

See data file utility

disk

allocating to a job 9-6
 coexistence considerations 9-6
 data compression 9-3
 definition 5-1
 exchange formats 9-1
 file expiration dates 9-3
 files 9-2
 measuring activity 5-5
 migrating 9-8
 organizing space 7-21
 overview 9-1

disk *(continued)*

- preparing 9-4
- removing a file from 7-3
- restoring 9-7
- space 7-21
- storage 1-2, 5-1
- system information 5-3
- types and storage capacities 9-1
- types of files 9-2
- user information 5-4

diskette

- exchange formats
 - basic data 9-1
 - H-data 9-2
 - I-data 9-2
- files 9-2
- I/O device 17-14
- information
 - copying 9-4
 - listing 9-5
 - removing 9-6
 - restoring 9-5
 - saving 9-5
 - storing 9-2

DISP-NEW keyword 5-5**DISP-NEW parameter 7-1****display**

- adjacent-form 14-14
- attributes 14-12
- code-link form 14-15
- color or highlighting 14-15
- control (S) specifications B-1
- copying output from output queue 4-6
- creating formats 14-17
- data management 14-10, 14-12
- data types 14-11
- description 1-3, 14-10
- designing 14-13
- enhancements 14-25
- erase input fields operation 14-12
- files 7-4, 14-26, 17-12
- fixed-form 14-14
- FORMAT procedure 14-18
- formats
 - programming languages 14-19
 - with COBOL 14-19
 - with RPG II 14-19
 - within a procedure 14-20
- free-form 14-14
- input operations and fields 14-11
- maximum number of devices 14-23
- menu-form 14-14
- multiple formats 14-15
- online help information 14-18
- output operations and fields 14-11
- overriding fields 14-13

display *(continued)*

- public authority 14-23
- self-check digits 14-17
- station device error considerations 19-8
- suppressing input 14-13
- system operator 15-9
- work station 15-8

display IDs

- removing 3-14

Display Job Log (DSPJOBLOG) command 16-28**Display Message (DSPMSG) command 2-4****Display Object Authority (DSPOBJAUT) command 11-5****Display System Value (DSPSYSVAL) command 2-1****Display System/36 (DSPS36) command 3-2****displays**

- Change S/36 Diskette IDs 3-7
- Change S/36 Display IDs 3-5
- Change S/36 Environment Configuration 3-5
- Change S/36 Environment Values 3-9
- Change S/36 MRT Security and Performance 3-11
- Change S/36 Printer IDs 3-6
- Change S/36 Tape IDs 3-7
- Change System/36 3270 Device Emulation Values 3-8
- Display System/36 Environment Configuration 3-13

DLTOVR

- See Delete Override command

document, definition 1-2**double-byte character set (DBCS) support 20-3****downloading forms and box characters D-4****drawer selection (DRAWER) option D-4****DROP statement 13-25****DSPJOBLOG**

- See Display Job Log command

DSPMSG

- See Display Message command

DSPOJBAUT

- See Display Object Authority command

DSPS36

- See Display System/36 command

DSPSYSVAL

- See Display System Value command

dual-routed messages 15-13**DUPKEY parameter 7-11****duplicate keys 7-11****E****Edit Object Authority (EDTOBJAUT) command 11-5****Edit System/36 Procedure Attributes (EDTS36PRCA) command 2-6****Edit System/36 Program Attributes (EDTS36PGMA) command 2-6****Edit System/36 Source Attributes (EDTS36SRCA) command 2-6**

EDTOBJAUT

See Edit Object Authority command

EDTS36PGMA

See Edit System/36 Program Attributes command

EDTS36PRCA

See Edit System/36 Procedure Attributes command

EDTS36SRCA

See Edit System/36 Source Attributes command

EMPHASIS option D-4**end filled area (ENDAREA) option D-8, D-9****end segment (ENDSEG) option D-9****End Subsystem (ENDSBS) command 6-4****End System/36 (ENDS36) command 2-2, 2-6****ENDAREA**

See end filled area option

ENDS36

See End System/36 command

ENDSBS

See End Subsystem command

ENDSEG

See end segment option

error

detection

description 19-3

program language 19-3

subroutines 19-3

System/36-Compatible COBOL language 19-4

System/36-Compatible RPG II language 19-4

user-coded routines 19-4

disk device failures 19-1

equipment failures 19-1

job log 19-4

power failure 19-1

prevention

automatic response function 19-2

description 19-2

testing and debugging 19-2

unscheduled ending of jobs 19-2

using WAIT and FILE OCL statements 19-3

programming 19-1

recovery

data backup 19-5

description 19-4

equipment 19-4

historical data 19-5

master files 19-5

methods 19-6

service aid procedures 19-8

return codes 19-4

system failures 19-1

system operator 19-2

types 19-1

user 19-2

expiration dates

diskette files 9-3

tape files 10-3

EXTEND parameter 7-23**extendable file 7-23****externally-described files 17-10****F****field definition (D) specification B-13****file and library storage 1-2, 5-1****File Library (FILELIB) OCL statement**

changing

current files library 6-10, 7-1

current library list search indicator 6-10, 7-31

files library 7-31

File Library (FLIB) procedure

changing

files library 5-2

session files library 6-10, 7-1

session library list search indicator 6-10, 7-31

FILE OCL statement 7-1, 19-3**file transfer subfiles**

APPC 13-33

description 13-30

subroutine parameters 13-31

support considerations 13-33

FILELIB

See File Library (FILELIB) OCL statement

files

activity 7-20

adding records 7-25

alternative indexed 7-12

attributes

delete-capable 7-24

extendable 7-23

job 7-21

list of 7-21

resident 7-21

scratch 7-21

blocked or unblocked data 10-1

blocking records 7-25

change errors 7-30

COPYFILE 10-2

copying 7-3

creating 7-1

creating a sequential set on tape 10-7

data-differentiated 7-2

database 17-8

date-differentiated 7-34

dates 7-2

DBLOCK 7-26

deadlock conditions 7-29

delete-capable 7-24

deleting records from 7-24

description 7-1

direct 7-5, 7-14, 7-17

diskette 9-2

display 14-26

files *(continued)*

- displaying 7-4
- dynamically created 5-5
- EXCHANGE 10-2
- exchanging with other systems 10-3
- expiration dates 9-3, 10-3
- extendable 7-23
- extending 7-35
- externally-described 17-10
- group 7-2
- indexed 7-5, 7-14, 7-17
- job 7-21
- label definition 7-1
- library 5-2, 7-31
- library QSSP 5-3
- LIBRFILE 10-2
- loading program 7-5
- locked records 7-29
- logical 7-9
- master 7-19
- members 7-34
- multiple file libraries 7-31
- multiple indexes 7-9, 7-10
- multiple names 7-30
- naming conventions 5-4, 7-1
- organization
 - activity 7-20
 - application 7-20
 - choosing 7-19
 - description 7-5
 - direct 7-5, 7-7
 - disk space 7-21
 - indexed 7-5, 7-8
 - master 7-19
 - processing 7-20
 - sequential 7-5
 - transaction 7-19
- physical 7-9
- placing data in 7-3
- printing 7-4
- processing
 - consecutive method 7-14
 - current record pointer 7-12
 - generalized method 7-17
 - in System/36 environment 17-8
 - keyed 7-13
 - methods 7-13
 - nonkeyed 7-13
 - programming considerations 7-31
 - random by key 7-17
 - random by relative record number 7-16
 - sequential by key 7-15
 - with deleted records 7-25
 - with duplicate keys 7-12
- program-described 7-33, 17-10
- record blocking 7-25

files *(continued)*

- record sharing protection 7-28
- record-mode 6-6
- relative record number 7-7
- releasing locked records 7-29
- remote 7-38
- removing 7-3
- renaming 7-3
- resident 7-21
- restoring multiple-index 7-11
- Save/Restore 10-3
- SAVEFLDR 10-3
- SAVELIBR 10-3
- saving multiple-index 7-10
- scratch 7-21
- searching 6-10
- sector-mode 6-5
- securing 7-3
- sequential 7-5, 7-14, 7-17
- sharing
 - considerations 7-26
 - deadlock conditions 7-29
 - levels 7-26, 7-36
 - record protection 7-28
 - releasing locked records 7-29
 - WAIT parameter 7-28
 - waiting for files to become available 7-27
 - within the same job 7-37
- specifying in a program 7-3
- storage 5-1, 7-4
- synonym records 7-8
- transaction 7-19
- use 7-19
- user 5-4

fill pattern D-10**fill pattern, definition** D-8**finance communications support** 13-38**first-level message, definition** 15-1**fixed-form displays** 14-14**fixed-form menu** 14-3**flag parameter** 14-21**FLIB procedure**

See File Library (FLIB) procedure

floating-point format 12-5**folders**

- accessing 8-1
- coexistence considerations 8-3
- creating 8-1
- deleting 8-1
- description 8-1
- listing information 8-1
- members 8-1
- migration considerations 8-1, 8-3
- naming conventions 5-4
- programming considerations 8-2
- removing 8-1

folders (*continued*)
renaming 8-1
reorganizing 8-2
restoring 8-2
saving 8-2
securing 8-1
user 5-4

font selection D-3
form printing D-11

FORMAT
parameter 9-1
procedure 1-3, 14-18

FORMAT2 parameter 9-1

FORMS
See forms characters option
forms characters (FORMS) option D-4
forms generation utility D-11
free-form displays 14-14
FROMLIBR procedure 9-7

G

generalized processing method 7-17
Grant Object Authority (GRTOBJAUT)
command 11-5
graphic transparent (IGTRANS) option D-9
graphics object file D-11
group files 7-2
group libraries, definition 6-1
GRTOBJAUT
See Grant Object Authority command

H

H-data exchange format 9-2
H-Spec
boundary indicator B-13
field name B-14
folder name B-10
help display file name B-10
help document name B-10
help format name B-9
help library name B-10
help text label B-9
lower right row and column B-12
online document B-13
restore display format B-12
sequence number B-9, B-13
suppress selection B-12
upper left row and column B-11
HALT parameter 14-24
help definition (H) specifications B-8
help information, online 14-18
highlighting D-4
HOLD PRT command 4-6

HRI
See human readable interpretation
human readable interpretation (HRI) D-12

I

I-data exchange format 9-2
ICF
See intersystem communications function
IDDU
See interactive data definition utility
IDDUDCT procedure 8-2
IDDUDFN procedure 8-2
IDDUDISK procedure 8-2
IDDULINK procedure 8-2
IDDUPRT procedure 8-2
IF expression, definition 16-26
IF LOAD 6-9
IF SUBR 6-9
IGTRANS
See graphic transparent option
IHTRANS D-5
INCLGRPH C-1
index
alternative 7-9
file organization 7-8
keys 7-11
multiple 7-9, 7-10
primary 7-8
indexed files 7-5, 7-14
informational messages 15-1
INIT procedure 2-4, 9-1
initial program security 11-4
initiator, definition 3-4
input
fields 14-11
operations 14-11
stream, definition 16-12
input/output system configuration (IOSYSCFG) 11-4
inquiry 16-6
intelligent printer data stream (IPDS)
advanced function support D-2
bar codes D-12
definition D-1
description D-1
files 7-4
parameters D-1
printer options D-2
printing
forms and graphs D-11
text D-11
subroutines D-1
transparent options D-5
interactive data definition utility (IDDU) 7-1

interactive programs, definition 16-1

intersystem communications function (ICF)

- additional considerations 13-30
- APPC (advanced program-to-program communications) 13-16
- asynchronous communications 13-16
- BSCEL 13-16
- COBOL statements for communications 13-24
- communications operations 13-20
- debugging programs 13-30
- definition 2-3
- description 13-16
- finance communications 13-16
- IDDU data dictionaries 13-27
- intrasystem communications 13-16
- mapping SESSION OCL to OVRICFDEVE 13-18
- OS/400 files 13-17
- QICDMF File 13-17
- retail communications 13-16
- return codes 13-28
- RPG II 13-23
- SNA upline facility (SNUF) 13-16
- system-supplied formats 13-25
- testing applications 13-28
- tying application to configuration 13-17

intrasystem communications

- APPC 13-45
- programming considerations 13-39, 13-45
- using CL override commands 13-44

IPDS

See intelligent printer data stream

IPTRANS D-5

J

job control (*JOBCTL) 11-3

JOB-YES parameter 7-23

jobs

- command processor 18-2
- concepts 18-1
- date 18-14
- definition 1-3
- description 18-2
- end-of-day processing 18-14
- ending
 - abnormal 18-6
 - description 18-6
 - function 18-6
 - normal 18-6
- evoking other jobs 18-11
- job priorities 18-7
- job steps 1-3, 18-1
- library 6-3
- managing 18-7
- preventing
 - cancellation 18-13
 - informational messages from appearing 18-13

jobs (continued)

- preventing (continued)
 - interruption 18-13
- procedure control expressions 18-4
- processing 18-1
- processing OCL statements 18-4
- queue
 - changing priority levels 18-9
 - definition 18-8
 - priority 18-8, 18-10
 - processing priorities 18-9
 - running jobs later 18-11
 - types 18-8
 - unattended system operation 18-12
 - using 18-8
 - WAIT OCL statement 18-11
- running
 - during initial program load (IPL) 18-13
 - without operators 18-14
- scheduling 18-7
- security classification 18-12
- starting 18-1, 18-4
- submitting to run later 18-11
- system input processing example 18-5
- using 18-1

journal files 5-4

K

key

- attention 2-5
- definition 12-5
- duplicate
 - checking for 7-12
 - processing a file 7-12
 - sequence 7-12
 - sorting 7-38
 - specifying 7-11
- function differences 14-1
- index 7-11
- random processing 7-17
- sequential processing 7-15
- system request 2-4
- using 7-11, 12-6

keyed processing 7-13

KEYSORT procedure 7-38

L

LABEL parameter 7-2

LDA

See local data area (LDA)

LEAVE parameter for tape files 10-5

libraries

- application 6-1
- assigning 6-3

libraries *(continued)*

- authority 6-4
- backup and recovery 6-4
- changing 6-4
- coexistence considerations 6-9
- copying 6-7
- creating 6-6
- creating members 6-6
- current 2-2, 6-3, 6-7
- definition 6-1
- deleting 6-3
- group 6-1
- job 6-3
- licensed program 5-4
- list search indicators 7-31
- listing files 6-7
- listing information 6-6
- lists
 - batch job information 6-8
 - current library 6-7, 7-31
 - description 6-7, 7-31
 - keyword (LIBL) 7-31
 - MRT job information 6-8
 - product libraries 6-7
 - search order 6-9
 - support for files 7-31
 - support for procedures 7-33
 - support for substitution expressions 7-33
 - support for utilities 7-32
 - system part 6-7
 - user part 6-7
- members
 - creating 6-6
 - definition 6-2
 - listing 6-6
 - load 6-2
 - names 6-2
 - procedure 6-2
 - removing 6-7
 - renaming 6-7
 - source 6-2
 - subroutine 6-2
- migration 6-10
- naming conventions 5-4, 6-1
- other licensed program 6-1
- overview 6-1
- programming guidelines 6-6
- QSSP 5-3
- record-mode files 6-6
- recovering from damage
 - #LIBRARY 6-4
 - Library QSSP 6-5
 - QS36ENV *S36 in #LIBRARY 6-5
- removing 6-3, 6-7
- renaming 6-3, 6-7
- restoring 6-3, 6-6

libraries *(continued)*

- saving 6-6
- searching
 - database files 6-10
 - indicators 7-31
 - migration commands 6-9
 - procedures 6-9
 - programs 6-9
 - source and load members 6-9
 - subroutines 6-9
- sector-mode files 6-5
- session 6-3
- set up (#LIBRARY) 3-9
- sharing 6-3
- sign-on 6-3
- specifying authority 6-4
- storage 5-1
- system (QSYS) 6-1
- System/36 environment 6-1
- types of members 6-2
- user 5-4
- user (#LIBRARY) 6-1
- using 6-2
- LIBRARY 5-4, 6-1**
- LIBRARY OCL statement 6-4**
- Library to Library (LIBRLIBR) procedure 6-7**
- library-level security 11-7**
- LIBRLIBR**
 - See Library to Library procedure
- licensed program libraries 5-4**
- licensed programs**
 - Client Access/400 1-5
 - OfficeVision for OS/400 1-4
 - Query 1-4
 - utilities and application development tools 1-4
- LINE option D-9**
- line type (LINETYPE) option D-10**
- line width (LINEWIDTH) option D-10**
- lines per inch (lpi) D-6**
- LINETYPE**
 - See line type option
- LINEWIDTH**
 - See line width option
- List File (LISTFILE) procedure 6-6, 7-4**
- List Library (LISTLIBR) procedure 6-6**
- LISTDATA procedure 7-4**
- LISTFILE**
 - See List File procedure
- listing files 6-7**
- listing information from diskette 9-5**
- LISTLIBR**
 - See List Library procedure
- load members, definition 6-2**
- LOAD OCL statement 17-5**
- loading program 7-5**

local data area (LDA) 16-13
LOG OCL statement 16-28
logical unit, definition 13-2

M

magnetic tape storage 1-2
managing jobs 18-7
mantissa, definition 12-5
MARKER option D-10
marker type (MARKTYPE) option D-10
MARKTYPE
 See marker type option
mathematical symbols D-4
MAXDEV attribute 14-23, 14-26, 16-29
MAXPGMDEV attribute 16-29

menus

BLDMENU procedure 1-3, 14-7
chaining 14-5
color or highlighting 14-9
command text message file 14-7
configuration 3-4
creating and changing 14-7
description 14-1
designing 14-4
differences
 from System/36 14-2
 help 14-3
 user 14-3
display file 14-7
fixed-form 14-3
form displays 14-14
formats 14-3
free-form 14-4
function key processing 14-1
online help information 14-7
screen design aid (SDA) 14-7
security 11-4, 14-3
system request 2-4, 18-6
user 14-2

merging graphics and text

description C-1
examples C-1
printer storage limitations C-3
printing a graphics file along with other output C-2
printing a graphics file only C-1
programming considerations C-3

message members

application design 15-2
concepts 15-1
creating or changing source 15-14
default responses 15-14
definition 1-3
displaying from procedures 15-16
file considerations 15-15
migrating 15-17

message members (*continued*)

programming guidelines 15-14
required parameters 15-17
severity levels 15-14
uses 15-1
with displays 15-17
with programs 15-17
within a procedure 15-15

messages

automatic reply handling 15-13
changing 15-3
command text file 14-7
concepts 15-1
console 15-13
converting text 15-4
creating 15-2
default responses 15-5, 15-7
displaying 15-8
dual-routed 15-13
embedding 15-10
enhancements 15-13
file considerations 15-15
files 15-2
formatting with control characters 15-9
from procedures 15-16
handling 2-3, 15-11
handling defaults 15-11
identification 13-33
inserting variable data 15-3
nondisplayed errors 15-12
problems 15-12
prompting 15-8
restrictions 15-13
sending 15-9, 15-11
severity levels 15-5
types 15-1
with displays 15-17
with programs 15-17
work station displays 15-8

mixed mode applications

architectural restrictions 17-1
AS/400 application in a System/36 environment
 job 17-10
AS/400 program followed by a System/36
 program 17-9
definition 17-1
file processing
 communications 17-12
 database 17-8
 display 17-12
 externally-described 17-10
 printer 17-11
 program-described 17-10
program control 17-5
System/36 program followed by a System/36
 program 17-10

mixed mode applications *(continued)*

- System/36 program followed by an AS/400 program 17-9
- using CL commands in System/36 procedures
 - error handling 17-4
 - special characters 17-3
 - substitution expressions 17-4
 - syntax 17-3

mixing CL, OCL commands 17-2

mixing System/36 programs and AS/400 programs

See mixed mode applications

modulus 10/modulus 11 checking B-19

MRT

See multiple requester terminal

multiple indexes for a file 7-9, 7-11

multiple requester terminal (MRT)

- configuring 3-11
- definition 2-5
- MRTDLY attribute 16-20, 16-25
- MRTMAX value 16-3, 16-17
- programs 2-5, 11-12, 16-3, 16-15, 16-23
- security 3-4, 11-12

N

NAME parameter 7-2

naming conventions

- diskette devices 17-14
- files 5-4, 7-1
- folders 5-4
- libraries 5-4
- library member 6-2
- library names 6-1
- printer files 17-12
- tape devices 17-14

national language support 1-4

NEP

See never-ending program

never-ending program 16-6

NLS 1-4

NONE 2-1

nonkeyed processing 7-13

nonrequester terminal (NRT) programs 16-4

NUMBER parameter 14-24

O

object authority

- alter (*OBJALTER) 11-5
- existence (*OBJEXIST) 11-5
- list management (*AUTLMGT) 11-5
- management (*OBJMGT) 11-5
- operational (*OBJOPR) 11-5
- reference (*OBJREF) 11-5

object, definition 5-2

OCL

See operation control language (OCL)

OCR characters D-4

OLINK procedure 6-9

online help information 14-18

operating in the System/36 environment 2-1

operating system, definition 2-3

operation control language (OCL), definition 2-3

options

- BEGAREA (begin filled area) D-8
- BEGSEG (begin graphics area) D-8
- CHARORI (character orientation) D-8
- CHARSIZE (character size) D-9
- CIRCLE D-9
- COLOR D-4, D-9
- DRAWER (drawer selection) D-4
- EMPHASIS D-4
- ENDAREA (end filled area) D-8, D-9
- ENDSEG (end segment) D-9
- FORMS (forms characters) D-4
- IGTRANS (graphic transparent) D-9
- LINE D-9
- LINETYPE (line type) D-10
- LINEWIDTH (line width) D-10
- MARKER (marker) D-10
- MARKTYPE (marker type) D-10
- PATTYPE (pattern type) D-10
- POSITION D-10
- QUALITY D-6
- ROTATE (page rotation) D-6
- TEXT D-6, D-11
- TRANS (transparent) D-5

organization of files 7-5

output

- fields 14-11
- operations 14-11
- queues 4-3, 5-4

Override Database File (OVRDBF) command 7-35

Override ICF Device Entry (OVRICFDEVE) command 13-18

Override ICF File (OVRICFF) command 13-44

OVRDBF

See Override Database File command

OVRICFDEVE

See Override ICF Device Entry command

OVRICFF

See Override ICF File command

P

page rotation (ROTATE) option D-6

pages, definition 16-5

parameters

- COBOL printer D-1
- CONTINUE 4-8, 17-11, 17-12
- DEFER 4-8

parameters *(continued)*

DISP-NEW 7-1
DUPKEY 7-11
EXTEND 7-23
FORMAT 9-1
FORMAT2 9-1
HALT 14-24
JOB-YES 7-23
LABEL 7-2
LEAVE 10-5
NAME 7-2
NUMBER 14-24
PRT 4-5
RETAIN-S 7-10
REWIND 10-5
RPG II printer D-1
SPCENV 2-1, 3-1
UNLOAD 10-5
WAIT 7-28

parent-child concept 11-11

pattern type (PATTYPE) option D-10

PATTYPE

See pattern type option

PDM

See programming development manager

POSITION option D-10

positional parameter, definition 13-1

print file, definition 4-7

Print Graphic (PRTGRAPH) procedure D-11

print graphics subroutines D-7

print spooling

commands 4-5
controlling 4-5
description 4-3

printed output

assigning priorities 4-8
attributes 4-9
combining files 4-7
controlling or displaying print spooling 4-3, 4-4, 4-5
creating and controlling 4-1
data management 4-1
delayed status 4-8
forms number 4-7
printer control guidelines 4-4
procedures 4-1
spool writer messages 4-4
system list 4-3
using output queues 4-3

printer control guidelines

change

Print key printer 4-5
printer configuration information 4-5
printer information in a procedure 4-5
session printer 4-4
system list device 4-5
system printer 4-5

printer control guidelines *(continued)*

description 4-4

printer data management 1-1, 4-1

printer files

attribute processing 17-11
CONTINUE-YES processing 17-11
naming conventions 17-12
open time processing 17-11

PRINTER OCL statement 4-1, 4-8

PRINTKEY procedure 4-5

priority, definition 18-7

procedure member, definition 6-2

procedure members, definition 6-2

procedures

// IF 6-9
ARCHIVE 8-2
attributes 16-21
BLDFILE (Build File) 7-1, 7-23
BLDGRAPH (Build Graphics Object File) D-11
BLDINDEX 7-9
BLDLIBR (Build Library) 6-6
BLDMENU (Build Menu) 1-3, 14-7
calling 16-22, 16-23, 16-29
CATALOG 5-5, 8-1, 10-5
changing printer information 4-5
CHNGEMEM (Change Member) 6-7
commands 2-5
concepts 16-21
controlling 16-27
COPYDATA 7-1, 7-3, 7-24
COPYPRT 4-6, 4-7
creating and changing 16-27
debugging 16-28
DELETE 7-2, 8-1
description 1-3
designing 16-1, 16-26
FLIB (File Library) 5-2, 6-10, 7-1
FORMAT 1-3, 14-18
FROMLIBR 9-7
functions 16-22
IDDUDCT 8-2
IDDUDFN 8-2
IDDUDISK 8-2
IDDULINK 8-2
IDDUPRT 8-2
INIT 2-4, 9-1
job log processing 16-28
KEYSORT 7-38
LIBRLIBR (Library to Library) 6-7
List File (LISTFILE) 7-4
LISTDATA 7-4
LISTFILE (List File) 6-6
listing 16-27
LISTLIBR (List Library) 6-6
members 1-3, 6-2, 16-22
migration utility 16-29

procedures *(continued)*

MRT 16-23, 16-29
naming 16-26
OLINK 6-9
parameters 16-22
parts 16-21
performance and coding techniques 16-26
PRINTKEY 4-5
programming considerations 16-27
PRTGRAPH (Print Graphic) D-11
RENAME 7-3, 8-1
RESTLIBR (Restore Library) 6-3, 6-4
RESTORE 7-1, 7-2, 7-3, 7-11
RETRIEVE 8-2
SAVE 7-2, 7-10
SAVEFLDR (Save Folder) 8-2
SAVELIBR (Save Library) 6-4, 9-7
searching for 6-9
SET 4-5
SEU (Source Entry Utility) 6-2
SYSLIST (System List) 4-5
TAPECOPY 7-1, 7-3
TAPEINIT 10-5
TESTFLDR 8-2
TEXTDOC 8-1
TEXTFLDR 8-1
TOLIBR (To Library) 6-4, 9-7
TRANSFER 7-1, 7-3, 9-2, 9-7
using 16-22
with menus 16-23

processing

batch 7-20
files
communications 17-12
database 17-8
display 17-12
printer 17-11
interactive 7-20
methods 7-13

profiles 11-2**program control** 17-5**program date, definition** 9-3**program temporary fix (PTF)**

definition 3-4
deleting 3-4
installing 3-4
IPL (initial program load) 3-4
removing 3-4
saving 3-4

program-described files, description 17-10**programming**

considerations
for accessing and maintaining folders 8-2
for diskette processing 9-4
for file processing 7-31
for managing disk storage 5-5
for MRT programs 16-15

programming *(continued)*

considerations *(continued)*
for multiple-user programs 16-13
for procedures 16-27
for tape processing 10-4
guidelines for libraries 6-6
languages 1-5
printer data management 4-1
using printer data management 4-1
using system list 4-1

programming development manager (PDM) 1-4**Programming Request for Price Quotation (PRPQ), definition** 13-46**programs**

application structure 16-5
attributes 16-6
batch and interactive 16-1
characteristics 16-2
COBOL D-1, D-2
comparison of types 16-4
description 16-1
designing 16-1
differences
batch and interactive 16-5
requesters and acquired display stations 16-5
SRT and MRT 16-5
MRT 16-3, 16-15
multiple-user 16-2, 16-13
no-user 16-2
NRT 16-4
number of users 16-2
one-user 16-2
RPG II D-1, D-2
SRT 16-2
summary table of users and requesters 16-4
types 16-2

PRPQ

See Programming Request for Price Quotation

PRT parameter 4-5**PRTAPI** D-1**PRTBAR** D-12**PRTGRAPH**

See print graphic procedure

PRTGRC D-7**PTF**

See program temporary fix

Q

QCONSOLE system value 3-1

QDEVNAMING system value 3-1

QICDMF file 13-17

QPRTEDEV system value 3-1

QS36ENV object 3-2

QSPCENV system value 2-1, 3-1

QSSP 6-1
QSSP library 5-3
QTEMP 7-31
QUALITY option D-6
Query 1-4

R

random processing 7-16
randomizing techniques A-6
RCLSTG
 See Reclaim Storage command
READ statement 13-25
read-under-format (RUF) technique
 external calls 17-12
 programming considerations 16-12
 using 14-20
 with AS/400 programs 17-13
Reclaim Storage (RCLSTG) command 6-4
record blocking
 blocking factors 7-26
 DBLOCK value 7-26
 sharing files 7-26
record-mode files 6-6
records
 deletion of records 12-6
 description 12-1
 fields
 alphanumeric 12-5
 naming 12-1
 new 12-6
 numeric 12-1
 required 12-1
 size 12-6
 format
 binary 12-4
 floating-point 12-5
 packed decimal 12-3
 zoned decimal 12-2
 keys 12-5
 layout 12-6
 length 12-6
 menus 14-4
related printed information H-1
relative record number 7-7
RELEASE PRT command 4-6
remote files 7-38
remote location name 13-1
Remove Authority List Entry (RMVAUTLE)
 command 11-7
removing
 a folder 8-1
 display IDs 3-14
 information from diskette 9-6
 libraries 6-7

RENAME procedure 7-3, 8-1
renaming a folder 8-1
renaming libraries 6-7
Reorganize Document Library Object (RGZDLO)
 command 8-2
resource ownership 11-8
RESTART PRT command 4-6
RESTLIBR
 See Restore Library procedure
Restore Document Library Object (RSTDLO)
 command 8-2
Restore Library (RESTLIBR) procedure 6-3, 6-4
RESTORE procedure 7-1, 7-2, 7-3, 7-11
Restore System/36 File (RSTS36F) command 2-6
Restore System/36 Folder (RSTS36FLR)
 command 2-6, 8-3
Restore System/36 Library Member (RSTS36LIBM)
 command 2-6, 6-10
restoring information 9-5
restoring multiple-index files 7-11
restoring the AS/400 System to System/36 9-7
retail communications support 13-39
RETAIN-S parameter 7-10
RETRIEVE procedure 8-2
Retrieve System/36 Attributes (RTVS36A) 3-2
Retrieve System/36 Environment Attributes
 (RTVS36A) command 2-6
REWIND parameter for tape files 10-5
RMVAUTLE
 See Remove Authority List Entry command
ROTATE
 See page rotation option
RSTDLO
 See Restore Document Library Object command
RSTS36F
 See Restore System/36 File command
RSTS36FLR
 See Restore System/36 Folder command
RSTS36LIBM
 See Restore System/36 Library Member command
RUF
 See read-under-format (RUF) technique
RUN OCL statement 17-5, 17-6
run priority, description 18-10

S

S-Spec
 132-column format B-7
 alarm B-3
 blink cursor B-4
 command keys B-4
 comment B-1
 erase input fields B-4
 format name B-1
 function keys B-3

S-Spec *(continued)*

- key mask B-7
- lowercase B-2
- null fill B-6
- number of lines to clear B-2
- override fields B-5
- return input B-2
- right-to-left display B-7
- sequence number B-1
- start line number B-2
- suppress input B-6

S/36 Configuration Menus 3-13

S/36 Environment Attributes 3-12

S/36 Environment Configuration 3-13

S36 2-1, 3-2

SAVDLO

See Save Document Library Object command

Save Document Library Object (SAVDLO)

command 8-2

Save Folder (SAVEFLDR) procedure 8-2

Save Library (SAVELIBR) procedure 6-4, 9-7

Save Library (SAVLIB) command 6-3

Save Object (SAVOBJ) command 6-3

SAVE procedure 7-2, 7-10

save system (*SAVSYS) 11-3

Save System/36 File (SAVS36F) command 2-6

Save System/36 Library Member (SAVS36LIBM)

command 2-6

SAVEFLDR

See Save Folder (SAVEFLDR) procedure

SAVELIBR

See Save Library (SAVELIBR) procedure

saving information 9-5

SAVLIB

See Save Library (SAVLIB) command

SAVOBJ

See Save Object (SAVOBJ) command

SAVS36F

See Save System/36 File command

SAVS36LIBM

See Save System/36 Library Member command

SBMJOB

See Submit Job command

scheduling jobs 18-7

screen design aid (SDA) 1-3, 1-4

screen format generator (\$SFGR)

- considerations 14-21
- creating, adding, changing, or deleting display file
 - formats 14-21
- data types 14-25
- definition 2-5
- format names 14-24
- FORMAT procedure parameters 14-24
- functions 14-22
- HALT parameter 14-24
- NUMBER parameter 14-24

screen format generator (\$SFGR) *(continued)*

- printed output 14-21
- public authority 14-23
- right-to-left cursor 14-24

SDA

See screen design aid

second-level message, definition 15-1

sector-mode files 6-5

securing folders 8-1

security

- all object (*ALLOBJ) 11-3
- attributes 3-11
- audit (AUDIT) 11-4
- command 11-11
- data 11-1
- definition 1-2
- initial program 11-4
- input/output system configuration (IOSYSCFG) 11-4
- introduction 11-1
- job control (*JOBCTL) 11-3
- levels 11-1
- library-level 11-7
- limited capability 11-4
- menu 11-4, 14-3
- passwords 11-1
- resource 11-1, 11-4
- save system (*SAVSYS) 11-3
- security administrator (*SECADM) 11-3
- service (*SERVICE) 11-3
- sign-on 11-1
- special authority 11-2
- spool control (SPLCTL) 11-4
- user class 11-2

sequential files

- consecutive processing 7-14
- generalized processing method 7-17
- organization 7-5
- random processing 7-16

sequential processing by key 7-15

service (*SERVICE) 11-3

session

- date, definition 18-14
- definition 2-3
- library 6-3
- printer, definition 4-4

SESSION OCL statement 13-18

Set Attention Program (SETATNPGM)

command 2-5

SET procedure 4-5

SETATNPGM

See Set Attention Program command

SEU

See source entry utility

severity code, definition 15-6

\$SFGR

See screen format generator

sharing files 7-26
sharing libraries 6-3
sign-on library 6-3
SIGNOFF command 2-4
single requester terminal program, definition 16-2
SNA
 See Systems Network Architecture (SNA)
SNA distribution services (SNADS) 13-46
SNA upline facility (SNUF) 13-41
SNADS
 See SNA distribution services
SNUF
 See SNA upline facility (SNUF)
Source Entry Utility (SEU) procedure 1-4, 6-2
source file, definition 6-2
source members, definition 6-2
SPCENV
 See special environment (SPCENV) parameter
special characters 12-5, 17-3
special environment (SPCENV) parameter 2-1, 3-1
special user authority 11-2
specifications
 display control (S) B-1
 field definition (D) B-13
 help definition (H) B-8
SPLCTL
 See spool control
spool control (SPLCTL) 11-4
spool files 4-3
spool writer messages 4-4
spooling, print 4-3
SRT 16-2
START PRT command 4-6, 4-7
Start Source Entry Utility (STRSEU) command 6-2
Start System/36 (STRS36) command 2-1, 2-2, 2-6, 11-11
Start System/36 (STRS36PRC) procedure 2-6
Start System/36 Procedure (STRS36PRC) command 2-2, 11-11
statements
 ACCEPT 13-24
 ACQUIRE 13-24
 ALLOCATE OCL 9-6, 10-5
 ATTR OCL 2-4
 CALL 17-8
 DEALLOC OCL 9-6, 10-5
 DEBUG 16-28
 DROP 13-25
 FILE OCL 7-1, 19-3
 FILELIB OCL 6-10, 7-1, 7-31
 LIBRARY OCL 6-4
 LOAD OCL 17-5
 LOG OCL 16-28
 PRINTER OCL 4-1, 4-8
 READ 13-25
 RUN OCL 17-5, 17-6

statements (continued)
 SESSION OCL 13-18
 WAIT OCL 18-11, 19-3
 WORKSTN OCL 4-5
 WRITE 13-25
STATUS command 4-5
STATUS SESSION command 4-2
STATUSF command 4-5
STOP PRT command 4-6
STRS36
 See Start System/36 command
STRS36PRC
 See Start System/36 Procedure command
STRSEU
 See Start Source Entry Utility command
subconsoles 2-3
Submit Job (SBMJOB) command 2-3
SUBR50 D-1, D-2
SUBR51 D-1, D-7
SUBR52 D-1, D-12
subroutine members, definition 6-2
subroutines
 COBOL D-1, D-2
 printing graphics D-7
 PRTAPI D-1, D-2
 PRTBAR D-1, D-12
 PRTGRC D-1, D-7
 RPG II D-2
 SUBR50 D-1, D-2
 SUBR51 D-1, D-7
 SUBR52 D-1, D-12
substitution expressions 17-4
subsystem, definition 2-1
synonym records 7-8, A-1
SYSLIST
 See System List procedure
system
 date, definition 7-2
 library, definition 5-2
 list device, definition 4-3
 list output 4-3
 program, definition 2-3
 request menu 2-4, 16-6, 16-11, 18-6
 security 11-1
System List (SYSLIST) procedure 4-5
system printer, changing 4-5
System Request menu 2-4, 16-6, 16-11, 18-6
system-defined authority
 All (*ALL) 11-6
 Change (*CHANGE) 11-6
 Exclude (*EXCLUDE) 11-6
 Use (*USE) 11-6
system-supplied formats 13-25
System/36 application, definition 17-1
System/36 environment
 access levels 11-8

System/36 environment *(continued)*

- accessing functions 2-2
- application job steps
 - AS/400 application in a System/36 environment job 17-10
 - AS/400 program followed by a System/36 program 17-9
 - System/36 program followed by a System/36 program 17-10
 - System/36 program followed by an AS/400 program 17-9
- architectural restrictions 17-1
- auxiliary storage pool 6-10
- CL commands 17-2
- Command Entry display 2-2
- commands and procedures 2-5
- commands to access functions 2-2
- configuration 3-1
- configuration commands 3-2
- configuring 3-2
- database file processing 17-8
- device identification 3-3
- double-byte character 1-4
- entering 11-11
- files 11-10
- function key processing 14-1
- group names 11-10
- information 5-3
- libraries 11-8
- library 5-4
- library QSSP 5-3
- licensed programs 1-4
- migration 6-10
- objects 5-1
- OCL statements E-22
- OLINK procedure 6-9
- operating 2-1
- operator control commands E-20
- OS/400 commands E-25
- overview
 - backup and recovery 1-4
 - configuration 1-1
 - data communications 1-3
 - designing records 1-2
 - diskette and magnetic tape storage 1-2
 - double-byte character set (DBCS) 1-4
 - file and library storage 1-2
 - files 1-2
 - folders 1-2
 - jobs and job processing 1-3
 - libraries 1-2
 - menus and displays 1-3
 - messages and message members 1-3
 - national language support 1-4
 - operating 1-1
 - printed output 1-1
 - programs and procedures 1-3

System/36 environment *(continued)*

- overview *(continued)*
 - security 1-2
 - parent-child concept 11-11
 - printed output
 - assigning priorities 4-8
 - combining files 4-7
 - controlling print spooling 4-4
 - copying and displaying 4-6
 - creating and controlling 4-1
 - data management 4-1
 - delayed status 4-8
 - forms number 4-7
 - print spooling 4-3
 - printer control guidelines 4-4
 - procedures 4-1
 - spool writer messages 4-4
 - system list 4-3
 - using output queues 4-3
 - procedure control statements E-24
 - procedures 1-1, E-1
 - program control 17-5
 - QDEVNAMING 3-1
 - resource ownership 11-8
 - resource security file 11-8
 - return code considerations 13-28
 - search order 6-9
 - security considerations 11-11
 - SFGR functions 14-22
 - subconsoles 2-3
 - subsystem considerations 13-7
 - system library 11-11
 - system library (QSSP) 6-1
 - system values
 - QCONSOLE 3-1
 - QDEVNAMING 3-1
 - QPRTDEV 3-1
 - QSPCENV 3-1
 - user identification file 11-8
 - user library (#LIBRARY) 6-1
 - user profile attribute 2-1
- Systems Network Architecture (SNA) 13-2**
- SYSVAL 2-1**

T **tape**

- automatic advance 10-4
- coexistence considerations 10-8
- copying information 10-6
- creating a sequential set of files 10-7
- drives
 - allocating to a job 10-5
 - migration considerations 10-10
 - multiple 10-7
 - supported 10-1

tape *(continued)*

- files
 - COPYFILE 10-2
 - EXCHANGE 10-2
 - exchanging with other systems 10-3
 - expiration dates 10-3
 - LIBRFILE 10-2
 - Save/Restore 10-3
 - SAVEFLDR 10-3
 - SAVELIBR 10-3
 - formats
 - IBM standard label 10-1
 - nonlabeled 10-2
 - header labels 10-1
 - I/O device 17-14
 - LEAVE processing 10-5
 - listing information 10-7
 - marks (TM) 10-1
 - preparing 10-5
 - programming considerations 10-4
 - removing information 10-7
 - restoring information 10-7
 - REWIND processing 10-5
 - saving information 10-6
 - securing read/write access 10-4
 - security
 - access 10-4
 - other systems 10-4
 - read/write access 10-4
 - write access 10-4
 - storage 10-1
 - trailer labels 10-1
 - UNLOAD processing 10-5
 - using multiple drives 10-7
 - volume label 10-1
- TAPECOPY procedure** 7-1, 7-3
- TAPEINIT procedure** 10-5
- telephone number list support** 13-45
- TESTFLDR procedure** 8-2
- testing communications applications** 13-28
- Text (TEXT) option** D-11
- TEXT option** D-6
- text rotation** D-6
- TEXTDOC procedure** 8-1
- TEXTFLDR procedure** 8-1
- To Library (TOLIBR) procedure** 6-4, 9-7
- TOLIBR**
See To Library procedure
- TRANS**
See transparent option
- TRANSFER procedure** 7-1, 7-3, 9-2, 9-7
- transparent (TRANS) option** D-5

U

- UNLOAD parameter for tape files** 10-5
- UPSI**
See user program status indicator switch
- user**
 - class 11-2
 - files 5-4
 - folders 5-4
 - libraries 5-4
 - profile attribute 2-1
 - profiles 11-2
- user program status indicator switch** 16-22

V

- values**
 - changing 3-9
 - QSPCENV 2-1, 3-1
 - system 3-1
- virtual circuit, definition** 13-5
- virtual printer support** 1-5
- VTAM** 13-41

W

- WAIT OCL statement** 18-11, 19-3
- WAIT parameter** 7-28
- word processing** 1-4
- work file, definition** 6-4
- Work Object Lock (WRKOBJLCK) command** 6-5
- work station id, definition** 2-3
- work station, definition** 1-1
- Work with System/36 Configuration (WRKS36) command** 2-6, 3-2
- Work with System/36 Procedure Attributes (WRKS36PRCA)** 2-6
- Work with System/36 Program Attributes (WRKS36PGMA)** 2-6
- Work with System/36 Source Attributes (WRKS36SRCA)** 2-7
- WORKSTN OCL statement** 4-5
- WRITE statement** 13-25
- WRKOBJLCK**
See Work Object Lock command

X

- X.21** 13-46
- X.25** 13-5

Reader Comments—We'd Like to Hear from You!

AS/400 Advanced Series
System/36 Environment Programming
Version 3

Publication No. SC41-4730-00

Overall, how would you rate this manual?

	Very Satisfied	Satisfied	Dissatisfied	Very Dissatisfied
Overall satisfaction				

How satisfied are you that the information in this manual is:

Accurate				
Complete				
Easy to find				
Easy to understand				
Well organized				
Applicable to your tasks				

THANK YOU!

Please tell us how we can improve this manual:

May we contact you to discuss your responses? Yes No

Phone: (____) _____ Fax: (____) _____ Internet: _____

To return this form:

- Mail it
- Fax it
 - United States and Canada: **800+937-3430**
 - Other countries: **(+1)+507+253-5192**
- Hand it to your IBM representative.

Note that IBM may use or distribute the responses to this form without obligation.

Name

Address

Company or Organization

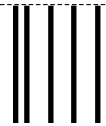
Phone No.



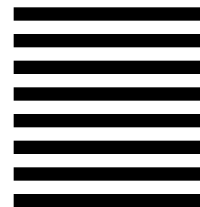
Fold and Tape

Please do not staple

Fold and Tape



NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES



BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

ATTN DEPT 542 IDCLERK
IBM CORPORATION
3605 HWY 52 N
ROCHESTER MN 55901-9986



Fold and Tape

Please do not staple

Fold and Tape



Printed in the United States of America
on recycled paper containing 10%
recovered post-consumer fiber.

SC41-4730-00





AS/400 Advanced Series

System/36 Environment Programming

Version 3