Application System/400™

SC21-9658-1

Programming: Data Management Guide

Second Edition (September 1989)

This major revision makes obsolete SC21-9658-0 and Technical Newsletter SN21-9118. See "About This Manual" for a summary of major changes to this edition. Changes or additions to the text and illustrations are indicated by a vertical line to the left of the change or addition. Changes are periodically made to the information herein; any such changes will be reported in subsequent revisions or technical newsletters.

This edition applies to Release 2 Modification Level 0 of the IBM Operating System/400 Licensed Program (Program 5728-SS1) and to all subsequent releases and modifications until otherwise indicated in new editions or technical newsletters.

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.

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 licensed program in this publication is not intended to state or imply that only IBM's licensed program may be used. Any functionally equivalent program may be used instead.

The numbers at the bottom right of illustrations are publishing control numbers and are not part of the technical content of this manual.

Publications are not stocked at the address given below. Requests for IBM publications should be made to your IBM representative or to your IBM-approved remarketer.

This publication could contain technical inaccuracies or typographical errors. A form for readers' comments is provided at the back of this publication. If the form has been removed, comments may be addressed to IBM Corporation, Information Development, Department 245, Rochester, Minnesota, U.S.A. 55901. IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

About This Manual

Data management support is the portion of the operating system that allows an application to work with files. A file is a fundamental object on the Operating System/400. Files allow data that is external to an application program to be read from or written to devices attached to the system. Files come in several varieties or types. For example, there are database files, printer files, and files used to communicate with other systems. Each file type has its own set of unique characteristics that determines how a file of that type can be used and what capabilities a file of that type can provide. In addition to these unique characteristics, there is another set of characteristics that are common to all file types. Therefore, to be able to use files to their full capabilities, it is necessary that you understand both the common characteristics of all file types and the unique characteristics of the file types you plan to use.

This manual may refer to products that are announced but are not yet available. This manual follows the convention that he means he or she.

This manual contains small programs which are furnished by IBM as simple examples to provide an illustration. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. All programs contained herein are provided to you "AS IS". THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE EXPRESSLY DISCLAIMED.

Who Should Use This Manual

This manual is intended primarily for the application programmer. This manual should also be useful for those responsible for tailoring their system to use double-byte data with the data management file support.

What You Should Know

Before using this manual, you should be familiar with general programming concepts and terminology, and have a general understanding of the AS/400 system and OS/400 operating system.

Roadmap to This Manual and Other Related Manuals

Because this manual describes the fundamental structure and concepts of the data management support on the system, it should be the first manual you read if you are unfamiliar with OS/400 data management. In particular, the first chapter should be the starting point. Once the concepts in that chapter are understood, you can proceed to read about the details necessary to develop any application. These details may be in this manual or in other manuals in the programming library. For example, a programmer writing an interactive application that updates a customer order file and prints orders would read the display, printer and spool chapters in this manual; and for information about support available for the customer order file, would read the *Database Guide*.

After you understand *what* data management support provides, you can proceed to learn *how* to use that support. The how-to aspect of data management is covered in two other groups of manuals: the high-level language manuals and the manuals that describe the various tools on the system for describing, creating, and maintaining files.

The high-level language manuals describe how to interact with the data management support. For example, they describe the syntax for extracting file descriptions from the system and including them in the program, and how to code a read operation using a keyed access path.

The tools manuals are similar to the language manuals in that the syntax of using data management is described. For example, the *DDS Reference* describes the syntax for creating field, record, and file descriptions that will be associated with a file; the *CL Reference* describes the syntax for the command that creates the file using the DDS file description.

Where appropriate, this manual refers you to the manuals in the other two groups.

How This Manual Is Organized

This manual is divided into four parts:

Part 1, "Introduction"
Part 2, "Common File Support"
Part 3, "Device File Support"
Part 4, "Appendixes"

After the introduction, Part 2 describes characteristics that are common to all files, regardless of type. The information in these chapters is of interest to you as a programmer, because the common file characteristics set the general framework by which an application and the file it uses interact. The unique file characteristics then fill in this general framework.

Part 3 provides you with detailed information about display, printer, tape, and diskette files and how they can be used in a program.

Part 4 covers double-byte character set support, and additional information on such items as the format of the feedback areas and values for return codes (both of which are means by which data management communicates its status to the program). Also included is information about edit codes used to edit printed and displayed output and information on System/36-compatible display management.

In the back of this manual are a glossary and an index. Use the glossary to find the meaning of an unfamiliar term. Use the index to look up a topic and see on which pages the topic is covered.

Detailed descriptions of other file types do not appear within this publication. Refer to "Related Printed Information" on page vi to determine where equivalent information can be found for other file types.

How This Manual Has Changed

Information added to this manual includes:

- Display support: automatic recovery for display errors.
- Printer support: new printers, changes to several parameter values, tables on specific parameter support by printer types.
- New tape support.
- Get-attributes operation information.
- System/36-compatible display management.

In addition, there are miscellaneous changes throughout the manual. Changes are noted by a vertical line to the left of the text.

Related Online Information

The following online information is available on the AS/400 system. After pressing the Help key on any menu, you can press the Help key a second time to see an explanation of how the online information works, including the index search function. You can press either the Help key or F1 for help.

Help for Displays

You can press the Help key on any display to see information about the display. There are two types of help available:

Field
Extended

Field help explains the field on which the cursor is positioned when you press the Help key. For example, it describes the choices available for a prompt. If a system message appears at the bottom of the display, position the cursor on the message and press the Help key to see information about the cause of the message and the appropriate action to take.

Extended help explains the purpose of the display. Extended help appears if you press the Help key when the cursor is outside the areas for which field help is available.

To exit the online information, press F3 (Exit). You return to the display on which you pressed the Help key.

Index Search

Index search allows you to specify words or phrases that identify the information that you want to see. To use index search, press the Help key, then press F11 (Search index). You can also use index search by entering the Start Index Search (STRIDXSCH) command on any command line or by selecting option 2 on the User Support and Education menu.

Help for Control Language Commands

To see prompts for parameters for a control language command, type the command, then press F4 (Prompt) instead of the Enter key. To see extended help for the command, type the command on any command line and press the Help key.

Online Education

AS/400 online education provides training on a wide variety of topics. To use the online education, press F13 (User support) on any system menu to show the User Support menu. Then select the option to use online education.

Question-and-Answer Function

The question-and-answer (Q & A) function provides answers to questions you may have about using the AS/400 system. To use the Q & A function, press F13 (User support) on any system menu to show the User Support menu. Then select the option to use the question-and-answer function. You can also use the question-and-answer function by entering the Start Question and Answer (STRQST) command on any command line.

Related Printed Information

Following is a list of related manuals that the *Data Management Guide* refers to. The manuals are listed with their full title and base order number. When these manuals are referred to in this manual, a shortened version of the title is used.

- *Application Development Tools: Character Generator Utility User's Guide*, SC09-1170

  This manual provides information about using the Application Development Tools character generator utility (CGU) to create and maintain a double-byte character set (DBCS) on the system.

- *Application Development Tools: Screen Design Aid User's Guide and Reference*, SC09-1171

  This manual provides information about using the Application Development Tools screen design aid (SDA) to design, create, and maintain display formats and menus.

- *Application Development Tools: Source Entry Utility User's Guide and Reference*, SC09-1172

  This manual provides information about using the Application Development Tools source entry utility (SEU) to create and edit source members.

- *Communications: Distributed Data Management User's Guide*, SC21-9600

  This manual provides information about remote file processing. It describes how to define a remote file to OS/400 DDM (distributed data management), how to create a DDM file, what file utilities are supported through DDM, and the requirements of OS/400 DDM as related to other systems.

- *Communications: Programmer's Guide*, SC21-9590

  This manual provides the information needed to write application programs that use AS/400 communications and ICF files. It also contains examples of communications programs and describes return codes.

- *Communications: User's Guide*, SC21-9601

  This manual provides communications information that is common among the AS/400 communications support functions, such as setting and changing values; communications configuration information, such as defining lines, controllers, and devices; information on handling communications errors; and information on defining and using display station pass-through.

- *Device Configuration Guide*, SC21-8106

  This manual provides information on how to do an initial configuration and how to change that configuration. It also contains conceptual information about device configuration.

- *Languages: System/36-Compatible COBOL User's Guide and Reference*, SC09-1160

  This manual provides information about using COBOL in the System/36 environment on the AS/400 system. It provides information on how to program in COBOL on the AS/400 system like a System/36 and how to use existing System/36 COBOL programs.

- *Languages: System/36-Compatible RPG II User's Guide and Reference*, SC09-1162

  This manual provides programming guide information for the RPG II language on the AS/400 system. It is intended for people who have a basic understanding of data processing concepts and of the RPG II language. The guide portion explains how to design, code, enter, compile, test, and run RPG II programs. The manual includes the RPG program cycle, the valid entries for each column of each RPG specification form and the differences between the way RPG II programs run on System/36 and the way they run on the AS/400 system. In addition, differences between compiling in the System/36 environment and the AS/400 environment are identified at the compiler level only.

- *Office: OfficeVision/400 Word Processing User's Guide*, SC21-9618

  This manual provides detailed information on how to use the word processing functions of AS/400 Office.

- *Programming: Backup and Recovery Guide*, SC21-8079

  This manual provides 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.

- *Programming: Command Reference Summary*, SC21-8076

  This manual provides quick reference information about the structure of the commands.

- *Programming: Concepts and Programmer's Guide for the System/36 Environment*, SC21-9663

  This manual provides information identifying the differences in the applications process in the System/36 environment on the AS/400 system.

  This manual helps the user understand the functional and operational differences (from a System/36 perspective) when processing in the System/36 environment on the AS/400 system.

- *Programming: Control Language Programmer's Guide*, SC21-8077

  This manual provides a wide-ranging discussion of programming topics, including: a general discussion of objects and libraries, control language (CL) programming, predefined and impromptu messages and message handling, and application testing.

- *Programming: Control Language Reference*, SBOF-0481

  This set of manuals provides a description of the control language (CL) and its commands. Each command is defined including its syntax diagram, parameters, default values, and keywords.

- *Programming: Database Guide*, SC21-9659

  This manual provides a detailed discussion of the database structure, including information on how to create, describe, and manipulate database files on the system.

- *Programming: Data Description Specifications Reference*, SC21-9620

  This manual provides detailed descriptions of the entries and keywords needed to describe database files (both logical and physical) and certain device files (for displays and printers) external to the user's programs.

- *Programming: Office Services Concepts and Programmer's Guide*, SC21-9758

  This manual provides information about writing applications that use Office functions. The manual introduces AS/400 Office application program interfaces (APIs). The manual also includes an overview of directory services, document distribution services, document library services, security services, word processing services, and information on finding new ways to integrate your applications with AS/400 Office.

- *Programming: Security Concepts and Planning*, SC21-8083

  This manual provides information about general security concepts and planning for security on the system.

- *Programming: System Reference Summary*, SC21-8104

  This manual provides quick reference information when working with the system. This manual contains summaries of information such as system values and OS/400 DDS keywords.

- *Programming: Work Management Guide*, SC21-8078

  This manual provides information about how to create a work management environment and how to change it.

- *System Operations: Operator's Guide*, SC21-8082

  This manual provides information about how to use the system unit operator panel, send and receive messages, respond to error messages, start and stop the system, use the display station function keys, control devices, and also process and manage jobs on the system.

- *Utilities: Interactive Data Definition Utility User's Guide*, SC21-9657

  This manual provides detailed information on how to use OS/400 IDDU (interactive data definition utility) to describe data dictionaries, files, and records to your system.

- *Utilities: Sort User's Guide and Reference*, SC09-1164

  This manual provides information about using the sort function for identifying input and output files, specifying sort options, utilizing effective sort run time, and identifying double-byte character set (DBCS) sort information.

Contents

Part 1. Introduction

Part 2. Common File Support

Part 3. Device File Support

Part 4. Appendixes

Figures

Tables

Part 1. Introduction

Data management is the part of the operating system that controls the storing and accessing of data by an application program. The data may be on internal storage (for example, database), on external media (diskette, tape, printer), or on another system. Data management, then, provides the functions that an application uses in creating and accessing data on the system and ensures the integrity of the data according to the definitions of the application.

Data management provides functions that allow you to manage files (create, change, override, or delete) using CL commands, and create and access data through a set of operations (for example, read, write, open, or close). Data management also provides you with the capabilities to access external devices and control the use of their attributes for creating and accessing data.

If you want to make more efficient use of printers and diskette devices, data management provides the capability of spooling data for input or output. For example, data being written to a printer can be held on an output queue until the printer is available for printing.

On the AS/400[1] system, each file has a description that describes the file characteristics and how the data associated with the file is organized into records, and, in many cases, the fields in the records. Whenever a file is processed, the operating system (Operating System/400 or OS/400)[1] uses this description.

Creating and accessing data on the system is done through the use of these file objects. Data management defines and controls several different types of files. Each type has associated CL commands to create and modify them, and the data can be created and accessed through the operations provided by data management.

File Types: The data management functions support the following types of files:

Database files are files whose associated data is stored permanently in the system.

Device files are files that provide access to externally attached devices such as displays, printers, tapes, diskettes, and other systems that are attached by a communications line. The device files supported are:

- Display files which provide access to display devices.
- Printer files which describe the format of printed output.
- Tape files which allow access to data files on tape devices.
- Diskette files which provide access to data files on diskette devices.
- Intersystem communications function (OS/400-ICF), hereafter referred to as ICF, files which allow a program on one system to communicate with a program on the same system or another system.

Save files are files that are used to store saved data on disk (without requiring diskettes or tapes).

Distributed data management (DDM) files are files that allow access to data files stored on remote systems.

[1] AS/400, Operating System/400, and OS/400 are trademarks of the International Business Machines Corporation.

Each file type has its own set of unique characteristics that determines how the file can be used and what capabilities it can provide. The concept of a file, however, is the same regardless of what type of file it is. When a file is used by a program, it is referred to by name, which identifies both the file description and, for some file types, the data itself. To be able to use files to their full capabilities, this manual is designed to help you understand both the common characteristics of all file types and the unique characteristics of display, printer, tape and diskette device files.

Part 2. Common File Support

The chapters in this part contain information that is applicable to all file types. This information includes concepts and more detailed information about some file-related functions.

Chapter 1, "File Processing" introduces the data management structure and objects and how an application program uses them to perform data management functions. Understanding the contents of this chapter is necessary before reading any of the other chapters in this manual and other manuals in the library that discuss the details of the file types not covered by this manual.

Chapter 2, "Overrides and File Redirection" contains the concepts and detailed information about temporarily making changes to files when an application program is run.

Chapter 3, "Copying Files" contains information on the system commands that can be used to copy data from one place to another, such as from a database file to a tape device, or from one database file to another.

Chapter 1. File Processing

This chapter discusses basic aspects of processing files. Topics include:

- File types supported by the system
- File descriptions: what they are, how to create them, and how to change them
- Definitions of and differences between externally described data and program-described data
- File operations supported by the system for use in high-level language programs
- File security considerations
- Sharing files in the same job
- Allocating file resources
- Temporarily changing a file when a program uses it
- Feedback areas maintained by the system
- Handling file errors when programs run

General Information about Files

In order for any program to work with the database, a device, or another system that is accessible through a communications line, it is necessary for such programs to use files. There are several different types of files supported by the operating system. By using a file of the correct type, you can have a program work with whatever it needs to. The files used are the doorways for the program to gain access to the items the program needs.

A file is a named object that exists independently of the application that uses it. The system object type for files is *FILE. Because a file is an object, it has the same general characteristics as any other object on the system. This means:

- A file comes into existence when it is created. There are CL commands that allow you to create the various types of files supported by the operating system.
- As part of the create process, the file is given a name by which it can be referred to and is stored in the library which you choose.
- Once a file is created, it can be used by your application.
- Once a file is created, it can be changed to a certain extent. There are CL commands that allow you to change the various types of files.
- A file can be secured. A file can be made public for anyone to use or can be restricted so only authorized persons can use it.
- A file has an owner.
- A file can be saved off the system and restored back on.
- A file can be deleted when it is no longer needed. The Delete File (DLTF) command allows you to delete files.

There are a number of different kinds of files on the system that you receive from IBM. These are created by IBM for your use and should serve a number of your needs. These are described in later chapters. If you have need for additional files, you can create and maintain these yourself. In some cases you will find that you can create new files by using the ones supplied by IBM as models.

File Types

The different types of files supported by the operating system can be classified into four general categories:

- Database files: Files that provide access to the database.
- Device files: Files that provide access to externally attached devices such as displays, printers, tapes, diskettes, and other systems that are attached by communications lines.
- Distributed data management (DDM) files: Files that provide access to data that is stored on a system other than the one the program is running on.
- Save files: Special-purpose files that are used to make data ready for saving or for transporting to another AS/400 system.

This section provides an overview of how an application program works with each of these general categories of files. Regardless of what type of file an application program uses, the overall processing of the file is the same.

Database Files

The key to understanding how your program can use a database file[1] is for you to understand the objects that are involved and how these objects are related. Figure 1-1 shows an overview of working with database files. When working with database files, the following items are involved:

- Application program

  This is a program that reads data from a database file or writes data to a database file. This program could be one that you develop in a programming language.

- Database file

  This is the file with which you want the application program to work. This file identifies which set of data within the database that you want the program to work with.

- Data

  This is the actual data (information). Each database file has data associated with it. This data is maintained as an ordered set of records where each record is an ordered set of one or more fields. The number of records associated with a file can grow or shrink as programs put new records into the file or delete existing records from the file.

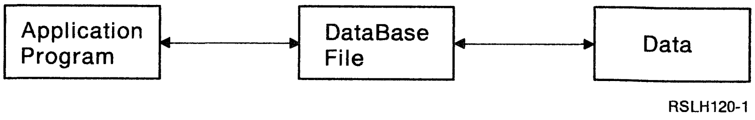


Figure 1-1. Using a Database File

The primary point of reference for working with data in the database is a database file. To work with any part of the database, the application program must first obtain a path to that part of the database which is of interest. This is accomplished by

[1] If you are interested in reading more information about database files at a later time, you can go to the *Database Guide*.

having the program perform an open operation against a database file. Opening a database file provides the program with access to the data related to that file. A path is established between the program and that part of the database. Opening additional database files provides the program with paths to additional parts of the database.

Once the file has been opened successfully, the program is now in a position to work with the actual data that is related to the file. The application may read existing records from the file or may write new records out to the file. During all read or write operations, the application program always refers to the database file that it opened. This reference to the opened database file clearly identifies what part of the database the operation is directed to. This is especially important when more than one database file is opened and used at the same time. Mistakenly referring to the wrong database file would direct the operation to the wrong set of data.

The data related to a database file is always organized and formatted according to a file description that was provided when the file was created. The program is free to work with the data only in a manner that is consistent with this description. For example, if the description states that the access path to the records is keyed, then the program can retrieve records by doing read-by-key operations. If the description states that the access path to the data is sequential, then read-by-key operations are invalid. This description, which is an integral part of the file, is used to control the flow of data on the path between the program and the database file. File descriptions are discussed in more detail later.

Device Files

Figure 1-2 on page 1-4 shows an overview of working with device files.[2] When working with device files, the following items are involved:

- Application program

  This is a program that wants to read data from a device or write data to a device. This may be a program you develop in a programming language.

- Actual device

  This is the externally attached hardware the application program wants to use. The externally attached hardware may be a tape, diskette, display, or printer unit. The externally attached hardware may also be another system that is connected to the system on which the program is running by a communications line. If the device is another system, the program that is using the device file may be connected to another program rather than to an actual hardware unit.

- Device description

  This is a description of the hardware capabilities of the device. For locally attached devices, the device description is created when the device is first attached to the system. In the event the device is attached by a communications line, there are also line and controller descriptions in addition to the device description. The sum total of all three descriptions, in this case, represents the hardware capabilities of the device. The device description is not discussed in detail within this publication. More information about device, line, and controller descriptions can be found in the *Device Configuration Guide* and the *Communications User's Guide*.

[2] If you are interested in reading more information about ICF files at a later time, go to the *Communications Programmer's Guide*. Display files, printer files, tape files, and diskette files are discussed later in this manual.

- Device file

  This is a file that contains information about how a device is to be used. By referring to a specific device file, the application states that it wants to use the device in a manner consistent with the information that is contained in the device file. The information in the device file must also be consistent with the capabilities of the device as is given by the device description. While the device is being used, the device file refers to the device description. The types of device files are: display, printer, tape, diskette, and ICF.

- Data

  This is the data (information) that the program wants to read from the device or write out to the device.

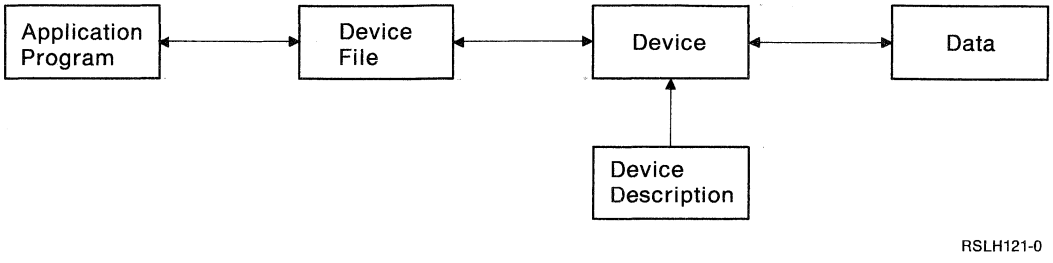


Figure 1-2. Using a Device File

The primary point of reference for working with a device is the device file. Before a program can work with a device, it must obtain a path to the device. This is accomplished by performing an open operation against a device file. Opening a device file establishes a path between the program and a device. Along this path flows the data between the program and the device. Each device file that is opened provides a path to a device for the program. A program may work with more than one device at a time by opening more than one device file. Display and ICF files allow more than one device to be attached to an open file.

A device file does not have a set of data uniquely associated with it like a database file does. The relationship between data and a device file is temporary and is established when the device file is opened. At that time, the device file is used to control the processing of data that flows between the application and the device. Once the path that was established by an open operation is closed, the relationship between the data and the device file that was used to control the processing of that data is ended.

Once a device file is opened, the program can proceed to read and write data. All operations refer to the device file rather than to the device. The device file selected controls the flow of data between the program and the device. How a device file controls the processing of data is determined by a file description, which is an integral part of the device file. A file description is provided when a device file is created. The contents of a file description vary by the type of device it is designed to work with. In general, the description in a device file is used to control the device, format output data from the program for presentation at the device, and format input data from the device for presentation to the program.

For certain device file types, spooling may be used. When spooling is used, data management stores the data in an intermediate holding area called a spooled file. Spooling is an attribute of a device file rather than another type of file. The fact that spooling is being used is not visible to the program. The program continues to work with device files just as if the device is being used directly. For input, when the program does a read operation to the device file, data management goes to the spooled file to get the data rather than going to the device. For output, when the application does a write operation, data management stores the data in a spooled file rather than putting the data out to the device.

There are a number of advantages to using spooling which would not be realized if the device was used directly. These advantages range from better device utilization to greater flexibility in the processing of the data. More information on spooling is presented in Chapter 6, "Spool Support."

Figure 1-3 shows an example of using spooling for output to a printer. As can be seen, the program works with a printer device file. The fact that the output data is going into a spooled file rather than the printer is not visible to the program. The printing of the data can occur at some later time, even after the program that produced the data has stopped.

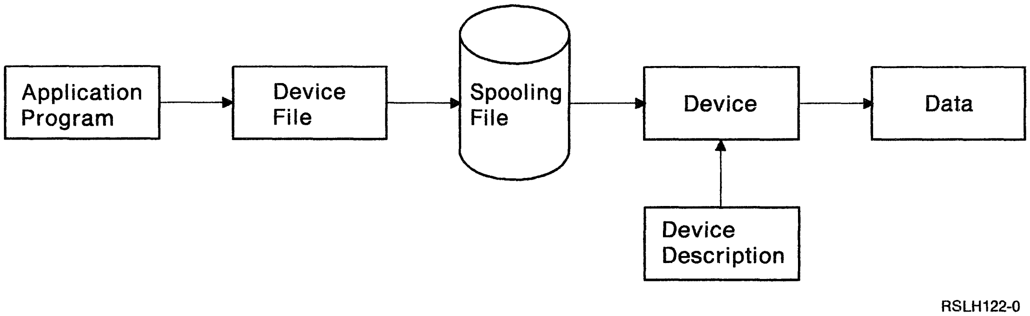


Figure 1-3. Using a Printer Device with a Spooled File

Distributed Data Management (DDM) Files

A DDM file is used to allow your application program to gain access to data that is stored on another system. The system on which your program is running is referred to as the source system, while the system at which the desired data is stored is referred to as the target system. The target system is connected to the source system by a communications line. The discussion which follows is limited to the case where both the source and target systems are AS/400 systems. In actuality, DDM files allow System/36, System/38, and AS/400 systems to be either source or target systems.[3]

Figure 1-4 on page 1-6 illustrates the use of DDM files. When working with DDM files, the items involved are:

- Application program

  This is a program that is running on the source system and wants to gain access to data on the target system. This program could be one which you develop in a programming language.

- DDM file

[3] If you are interested in reading more about DDM files at a later time, you can go to the *DDM User's Guide*.

This is the file with which you want the application program to work. The DDM file provides the ability for the application program to gain access to the target system first and to a database file on the target system second.

- Database file

  This is the database file on the target system which the application program wishes to access. Although the program is working with the DDM file, it appears to the program like it is working directly with this database file. The DDM file serves as a underlying access method to connect the program to the database file.

- Data

  This is the actual data (information) that is associated with the database file on the target system. On read operations, the data is moved from the target system to the source system, where it is made available to the program. On write operations, the data received from the program is moved from the source system and is stored on the target system.

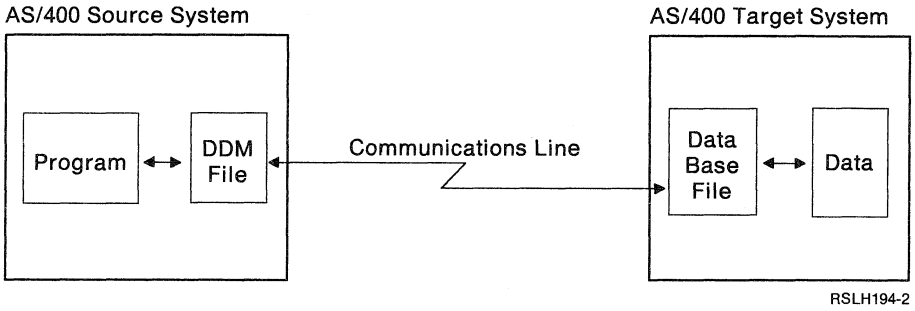


Figure 1-4. Using DDM Files

Before your application can work with a database file on the target system, it must establish a path:

- First, to the target system itself
- Second, to the correct database file on that target system

Your application program can establish a path to both the target system and the database file on the target system by opening a DDM file. In this respect, a DDM file combines aspects of both device files and database files. That is, by opening a DDM file, your program gains access to a device (the target system) and a database file.

After a path has been established, the program can proceed to work with the database file. The program can read data from the database file and write new data to the database file. On all of these operations, the program is referring to the DDM file it opened. Even though the program is referring to the DDM file, it appears as if the program is working directly with the database file. In particular, it is the database file description that controls how your program can work with the data.

There is no unique set of data associated directly with the DDM file itself. The data being processed is associated with the database file that is accessed via the DDM file. The DDM file is serving only as an intermediary to connect the program to the database file. Once the path that was established by the open operation is closed, the fact that the DDM file was an intermediary for processing the data in the database file is not remembered.

Because of this capability of a DDM file, a program that is to work with a database file can be designed without regard to where the database file actually resides. The program need only open a file. If the file happens to be a DDM file, only the operating system needs to be aware that the database file is on another system. The program itself need not be concerned and may operate in the same manner all the time.

Save Files

Save files are files that are used to prepare data in a format that is correct for backup and recovery purposes or for transportation to another system. A save file can be used to contain the output that is produced from CL commands such as the Save Library (SAVLIB) or Save Object (SAVOBJ) commands. This is the typical way in which this type of file is used. However, it is possible for your application program to also use save files[4].

Figure 1-5 illustrates the use of save files. When working with save files, the items involved are:

- Application program

  This is the program that reads data from the save file or writes data to the save file. This could be a program that you develop in a programming language.

- Save file

  This is the save file with which you want the program to work. The save file identifies a unique set of data.

- Data

  This is the data that is associated with the save file. The data within a save file is organized and formatted according to the rules that the operating system has set up for save files. Because the data is in a special format, you do not have the flexibility to control these rules to the same extent that is possible with database files.

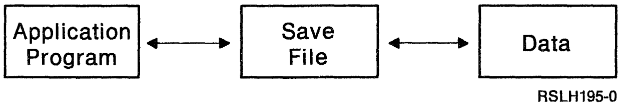


Figure 1-5. Using Save Files

Before a program can work with a save file, it must establish a path to that file. The program establishes that path by opening the save file that is of interest. After opening the file, the program can perform read and write operations against that file. On all operations, the save file that was opened is the point of reference. A program can gain access to more than one save file at a time simply by opening more than one save file.

The data that flows between the program and the save file is controlled by a file description that is associated with the save file. This description is an integral part of the save file and comes into existence when the save file is created. A program is allowed to use the data in a save file only in a manner that is consistent with this description. In this respect, save files are similar to database files. A major differ-

[4] If you are interested in reading more information about save files at a later time, you can go to the *Backup and Recovery Guide*.

ence between the two, however, is the amount of control you have over the description. When you create a database file, you are the one who provides the database description. You have complete control over what the file description should be. This is not true for save files. Data in a save file is in a special format that is suitable for backup and recovery purposes. Because of this, the operating system controls the description of the data in the save file. When a save file is created, this system-generated data description is automatically associated with the new save file.

Contrast among File Types

In comparing the figures above, you can see the overall similarity among the files from the four categories. The overall structure is the same. There are two major differences which you should be aware of, however:

- Visibility of the device

  In the case of database, DDM, and save files, the program need not be aware that the data is coming from or going to a device. The data management support for these three categories automatically handles all device-related operations and characteristics. As a result, your program need only work with the file without any concern as to how the data is stored.

  In the case of device files, because the devices are externally attached and because of the variety that can exist, the system cannot mask the device from the program.

- Data storage

  In the case of a database or save file, there is actual data that is associated with the file. A database or save file uniquely identifies a set of data. In the case of a device or DDM file, no actual data is associated with the file. The file is a tool that is used to process data that is associated with an external source such as a device or another system.

File Descriptions

Associated with each file is a file description. A file description is information that describes the characteristics of the file. The file description for a file comes into existence when the file is created. It is an integral part of the file and remains with the file until the file is deleted. Certain parts of a file description can be changed after the file is created. Such a change can be done as a permanent or temporary change.

The information that can be contained in a file description is determined by the file type, since different file types have different capabilities. A file description is essentially a declaration which states that from all the possible things that can be done with a file of a certain type, these are the ones that are to be valid for this particular file. As a result, a file description plays an important role, for it determines how a program is able to use the file. Whenever a program wants to access a file, the system uses the file description of that file to control the access. If the program attempts an operation that is inconsistent with the file description, the system does not allow the operation and will, instead, return an error condition.

The description for a file is constructed from information you provide when the file is created (with the exception of save files in which case the system controls the description). Through a file description you can specify:

- Device controls
- Linkage information for communications paths
- Spooling attribute
- Data organizations
- Data presentation formats
- Data storage formats

This overview introduces you to the concept of a file description by using examples. Keep in mind when reading this section that many other things can be specified in a file description. It is beyond the scope of this introduction to provide a complete list of all the things that can be specified in a file description. Refer to the chapters that follow later in this manual for additional information on tape, diskette, printer, and display files. Refer to the publications that are listed at the front of this manual for additional information on ICF, DDM, database, and save files.

Because data is organized by field, record, and file, file descriptions reflect this organization. A field is the smallest unit of data that is recognized and handled by the data management support of the system. A record is an ordered set of one or more fields. A file is an organized set of zero or more records (a file with zero records is empty). When constructing a file description, you may need to consider providing descriptions at three different levels, depending on the type of file you are about to create and what purpose the file is to serve. These three levels are:

- Field-level descriptions
- Record-level descriptions
- File-level descriptions

Field-Level Descriptions: Field-level descriptions allow you to give the detailed characteristics of the smallest unit of data that can be handled by the data management support of the system. For a database file, a field-level description tells the system how data for the field is to be stored in the file. Through a field-level description you can specify how long the field should be in the database file and what type of data it is. You can also use field descriptions in a database file to tell the system what the data is to look like when it is presented to a program as the result of a read operation (or conversely, when the program presents data to the system on a write operation). The presentation format need not be the same as the storage format. The system uses the field-level descriptions to move field data into and out of a database file, performing mapping whenever necessary, and insuring that new field data is valid according to the description you gave.

A field description can be used in a display file to tell the system how the field is to be presented at the display device as well as how it will be presented from the program to the system on output and from the system to the program on input. You can specify where each field is relative to the start of a record and what the characteristics of each field will be while in the system. The system uses this information to determine where the data for each field should be acquired from for output. It also uses this information to determine where and how input from the device should be placed so the program can use it. Through a field description you can also specify such things as whether the field is an input-capable field or output only, what type of data is valid for the field, whether the field should be highlighted in some way, and where on the screen the field is to appear. The system uses field-level descriptions in preparing output data from the program for presentation at the display. It also uses these descriptions to enable fields for input and to validate input data before it is given to the program.

Field-level descriptions are valid for use with database, display, printer, and ICF files. They are not valid for use with tape, diskette, DDM, and save files. In the case of DDM files, although field-level descriptions are not valid with the DDM file itself, the field-level descriptions of the database file on the target system are used. In the case of tape and diskette, the system does not operate at the field level for data being processed through the file. The support operates at the record level only.

In the case of ICF files, you may use field-level descriptions. However, the system does not actively use these descriptions. For ICF files, the system operates at the record level. The system allows you to have field-level descriptions so you can centralize a standard description of the fields with the file. These standard field descriptions can be incorporated with any application program that will work with the file. This topic is discussed later in "Externally Described Data" on page 1-12.

The extent to which you use field-level descriptions is your choice. Using field-level descriptions allows the system to do more for you and also allows the system to insure to a greater extent that the data is valid. If you do not use field-level descriptions, the system only is able to handle data at the next-higher level, the record level. This is a larger unit of data, the details of which the system knows little about. Because the system does not know where the fields are in the record, functions like field mapping and field validity checking cannot be done.

Record-Level Descriptions: Record-level descriptions are used to tell the system what a particular record looks like. That is, it provides a record format. If field-level descriptions are also being used, the record format is given in terms of one or more field-level descriptions. You simply identify what fields make up the record format and the order of these fields within the record format. If field-level descriptions are not used, the record format is given by specifying how long the record is. In the case of save files, the record length is determined by the system. You cannot change this.

A record is the unit of transfer between the system and the application program. On a read operation, the program receives a record from the system. On a write operation, the program gives a record to the system. Therefore, a record-level description is required for all file types. Without knowing the record format, the system would not know how much data is being transferred on an operation.

When the record format is given simply as a *length*, the system must handle the entire record as a unit. It has no knowledge of what the data within the record looks like. It cannot operate on one part of the record one way and another part a different way. When a record format is given in terms of *field-level descriptions*, however, the system has detailed knowledge of the structure of a record and can treat each field in a unique way. Constructing record formats using field-level descriptions gives you many capabilities. For example, with database files it is possible to give a record format that describes how data is to be stored in the file. It is also possible to define another format over the same data that describes how a program is to see the data. Because the system understands the fields in each format and the relationship between the formats, it can perform data mapping between the format that the program works with and the format the data is stored under. This would not be possible if the record format was specified as a string of data of a certain length.

Another example is the use of these types of record formats in display files. Because each field in the record is described with detailed presentation characteristics and the location of each field within the record is also described, the system can present the data for each field in a unique way on the screen. One field can be an input-capable field while another can be an output field only. One field can be highlighted in some way while another is not. Such capabilities would not be possible if the system only knew the record as one long string.

File-Level Descriptions: File-level descriptions are descriptions that apply to the file as a whole. What can be specified at this level varies by type of file. For example, for a database file you can specify what record formats are valid for the file, how the data is to be organized (sequentially or by key), and if by key, what fields should be used as the key fields. For a display file you can specify what record formats are valid for the file, what device(s) the file should be usable with, and what graphic character set is to be assumed for the data that will be entered through the file. For a tape file you can specify block size, label information, and recording density. For a printer file you can specify the size of the page, characters per inch, and lines per inch. For an ICF file you can specify what devices the file should be used with and the maximum wait time for incoming data.

Constructing File Descriptions

A number of options are available on the system that you can use to enter file descriptions. This overview gives you a brief description of each of these options. For more information you can refer to the publications that are listed in the front of this manual. The options that are of interest are:

- CL commands
- Data description specifications (DDS)
- Interactive data definition utility (IDDU)
- Screen design aid (SDA)

CL Commands: There is a CL command which allows you to create each type of file supported by the system. On each such command are parameters that relate directly to file-level descriptions. For example, to create a tape file you use the Create Tape File (CRTTAPF) command. On this command you will find parameters which can be used to specify such file-level descriptions as block length, recording density, label information, and record length. For tape, diskette, save, and DDM files, using the CL commands is the only method available for entering file descriptions. For other files, the CL commands are made to work in conjunction with the next option, DDS.

DDS: DDS is a language that allows you to specify the file description for a database, printer, display, or ICF file. DDS allows you to specify file-level, record-level and field-level descriptions. To use this option you enter source statements that contain the descriptions. The source statements you enter must be placed into a source file. After you have completed the DDS for a file description, you can use that source as input to create the related file. You simply refer to the source file on the CL command that creates the file. The parameters on the CL command and the DDS source statements are merged to form the final file description for the file.

DDS provides statements by which you can enter all aspects of a file description. However, you should use only those statements that are valid for the type of file you want to create. If you should happen to use DDS statements that are invalid for the type of file you want to create, the system detects these and informs you of the error. Creation of a new file occurs only if no errors are detected that would make the new file invalid.

You may use any method you wish to enter the DDS source statements into the source file. An easy method is to use the source entry utility (SEU) as an interactive method to enter DDS. Since SEU supports the syntax checking of DDS source as it is being entered, using this utility can give you early detection of any errors you might make.

IDDU: IDDU is an interactive utility that can be used to create file descriptions for database files only. What you can do through IDDU overlaps with what you can do through DDS to a great extent, although IDDU does not support all the capability that is available through DDS. The major advantages of using IDDU over DDS are:

- IDDU is interactive. Instead of having to enter DDS source statements to specify your file description, you respond to interactive screens. The screens are simple and straightforward, thereby making this task easier.
- The specifications which you enter through IDDU are kept in a data dictionary where they can be used over and over in different combinations. This is in contrast to a DDS source file which really represents a single file description. You can create more than one file using the same DDS source file. However, the files end up with the same file description unless you explicitly go in and change the DDS source file in between file creations. With IDDU, you just go back to the data dictionary and reuse what is already there in a different way, thereby generating a new file description.

To create a file using the specification you entered through IDDU, you must also use IDDU. You cannot use the CL create commands that were previously discussed. IDDU does not generate DDS; it generates data dictionary entries. Data dictionary entries are different from DDS statements.

The specifications that you enter using IDDU are referred to as definitions rather than descriptions. This term is used to refer to the specifications as they exist in the data dictionary. When you create a file, the definitions in the data dictionary are used as input to generate a file description. The resulting file description is part of the file and is separate from the definitions which still remain in the dictionary.

SDA: SDA is an interactive utility that assists you in the design of displays (screens) for your applications. On the displays that are presented from SDA, you can dynamically lay out where each field should go on the application program display and describe the characteristics each field should have. You are able to see what the display will look like as you interact with SDA. On completion of the task, SDA transforms what you enter at the displays into DDS source statements which are then used to create the display file.

Externally Described Data

Historically, what records in a file looked like was not recorded anywhere outside of the programs that worked with those files. That is, what fields made up the record, how long each field was, and what type of data was in each field was specified only in the language statements of the program. Because there was no easy method to centralize a standard description of a file so that all programs would look at the data in the same way, mistakes were made. Some programs would process the data one way while other programs would process it a different way.

From discussions in the preceding sections, you have seen that on this system there is a way to centralize a standard description for database, display, printer, and ICF files (and indirectly for DDM files). It is the file description that is associated with any file of these types. By using a field-level description, you can produce a very detailed and standard description of both the file and any data that can be processed through the file. This standard description exists independently of any program that uses the file. On this system, the term *externally described data* is used to refer to the fact that the detailed description of the file and data is not embedded in any using program, but rather is contained in the file itself. To understand how the data is formatted, you need look only at the file description.

The system uses the file descriptions to process data on both input and output from the application. For data that is read by the program, the system uses the file description to determine how the data should be presented to the program. If any mapping is required, the system performs this mapping. The program will always see incoming data according to the file description. For data that is written by a program, the system uses the file description to determine the format of the data as the program gave it to the system. Then the system takes the data from the program and performs the necessary mapping and additional functions (such as highlighting a field on a display) to ensure that the data is in the correct format before storing it in a database file or presenting it at a device.

You can take advantage of externally described data also. Because the description is centralized, it is possible to incorporate the file description in the using program. Most programming languages supported by the system provide this capability. How this is done varies by programming language. Generally, you need to declare that the external descriptions are to be used. This is done by stating that the file is to be used as an externally described file. You need not redefine what the fields in the records look like. The language compiler or interpreter will go to the file and extract the file description. Those descriptions will then be incorporated into the program just as if you had declared these things explicitly in the source program.

There are several advantages to using externally described data:

- Increased programmer productivity. The language automatically describes the record layouts for you without additional coding. You need to describe records and fields only once (when the file is created), and can then refer to these fields within the program.
- Ease of file and program maintenance. When fields are added, deleted, or changed, it can be done in one place instead of maintaining the record layout in each program that uses the file.
- Increased data integrity. Since the fields and records are described in one central location, there is less chance of programming errors describing the data in the file to the program. All application programs using the file will have the same view of the data. Moreover, the system view of the data becomes the same as the application program view.
- Level checking provided. Level checking is an automatic method used when the program is run that determines if the file description has changed since the program was last compiled. Depending on the type of change, the program only may need to be recompiled without modification. This allows better control over program maintenance. There is more information on the level-checking function in “Detecting File Description Changes” on page 1-28.

Program-Described Data

Some file types do not allow a detailed description to be used since field-level descriptions are not supported. Even for those files that do have field-level descriptions, you are not required to use the external descriptions in your program. If you cannot use externally described data because of the file type or if you choose not to, then you must declare variables in your source program which define to the compiler or interpreter what the program thinks the data looks like. Such declarations are referred to as *program-described data*. That is, the description of the data related to the file is represented by the variable declarations you have coded in the program.

For tape, diskette, and save files, program-described data is the only option supported.

When externally described data is used, both the program and the system have the same view of the data. When program-described data is used, this may not be true. There are two cases to consider:

- If the file does not have any field-level descriptions with it, the system must operate at the record level. The only concern in this case is that the record length the program is using is the same as what the system is using. It need not be, but the system always operates with the record length it has. If this length is different from what the program is using, the system truncates or pads as appropriate. The exception to this is save files. For save files the program must operate with the same record length as the system or a severe error condition will result when the file is first opened.
- The second case that must be considered is when the file does have field-level descriptions, but the program has elected not to use them. In this case, even though the program does not use the field-level descriptions, the system does. The system still expects the program to present data according to the file description and, conversely, will provide data to the program according to the description. Should the program description be different from the file description, an error could result. This problem could arise if database files are used in this manner.

Changing a File Description

After a file has been created, it can be changed. What method is used to make a change depends on what part of the file description needs changing.

If you need to change the file-level description that was specified on the CL command used to create the file, there are corresponding CL commands for changing each file type.

If you need to change the file-level, record-level, or field-level information contained in DDS, you must first update the DDS for the file. You have the same options for how you update DDS as you had for entering it when the file was created. After the DDS is updated, you must delete the old file and use the appropriate CL command again to create the file with the updated DDS.

If you need to change both the CL command file-level descriptions and the DDS, you can do that by specifying the new values on the create CL command used to create the new file.

Whether programs that use a changed file will use the new file description the next time the program is run depends on what was changed in the file. If the file-level description was changed with a CL command, any program that uses the file will automatically use those new descriptions. If the DDS descriptions were changed and the program uses the file as a program-described file, then the system will use the new file description, but the program view of it may not be correct anymore, which could result in problems. If the DDS descriptions were changed and the program uses the file as an externally described file, then the record-level and field-level descriptions used when the program was compiled may not match the changed file. The system may detect such a mismatch when the program opens the file and gives the program an error condition. See "Detecting File Description Changes" on page 1-28.

The system also supports a way for you to temporarily change the file-level descriptions when a file is opened in a way that affects only the program opening the file. These temporary changes are discussed later in this chapter. See "Opening Files" on page 1-25.

Data Management Operations Overview

Data management supports many operations that high-level language programs can use to process data. These include the following, which are grouped by category:

- File Preparation

  OPEN Attaches a file to a program and prepares it for I/O operations. A file may be opened for any combination of read, write, update, or delete operations.

  ACQUIRE Attaches a device or establishes a communications session for an open file in preparation for I/O operations.

- Input/Output

  READ Transfers a record from the file to the program. The data is made available to the program once the read has been successfully completed.

  WRITE Transfers a record from the program to the file.

  WRITE-READ Combines the WRITE and READ operations as one operation.

  UPDATE Updates a record with changed data. The record must have been successfully read prior to the update operation.

  DELETE Deletes a record in a file. The record must have been successfully read prior to the delete operation.

- Commitment Control

  COMMIT Guarantees a group of changes are made as a complete transaction across multiple records and/or multiple files.

  ROLLBACK Rolls back a group of changes to the last commit point.

- Completion

  FEOD Positions the file at the last volume or at the end of data. For those programs processing files for output, the last buffer of data is written. For those programs processing files for input, an end-of-file condition is forced for the next input operation.

  RELEASE Detaches a device or a communications session from an open file. I/O operations can no longer be performed for this device or session.

  CLOSE Detaches a file from a program, ending I/O operations. Any remaining data in the output buffer that has not been written will be written prior to the completion of the close.

The operations listed above have certain restrictions based on file type and language support. For example, a program may not write to a file that has been opened for read only. Similarly, a read-by-key may not be issued for an ICF file.

Since file overrides can occur during processing (see Chapter 2, "Overrides and File Redirection" for additional information), an operation may not be allowed for the type of file that is ultimately being processed.

Table 1-1 lists the file types and the main operations that are allowed. There is additional function supported for some file types that is accomplished by additional operations or modifications to these operations. For information on these additional functions and how the operations given here apply to display, printer, tape, and diskette files, refer to the appropriate chapter elsewhere in this manual. For equivalent information for database, ICF, DDM, and save files, refer to the *Database Guide*, the *Communications Programmer's Guide*, the *DDM User's Guide*, and the *Backup and Recovery Guide*, respectively.

Table 1-1. File Types and Their Main Operations

	File Types							
Operation	**Database**	**Diskette**	**Tape**	**Printer**	**Display**	**ICF**	**DDM**	**Save**
OPEN								
Read	X	X	X	-	X	X	X	X
Write	X	X	X	X	X	X	X	X
Update	X	-	-	-	X*	-	X	-
Delete	X	-	-	-	X*	-	X	-
READ								
By relative record number	X	-	-	-	X*	-	X	-
By key	X	-	-	-	-	-	X	-
Sequential	X	X	X	-	X	X	X	X
Previous	X	-	X	-	X*	-	X	-
Next	X	X	X	-	X	X	X	X
Invited Device	-	-	-	-	X	X	-	-
WRITE-READ	-	-	-	-	X	X	-	-
WRITE								
By relative record number	X	-	-	-	X*	-	X	-
By key	X	-	-	-	-	-	X	-
Sequential	X	X	X	X	X	X	X	X
FEOD	X	X	X	X	-	-	X	X
UPDATE								
By relative record number	X	-	-	-	X*	-	X	-
By key	X	-	-	-	-	-	X	-
DELETE								
By relative record number	X	-	-	-	-	-	X	-
By key	X	-	-	-	-	-	X	-
ACQUIRE	-	-	-	-	X	X	-	-
RELEASE	-	-	-	-	X	X	-	-
COMMIT	X	-	-	-	-	-	-	-
ROLLBACK	X	-	-	-	-	-	-	-
CLOSE	X	X	X	X	X	X	X	X

* Operation allowed only for subfile record formats

Table 1-2 maps the OS/400-supported operations given in Table 1-1 on page 1-17 to the high-level language operations (RPG/400[5], PL/I, COBOL/400[5], BASIC, PASCAL, and C/400[5]) supported on the system. For additional information on each operation and how it correlates to the file declaration in the program, see the appropriate language manual. Note that not all OS/400 operations are supported in all languages.

Table 1-2. High-Level Languages and Their OS/400 Operations

	High-Level Languages					
Operation	**RPG/400**	**PL/I**	**COBOL/400**	**BASIC**	**PASCAL**	**C/400**
OPEN						
Read	OPEN, primary file	OPEN INPUT	OPEN INPUT	OPEN INPUT	RESET, GET, READ, READLN	fopen
Write	OPEN, primary file	OPEN OUTPUT	OPEN OUTPUT, OPEN EXTEND	OPEN OUTPUT	REWRITE, WRITE, WRITELN	fopen
Update	OPEN, primary file	OPEN UPDATE	OPEN I-O	OPEN OUTIN	UPDATE	fopen
Delete	OPEN, primary file	OPEN UPDATE	OPEN I-O	OPEN OUTIN	UPDATE	fopen
READ						
By relative record number	READ, CHAIN	READ KEY	READ	READ REC	GET, READ	-
By key	READ, READE, CHAIN	READ KEY	READ KEY	READ KEY	-	-
Sequential	READ, primary file	READ NEXT, GET	READ	READ NEXT, GET	GET, READ, READLN	fread, fgetc, fgets QXXPGMDEV QXXFORMAT
Previous	READP	READ PRV	READ	READ PRIOR	GET, READ, READLN	-
Next	READ, READE	READ NXT, GET	READ, READ NEXT	READ NEXT GET	GET, READ, READLN	fread
Invited Device	READ	-	READ	-	-	QXXREADINVDEV
WRITE-READ	EXFMT	-	-	-	-	-
WRITE						
By relative record number	WRITE, EXCPT primary file	WRITE KEY	WRITE	WRITE REC	PUT, WRITE, WRITELN	-
By key	WRITE, EXCPT	WRITE KEY	WRITE	WRITE	-	-
Sequential	WRITE, EXCPT primary file	WRITE, PUT	WRITE	WRITE	PUT, WRITE, WRITELN	fread, fputc, fputs QXXPGMDEV QXXFORMAT
FEOD	FEOD	-	-	-	-	-
UPDATE						
By relative record number	UPDAT, primary file	REWRITE KEY	REWRITE	REWRITE REC	PUT, WRITE WRITELN	-
By key	UPDAT, primary file	REWRITE KEY	REWRITE	REWRITE KEY	-	-
DELETE						
By relative record number	DELET, primary file	DELETE KEY	DELETE	DELETE REC	-	-
By key	DELET, primary file	DELETE KEY	DELETE	DELETE KEY	-	-
ACQUIRE	ACQ	-	ACQUIRE	-	-	QXXACQUIRE
RELEASE	REL	-	DROP	-	-	QXXRELEASE
COMMIT	COMIT	PLICOMMIT subroutine	COMMIT	-	use CL COMMIT	QXXCOMMIT
ROLLBACK	ROLBK	PLIROLLBACK subroutine	ROLLBACK	-	use CL ROLLBACK	QXXROLLBCK
CLOSE	CLOSE, RETRN	CLOSE, STOP	CLOSE, STOP RUN, CANCEL	CLOSE, END	CLOSE, END	fclose

[5] C/400, COBOL/400, and RPG/400 are trademarks of the International Business Machines Corporation.

Security Considerations

This section describes some of the file security functions. The topics covered include the authorizations needed to use files and considerations for specifying these authorities when creating a file. For more information about using the security function on the system, see *Security Concepts and Planning*.

File Object Authority

The following describes the types of authority that can be granted to a user for a file:

Object Operational Authority: Object operational authority is required to:

- Open the file for processing. You must also have read authority to the file. For device files that are not using spooling, you must have object operational and also all data authorities to the device.
- Compile a program which uses the file description.
- Display the file description.
- Delete the file.
- Transfer ownership of the file.
- Grant and revoke authority.
- Change the file description.
- Move or rename the file.

Object Existence Authority: Object existence authority is required to:

- Delete the file.
- Save, restore, and free the storage of the file.
- Transfer ownership of the file.

Object Management Authority: Object management authority is required to:

- Grant and revoke authority. You can grant and revoke only the authority that you already have.
- Change the file description.
- Move or rename the file.

File Data Authorities

Data authorities can be granted to a file. You need:

- Read authority to open any file for input, compile a program using the file, or display the file description.
- Add authority to open a database file and save file for output.
- Update authority to open a database file for update.
- Delete authority to open a database file for delete.

For files other than database and save files, the add, update, and delete authorities are ignored.

Authorities Required for File Operations

Table 1-3 lists the file object authority and the data authority required for file functions. This is the same information that was presented in the previous two sections, but it is listed by function rather than by authority.

Table 1-3. Object Authority and Data Authority Required for File Operations

Function	Object Operational	Object Existence	Object Management	Read	Add	Update	Delete
Open, I/O, close file[1]	X			X	X[2]	X[3]	X[3]
Compile a program using the file description	X			X			
Display file description	X			X			
Delete file	X	X					
Save/restore		X					
Transfer ownership	X	X					
Grant/revoke authority	X		X				
Change file description	X		X				
Move file	X		X				
Rename file	X		X				

[1] For device files that are not using spooling, you must also have object operational and all data authorities to the device.

[2] Open for output for database and save files.

[3] Open for update or delete for database files.

Specifying Authorities when Creating Files

When you create a file, you can specify public authority through the AUT parameter on the create command. Public authority is authority available to any user who does not have specific authority to the file or who is not a member of a group that has specific authority to the file. That is, if the user has specific authority to a file or the user is a member of a group with specific authority, then the public authority is not checked when a user performs an operation to the file. Public authority can be specified as:

- *CHANGE. All users that do not have specific user or group authority to the file have authority to use the file. The *CHANGE value is the default public authority. *CHANGE grants any user object operational and all data authorities.
- *USE. All users that do not have specific user or group authority to the file have authority to use the file. *USE grants any user object operational and read data authority.
- *EXCLUDE. Only the owner, security officer, users with specific authority, or users who are members of a group with specific authority can change or use the file.

- *ALL. All users that do not have specific user or group authority to the file have all data authorities along with object operational, object management, and object existence authorities.
- Authorization list name. An authorization list is a list of users and their authorities. The list allows users and their different authorities to be grouped together.

You can use the Edit Object Authority (EDTOBJAUT), Grant Object Authority (GRTOBJAUT), or Revoke Object Authority (RVKOBJAUT) commands to grant or revoke the public authority of a file.

Sharing Files in the Same Job

By default, the system lets one file be used by many users and more than one job at the same time. The system allocates the file and its associated resources for each use of the file in such a way that conflicting uses are prevented.

Within the same job, files can be shared if one program opens the same file more than once or if different programs open the same file. Even though the same file is being used, each open operation creates a new path from the program to the data or device, so that each open represents an independent use of the file.

Historically, the file sharing just discussed was the limit of the file sharing available. However, OS/400 data management support offers another closer level of sharing within a job that allows more than one program to share the same path to the data or device.

This level of sharing is available by specifying the SHARE parameter on the create file, change file, and override file commands. Using the SHARE parameter allows more than one program to share the file status, positions, and storage area, and can improve performance by reducing the amount of main storage the job needs and by reducing the time it takes to open and close the file.

Using the SHARE(*YES) parameter lets an open data path (ODP) be shared between two or more programs running in the same job. An open data path is the path through which all input/output operations for the file are performed. It connects the program to a file. If not specified otherwise, every time a file is opened a new open data path is built. You can specify that if a file is opened more than once and an open data path is still active for it in the same job, the active ODP for the file can be used with the current open of the file, and a new open data path does not have to be created. This reduces the amount of time required to open the file after the first open, and the amount of main storage required by the job. SHARE(*YES) must be specified for the first open and other opens of the same file for the open data path to be shared. A well-designed (for performance) application will normally do a shared open on database files that will be opened in multiple programs in the same job. Specifying SHARE(*YES) for other files depends on the application.

Sharing files allows you to have programs within a job interact in ways that would otherwise not be possible. However, you should read the considerations for open, I/O, and close in this section and in the appropriate manuals for all the file types to understand how this support works and the rules programs must follow to use it correctly.

Note: Most high-level language programs process an open or a close operation independent of whether or not the file is being shared. You do not specify that the file is being shared in the high-level language program. You indicate that the file is being shared in the same job through the SHARE parameter. The SHARE parameter is specified only on the create, change, and override file commands. Refer to your appropriate language manual for more information.

Open Considerations for Files Shared in a Job

The following items should be considered when opening a file that is shared in the same job by specifying SHARE(*YES).

- You must make sure that when the shared file is opened for the first time in a job, all the open options that are needed for subsequent opens of the file are specified. If the open options specified for subsequent opens of a shared file do not match those specified for the first open of a shared file, an error message is sent to the program. (You can correct this by making changes to your program to remove any incompatible options.)

  For example, PGMA is the first program to open FILE1 in the job and PGMA only needs to read the file. However, PGMA calls PGMB which will delete records from the same shared file. Because PGMB will delete records from the shared file, PGMA will have to open the file as if it, PGMA, is also going to delete records. You can accomplish this by using the correct specifications in the high-level language. (In order to accomplish this in some high-level languages, you may have to use file operation statements that are never run. See your appropriate language manual for more details.)

- Sometimes sharing a file within a job is not possible. For example, one program may need records from a file in arrival sequence and another program may need the records in keyed sequence. Or, you may use the same file for printing output, but want the output from each program to be produced separately. In these situations, you should not share the open data path. You would specify SHARE(*NO) on the override command to ensure that the file was not shared within the job.

- If debug mode is entered with UPDPROD(*NO) after the first open of a shared file in a production library, subsequent shared opens of the file share the original open data path and allow the file to be changed. To prevent this, specify SHARE(*NO) on the override command before opening files while debugging your program.

- The use of commitment control for the first open of a shared file, requires that all subsequent shared opens also use commitment control.

- If you did not specify a library name in the program or the override command (*LIBL is used), the system assumes that the library list has not changed since the last open of the same shared file with *LIBL specified. If the library list has changed, you should specify the library name on the override command to ensure that the correct file is opened.

- Overrides and program specifications specified on the first open of the shared file are processed. Overrides and program specifications specified on subsequent opens, other than those that change the file name or the value specified on the SHARE or LVLCHK parameters on the override command, are ignored.

Input/Output Considerations for Files Shared in a Job

The system uses the same input/output area for all programs sharing the file, so the order of the operations is sequential regardless of which program does the operation. For example, if Program A is reading records sequentially from a database file and it reads record 1 just before calling Program B, and Program B also reads the file sequentially, Program B reads record 2 with the first read operation. If Program B then ends and Program A reads the next record, it receives record 3. If the file was not being shared, Program A would read record 1 and record 2, and Program B would read record 1.

For device files, the device remains in the same state as the last I/O operation.

For display and ICF files, programs other than the first program that opens the file may acquire more display or program devices or release display or program devices already acquired to the open data path. All programs sharing the file have access to the newly acquired devices, and do not have access to any released devices.

Close Considerations for Files Shared in a Job

The processing done when a program closes a shared file depends on whether there are other programs currently sharing the open data path. If there are other programs, the main function that is performed is to detach the program requesting the close from the file. For database files, any record locks held by the program are also released. The program will not be able to use the shared file unless it opens it again. All other programs sharing the file are still attached to the ODP and can perform I/O operations.

If the program closing the file is the last program sharing the file, then the close operation performs all the functions it would if the file had not been opened with the share option. This includes releasing any allocated resources for the file and destroying the open data path.

The function provided by this last close operation is the function that is required for recovering from certain run-time errors. If your application is written to recover from such errors and it uses a shared file, this means that all programs that are attached to the file when the error occurs will have to close the file. This may require returning to previous programs in the program stack and closing the file in each one of those programs.

Allocating File Resources

When a high-level language program uses a file, several operations require that the system allocate the resources needed to perform that operation. This is generally done to ensure that multiple users do not use the file in conflicting ways. For example, the system will not allow you to delete a file while any application program is using it. This is prevented because when the file was opened, the system obtained a lock on the file. The delete file operation also attempts to get a lock on the file and is unsuccessful because the program using the file still has the lock from when the file was opened, and the locks conflict.

When you write a high-level language program, you should be aware of what resources are allocated for each file type. Normally, the system will perform the allocation whenever an operation is requested that requires it. For example, the resources for each file used in a program are allocated when the file is opened. If you prefer to ensure that all the resources that are needed by a program are available before the program is run, you may use the Allocate Object (ALCOBJ) CL command in the job prior to running the program. In particular, the ALCOBJ command can allocate database files and most devices.

Examples of operations that require resource allocation are:

- Open
- Acquire
- Starting a program on a remote system

The file resources that must be allocated depend on the type of file and the operation being performed. File resources consist of the following:

- Open
  - For printer and diskette files that are spooled (SPOOL(*YES)), the file resources include the file description, the specified output queue, and storage in the system for the spooled data. Because the data is spooled, the device need not be available.
  - For database files, the file resources consist of the entire file, including the file, member, data, and the associated access path.
  - For printer and diskette files that are not spooled (SPOOL(*NO)) as well as for tape files, display files, and some ICF files, the file resources include the file description and the device. For ICF files that use APPC, APPN, or intrasystem communications, the file resources include the file description and the session resources associated with the device.
  - For save files, the file resources consist of the entire file, including the file and data.
  - For DDM files, the file resources include the file description and the session resources associated with the device.
- Acquire operation

  For display files and ICF files not using APPC/APPN, or intrasystem communications, the device is allocated as a resource. For ICF files using APPC/APPN, or intrasystem communications, resources include the session resources associated with the device.
- Starting a program on a remote system

  Session resources needed for APPC and APPN.

When allocating resources, the system waits for a predefined time if the resources are not immediately available. If the resources do not become available within the time limit, an error is generated. If you are using the ALCOBJ command, the command fails. If your program is performing a file operation, that operation fails and an error message is sent to the program message queue. You may attempt to use the error handling functions of your high-level language to try the operation again. For example, if an open operation fails because another job is using the device associated with the file, you could retry the open operation a specified number of times, in the hope that the other job would finish with the device and your program would then be able to use it.

The length of time that the system waits when allocating resources is specified on the ALCOBJ command and on the WAITFILE parameter of the CL command used to create the file. If the ALCOBJ command is used prior to running a program, then the value of the WAITFILE parameter does not matter, because the resources will be available.

The following chart describes the values allowed for the WAITFILE parameter:

Values	**Definition**
*IMMED	This value specifies that no wait time is allowed. An immediate allocation of the file resources is required.
*CLS	The job default wait time is used as the wait time for the file resources to be allocated.
number-of-seconds	Specify the maximum number of seconds that the program is to wait for the file resources to be allocated. Valid values are 1 through 32767 (32 767 seconds).

If your application has error handling procedures for handling device errors occurring on device files, you should specify a value of something other than *IMMED to allow the system to recover from the error. The allocation of resources requested by your program on an open or acquire operation that allows your program to recover from the error will not be successful until the system recovery procedures have been completed for the device.

Opening Files

When an application wants to use a file, it does so by referring to that file by name. The file description for that file will then control how the program and the system will interact.

An application program has an option as to how the file description is used. The program may choose to use the description as it currently exists. In this case, the system uses the file description as is, without any change. A number of parameters contained in a file description can be changed, however. Therefore, the second option the application has is to change some or all of these parameters. A change made to a file description can be permanent or temporary. Permanent changes are discussed in "Changing a File Description" on page 1-14.

Temporary changes can provide greater flexibility to the application. Temporary changes are made when the program is first establishing a path to the file by opening the file. Temporary changes can be made in one of two ways:

- By information that is specified within the program itself and which is passed as parameters on the open operation.
- By using override CL commands in the input stream that is used to set up the run-time environment for the application.

The ability to use the first way depends very much on which programming language is used to write the program. Some programming languages do not allow you to control the open process to any great extent. These languages do the open process more or less automatically and control what information gets passed. Other languages allow you to have greater control over the open process.

The second option can be used regardless of which programming language you use. Override CL commands are provided for each file type. By including override commands with the application, you may temporarily change the file description in a file that the program wants to use.

Both options can be used together. Some parameters can be changed by information contained in the application while others can be changed by using an override command. The same parameter may be changed from both places. The operating system follows this order when making temporary changes to a file:

1. The file description provides a base of information.
2. Change information received from the application during the open process is applied first to the base information.
3. Change information found in the override command is applied last. If the same information is changed from both places, the override has precedence.

Temporary changes are seen only by the application that causes the change to be made. The file, as seen by another application, remains unchanged. In fact, two applications may use the same file at the same time, and each may change it temporarily according to its needs. Neither application is aware the other has made a temporary change. Figure 1-6 and Figure 1-7 illustrate the permanent and temporary change processes.

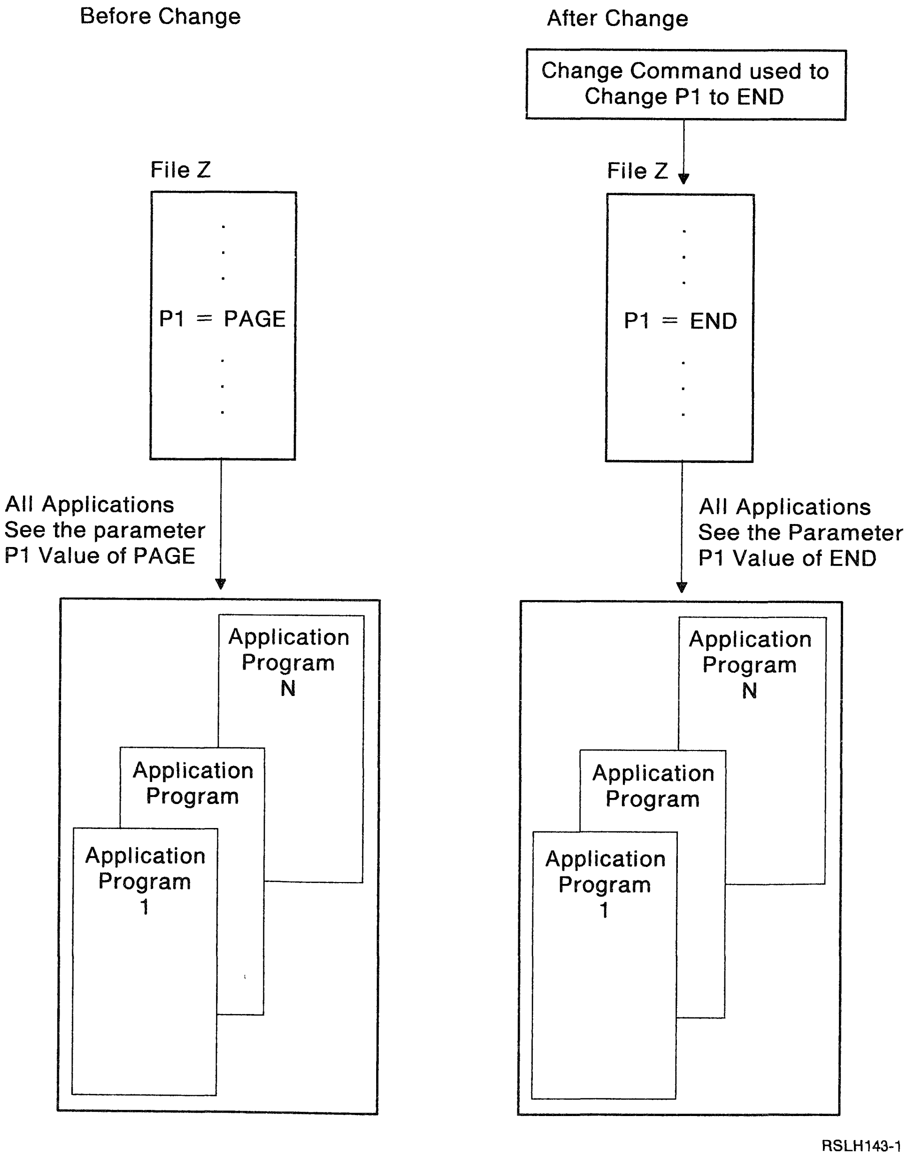


Figure 1-6. Permanently Changing a File

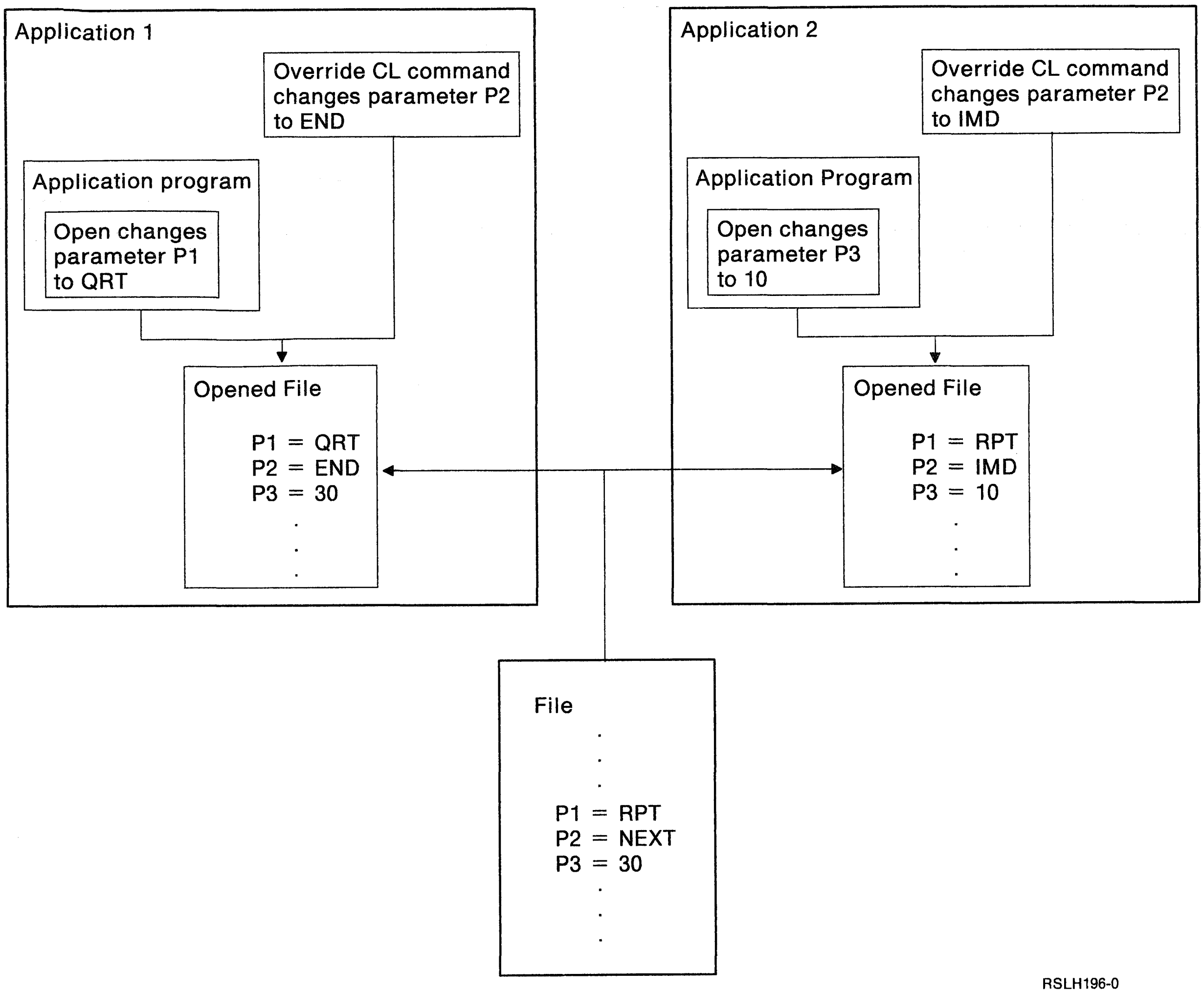


Figure 1-7. Temporarily Changing a File

Once an application establishes a connection between itself and the file by opening the file, it can then proceed to use the file for either input or output operations. In the case of a database file, the open process establishes a path between the application and the actual database file. For device files, a path is established between the application and the actual device, or to a spooled file if the spooling attribute is active for the device file. In all cases, the application is connected to what it wants to use, and those connections determine what input or output operations are valid. Not all operations are valid with all file types. The application must be aware of what file types it uses and then use only those operations which are valid for those types.

Detecting File Description Changes

When a program that uses externally described files is compiled, the high-level language compiler extracts the record-level and field-level descriptions for the files referred to in the program and makes those descriptions part of the compiled program. When you run the program, you can verify that the descriptions with which the program was compiled are the current descriptions.

The system assigns a unique level identifier for each record format when the file it is associated with is created. The system uses the following information to determine the level identifier:

- Record format name
- Field name
- Total length of the record format
- Number of fields in the record format
- Field attributes (for example, length and decimal positions)
- Order of the field in the record format

Display, printer, and ICF files may also use the number of and order of special fields called indicators to determine the level identifier.

If you change the DDS for a record format and change any of the items in the preceding list, the level identifier changes.

To check the record format identifiers when you run the program, specify LVLCHK(*YES) on the create or change file commands.

The level identifiers of the file opened and the file description that is part of the compiled program are compared when the file is opened and LVLCHK(*YES) is specified. The system does a format-by-format comparison of the level identifiers. If the identifiers differ or if any of the formats specified in the program do not exist in the file, a message is sent to the program to identify the condition.

When the identifiers differ, this means that the format was changed and the changes may affect your program. You can compile the program again so that the changes are included, or you can determine if the changes affect your program before deciding what action to take. There are several CL commands available to you to check the changes. You can use the Display File Field Description (DSPFFD) command to display the record-level and field-level descriptions or, if you have the source entry utility (SEU), you can display the source file containing the DDS for the file. The format level identifier defined in the file can be displayed by the Display File Description (DSPFD) or the DSPFFD commands. The format level identifier which was used when the program was created can be displayed by the Display Program References (DSPPGMREF) command.

If the changes do not affect your program, you can enter an override command with LVLCHK(*NO) specified so that you can process your program. If LVLCHK(*NO) is specified, the system does not perform the level identifier check when the file is opened. For example, if a field is added to the end of a record format, but the program does not use the new field, the program does not have to be created again, even though the record identifier has changed.

There are also some changes to a file description that will not cause an error when the file is opened. These can be caused because the record format identifiers did not change or because your program does not use the changed formats. Formats can be added to or removed from a file without affecting existing programs that do not use the added or deleted formats.

Even though the level identifier does not change, some DDS functions that you add or delete could require changes in the logic of your program. You should review the functions you added or deleted to determine whether changes are required to the program logic.

Normally, the use of LVLCHK(*YES) is a good file integrity practice. The use of LVLCHK(*NO) can produce results that cannot be predicted.

Open and I/O Feedback Area

The system keeps track of the status of a file in feedback areas once it is successfully opened. As operations are performed on a file, these feedback areas are updated to reflect the latest status. These feedback areas give you greater control over applications and provide important information when errors occur.

The feedback areas are established at open time, and there is one feedback area for each open file. One exception is for shared files, which share feedback areas as well as the data path between the program and the file. For more information on shared opens, see "Sharing Files in the Same Job" on page 1-21.

Some high-level languages on the system allow you to access the status and other information about the file against which operations are being performed. There are two feedback areas of interest to you:

- Open feedback area

  This area contains information of a general nature about the file after it has been successfully opened. Examples include the name and library of the file and the file type. See "Open Feedback Area" on page A-1 for a complete list of the information that can be retrieved from the open feedback area. In addition to general information about the file, file-specific information is also contained in the open feedback area after the file is opened. The applicable fields depend on the file type.

  The open feedback area also contains information about each device or communications session defined for the file.

- Input/output feedback area

  There are two sections of the I/O feedback area:

  - Common area

    This area contains information about I/O operations that were performed on the file. This includes the number of operations and the last operation. See "I/O Feedback Area" on page A-11 for a complete list of the information that can be retrieved from the common I/O feedback area.

  - File-dependent feedback area

    This area contains file-specific information for display, database, printer, and ICF files; for example, the major/minor return code and amount of data received from the device. See "I/O Feedback Area for ICF and Display Files" on page A-14, "I/O Feedback Area for Printer Files" on page A-18, and "I/O Feedback Area for Database Files" on page A-19 for a complete list of the information that can be retrieved from the file-dependent I/O feedback area.

The above information areas can be useful to you. For example, when an error occurs with a device file, the program could determine predefined error handling operations based on the major/minor return code in the file-dependent feedback area. If data is being received from a communications device and the application on the other end sends an error, the program could determine that the next operation should be to wait until the next block of data is sent indicating the error. Possibly, the next operation may be to close the file and end the conversation with the application on the other side or wait for the next request from the application.

Another way might include detecting what type of file was actually opened to determine the type of operations that are allowed. If the file type is printer, only output operations would be allowed.

Error Handling

This section describes error conditions that an application program may encounter during its operation and the provisions that can be made within the program itself to attempt to deal with these conditions. The *CL Programmer's Guide* discusses how to use the debug functions to resolve unexpected errors encountered in the application programs. The chapter on handling problems in the *Operator's Guide* describes the programs that are available for analyzing and reporting system errors and hardware failures.

Errors can be detected when a file is opened, when a program device is acquired or released, during I/O operations to a file, and when the file is closed. When appropriate, the system will automatically try to run a failing operation again, up to a retry limit. When a retry is successful, neither operator nor program action is required. Errors that can affect the processing of the program may be reported in any or all of the following ways:

- A notify, status, diagnostic, or escape message may be sent to the program message queue of the program using the file. These messages may also appear in the job log, depending on the message logging level set for the job.
- A file status code may be returned by the high-level language.
- A major/minor return code is returned in the I/O feedback area for ICF, display, and printer files.
- A notify, status, diagnostic, or escape message may be sent to the operator message queue (QSYSOPR) or the history message queue (QHST).
- Information regarding the error may be saved in the system error log for use by the problem analysis and resolution programs.
- An alert message may be sent to an operator at another system in the network.
- The normal program flow may be interrupted and control may be transferred to an error-handling subroutine, or other language operations may occur. For additional information about how to handle run-time errors, see the appropriate high-level language manual.

Only some of these are significant to a program that is attempting error recovery.

Not all file errors allow programmed error recovery. Some errors are considered permanent; that is, the file, device, or program cannot work until some corrective action is taken. This might involve resetting the device by varying it off and on again, or correcting an error in the device configuration or the application program. Some messages and return codes are used to inform the user or the application program of conditions that are information rather than errors, such as change in the status of a communications line, or system action taken for an unexpected condition. In many cases, it is possible for the application program to test for an error condition and take some preplanned recovery action which allows the program to continue without intervention from the operator.

Messages and Message Monitors

Displayed messages are the primary source of information for an operator or a programmer who is testing a new application. A message usually contains more specific information than the file status code, the indicators, or the major/minor return code. The control language allows messages to be monitored so that the CL program can intercept a message and take corrective action. See the *CL Programmer's Guide* for more information about message types and message monitors. In most high-level languages, either the file status code or major/minor return code (described in the following section) is a more convenient source of information.

Message numbers are assigned in categories to make it easier for a program to monitor for any of a group of related messages. The following message number ranges are assigned for file error messages:

Table 1-4. OS/400 Data Management Message Number Ranges

Message IDs	Operation	Message Type
CPF4001 – 40FF	Open	Diagnostic and status.
CPF4101 – 43FF	Open	Escapes that make the file unusable.
CPF4401 – 44FF	Close	Diagnostic and status.
CPF4501 – 46FF	Close	Escapes that make the file unusable.
CPF4701 – 48FF	I/O, Acquire, and Release	Notify with a default reply of cancel, status and escapes that do not make the file or device unusable.
CPF4901 – 49FF	I/O, Acquire, and Release	Notify with a default reply of ignore.
CPF5001 – 50FF	I/O, Acquire, and Release	Notify with a default reply of cancel.
CPF5101 – 53FF	I/O, Acquire, and Release	Escapes that make the file or device unusable.
CPF5501 – 56FF	I/O, Acquire, and Release	Escapes that make the file or device unusable.

Some status messages, CPF4018 for example, are preceded by a diagnostic message that provides additional information. Diagnostic messages may be kept in the job log, depending on the message logging level of the job. If a CL program monitors for CPF4018, CPF5041, or similar messages, it can retrieve the accompanying diagnostic message from the program message queue.

If an error occurs for which an escape message is issued and the message is not monitored, your program will be ended and the message displayed for the operator. Status messages may also be monitored, but if they are not monitored, the program continues. Most high-level languages except CL monitor for all the file errors that are likely to be encountered, and provide some standard recovery. Depending on the severity of the error, the high-level language may simply end the program and issue a message of its own. Alternatively, the application programmer may code an error recovery routine to handle errors that are anticipated in that particular application.

Within these error-handling routines, it is usually necessary to examine the file status or major/minor return codes to determine the cause of the error. The manuals for the language you are using explain how to access file status and major/minor return codes. The language manuals also explain the file status codes as they are defined for each language.

Major/Minor Return Codes

Major/minor return codes are used to report errors and certain status information for ICF, display, and printer files. They are not used for other files. They are usually stated as four characters: the first two referring to the major code and the second two referring to the minor code. The major code indicates the general type of error, and the minor provides further detail. Minor codes, except zero, have the same or a similar meaning, regardless of the major code with which they are combined.

The application program can test the return code after each I/O operation. If the major return code is 00, the operation completed successfully and the minor return code contains status information that indicates whether a read or a write operation should be performed next. A major return code of 04 or above indicates that an error occurred. The program may test for any specific errors for which programmed recovery is attempted. The application program may test for a specific condition by comparing the major and minor codes as a unit, or may identify a class of conditions by testing the major code alone.

Most major/minor return codes are accompanied by any one of several message numbers, for which the typical recovery action is similar. File status codes are defined by the individual languages and may be set based on the major/minor return codes.

The following chart defines the major return codes. Appendix C, "Display File Return Codes" and Appendix D, "Printer File Return Codes" contain specific definitions of the major/minor return codes as they are used for display files and printer files and the message numbers associated with each. Similar specific definitions for each of the communication types valid on an ICF file can be found in the manuals for each communication type.

Table 1-5 (Page 1 of 2). Major Return Code Definitions

Code	Definition
00	The operation requested by your program completed successfully. The minor includes state information, such as change direction.
02	Input operation completed successfully, but job is being ended (controlled). The minor includes state information.
03	Successful input operation, but no data was received. The minor includes state information.

Table 1-5 (Page 2 of 2). Major Return Code Definitions

Code	Definition
04	Error occurred because an output operation was attempted while data was waiting to be read.
08	An acquire operation failed because the device has already been acquired or the session has already been established.
11	A read-from-invited-program-devices operation failed because no device or session was invited.
34	An input exception occurred. The data length or record format was not acceptable for the program.
80	A permanent (nonrecoverable) system or file error occurred. Programmer action is required to correct the problem.
81	A permanent (nonrecoverable) device or session error occurred during an I/O operation.
82	A device or session error occurred during an open or acquire operation. Recovery may be possible.
83	A device or session error occurred during an I/O operation. Recovery may be possible.

Actions for Error Recovery

The following sections describe the error recovery action that is appropriate for each group of major return codes.

Normal Completion

A major/minor return code of 0000 indicates that the operation requested by your program was completed successfully. Most of the time, no message is issued. In some cases, a diagnostic message might be used to inform the user of some unusual condition that the system was able to handle, but which might be considered an error under some conditions. For example, a parameter that is not valid might be ignored, or some default action taken.

For communications devices, a major return code of 00, indicating successful completion with data received, is accompanied by a minor return code that indicates what operation the application program is expected to perform next. The nonzero minor does not indicate an error. No message is issued.

Completion with Exceptions

Several rather specific major return codes have been assigned to conditions for which a specific response from the application program is appropriate.

A major return code of 02 indicates that the requested input operation completed successfully, but the job is being ended (controlled). The application program should complete its processing as quickly as possible. The controlled cancel is intended to allow programs time to end in an orderly manner. If your program does not end within the time specified on the ENDJOB command, the job will be ended by the system without further notice.

A major return code of 03 indicates that an input operation completed successfully without transferring any data. For some applications, this might be an error condition, or it might be expected when the user presses a function key instead of entering data. It might also indicate that all the data has been processed, and the application program should proceed with its completion processing. In any case, the contents of the input buffer in the program should be ignored.

A major/minor code of 0309 is used to indicate that no data was received *and* the job is being ended (controlled). A major/minor code of 0310 indicates that there is no data because the specified wait time has ended. Other minor return codes accompanying the 02 or 03 major code are the same as for a 00 major code, indicating communications status and the operation to be performed next.

A major return code of 04 indicates that an output exception occurred. Specifically, your program attempted to send data when there was data waiting to be received. This is probably the result of not handling the minor return code properly on the previous successful completion. Your program can recover by simply receiving the incoming data and then repeating the write operation.

A major return code of 34 indicates that an input exception occurred. The received data was either too long or incompatible with the record format. The minor return code indicates what was wrong with the received data, and whether the data was truncated or rejected. Your program can probably handle the exception and continue. If the data was rejected, you may be able to read it by specifying a different record format.

Two other return codes in this group, 0800 and 1100, are both usually the result of application programming errors, but are still recoverable. 0800 indicates that an acquire operation failed because the device has already been acquired or the session has already been established. 1100 indicates that the program attempted to read from invited devices with no devices invited. In both cases, the request that is not valid is ignored, and the program may continue.

No message is issued with a 02 major code or most minor codes with the 03 major code, but the other exceptions in this group are usually accompanied by a message in the CPF4701 – CPF47FF or CPF5001 – CPF50FF range.

Permanent System or File Error

A major return code of 80 indicates a serious error affecting the file. The application program must close the file and reopen it before attempting to use it again, but recovery is unlikely until the problem causing the error is found and corrected. To reset an error condition in a shared file by closing it and opening it again, all programs sharing the open data path must close the file. This may require returning to previous programs in the program stack and closing the shared file in each of those programs. The operator or programmer should refer to the text of the accompanying message to determine what action is appropriate for the particular error.

Within this group, several minor return codes are of particular interest. A major/minor code of 8081 indicates a serious system error for which an APAR probably will be required. The message sent with the major/minor return code may direct you to run the Analyze Problem (ANZPRB) command to obtain more information.

A major/minor code of 80EB indicates that invalid or incompatible options were specified in the device file or as parameters on the open operation. In most cases you can close the file, end the program, correct the parameter that is not valid with an override command, and run the program again. The override command affects only the job in which it is issued. It allows you to test the change easily, but you may eventually want to change or re-create the device file as appropriate to make the change permanent.

Permanent Device or Session Error on I/O Operation

A major return code of 81 indicates a serious error affecting the device or session. This includes hardware failures affecting the device, communications line, or communications controller. It also includes errors due to a device being disconnected or powered off unexpectedly and abnormal conditions that were discovered by the device and reported back to the system. Both the minor return code and the accompanying message provide more specific information regarding the cause of the problem.

Depending on the file type, the program must either close the file and open it again, release the device and acquire it again, or acquire the session again. To reset an error condition in a shared file by closing it and opening it again, all programs sharing the open data path must close the file. In some cases, the message may instruct you to reset the device by varying it off and on again. It is unlikely that the program will be able to use the failing device until the problem causing the error is found and corrected, but recovery within the program may be possible if an alternate device is available.

Some of the minor return codes in this group are the same as those for the 82 major return code. Device or line failures may occur at any time, but an 81 major code occurs on an I/O operation. This means that your program had already established a link with the device or session. Therefore, some data may have been transferred, but when the program is started again, it starts from the beginning. A possible duplication of data could result.

Message numbers accompanying an 81 major code may be in the range indicating either an I/O or a close operation. A device failure on a close operation simply may be the result of a failure in sending the final block of data, rather than action specific to closing the file. An error on a close operation may result in the file being left partially closed. Your error recovery program should respond to close failures with a second close operation. The second close will always complete, regardless of errors.

Device or Session Error on Open or Acquire Operation

A major return code of 82 indicates that a device or session error occurred during an open or acquire operation. Both the minor return code and the accompanying message will provide more specific information regarding the cause of the problem.

Some of the minor return codes in this group are the same as those for the 81 major return code. Device or line failures may occur at any time, but an 82 major code indicates that the device or session was unusable when your program first attempted to use it. Thus no data was transferred. The problem may be the result of a configuration or installation error.

Depending on the minor return code, it may be appropriate for your program to recover from the error and try the failing operation again after some waiting period. The number of times you try should be specified in your program. It may also be possible to use an alternate or backup device or session instead.

Message numbers accompanying an 82 major code may be in the range indicating either an open or an acquire operation. If the operation was an open, it is necessary to close the partially opened file and reopen it to recover from the error. If the operation was an acquire, it may be necessary to do a release operation before trying the acquire again. In either case, the file wait time should be specified long enough to allow the system to recover from the error.

Recoverable Device or Session Errors on I/O Operation

A major return code of 83 indicates that an error occurred in sending data to a device or receiving data from the device. Recovery by the application program is possible. Both the minor return code and the accompanying message provide more specific information regarding the cause of the problem.

Most of the errors in this group are the result of sending invalid commands or data to the device, or sending valid data at the wrong time or to a device that is not able to handle it. The application program may recover by skipping the failing operation or data item and going on to the next one, or by substituting an appropriate default. There may be a logic error in the application.

Related Information on File Types

Refer to the following manuals for more information on the file types discussed in this chapter:

- DDM files: *DDM User's Guide*
- ICF files: *Communications Programmer's Guide*
- Database files: *Database Guide*
- Save files: *Backup and Recovery Guide*
- Display, printer, tape, and diskette files: Part 3, "Device File Support" of the *Data Management Guide*

Chapter 2. Overrides and File Redirection

Overrides are used to temporarily change a file name, a device name or remote location name associated with the file, or some of the other attributes of a file. Override commands may be entered interactively from a terminal or as part of a batch job. They may be included in a control language (CL) program, or they may be issued from other programs via a call to the program QCMDEXC. Regardless of how they are issued, overrides remain in effect only for the job, program, or terminal session in which they are issued. Furthermore, they have no effect on other jobs that may be running at the same time.

Overrides are particularly useful for making minor changes to the way a program functions or for selecting the data on which it operates, without having to recompile the program. Their principal value is in allowing you to use general purpose programs in a wider variety of circumstances. Examples of items where overrides may be used are:

- Changing the name of the file to be processed
- Selecting the database file member to be processed
- Indicating whether output is to be spooled
- Directing output to a different tape device
- Changing printer characteristics such as lines per inch and number of copies
- Selecting the remote location to be used with an ICF file
- Changing the characteristics of a communications session

It is also possible to use overrides to direct data input or output to a device of a different type; for example, to send data that was intended for a diskette to a printer instead. This use of overrides requires somewhat more foresight than the override applications listed above, because the program must be able to accommodate the different characteristics of the two devices involved. The special considerations required for overrides that change the file type are discussed in "File Redirection" on page 2-24.

There are two types of overrides: *file* overrides and *program device entry* overrides. They are discussed separately in the remainder of this chapter.

Overriding Files

When you create an application program, files are associated with it by the file names specified in the program. The system lets you override these file names and/or the attributes of the specified file when you compile a program or run a program. The system supplies three override functions: applying overrides, deleting overrides, and displaying overrides. You can process override functions for files using the following CL commands:

DLTOVR Delete Override: Deletes one or more file overrides (including message file overrides) that were previously specified in a call level.

DSPOVR Display Override: Displays file overrides at any active call level for a job.

OVRDBF Override with Database File: Overrides (replaces) the database file named in the program, overrides certain parameters of a database file that is used by the program, or overrides the file and certain parameters of the file to be processed.

OVRDKTF Override with Diskette File: Overrides (replaces) the diskette file named in the program, overrides certain parameters of a diskette file that is used by the program, or overrides the file and certain parameters of the file to be processed.

OVRDSPF Override with Display File: Overrides (replaces) the display file named in the program, overrides certain parameters of a display file that is used by the program, or overrides the file and certain parameters of the file to be processed.

OVRICFF Override with Intersystem Communications Function File: Used to override the file named in the program and override certain parameters of the file being processed.

OVRMSGF Override with Message File: Used to override a message file used in a program. The rules for applying the overrides in this command are different from the other override commands. For more information on overriding message files, see the *CL Programmer's Guide*.

OVRPRTF Override with Printer File: Overrides (replaces) the printer file named in the program, overrides certain parameters of a printer file that is used by the program, or overrides the file and certain parameters of the file to be processed.

OVRSAVF Override Save File: Overrides (replaces) the file named in the program, overrides certain attributes of a file that is used by the program, or overrides the file and certain attributes of the file to be processed.

OVRTAPF Override with Tape File: Overrides (replaces) the file named in the program, overrides certain attributes of a file that is used by the program, or overrides the file and certain attributes of the file to be processed.

Overrides may be used to change most, but not all, of the file attributes that are specified when the file is created. In some cases, attributes may be specified in overrides that are not part of the original file definition. Refer to the command descriptions in the *CL Reference* for details.

The scope and function of an override command is determined in several ways by its call level. (See "Call Level with Override Commands" on page 2-5.) In general, overrides remain in effect from the time they are specified until they are replaced or deleted, or until the program in which they are specified ends.

Overrides applied include any that are in effect at the time a file is opened by an application program, when a program that opens a file is compiled, or when certain system commands are used. (See "Applying Overrides when Using High-Level Language Programs" on page 2-3, "Applying Overrides when Compiling a Program" on page 2-14, and "Effect of Overrides on Some System Commands" on page 2-15). Thus any overrides that are to be applied must be specified either before the file is opened by a program or before a program that opens the file is compiled. It is not necessary that overrides be supplied for every file used in a program. Any file name for which no override is supplied is used as the actual file name.

Overriding a file is different from changing a file in that an override does not permanently change the attributes of a file. For example, if you override the number of copies for a printer file by requesting six copies instead of two, the file description for the printer file still specifies two copies, but six copies are printed. The system uses the file override command to determine which file to open and/or what its file attributes are.

Handling overrides for message files is different in some respects from handling overrides for other files. Only the name of the message file, not the attributes, can be overridden. For more information on message handling, refer to the *CL Programmer's Guide*.

Applying Overrides when Using High-Level Language Programs

There are three different types of file overrides. These are discussed in the following sections.

Overriding File Attributes

The simplest form of overriding a file is to override some attributes of the file. File attributes are built as a result of the following:

- Create file and add member commands. Initially, these commands build the file attributes.
- Program using the files. At compile time, the user program can specify some of the file attributes. (The attributes that can be specified depend on the high-level language in which the program is written.)
- Override commands. At program run time, these commands can override the file attributes previously built by the merging of the file description and the file parameters specified in the user program.

For example, assume that you create a printer file OUTPUT whose attributes are:

- Page size of 66 by 132
- Six lines per inch
- Two copies of printed output
- Two pages for file separators
- Overflow line number of 55

The Create Printer File (CRTPRTF) command looks like this:

```
CRTPRTF FILE(QGPL/OUTPUT) SPOOL(*YES)  +
  PAGESIZE(66 132) LPI(6) +
  COPIES(2)  FILESEP(2)  OVRFLW(55)
```

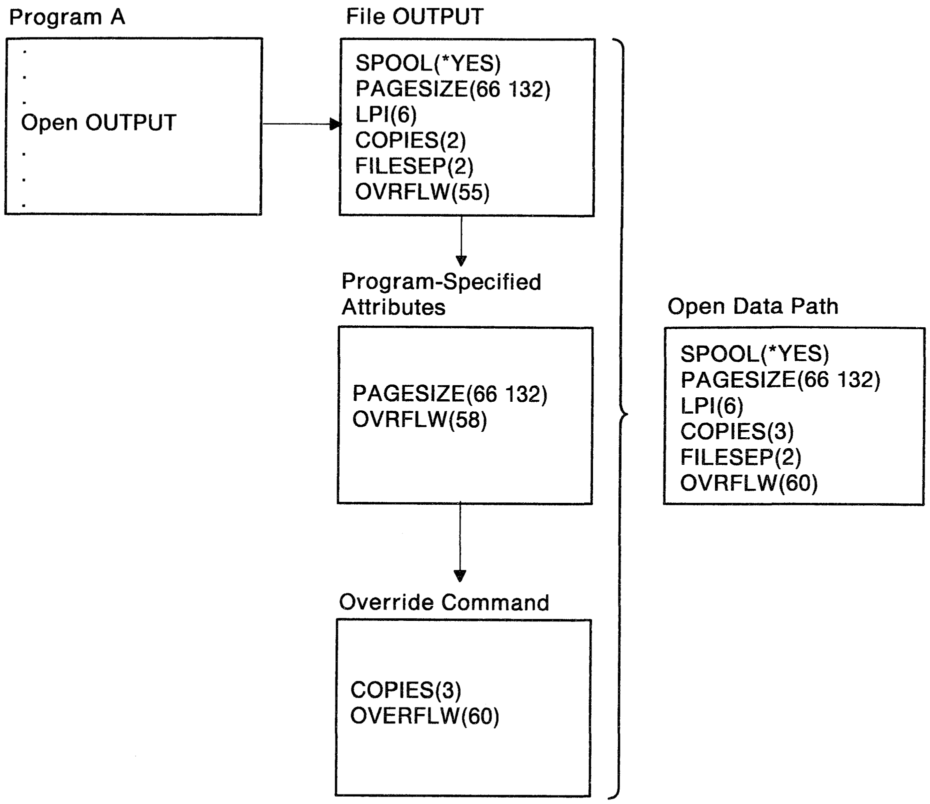
The printer file OUTPUT is specified in your application program with an overflow line number of 58. However, before you run the application program, you want to change the number of copies of printed output to 3 and the overflow line to 60. The override command looks like this:

```
OVRPRTF  FILE(OUTPUT)  COPIES(3)  OVRFLW(60)
```

Then you call the application program, and three copies of the output are printed.

When the application program opens the file, the file overrides, program-specified attributes, and file attributes are merged to form the open data path (ODP) which is used during the running of the program. File overrides have precedence over program-specified attributes. Program-specified attributes have precedence over file-specified attributes. In this example, when the file is opened and output operations are performed, spooled output will be produced with a page size of 66 by 132, six lines per inch, three copies, two file separator pages, and overflow at 60 lines.

The following chart explains this example:



RSLH179-1

Overriding File Names or Types

Another simple form of overriding a file is to change the file that is used by the program. This may be useful for files that have been moved or renamed after the program has been compiled. For example, you want the output from your application program to be printed using the printer file REPORTS instead of the printer file OUTPUT (OUTPUT is specified in the application program). Before you run the program, enter the following:

```
OVRPRTF FILE(OUTPUT) TOFILE(REPORTS)
```

The file REPORTS must have been created by a CRTPRTF command before it can be used.

If you want to override to a different type of file, you use the override command for the new type of file. For example, if you are overriding a diskette file with a printer file, use the Override with Printer File (OVRPRTF) command.

Use the information under "File Redirection" on page 2-24 to determine if files can be overridden to another type of file.

Overriding File Names or Types and File Attributes of the New File

This form of overriding files is simply a combination of overriding file attributes and overriding file names or types. With this form of override, you can override the file that is to be used in a program and you can also override the attributes of the overriding file. For example, you want the output from your application program to be printed using the printer file REPORTS instead of the printer file OUTPUT (OUTPUT is specified in the application program). In addition to having the application program use the printer file REPORTS, you wish to override the number of copies produced to three. Assume the file REPORTS was created with the following command:

```
CRTPRTF FILE(REPORTS) SPOOL(*YES)  +
  PAGESIZE (68 132) LPI(8)  +
  OVRFLW(60) COPIES(2) FILESEP(1)
```

Before you run the program, type the following command:

```
OVRPRTF FILE(OUTPUT) TOFILE(REPORTS) COPIES(3)
```

Then call the application program, and three copies of the output are produced using the printer file REPORTS.

Note that this is *not* equal to the following two override commands:

```
Override 1  OVRPRTF    FILE(OUTPUT)   TOFILE(REPORTS)
Override 2  OVRPRTF    FILE(REPORTS)  COPIES(3)
```

Only one override is applied for each call level for an open of a particular file, so if you want to override the file that is used by the program and also override the attributes of the overriding file from one call level, you must use a single command. If two overrides are used, override 1 will cause the output to be printed using the printer file REPORTS, but override 2 will be ignored.

Call Level with Override Commands

Call levels identify the subordinate relationships between related programs when one program is called from another program. For example:

```
Call Level 1  PGM A
              xxxxxx
              xxxxxx
              CALL PGM B

Call Level 2    PGM B
                xxxxxx
                xxxxxx
                TFRCTL PGM C

                PGM C
                xxxxxx
                xxxxxx
                CALL PGM D

Call Level 3      PGM D
                  xxxxxx
                  RETURN
```

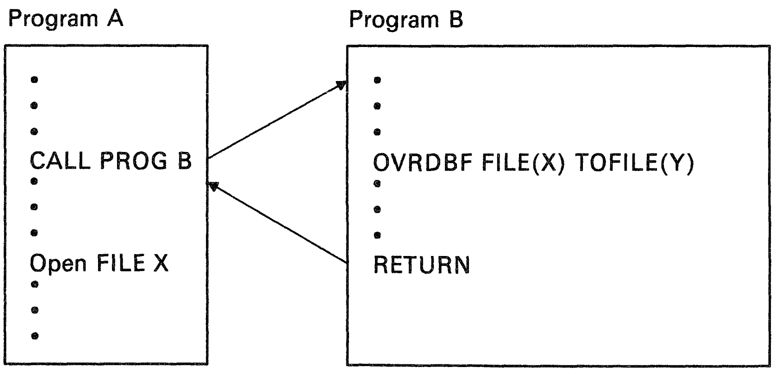
Several commands, such as Work with Job (WRKJOB), Work with Active Jobs (WRKACTJOB), or Display Job (DSPJOB), have options that allow you to display the program stack of an active job. There is a one-to-one relationship between a program displayed in the program stack and the call level for that program. The first program name displayed (at the top of the list) on the program stack is the program at call level 1 for that job. The second program name displayed is the program at call level 2 for that job. The last program name displayed is the program at the highest call level for that job.

In the previous example, the TFRCTL to PGMC causes PGMB to be removed from the program stack and replaced by PGMC. A call causes another program to be placed in the program stack. A return causes a program to be removed from the stack.

Call levels affect override processing by the following general principles. Specific examples of each can be found throughout this chapter.

- An override command entered interactively exists at the call level of the caller of that command processor. For example, an override entered on the command entry display cannot be deleted or replaced from a command processor called from the command entry display.
- The call level of an override coded in a CL program is the call level of the CL program.
- An override outside a program in a batch job takes the call level of the batch job command processor.
- If an override command is run via a call to the QCMDEXC program, the override takes the call level of the program that called the QCMDEXC program.
- Exits (ENDPGM, RETURN, or abnormal exits) from a call will delete overrides at that call level. For example, a RETURN command deletes all overrides at that call level. Thus, overrides in called programs that end with a RETURN or ENDPGM command do not apply to the calling program. This is not true for programs using the Transfer Control (TFRCTL) command.

  In the following example, the RETURN command deletes the override in program B, and FILE X is opened in program A.

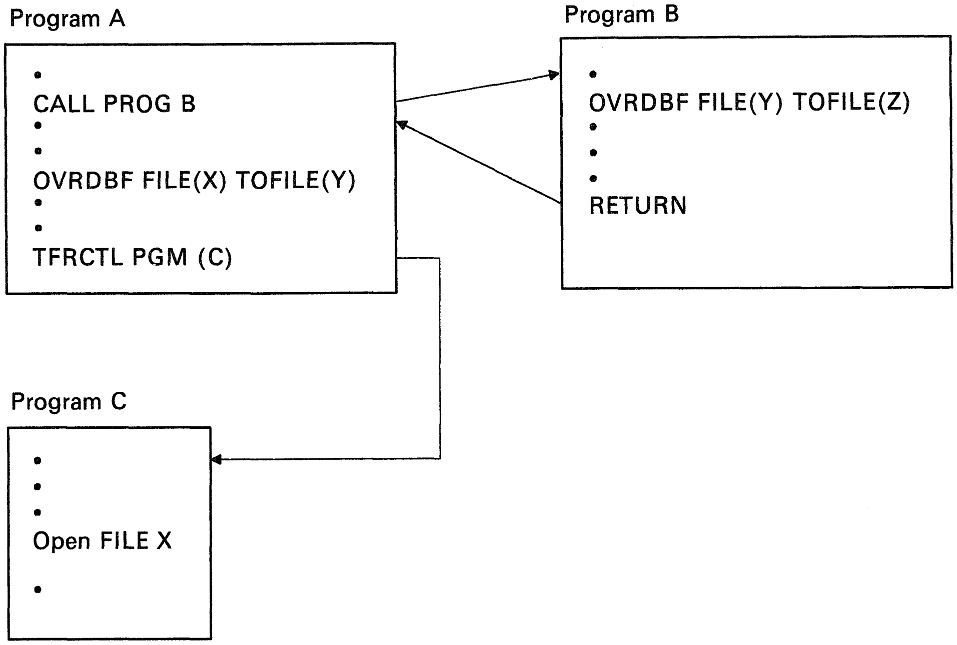


```
•
•
•
CALL PROG B
•
•
•
Open FILE X
•
•
•
```

```
•
•
•
OVRDBF FILE(X) TOFILE(Y)
•
•
•
RETURN
```

- As long as a program is running at the same call level that requested an override, that override stays in effect unless it is deleted. The TFRCTL command causes one program to be replaced by another program at the same call level. The program, to which control is transferred, runs at the same call level as the program that contained the TFRCTL command. An override command in a program that transfers control to another program is not deleted during the transfer of control.

In the following example, the override in program A is applied to the open operation in program C because the TFRCTL command keeps the call level. FILE Y is opened. The override in program B is deleted.



Program A

```
•
CALL PROG B
•
•
OVRDBF FILE(X) TOFILE(Y)
•
•
TFRCTL PGM (C)
```

Program B

```
•
OVRDBF FILE(Y) TOFILE(Z)
•
•
•
RETURN
```

Program C

```
•
•
•
Open FILE X
•
```

P7730134-0

- Several overrides to a single file, each at a different call level, are allowed and are applied by the system in inverse call level order. For example, an override to a file at call level 3 is overridden by the override to that file encountered at call level 2, and so on. The final authority for any attribute is the outermost level of call nesting which specifies that attribute. For example, a user at a terminal calls a CL program which then calls an RPG program. Any attributes not overridden by the user are taken from an override in the CL program, if present. Any remaining attributes are taken from the RPG program, or finally, the device file or database file.

  If these overrides change file types and attributes, the attributes applied to the file opened are only those specified by overrides for the same file type as the file actually opened.

- You can protect an override from being overridden by overrides at lower call levels by coding SECURE(*YES) on it.

- When overrides are applied, only one override can be used from a call level for any particular file. If two or more overrides for the same file are requested at the same call level, the last override makes the others obsolete. This is true even if the override involves a name change, as the following example illustrates. It is really one file that is being overridden, and therefore only one override is allowed at each level.

In the following example, when the program attempts to open FILE A, FILE A is overridden with FILE B because of override 2. Because only one override can be applied for each call level, override 1 is ignored, and the file opened by the program is FILE B.

```
                   PGM:
                   .
                   .
                   .
Override 1         OVRDBF   FILE(B)   TOFILE(C)
Override 2         OVRDBF   FILE(A)   TOFILE(B)
                   .
                   .
                   .
                   OPEN  FILE  A
                   .
                   .
                   .
                   ENDPGM
```

To open FILE C, replace the two Override with Database File (OVRDBF) commands with the following command:

```
OVRDBF FILE(A) TOFILE(C)
```

This does not prevent applying an override at the same call level in which the file is created. File attributes on the override take the place of corresponding attributes on the file create statement, regardless of which is encountered first.

Applying Overrides at the Same Call Level

As long as a program is running at the same call level that requested an override, that override stays in effect unless it is deleted (see "Deleting Overrides" on page 2-17) or replaced by another override requested from the same call level. In the following example, program A transfers control to program B, and program B runs in the same call level as program A. The Override with Database File (OVRDBF) command causes the file to be positioned at the last record of the member when it is opened and is used for both programs A and B.

```
CALL PGM(A)
```

Program A:

```
OVRDBF FILE(INPUT) POSITION(*END)
```

(INPUT is opened and positioned at the last record of the member and closed after processing.)

```
TFRCTL PGM(B)
```

Program B:

(INPUT is opened and positioned at the last record of the member.)

When two overrides are entered for the same file name at the same call level, the second override replaces the first override. This allows you to replace an override at a single call level, without having to delete the first override (see "Deleting Overrides" on page 2-17). For example:

```
Override 1   OVRDKTF  FILE(QDKTSRC)  LABEL(X)
             CALL  PGM(REORDER)
Override 2   OVRDKTF  FILE(QDKTSRC)  LABEL(Y)
             CALL  PGM(REORDER)
```

Assume that program REORDER uses the diskette file QDKTSRC. Override 1 causes the first call to program REORDER to use the source file with a label of X for its processing. Override 2 causes the second call to program REORDER to use the source file with a label of Y for its processing.

Applying Overrides from Multiple Call Levels

When you have more than one override for the same file at several call levels (nested calls), the order in which the overrides are applied to the file is from the innermost override (the override with the largest call level) to the outermost override (the override with the smallest call level). To prevent file overrides at lower call levels, see "Securing Files" on page 2-11. In the following example, override 2 is the innermost and override 1 is the outermost:

```
Override 1   OVRPRTF  FILE(OUTPUT)   COPIES(6)  SPOOL(*YES)
             CALL  PGM(A)

               Program A
Override 2     OVRPRTF  FILE(OUTPUT)  COPIES(2)  LPI(6)
               CALL  PGM(X)
```

When program X opens the file OUTPUT, the opened file has the following attributes:

COPIES(6)	From Override 1
SPOOL(*YES)	From Override 1
LPI(6)	From Override 2

The attribute of COPIES(2) specified in override 2 is not used because COPIES(6) specified in override 1 takes precedence. If a third override for OUTPUT is embedded in program X, override 2 and then override 1 each override it.

A similar situation exists when you change the name of the file used in the program and you also change some of the attributes of the file. For example:

```
Override 1   OVRDBF  FILE(PAYROLL)  MBR(CURRENT)
             CALL  PROG1

               Program PROG1
Override 2     OVRDBF  FILE(INPUT)  TOFILE(PAYROLL)
               CALL  PROG2
```

When program PROG2 is ready to open INPUT, it opens PAYROLL instead (because of override 2). Also, the member used for processing is CURRENT (because of override 1).

When several overrides that override the file type to be used by a program are applied, only the attributes specified on the overrides of the same type as the final file are applied. In the following example, assume that program MAKEMASTER attempts to open the diskette file DKA:

```
Override 1   OVRDKTF  FILE(PRTA)  TOFILE(DKB)  LABEL(DKFIRST)
             CALL  PGM(A)

              Program A
Override 2    OVRPRTF  FILE(DKA)  TOFILE(PRTA)  SPOOL(*YES)
              CALL  PGM(B)

               Program B
Override 3     OVRDKTF  FILE(PRTB)  TOFILE(DKA)  DEV(DKT01)  LABEL(DKLAST)

Override 4     OVRDKTF  FILE(DKA)  TOFILE(DKC)  DEV(DKT02)  LABEL(DKTTST)
               CALL  PGM(C)

                Program C
Override 5      OVRPRTF  FILE(DKA)  TOFILE(PRTB)  SCHEDULE(*JOBEND)
                CALL  PGM(D)

                 Program D
Override 6       OVRDKTF  FILE(DKA)  VOL(MASTER)
                 CALL  PGM(MAKEMASTER)

                  Program MAKEMASTER
                  (Program MAKEMASTER attempts to open file DKA, but
                  actually opens the diskette file DKB.)
```

In the preceding example, the file that program MAKEMASTER actually opens is the diskette file DKB because of the following reasons:

- Override 6 (applied first) does not cause file DKA to be overridden with any other file.
- Override 5 (applied second) causes file DKA to be overridden with printer file PRTB.
- Override 4 is ignored at this level because override 5 changed the file name to PRTB.
- Override 3 (applied third) causes file PRTB to be overridden with diskette file DKA.
- Override 2 (applied fourth) causes file DKA to be overridden with printer file PRTA.
- Override 1 (applied last) causes file PRTA to be overridden with diskette file DKB.

Therefore, the file that program MAKEMASTER opens is the diskette file DKB. Because the file to be opened is a diskette file, the attributes overridden are only those specified on the Override with Diskette File (OVRDKTF) commands: VOL(MASTER) from override 6; DEV(DKT01) from override 3; and LABEL(DKFIRST) from override 1.

The attributes specified on the Override with Printer File (OVRPRTF) commands are ignored (even though they might have been allowed on the OVRDKTF commands). Refer to "File Redirection" on page 2-24 for more information on the effect of overrides that change the file type.

CL Program Overrides

If a CL program overrides a file and then calls a high-level language program, the override remains in effect for the high-level language program. However, if a high-level language program calls a CL program that overrides a file, the override is deleted automatically when control returns to the high-level language program.

```
High-level language program:
CALL  CLPGM1
  CL program:
  OVRDKTF  FILE(DK1)  TOFILE(MSTOUT)
  .
  .
  .
  ENDPGM
High-level language program:
OPEN  DK1
```

The file opened is DK1, not MSTOUT. This is because the override in the CL program is deleted when the CL program ends.

To perform an override from a high-level language program, call the QCMDEXC program from the high-level language program. The override, via a call to QCMDEXC, takes the call level of the program that called QCMDEXC.

```
High-level language program:
CALL  QCMDEXC  PARM('OVRDKTF FILE(DK1) TOFILE(MSTOUT)'  32)
OPEN  DK1
```

The file opened is MSTOUT because of the override requested by the call to the QCMDEXC program.

In an actual program, you might want to use data supplied by the program as a parameter of the override. This can be done by using program variables in the call to QCMDEXC. For more information on the use of program variables, refer to the appropriate language manual.

Securing Files

On occasion, you may want to prevent the person or program that calls your program from changing the file names or attributes you have specified. You can prevent additional file overrides by coding the SECURE(*YES) parameter on a file override command for each file needing protection. This protects your file from overrides at lower call levels.

The following shows an example of a protected file:

```
Override 1   OVRPRTF FILE(PRINT1) SPOOL(*NO)

Override 2   OVRDBF FILE(NEWEMP) TOFILE(OLDEMP) MBR(N67)
             CALL PGM(CHECK)

               Program CHECK
Override 3     OVRDBF FILE(INPUT) TOFILE(NEWEMP) MBR(N77) SECURE(*YES)
               CALL PGM(NEMPRPT)

                 Program NEMPRPT
                 (NEWEMP and PRINT1 are opened.).

Override 4     OVRDBF FILE(INPUT) TOFILE(NEWEMP) MBR(N77)
               CALL PGM(NEMPLST)

                 Program NEMPLST
                 (OLDEMP and PRINT1 are opened.)
```

When program NEMPRPT is called, it attempts to open the files INPUT and PRINT1. NEMPRPT actually opens files NEWEMP, member N77. Because override 3 specifies SECURE(*YES), override 2 is not applied. When program NEMPLST is called, it also attempts to open the files INPUT and PRINT1. NEMPLST actually opens files OLDEMP, member N67. Because override 4 has the same name as override 3 and is at the same call level as override 3, it replaces override 3. Thus, the file is no longer protected from overrides at lower call levels, and override 2 is applied for program NEMPLST.

PRINT1 is affected only by override 1, which is in effect for both programs NEMPRPT and NEMPLST.

Using a Generic Override for Printer Files

The OVRPRTF command allows you to have one override for all the printer files in your job with the same set of values. Without the generic override, you would have to do a separate override for each of the printer files.

Following are specific examples of applying the OVRPRTF command.

Applying OVRPRTF with *PRTF

The OVRPRTF command can be applied to all printer files by specifying *PRTF as the file name.

The OVRPRTF command with *PRTF is applied if there is no other override for the printer file name at the same call level. The following example shows how *PRTF works:

```
Override 1   OVRPRTF FILE(OUTPUT) COPIES(6) LPI(6)

Override 2   OVRPRTF FILE(*PRTF) COPIES(1) LPI(8)
             CALL PGM(X)
```

When program X opens the file OUTPUT, the opened file has the following attributes:

COPIES(6)	From Override 1
LPI(6)	From Override 1

When program X opens the file PRTOUT, the opened file has the following attributes:

COPIES(1)	From Override 2
LPI(8)	From Override 2

Applying OVRPRTF with *PRTF from Multiple Call Levels

When you have more than one override for the printer file at several call levels (nested calls), the order in which the overrides are applied to the file is from the innermost override (the override with the largest call level) to the outermost override (the override with the smallest call level).

```
            Program A
Override 1  OVRPRTF FILE(*PRTF) COPIES(1)
Override 2  OVRPRTF FILE(PRT2) COPIES(2)
Override 3  OVRPRTF FILE(PRT4) COPIES(2)
            CALL PGM(B)

              Program B
Override 4    OVRPRTF FILE(*PRTF) LPI(4)
Override 5    OVRPRTF FILE(PRT3) LPI(8)
Override 6    OVRPRTF FILE(PRT4) LPI(8)
              CALL PGM(X)
```

When program X opens the file PRT1, the opened file has the following attributes:

COPIES(1)	From Override 1
LPI(4)	From Override 4

Because no specific overrides are found for PRT1, *PRTF overrides (1 and 4) are applied.

When program X opens the file PRT2, the opened file has the following attributes:

COPIES(2)	From Override 2
LPI(4)	From Override 4

Because no specific override is found for PRT2 in program B, override 4 is applied. In program A, override 2 specifies PRT2 and is applied.

When program X opens the file PRT3, the opened file has the following attributes:

COPIES(1)	From Override 1
LPI(8)	From Override 5

In program B, override 5 specifies PRT3 and is applied. Because no specific override is found for PRT3 in program A, override 1 is applied.

When program X opens the file PRT4, the opened file has the following attributes:

COPIES(2)	From Override 3
LPI(8)	From Override 6

In program B, override 6 specifies PRT4 and is applied. In program A, override 3 specifies PRT4 and is applied.

Applying Overrides when Compiling a Program

Overrides may be applied at the time a program is being compiled for either of two purposes:

- To select the source file
- To provide external data definitions for the compiler to use in defining the record formats to be used on I/O operations

Overrides to the source file are handled just like any other override. They may select another file, another member of a database file, another label for diskette or tape, or change other file attributes.

Overrides may also be applied to files that are used within the program being compiled, if they are being used as externally described files in the program. These files are not opened at compile time, and thus the overrides are not applied in the normal manner. These overrides are used at compile time only to determine the file name and library that will be used to define the record formats and fields for the program to use I/O operations. Any other file attributes specified on the override are ignored at compile time. It is necessary that these file overrides be active at compile time only if the file name specified in the source for the program is not the file name that contains the record formats that the application needs.

The file name that is opened when the compiled program is run is determined by the file name that the program source refers to, changed by whatever overrides are in effect at the time the program runs. The file name used at compile time is not kept. The record formats in the file that is actually opened must be compatible with those used when the program was compiled. Obviously, the easiest way to assure record compatibility is to have the same overrides active at run time that were active at compile time. If your program uses externally described data and a different field level file is used at run time, it is usually necessary to specify LVLCHK(*NO) on the override. See “File Redirection” on page 2-24 for details.

In the following example, assume that the source for the program INVENTORY, taken from member INVN42, contains an open to the printer file LISTOUT:

```
Override 1   OVRDKTF FILE(RPGSRC) TOFILE(SRCPGMS) MBR(INVN42)
Override 2   OVRPRTF FILE(OUTPUT) TOFILE(REPORTS)
             CALL PGM(A)

               Program A
Override 3     OVRPRTF FILE(LISTOUT) TOFILE(OUTPUT)
Override 4     OVRDKTF FILE(RPGSRC) WAITFILE(30)
               CRTRPGPGM PGM(INVENTORY) SRCFILE(RPGSRC)
               RETURN

Override 5   OVRPRTF FILE(LISTOUT) TOFILE(REPORTS) LPI(8)
             CALL PGM(INVENTORY)
```

The program INVENTORY opens the printer file REPORTS in place of printer file LISTOUT and creates output at 8 lines per inch.

The program INVENTORY is created (compiled) from the member INVN42 in the database file SRCPGMS. Override 4 (applied first) overrides an optional file attribute. Override 1 (applied last) causes the file RPGSRC to be overridden with the database file SRCPGMS, member INVN42.

The program INVENTORY is created with the printer formats from the file REPORTS. Override 3 (applied first) causes the file LISTOUT to be overridden with OUTPUT. Override 2 (applied last) overrides OUTPUT with REPORTS. Other attributes may be specified here, but it is not necessary because only the record formats are used at compile time.

At run time, override 3 is no longer active, because program A has ended. Therefore override 2 has no effect on LISTOUT. However, override 5, which is active at run time, replaces LISTOUT with REPORTS and specifies 8 lines per inch. Because the same file is used for both compilation and run-time, level checking may be left on.

Effect of Overrides on Some System Commands

The following commonly used commands ignore overrides entirely:

ADDLFM
ADDPFM
ALCOBJ
APYJRNCHG
CHGOBJOWN
CHGPTR
CHGSBSD
CHGXXXF (all change file commands)
CLRPFM
CLRSAVF
CPYIGCTBL
CRTDKTF
CRTDUPOBJ
CRTAUTHLR
CRTSBSD
CRTTAPF
DLCOBJ
DLTF
DLTAUTHLR
DSPDBR
DSPFD
DSPFFD
DSPJRN
EDTOBJAUT
EDTDLOAUT
ENDJRNPF
GRTOBJAUT
INZPFM
MOVOBJ
RGZPFM
RMVJRNCHG
RMVM
RNMOBJ
RSTUSRPRF
RVKOBJAUT
SAVCHGOBJ
SAVLIB
SAVOBJ
SAVPGMPRD
SAVSAVFDTA
SAVSYS
SBMDBJOB
SIGNOFF
STRDBRDR
STRJRNPF

Overrides are not applied to any system files that are opened as part of an end-of-routing step or end-of-job processing. For example, overrides cannot be specified for the job log file. In some cases, when you need to override something in a system file, you may be able to change it through a command other than an override command. For example, to change the output queue for a job log, the output queue could be changed before sign-off using the OUTQ parameter on the Change Job (CHGJOB) command to specify the name of the output queue for the job. If the printer file for the job log contains the value *JOB for the output queue, the output queue is the one specified for the job.

The following commands allow overrides for the SRCFILE and SRCMBR parameters only:

CRTCMD
CRTICFF
CRTDSPF
CRTLF

CRTPF
CRTPRTF
CRTSRCPF
CRTTBL
CRTXXXPGM
(All create program commands. These commands also use overrides to determine which file will be opened by a compiled program. See "Applying Overrides when Compiling a Program" on page 2-14 for more information.)

The following command allows overrides for the TOFILE, MBR, SEQONLY, LVLCHK, and INHWRT parameters:

OPNQRYF

The following commands allow overrides, but do not allow changing the MBR to *ALL:

CPYFRMPCD

CPYTOPCD

The following commands do not allow overrides to be applied to the display files they use. Overrides to the printer files they use should not change the file type or the file name. Various restrictions are placed on changes that may be made to printer files used by these commands, but the system can not guarantee that all combinations of possible specifications will produce an acceptable report.

DMPOBJ and DMPSYSOBJ (In addition to the preceding limitations, these commands do not allow overrides to the file they dump.)

DSPXXXXXX (All display commands. The display commands that display information about a file do not allow overrides to that file.)

DSPIGCDCT
EDTIGCDCT

GO (Message file can be overridden.)

PRTXXXXXX (All print commands.)

QRYDTA

TRCXXX (All trace commands.)

WRKXXXXXX (All work-with commands.)

Deleting Overrides

When a program returns from a call level containing overrides, the overrides are deleted. When control is transferred to another program (TFRCTL command) so that the program is running at the same call level, the overrides are not deleted. If you want to delete an override before the program has completed running, you can use the Delete Override (DLTOVR) command. This command deletes overrides only in the call level in which the command is entered. To identify an override, use the file name specified on the FILE parameter of the override command. You can delete all overrides at this call level by specifying value *ALL for the FILE parameter. For example:

```
Override 1          OVRDBF FILE(DBA) TOFILE(DBB)
Override 2          OVRPRTF FILE(PRTC) COPIES(2)
Override 3          OVRDKTF FILE(DKT) EXCHTYPE(*BASIC)
Delete Override 1   DLTOVR FILE(DBA)
Delete Override 2   DLTOVR FILE(*ALL)
```

Delete override 1 causes override 1 to be deleted. Delete override 2 causes the remaining overrides (overrides 2 and 3) to be deleted.

In the following example, assume that commands 1, 2, and 11 are entered interactively, at call level 1:

```
Command 1       OVRDBF FILE(DBA) TOFILE(DBB) SECURE(*YES)

Command 2       CALL PGM(A)

                  Program A
Command 3         OVRPRTF FILE(DBB) TOFILE(PRTC) LPI(6)
Command 4         TFRCTL PGM(B)

                  Program B
Command 5         OVRDKTF FILE(DKTE) TOFILE(DKTF)
Command 6         CALL PGM(QCMDEXC) +
                    PARM('OVRDSPF FILE(DSPG) TOFILE(DSPH)'  31)
Command 7         DLTOVR FILE(DBA DBB)
Command 8         MONMSG MSGID(CPF9841)
Command 9         CALL PGM(QCMDEXC)  +
                    PARM('DLTOVR FILE(*ALL)'  17)

Command 10        RETURN
Command 11      DLTOVR  FILE(*ALL)
```

Command 1 causes an override at level 1 from file DBA to file DBB.

Command 2 calls program A and creates a new call level, level 2.

Command 3 causes an override at level 2 from file DBB to file PRTC. Also, the LPI attribute of file PRTC is overridden to 6.

Command 4 transfers control from program A to program B at the same call level, level 2.

Command 5 causes an override at level 2 from file DKTE to file DKTF.

Command 6 causes an override at level 2 from file DSPG to file DSPH. A call to QCMDEXC does not cause a new call level.

Command 7 deletes any overrides of files DBA and DBB at level 2. The override specified by command 3 is deleted, but the override specified by command 1 is not deleted. Because an override for DBA cannot be found at level 2, the override-not-found escape message (CPF9841) is sent.

Command 8 monitors for a message to prevent a function check, but specifies no action to be taken if the message is sent.

Command 9 deletes all remaining overrides at level 2. Overrides specified by commands 5 and 6 are deleted, but the override specified by command 1 is not deleted.

Command 10 causes a return to level 1, and level 2 is deleted. If any overrides were specified at level 2 between command 9 and command 10, they are deleted at this point.

Command 11 causes all remaining overrides specified at level 1 to be deleted. The override specified by command 1 is deleted.

Displaying Overrides

You can use the Display Override (DSPOVR) command to display file overrides at multiple call levels for a job. You can display all file overrides, or file overrides for a specific file.

The file overrides may be merged before being displayed. A merged override is the result of combining overrides from level 1 to the current level or any specified call level, producing a composite override which will be applied when the file is used at the specific call level. The current call level is the call level of the program that is currently running. This program is the last program name displayed on the program stack. This command may be requested from either a batch or interactive environment. You can also access this function from option 15 (Display file overrides) from the Work with Job menu (using the WRKJOB command) or by selecting option 15 (Display file overrides) from the Display Job menu (using the DSPJOB command).

1. To display the merged file override for a particular file at a specific call level, you type:

   ```
   DSPOVR FILE(REPORTS) MRGOVR(*YES) LVL(3)
   ```

   This command produces a display showing the merged override for the file REPORTS at call level 3 with text descriptions of each keyword and parameter. Any applicable overrides at call levels 1, 2, and 3 are used to form the merged override, but overrides at higher call levels are ignored. If the call level specified is not active, all applicable overrides up to the current level are used.

2. To display all file overrides for a specific file up to a specific call level, you type:

   ```
   DSPOVR FILE(REPORTS) MRGOVR(*NO) LVL(2)
   ```

   This command produces a display showing the file name, the call level for which the override was requested, the type of override, and the override parameters in keyword-parameter form. If no file overrides are found for the file up to and including the specified call level, escape message CPF9842 is sent. If you are using DSPOVR in a CL program, you might want to add a MONMSG command following the DSPOVR command to prevent your program from ending if there are no overrides for the file. This technique is illustrated in some of the examples later in this chapter. For more information on the MONMSG command, refer to the *CL Programmer's Guide*.

3. To display the merged file overrides for all files at the current call level, you type:

   ```
   DSPOVR FILE(*ALL) MRGOVR(*YES) LVL(*)
   ```

   This command produces a display showing the file name, the type of override, and the merged overrides in keyword-parameter form, where only the keywords and parameters entered on the commands are displayed. This is the same as what happens when you type DSPOVR with no parameters. Only those keywords for which parameters were specified are displayed. The associated text descriptions are not displayed. Overrides at call levels greater than 999 are not displayed.

4. When overrides are displayed not by the DSPOVR command, but through an option on one of the system interfaces to work with jobs (for example, WRKJOB), all file overrides from level 1 to the current call level are displayed. This would be the same as typing the following command:

   ```
   DSPOVR FILE(*ALL) MRGOVR(*NO) LVL(*)
   ```

   This produces a display showing the file name, the call level for which the override was requested, the type of override, and the override parameters in keyword-parameter form for each override.

   Because the display overrides function uses a copy of the internal control blocks, overrides that were deleted between the time the display overrides function was called and the time the output was produced may not be reflected in the output. This can occur only when the overrides in another job are being displayed.

Note that when specifying a call level, as in the first two examples in this section, the call level on which you first entered commands may not be level 1. Depending on the contents of the first program and first menu specified in your user profile, and any other programs or menus you may have come through, it is not unusual to be entering commands at level 3 or 4. You may enter WRKJOB and select option 11 (program stack) to see what programs are running above your current level.

Unless you know exactly what you want to see, it is usually best to request the override display with no parameters, because options on the basic override display allow you to select a detailed display of any override you are interested in. The specific options available are:

- From the merged display of all overrides, you can request the display that is not merged, as in step 4.
- From the display (not merged) of all overrides, you can request the merged display.
- From the merged display of all overrides, you can request a merged detail display of any override, equivalent to step 1 on page 2-18.
- From the merged display of all overrides, you can request a display of all the individual overrides that contributed to the merged display, showing the call level at which each was requested.
- From either the display of contributing overrides or the display (not merged) of all overrides, you can request a detail display of the override for a particular file at a single call level.

The following example is intended only to illustrate what the various forms of the display override command can do. The DSPOVR command is typically entered interactively or added temporarily to a CL program, or to any high-level language program via QCMDEXC, to verify that the proper overrides are in effect at the time a program is called or a file is opened. Assume that commands 1, 2, 3, and 17 are entered at call level 1:

```
Command 1     OVRPRTF FILE(PRTA) COPIES(3)
Command 2     OVRDBF FILE(DBC) WAITFILE(*IMMED)
Command 3     CALL PGM(A)

                Program A
Command 4       OVRPRTF FILE(PRTB) TOFILE(PRTA) COPIES(6)
Command 5       OVRDBF FILE(DBC) WAITFILE(60)
Command 6       DSPOVR FILE(PRTB) MRGOVR(*YES)
Command 7       CALL PGM(B)

                  Program B
Command 8         CALL PGM(QCMDEXC)  +
                    PARM('OVRDSPF FILE(DSPE) TOFILE(DSPF)'  31)
Command 9         OVRDBF FILE(DBC) TOFILE(DBD)
Command 10        DSPOVR FILE(DBC) MRGOVR(*NO) LVL(3)
Command 11        DSPOVR FILE(DBD) MRGOVR(*NO) LVL(2)
Command 12        MONMSG MSGID(CPF9842)
Command 13        CALL PGM(QCMDEXC)  +
                    PARM('DSPOVR FILE(*ALL) MRGOVR(*YES)  +
                    LVL(*) OUTPUT(*)'  47)
Command 14        RETURN

Command 15      DSPOVR FILE(*ALL) MRGOVR(*NO)
Command 16      RETURN

Command 17    DSPOVR FILE(*ALL) MRGOVR(*NO) LVL(2) OUTPUT(*)
```

Command 1 causes an override at level 1 of the COPIES attribute of file PRTA to 3 copies.

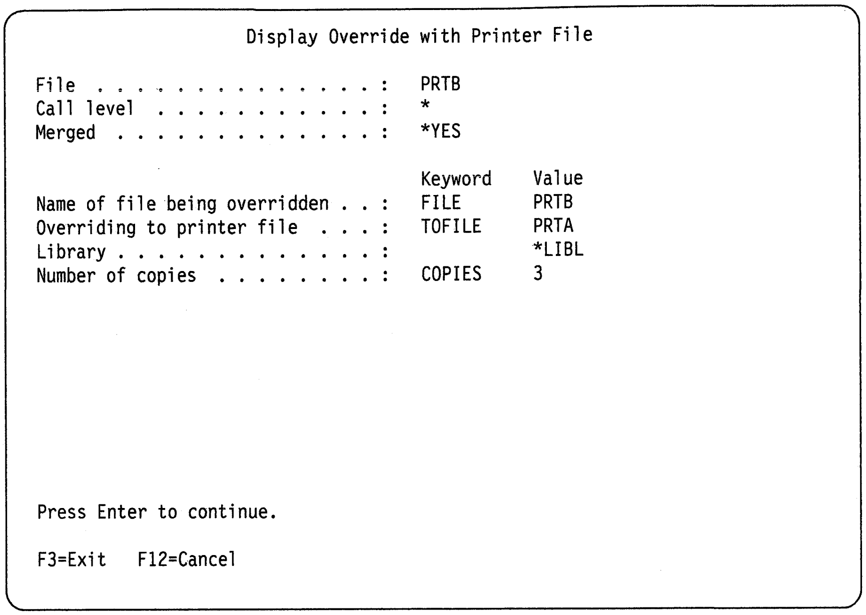
Command 2 causes an override at level 1 of the WAITFILE attribute of file DBC to *IMMED.

Command 3 calls program A and creates a new call level, 2.

Command 4 causes an override at level 2 from file PRTB to file PRTA. Also, the COPIES attribute is overridden to 6.

Command 5 causes an override at level 2 of the WAITFILE attribute of file DBC to 60.

Command 6 displays a merged override for file PRTB at level 2 with text descriptions of each keyword and parameter, as shown in the following display. The to-file is PRTA because of command 4, and the COPIES attribute is 3 because of command 1.



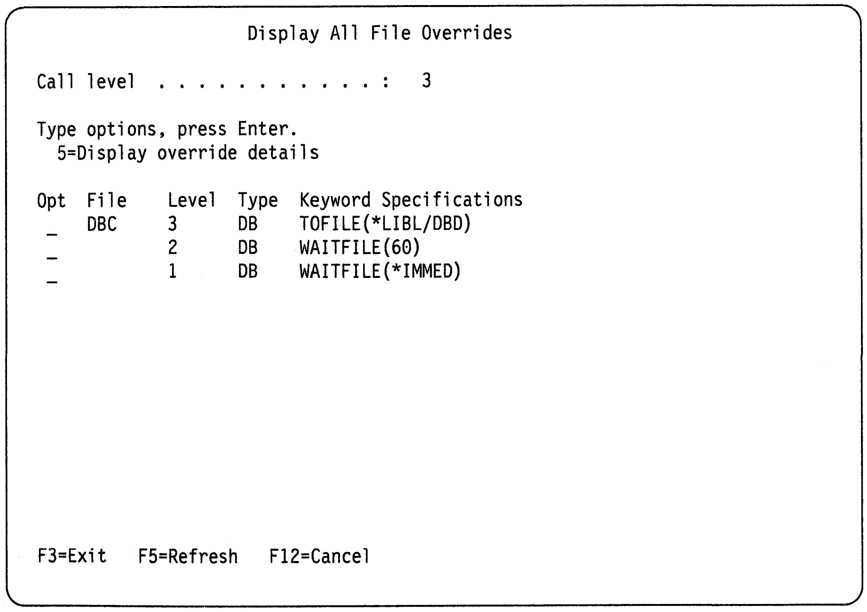
```
                     Display Override with Printer File

File . . . . . . . . . . . . . . . :   PRTB
Call level . . . . . . . . . . . . :   *
Merged . . . . . . . . . . . . . . :   *YES

                                       Keyword    Value
Name of file being overridden . . :   FILE       PRTB
Overriding to printer file  . . . :   TOFILE     PRTA
Library . . . . . . . . . . . . . :              *LIBL
Number of copies  . . . . . . . . :   COPIES     3








Press Enter to continue.

F3=Exit   F12=Cancel
```

Command 7 calls program B and creates a new call level, 3.

Command 8 causes an override at level 3 from file DSPE to file DSPF. An override done via a call to the QCMDEXC program takes the call level of the program that called the QCMDEXC program.

Command 9 causes an override of file DBC to file DBD.

Command 10 displays all overrides for file DBC from level 1 to level 3, as shown in the following display. The overrides specified by commands 9, 5 and 2 are displayed in keyword-parameter form. Observe that this form of the DSPOVR command shows all the overrides for the selected file, regardless of redirection. The three overrides that are shown would not be merged because of the name change at level 3.

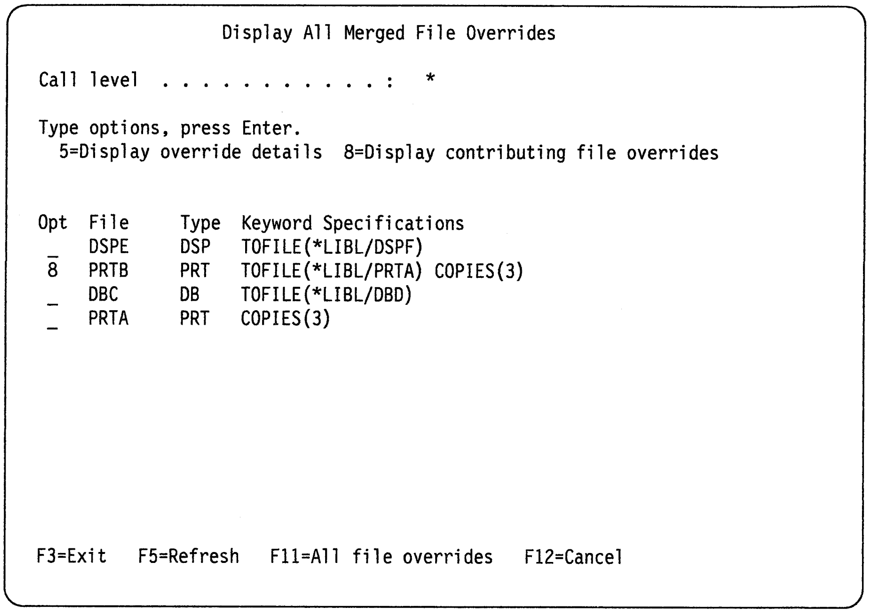


```
                     Display All File Overrides

Call level . . . . . . . . . . . . :   3

Type options, press Enter.
  5=Display override details

Opt  File     Level  Type  Keyword Specifications
 _   DBC      3      DB    TOFILE(*LIBL/DBD)
 _            2      DB    WAITFILE(60)
 _            1      DB    WAITFILE(*IMMED)












F3=Exit   F5=Refresh   F12=Cancel
```

Command 11 attempts to display all file overrides for file DBD from level 1 to level 2. Because no overrides for file DBD exist at levels 1 or 2, no overrides are displayed, and the override-not-found escape message (CPF9842) is sent.

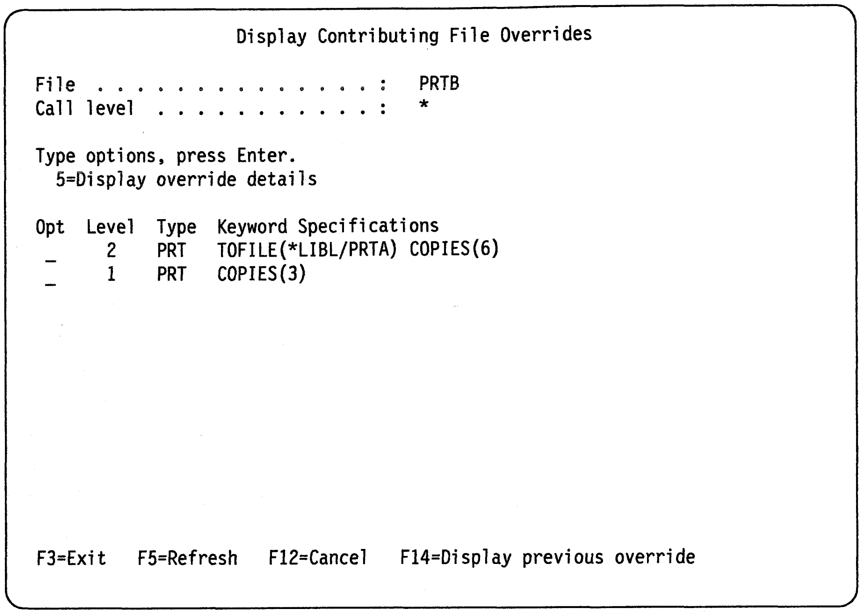
Command 12 monitors for message CPF9842 on the preceding command. The monitor specifies no action to be taken, but will prevent a function check if the message is sent.

Command 13 displays the merged overrides at levels 1 to 3 for all files in keyword-parameter form, as shown in the next display. File DBC is overridden to file DBD because of command 9 (commands 5 and 2 are therefore not effective), file DSPE is overridden to file DSPF because of command 8, and file PRTB is overridden to file PRTA and COPIES(3) because of commands 4 and 1.



```
                    Display All Merged File Overrides

 Call level . . . . . . . . . . . :    *

 Type options, press Enter.
   5=Display override details  8=Display contributing file overrides


 Opt  File     Type  Keyword Specifications
  _   DSPE     DSP   TOFILE(*LIBL/DSPF)
  8   PRTB     PRT   TOFILE(*LIBL/PRTA) COPIES(3)
  _   DBC      DB    TOFILE(*LIBL/DBD)
  _   PRTA     PRT   COPIES(3)










 F3=Exit   F5=Refresh   F11=All file overrides   F12=Cancel
```

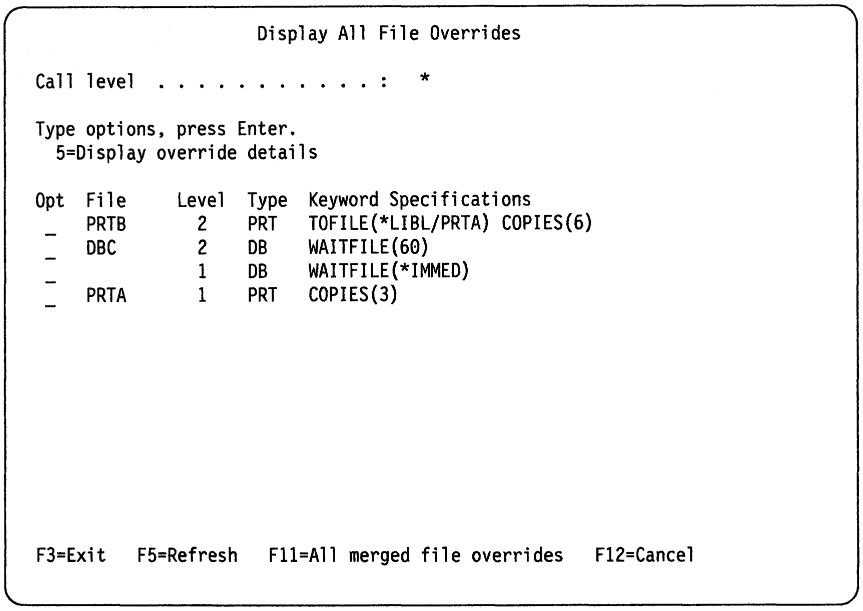
If you enter a 5 on the line for PRTB, you get a detail display like the one shown by command 6. If you enter an 8 on this same line, you get a display showing commands 4 and 1 on separate lines, as shown in the following display. These are the overrides that were merged to form the PRTB override.



```
                       Display Contributing File Overrides

File . . . . . . . . . . . . . . . :   PRTB
Call level . . . . . . . . . . . . :   *

Type options, press Enter.
  5=Display override details

Opt  Level  Type  Keyword Specifications
 _     2    PRT   TOFILE(*LIBL/PRTA) COPIES(6)
 _     1    PRT   COPIES(3)










F3=Exit   F5=Refresh   F12=Cancel   F14=Display previous override
```

Command 14 causes a return to level 2, and level 3 is deleted. The overrides issued at level 3 are implicitly deleted.

Command 15 displays all overrides issued from level 1 to the current call level (level 2), as shown in the next display. The overrides specified in commands 1, 2, 4, and 5 are displayed in keyword-parameter form. The override issued in command 9 is not displayed because call level 3 is no longer active. Pressing F11 on this display allows you to see a display similar to the one shown by command 13.



```
                         Display All File Overrides

Call level . . . . . . . . . . . . :   *

Type options, press Enter.
  5=Display override details

Opt  File      Level  Type  Keyword Specifications
 _   PRTB        2    PRT   TOFILE(*LIBL/PRTA) COPIES(6)
 _   DBC         2    DB    WAITFILE(60)
 _               1    DB    WAITFILE(*IMMED)
 _   PRTA        1    PRT   COPIES(3)








F3=Exit   F5=Refresh   F11=All merged file overrides   F12=Cancel
```

Command 16 causes a return to level 1, and level 2 is deleted. The overrides issued at level 2 are implicitly deleted.

Command 17 displays all overrides issued from level 1 to level 2 in keyword-parameter form. Because level 2 is no longer active, only the overrides specified at level 1 in commands 1 and 2 are displayed.

File Redirection

File redirection refers to using overrides to change the file name and library or the type of the file to be processed. For example, you can substitute one database file for another or change from using an ICF file to using a display file. This section applies to using an application program only. System code may or may not support file redirection. Refer to "Effect of Overrides on Some System Commands" on page 2-15 for rules on how system code processes overrides.

You use the OVRDBF command to redirect a file to a Distributed Data Management (DDM) file. If the remote system is another AS/400 system, all normal rules discussed in this chapter apply. If the remote system is not an AS/400 system or System/38, then normally you should not specify an expiration date or end-of-file delay. For more information, refer to the *DDM User's Guide*.

When you replace the file that is used in a program with another file of the same type, the new file is processed in the same manner as the original file. If a field level file, or any other file containing externally described data is redirected, it usually is necessary to either specify LVLCHK(*NO) or recompile the program. With level checking turned off, it is still necessary that the record formats in the file be compatible with the records in the program. If the formats are not compatible, the results cannot be predicted.

Overrides that have a TOFILE parameter value other than *FILE remove any database member specifications that may be on overrides applied at higher call levels. The member name will default to *FIRST unless it is specified with the change to the file name or library or on another override at a lower call level.

If you change to a different type of file, the device-dependent characteristics are ignored and records are read or written sequentially. Some device parameters must be specified in the new device file or the override. Defaults are taken for others. The effect of specific redirection combinations is described later in this section.

Any attributes specified on overrides of a different file type than the final file type are ignored. The parameters SPOOL, SHARE, and SECURE are exceptions to this rule. They will be accepted from any override applied to the file, regardless of device type.

Some redirection combinations present special problems due to the specific characteristics of the device. In particular:

- File redirection is not recommended for save files.
- Nonsequentially processed database files can be redirected only to another database file or a DDM file.
- Display files and ICF files that use multiple devices (MAXDEV or MAXPGMDEV > 1) can be redirected only to a display file or ICF file.
- Redirecting a display file to any other file type, or another file type to a display file, requires that the program be recompiled with the override active if there are any input-only or output-only fields. This is necessary because the display file omits these fields from the record buffer in which they are not used, but other file types do not.

Table 2-1 summarizes valid file redirections:

Table 2-1. File Redirections

	From-File					
To-File	**Printer**	**ICF**	**Diskette**	**Display**	**Database**	**Tape**
Printer	O*	O	O	O	O	O
ICF	O	I/O O I	O I	I/O O I	O I	O I
Diskette	O	O I	O I	O I	O I	O I
Display	O	I/O O I	O I	I/O O I	O I	O I
Database	O	O I	O I	O I	O I	O I
Tape	O	O I	O I	O I	O I	O I

I = input file O = output file I/O = input/output file
* = redirection to a different type of printer

To use this chart, identify the file type to be overridden in the FROM-FILE columns and the file type overriding in the TO-FILE column. The intersection specifies an I or O or both, meaning that the substitution is valid for these two file types when used as input files or as output files.

For instance, you can override a diskette output file with a tape output file, and a diskette input file with a tape input file. The chart refers to file type substitutions only. That is, you cannot change the program function by overriding an input file with an output file.

The following charts describe the specific defaults taken and what is ignored for each redirection combination:

From	To
Printer	ICF: Records are written to the file one at a time. Printer control information is ignored.
	Display: Records are written to the display with each record overlaying the previous record. For program-described files, you can request each record using the Enter key. Printer control information is ignored.
	Database: Records are written to the database in sequential order. Printer control information is ignored.
	Diskette: The amount of data written on diskette is dependent on the exchange type of the diskette. Diskette label information must be provided in the diskette file or on an override command. Printer control information is ignored. Refer to Chapter 8, "Diskette Support" for a description of exchange types.
	Tape: Records are written to the tape in sequential order. Tape label information must be specified in the tape file or on an override command. Printer control information is ignored.

From	To
ICF input	Display: Records are retrieved from the display one at a time. Type in the data for each record and press the Enter key when the record is complete.
	Database: Records are retrieved from the database.
	Diskette: Records are retrieved in sequential order. Diskette label information must be provided in the diskette file or on an override command. Refer to Chapter 8, "Diskette Support" for a description of exchange types.
	Tape: Records are retrieved in sequential order. Tape label information must be specified in the tape file or on the override command.

From	To
ICF output	Printer: Records are printed and folding or truncating is performed as specified in the printer file. Display: Records are written to the display with each record overlaying the previous record. Database: Records are written to the database in sequential order. Diskette: The amount of data written on diskette is dependent on the exchange type of the diskette. Diskette label information must be provided in the diskette file or on an override command. Refer to Chapter 8, "Diskette Support" for a description of exchange types. Tape: Records are written to the tape in sequential order. Tape label information must be specified in the tape file or on the override command.

From	To
ICF input/output	Display: Input records are retrieved from the display one at a time. Type in the data for each record and press the Enter key when the record is complete. Output records are written to the display with each record overlaying the previous input or output record. Input and output records are essentially independent of each other and may be combined in any manner.

From	To
Diskette input	ICF: Records are retrieved from the ICF file one at a time. Display: Records are retrieved from the display one at a time. Type in the data for each record and press the Enter key when the record is complete. A non-field-level device file must be specified. Diskette label information is ignored. Database: Records are retrieved in sequential order. Diskette label information is ignored. Tape: Records are retrieved in sequential order. If a label value is specified in the program, that value is used as the label for the tape file.

From	To
Diskette output	ICF: Records are written to the ICF file one at a time. Database: Records are written to the database in sequential order. Display: Records are written to the display with each record overlaying the previous record. You can request each output record using the Enter key. Printer: Records are printed and folding or truncating is performed as specified in the printer file. Tape: Records are written on tape in sequential order.

From	To
Display input	ICF: Records are retrieved from the ICF file one at a time. Diskette: Records are retrieved in sequential order. Diskette label information must be provided in the diskette file or on an override command. Refer to Chapter 8, "Diskette Support" for a description of exchange types. Database: Input records are retrieved. Tape: Records are retrieved in sequential order. Tape label information must be specified in the tape file or on an override command.

From	To
Display output	ICF: Records are written to the ICF file one at a time. Database: Records are written to the database in sequential order. Diskette: The amount of data written on diskette is dependent on the exchange type of the diskette. Diskette label information must be provided in the diskette file or on an override command. Refer to Chapter 8, "Diskette Support" for a description of exchange types. Tape: Records are written on tape in sequential order. Tape label information must be specified in the tape file or on an override command. Printer: Records are printed and folding or truncating is performed as specified in the printer file.

From	To
Display input/output	ICF: Input records are retrieved from the ICF file one at a time. Output records are written to the ICF file one at a time. The relationship between the input and output records is determined by the application program.

From	To
Database input (sequentially processed)	ICF: Records are retrieved from the ICF file one at a time. Display: Records are retrieved from the display one at a time. Type in the data for each record and press the Enter key when the record is complete. A non-field-level device file must be specified. Diskette: Records are retrieved in sequential order. Diskette label information must be provided in the diskette file or on an override command. Refer to Chapter 8, "Diskette Support" for a description of exchange types. Tape: Records are retrieved from tape in sequential order. Tape label information must be specified in the tape file or on an override command.

From	To
Database output (sequentially processed)	Printer: The number of characters printed is determined by the page size specified. If folding is specified, all of a record is printed. ICF: Records are written to the ICF file one at a time. Display: Records are written to the display with each record overlaying the previous record. You can request each output record using the Enter key. Diskette: The amount of data written on diskette depends on the exchange type of the diskette. Diskette label information must be provided in the diskette file or on an override command. Refer to Chapter 8, "Diskette Support" for a description of exchange types. Tape: Records are written on tape in sequential order. Tape label information must be specified in the tape file or on an override command.

From	To
Tape input	ICF: Records are retrieved from the ICF file one at a time. Display: Records are retrieved from the display one at a time. Type in the data for each record and press the Enter key when the record is complete. A non-field-level device file must be specified. Tape label information is ignored. Database: Records are retrieved in sequential order. One record is read as a single field. Tape label information is ignored. Diskette: Records are retrieved in sequential order. If a label value is specified in the program, that value is used as the label for the diskette file.

From	To
Tape output	Printer: Records are printed, and folding or truncating is performed as specified in the printer file. ICF: Records are written to the ICF file one at a time. Tape label information is ignored. Diskette: The amount of data written on diskette depends on the exchange type of the diskette. If a label value is specified in the program, that value is used as the label for the diskette file. Refer to Chapter 8, "Diskette Support" for a description of exchange types. Display: Records are written to the display with each record overlaying the previous record. You can request each output record using the Enter key. Database: Records are written to the database in sequential order.

Overriding Program Device Entries

In addition to the file attributes and record formats similar to those in other device files, an ICF file also contains program device entries which provide the link between the application and each of the remote systems or devices with which your program communicates.

The following lists the CL commands that provide override functions for device entries:

DLTOVRDEVE Delete Override Device Entry: Deletes one or more program device overrides that were previously specified in a call level.

OVRICFDEVE Override Intersystem Communications Function Device Entry: Used to temporarily add the program device entry and the remote location name to the ICF file or to override a program device entry with the specified remote location name and attributes for an ICF file.

A program device entry has two functions:

- It associates a program device name with a remote location.
- It establishes a set of program communications-type dependent attributes.

Multiple program device entries can be defined. Each program device entry must have a unique program device name. The maximum number of entries is determined by the MAXPGMDEV parameter specified at file creation.

Program device entries may be defined by the Add Intersystem Communications Function Device Entry (ADDICFDEVE) command or the OVRICFDEVE command. The add command makes a permanent addition to the file, and the override command makes a temporary change to the program device information. It is not necessary to add a program device entry before overriding it. Several add commands may be used to add multiple program devices to the same file. Several override commands may be used to change different device entries. Refer to the *Communications Programmer's Guide* for more information on program device entries and for a list of the parameters supported by each communications type.

Overriding Remote Location Name

The device entry override may be used to temporarily define or change the remote location name associated with the program device entry.

The following example demonstrates the use of the OVRICFDEVE command to override the remote location name:

```
OVRICFDEVE PGMDEV(PGMDEVA) RMTLOCNAME(CHICAGO)
CALL RPGPGM
```

In this example, when RPGPGM specifies PGMDEVA, remote location CHICAGO is used. Refer to the *Communications Programmer's Guide* for more information on remote location name and its relationship to configuration.

Overriding Session Attributes

The device entry override may also be used to temporarily change the characteristics of the communications session that is established when the program device is acquired.

Although some of the session attributes have system-level defaults, the default for the majority of these attributes is information supplied during communications configurations.

Session attributes are identified as parameters on the ADDICFDEVE or OVRICFDEVE command. Parameters not specified on either command take on the appropriate system default or specified configuration value. If the same parameter is specified on both the ADDICFDEVE and OVRICFDEVE commands, the value specified on OVRICFDEVE overrides the value declared on the ADDICFDEVE command.

The following example demonstrates the use of the OVRICFDEVE command to override the format selection processing attribute:

```
OVRICFDEVE PGMDEV(PGMDEVA) FMTSLT(*PGM)
```

In this example, format selection is changed to *PGM. This overrides what was previously defined in the program device entry. Refer to the appropriate communications programming manual for more information on the use of the session attributes. Refer to the *CL Reference* for more information on the format and allowable values of the parameters on the OVRICFDEVE command.

Overriding Remote Location Name and Session Attributes

This form of the override device entry is a combination of the previous two forms. With this form of override, you can override the remote location that is used by a program, and you can also override the session attributes.

Applying OVRICFDEVE Command

Device entry overrides follow most of the same rules as file overrides. They are effective from the time they are specified until they are replaced or deleted or until the program in which they were specified ends. Any program device entry overrides that are in effect at the time the device is acquired are applied.

The OVRICFDEVE command can be used to initialize an environment or change the environment while running.

In the following example, the OVRICFDEVE commands are initializing an environment:

```
Override 1   OVRICFDEVE PGMDEV(PGMDEV1)  +
               RMTLOCNAME(BOSTON) . . .
Override 2   OVRICFDEVE PGMDEV(PGMDEV2)  +
               RMTLOCNAME(ROCHMN) . . .
             CALL PGM(A)
             CALL PGM(B)
             .
             .
             .
             CALL PGM(X)
```

When the program uses any ICF file and acquires the program device named PGMDEV1, then the remote location named BOSTON and attributes from override 1 are used when establishing the communication session.

When the program uses an ICF file and acquires the program device named PGMDEV2, then the remote location named ROCHMN and attributes from override 2 are used when establishing the communication session.

In the following example, the OVRICFDEVE commands are used to change the running environment:

```
Override 1   OVRICFDEVE PGMDEVE(PGMDEV1)  +
               RMTLOCNAME(BOSTON) . . .
             CALL PGM(A)
Override 2   OVRICFDEVE PGMDEVE(PGMDEV2)  +
               RMTLOCNAME(ROCHMN) . . .
             CALL PGM(A)
```

The first time program A is called, an ICF file is opened and the program device named PGMDEV1 acquired. The remote location named BOSTON and attributes from override 1 are used when establishing the communication session.

The second time program A is called, an ICF file is opened and the program device named PGMDEV2 is acquired. The remote location named ROCHMN and attributes from override 2 are used when establishing the communication session.

Applying OVRICFDEVE from Multiple Call Levels

When you have more than one override for the same program device at several call levels (nested calls), the order in which the overrides are applied to the program device is from the innermost override (the override with the largest call level) to the outermost override (the override with the smallest call level).

To prevent override at lower call levels from being applied, see "Applying OVRICFDEVE with SECURE" on page 2-34.

In the following example, override 2 is the innermost, and override 1 is the outermost:

```
Override 1   OVRICFDEVE PGMDEV(PGMDEV1)  +
               FMTSLT(*PGM) BATCH(*NO)
             CALL PGM(A)

               Program A
Override 2     OVRICFDEVE PGMDEV(PGMDEV1)  +
                 FMTSLT(*RECID) APPID(PAYROLL)
               CALL PGM(X)
```

When program X acquires program device PGMDEV1, the following attributes are used:

FMTSLT(*PGM)	From Override 1
BATCH(*NO)	From Override 1
APPID(PAYROLL)	From Override 2

The attribute of FMTSLT(RECID) specified in override 2 is not used because it was overridden by FMTSLT(*PGM) specified in override 1. Override 1 overrides override

2. If there is a third override for program device PGMDEV1 embedded in program X, it is overridden by override 2 and then override 1.

A similar situation exists when you change the remote location to be used with the program device and you also change some of the attributes of the program device. For example:

```
Override 1   OVRICFDEVE PGMDEV(PGMDEV1)  +
               RMTLOCNAME(NYCAPPC)
             CALL PGM(A)

               Program A
Override 2     OVRICFDEVE PGMDEV(PGMDEV1)  +
                 RMTLOCNAME(MPLSAPPC) CNVTYPE(*USER)
               CALL PGM(X)
```

When program X is ready to acquire PGMDEV1, it acquires remote location NYCAPPC instead of MPLSAPPC (because override 1 overrides override 2 remote location). Also, the conversation type is *USER (because of override 2).

Applying OVRICFDEVE with SECURE

On occasion, you may want to protect program devices used by a program from overrides at lower call levels.

You can prevent additional program device overrides by coding the SECURE(*YES) parameter on a program device override command for each program device needing protection. This protects you from overrides at lower call levels.

The following shows an example of a protected program device:

```
Override 1   OVRICFDEVE PGMDEV(PGMDEV1)  +
               RMTLOCNAME(BOSTON)
Override 2   OVRICFDEVE PGMDEV(PGMDEV4)  +
               RMTLOCNAME(ROCHMN)
             CALL PGM(A)

               Program A
Override 3     OVRICFDEVE PGMDEV(PGMDEV5)  +
                 RMTLOCNAME(NYC)
               CALL PGM(B)

                 Program B
Override 4       OVRICFDEVE PGMDEV(PGMDEV1)  +
                   RMTLOCNAME(MPLS) SECURE(*YES)
                 CALL PGM(X)
```

When program X acquires program device PGMDEV1 for an ICF file, the remote location MPLS and attributes from override 4 are used. Because override 4 specifies SECURE(*YES), override 1 is not applied.

Deleting Device Entry Overrides

When a program returns from a call level containing program device entry overrides, the overrides are deleted, just as any file overrides are deleted. When control is transferred to another program (TFRCTL command) so that the program is running at the same call level, the overrides are not deleted. If you want to delete an override before the run is completed, you can use the Delete Override Device Entry (DLTOVRDEVE) command. This command only deletes overrides in the call level in which the command is entered. A DLTOVRDEVE command does not delete the effects of an ADDICFDEVE command. To remove an ADDICFDEVE command, you must use the Remove Intersystem Communications Function Device Entry (RMVICFDEVE) command. To identify an override, use the program device name specified on the PGMDEV parameter of the override. You can delete all overrides at this call level by specifying value *ALL for the PGMDEV parameter. For example:

```
Override 1          OVRICFDEVE PGMDEV(PGMDEV1)  +
                      RMTLOCNAME(BOSTON)
Override 2          OVRICFDEVE PGMDEV(PGMDEV4)  +
                      RMTLOCNAME(ROCHMN)
Override 3          OVRICFDEVE PGMDEV(PGMDEV5)  +
                      RMTLOCNAME(NYC)
Delete Override 1   DLTOVRDEVE PGMDEV(PGMDEV1)
Delete Override 2   DLTOVRDEVE PGMDEV(*ALL)
```

Delete override 1 causes override 1 to be deleted. Delete override 2 causes the remaining overrides (overrides 2 and 3) to be deleted.

Displaying Device Entry Overrides

Device entry overrides are not displayed by the Display Override (DSPOVR) command. There is no corresponding command to display device entry overrides.

Chapter 3. Copying Files

You can use the copy function to move data between device files, database files, or device and database files. Unlike traditional copy utilities, the AS/400 copy function is field-level sensitive. Therefore, if you use the copy function, you can rearrange, enlarge, or drop any of the fields. The system also provides a way to define database files. Specific copy commands simplify dealing with tape and diskette devices and database source files.

You can copy records to and from files using the following commands:

CPYF Copy File: Copies all or part of a file from the database or external device to the database or external device.

CPYFRMDKT Copy From Diskette: Copies from a diskette file to a database or device file.

CPYTODKT Copy To Diskette: Copies a database or device file to a diskette file.

CPYFRMTAP Copy From Tape: Copies from a tape file to a database or device file.

CPYTOTAP Copy To Tape: Copies a database or device file to a tape file.

CPYSRCF Copy Source File: Copies a database source file to a physical source file.

If you use the Copy File (CPYF) command, you can copy the whole file or a subset of the file by specifying certain selection values on the command parameters. You can use these commands to copy data and source:

- Between database files. Records can be copied from a physical or logical file, but can only be copied to a physical file.
- From an external device (such as diskette or tape) to the database, and from the database to an external device.
- From an external device to another external device.
- From a spooled inline file to the database or to an external device. (Inline data files are processed by a program as a program-described file. The records in the file are processed sequentially from the beginning of the file to the end of the file. Inline data files can be either unnamed or named.)
- From a local database file to a file on a remote system.
- From a file on a remote system to a local database file.

The combinations of device and database files for which copy operations can be performed are shown in Table 3-1 on page 3-2. An X indicates that the corresponding file types are a valid combination for copying a file.

Table 3-1. Copy Operations

From-File	To Physical File	To Printer File	To Diskette File	To Tape File	To *PRINT File[1]	To DDM
Physical	X[2]	X	X	X	X	X
Logical	X[2]	X	X	X	X	X
Diskette	X	X	X[3]	X	X	X
Tape	X	X	X	X	X	X
Spooled Inline[4]	X	X	X	X	X	X
DDM[5]	X	X	X	X	X	

[1] If TOFILE(*PRINT) is specified, the from-file records are copied to the IBM-supplied printer device file QSYSPRT and formatted according to the OUTFMT parameter.

[2] If the to-file does not exist before the copy operation and the from-file is a physical or logical file, the copy operation will create a physical file as the to-file if you specified CRTFILE(*YES) on the CYPF command.

[3] If the from-file and the to-file are both diskette files, the to-file must be a spooled file.

[4] A spooled inline file (which is handled like a device file) is included as part of a batch job when the job is read by a reader program.

[5] Distributed data management (DDM) does not allow for copying from one remote system to another remote system.

While copying records, the CPYF command can perform the following functions:

- Copy from or to the first file member, a particular file member, a generic set of members, or all file members (FROMMBR and TOMBR parameters).
- Add a member to a physical to-file if the member does not exist.
- Add records to an existing file member or replace the contents of an existing member (MBROPT parameter).
- Select certain records to copy by one of the following methods:
  - Selecting records by record format name when a multiformat logical file is copied (RCDFMT parameter).
  - Specifying records starting at a relative record number and/or ending at a relative record number (FROMRCD and TORCD parameters).
  - Specifying records starting with a specific record key value and/or ending with another specific record key value (FROMKEY and TOKEY parameters).
  - Specifying the number of records to be copied (NBRRCDS parameter).
  - Selecting records by the contents of one or more character positions in the record or in a field in the record (INCCHAR parameter).
  - Selecting records by the values contained in one or more fields in the record (INCREL parameter).
  - Disregard or include deleted records in the from-file during the copy if processing the from-file in arrival sequence (COMPRESS parameter).
- Print copied records and/or excluded records (PRINT parameter) in a specified format (OUTFMT parameter).

- Copy records whose from-file and to-file record formats are different (FMTOPT parameter). When formats are different, you can:
  - Map fields whose names are the same in the from-file and to-file record formats and whose field attributes are compatible (*MAP value).
  - Drop fields in the from-file record format that do not exist in the to-file record format (*DROP value).
  - Copy data directly (left to right) disregarding any differences (*NOCHK value).
- Copy from a source file to a data file or from a data file to a source file. If the from-file or to-file is a device file, this function is automatic. If both files are database files, FMTOPT(*CVTSRC) is required.
- Change sequence numbers and/or zero dates in the sequence number and date source fields when copying to a source physical file (SRCOPT parameter). When renumbering is to be done, the starting sequence number and the increment value can be specified (SRCSEQ parameter).
- End the copy after a specified number of recoverable errors are encountered (ERRLVL parameter).
- Create the to-file as part of the copy operation (CRTFILE parameter).

Refer to the following tables (Table 3-2 on page 3-4 and Table 3-3 on page 3-5) for a summary of what specific copy functions (using the CPYF command) can be used for copying records by the types of files being copied from and to. The functions with their associated parameters are listed down the left side, and the file types (and if each can be a from-file and/or a to-file) are shown across the top. An X indicates that the associated parameter is valid for the type and use of file under which it occurs.

Table 3-2. Summary of Copy Functions for Database Files

		Database Files[1]			
		Physical		**Logical**	
Copy Function	**Parameter**	**From**	**To**	**From**	**To**
Select files	FROMFILE	X		X	
	TOFILE		X		
Select members	FROMFILE	X		X	
	TOFILE		X		
Add to or replace existing records	MBROPT		X		
Create the to-file	CRTFILE[2]	X	X	X	
Print copied and/or excluded records	PRINT[3]	X	X	X	
Select by record format	RCDFMT			X	
Select by relative record number	FROMRCD	X		X[4]	
	TORCD	X		X[4]	
Select by key field value	FROMKEY	X		X	
	TOKEY	X		X	
Specify number of records to copy	NBRRCDS	X		X	
Select by character content	INCCHAR	X		X	
Select by field value	INCREL	X		X	
Process different database record formats	FMTOPT	X	X	X	
Update sequence number and/or date	SRCOPT	X	X	X	
Specify start value and increment	SRCSEQ	X	X	X	
Print character and/or hex format	OUTFMT[3]	X	X	X	
Maximum recoverable errors allowed	ERRLVL	X	X	X	
Disregard or include deleted records	COMPRESS[5]	X	X		

[1] DDM files will appear to act like database files, with exceptions noted in the *DDM User's Guide.*

[2] If the to-file does not exist before the copy operation and the from-file is a physical or logical file, the copy operation will create a physical file as the to-file if you specified CRTFILE(*YES) on the CPYF command.

[3] You can specify a program-described printer file so that the copy will produce a list with no special formatting or page headings, or you can specify TOFILE(*PRINT) to produce a formatted list. You can specify PRINT(*COPIED) to produce a formatted list of the copied records, and you can specify PRINT(*EXCLD) to produce a formatted list of the records excluded by the INCCHAR or INCREL parameters. When you request a list by specifying the TOFILE(*PRINT) parameter, the OUTFMT parameter specifies whether the data is printed in character or in both character and hexadecimal form.

[4] You can specify the FROMRCD and TORCD parameter values for a logical file if it has an arrival sequence access path.

[5] You cannot specify COMPRESS(*NO) if:

- The to-file member or a logical file member based on the to-file member has a keyed access path with any of the following attributes:
  - Unique keys (UNIQUE keyword specified in the DDS)
  - Floating-point key field or logical numeric key field and not MAINT(*REBLD)
  - Select/omit specifications in the DDS (without the DYNSLT keyword specified) and not MAINT(*REBLD)
- Field-level mapping or source/data conversion is required (FMTOPT parameter).
- If an EOFDLY wait time is specified for the from-file on an Override Database File (OVRDBF) command.

Note: To copy deleted records, the from-file must be processed in arrival sequence.

Table 3-3. Summary of Copy Functions for Device Files

		Device Files							
		Spooled Input		Diskette		Tape		Printer	
Copy Function	**Parameter**	**From**	**To**	**From**	**To**	**From**	**To**	**From**	**To**
Select files	FROMFILE	X		X[3]		X			
	TOFILE				X[3]		X		X
Select members	FROMMBR			X		X			
	TOMBR				X		X		
Add to or replace existing records	MBROPT								
Create the to-file	CRTFILE[1]								
Print copied and/or excluded records	PRINT[2]	X		X	X	X	X		X
Select by record format	RCDFMT								
Select by relative record number	FROMRCD	X		X		X			
	TORCD	X		X		X			
Select by key field value	FROMKEY								
	TOKEY								
Specify number of records to copy	NBRRCDS	X		X		X			
Select by character content	INCCHAR	X		X		X			
Select by field value	INCREL								
Process different database record formats	FMTOPT								
Update sequence number and/or date	SRCOPT								
Specify start value and increment	SRCSEQ								
Print character and/or hex format	OUTFMT[2]	X		X	X	X	X		X
Maximum recoverable errors allowed	ERRLVL					X			
Disregard or include deleted records	COMPRESS[4]								

[1] If the to-file does not exist before the copy operation and the from-file is a physical or logical file, the copy operation will create a physical file as the to-file if you specified CRTFILE(*YES) on the CPYF command.

[2] You can specify a program-described printer file so that the copy will produce a list with no special formatting or page headings, or you can specify TOFILE(*PRINT) to produce a formatted list. You can specify PRINT(*COPIED) to produce a formatted list of the copied records, and you can specify PRINT(*EXCLD) to produce a formatted list of the records excluded by the INCCHAR or INCREL parameter. When you request a list by specifying the TOFILE(*PRINT) parameter, the OUTFMT parameter specifies whether the data is printed in character or in both character and hexadecimal form.

[3] If the from-file and to-file are diskette files, you must specify that the to-file be spooled (SPOOL(*YES)) on a CRTDKTF, CHGDKTF, or OVRDKTF command.

[4] You cannot specify COMPRESS(*NO) if:

- The to-file member or a logical file member based on the to-file member has a keyed access path with any of the following attributes:
  - Unique keys (UNIQUE keyword specified in the DDS)
  - Floating-point key field or logical numeric key field and not MAINT(*REBLD)
  - Select/omit specifications in the DDS (without the DYNSLT keyword specified) and not MAINT(*REBLD)
- Field-level mapping or source/data conversion is required (FMTOPT parameter).
- If an EOFDLY wait time is specified for the from-file on an OVRDBF command.

Note: To copy deleted records, the from-file must be processed in arrival sequence.

Basic Copy Function

As indicated in Table 3-2 and Table 3-3, you can copy from a physical or logical database file, diskette file, tape file, or from a spooled inline file. The to-file can be a physical database file, diskette file, tape file, program-described printer file, or *PRINT. When TOFILE(*PRINT) is specified, the CPYSRCF command uses a different format from the other copy commands. This format is organized to show source information in a more readable format and for multiple member copies, the members are copied and listed in alphabetical order.

If you are copying from a database file and the to-file does not exist, you must specify CRTFILE(*YES) and identify the file name and library name on the TOFILE parameter in order to create the to-file. You cannot copy from a diskette to a diskette unless the to-file is spooled and a diskette spool writer is not active.

The from-file, to-file, and the QSYSPRT printer file (if TOFILE(*PRINT), PRINT(*COPIED), or PRINT(*EXCLD) is specified) are opened with the SHARE(*NO) attribute. Because the copy may not function correctly with a shared file, it will end with an error message if the from-file, to-file, or QSYSPRT printer file is overridden to SHARE(*YES) and the file has already been opened in the job.

If you specify TOFILE(*PRINT), the records are copied to the IBM-supplied printer file QSYSPRT, and the list is formatted by the OUTFMT parameter.

If you do not want a formatted list or if you want to use first-character forms control (CTLCHAR(*FCFC) on the Create Printer File (CRTPRTF) or Override with Printer File (OVRPRTF) command), you should specify a program-described printer file name (such as QSYSPRT) instead of *PRINT on the TOFILE parameter.

File Types

When the from-file and to-file are different types (source and data), the following is true:

- If the from-file or to-file is a device file (or a spooled inline file), the copy function will automatically add or delete the source sequence number and date fields for each record copied.
- If the from-file and to-file are database files, you must specify FMTOPT(*CVTSRC) to perform the operation. The sequence number and date fields are added or deleted as they are for a device file, and the data part of each record is copied without regard to the field definitions in the file record formats. For a source physical to-file, the SRCSEQ parameter can be used to control how sequence numbers are created if SRCOPT(*SEQNBR) is also specified.

Record Sequence

With the copy function, you can process records in a database file in either arrival sequence or keyed sequence (if the file has a keyed access path). An arrival sequence copy transfers records in the order in which they physically exist in the from-file. This order is represented by relative record numbers. The relative record number is the position where the records physically exist in storage. Because records are always added to the end of the file, the relative record number represents the order in which records arrived in the file.

A keyed sequence copy selects and transfers records by key value from a keyed physical file. This may result in a different physical order in the to-file. The to-file will be a reorganized version of the from-file. The relative record number of a specific record may change when a file is copied by key value:

Relative Record Number	Arrival Sequence	Keyed Sequence
1	1011	0016
2	0762	0762
3	0810	0810
4	3729	1011
5	0016	3729

You can copy a keyed physical file in arrival sequence by specifying the FROMRCD or TORCD parameter on the CPYF command. When you do this, the keyed sequence access path is not used to retrieve the records in key sequence. The records are retrieved in arrival sequence. This is helpful when the physical relative record location in the file is significant and needs to remain the same as it is in the original file. Specifying FROMRCD(1) is a good way to copy all the records in arrival sequence. Copying a physical file in arrival sequence instead of keyed sequence is also faster.

The kind of copy you run is determined by the type of from-file and the method of selecting records to copy. In general, files are copied using their keyed sequence, if they have one, otherwise, their arrival sequence. For more information on the selection methods, refer to "Selecting Records to Copy" on page 3-17.

A copy from a keyed file to a keyed file usually places records at the end of the to-file in key field order, by the from-file key, regardless of their physical order in the from-file. But if you select records in the from-file by relative record number (using the FROMRCD or TORCD parameters), they are physically placed at the end of the to-file in relative record number order, regardless of their keyed sequence in the from-file. The following example shows the result of a CPYF command specifying from record 3 to record 5:

FROM-FILE			TO-FILE	
Relative Record Number	Key		Relative Record Number	Key
1	1011		.	---
2	0762		.	---
3	0810	Arrival	1401	0810
4	3729	Sequence	1402	3729
5	0016	Copy	1403	0016

RSLH715-0

When the to-file has a keyed sequence, the records appear in correct order in the to-file when using the keyed sequence access path. A copy by relative record number always copies by arrival sequence.

Resending Copy File Completion Message

If the CPYF command is run from a CL program, the completion message indicating the number of records copied is not sent directly to the system operator. You can direct this message to the system operator by resending it (SNDPGMMSG command) from the CL program, using the following CL program as an example:

```
PGM
DCL &MSGID TYPE(*CHAR) LEN(7)
DCL &MSGDTA TYPE(*CHAR) LEN(82)
CPYF FROMFILE(LIB1/XXX) TOFILE(LIB2/XXX) MBROPT(*ADD)
RCVMSG MSGID(&MSGID) MSGDTA(&MSGDTA) MSGTYPE(*COMP) RMV(*NO)
SNDPGMMSG MSGID(&MSGID) MSGF(QCPFMSG) MSGTYPE(*INFO) TOMSGQ(QSYSOPR)  +
  MSGDTA(&MSGDTA)
ENDPGM
```

The copy function sends one of the following completion messages for each from-file member/label successfully copied to the to-file:

- CPC2955 is the normal copy completion message.
- CPC2956 is used when COMPRESS(*NO) is specified.
- CPC2957 indicates that no records were copied.

Monitoring for Copy Errors

The escape message CPF2817 is sent to indicate many different error conditions. Except for the empty from-file member case which is described later, when this message is sent:

- A physical file is not created (even if CRTFILE(*YES) was specified on the CPYF command).
- No members are added to a to-file that is a physical file.
- No to-file member is cleared (even if MBROPT(*REPLACE) was specified).
- The to-file is not opened, so no file is created on a diskette or tape volume. If the to-file is spooled, no spool output file is created.
- No records are copied.

The CPF2817 escape message is always preceded by at least one diagnostic message that indicates the specific error condition. The message identifier of the diagnostic message which immediately precedes the CPF2817 escape is used as message replacement data (MSGDTA parameter on the SNDPGMMSG command) for the CPF2817 escape message. This allows you to monitor for specific error cases from the CPF2817 escape message by using the CMPDTA parameter on the MONMSG command.

For example, message CPF2802 is a diagnostic that indicates the from-file cannot be found. You can monitor for just the `from-file not found` condition as follows:

```
PGM
                              /* The replacement text of escape
                              CPF2817 contains the msg ID
                              CPF2802 for the 'from-file not
                              found' condition */
CPYF    FROMFILE(NOLIB/NOFILE) TOFILE(D504/KEY)  +
  FROMMBR(NOMBR) TOMBR(MBR1) MBROPT(*ADD)
MONMSG MSGID(CPF2817) CMPDTA(CPF2802) EXEC(SNDPGMMSG TOPGMQ(*EXT)  +
  MSG('File NOFILE in NOLIB not found'))
ENDPGM
```

Any error other than from-file not found, including any other error reported with a CPF2817 escape message, causes a function check in this program because the MONMSG command applies only to the CPF2817 escape when it has the compare data from message CPF2802.

The following messages can be sent as diagnostic messages immediately followed by a CPF2817 escape message. Some of these messages can also be sent as other message types (such as an informational or escape message). When the message is sent as a diagnostic message type, the message identifier appears in the replacement text of the CPF2817 escape message. You can monitor the condition by using the CMPDTA parameter on the MONMSG command:

CPF2801
CPF2802
CPF2803
CPF2804
CPF2805
CPF2806
CPF2808
CPF2810
CPF2811
CPF2812
CPF2813
CPF2814
CPF2816
CPF2819
CPF2820
CPF2821
CPF2822
CPF2823
CPF2825
CPF2826
CPF2827
CPF2831
CPF2832
CPF2833
CPF2834
CPF2836
CPF2837
CPF2839
CPF2840
CPF2841
CPF2842
CPF2843
CPF2844
CPF2847
CPF2848
CPF2849
CPF2851
CPF2853
CPF2854
CPF2855
CPF2856
CPF2857
CPF2860
CPF2861
CPF2862
CPF2863
CPF2864
CPF2865
CPF2868
CPF2869
CPF2870
CPF2871
CPF2872
CPF2873
CPF2874
CPF2877
CPF2878
CPF2879
CPF2881
CPF2883
CPF2884
CPF2890
CPF2891
CPF2893
CPF2960
CPF2962
CPF2963
CPF2965
CPF2969
CPF9807
CPF9808
CPF9820
CPF9830

Monitoring for Zero Records in the From-File

There are some special considerations for copy when the from-file is a physical or logical file and one or more members to be copied are empty. A member is considered empty in the following cases:

- If COMPRESS(*NO) is specified on the CPYF command and the from-file member contains no records.
- If COMPRESS(*YES) is specified or assumed for any copy command and the from-file members contain no undeleted records.

When the to-file is a printer file (including *PRINT), or when the to-file is a physical file and MBROPT(*ADD) is specified, empty from-file members are copied because no existing data will be destroyed. Each member copied is identified by a normal copy completion message. If the to-file is spooled, an empty spool output file is produced for each empty from-file member. If the CPYF command PRINT parameter specifies *COPIED or *EXCLD, the empty members are shown in the lists with no records printed.

An empty from-file member is never copied to a diskette or tape file, or to a physical file when MBROPT(*REPLACE) is specified. Empty from-file members are skipped for these types of to-files, and a CPF2869 message is sent (as either an informational or diagnostic message) to identify each empty member. The empty members are skipped to avoid destroying existing data. When an empty from-file member is skipped, the following considerations apply:

- A tape or diskette file is not produced on the output volume. If the diskette file is spooled, no spool output file is created.
- An existing to-file physical file member is not cleared.
- If the to-file does not exist and CRTFILE(*YES) was specified on the CPYF command, a physical file is created.
- If the to-file is a physical file and the to-file member does not exist, a member is added to the file.
- If the CPYF command PRINT parameter specifies *COPIED or *EXCLD, the empty members are not shown in the lists.

When the copy command specifies a generic name or *ALL for the FROMMBR parameter, each empty from-file member skipped is identified by message CPF2869, sent as an informational message. If all the from-file members are skipped, a CPF2870 diagnostic message is sent after all the CPF2869 informational messages, followed by a CPF2817 escape message.

When the copy command specifies a single member name or *FIRST for the FROMMBR parameter, or when there is an override for the from-file that forces a single member to be processed, an empty member that is skipped is identified by message CPF2869 sent as a diagnostic message. The CPF2869 diagnostic message is followed by a CPF2817 escape message.

In the following example, the from-file and to-file are both database files and EMPTY1 and EMPTY2 are empty members in the from-file.

```
PGM
                   /* No need to monitor for zero records
                      when MBROPT(*ADD) specified    */
CPYF    FROMFILE(D504/GEORGE) TOFILE(D504/KEN)  +
  FROMMBR(EMPTY1) TOMBR(MBR1) MBROPT(*ADD)
CPYF    FROMFILE(D504/GEORGE) TOFILE(D504/KEN)  +
  FROMMBR(EMPTY2) TOMBR(MBR2) MBROPT(*REPLACE)
MONMSG  MSGID(CPF2817) CMPDTA(CPF2869) +
  EXEC(CLRPFM  FILE(D504/KEN) MBR(MBR2))
                   /* Monitor for zero records and
                      send a message when all members
                      to copy are empty */
CPYF    FROMFILE(D504/GEORGE) TOFILE(D504/NEWFILE)  +
  FROMMBR(EMPTY*) TOMBR(NEWMBR) MBROPT(*REPLACE)
MONMSG  MSGID(CPF2817) CMPDTA(CPF2870) +
  EXEC(SNDPGMMSG TOPGMQ(*EXT) MSG('All members to  +
  copy are empty'))
ENDPGM
```

For the first CPYF command, MBROPT(*ADD) is specified, so an escape message is not sent to the program because of the empty from-file member. Note that if MBR1 does not exist before the copy, it is added to the to-file (if either the from-file member is empty or contains data).

For the second CPYF command, copy does not clear the to-file member when the from-file member is empty, so the MONMSG command after the second CPYF command starts the CLRPFM command to clear the to-file member when the from-file member is empty.

For the third CPYF command, the CPF2817 escape message has compare data of CPF2870 if all members to be copied are empty because the generic from-file member name, EMPTY*, requests that multiple members be copied.

Creating a Duplicate To-File Member

When your application requires an exact duplicate of the records in the to-file member (if either the from-file is empty or contains data), an alternative solution is to use the Clear Physical File Member (CLRPFM) command:

```
CLRPFM FILE(X) MBR(XYZ)
CPYF FROMFILE(Y) TOFILE(X) TOMBR(XYZ) MBROPT(*ADD)
```

Because MBROPT(*ADD) is specified, the CPYF command completes normally even if there is no data in file Y. MBR(XYZ) in file X contains an exact duplicate of the records in the member in file Y.

Adding and Replacing Records (MBROPT Parameter)

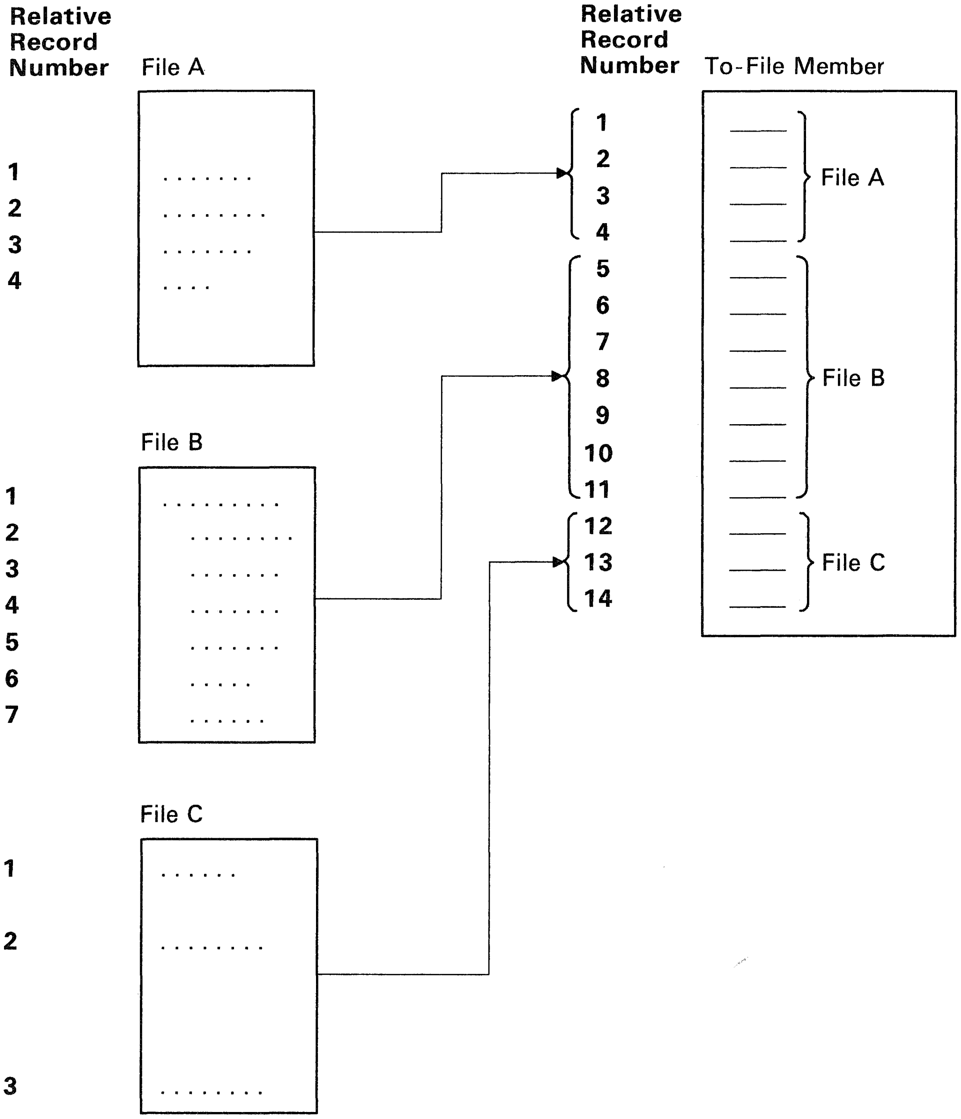
To copy to a physical file you must specify on the MBROPT parameter that copied data is either to be added to (*ADD) or replaced entirely (*REPLACE). The default value *NONE is valid only for a copy to a device file.

By specifying *REPLACE, you essentially clear the member. The copied records are the only records in the member when the operation is completed. You must have authority to clear the member in order to specify MBROPT(*REPLACE).

When you specify *ADD, each record copied is added to the end of the existing records in the member. It is important to note that this is always true, even for keyed files, though with keyed files, the added records appear to be merged in key sequence when accessed through a keyed access path.

When *ADD is specified, the copy completes normally even if the from-file contains no records. When *REPLACE is specified, the copy fails if the from-file does not contain any records.

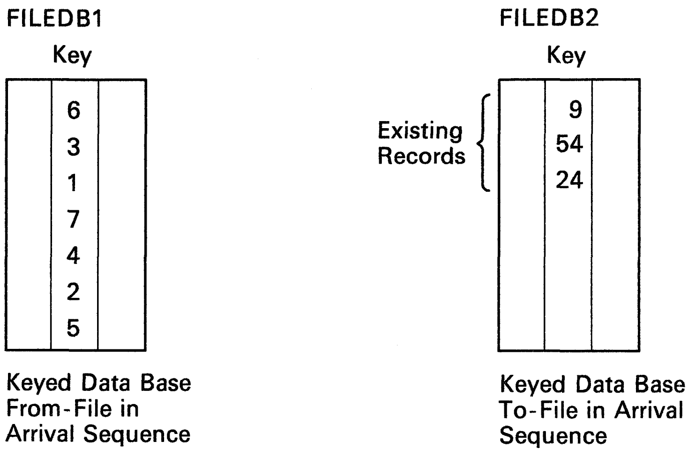
For three copies with MBROPT(*ADD) to a database file that is not keyed, the resulting to-file would look like this:



P7730138-0

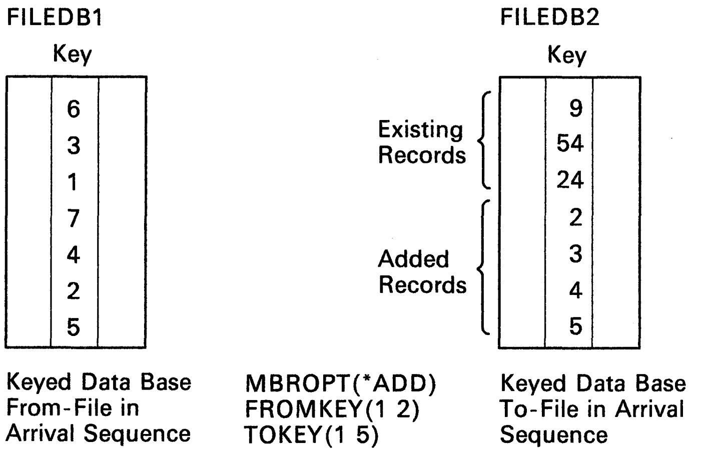
See "Adding or Changing Source File Sequence Number and Date Fields (SRCOPT/SRCSEQ Parameters)" on page 3-27 for source file considerations in this operation, and "Copying Deleted Records (COMPRESS Parameter)" on page 3-22 for considerations for deleted records.

If MBROPT(*ADD) is specified, records are always physically added at the end of the file, even if it is a keyed sequence file. In the following illustration, FILEDB1 is a keyed physical from-file, and FILEDB2 is a keyed physical to-file. The files are shown as they physically appear in storage. FILEDB2 already has three records in it:


FILEDB1
Key
6
3
1
7
4
2
5
Keyed Data Base From-File in Arrival Sequence
FILEDB2
Key
Existing Records
9
54
24
Keyed Data Base To-File in Arrival Sequence

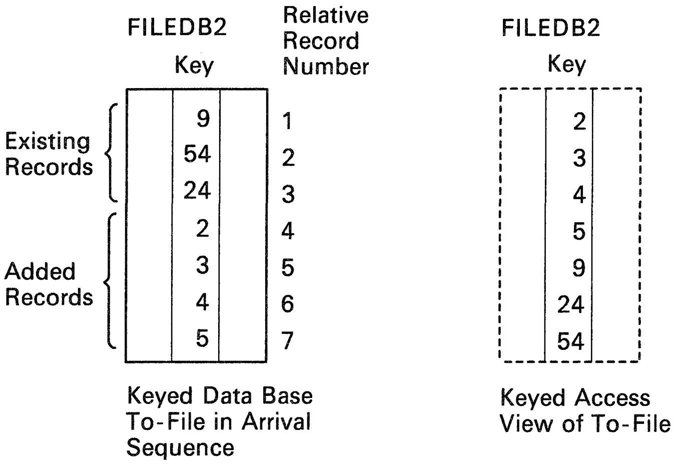
P7730139-0

If you specify MBROPT(*ADD), FROMKEY(1 2), and TOKEY(1 5), four records are added in key field order to the end of FILEDB2:



P7730140-0

The added records, however, appear to be merged in the new file when viewed through a keyed sequence access path:



P7730141-0

Several ways to select records for copying are explained in this chapter. One method is selection by relative record number. (See "Selecting Records Using Relative Record Numbers (FROMRCD/TORCD Parameters)" on page 3-18.) Using the preceding example, if you selected records to copy to a third file from FILEDB2 by relative record number, from number 3 through 5, you would copy the records with a key value of 24, 2, and 3, not 4, 5, and 9.

Creating the To-File (CRTFILE Parameter)

To copy a physical or logical file when there is no existing to-file to receive the data, you can create the to-file by specifying CRTFILE(*YES). The name of the new to-file is specified on the TOFILE parameter and must be qualified with the name of an existing library for which you have the required authority.

If copy is used to create the to-file, then the to-file specified cannot be overridden to a different file or library.

CRTFILE(*YES) also adds members in the newly created file. See the section on adding members later in this chapter.

Because no records exist in the newly created file and member, the MBROPT parameter is ignored, and records are automatically added to the new file. An error occurring during copy processing after a file is created and/or members are added has no effect on the newly created file and/or added members.

The to-file that is created has the same record format and type of access path as the from-file. The text of the from-file is used as the text for the to-file. The text of from-file members that are copied is used as the text of any to-file members created. If the from-file is a logical file with multiple record formats, the to-file is created with the record format specified on the RCDFMT parameter on the CPYF command. (See "Selecting Records Using a Specified Record Format Name (RCDFMT Parameter)" on page 3-18.) When the from-file is a logical file, the following physical file attributes are assigned by the system: SIZE(*NOMAX), ALLOCATE(*NO), and CONTIG(*NO). If the from-file is a logical file and its keyed access path was created with the *NONE keyword specified for all key fields in the source DDS, then the physical to-file is created with an arrival sequence access path.

For copy to create a physical file for the to-file, you must have authority to the Create Physical File (CRTPF) command.

The created to-file has the same *public* authority as the from-file. No private authorizations are given to the created to-file except for the user issuing the CPYF command. The user who ran the command is the owner of the created to-file and has all private rights associated with the file. If the to-file is a DDM file, CRTFILE(*YES) is not supported.

The created to-file does not maintain the file capabilities of the from-file. The to-file allows update, delete, read, and write operations, regardless of whether these operations were allowed on the from-file.

If the from-file is an SQL table, the created to-file will be a physical file that is not an SQL table.

Selecting Members/Labels to Copy (FROMMBR/FROMLABEL/TOMBR/TOLABEL Parameter)

For database files, you can specify the name of the from-file member to be copied on the FROMMBR parameter and the name of the to-file member that is to receive the copied records on the TOMBR parameter. A special value *FIRST can also be specified to copy from or to the first member (in creation order) in the database file.

For diskette or tape device files, you can specify the label identifier of the data file to be copied on the FROMMBR or FROMLABEL parameter and the label identifier of the data file that is to receive the copied records on the TOMBR or TOLABEL parameter. If the special value *FIRST, *DKTF, or *TAPF is specified on the copy command, then the label from the device file description is used.

If the from-file is a database or diskette file, a generic name can be specified on the FROMMBR or FROMLABEL parameter. All members or labels that start with a character string you specified for the generic name are copied. To specify a generic name, enter the starting character string that each member/label has in common, and follow it with an * (asterisk). For example, if you specified FROMMBR(ORD*), all database members or diskette labels starting with ORD would be copied.

You can also indicate on the FROMMBR or FROMLABEL parameter that you want to copy all the members from a database file or all the labels from a diskette file by specifying special value *ALL.

You can copy a generic set or all members/labels in the following copy operations:

Diskette To:	**Database To:**
Database (physical file)	Database (physical file)
Diskette (Note 1)	Diskette (Note 1)
Tape (Note 2)	Tape (Note 2)
Printer	Printer
*PRINT	*PRINT

Notes:

1. The to-file must be spooled for diskette-to-diskette copy operations.
2. Multiple from-file members or labels can only be copied to a single tape file label.

If a generic set or all labels are being copied from a diskette and a label *being copied* is continued on another diskette volume, then all the labels on the continuation volume are copied (or checked if they should be copied where a generic label was specified).

Multiple database members or diskette labels can be copied to corresponding like-named to-file members or labels. They can also be copied and concatenated, one after another, into a single to-file member or label. If the to-file is a spooled printer file, then each member/label is copied to a separate spool output file. If TOFILE(*PRINT) is specified, then all the members/labels are copied to a single print output file, with the records for each member/label starting on a new page.

A single member/label or multiple members/labels can be copied to corresponding like-named to-file members/labels by specifying TOMBR(*FROMMBR), TOLABEL(*FROMMBR), or TOMBR(*FROMLABEL) depending on the copy command used. If the to-file is tape, this cannot be specified unless you are copying from a single from-file member/label.

If the from-file is diskette or tape, the from-file label is used as the label for a diskette or tape to-file. If the to-file is a database file, the nonblank characters to the extreme right of the from-file label are used for the to-file member name, up to a maximum of either 10 characters or to the period at the extreme right in the from-file label. The copy operation ensures that only a valid member name is used for a database to-file, but does not ensure that a to-file label is valid for tape or diskette, so an invalid or nonstandard label identifier may be used for the to-file.

If the from-file is a tape file that is not labeled, then a to-file member or label name is created that corresponds to the data file on the tape from-file in the form of CPYnnnnn, where nnnnn is the tape sequence number of the data file.

For a database from-file or to-file, if a MBR parameter is specified on an OVRDBF (Override Database File) command, then the override member name is used instead of the value specified on the copy command. If the TOFILE parameter is specified with no MBR parameter value on the OVRDBF command, then the first member (in creation order) in the database file is used instead of the member specified on the copy command. For a diskette or tape from-file or to-file, if a LABEL parameter is specified on an OVRDKTF or OVRTAPF command, respectively, the override label name is used instead of the label specified on the copy command.

If multiple members/labels are being copied to corresponding like-named to-file members/labels (that is, TOMBR(*FROMMBR), TOLABEL(*FROMMBR), or TOMBR(*FROMLABEL) was specified), then an override to a single to-file member/label is not allowed unless the from-file is also overridden to a single member/label.

If a tape or diskette label is specified in the FROMMBR or TOMBR parameter, it can have a maximum length of 10 characters. If the label contains special characters or more than 10 characters, it must be specified on one of the following commands:

- Create Tape File (CRTTAPF)
- Change Tape File (CHGTAPF)
- Override with Tape File (OVRTAPF)
- Create Diskette File (CRTDKTF)
- Change Diskette File (CHGDKTF)
- Override with Diskette File (OVRDKTF)

Adding Members to the To-File

The copy function adds a member to the to-file when the member does not exist. The member name used is either the TOMBR parameter value from the copy command, or the member name specified in an override for the to-file.

If TOMBR(*FROMMBR) or TOMBR(*FROMLABEL) is specified on the copy command (and is not overridden), the from-file member names or label identifiers are used for the members added to the file.

If TOMBR(*FIRST) is specified on the copy command, or if there is an override that specifies a TOFILE parameter with no MBR parameter, then no member name is known. The copy function does not add a member in this case unless CRTFILE(*YES) was specified on the CPYF command and the copy function must create the to-file. When the copy function creates the to-file without a specific member name specified, the from-file name is used for the member that is added to the to-file.

If the from-file is a database file, the member text and SEU source type of the from-file member are used for the member added to the to-file. If the from-file is a device file, the text is taken from the first-level text of message CPX0411, and the SEU source type is TXT.

Selecting Records to Copy

You can select records to be copied based on the following (parameters given in parentheses):

- Record format name when a multiformat logical file is copied (RCDFMT)
- Relative record numbers (FROMRCD and TORCD)
- Record key values (FROMKEY and TOKEY)
- Number of records (NBRRCDS)
- One or more character values starting in a specified position in the record or a field (INCCHAR)
- Field value in a record (INCREL)
- Deleted records (COMPRESS)

For a detailed description of all considerations for each parameter, see the *CL Reference*.

Selecting Records Using a Specified Record Format Name (RCDFMT Parameter)

When you copy from a logical file to a physical file and the logical file has more than one record format, you must specify a record format name unless you specify FMTOPT(*NOCHK). If FMTOPT(*NOCHK) is used, then RCDFMT(*ALL) can be specified to copy all from-file record formats to the to-file. This record format name is used to select records to be copied.

In the following example, records are copied from the logical file ORDFILL to the physical file INVOICE using the record format ORDHDR:

```
CPYF   FROMFILE(DSTPRODLB/ORDFILL)  TOFILE(DSTPRODLB/INVOICE)  +
  RCDFMT(ORDHDR) MBROPT(*ADD)
```

See "Copying between Different Database Record Formats (FMTOPT Parameter)" on page 3-24 for information about what happens when the from-file and to-file are both database files and their record formats are different.

When you copy from a logical file that has more than one record format to a device file, you can specify either a single record format to be used or specify RCDFMT(*ALL) to copy using all the record formats. If the record formats have different lengths, the shorter records are padded with blanks.

Selecting Records Using Relative Record Numbers (FROMRCD/TORCD Parameters)

Relative record numbers can be specified for a copy from any file type except a keyed logical file (a keyed physical file can be copied in arrival order if relative record numbers are specified for the FROMRCD or TORCD parameter). Records can be copied from a specified record number (FROMRCD parameter) to a specified record number (TORCD parameter) or until a specified number of records (NBRRCDS parameter) has been copied (see "Selecting a Specified Number of Records (NBRRCDS Parameter)" on page 3-20). If the end of the file is reached before the specified ending record number or number of records is reached, the copy completes normally.

When a relative record number is specified, records are copied, starting with the specified relative record number, in the order in which they physically exist in the database file being copied from. This is true even if the physical file has a keyed sequence access path. You can use the COMPRESS parameter with the FROMRCD and TORCD parameters to further define which records are to be selected for copying (see "Copying Deleted Records (COMPRESS Parameter)" on page 3-22).

Deleted records retain their position among records that are not deleted, but not necessarily their relative record number when they are copied if they are in the specified subset and COMPRESS(*NO) is specified. If COMPRESS(*YES) is specified, the deleted records are skipped and are not copied. In this case, when the record number specified (FROMRCD parameter) is a deleted record, copying starts with the first nondeleted record that follows.

In the following example, the records from relative record number 500 to relative record number 1000 in the file EMP1 are copied to the file EMP1T.

```
CPYF   FROMFILE(PERSONNEL/EMP1)  TOFILE(TESTLIB1/EMP1T)  +
  MBROPT(*REPLACE) FROMRCD(500)  TORCD(1000)
```

Note: If you use record numbers to select records, you cannot use record keys (FROMKEY/TOKEY parameters) to select records on the same CPYF command.

Selecting Records Using Record Keys (FROMKEY/TOKEY Parameters)

Record keys can only be specified to copy from a keyed data base file. Records can be copied from a specified key value (FROMKEY parameter) to a specified key value (TOKEY parameter) or until a specified number of records (NBRRCDS parameter) is reached (see "Selecting a Specified Number of Records (NBRRCDS Parameter)" on page 3-20). If the end of the file is reached before the specified ending key value or number of records is reached, the copy completes normally.

You can specify *BLDKEY on the FROMKEY and TOKEY parameters to use a list of character and numeric values in their natural display form for the fields in a key. Each element is converted to the corresponding key field data type and the composite key value provided to the database.

If you specify fewer values than the complete database key contains, a partial key is built and passed to the database. If you specify more values than the database key contains, an ending error occurs. The values are always applied to the consecutive fields to the extreme left in the key so that it is impossible to skip key fields.

Character fields are padded on the right with blanks. Numeric fields are adjusted to the implied decimal point in the key field with the correct zero padding.

All regular rules for specifying numeric fields in an external character format apply. Note that the floating-point value of *NAN (Not a Number) is not allowed.

Using *BLDKEY is the easiest way to specify (and ensure correct padding) values for packed, binary, and floating-point fields.

An example of the build-key function is:

Key Field Number	**Type**	**Length**	**Decimal Precision**	**Value**
1	CHAR	6		KEN
2	ZONED	6	2	54.25
3	BINARY	4	1	10.1

You could specify the FROMKEY (or TOKEY) parameter as follows:

```
FROMKEY( 2  x'D2C5D5404040F0F0F5F4F2F50065')
```

or, you could use the *BLDKEY value and specify the FROMKEY as follows:

```
FROMKEY(*BLDKEY    (KEN 54.25 10.1))
```

Another example using key fields 1 and 2 is:

```
FROMKEY(2  'KEN    005425')
```

or, the *BLDKEY value can be specified:

```
FROMKEY(*BLDKEY    (KEN 54.25))
```

In the following example, the records in the file EMP1 are copied to the file EMP1T. EMP1T is a file in a test library. Because you only need a subset of the records, you specify a from-key value and a to-key value. Both are full key values. Note that a 1 specified in the FROMKEY and TOKEY parameters indicates the number of key fields to be used in searching the record keys, starting with the first key field.

```
CPYF   FROMFILE(PERSONNEL/EMP1)  TOFILE(TESTLIB1/EMP1T)  +
  MBROPT(*REPLACE)  FROMKEY(1 438872)  TOKEY(1 810199)
```

All positions in a key value should be specified, because if the value is shorter than the key field length, it will be padded on the right with zeros. Thus, a 5-position key field specified as FROMKEY(1 8) causes a search for a key equal to hex F800000000. If the key value contains blanks or special characters, it must be enclosed in apostrophes.

Note: If you use record keys to select records, you cannot use relative record numbers (FROMRCD/TORCD parameters) to select records on the same CPYF command.

You should not specify COMPRESS(*NO) when selecting records by record key from a keyed physical file. Because deleted records are not contained in the keyed access path of a file, they are never copied, so the compression is automatic.

Because deleted records are canceled in a copy by this method, it is also possible that the relative record numbers have changed in the new file, even if you have specified MBROPT(*REPLACE).

Selecting a Specified Number of Records (NBRRCDS Parameter)

When you specify a FROMKEY or FROMRCD parameter, you can specify the number of records (NBRRCDS parameter) to be copied instead of the TOKEY or TORCD parameter (you cannot specify both the NBRRCDS and the TORCD or TOKEY parameters). The specified number of records is copied starting with the specified from-key value or from-record number.

You can specify the NBRRCDS parameter without specifying the FROMKEY or FROMRCD parameter. Records are copied starting with the first record in the file. Note that the number of records specified is the number of records actually copied to the to-file, which includes deleted records in the from-file, if COMPRESS(*NO) is specified and does not include records excluded by the INCCHAR and/or INCREL parameters.

In the following example, 1000 records in the file EMP1 are copied to the file EMP1T. Records are copied from the first member in EMP1 and replace the records in the first member in EMP1T.

```
CPYF   FROMFILE(PERSONNEL/EMP1)  TOFILE(TESTLIB1/EMP1T)  +
  MBROPT(*REPLACE)  NBRRCDS(1000)
```

You can also use the NBRRCDS parameter to examine a subset of records on a list:

```
CPYF   FROMFILE(PERSONNEL/EMP1)  TOFILE(*PRINT)  FROMRCD(250)  +
  NBRRCDS(10) OUTFMT(*HEX)
```

Selecting Records Based on Character Content (INCCHAR Parameter)

Records can be selected based on the content of specified characters starting in a specified position in the record or field. The INCCHAR parameter can be used with the FROMKEY or FROMRCD parameter, to select records first by their key value or relative record number and then by characters in some position in the record or field.

You can test for any character string of 1 through 256 bytes. If the character string contains any special characters or blanks, the whole string must be enclosed in apostrophes.

You can also specify *CT (contains) as the operator for the INCCHAR parameter. This specifies that each record in the from-file is to be scanned for the selection character string. You can specify any valid starting position in the field or record for the start of the scan, and the data will be scanned from that position to the byte to the extreme right of the field or record.

If you specify both the INCCHAR and INCREL parameters, a record is copied only if it satisfies both the INCCHAR and INCREL conditions.

The following example tests for all records in the file DBIN that have an XXX starting in position 80, and copies them to the file DKTOUT. Because this example includes testing for positions relative to the length of the whole record, *RCD must be specified on the INCCHAR parameter.

```
CPYF FROMFILE(DBIN) TOFILE(DKTOUT) INCCHAR(*RCD 80 *EQ XXX)
```

If you were testing for an XXX in a position in a particular field in the record, you would specify the field name instead of *RCD, and the starting position of the character relative to the start of the field.

```
CPYF FROMFILE(DBIN) TOFILE(DKTOUT) INCCHAR(FLDA 6 *EQ XXX)
```

A field name cannot be specified if RCDFMT(*ALL) is specified when copying from a multiple-format logical file, or if the from-file is a device file or spooled inline file.

Selecting Records Based on Field Value (INCREL Parameter)

The INCREL parameter is used to select records for copying by testing for the value of an entire field. Unlike the INCCHAR parameter, you can use the INCREL parameter only when copying from a database file, and you can test for different values in different fields on one CPYF command.

You can use as many as 50 AND and OR relationships on one INCREL parameter. The OR relationship is used to group the AND relationships. For example, the following INCREL parameter essentially says this: If field FLDA is greater than 5 and field FLDB is less than 6, select the record, or if FLDB is equal to 9 (FLDA is any value), select the record.

```
INCREL((*IF FLDA *GT 5)  (*AND FLDB *LT 6)  (*OR FLDB *EQ 9))
```

The value you specify must be compatible with the field type.

If you specify both the INCCHAR and INCREL parameters, a record is copied only if it satisfies both the INCCHAR and INCREL conditions.

The INCREL parameter cannot be specified if RCDFMT(*ALL) is specified when copying from a multiple-format logical file.

Copying Deleted Records (COMPRESS Parameter)

You can copy deleted and undeleted records from one physical file member to another by specifying COMPRESS(*NO) on the CPYF command. You may want to copy deleted records to preserve the relative record numbers of records copied from the from-file. If COMPRESS(*NO) is not used, only records that are not deleted are copied from the from-file.

To use COMPRESS(*NO), the from-file and to-file must both be physical files, they must both be the same type (either source or data), and they must either have identical record formats or FMTOPT(*NOCHK) must be specified to perform the copy. COMPRESS(*NO) also requires that you use all the following (default) parameter values on the CPYF command:

```
PRINT(*NONE)
INCCHAR(*NONE)
INCREL(*NONE)
SRCOPT(*SAME)
ERRLVL(0)
```

COMPRESS(*NO) is not allowed for certain types of access paths over the to-file, including when the access path is contained in a logical file and is based on the to-file member. The following types of access paths over a to-file member do not allow COMPRESS(*NO) to be specified:

- Unique keys (UNIQUE keyword specified in the DDS for the file).
- Select/omit specifications without the DYNSLT keyword (in the DDS for the file), and immediate or delayed maintenance (MAINT(*IMMED) or MAINT(*DLY) specified on the CRTPF or CRTLF command).
- Floating-point key field or logical numeric key field (in the DDS for the file), and immediate or delayed maintenance (MAINT(*IMMED) or MAINT(*DLY) specified on the CRTPF or CRTLF command). Note that a logical numeric key field is one of the following:
  - A numeric key field in a logical file.
  - A field specified as a *to* field on the JFLD keyword that has different attributes than in the based-on physical file.
  - A field specified as a sequencing field on the JDUPSEQ keyword that has different attributes than in the based-on physical file.

You cannot specify COMPRESS(*NO) for any of the following cases:

- If the to-file is being journaled (JRNPF command).
- If the to-file member is in use, or if any access path over the to-file member is in use.
- If an EOFDLY wait time is specified for the from-file on an OVRDBF command.

COMPRESS(*NO) may allow the system to perform the copy more quickly because records are transferred in blocks, but this is *not always* true. Usually, the COMPRESS(*NO) function does not significantly affect performance. One of the factors you should consider before you specify COMPRESS(*NO) is that the internal system function that must be used to perform this type of copy invalidates any keyed access paths that use the to-file member before the records are copied, and then rebuilds the access paths after the copy is complete. The run time and resource required to rebuild the keyed access paths may be larger than the performance benefit gained by copying deleted records.

If COMPRESS(*NO) is *not* specified, the system may still use the internal functions to perform the copy, but the choice of how the copy is performed is based on the number of records in the from-file and to-file members before the copy, and the number of keyed access paths over the to-file member.

If MBROPT(*REPLACE) is specified, all keyed access paths over the to-file member must be invalidated and rebuilt, so specifying COMPRESS(*NO) does not cause any additional overhead for rebuilding access paths.

If the from-file is a keyed physical file and neither a FROMRCD nor TORCD relative record number value is specified on the CPYF command to force the file to be processed in arrival sequence, COMPRESS(*NO) has no meaning because a keyed access path never contains any deleted records.

Printing Copied and Excluded Records (PRINT/OUTFMT Parameters)

You can print a list of all records copied or all records excluded or both. Records can be printed in character format or in character and hexadecimal format. The records are printed using the IBM-supplied printer file QSYSPRT.

You can specify *EXCLD on the PRINT parameter to print only excluded records or *COPIED to print only copied records. Records are copied or excluded based on the specifications on the INCCHAR or INCREL parameter. If both are specified all the excluded records are in one list, and all the copied records are in another. If multiple members are being copied, a list is produced for each member/label. Excluded records are printed in the from-file record format, and copied records are printed in the to-file record format.

The OUTFMT parameter defaults to *CHAR; records are printed in character format. If you specify *HEX, records are printed in both character and hexadecimal format.

If you specify TOFILE(*PRINT), the OUTFMT parameter again specifies the format that is used to print the records (see "Basic Copy Function" on page 3-6).

In the following example, all records that are not copied are printed:

```
CPYF    FROMFILE(DKTIN)  TOFILE(LIB1/PF)  MBROPT(*ADD)  +
  INCCHAR(*RCD 80 *EQ X) PRINT(*EXCLD)
```

The records are printed in character format.

Copying between Different Database Record Formats (FMTOPT Parameter)

When you copy from a database file to a database file, you must use the FMTOPT parameter if the record formats are not identical or if the files are different types (source or data). If either file is a device file or spooled inline file, the FMTOPT parameter does not apply. The records are truncated or padded with blanks or zeros when record lengths are different. A message is sent if the records are truncated.

If the from-file or to-file is a device file, or a spooled inline file, copy automatically adds or deletes the source sequence number and date fields for each record copied.

If one file is a data file and the other a source file, you must specify FMTOPT(*CVTSRC) to perform the copy. The sequence number and date fields are added or deleted as appropriate and the data part of each record is copied without regard to the other field definitions in the file record formats. The SRCSEQ parameter can be used to control how the sequence numbers are created, provided SRCOPT(*SEQNBR) is also specified.

For database-to-database copies, you can reconcile any differences in record formats by specifying:

- *DROP to drop those fields in the from-file record format for which there are no fields of the same name in the to-file record format.
- *MAP to convert fields in the from-file to the attributes of like-named fields in the to-file and to fill extra fields in the to-file, that are not in the from-file, with their default values. The default values are:
  - The parameter value for the DFT keyword, if specified for the field
  - Blanks (for character fields without the DFT keyword)
  - Zeros (for numeric fields without the DFT keyword)

*MAP is required if fields with the same name are in different positions in the file record formats, even though these fields have the same attributes.

- *DROP and *MAP to drop fields in the from-file not named in the to-file and to convert remaining fields through mapping rules to fit the to-file fields that have different attributes or positions.
- *NOCHK to disregard the differences. Data is copied left to right directly from one file to the other. The copied records are either truncated or padded with blanks or zeros. Because no checking is done, fields in the to-file may contain data that is invalid for the field as defined.

Dropping and mapping fields are based on a comparison of field names. Unless all the fields in the from-file have the same name in the to-file, you must specify *DROP. If the names are the same, but the attributes or position in the record is different, you must specify *MAP. Dropped fields are not copied. There must be at least one like-named field in both record formats to do mapping.

When *MAP is specified, fields in the to-file record format that do not exist in the from-file record format are filled with their default values, as described earlier in this section. For fields that have the same name and attributes, the field in the from-file record format is mapped to the field with the same name in the to-file record format, even if their positions in the formats are different. For example, the field

CUSNO is the first field in the record format ORDHD, but it is the second field in record format ORDHD1. When the CUSNO field is copied with *MAP, it is mapped to the second field of ORDHD1.

The following table summarizes the database-to-database copy operations for each value on the FMTOPT parameter:

Table 3-4. Database-to-Database Copy Operations

FMTOPT Parameter Values	**Database File Record Formats**				
	ALL **Field Names in From-and To-Files Are the Same (like-named)**		**SOME** **Field Names in From-and To-Files Are the Same**		**NO** **Field Names in Either File Are the Same**
	Attributes and relative order also the same (see note 1)	Attributes and relative order not the same (see note 1)	Like-named fields have identical attributes and relative order (see note 1)	Not all like-named fields have identical attributes and relative order (see note 1)	
*NONE	Complete copy	Command ends	Command ends	Command ends	Command ends
*DROP	Complete copy (value ignored)	Command ends	If there are extra fields in the from-file, they are dropped, all others are copied. If there are extra fields in the to-file, the command ends. If there are extra fields in the from-file and in the to-file, the command ends.	Command ends	Command ends
*MAP (see note 2)	Complete copy (value ignored)	Complete copy (corresponding fields are mapped)	If there are extra fields in the from-file, the command ends. If there are extra fields in the to-file, they are filled, and the like-named fields are mapped. If there are extra fields in the to-file and the from-file, the command ends.		Command ends
*MAP and *DROP (see note 2)	Complete copy (value ignored)	Complete copy (corresponding fields are mapped)	Extra fields in the from-file are dropped; like-named fields are mapped; extra fields in the to-file are filled.		Command ends
*NOCHK	Complete copy (value ignored)	Complete copy (direct data transfer disregarding fields) (see note 3)			

Notes:

1. Field attributes include the data type (character, zoned, packed, binary or floating point), field length, and the decimal position (for numeric fields).
2. Mapping consists of converting the data in a from-file field to the attributes of the corresponding (like-named) to-file field. If the attributes of any corresponding fields are such that the data cannot be converted, the copy is ended.
3. The records are padded or truncated as necessary. Data in the from-file may not match the to-file record format.

Conversion Rules

Table 3-5 shows the field conversions that are allowed between mapped fields in the from-file and to-file record formats. If fields with the same name have incompatible attributes between the from-file and to-file formats, only FMTOPT(*NOCHK) can be used to perform the copy. An X indicates that the conversion is valid, and a blank indicates an invalid field mapping:

Table 3-5. Field Conversions

From Field	To Character or Hexadecimal Field	To Packed Decimal Field	To Zoned Decimal Field	To Binary (No Decimals Positions) Field	To Floating Point Field	To Binary Field (with Decimals Positions)
Character or Hexadecimal	X					
Packed		X	X	X	X	
Zoned		X	X	X	X	
Binary (no decimal positions)		X	X	X	X	
Floating Point		X	X	X	X	
Binary (with decimal positions)						X (see note)

Note: A binary numeric field with one or more decimal positions can only be copied to a binary field with the same number of decimal positions.

When mapping character fields, the field being copied is truncated on the right if it is longer than the field into which the copy is made. For example, a character field of length 10 is copied into a character field of length 6; ABCDEFGHIJ becomes ABCDEF. If the field being copied is shorter than the field into which it is copied, the field is padded on the right with blanks. For example, a character field of length 10 is copied into a character field of length 12; ABCDEFGHIJ becomes ABCDEFGHIJƀƀ (ƀ = blank).

When mapping numeric fields and the field being copied is longer than the field into which the copy is made, the field being copied is truncated on the left and right of the decimal point. For example, a zoned decimal field of length 9 with 4 decimal positions is copied to a zoned decimal field of length 6 with 3 decimal positions; 00115.1109 becomes 115.110.

If significant digits must be truncated to the left of the decimal point, the value is not copied, and the field is set to its default value (either the parameter value of the DFT keyword, if specified, or zero, if the DFT keyword is not specified). Also, if significance will be lost because a floating-point numeric value exponent is too large, the to-file field is set to its default value.

When mapping numeric fields and the field being copied is shorter than the field into which the copy is made, the field being copied is padded with zeros on the left and right of the decimal point. For example, a packed decimal field of length 7 with 5 decimal positions is copied to a packed decimal field of length 10 with 6 decimal positions; 99.99998 becomes 0099.999980.

Adding or Changing Source File Sequence Number and Date Fields (SRCOPT/SRCSEQ Parameters)

Copying Device Source Files to Database Source Files

When you copy from a device source file to a database source file, the system adds sequence number and date fields at the start of the records. The first record is assigned a sequence number of 1.00, the next 2.00, and so on, increasing in increments of 1.00. If more than 9999 records are copied, the sequence number is wrapped back to 1.00 and continues to be incremented unless the SRCOPT and SRCSEQ parameters are specified on the copy command. Date fields are initialized to zeroes. If several copies to the same file are made with MBROPT(*ADD) specified, you will have duplicate sequence numbers in the file. This can be corrected using the Reorganize Physical File Member (RGZPFM) command.

When copying to or from a device, it is more efficient to use a device data file than a device source file. The copy function automatically adds or removes the source sequence number and date fields as necessary.

Copying Database Source Files to Device Source Files

When you are copying to a device source file, the system removes the date and sequence number fields from the start of the records.

When copying to or from a device, it is more efficient to use a device data file than a device source file. The copy function automatically adds or removes the source sequence number and date fields as necessary.

Copying Database Source Files to Database Source Files

You can copy between database source files by using the CPYSRCF or CPYF command. The CPYSRCF command may be easier to use because the parameter defaults are particularly suited for copying database source files.

Note: When TOFILE(*PRINT) is specified, the CPYSRCF command uses a format to show source information that is easy to read. Also, for multiple member copies, the members are listed (and copied) in alphabetical order.

When you copy from a database source file to a database source file the sequence number and date fields are not changed unless the SRCOPT parameter is specified. If SRCOPT(*SEQNBR) is specified, the sequence numbers are updated. If SRCOPT(*DATE) is specified, the date field is initialized to 0. Both *SEQNBR and *DATE can be specified.

If SRCOPT(*SEQNBR) is specified in order to update the sequence numbers, the SRCSEQ parameter is taken into consideration. The SRCSEQ parameter specifies the starting value assigned to the first record copied and the increment value. The defaults are 1.00 and 1.00. A whole number of no more than 4 digits and/or a fraction of no more than 2 digits can be specified for the starting value and the increment value. You must use a decimal point for fractions. If you specify SRCSEQ(100.50), then the records copied will have sequence numbers 100.00, 100.50, 101.00, 101.50, and so on.

If the file to be copied contains more than 9999 records, a fractional increment value should be used so that each record has a unique sequence number. If a starting value of .01 and an increment value of .01 are specified, the maximum number of records copied with unique sequence numbers is 999 999. When the maximum sequence number is exceeded (9999.99), all remaining records on that copy are initialized to 9999.99. The system does not wrap back to 1.00.

If the database source file being copied to has only an arrival sequence access path, then because the file does not have a keyed sequence access path (arranged by sequence number), records cannot be inserted in the middle of the file keyed access path. The records are always physically placed at the end of the file.

Recoverable Error Considerations (ERRLVL Parameter)

When you are copying to or from a database file or from a tape device file, you can limit the number of recoverable errors that you accept before the copy is ended. Use the ERRLVL parameter to specify this limit. The types of errors this parameter applies to are:

CPF5026 Duplicate key in the access path of this member

CPF5029 Data or key conversion error

CPF5030 Partial damage on member

CPF5034 Duplicate key in the access path of another member

CPF5036 Invalid length tape block read

CPF5097 *NAN (Not a Number) value not allowed in floating-point key field

The ERRLVL parameter specifies the maximum number of recoverable errors allowed for each member/label pair copied. Note that the value specified for ERRLVL indicates the total errors allowed on both the from-file and the to-file combined for each member/label pair copied. Each time an error occurs, the count for that member/label pair is increased by 1, a message identifying the last good record read or written is printed on all copy lists if TOFILE(*PRINT), PRINT(*COPIED), or PRINT(*EXCLD) was specified, and copying continues. If the from-file is completely copied without exceeding the limit, the counter is reset to 0, and the copy of the next member is started. If the limit is exceeded during the copy of a member, copying ends, even if there are more records or additional members to be copied.

Partial object damage to the contents of a database file is an error that occurs when a portion of the file cannot be read from disk. If a file is damaged in such a way, you can bypass records in error by copying the good records and manually adding the records that were not copied because of the damage.

Position Error Considerations

A position error occurs when the copy file function cannot locate the first record to be copied in the from-file member. This can happen when using the CPYF, CPYSRCF, CPYTODKT, or CPYTOTAP commands. If any of the following are true, you may receive a position error for the from-file member:

- The FROMKEY parameter is specified and all records in the member are less than the FROMKEY value or the member is empty.
- The FROMRCD parameter is specified beyond the end of the member or the member is empty.
- The value of the from-file member position (the POSITION parameter of the OVRDBF command) is beyond the end of the member, is not valid for the access path of the from-file, or the member is empty.

If a member position error occurs, the member may not be added to the to-file, and no information about the member will be added to the print lists.

If a member position error occurs during a copy operation involving multiple members, the copy operation will continue with the next member.

If a member position error occurs for all members being copied, a print list will not be produced and the to-file may not be created.

Allocation Considerations

When a database file is copied, each from-file member is allocated with a shared-for-read (*SHRRD) lock state. The member is only allocated while it is being copied. When a device file is copied, it is allocated with a shared-for-read (*SHRRD) lock state. Generally, the member being copied to is allocated with a shared-for-update (*SHRUPD) lock state. However, if MBROPT(*REPLACE) is specified, the member being copied to is allocated with an exclusive (*EXCL) lock state, and the records in the file being copied to are removed. A shared-for-read lock state lets other users read and update the file while it is being copied.

When you are copying one physical file to another, stronger locks *may* be placed on the members to allow internal system functions to perform the copy.

- The from-file member can be allocated with an exclusive-allow-read (*EXCLDRD) lock state.
- The to-file member can be allocated with an exclusive (*EXCL) lock state.

These stronger locks are required when COMPRESS(*NO) is specified. If you cannot get them, the copy operation may be performed with COMPRESS(*YES) specified instead.

If a member is allocated by another job with too strong a lock state, or if the library containing the file is renamed during the copy operation, the copy operation may end with an error message.

If you anticipate that problems may arise because of this, you can preallocate the files and members using the Allocate Object (ALCOBJ) command. (See the *CL Programmer's Guide* for information about preallocating objects.)

Authority

The following table summarizes the authority required for the from-file and the to-file to perform a copy operation:

Table 3-6. Authority Required to Perform Copy Operation

	From-File	To-File
DDM file	*OBJOPR *READ	*OBJOPR[1] *ADD
Device file[2]	*OBJOPR *READ	*OBJOPR *READ
Logical file	*OBJOPR[3]	Not allowed
Physical file	*OBJOPR *READ	*OBJOPR[1] *ADD

[1] This is the authority required for MBROPT(*ADD). If MBROPT(*REPLACE) is specified, *OBJMGT and *DLT authority are also required.

[2] *OBJOPR authority is also required for any devices used for the file.

[3] Also requires *READ authority to the based-on physical file members for the logical file members copied.

If the to-file does not exist and CRTFILE(*YES) is specified so that CPYF will create the to-file, then you must have operational authority to the CRTPF command.

Performance

The performance of the copy operation depends on the types of files copied, the file attributes, and the optional copy parameters specified.

A copy that requires maintenance of a keyed sequence access path is slower than a copy from or to an arrival sequence access path. Copy performance will be improved if the from-file is reorganized so that its arrival sequence is the same as its keyed sequence access path, or if records are selected using the FROMRCD or TORCD parameter, so that the keyed sequence access path is not used.

The fewer logical access paths there are over the to-file, the faster the copy will be, because those access paths have to be updated as the records are copied.

The smaller the record lengths of the file, the faster the copy.

In general, the fewer optional copy parameters specified, the faster the copy. The following parameters affect the performance of the copy operation:

INCCHAR
INCREL
ERRLVL
FMTOPT
SRCOPT
PRINT

Using the COMPRESS function does not significantly affect performance. You should request COMPRESS(*NO) if you want deleted records in the to-file, for example, when the relative record numbers need to be identical.

Part 3. Device File Support

The chapters in this part describe data management support for the following devices:

- Display
- Printer
- Tape
- Diskette

For more detailed information about ICF files, refer to the *Communications Programmer's Guide*.

In addition, since spool is an extension to device file support, a detailed description of the spool support provided by the system is included.

Should you need an introduction to file support before reading the chapters that follow, refer to Chapter 1, "File Processing."

Chapter 4. Display Device Support

Display device support is designed to simplify the use of display devices by application programs. Display device files typically need more programming than printer device files or database files. For example, in addition to specifying the usual information such as field name, length, data type, and decimal positions for a field on a display device, you must specify both:

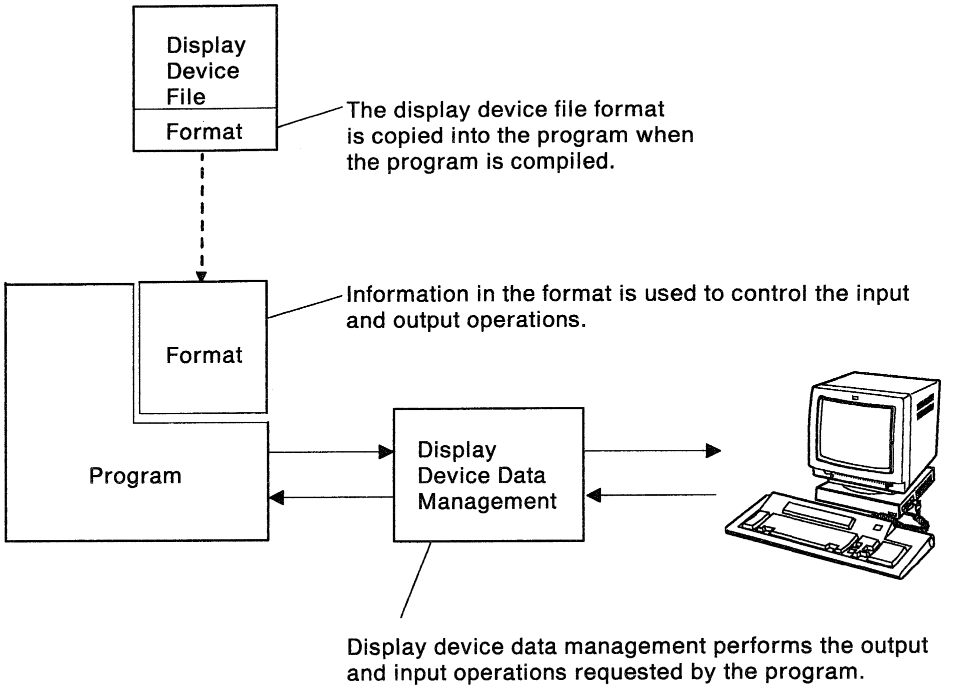
- The position of the field on the display
- The type of field (input, output, or both)

You can also specify many other things in a display file, such as:

- Additional field attributes, such as highlight, underline, blink, reverse image, protected, and color
- Output editing (such as decimal point and comma)
- Input validity checking (such as numeric only, mandatory entry, mandatory fill, self-check digit, range check, and list of values)
- Command function key, and other special key, specifications
- Displaying error messages
- Output and input of multiple repeated lines of data
- Print key support, which lets you write the contents of the display to a printer file or to a work station printer
- Help key support, which lets you define online help information for individual areas of the display to be presented when the Help key is pressed
- Many other functions, such as duplicating fields, overlaying information on the display, clearing portions of the display, positioning the cursor, and locking and unlocking the keyboard

All of these functions are controlled through display device support in a way that reduces the amount of programming required to write work station applications. Most of this information is specified in external data description specifications (DDS), which are stored in a display file. For example, an input field may have validity checking that specifies the field is mandatory entry, with a value from 1 through 5. If no data is entered or if anything other than a number from 1 through 5 is entered, device data management locks the keyboard, displays the field with reverse image, positions the cursor to this field, and displays an error message. All this is accomplished without any user application programming.

The following shows how display device support works:


Display
Device
File
Format
The display device file format is copied into the program when the program is compiled.
Format
Information in the format is used to control the input and output operations.
Program
Display
Device Data
Management
Display device data management performs the output and input operations requested by the program.

RSLH163-1

Objects that Support Displays

There are basically two objects that support displays:

- Display device description which is a system object that describes the display device itself. A device description contains information such as device address, device type, model number, and features. This may be created by system personnel or, for locally attached devices, can be created during the automatic configuration of the system.
- Display device file which is a device file used to describe how data is to be written to and read from a display device. This includes the formatting and other characteristics related to the data sent to and from the display.

To use a display that is attached to the system, both of these objects must exist. First, the device description must exist to describe the device to the system. Second, a display device file must exist to describe the data to be displayed on that device. Any number of display device files can be associated with a device, but only one device description can exist for a device. For remote work stations, corresponding controller descriptions must exist in addition to the device descriptions. Refer to the *Device Configuration Guide* for more information about controller and device descriptions.

Display Device Descriptions

Device descriptions are used to describe the characteristics of the following display devices to the system:

- 5250 display family: Locally or remotely attached
- ASCII displays: Attached to a local or remote work station controller through a link protocol converter
- 3270 display family: Remotely attached through a 3174/3274 controller
- IBM personal computers: Attached on a token-ring network or locally or remotely attached emulating a 5250 family display

Control Language Commands for Display Device Descriptions

The following commands are used for defining display device descriptions:

CHGDEVDSP Change Device Description Display: Changes certain parameters on a display device description such as port number and address.

CRTDEVDSP Create Device Description Display: Creates a device description for a display. You can give the device a name, specify its descriptive text, tell how it is attached and where, and give its model and type. The system uses this information to find the display in order to present information on it properly.

DLTDEVD Delete Device Description: Removes the definition of the device from the system.

DSPDEVD Display Device Description: Shows the current definition of the device.

WRKCFGSTS Work with Configuration Status: Shows the current status of selected devices, controllers, and lines on your system.

Display Device Files

Device files describe how the system is to operate on data as it passes between your program and a device. Display device files obviously perform this function for display devices. Throughout this manual the specific device files will be called by the device type, so in this section they are simply called *display files*.

Defining Display Files

Display files can be defined in one of two ways:

- Using the screen design aid (SDA) utility
- Using the Create Display File (CRTDSPF) command

With SDA you design the way the screens for your application are going to look right on the display. You can gradually define your screens, adding the attributes you need and testing as you go along. SDA then generates the data description specifications (DDS) that describe the display file and creates the display file for you. SDA will create the DDS for you; you do not need extensive knowledge of DDS coding forms, keywords, or syntax.

If you choose to use the CRTDSPF command, you must code your own DDS. The command compiles the DDS and creates the display file object.

All the examples in this chapter are shown in terms of their DDS because that provides the most concise description of the functions that are presented. However, keep in mind that all of these functions are available to you if you choose to use SDA to define your display files. To focus on the content of DDS input, headings for DDS forms within the chapters of this manual are purposely omitted. For your reference, Appendix G, "DDS Coding Form," shows a sample DDS coding form with the section headings.

Control Language Commands for Display Files

The following CL commands are used to work with display files:

CHGDSPF Change Display File: Changes one or more of the attributes of the display file.

CRTDSPF Create Display File: Creates a display file containing a file description and record formats.

DLTF Delete File: Deletes files.

DSPFD Display File Description: Displays the current characteristics of a file.

DSPFFD Display File Field Description: Displays the structure of a file that was created using DDS.

OVRDSPF Override with Display File: Temporarily changes the display file name used by the program and can also change certain characteristics of the file.

Concepts of Display Files

A display file is used to define the format of the information that is to be presented on a display, and how that information is to be processed by the system on its way to and from the device. The format of the data used in display files can be program-described or externally described. Both types of files can contain file and record format level information that is obtained from a create or change file command. The difference between program-described and externally described files is how they are defined. Externally described display files are defined using data description specifications (DDS). When a display file is externally described, it can be made up of multiple record formats and each record format can contain multiple field definitions. For display files using program-described data, the fields are described in the program that uses the file, and the file is created without using DDS. Display files defined using program-described data are restricted in their function. (See "Using Display Files with Program-Described Data" on page 4-142.) Display files, record formats and fields are described in detail in the following sections.

Files

Display files are used to format information on the display. They describe input and output fields, constants, the use of command function and command attention keys, and the handling of errors.

File attributes that you can specify on the Create Display File (CRTDSPF) command are:

- MAXDEV: Specifies the maximum number of displays that can be connected to the device file when the file is open.

  Note: A display file created with MAXDEV greater than one is called a multiple-device display file. Multiple device display files support some special functions that are described under "Using Multiple Devices" on page 4-114.

- DEV: Specifies the device that can be connected to the display file when it is open. Specifying *REQUESTER for this parameter allows the program to use the device from which the program was called.

- LVLCHK: Specifies if you want the level identifier of the record format checked to determine if the program is using the correct version of the record format at run time.

- RSTDSP: Specifies if you want the data being displayed to be saved if the file is suspended. This can be an important consideration if you display several records simultaneously using the DDS keywords OVERLAY, PUTOVR, or CLRL.

- DFRWRT: Delays the writing of data to the display until a read request is made.

- WAITFILE: Specifies the amount of time to wait for file resources (such as a device description) to become available for use with the file. This value is used when display files are opened and is also used by the acquire device function. These functions will be described in "Display File Operations" on page 4-20.

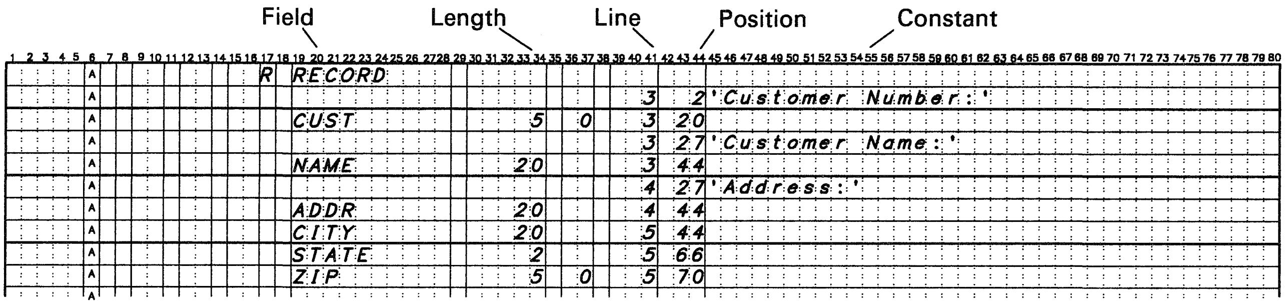
- WAITRCD: Specifies the amount of time to wait for input to become available when performing a RCVF WAIT(*NO) in a CL program or when performing a read-from-invited-devices operation in a high-level language program. For more information, see the discussions in "*NOWAIT on RCVF and SNDRCVF Commands" on page 4-30 and "Read-from-Invited-Devices Operation" on page 4-27.

Record Formats

A record format in a display file is used to describe both the format of the record as it is used in the application program and the format of the record when it is displayed (see Figure 4-1).

A record format contains field descriptions. Generally, for each field, you describe the location of the field on the display, the length of the field, the type of data contained in the field (character, zoned decimal, or floating point), and the field type (output, input, or output/input). The fields in the record passed to the program are arranged in the order in which you describe the fields in DDS. The order in which the fields are displayed is based on the display positions you assign to the field in the DDS. The field and indicator locations in the input records and output records used by the program are shown in the printed output produced when you create a display file using the CRTDSPF command. Indicators are special fields used to control display processing. Using DDS keywords is described in detail in the *DDS Reference*.

DDS for Display File:


Field
Length
Line
Position
Constant

```
     A          R RECORD
     A                                  3  2'Customer Number:'
     A            CUST           5  0   3 20
     A                                  3 27'Customer Name:'
     A            NAME          20      3 44
     A                                  4 27'Address:'
     A            ADDR          20      4 44
     A            CITY          20      5 44
     A            STATE          2      5 66
     A            ZIP            5  0   5 70
     A
```

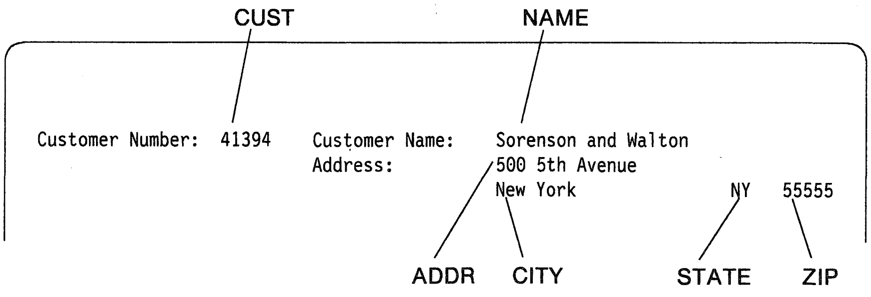
RSLH118-2

Record Format Used by the Program:

CUST	NAME	ADDR	CITY	STATE	ZIP

P7730272-0

Record Format on the Display:



```
Customer Number:  41394    Customer Name:    Sorenson and Walton
                           Address:          500 5th Avenue
                                             New York                NY   55555
```

RSLH714-0

Figure 4-1. Record Formats in the Program and on the Display

Indicators

Indicators are one-character fields that exist either in the input records and output records used by the program or in a special indicator area. An indicator is on if it has the value 1 and off if it has the value 0. You can use indicators to pass information from a program to the system or from the system to a program. You specify how indicators are to be used through the DDS for the display file.

There are two types of indicators for display files:

- Option indicators: Pass information from an application program to the system. These typically are used to control the processing of a particular record format by the system.
- Response indicators: Pass information from the system to an application program when an input request completes. Response indicators can inform the program which function keys were pressed by the work station user or whether data was changed by the work station user.

Fields

The fields you describe in the record format can be used in the following ways:

- *Input fields.* Fields that are passed from the device to the program when the program reads a record. Input fields can be initialized with a default value (specified in the record format for the display file). If the work station user does not change the field and the field is selected for input, this data (default value) is passed to the program. Input fields that are not initialized are displayed as blanks into which the work station user can enter data. By default, input fields are underlined on the display.

  Note: Trailing blanks on input fields are replaced by null characters; therefore, the Insert key can be used in all input fields that end in blanks.

- *Output fields.* Fields that are passed from the program to the device when the program writes a record to a display. Output fields contain data provided by the program, not by the work station user. The DFT or DFTVAL keyword can be used to specify an initial value for a named output field, see "Using the DFT and DFTVAL Keywords" on page 4-106 for more information.

  In the case of subfiles, which are special records used to display lists of information, output fields are returned to the program as if they were output/input fields.

- *Output/input fields.* Fields that are passed from the program when the program writes a record to a display and passed to the program when the program reads a record from the display and the field is selected for input. (Output/input fields are indicated as both fields (B) in DDS.) By default, these fields are underlined on the display. Output/input fields are usually used when the program displays data that can be changed by a work station user. The DFT or DFTVAL keyword can be used to specify an initial value for a named output field, see "Using the DFT and DFTVAL Keywords" on page 4-106 for more information.

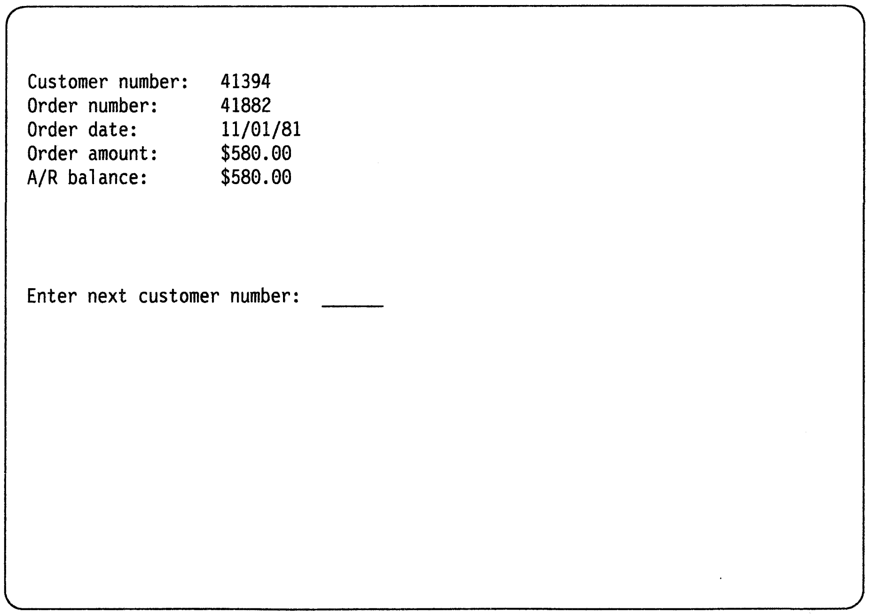
- *Hidden fields.* Fields that are passed from and to the program but are not sent to the display. Hidden fields are useful in applications involving subfiles. For example, a subfile record can contain record key information in a hidden field. The hidden field cannot be seen by the work station user, but is returned to the program with the subfile record so that the program can return the record to the database.

- *Constant fields.* Fields that are passed to the display but are unknown to the program. These fields are unnamed and have their constant values defined in the DDS for the file. DATE, TIME, and MSGCON are keywords that are allowed only on constant fields and whose constant values are determined during program run time (DATE and TIME) or DDS compile time (MSGCON).
- *Message fields.* Output fields that are treated as messages.
- *Program-to-system fields.* Output-only fields that are named, numeric or alphameric. They are used to communicate between an application program and the system. Program-to-system fields do not appear on the display. That is, your program can place data in these fields and the system will use that data to control its processing on an output operation, but the work station user cannot see the contents of these fields.

A field is input-capable if it is an input field or an output/input field. Each input-capable field has a special attribute called a modified data tag (MDT). The MDT is set on by the display device when any data is keyed into the field. It can also be set on and cleared by the application program.

The maximum number of fields that you can specify for each record format is 32 763. (See "Limitations on the Number of Input-Capable Fields" on page 4-70 for information on the number of input-capable fields that can be specified.) The maximum combined length for all fields and indicators in a record format is 32 763.

The following display shows output fields and input fields displayed in response to a request (in the form of entering a customer number in an input field) from a work station user.



```
Customer number:   41394
Order number:      41882
Order date:        11/01/81
Order amount:      $580.00
A/R balance:       $580.00



Enter next customer number:  _____
```

The prompts, *Customer number:*, *Order number:*, *Order date:*, *Order amount:*, *A/R balance:*, and *Enter next customer number:* are constants. The data associated with these fields (41394, 41882, 11/01/81, $580.00, and $580.00) is displayed in output fields. The data is passed from the application program to the system, and the system displays it. The field following the constant *Enter next customer number:* is an input field. The work station user must enter data into this field (the cursor is positioned at the beginning of the input field). Input fields are underlined by default. Editing of the field is normally performed within DDS.

You must specify the location for each field except when the field is a hidden field, a message field, or a program-to-system field (H, M, or P specified in position 38 in DDS), or when the field is in a subfile message record format. You cannot specify line 1, position 1 for location, except when you define a record that can start in any line.

The maximum length of a character field or numeric (zoned decimal) field is the number of positions remaining (relative to the start location of the field) on the display minus 1. The numeric (zoned decimal) field is limited to a maximum length of 31.

Specifications for the fields you describe can be retrieved from a previously described field. The previously described field can be either in a database file or already defined in the DDS source for the display file. When you use field specifications from a database file, binary and packed decimal fields are changed to zoned decimal fields. These fields that you use to define other fields are called *reference fields.*

Option indicators can be used to select fields to display different data on different output operations instead of defining a different record format for each combination of fields. Using option indicators, you can define fields in the same record to overlap each other. That is, you could define two fields to occupy the same positions on the display, and use option indicators to select which of the overlapping fields is to be displayed (see "Indicators and Condition Names" on page 4-77). If more than one overlapping field is selected on the same output operation, only the first field selected is displayed.

Basic Display File Use

You build the user interface for an application program by using the support provided by display management. The basic support provided by display management:

- Allows you to define screens into which the work station user can easily enter input. These screens are made up of record formats which are stored in a display file.
- Can automatically associate the display file used by a program with the work station from which the program is called (the *REQUESTER device).
- Presents screens at a display at the request of an application program so that the work station user can use the information provided by the program. This program can control which fields from the display file formats are displayed on the screen.
- Maintains lists of data called subfiles so that the work station user can work with lists of information. This feature can reduce the amount of system activity required between the work station user and the program.
- Handles some validity checking of the work station user's data returning to the application program.
- Displays error messages to the work station user based on indicators passed to the system by the application program.

Display support is *record-oriented,* which means that all data written to and read from the display by the application program is done with records. Records consist of fields, which are individual items of data. The high-level languages in which application programs are written have I/O statements that give data to the system to be written to the display and receive data from the system that it read from the display in the form of records.

The I/O statements of the high-level languages also refer to record formats, which are defined using DDS. On output, a record format describes how the data given by the program is to be presented on the display as well as how the data is to be processed before presenting it. On input, the record format is used to control some display functions, to extract the program data from all the data which is on the display, and to present that data back to the application program.

Working with One Record

The system allows you to have more than one record on the display at a time. However, to illustrate the basic points of display files working with only one record will be discussed first.

Write-Read Operation

Some high-level languages have a write-read operation which writes information on the display and reads the user response in one statement. For example, RPG has the execute format (EXFMT) operation. This kind of operation is useful if you need to both present new information on the display and request information from the work station user at the same time. You can also use a write operation followed by a read operation to the same record format to simulate this operation in languages that do not support a combined write-read operation.

When this operation is performed, the following happens:

1. The program calls the system display support giving it the data to show on the display and the record format to use when writing and reading that data.
2. The system combines that data with the information it finds in the record format and constructs the data stream to be sent to the display.
3. The data stream is then sent to the display and the keyboard is unlocked.
4. The user keys the data in the fields which allow input and presses the Enter key or some other function key.
5. The data is then sent from the display to the system. The system decodes it and extracts only the information that the application program needs to know and returns that data to the application program.

When you work with only one record format, this write-read style of working with it is the most common. On the write portion of the operation, you provide the data that the user will see. On the read portion, you receive data back that the user has entered or changed.

Working with More than One Record

You can also work with more than one record on the display at a time. Each record takes up some of the lines of the display, and, in general they never overlap. Ordinarily, writing a record to a display will cause the other records on the display to be cleared. You tell the system that you want to keep the other records on the display by using the OVERLAY keyword in the DDS for the record.

In order to write several records onto the display, you need to separate the two halves of the write-read operation. When you perform a write operation, the system takes the data from the record that you have provided it, combines it with the information specified in the record format, and places it on the display. This arrangement lets you write several records on the display before any one of them is read back in. You can lock the keyboard until the read operation is performed by specifying the LOCK DDS keyword on all the record formats or by specifying DFRWRT(*YES) on the file.

You would use multiple record formats when you have information from your application that needs to be presented together but naturally falls into two or more pieces. For example, you could construct an application that displays information for a state at the top of a display with one record format, and with another record format you could provide the information for a particular region within that state.

You can use multiple record formats when you want to present lists of information in a subfile to the work station user. Typically one record format on the display acts as the subfile control record that directs the system to display the data in a subfile, and another record format provides instructions or a list of the function keys above or below the list.

Defining Function Keys

A display device typically has both a keyboard and a display screen. In order to write an application using a display device, you have to control both the functions of the keys at the keyboard, and the contents of the display. The application controls which function keys have meaning at any time by enabling them with the CFnn and CAnn DDS keywords when defining the file. You can define these on each record format in the file, or you can define them once for the whole file.

When you define your function keys in the file, you need to decide how you will tell which function key was pressed when you perform the read operation. One way of knowing this is to define an indicator to be returned to you or set when a particular key is pressed. The kind of indicators that return information back to the application are called response indicators. You have 99 indicators with which you can receive responses of this kind.

Note: Response indicators can be used for things other than finding out about function keys. For example, you can use them to tell when the data in a field on the display has changed.

Another way of knowing what function key was pressed is to examine the *input/output feedback area* that is provided by the data management support of the system. The way you get information from the input/output feedback area depends on the programming language you are using. This is described in the appropriate language manual.

Getting Messages on the Display

The system provides several ways of presenting messages on the display.

- You can get messages on the message line supported by the hardware using the ERRMSG and ERRMSGID keywords.
- You can put messages into an output field on the display using the MSGID keyword.
- You can construct lists of messages to be presented in a subfile using some of the system message support.

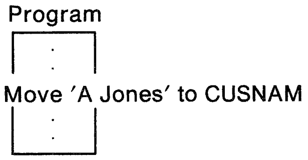
However, the easiest way to present messages is to use the ERRMSG keyword in your DDS. When you use ERRMSG, the record that you want to present the message for must already be on the display. If it is not, the ERRMSG function is not performed.

When you use the ERRMSG keyword to present a message, that message is written to the message line of the display, which is usually the line at the bottom of the screen. The work station user would then press the Reset key to clear the message from the display and unlock the keyboard to continue typing. You provide the text of the message right on the ERRMSG keyword. When you write a record that has the ERRMSG keyword in effect, it causes that message to appear. Typically, you would use an option indicator to cause the ERRMSG keyword to take effect. When the application program turns on an option indicator, the keywords that have that option indicator specified in the DDS then take effect. In this case, an application program would leave an indicator that optioned an ERRMSG keyword off until the message needed to be displayed.

A Small Application

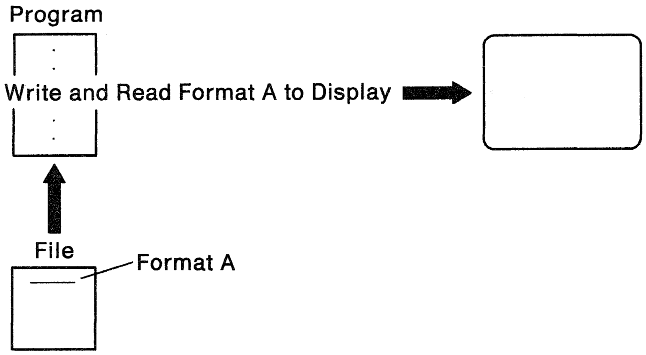
The following steps illustrate how an application program could use a display file and some of the concepts we have just introduced. The example uses functions provided by high-level language programs and DDS. The example detects an error in the input and uses the display support functions expressed in the DDS to display an error message.

1. The application program moves the data *A Jones* into a variable named CUSNAM.



RSLH128-1

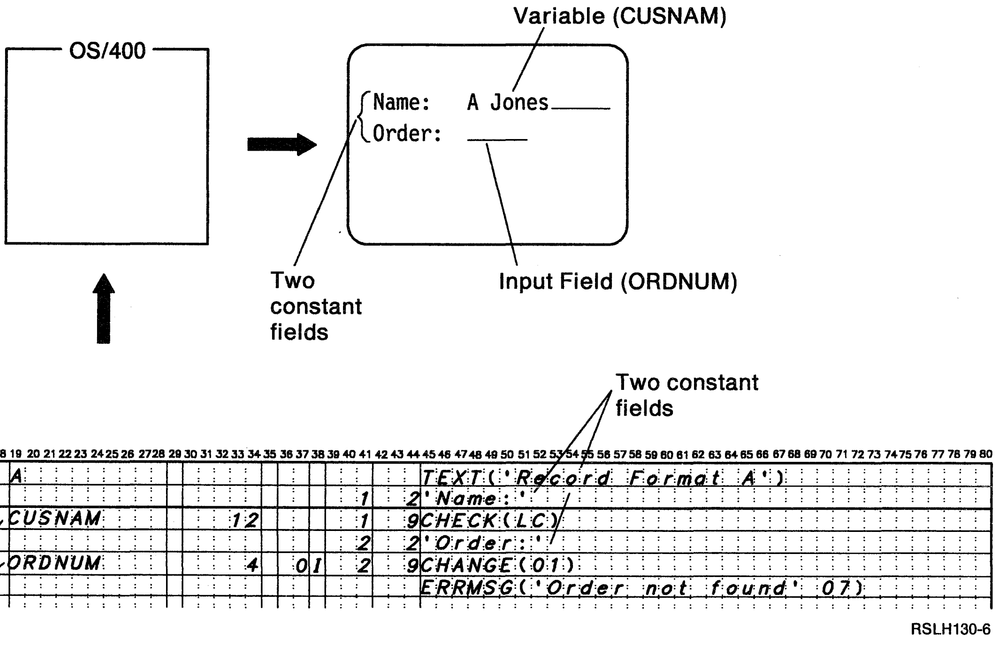
2. The program writes record format A to the work station and then reads record format A from the work station by using the file I/O statements available to it in the high-level language language.



RSLH129-2

3. Using record format A, the system displays the data on the screen. The two constant fields (*Name:* and *Order:*) are specified in the record format. The variable from the application program is displayed following *Name:*.

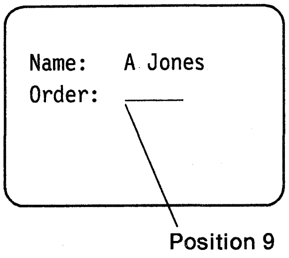
   The input field following *Order:* is to be filled in by the work station user. The underline of the field is done automatically by OS/400.



```
     A          R A                         TEXT('Record Format A')
     A                                  1  2'Name:'
     A            CUSNAM        12      1  9CHECK(LC)
     A                                  2  2'Order:'
     A            ORDNUM         4  0I  2  9CHANGE(01)
        07                                  ERRMSG('Order not found' 07)
```

RSLH130-6

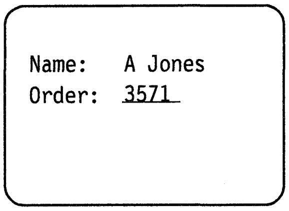
4. The cursor is placed at the first position of the input field by the system since this is the first field on the display where data may be entered.



```
Name:   A Jones
Order:  _____
```

RSLH123-2

5. The work station user keys the order number 3571 into the input field and presses the Enter key to indicate that he has finished entering the data.



```
Name:   A Jones
Order:  3571_
```

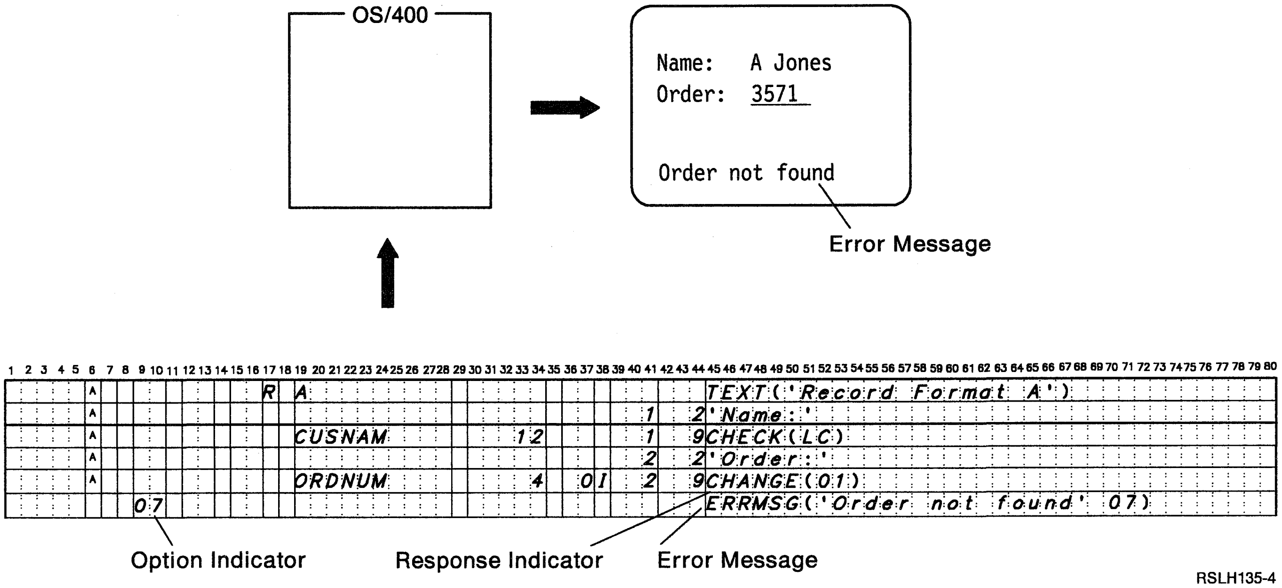
RSLH124-1

6. The system places the data, 3571, into the variable ORDNUM and turns on indicator 01 to indicate that the field has been changed.

7. After the read operation in the program completes, the program checks the value of the order number, and if it is incorrect, the program turns on indicator 07, which will tell the system to display an error message. The program then writes record format A to the display again. This gives control to the system which displays the error message caused by indicator 07.

   The system displays the error message on the bottom line and displays the input field (3571) in reverse image with the cursor at the beginning of the input field.

   The indicator included in the ERRMSG keyword, 07, is always turned off when the record is returned to the program. This takes away the need for the program to reset error indicators.



```
Name:   A Jones
Order:  3571

Order not found
```

```
     A          R A                         TEXT('Record Format A')
     A                                  1  2'Name:'
     A            CUSNAM        12      1  9CHECK(LC)
     A                                  2  2'Order:'
     A            ORDNUM         4  0I  2  9CHANGE(01)
        07                                  ERRMSG('Order not found' 07)
```

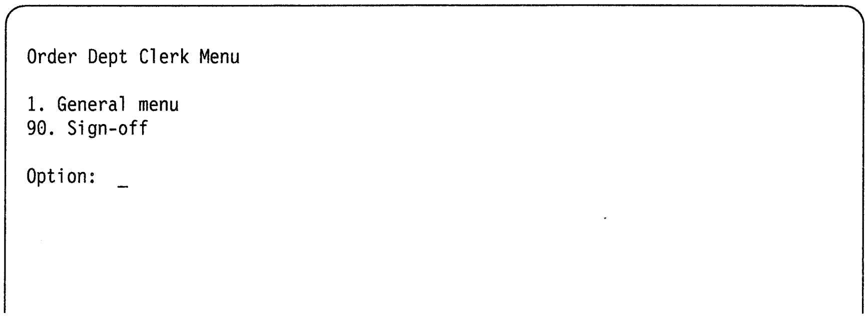
RSLH135-4

Examples of Screen Definitions

This section contains examples of describing menus, prompts, and information displays. See the *CL Programmer's Guide* for a discussion of creating display file and program menus. Also, refer to the *SDA User's Guide/Reference* for detailed information on easily creating the DDS for a display file.

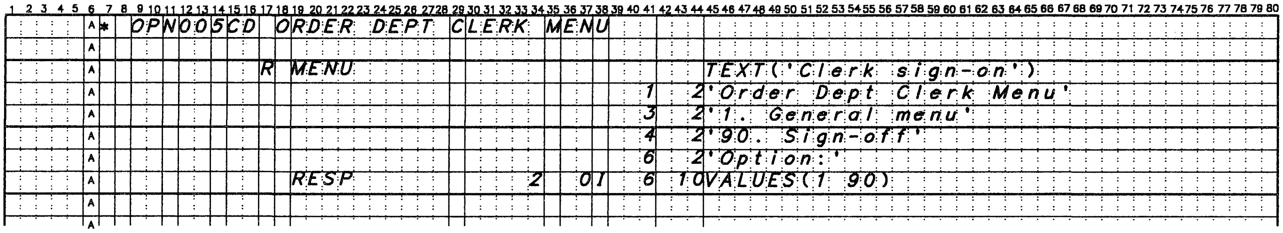
Defining a Menu Display

This example shows how to describe the record format for the Order Department Clerk Menu of an order entry application. The Order Department Clerk Menu looks like this:



```
Order Dept Clerk Menu

1. General menu
90. Sign-off

Option:  _
```

The DDS to format this menu looks like this:



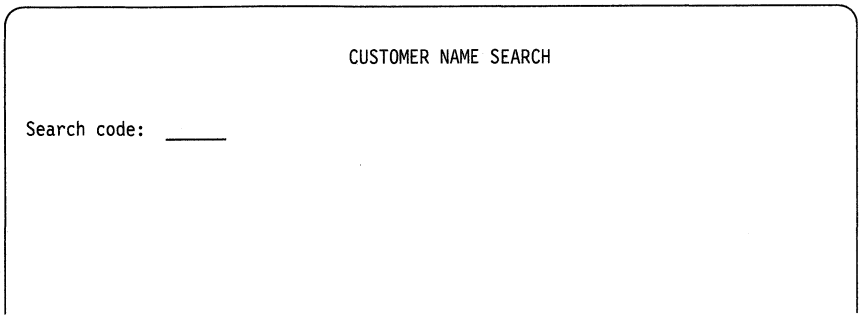
```
     A* OPN005CD ORDER DEPT CLERK MENU
     A
     A          R MENU                          TEXT('Clerk sign-on')
     A                                  1  2'Order Dept Clerk Menu'
     A                                  3  2'1. General menu'
     A                                  4  2'90. Sign-off'
     A                                  6  2'Option:'
     A            RESP           2   0I 6 10VALUES(1 90)
     A
     A
```

RSLH167-0

The allowable options are 1 and 90, as indicated by the VALUES keyword for the RESP input field. When the system reads this record format from the display, it will check to make sure that the value entered in the field is either 1 or 90. If it is not, the system will present a message to the work station user. Therefore, the application program does not need to make this check.

Defining a Data Entry Display

This example shows how to describe the record format for a *Customer name search* prompt. This display has one field which asks for a zip code that is to be used to search a customer name file. The prompt and the DDS to format this prompt look like this:



```
                    CUSTOMER NAME SEARCH


Search code: _____
```

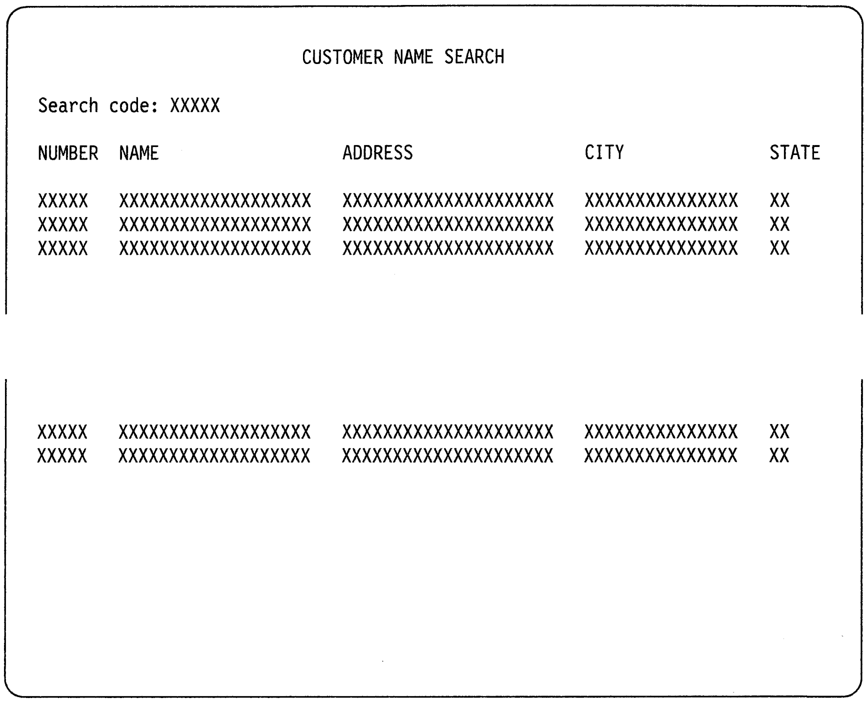
```
     A* DISPLAY CUS220D CUSTOMER NAME SEARCH
     A                                      REF(DSTREF)
     A          R NAMESR
     A                                  1 29'CUSTOMER NAME SEARCH'
     A                                  3  2'Search code:'
     A            SEARCH    R        I  3 15
```

RSLH168-0

The zip code is entered into the *Search code* field, which is underlined by default. All input fields are underlined by the system unless they are explicitly defined differently. The R in position 29 indicates that the field description of SEARCH is contained in another file (DSTREF), which is referenced in the file-level REF keyword. By default, the display size is only 24 by 80. As it stands, this prompt could not be displayed on any other size screen, because neither the DSPSIZ keyword nor display size conditioning is specified.

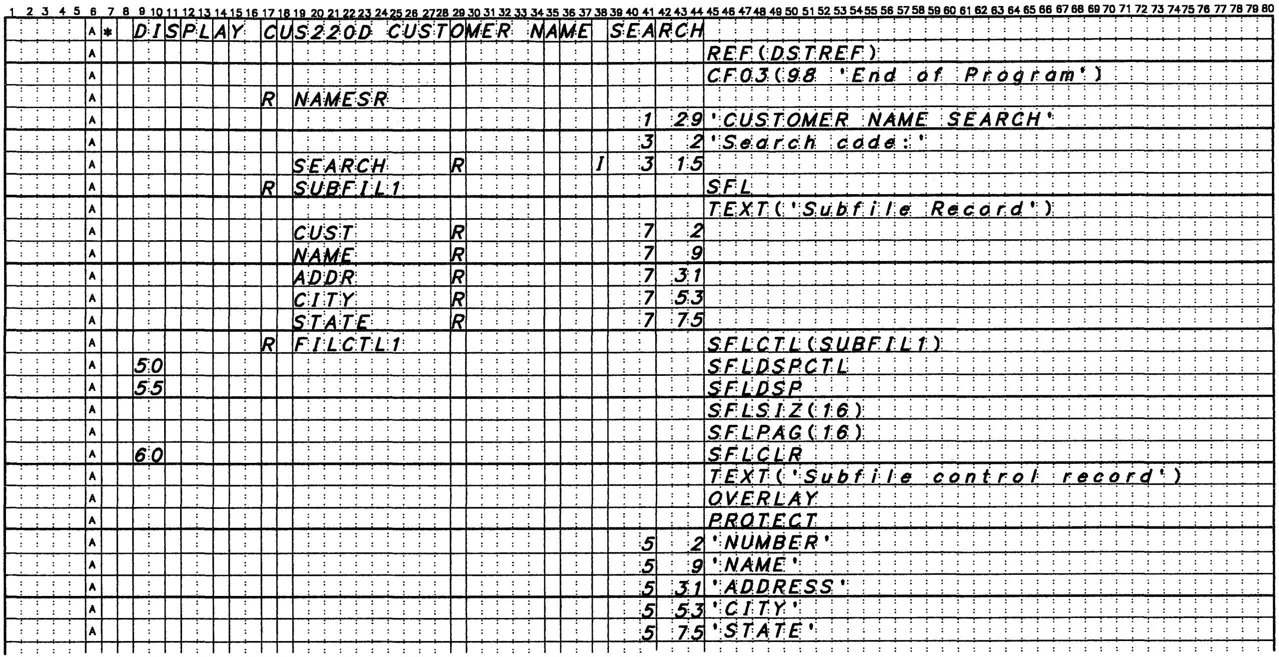
Defining a List Display

This example describes how to display a group of records containing customer names and addresses. The records are grouped in a subfile (see “Subfiles” on page 4-31) and are displayed on the same display as the *Search code* prompt, which remains on the display. (The *Search code* prompt is described under “Defining a Data Entry Display” on page 4-17.) The entire Customer Name Search display looks like this (the Xs indicate where the records are displayed):



```
                          CUSTOMER NAME SEARCH

Search code: XXXXX

NUMBER  NAME                   ADDRESS                 CITY               STATE

XXXXX   XXXXXXXXXXXXXXXXXXXX   XXXXXXXXXXXXXXXXXXXXXX   XXXXXXXXXXXXXXXX   XX
XXXXX   XXXXXXXXXXXXXXXXXXXX   XXXXXXXXXXXXXXXXXXXXXX   XXXXXXXXXXXXXXXX   XX
XXXXX   XXXXXXXXXXXXXXXXXXXX   XXXXXXXXXXXXXXXXXXXXXX   XXXXXXXXXXXXXXXX   XX



XXXXX   XXXXXXXXXXXXXXXXXXXX   XXXXXXXXXXXXXXXXXXXXXX   XXXXXXXXXXXXXXXX   XX
XXXXX   XXXXXXXXXXXXXXXXXXXX   XXXXXXXXXXXXXXXXXXXXXX   XXXXXXXXXXXXXXXX   XX
```

The following DDS is for formatting the entire display including the DDS for the *Search code* prompt:



```
     A* DISPLAY CUS220D CUSTOMER NAME SEARCH
     A                                      REF(DSTREF)
     A                                      CF03(98 'End of Program')
     A          R NAMESR
     A                                  1 29'CUSTOMER NAME SEARCH'
     A                                  3  2'Search code:'
     A            SEARCH    R        I  3 15
     A          R SUBFIL1                   SFL
     A                                      TEXT('Subfile Record')
     A            CUST      R           7  2
     A            NAME      R           7  9
     A            ADDR      R           7 31
     A            CITY      R           7 53
     A            STATE     R           7 75
     A          R FILCTL1                   SFLCTL(SUBFIL1)
     A  50                                  SFLDSPCTL
     A  55                                  SFLDSP
     A                                      SFLSIZ(16)
     A                                      SFLPAG(16)
     A  60                                  SFLCLR
     A                                      TEXT('Subfile control record')
     A                                      OVERLAY
     A                                      PROTECT
     A                                  5  2'NUMBER'
     A                                  5  9'NAME'
     A                                  5 31'ADDRESS'
     A                                  5 53'CITY'
     A                                  5 75'STATE'
```

RSLH170-1

The records contained in the subfile are described in record format SUBFIL1, which is the subfile record format. The descriptions of the fields are taken from the field reference file DSTREF, due to the file-level REF keyword and the R in position 29. The keyword SFL defines SUBFIL1 as a subfile record format.

The search code specified in the *Search code* prompt defines the group of records to be displayed. When the search code is entered, the application places the records in the group into the subfile SUBFIL1 one at a time. When all the records in the group (up to 16) are in the subfile, the subfile is displayed by writing the subfile control record format FILCTL1. The keyword SFLCTL contains the name of the corresponding subfile record format. The column headings are unnamed constant fields specified in FILCTL1 and are displayed on line 5. The first subfile record is displayed on line 7 as specified in SUBFIL1.

The OVERLAY and PROTECT keywords in FILCTL1 keep the search code (SEARCH) on the display and prevent another search code from being entered by protecting the input field.

To end the program, the work station user presses the CF03 key, which is specified at the file level. The application program detects this by looking at response indicator 98 and ends its processing.

Display File Operations

There are three basic requests for transferring data to or from a display that all programming language requests result in:

- Write requests to write data (for example, Send File (SNDF) command)
- Read requests to read data (for example, Receive File (RCVF) command)
- Combination Write-Read requests to write data and read data (for example, Send/Receive File (SNDRCVF) command)

Table 4-1 shows the I/O requests supported by the operating system and the equivalent high-level language operations:

Table 4-1. Display File Operations Supported by the Operating System and the Equivalent High-Level Language Commands

Operation	CL Command	RPG/400 Operations	COBOL/400 Statements	BASIC Statements	C/400 Functions
Open		OPEN	OPEN	OPEN	fopen
Acquire		ACQ	ACQUIRE		QXXACQUIRE
Release		REL	DROP		QXXRELEASE
Get Attributes		POST	ACCEPT		QXXDEVATR
Write	SNDF	WRITE, output specifications	WRITE	WRITE	fwrite QXXPGMDEV QXXFORMAT
Read(wait)	RCVF WAIT(*YES)	Primary or secondary file input, READ	READ	READ	fread QXXPGMDEV QXXFORMAT
Read(nowait)	RCVF WAIT(*NO)				
Cancel Read	ENDRCV				
Wait	WAIT				
Invite Device	SNDF[1]	WRITE[1]	WRITE[1]		QXXPGMDEV QXXFORMAT fwrite[1]
Read from Invited Device		READ	READ		QXXREADINVDEV
Cancel Invite	SNDF	WRITE	WRITE		fwrite QXXPGMDEV QXXFORMAT
Write-Read(wait)	SNDRCVF WAIT(*YES)	EXFMT			
Write-Read(nowait)	SNDRCVF WAIT(*NO)				
Close	RETURN, RCLRSC	CLOSE, RETRN	CLOSE, CANCEL, STOP RUN	CLOSE, END	fclose

[1] This is the write of a record format with the INVITE DDS keyword optioned on.

If an error occurs during an I/O operation to a display file, the major/minor return code field in the file dependent I/O feedback area may be used to help diagnose the error and determine the error recovery action needed.

Operations on Files

Several operations exist that may be performed on a display file. These operations are described in the following sections.

Open Operation

An open operation performs the setup needed for an application to perform I/O operations. When the file is opened, one device can also be prepared for I/O operations. This device is said to be implicitly acquired when the file is opened. The following rules determine the devices implicitly acquired when a display file is opened:

- When a display file with a single display device defined is opened, the display device specified on the DEV keyword of the CRTDSPF, CHGDSPF, or the OVRDSPF command is acquired. If *REQUESTER was specified, then the display at which the work station user requested the program is used.
- When a multiple-device display file is opened by a CL program, all the display devices specified on the DEV keyword of the CRTDSPF, CHGDSPF, or the OVRDSPF command are acquired. A CL program opens the display file on the first use of a SNDF, RCVF, or SNDRCVF command.
- When a multiple-device display file is opened by any other high-level language, the first display device specified on the DEV keyword of the CRTDSPF, CHGDSPF, or the OVRDSPF command is acquired. Additional devices whose names were specified on the DEV keyword may be prepared for I/O operations through the acquire-device operation.
- When a display file has a value of *NONE for the DEV keyword, no display device is acquired. All devices must be acquired by the acquire operation before being used for I/O operations.

The value specified on the WAITFILE parameter specified on the CRTDSPF, CHGDSPF, or OVRDSPF command is used to determine how long the open operation should wait for file resources to become available so they can be allocated. If a file resource, such as a device, does not become available and the wait time specified ends, the open operation fails.

Implicitly acquiring a device when the file is opened results in the following:

- The screen is cleared completely and the cursor is placed in the upper-left corner of the display.
- The keyboard is unlocked.

Any device that is to be implicitly acquired at open must be varied on. Switched-line devices can also be acquired if they are in a vary on pending state. Also, a device other than the *REQUESTER device cannot be acquired if it is signed on.

After a file is successfully opened, an open feedback area is available to the program. This feedback area contains, among other things, the number of rows and columns on the display and the name and library of the file after overrides have been applied.

Close Operation

The close operation makes the display file ineligible for any further I/O operations between the program and the system. Refer to the appropriate high-level language manual for information on how to start the close operation.

If the display file is not being shared, the close operation also implicitly releases all the devices acquired to the file and deallocates any file resources allocated by the open operation or the acquire-device operation.

If the close operation is successful, the only valid operation to the file is open. If the close operation fails, the program should issue the close operation a second time.

Acquire-Device Operation

The acquire-device operation prepares a device for I/O operations between the program and the system through a specific file. This operation is used in multiple device file applications or if you are performing error recovery in your application. A successful acquire-device operation results in the following:

- The screen is cleared completely and the cursor is placed in the upper-left corner of the display.
- The keyboard is unlocked.

The value specified on the WAITFILE parameter specified on the CRTDSPF, CHGDSPF, or OVRDSPF command is used to determine how long the acquire-device operation should wait for a device to become available so it can be allocated. If a device does not become available and the wait time ends, the acquire-device operation fails.

A device cannot be allocated unless it is varied on. Switched-line devices can be allocated if they are in a vary-on-pending state. Also, a device other than the *REQUESTER device cannot be allocated if it is signed on.

After a successful acquire-device operation, the device is said to be acquired to the file.

If an acquire-device operation is not successful, the release-device operation is the only valid operation to the device.

Acquiring devices from the file is controlled primarily by the application. The system allows only one *REQUESTER device to be acquired to any device file, including a multiple-device display file.

No automatic security checking is performed on the user of a display device when the program acquires it. The program must handle any security checking required.

Release-Device Operation

The release-device operation makes a device ineligible for any further I/O operations through a file. This operation is used in multiple device file applications or if you are performing error recovery in your program. If the device being released is invited, the invite is ended. If the invited device had data available, the data is lost. The release-device operation can only be performed on devices which are currently acquired to the file.

The release-device operation can also be used to recover from errors from acquire-device, I/O, and release-device operations. After a device is released, it must be acquired again with another acquire-device operation before any I/O operations can be directed to it. If a program is written to recover from errors by releasing a device and then acquiring it again, a value other than *IMMED should be specified on the WAITFILE keyword. This is because it takes the system a short time to transfer the allocation of a display device description from a job, to the subsystem, and back again.

Get-Attributes Operation

The get-attributes operation allows the user to obtain certain information about a specific device. This operation is most useful for multiple device file applications as the information supplied is also available in the open and input/output feedback areas for the device that is implicitly acquired when the file is opened. Some information is available whenever the file is open while some information is available only when the device is acquired (this includes both implicit and explicit acquires). For more information on how to perform the get-attributes operation, see the appropriate high-level language manual. The following information on a specific device may be available to the application:

- The specific model of the display device. Valid only when the device is acquired to the file.
- The screen size of the display device. Valid only when the device is acquired to the file.
- Device acquired indicator. Indicates whether or not the device is currently acquired to the file.
- Device invite status. Indicates whether or not the device is invited, and if so whether or not the device has data available. Valid only when the device is acquired to the file.
- *REQUESTER device indicator. Indicates whether or not the device is the *REQUESTER device.

See Appendix A, “Feedback Area Layouts,” for a description of all the information available from the get-attributes operation.

Operations on Record Formats

Several operations exist that may be performed using record formats contained in a display file. When a record format operation is performed, the program is usually requesting that data be moved between the program and the display. This data is placed in a buffer accessible by the program for sending to and from the system. The high-level language in which you are writing may manage the allocation and layout of the buffers for you (as RPG/400, BASIC, and CL do) or it may require that you declare an area to use for the file buffer, such as COBOL/400.

Write Operation

A write operation passes a record from the program to the system. The display file record format contains the information necessary for the system to handle the record. Basically, the write operation results in the following:

- The screen is cleared completely.
- The record is written to the screen. All input-capable fields are initialized with modified data tags off.
- The keyboard is unlocked. (See "Keyboard-Locking Considerations" on page 4-113 for more information on keyboard locking.)

You can control these steps using the following DDS keywords:

- OVERLAY. The screen is not cleared completely; only record formats with lines affected by the format being displayed are cleared. The OVERLAY keyword does not prevent the screen from being erased if it is in effect for the first write operation after a file is opened unless the DDS keyword ASSUME is specified for any record format in the file. See "Using the OVERLAY and ERASE Keywords" on page 4-92.

- CLRL(*NO). The overlapped record format is not cleared completely. Only the lines required by the overlapping record format are overlaid. See "Using the Clear Line (CLRL) Keyword" on page 4-95.

- PUTOVR. Only those fields for which the OVRDTA keyword is specified are sent to the display when a subsequent write or write-read operation is issued to the same record format. If the OVRATR keyword is in effect and the OVRDTA keyword is not in effect, only the attributes for the field are sent to the display. See "Using the PUTOVR, OVRDTA, and OVRATR Keywords" on page 4-100.

- ERASE. Specified records are erased. See "Using the OVERLAY and ERASE Keywords" on page 4-92.

- ERASEINP. Input fields are erased. See "Controlling Display Functions on Output Operations" on page 4-90.

- MDTOFF. Modified data tags are reset. See "Controlling Display Functions on Output Operations" on page 4-90.

- PROTECT. All input fields currently on the display are protected. See "Controlling Display Functions on Output Operations" on page 4-90.

- PUTRETAIN. Fields of the previous record are retained. (Even though the field data is not retransmitted, the field attributes can be retransmitted, which is useful for highlighting the contents of a field.) See "Using the PUTRETAIN Keyword" on page 4-104.

- LOCK. The keyboard is not unlocked. See "Controlling Display Functions on Output Operations" on page 4-90.

Note: When you use the CLRL, OVERLAY, PUTOVR, PUTRETAIN, ERRMSG, or ERRMSGID keyword, you should specify RSTDSP(*YES) on the CRTDSPF or

CHGDSPF command; otherwise, the data on the display may be lost if the file is suspended.

Other DDS functions can affect the write operation. For a write operation to a user-defined data stream (USRDFN keyword), the functions performed are determined by the user-supplied controls.

Device support saves all data read from input-capable fields for records currently on the display in a save area. The output/input fields within this save area are updated on output operations.

For output operations, the following happens:

- Input-only fields are initialized to zeros (numeric fields), blanks (character fields), or a default value (DFT keyword) from the display file.
- Output/input fields, hidden fields, and program-to-system fields are initialized to the contents of the output buffer. If this output operation is caused by the initialize record function (INZRCD keyword), no output buffer is available. Output/input fields and hidden fields are initialized similar to input-only fields. Output/input fields are input and output capable.
- Output-only fields are not part of the input buffer unless they are part of a subfile record, in which case they are saved as if they were output/input fields.
- All response indicators for this record are set off.

Note: For input-capable fields, if the PUTRETAIN or ERASEINP keyword is in effect, the save area for the field remains unchanged.

Neither the input nor the output buffer is changed during write operations.

Read Operation

A read operation passes a record from the system to the program. The display file record format contains the information necessary for the system to handle the record. The work station user must perform a required action such as pressing the Enter key or a function key to pass the data to the system. Basically, the read operation results in the following:

1. If the keyboard is locked, it is unlocked.
2. The action performed by the work station user locks the keyboard.
3. All input-capable fields and response indicators (if INDARA is not specified) of the record are mapped to the input buffer of the program.
4. If the device is in the invited state, the read operation retrieves the input when it becomes available and the device is no longer invited. See “Invite-Device Operation” on page 4-26 for more information on invited devices.

You can control these steps using the following DDS keywords:

- UNLOCK. The keyboard is left unlocked after the read operation.
- GETRETAIN. Input fields in the record just read are retained on the display.

For input operations, the following happens in the order given:

1. For an input-only operation, all response indicators for this record are set off and the read operation is issued.
2. Character fields received from the display are right- or left-adjusted and padded with blanks or truncated as necessary. The default is left-adjust, which can be overridden using the AUTO or CHECK keyword.
3. Numeric fields received from the display have the following done to them:
   a. If the field is negative, the zone portion of the units position is set to a D (see "Negative Numeric Input Data Considerations" on page 4-71).
   b. All nonnumeric characters are removed and the numeric characters are compressed.
   c. Signed numeric fields are right-adjusted and numeric-only fields are decimal aligned.
   d. The field is padded with zeros or truncated as necessary.
   e. Field validation is performed.

All fields received from the display whether they are part of the selected record or not are handled in this way.

If any field validation errors are detected, a message is sent to the work station user so that the error can be corrected. This process is repeated until there are no longer any errors. The save area for the requested record is then copied into the input buffer.

Note: To process input data for a read operation with no record format name, device support uses the last record written to the display that contains at least one of the following:

- Input-only fields
- Output/input fields
- Hidden fields

If no such format is on the display, device support uses the last format written to the display that did not contain these kinds of fields, for example, an output-only record that specifies valid command keys. If no such record exists on the display, an error message is returned to the program.

Write-Read Operation

The write-read operation is a combination of a write operation and a read operation to the same record format in one high-level language statement like the SNDRCVF command in a CL program. It behaves as if you had specified a read operation immediately following a write operation.

Invite-Device Operation

The invite-device operation is used to send a request for input to a display device and return to the program without waiting for the input to arrive. This allows a program to request input from one or more devices but continue processing without waiting for any of the devices to respond. When the program is ready to process the input, the data can be received from any of the invited devices by performing a read-from-invited-devices operation. The invite-device operation is done by performing a write operation using a record format with the INVITE DDS keyword in effect. Refer to the appropriate high-level language manual to determine how to perform a write operation and how to use indicators to control the INVITE DDS keyword. See the *DDS Reference* for information on how to define a record format with the INVITE DDS keyword.

Once a device is invited, the valid operations to receive data from the device are the read-from-invited-devices operation and the read(wait) operation directed to a specific device. Cancel invite is also a valid operation to an invited device.

If the multiple-device display file was created with DFRWRT(*YES) specified, an output operation with the INVITE DDS keyword optioned on will cause the output that has been postponed to be displayed on the screen before the device is invited.

If you want to invite a device but have no data to send to it, perform the output operation with a record format which contains the INVITE DDS keyword optioned on but has no output-capable fields.

Read-from-Invited-Devices Operation

The read-from-invited-devices operation provides a means of waiting for and receiving data from any one of the invited devices. This method of inviting a device and then reading from the invited device is useful when the application must control the amount of time spent waiting for the work station user to respond. When the read-from-invited-devices operation is performed, the program waits for the time interval specified on the WAITRCD keyword of the CRTDSPF, CHGDSPF, or the OVRDSPF command. The wait can be ended in the following ways:

- Data becomes available from an invited device. The device name, the results of the operation, and any input data are passed to the program. Once data has been received, the device is no longer invited and must be invited again by an invite-device operation if more data is to be received from the device by a read-from-invited-devices operation.
- No-device-invited signal. Indicates that none of the devices associated with the file are in the invited condition. Refer to the appropriate high-level language manual for information on how this will be communicated to the program.
- Job-ended-controlled signal. Indicates that the job that the program is running in is being ended with the controlled option through the End Job (ENDJOB), End System (ENDSYS), Power Down System (PWRDWNSYS), or End Subsystem (ENDSBS) command. Refer to the appropriate high-level language manual for information on how this will be communicated to the program. This occurs only once in a process no matter how many multiple-device display files are in use. All invited devices remain invited.
- No-invited-devices-have-data-available signal. This occurs when no devices associated with the file have data available, the WAITRCD time is *IMMED, and none of the previous conditions apply. The invited devices remain invited. Refer to the appropriate high-level language manual for information on how this will be communicated to the program.
- Time-out-on-wait-for-data-from-invited-devices signal. This occurs when the WAITRCD value is a finite number of seconds, no data became available during that interval, and none of the previous conditions apply. Refer to the appropriate high-level language manual for information on how this will be communicated to the program. The invited devices remain invited.

Also, COBOL/400 provides a means of performing the read-from-invited-devices operation as if WAITRCD(*IMMED) had been specified. See the COBOL/400 manuals for information on the NODATA phrase and its effect on the read-from-invited-devices operation.

Read-from-invited-devices operation considerations:

- The read-from-invited-devices operation only accepts data from devices which are currently invited.
- If more than one device acquired to the display file has an invite outstanding, a read-from-invited-devices operation will return the next available record from one of the invited devices. If records are received from more than one device before the read-from-invited-devices operation, the other records will be kept for a subsequent read-from-invited-devices operation or for a subsequent read(wait) operation directed to a specific device.
- Once a device has responded and the input is received by the read-from-invited-devices operation, that device is no longer invited. It can be invited again by another invite-device operation but this should not be done until all the record formats on the display with input-capable fields have been read.
- A record format cannot be specified on the read-from-invited-devices operation. The record format returned from a display is the same as the last record format written to the display device.
- The timing function associated with the WAITRCD parameter may not force an end to the wait if the system is processing the Help key. In the following cases, the read-from-invited-devices function will not end until the work station user exits from the help information:
  - The system is displaying help that is defined by H specifications in the DDS for the display file.
  - The system is displaying help for a message when the work station is the requester device for the job and the device file specifies MAXDEV(1).

  You can force message help to end when the WAITRCD time ends by specifying a value greater than 1 for the MAXDEV parameter on the CRTDSPF or CHGDSPF command.

Cancel-Invite Operation

The cancel-invite operation is used to cancel the input request issued to a device that was previously invited through the invite-device operation. The input request is canceled by performing a write operation to the invited device. One of the following occurs:

- If the write request is received before the work station user responds to the input request from the invite-device operation, the input request is canceled and the record format specified on the write operation is sent to the device. If the record format has the DDS keyword INVITE optioned on, the device is invited again.
- If the write request is received after the work station user responds to the input request from the invite-device operation, the input request is not canceled and the write operation fails. The read-from-invited-devices operation or a read(wait) operation must be issued to receive the available data.

Releasing a device also implicitly cancels any input requests directed to the device. If the device has data available, the data is lost.

Additional Considerations for a Read Operation

When using a read operation, several considerations should be examined. The following sections describe these considerations.

UNLOCK Keyword on a Read Operation: For the 5250 display station, the read operation with the UNLOCK keyword in effect results in the following:

1. The 5250 display station does a hardware validity check on the fields. If no errors are found, the following is done:
   a. If the UNLOCK keyword is specified without the GETRETAIN keyword or if the UNLOCK(*ERASE) keyword is specified, all input-capable fields that are changed are cleared.
   b. If the UNLOCK keyword is specified with the GETRETAIN keyword or if the UNLOCK(*MDTOFF) keyword is specified, all modified data tags (MDTs) are reset.
   c. If the UNLOCK(*ERASE *MDTOFF) keyword is specified, all input-capable fields that are changed are cleared and their MDTs are reset.
   d. The cursor is repositioned to the field where the work station user can enter the next record.
   e. The keyboard is unlocked.
2. The system validity checks all the fields for all records on the display. If errors are detected, normal error retry is performed. A work station user could be keying into the next record when an error message is displayed.

   Note: The error message could refer to data that is no longer on the display because the data was erased.
3. Control returns to the program.

Notes:

1. If an application program detects input errors and sends error messages to the display, the message may refer to input that has been keyed over.
2. If the CHANGE keyword is specified and either the UNLOCK keyword is specified without the GETRETAIN keyword or the UNLOCK(*ERASE) keyword is specified, the associated response indicator is set on for the next input record.
3. When a read operation with the UNLOCK keyword (and without the GETRETAIN keyword) or the UNLOCK(*ERASE) keyword is used for a record while a subfile is on the screen, subfile records may be returned to the program on a subsequent get-next-changed operation to the subfile even though the work station user did not enter data into the subfile record. It is recommended that you use the UNLOCK(*ERASE *MDTOFF) keyword instead of the UNLOCK keyword (without the GETRETAIN keyword) or the UNLOCK(*ERASE) keyword. If you must use either of the latter, you should make sure that your high-level language program compares for blanks to handle the possibility that an unmodified field containing all blanks is returned to the program.

Read Operation Following a Read Operation: When a read operation is issued, the system reads all the records on a display. However, only one record is passed to the program for each read operation. The system saves all the other records in anticipation of more read operations.

If each read operation refers to a different record on the display, no action is required of the work station user. However, if each read refers not to a different record on the display but the same record and if the RTNDTA keyword is not specified, the work station user must perform an action such as pressing the Enter key or a CF key to start the next read operation because each record entered is passed to the program only once. If the RTNDTA keyword is specified, the work station user does not have to perform any action because the same input buffer that was returned to the program on the previous read operation for the record is returned again.

The system saves the contents of input-capable fields for records that are active on the display. This saved data is passed to the user program and can be altered by:

- Initializing the data with a constant on a write operation. A field can be initialized with the value specified in a DFT keyword.
- Entering data through directly keying the data in or using a light pen to select data. (The MDT for a field can be set on to simulate user input.)
- Entering data from a program on a write operation. This applies to output/input fields (and output-only fields for subfiles).
- Initializing the data with blanks (character fields) or zeros (numeric fields) on an output operation for the same record unless the PUTRETAIN keyword is specified. This applies to input-only fields.

Additional Considerations for CL Programs

The operations just described are available directly to CL programs through CL commands. See the *CL Programmer's Guide* for more information on using display files in CL programs.

****NOWAIT on RCVF and SNDRCVF Commands:*** *NOWAIT allows overlapping of I/O operations and the running program, requests for input from more than one display, and receiving input as it is available. This provides support equivalent to the invite-device function.

On a read operation with the nowait option, the system sends the request to the display and returns to the program. However, the requested record is not available when control returns. The purpose of this operation is to make the device eligible to send input data while the program performs other work.

To retrieve the record, issue a WAIT command. The WAIT command issues a read-from-invited-devices operation. The program waits until data is available from the device or the WAITRCD time elapses. Then, the device name and any input data are passed to the user program. If more than one read-with-nowait operation has been issued (each to a different display) and more than one completes, the WAIT command processes only the first read-with-nowait operation that is completed.

A WAIT command can be issued to process each of the other read-with-nowait operations. They are processed in the order of completion.

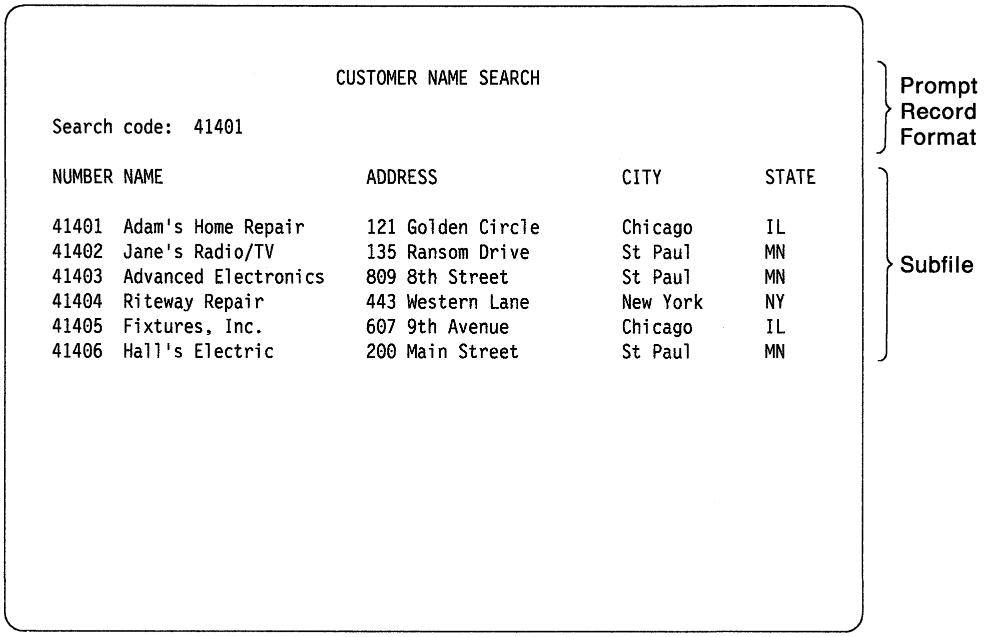
When a record containing the INVITE keyword is sent to the display, the operation is handled as a write-read operation with a nowait option. The INVITE DDS keyword is ignored on the write-read operation.

A write-read operation with a nowait option is the same as a write followed by a read-with-nowait.

ENDRCV Command: The ENDRCV command is used to end a request for input made with the *NOWAIT option. The ENDRCV command ends the input request even if data is available from the device. If data is being sent by the device when the ENDRCV operation is performed, the data is lost. If the device is not invited, the application program is signaled with an error condition.

Subfiles

A subfile is a group of records that have the same record format and are read from and written to a display device in one operation. The following shows an example of a subfile:



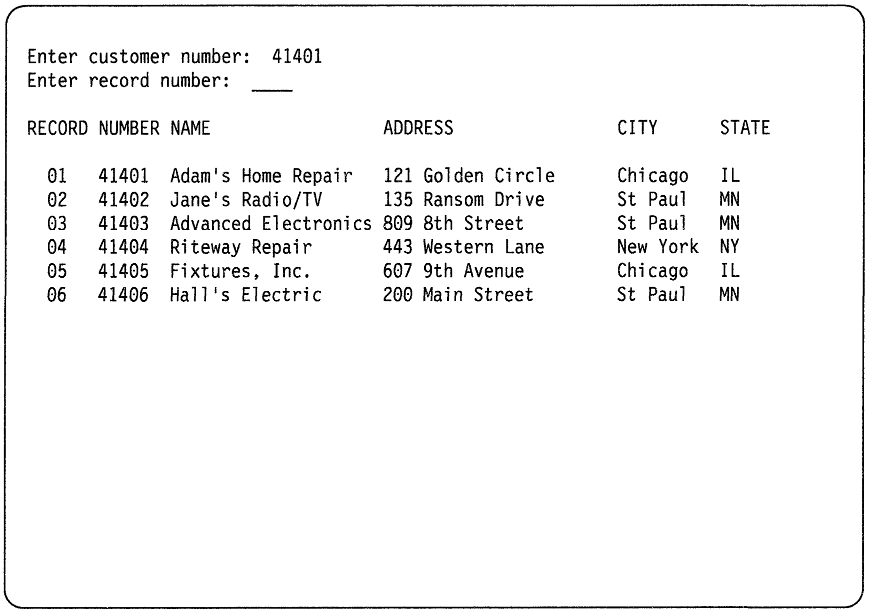
```
                              CUSTOMER NAME SEARCH

Search code:  41401

NUMBER NAME                   ADDRESS             CITY          STATE

41401  Adam's Home Repair     121 Golden Circle   Chicago       IL
41402  Jane's Radio/TV        135 Ransom Drive    St Paul       MN
41403  Advanced Electronics   809 8th Street      St Paul       MN
41404  Riteway Repair         443 Western Lane    New York      NY
41405  Fixtures, Inc.         607 9th Avenue      Chicago       IL
41406  Hall's Electric        200 Main Street     St Paul       MN
```

RSLH172-0

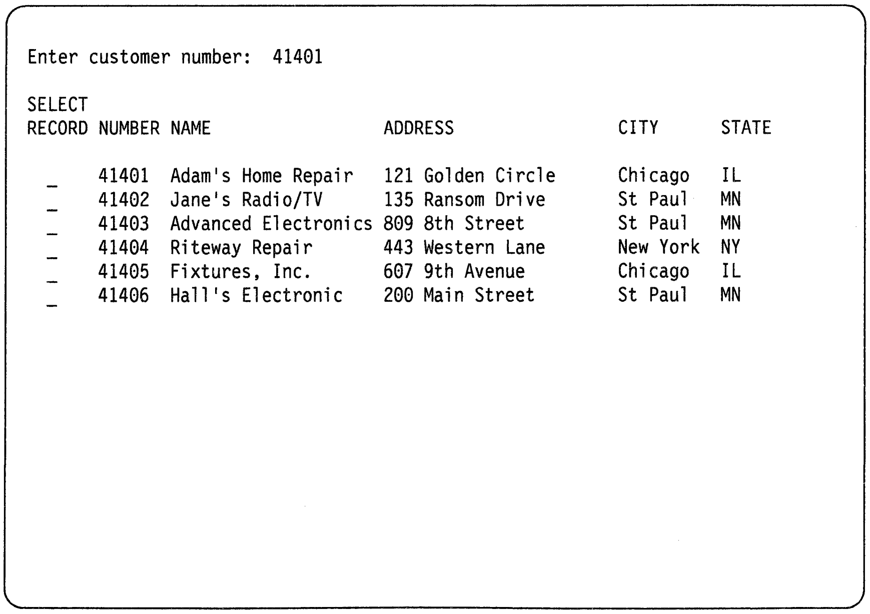
Subfiles are useful when multiple records that are alike must be displayed. You can describe a subfile so that the number of records to be displayed fits on one display or exceeds the number of lines available on the display.

You can use subfiles for:

- Display only, which allows the work station user to review the subfile records on the display, for example, all the line items for a particular order number, or a group of records containing customer names and addresses as shown previously.
- Display with selection, which allows the work station user to request more information about one of the items on the display. In the first example that follows, the work station user can request the records for a particular customer by entering the record number in the record number field. In the second example, the work station user can request the records for a particular customer by placing an X in the select number field.

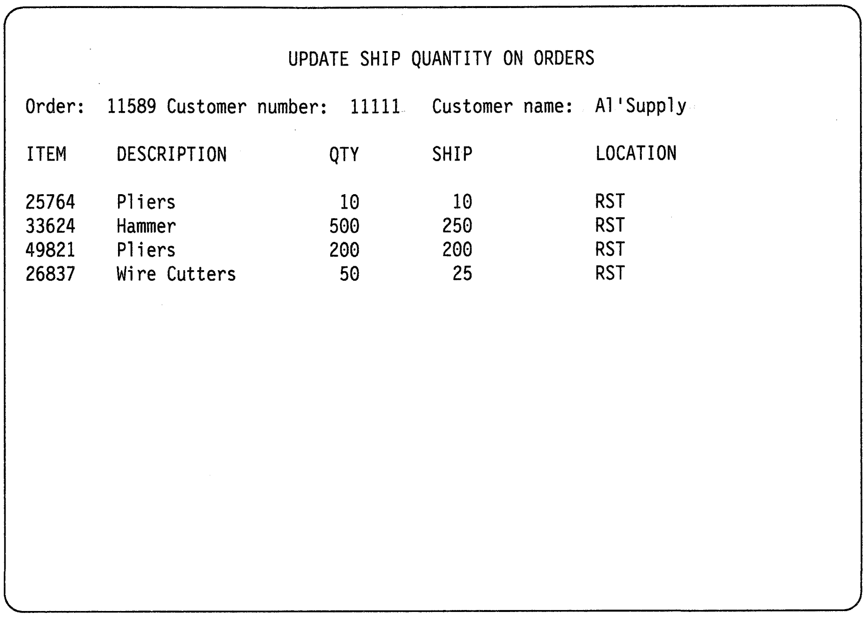


```
Enter customer number:  41401
Enter record number:  ____

RECORD NUMBER NAME                 ADDRESS              CITY      STATE

  01   41401  Adam's Home Repair   121 Golden Circle    Chicago   IL
  02   41402  Jane's Radio/TV      135 Ransom Drive     St Paul   MN
  03   41403  Advanced Electronics 809 8th Street       St Paul   MN
  04   41404  Riteway Repair       443 Western Lane     New York  NY
  05   41405  Fixtures, Inc.       607 9th Avenue       Chicago   IL
  06   41406  Hall's Electric      200 Main Street      St Paul   MN
```



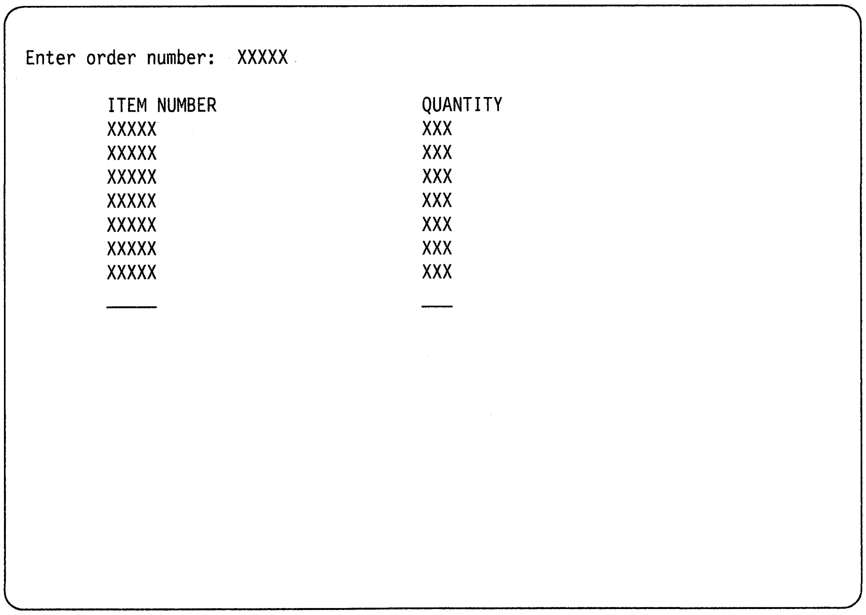
```
Enter customer number:  41401

SELECT
RECORD NUMBER NAME                 ADDRESS              CITY      STATE

  _    41401  Adam's Home Repair   121 Golden Circle    Chicago   IL
  _    41402  Jane's Radio/TV      135 Ransom Drive     St Paul   MN
  _    41403  Advanced Electronics 809 8th Street       St Paul   MN
  _    41404  Riteway Repair       443 Western Lane     New York  NY
  _    41405  Fixtures, Inc.       607 9th Avenue       Chicago   IL
  _    41406  Hall's Electronic    200 Main Street      St Paul   MN
```

- Modification, which allows the work station user to change one or more of the records in the subfile. In the following example, the work station user can change the *QTY* and *SHIP* values.



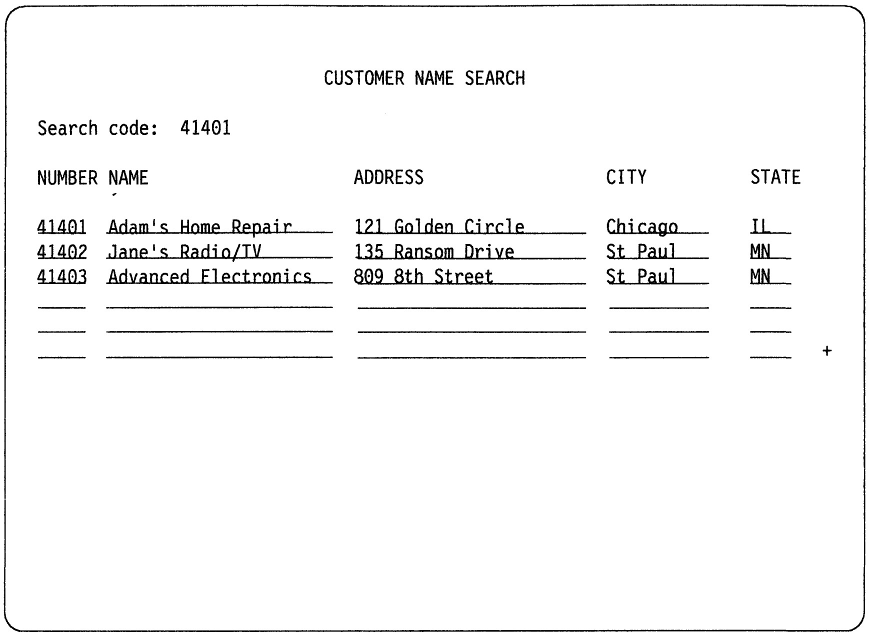
```
                           UPDATE SHIP QUANTITY ON ORDERS

Order:  11589 Customer number:  11111   Customer name:  Al'Supply

ITEM     DESCRIPTION          QTY       SHIP            LOCATION

25764    Pliers                10         10            RST
33624    Hammer               500        250            RST
49821    Pliers               200        200            RST
26837    Wire Cutters          50         25            RST
```

- Input only, without validity checking, which allows the work station user to enter data as fast as possible; or input only, with validity checking, which allows the work station user to enter data that is validity checked by the system or by the program for valid entries. For example:



```
Enter order number:  XXXXX

        ITEM NUMBER                    QUANTITY
        XXXXX                          XXX
        XXXXX                          XXX
        XXXXX                          XXX
        XXXXX                          XXX
        XXXXX                          XXX
        XXXXX                          XXX
        XXXXX                          XXX
        _____                          ___
```

- Combination of tasks, which can, for example, allow the work station user to change data as well as to enter new records. In the following example, the work station user can change existing names and addresses or enter new records.



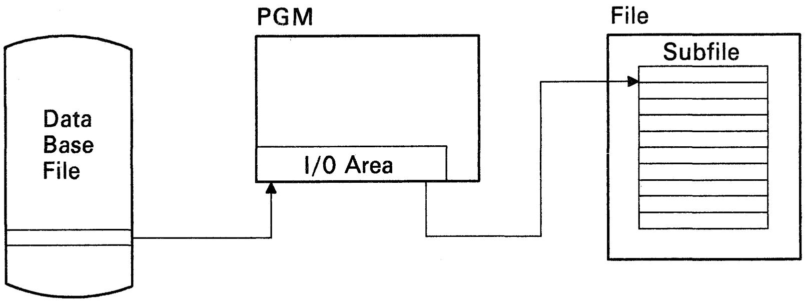
```
                         CUSTOMER NAME SEARCH

Search code:  41401

NUMBER NAME                    ADDRESS                    CITY           STATE

41401  Adam's Home Repair___   121 Golden Circle______    Chicago___     IL__
41402  Jane's Radio/TV______   135 Ransom Drive_______    St Paul___     MN__
41403  Advanced Electronics_   809 8th Street_________    St Paul___     MN__
_____  _____________________   _______________________    __________     ____
_____  _____________________   _______________________    __________     ____
_____  _____________________   _______________________    __________     ____   +
```

RSLH182-0

Overview of Subfile-Program Operations

To use a subfile, you perform the following basic operations in your high-level language program:

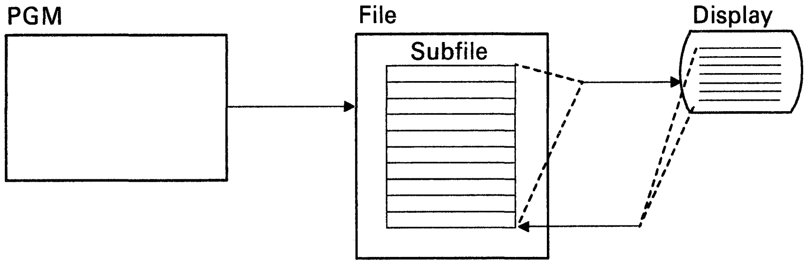
1 Initialize the subfile, perhaps by reading records from a database file and writing them to the subfile. Place the records in the subfile one at a time until the subfile is full or until there are no more records.



P7730127-0

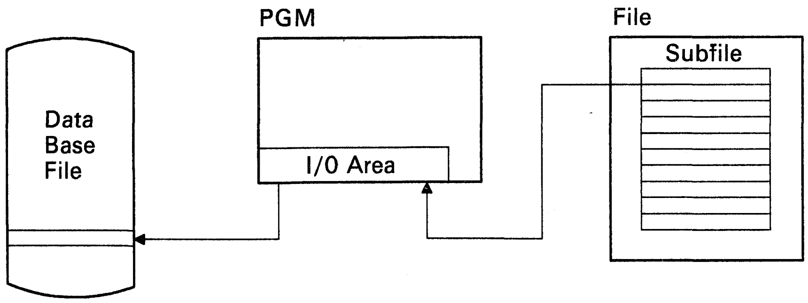
2 Send the subfile to the display in one output operation using the subfile control record format.

3 After the work station user reviews the records, changes them, or enters new records (depending on the function of the subfile), read the subfile control record format.



P7730128-0

4 Process each record in the subfile individually, updating the database file or writing new records to the database file as required. If the function of the subfile is to update records, the program need only process the changed records by using the READC operation in RPG/400 or the Read Subfile Next Modified verb in COBOL/400.



P7730129-0

Notes:

1. The data in an input-only field of a regular display file record format is processed when the Enter key is pressed. It goes to the input buffer and disappears from the screen when that record is redisplayed.

2. A subfile display, unlike a regular record format, may display only a portion of a subfile at a time. The portion that is displayed is called a subfile page. The data entered into an input-only field on a subfile display goes to the subfile when a function key (such as a roll key) is pressed. At this point the field displays a value in the subfile, and what happens when the Enter key is pressed depends on the application code. In a READC operation in RPG, for example, the data is moved from the subfile to the program. It does not remove it from the subfile, and it will continue to be displayed in the input-only field as if the fields were initialized to that value. Otherwise, it would appear that the subfile was empty when the data was actually there.

   If the subfile is processed (for example, by an UPDAT operation in RPG), then the data is removed from the subfile, and the input-only fields are blanked out, reflecting the true condition of empty fields. This should be done after the READC operation moves the data to the program.

Operations on Subfiles

An I/O request by a calling program to a subfile record format either writes a record to a subfile or reads a record from a subfile but never causes actual I/O to the display. To write subfile records to a display, the program must issue a request to the subfile control record format.

The valid requests that can be made to a subfile depend on whether the request is made to the subfile record format or the subfile control record format:

- For a subfile record format, valid requests are put relative, update, get relative, and get-next-changed.
- For a subfile control record format, valid requests are write, write-read, and read.

Table 4-2 shows the I/O requests supported by the system and the equivalent high-level language operations:

Table 4-2. Subfile Operations Supported by the System and the Equivalent High-Level Language Commands

Operation	RPG/400 Operation	COBOL/400 Statement	BASIC Statement
Put Relative	WRITE, output specifications	WRITE SUBFILE	WRITE REC =
Update	UPDAT	REWRITE SUBFILE	REWRITE REC =
Get Relative	CHAIN	READ SUBFILE	READ REC =
Get Next Changed	READC	READ SUBFILE NEXT MODIFIED	READ MODIFIED
Write	WRITE	WRITE	WRITE
Read	READ	READ	READ
Write-Read	EXFMT		

I/O Requests to a Subfile Record

There are two types of records in a subfile:

- **Active record**. A record is active because of one of the following:
  - A write operation added the record to a subfile.
  - The record was initialized as active (SFLINZ keyword).

  An active record can also be a changed record. A record is changed because of one of the following:
  - A write or update operation with the SFLNXTCHG keyword in effect was issued to the record.
  - The work station user changed the record.
- **Inactive record**. A record is inactive because of one of the following:
  - No write operation added the record to a subfile.
  - The record was initialized as inactive (SFLINZ and SFLRNA keywords).

Put-Relative Operation: The put-relative operation adds a record (passed from a program) at a specified location in a subfile. The location must be a valid relative record number in the subfile. The minimum relative record number is always 1. If the subfile size equals the subfile page, the maximum relative record number value is the subfile size value. If the subfile size is greater than the subfile page, the maximum relative record number value is 9999 because the system automatically extends the subfile as required. In addition, the relative record number cannot be the number of an active record already in the subfile. The relative record number is ignored when field selection is specified for the subfile record. When a put-relative operation adds a record at the last record location (the subfile size value) in the subfile, a subfile-full condition occurs (message CPF5003). Both RPG/400 and COBOL/400 have special support for notifying the application program of this condition. See the appropriate high-level language manual.

The contents of input-capable fields without a default value specified are handled as follows:

- Numeric fields are initialized to zeros
- Character fields are initialized to blanks
- Floating point fields are initialized to nulls

Update Operation: The update operation updates an active record that already exists in the subfile. This active record must have been read before the update operation by a get request (either get relative or get-next-changed). No other I/O operations may be performed on the subfile to be updated between the read and the update. In addition, the subfile being updated may not be redisplayed between the read and the update (for example, using subfile roll or SFLDROP processing).

Notes:

1. Some high-level languages do not allow I/O to any format in the display file between the read and the update of a single subfile record in the display file. Refer to the documentation for the high-level language you are using for more information.
2. If field selection is specified for the subfile record, only the fields that were selected when the record was placed in the subfile can be updated. Selecting different fields will cause results that cannot be predicted.

Get-Relative Operation: The get-relative operation reads an active record at a specified location in the subfile. The location must be a valid relative record number in the subfile. The entire record, including response indicators (defined at the file level and on fields in a subfile record), input, output, output/input, and hidden fields, is passed to the program, the relative record number is placed in the input/output feedback area, and the record is no longer identified as a changed record. Response indicators defined at the file level are always returned as off. Response indicators defined on fields in a subfile record, such as the BLANKS or CHANGE keywords, are returned as on or off depending on the information in the field at the time the get operation was done. If the record specified on the get-relative operation is not active, an invalid record number condition occurs (message CPF5020). This condition becomes a record-not-found condition in some high-level languages. See the appropriate high-level language manual.

Note: The get-relative operation and the get-next-changed operation both update the relative record number in the input/output feedback area. Subsequent get-next-changed-record requests retrieve sequentially changed records greater than this relative record number.

Get-Next-Changed Operation: The get-next-changed operation reads the next changed record in the subfile which is greater than the relative record number in the input/output feedback area. If this is the first read operation, the first changed record in the subfile is read. The entire record, including response indicators (defined at the file level and on fields in a subfile record), input, output, output/input, and hidden fields, is passed to the program, the relative record number is placed in the data management feedback area, and the record is reset to a *not changed* record. Response indicators defined at the file level are always returned as off. Response indicators defined on fields in a subfile record, such as the BLANKS or CHANGE keywords, are returned as on or off depending on the information in the fields at the time the get operation was done. If there are no more changed records in the subfile, a message (CPF5037) indicating that the last changed record has already been retrieved, is sent to the program. See the appropriate high-level language manual for a description of how this condition is reported to your program.

If a record retrieved by a get-next-changed operation is updated and the SFLNXTCHG keyword is specified for an updated record, the updated record is set again as a changed record. This allows the program to ensure that the work station user has changed the record. For example, if the program detects an error in a record, it is advantageous to require the work station user to correct the error. The use of the SFLNXTCHG keyword allows the program to read that record again on a get-next-changed operation so it can continue to reject the record until the error has been corrected. The next get-next-changed operation does not retrieve this updated record. The record cannot be retrieved again with a get-next-changed operation until all the changed records following it in the subfile have been processed. This is because the changed records are accessed sequentially and the sequence does not start at the beginning until after the message indicating that there are no more changed records in the subfile has been sent to the program. A get-next-changed operation following this message gets the first changed record in the subfile. Because no I/O operation has been issued to the display, any changed record would be a record that was processed using the SFLNXTCHG keyword.

I/O Requests to a Subfile Control Record

Write Operation: The only way you can display subfile records is by issuing a write operation (or write-read operation) to the subfile control record format. You can control the write operation using the following DDS keywords:

- SFLDSP: The subfile is displayed.
- SFLDSPCTL: The subfile control record is displayed.
- SFLCLR: The subfile is cleared of active records.
- SFLDLT: The subfile is deleted.
- SFLINZ: The subfile is initialized with active records, or if the SFLRNA keyword is specified, with inactive records. When the subfile is initialized, all option indicators in the subfile record are assumed to be off.
- SFLEND: The work station user is notified when the last available record is displayed.
- SFLRCDNBR: The specified subfile page is displayed.

Note: These keywords are described under "Describing Subfiles Using DDS" on page 4-42.

Read Operation: A read operation must be issued to a displayed record format in order for the subfile records on the display to be placed into the subfile for processing by the program. For a description of how response indicators are set, see "Indicators and Condition Names" on page 4-77. The subfile records from the display are placed in their corresponding record positions in the subfile. See "Display File Operations" on page 4-20 for more information on the read operation.

Write-Read Operation: The write-read operation is a single operation that combines the write and read operations and is more efficient than a single write operation followed by a single read operation.

Controlling the Appearance of Subfiles

Records in a subfile can be displayed either vertically or horizontally. In a vertically displayed subfile, a record is displayed on one or more lines, with each record beginning a new line (see Figure 4-2). In a horizontally displayed subfile, a record is complete on one line, and more than one record is displayed on a line (see Figure 4-3 on page 4-40). You can specify that a subfile is to be displayed horizontally by using the SFLLIN keyword to define the number of spaces between each subfile record on a display line. Figure 4-4 on page 4-40 shows an example of a vertical subfile and a horizontal subfile being displayed at the same time.

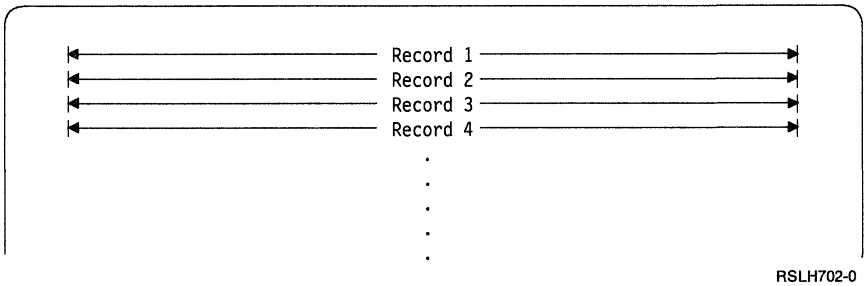


Figure 4-2. Vertically Displayed Subfile

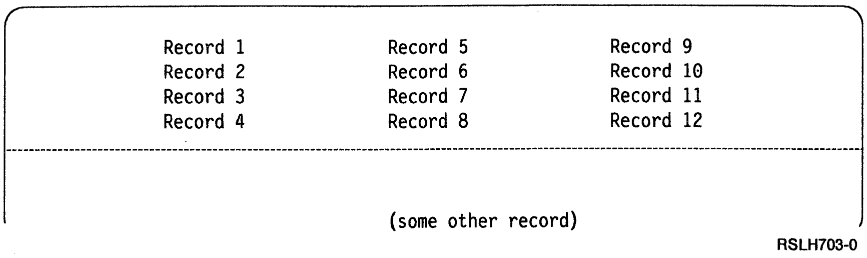


Figure 4-3. Horizontally Displayed Subfile

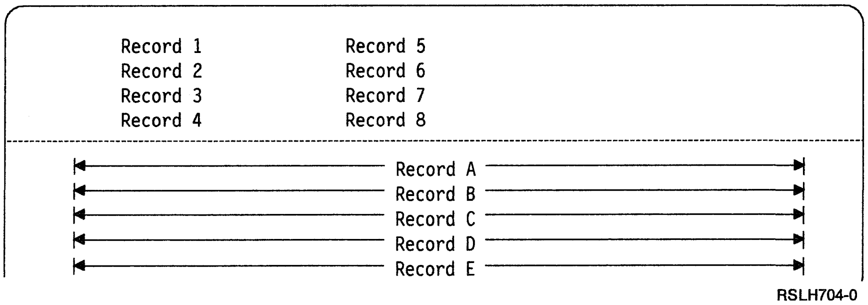


Figure 4-4. Horizontally and Vertically Displayed Subfiles Displayed at the Same Time

If a subfile is larger than the space allowed for the subfile on the screen, the work station user can roll the display from one group of records in the subfile to another. Each group of records displayed at the same time is called a page. When you create a display file with a subfile, you must specify the size of the page for a subfile by specifying the number of records in the page (SFLPAG keyword). Usually page size is based on the number of lines available on the display. You must also specify the size of the subfile by specifying the number of records in the subfile (SFLSIZ keyword).

Page size and subfile size can be the same; that is, all records in the subfile fit on one page. When page size equals subfile size, variable-length subfile records are supported. One record can take up only a single line while another record can take up more than one display line. Each record is placed in the first record position available in the subfile; this position is always a new line. In addition, the SFLDROP and SFLROLVAL keywords are ignored by device support when page size equals subfile size.

For more information on page size and subfile size, see "Considerations When Using Subfiles" on page 4-50.

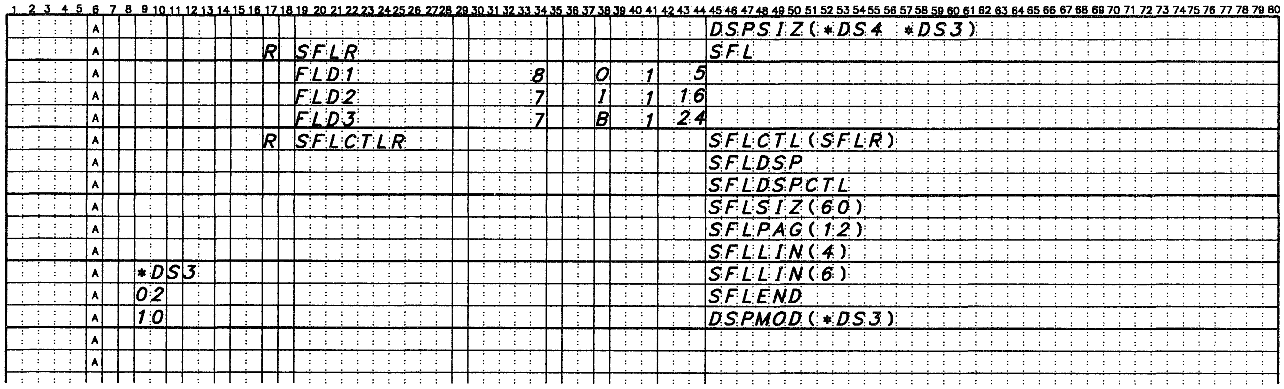
If records are to be displayed horizontally, the number of records to be displayed in a subfile (SFLPAG keyword) is adjusted so that the last line on the screen can be used to display a full line of records. For example, if the number of spaces between each record on a line (SFLLIN keyword) is specified such that six records fit on a line and 20 is specified for the page size (SFLPAG keyword), 20 is changed to 24, which is the nearest multiple of six. The number of records in the subfile (SFLSIZ keyword) is incremented by the same amount.

Note: For the initial display of a subfile, the more records placed in a subfile before it is displayed, the slower the response time.

Displaying Horizontal Subfiles with DSPMOD

When changing display modes, only the display is cleared. The data is not cleared from the subfile. SFLDSP or SFLDSPCTL must be in effect for DSPMOD to be active in the control record.

The following example shows how to specify DSPMOD with subfiles:

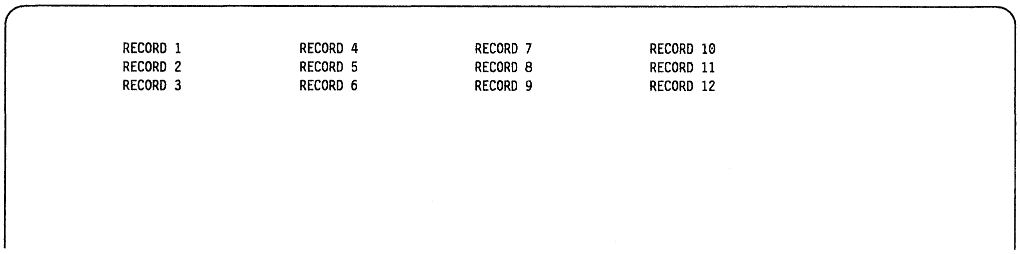


```
     A                                      DSPSIZ(*DS4 *DS3)
     A          R SFLR                      SFL
     A            FLD1           8   O  1  5
     A            FLD2           7   I  1 16
     A            FLD3           7   B  1 24
     A          R SFLCTLR                   SFLCTL(SFLR)
     A                                      SFLDSP
     A                                      SFLDSPCTL
     A                                      SFLSIZ(60)
     A                                      SFLPAG(12)
     A                                      SFLLIN(4)
     A  *DS3                                SFLLIN(6)
     A  02                                  SFLEND
     A  10                                  DSPMOD(*DS3)
     A
     A
```

RSLH150-2

In the example above, if the user's program turns indicator 10 off and issues a write-read operation to the subfile control record format (SFLCTLR), the subfile is displayed as follows:

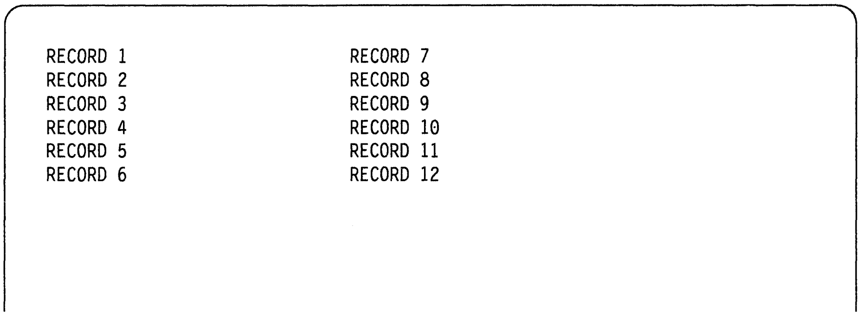
- In 27 by 132 (*DS4) mode because indicator 10, for the DSPMOD keyword, is off.
- Horizontally because SFLLIN is specified. The SFLLIN value indicates the number of bytes between records. Because each record is 30 bytes long and the space between each record is 4 bytes long, four records can be displayed on one horizontal line, (4 x 30) + (3 x 4) = 132 bytes. The subfile is displayed on three lines because SFLPAG(12) is specified. The following example shows the subfile displayed in *DS4 mode.



```
RECORD 1          RECORD 4          RECORD 7          RECORD 10
RECORD 2          RECORD 5          RECORD 8          RECORD 11
RECORD 3          RECORD 6          RECORD 9          RECORD 12
```

If the user presses the Enter key, control is returned to the user's program. If the user's program turns on indicator 10 and then issues another write-read operation to the subfile control record format (SFLCTLR), the subfile is displayed as follows:

- In 24 by 80 (*DS3) mode because indicator 10, for the DSPMOD keyword, is on.
- Horizontally because SFLLIN is specified for the *DS3 mode. If SFLLIN was not specified for the *DS3 mode, the subfile would have been displayed vertically. If the SFLLIN keyword is to be used for more than one screen size, a screen size condition name for each secondary screen size is required. Because each record is 30 bytes long and the space between each record is 6 bytes long, two records can be displayed on one horizontal line, (2 x 30) + 6 = 66 bytes). The subfile is displayed on six lines because SFLPAG(12) is specified. To ensure other records are not erased, the SFLPAG may need to be specified for the secondary screen size. The following example shows the subfile displayed in *DS3 mode.



```
RECORD 1                     RECORD 7
RECORD 2                     RECORD 8
RECORD 3                     RECORD 9
RECORD 4                     RECORD 10
RECORD 5                     RECORD 11
RECORD 6                     RECORD 12
```

Describing Subfiles Using DDS

Each subfile you describe using DDS requires a subfile record format and a corresponding subfile control record format. The subfile record format defines the fields in one row of the subfile. The subfile control record format can contain heading information, and it controls unique subfile functions such as size, initialization, and clearing. The DDS for the subfile record format must precede the DDS for the subfile control record format.

The high-level language program uses the subfile record format to perform input (read a subfile), output (write new records to a subfile), and update operations to the subfile. Operations to the subfile record format are performed between the subfile and the high-level language program; the display is not changed on operations to a subfile record format.

The high-level language program performs operations on the subfile control record format to write the subfile to the display and to read the subfile from the display. See also "I/O Requests to a Subfile Control Record" on page 4-39.

There are two types of records in a subfile:

- Active record. A record is active because of one of the following:
  - A write operation added the record to a subfile.
  - The record was initialized as active (SFLINZ keyword).

  An active record can also be a changed record. A record is changed because of one of the following:

  - A write or update operation with the SFLNXTCHG keyword in effect was issued to the record.
  - The work station user changed the record.
- Inactive record. A record is inactive because of one of the following:
  - No write operation added the record to a subfile.
  - The record was initialized as inactive (SFLINZ and SFLRNA keywords).

The SFL keyword is required on the subfile record format. The following DDS keywords are required on a subfile control record format:

- SFLCTL, which identifies the subfile control record format for the subfile record format that immediately precedes it
- SFLSIZ, which specifies the size of the subfile
- SFLPAG, which specifies the size of the subfile page
- SFLDSP, which specifies when to begin displaying records in a subfile (if the SFLDSP keyword is in effect and the subfile does not exist, an error message, CPF5013, is issued to the high-level language program)

You can also specify the following in the DDS for subfiles:

- That the subfile is to be cleared of all records before new records are written (SFLCLR keyword). The subfile is not erased from the display, however, until the SFLDSP keyword is in effect on the subfile control record. If the SFLCLR keyword is specified for a subfile with no records, it is ignored.

- That the subfile is to be deleted so that another subfile can be used or so that processing of the device file can continue with no subfile used (SFLDLT keyword). Normally, subfiles should not be deleted by the program. When the file containing the subfile is closed, the subfile is deleted automatically by the system. However, if the file is shared and is still open by another program, the subfile is not deleted, and you must delete it in your program. You should only delete a subfile if the maximum number of subfiles are already being used and you need to use another one. The SFLDLT keyword is ignored if the subfile does not exist.

- That a command key be used to fold or truncate records in a subfile (SFLDROP keyword). The initial display of the records is automatically truncated. Then, the work station user can press the command key to display the folded version of the subfile record. If page size equals subfile size or the subfile fits on one display line, the SFLDROP keyword is ignored.

- When to display a subfile control record (SFLDSPCTL keyword). The SFLDSP and SFLDSPCTL keywords are the only keywords that cause the contents of the display to change. The SFLDSPCTL keyword must be in effect if an input operation is done to retrieve the status of a CF or CA key even if no fields are displayed.

- That the Enter key be used as the Roll Up key and that a command key be used to return to the high-level language program (SFLENTER keyword).

- That a + (plus sign) be displayed in the lower corner to the extreme right of the subfile display area (page) when there are more records than fit on the display and that the + be replaced by a blank when the last record is displayed (SFLEND keyword). An option indicator must be specified with the SFLEND keyword.

- The number of spaces between each record on a line when more than one record is displayed on a line (SFLLIN keyword). This is used for a horizontally displayed subfile. If the display file supports more than one screen size and the SFLLIN keyword is to apply to the secondary screen size in addition to the default (or primary) screen size, screen size condition names must be specified.

- That you want to roll by a specified number of records instead of by page (SFLROLVAL keyword).

- That a record is to be considered a changed record and returned to the program when a get next changed operation is performed if the record was changed by the work station user (SFLNXTCHG keyword) or not.

- That all records in a subfile are to be initialized by the field descriptions in the subfile record format in the device file (SFLINZ keyword). When the SFLINZ keyword is in effect on an output operation to the subfile control record (SFLCTL), the system assumes that all option indicators on the subfile record are off; therefore, only those option indicators that are preceded by N are in effect. The subfile records are displayed if SFLDSP is in effect on an output operation. When the SFLINZ keyword is in effect on an output operation, the contents of input-capable fields without a default value are handled as follows:

  - Numeric fields are initialized to zeros.
  - Character fields are initialized to blanks.
  - Floating point fields are initialized to nulls.

- That when a subfile is initialized (SFLINZ keyword), it is to be initialized with no active records (SFLRNA keyword) although the subfile is active. A record becomes active when one of the following happens:

  - An output operation is issued to the subfile for a specific record. The record is not considered changed unless the SFLNXTCHG keyword is used.
  - A user enters data into a displayed record. The record is considered active and changed.

  The records are displayed if the SFLDSP keyword is in effect. If default values were specified for fields in the records, they are included in the display.

- That a page of a subfile is to be selected for displaying by a record number (SFLRCDNBR keyword).

- That a message is to be written to the message line on the display when your program does an output operation to the subfile control record (SFLMSG and SFLMSGID keywords).

- That the subfile is to contain messages from a program message queue (SFLMSGKEY, SFLMSGRCD, and SFLPGMQ keywords).

The DDS keywords can be specified in any order; however, the subfile record format (SFL) must precede the subfile control record format (SFLCTL).

You can use option indicators to condition many of the DDS subfile keywords. See the *DDS Reference* for the keywords that require or allow option indicators.

You can specify a maximum of 512 subfiles in a display file, since the maximum number of record formats allowed in a display file is 1024. Not more than 12 subfiles can be active at the same time to the same device. One or more active subfiles can be displayed at the same time on the device. A subfile must contain at least one displayable field, and the subfile record format must not overlap the subfile control record format. If these records overlap, the display file cannot be created.

All named fields in a subfile record are returned to the program; whereas for records not in a subfile, only named input-capable fields (input, output/input, and hidden fields) are returned to the program.

If any input data validity checking is specified for the subfile record, the validity checking is performed before any roll function is performed. If the data fails validity checking, the roll function is not performed.

When the relative record number of the record written to the subfile equals the subfile size, the system sends the program a CPF5003 message indicating that the subfile is full. (Not all records need to be active; that is, this message is sent even if the only record written to the subfile was the last record in the subfile.) If the subfile size does not equal the page size and the program then writes more records to the subfile, the system automatically extends the subfile as additional records are added. The program is not notified that the subfile has been extended. (A subfile cannot be extended past 9999 records.) Also, if the subfile size equals the page size, the program is not notified that the subfile is full unless the last record written to the subfile occupies the last line available on the subfile display area.

Processing of an extended subfile is less efficient because the extended space is not connected with the subfile. You can avoid extension by specifying a larger subfile size, but you will be wasting space if the extended space is used very seldom or never.

Figure 4-5 illustrates the order in which some of the DDS keywords used for subfile control are processed at run time:

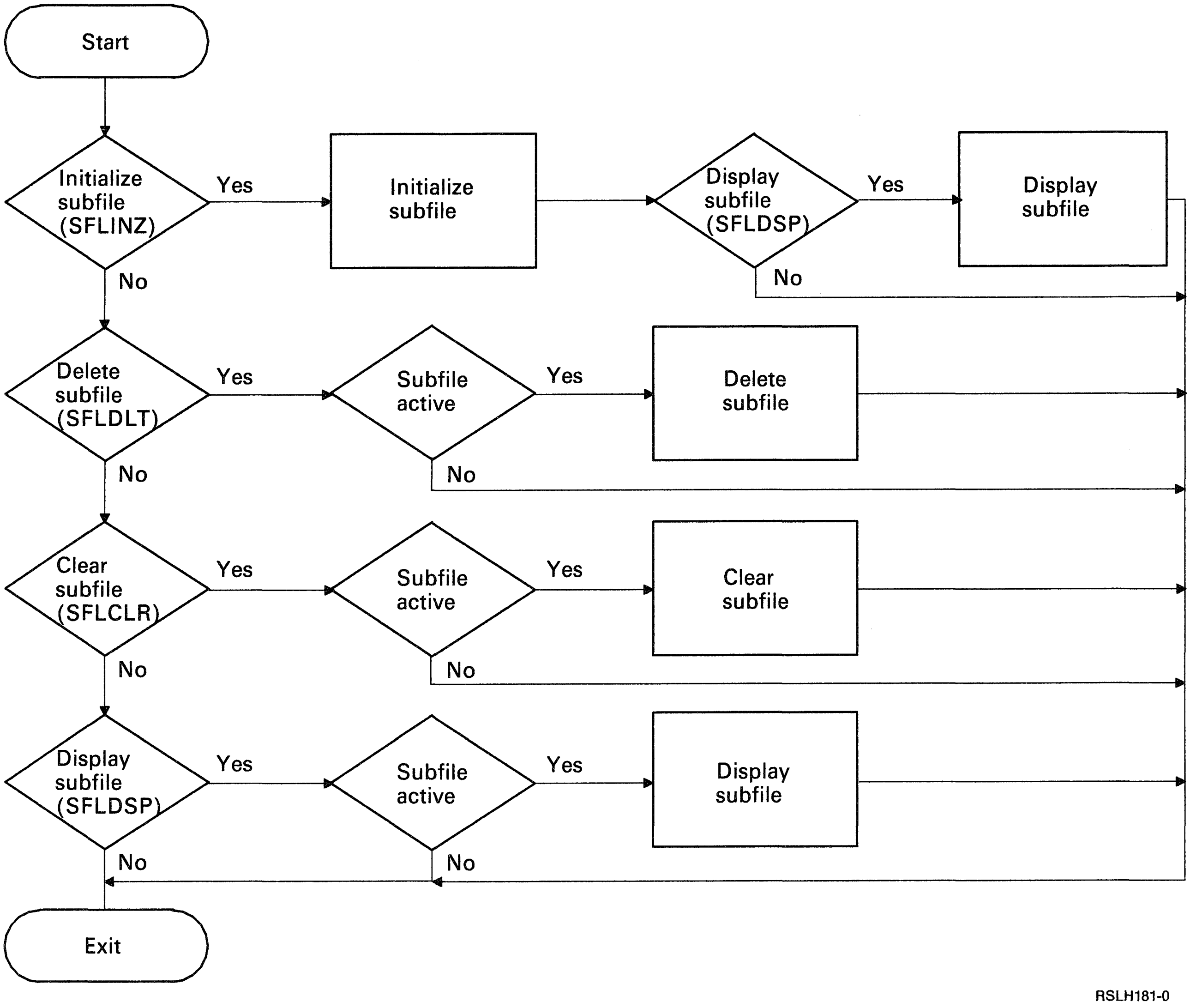
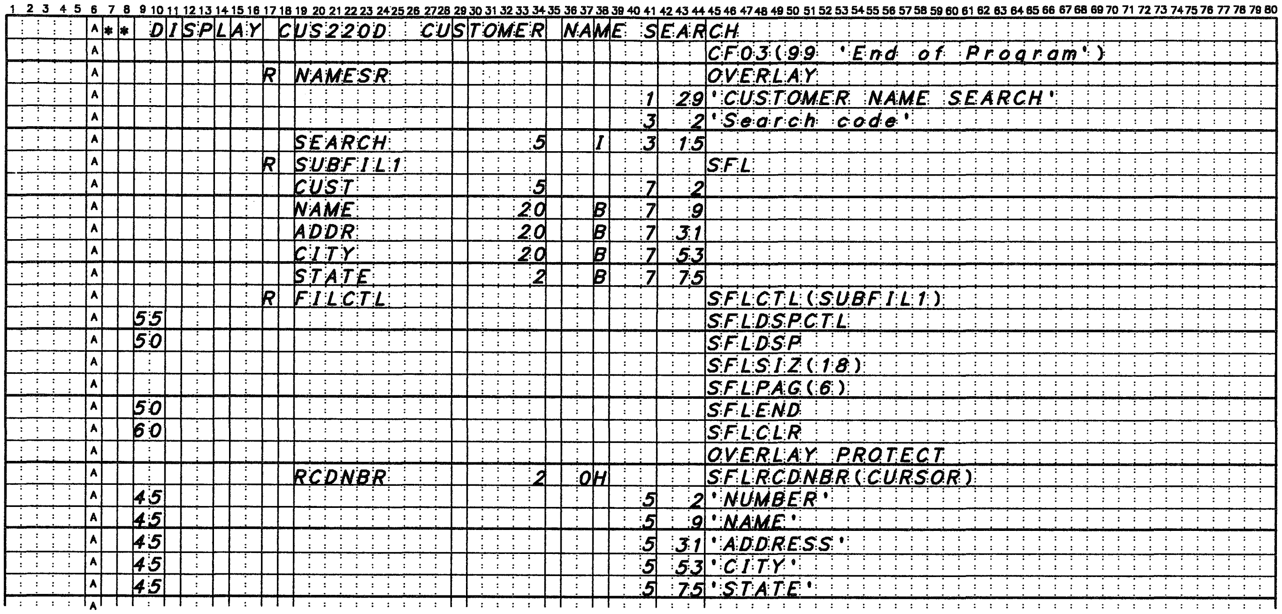


Figure 4-5. DDS Keyword Processing Order for Subfile Control

Example of Subfile DDS and Program Logic

The following shows the DDS that describes the customer name search subfile shown earlier in this section. The DDS is followed by a description of the logic a program would use to process this subfile.



```
     A** DISPLAY CUS220D  CUSTOMER NAME SEARCH
     A                                      CF03(99 'End of Program')
     A          R NAMESR                    OVERLAY
     A                                  1 29'CUSTOMER NAME SEARCH'
     A                                  3  2'Search code'
     A            SEARCH         5   I  3 15
     A          R SUBFIL1                   SFL
     A            CUST           5      7  2
     A            NAME          20   B  7  9
     A            ADDR          20   B  7 31
     A            CITY          20   B  7 53
     A            STATE          2   B  7 75
     A          R FILCTL                    SFLCTL(SUBFIL1)
     A  55                                  SFLDSPCTL
     A  50                                  SFLDSP
     A                                      SFLSIZ(18)
     A                                      SFLPAG(6)
     A  50                                  SFLEND
     A  60                                  SFLCLR
     A                                      OVERLAY PROTECT
     A            RCDNBR         2  0H      SFLRCDNBR(CURSOR)
     A  45                              5  2'NUMBER'
     A  45                              5  9'NAME'
     A  45                              5 31'ADDRESS'
     A  45                              5 53'CITY'
     A  45                              5 75'STATE'
     A
```

RSLH151-3

The following is an example of the logic a user program would use to process the subfile just shown. A write-read operation is a combined input and output operation. A read is an input operation. A write is an output operation. See the appropriate high-level language manual for the operations that can be performed in the high-level language program. See "Display File Operations" on page 4-20 and "Operations on Subfiles" on page 4-36 for a description of the operations mentioned in this example.

User Program

1. Opens a file and issues a write-read operation to the NAMESR record format to prompt for a search code.

Work Station User

2. Enters a zip code in the search code field. The program uses the search code field as a key field to find the first database record in the file with that key field. The program will build the subfile using that record as the first record in the subfile.

User Program

3. Obtains records (read operation) from the database file and places (write operation to SUBFIL1) them in the subfile one record at a time until the subfile is full or there are no more records to place in the subfile.
4. When all records are in the subfile, issues a write-read operation to the subfile control record format (FILCTL) with the following:
    a. The indicator for the SFLEND keyword is on, so a + (plus sign) is displayed in the lower corner to the extreme right of the screen when there are more records than fit on one subfile page. When the last subfile page is displayed, the system replaces the + with a blank.
    b. The indicator for the SFLDSP keyword is on, so the first subfile page is displayed.
    c. The indicator for the SFLDSPCTL keyword is on, so the subfile control record is displayed.
    d. The SFLRCDNBR(CURSOR) keyword is specified for a field. The program placed a value of 1 in this field, so the subfile page that contains relative record number 1 is displayed first and the cursor is positioned at the first input field in that record. (If no input field exists, the cursor is positioned at the first selected output field or constant in that record.)
    e. The constant field (heading line) indicators are on, so the constants in the subfile control record are displayed.
    f. The OVERLAY and PROTECT keywords are in effect in the subfile control record so that the prompt (NAMESR) can remain on the display without being changed.

Work Station User

5. Updates displayed records, using the roll keys to display different subfile records as needed. Presses the Enter key after completing all updates to the subfile.

User Program

6. Completes the input portion of the write-read operation to the subfile control record format. In this example, the subfile control record does not contain any input fields. The input portion of the write-read operation allows the work station user to enter data into the subfile.

7. Issues the get-next-changed operation to the subfile record to process the first subfile record changed by the user.

8. Uses each changed record to update the corresponding database file record.

9. Repeats steps 7 and 8 until a no-more-modified-records condition exists for step 7. When this condition is detected, step 10 is performed.

10. Issues a write-read operation to the prompt (NAMESR) to determine if the program should end or display another group of database records. The OVERLAY keyword is specified, so that the current display contents for the subfile are left unchanged. If the work station user presses the CF1 key, which turns on response indicator 99, the program will close the display file and end. If the work station user enters another search code for another group of records to be displayed, step 11 is performed.

11. Issues a write operation to the subfile control record (FILCTL) with the following:

    a. The indicator for the SFLCLR keyword is on, so the subfile is cleared of all records (the display is unchanged).
    b. The indicator for the SFLDSP keyword is off, so the contents of the display remains unchanged.
    c. The indicator for the SFLDSPCTL keyword is off, so the subfile control record is not displayed again.

12. Repeats steps 3 through 10. The subfile was cleared (SFLCLR keyword) in step 11 so that new records can be placed in the subfile. The write operation to the subfile control record in step 4 has the constant field indicator off so that heading information is not sent to the display again. As long as the subfile control record remains on the display (no intervening write operations without the OVERLAY keyword in effect have been performed), the fields do not have to be sent to the display again.

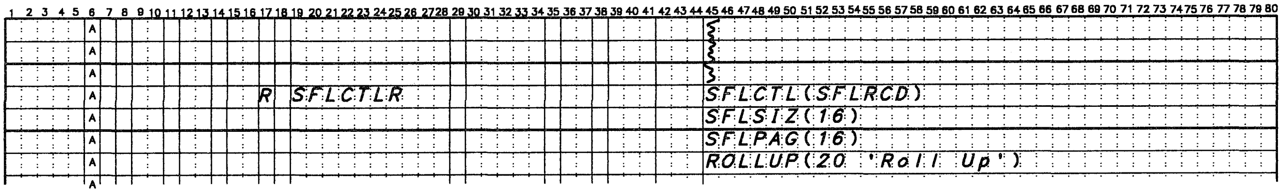
Considerations When Using Subfiles

Subfile Size Equals Page Size

You must specify the size of the subfile and the number of subfile records to be displayed at one time with the SFLSIZ and SFLPAG keywords. The use of subfile size equals page size is recommended when the number of subfile records to be displayed will fit on one page or when the number of records to be placed in the subfile is unknown and large. It is not an efficient use of resources to retrieve many database records to fill a large subfile if the work station user normally finds the needed information on the first page.

When subfile size equals page size, the system does not automatically support the use of the Roll Up and Roll Down keys. If you want the work station user to roll through the subfile using these keys, you must specify the ROLLUP or ROLLDOWN keyword in the subfile control record, and your program must handle the roll up or roll down function.

For example, if a subfile is used to allow the work station user to search through a long list, you can specify SFLSIZ equals SFLPAG and ROLLUP on the subfile control record:



```
     A                                      }
     A                                      }
     A                                      }
     A          R SFLCTLR                   SFLCTL(SFLRCD)
     A                                      SFLSIZ(16)
     A                                      SFLPAG(16)
     A                                      ROLLUP(20 'Roll Up')
     A
```

RSLH153-1

When the work station user presses the Roll Up key, indicator 20 is set on and control returns to the program. In your program, you would:

- Clear the subfile (a write operation to the subfile control record with the SFLCLR keyword in effect)
- Use the indicator to control a return to the logic that fills the subfile with another page of records
- Display the new subfile page

You could also allow the work station user to press a CF key to return to the start of the search. When the work station user presses the CF key, the associated indicator is set on and control returns to the program. In your program, you would:

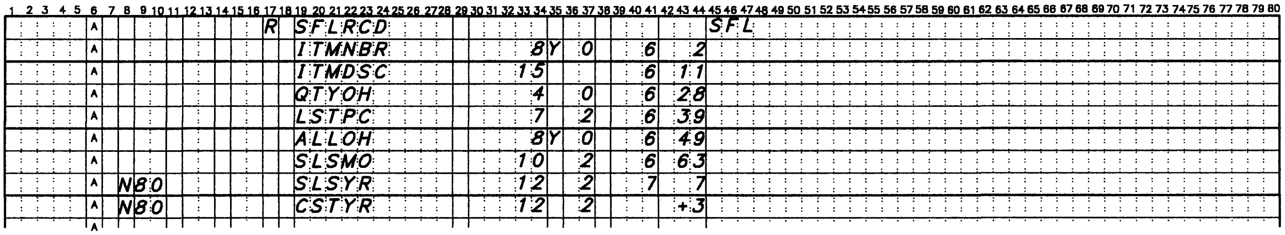
- Clear the subfile.
- Use the indicator to control a return to the logic that built the first subfile page based on the search code entered. (The program needs to keep the original search code in order to do this.)

Using the ROLLDOWN keyword when subfile size equals page size requires more lines of code in the program because the program must keep track of the record position in the subfile and in the database file.

When the subfile size equals page size, you can use field selection and variable-length records in the subfile. If you use field selection, consider the following:

- If the fields are selected through the use of option indicators, the relative record number is ignored and each record is placed in the first available record position in the subfile.
- If a record is being updated, the field selection that does not match that on the original output is ignored. For example, assume that FIELD1 and FIELD2 are selected when the record is placed in the subfile. If the update selects FIELD2 and FIELD3, fields would overlay the original FIELD1 and FIELD2 fields, and the results could not be predicted.
- If field selection is specified on the subfile record, the number of records that can be displayed on the screen depends on the number of fields selected. When field selection is specified, the SFLPAG(value) keyword specifies the number of screen lines available to display the subfile record. In other cases, the SFLPAG(value) keyword specifies the number of subfile records that can be displayed at one time.
- The SFLDROP and SFLROLVAL keywords are ignored.

When variable-length records are used, each record in the subfile is displayed beginning on the first available line on the page. If you use field selection for variable-length records, each record can take up a different number of lines on the display. Therefore, the number of records that actually fit in the subfile depends on the field selection of each record written to the subfile. The following shows an example of the DDS for a variable-length record:



```
     A          R SFLRCD                    SFL
     A            ITMNBR         8Y 0   6  2
     A            ITMDSC        15      6 11
     A            QTYOH          4  0   6 28
     A            LSTPC          7  2   6 39
     A            ALLOH          8Y 0   6 49
     A            SLSMO         10  2   6 63
     A N80        SLSYR         12  2   7  7
     A N80        CSTYR         12  2     +3
     A
```

RSLH154-0

When indicator 80 is on, each record in the subfile fits on one line. However, when indicator 80 is off, each record uses two lines on the display.

Another typical use of variable-length records is where two or more entirely different formats are used to make up one format. In this case, each field would be separately conditioned by option indicators so that one record format might use multiple lines while another format uses only one line.

Subfile Size Not Equal to Page Size

Subfile size not equal to page size should be used when a finite number of records can be placed in the subfile and that number is small (for example, 50). The SFLSIZE keyword specifies the subfile size. Display data management allocates space to contain the subfile records based on the value specified for SFLSIZ. You should specify a value equal to the number of records that you normally have in the subfile. If your program places a record with a relative record number larger than the SFLSIZ value into the subfile, display data management extends the subfile to contain it (up to a maximum of 9999 records).

When the subfile size is not equal to the page size, the use of the Roll Up and Roll Down keys is automatically supported.

To inform the work station user that there are more records in the subfile, use the SFLEND keyword on the subfile control record. When SFLEND is in effect (for example, the option indicator is on), a + (plus sign) is placed in the lower position to the extreme right of the screen on each page except the last page. On the last subfile page, the + is replaced with a blank.

When the subfile size is not equal to page size, you can use the SFLROLVAL keyword to allow the work station user to enter a value to specify how many records should be rolled up or down when the appropriate key is pressed. If the SFLROLVAL keyword is not used, the subfile is rolled by the SFLPAG value except for subfiles using SFLDROP. If the SFLDROP keyword is used, more records are displayed than the SFLPAG value when records are displayed in the truncated format. For truncated records, the display rolls by the number of records displayed in the truncated format. When the SFLROLVAL keyword is used and the Roll Up key is pressed, the uppermost record number in the displayed subfile is added to the roll value to determine the new uppermost record number. If this value is greater than the last record in the subfile, the last full page of records is displayed. If the Roll Up key is pressed when the last subfile page is displayed and the roll value is not less than the page size value, an error message is issued. If the roll value is less than the page size value, the roll function is performed.

Variable-length records and field selection cannot be used when the subfile size is not equal to the page size.

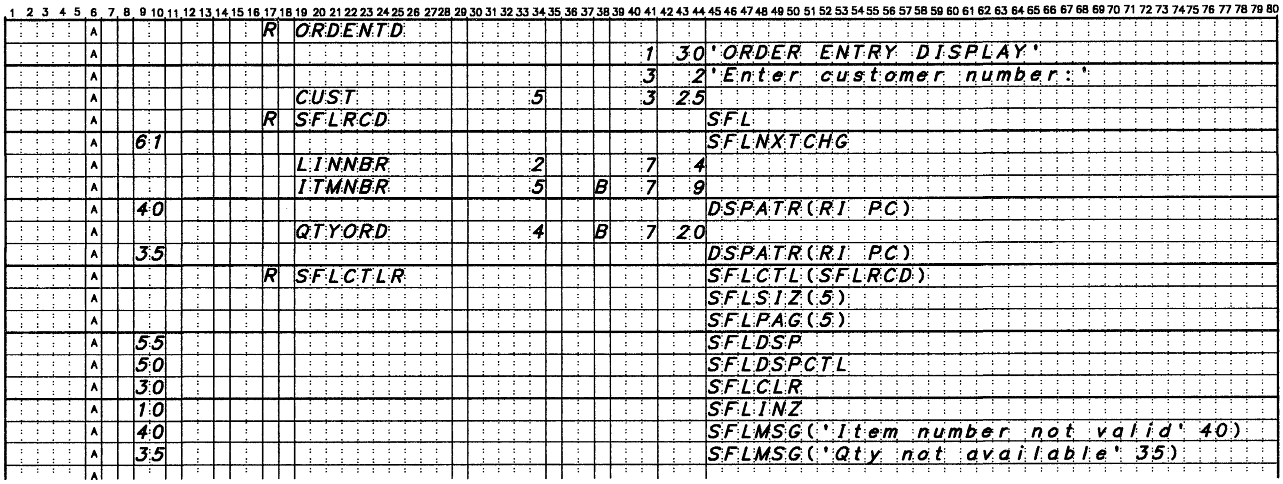
A technique to use to improve performance when you are using a multiple page subfile is to write only one page of subfile records at a time but use the OS/400 support to roll through the subfile. To do this, you need to define the ROLLUP keyword in DDS with a response indicator and also use the SFLRCDNBR keyword. In your program, you would write the records needed to fill one subfile page and then display that page. When the work station user wants to see more records, he presses the Roll Up key. The program then writes another page of records to the subfile, places the relative record number of a record from the second page into the SFLRCDNBR field, and displays the record.

The second page of subfile records is now displayed, and if the work station user presses the Roll Down key, the roll down is handled by the system. If the work station user presses the Roll Up key while the first page is displayed, the system will also handle the roll up. The program is notified only when the work station user attempts to roll up beyond the records currently in the subfile. The program would then handle any additional roll up requests in the same manner as for the second page. When you use this technique, the subfile appears to be more than one page because of the use of the roll keys. Yet, you can maintain good response time because the program only fills one subfile page before writing it to the display.

Validity Checking

In addition to the DDS validity checking keywords (CHECK, COMP/CMP, RANGE, and VALUES), you can also do validity checking on subfile data in your program and require the work station user to correct the error.

For example, assume that you are using a subfile for an order entry program and you want to check the item number field to be sure it is a valid order number. You also want to check the quantity ordered field to ensure there are enough items on hand to fill the order. To do this, you can use the SFLNXTCHG keyword on the subfile record (SFL) to allow your program to diagnose the errors and require the work station user to correct them. The following DDS shows an example of using the SFLNXTCHG keyword:



```
     A          R ORDENTD
     A                                  1 30'ORDER ENTRY DISPLAY'
     A                                  3  2'Enter customer number:'
     A            CUST           5      3 25
     A          R SFLRCD                    SFL
     A  61                                  SFLNXTCHG
     A            LINNBR         2      7  4
     A            ITMNBR         5   B  7  9
     A  40                                  DSPATR(RI PC)
     A            QTYORD         4   B  7 20
     A  35                                  DSPATR(RI PC)
     A          R SFLCTLR                   SFLCTL(SFLRCD)
     A                                      SFLSIZ(5)
     A                                      SFLPAG(5)
     A  55                                  SFLDSP
     A  50                                  SFLDSPCTL
     A  30                                  SFLCLR
     A  10                                  SFLINZ
     A  40                                  SFLMSG('Item number not valid' 40)
     A  35                                  SFLMSG('Qty not available' 35)
     A
```

RSLH155-1

When the program detects an error, it sets on the indicator that conditions the SFLNXTCHG keyword and issues a write operation to the subfile control record with the SFLDSP keyword in effect. For more information on the SFLNXTCHG keyword, refer to the *DDS Reference*. The field in error is displayed in reverse image, and the cursor is positioned at that field. The associated error message is also displayed. The work station user then corrects the error.

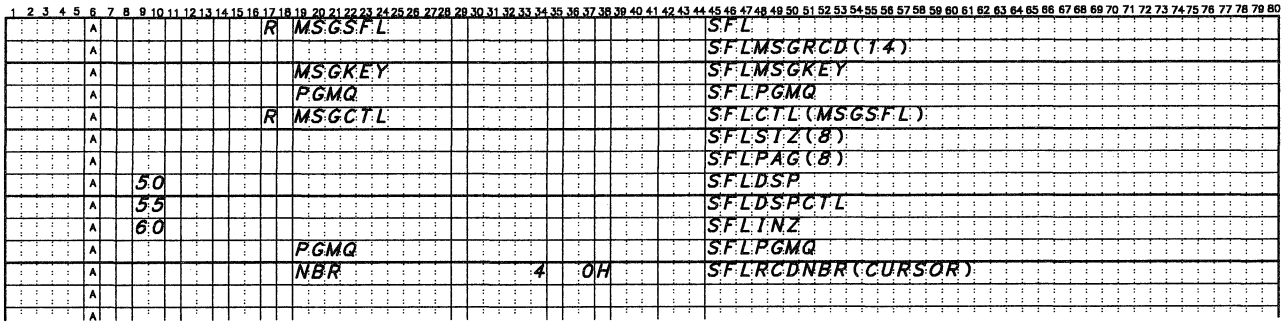
A decision you must make when using the SFLNXTCHG keyword is whether to allow the user to change the subfile fields that were not in error. If you do not want the work station user to change those fields you can protect them with the DSPATR(PR) keyword. For those fields you do not want changed, the DSPATR(PR) keyword must be in effect only when the SFLNXTCHG keyword is in effect. If you allow the work station user to change the fields, you can:

- Define hidden fields for those fields that are to be checked.
- Move the data originally entered by the work station user into the hidden fields when an error occurs on a subfile.
- Compare the data in the hidden fields to the fields just read to identify which fields have been changed so you can update the records that have already been processed when the work station user makes the changes.

Message Subfile

You can use a subfile to display messages for multiple errors. The messages to be placed in the subfile are on a program message queue. Each message written to the subfile is displayed on a separate line and is truncated, if necessary. Each message line contains an attribute character in position 1 that is displayed as a blank, followed by the message text. For the 24 by 80 display mode, 76 characters are displayed. For the 27 by 132 display mode, 128 characters are displayed. Since both the message identifier and the message data are available from the program message queue, second-level text and substitution text are supported for the messages placed in a message subfile. If the SFLMSGRCD keyword is specified, the SFLPGMQ and SFLMSGKEY keywords must also be specified.

The following shows an example of the DDS for a message subfile:



```
     A          R MSGSFL                    SFL
     A                                      SFLMSGRCD(14)
     A            MSGKEY                    SFLMSGKEY
     A            PGMQ                      SFLPGMQ
     A          R MSGCTL                    SFLCTL(MSGSFL)
     A                                      SFLSIZ(8)
     A                                      SFLPAG(8)
     A  50                                  SFLDSP
     A  55                                  SFLDSPCTL
     A  60                                  SFLINZ
     A            PGMQ                      SFLPGMQ
     A            NBR            4  0H      SFLRCDNBR(CURSOR)
     A
     A
```

RSLH156-0

The SFLRCDNBR(CURSOR) keyword is used to position the cursor at the first displayed character in the message subfile that is specified in the SFLRCDNBR field so the Roll Up and Roll Down keys will apply to the message subfile.

For information on sending and receiving messages and on the program message queue, see the *CL Programmer's Guide*. Message subfiles are the only kind of subfiles supported for CL programs and for C/400 programs.

Positioning the Cursor

The DSPATR(PC) keyword lets you position the cursor for each page of the subfile record that is displayed. Write and update operations can be used to control DSPATR(PC) for:

- The initial display of subfile records. (A write or write-read operation to the subfile control record when the SFLDSP and SFLDSPCTL keywords are used.)
- Subfile records displayed using a roll key or a fold or truncate key (SFLDROP keyword).

Cursor Positioning for the Initial Display of Subfile Records and the Subfile Control Record: The cursor is positioned by the first of the following conditions that applies:

- The CSRLOC keyword on the subfile control record.
- The DSPATR(PC) keyword within the records being displayed.
- The DSPATR(PC) keyword within a field in the subfile control record.
- The SFLRCDNBR(CURSOR) keyword within the subfile control record.
- If nothing is specified, the cursor is positioned at the first input-capable field on the display.

Note: If the keyboard is unlocked prior to the output operation that displays the subfile, explicit cursor positioning is not performed.

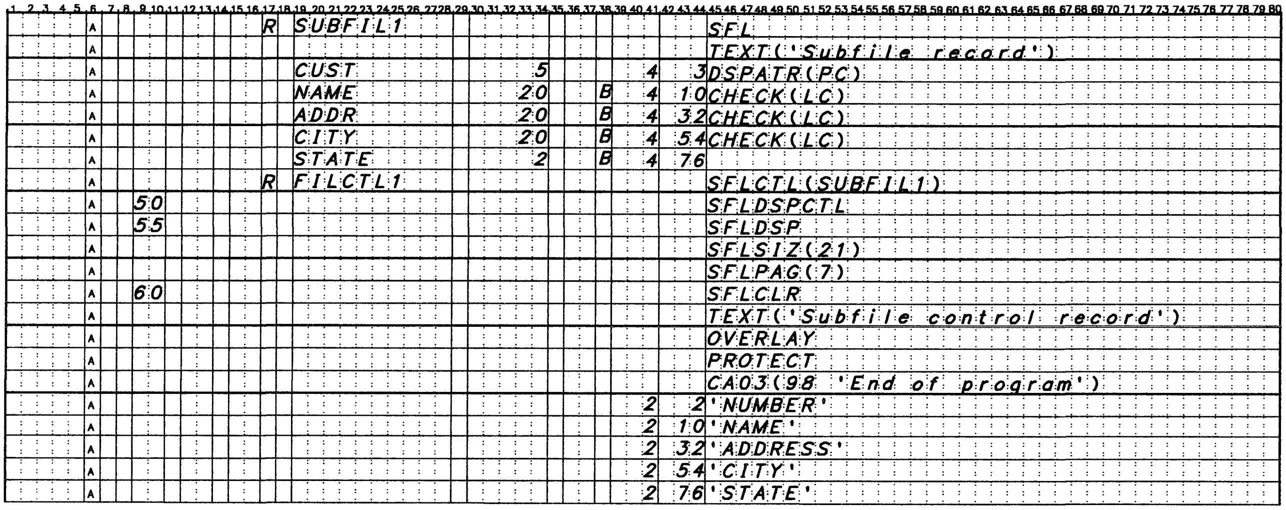
Use the keywords in the following order:

- The CSRLOC keyword can be used to position the cursor anywhere on the screen.
- The DSPATR(PC) keyword can be used to position the cursor at any field in the first record displayed when the output operation specifies the SFLDSP keyword.
- The DSPATR(PC) keyword can be used to position the cursor at any field of the subfile control record.
- The SFLRCDNBR(CURSOR) keyword can be used to position the cursor at the first input-capable field of the record whose record number is used to select which page is to be displayed first. If no input fields exist, the cursor is positioned at the first selected output field in that record.
- If neither the DSPATR(PC) nor SFLRCDNBR(CURSOR) keyword is used, the cursor is positioned at the first input-capable field on the display.

Cursor Positioning When a Roll Key Is Used: The positioning of the cursor when a roll key is used is dependent on whether the DSPATR(PC) keyword is used:

- If the DSPATR(PC) keyword is not used, the cursor is positioned at the same location as when the roll key was pressed.
- If the DSPATR(PC) keyword is used, the cursor is positioned at the first field in the displayed subfile records with the DSPATR(PC) keyword in effect.

The following example illustrates both and shows part of the DDS for a subfile in which records are displayed vertically. Customer number, name, address, city, and state are displayed. A work station user can change customer name, address, city, and state. Customer number cannot be changed; it is an output field only. The DSPATR(PC) keyword has been specified for the customer number field (CUST). Subfile size is 21 and page size is 7.

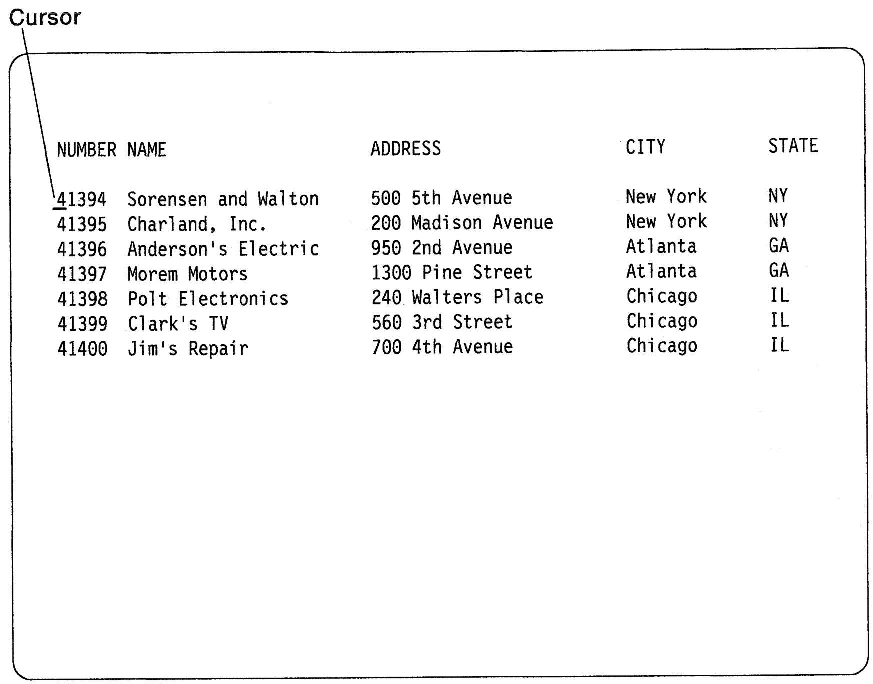


```
     A          R SUBFIL1                   SFL
     A                                      TEXT('Subfile record')
     A            CUST           5      4  3DSPATR(PC)
     A            NAME          20   B  4 10CHECK(LC)
     A            ADDR          20   B  4 32CHECK(LC)
     A            CITY          20   B  4 54CHECK(LC)
     A            STATE          2   B  4 76
     A          R FILCTL1                   SFLCTL(SUBFIL1)
     A  50                                  SFLDSPCTL
     A  55                                  SFLDSP
     A                                      SFLSIZ(21)
     A                                      SFLPAG(7)
     A  60                                  SFLCLR
     A                                      TEXT('Subfile control record')
     A                                      OVERLAY
     A                                      PROTECT
     A                                      CA03(98 'End of program')
     A                                  2  2'NUMBER'
     A                                  2 10'NAME'
     A                                  2 32'ADDRESS'
     A                                  2 54'CITY'
     A                                  2 76'STATE'
```

RSLH157-3

Figure 4-6. Using DSPATR(PC) to Position the Cursor in a Subfile

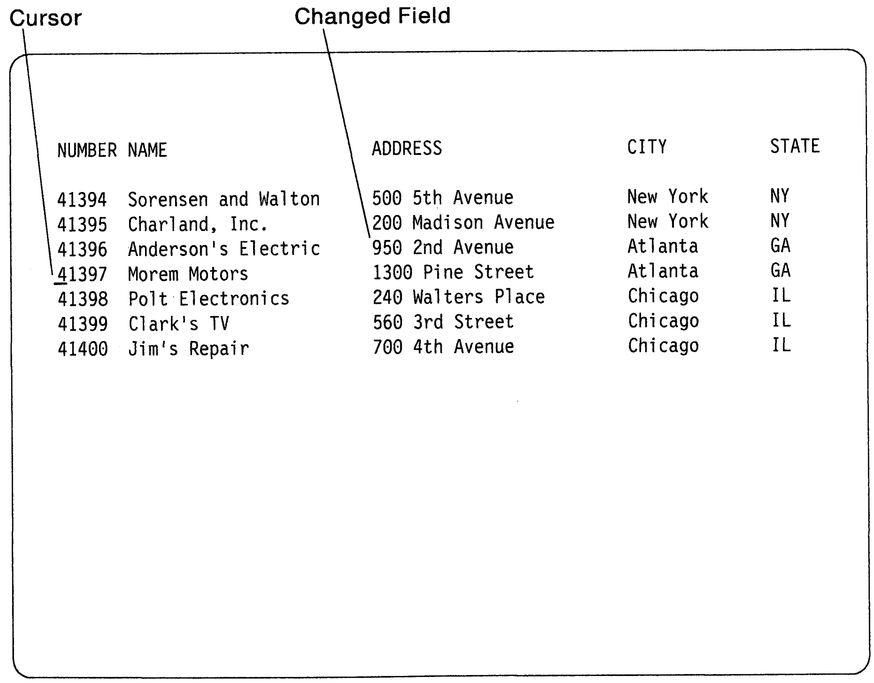
The initial display looks like this:



```
NUMBER NAME                ADDRESS              CITY        STATE

41394  Sorensen and Walton 500 5th Avenue       New York    NY
41395  Charland, Inc.      200 Madison Avenue   New York    NY
41396  Anderson's Electric 950 2nd Avenue       Atlanta     GA
41397  Morem Motors        1300 Pine Street     Atlanta     GA
41398  Polt Electronics    240 Walters Place    Chicago     IL
41399  Clark's TV          560 3rd Street       Chicago     IL
41400  Jim's Repair        700 4th Avenue       Chicago     IL
```

RSLH709-0

The first seven records in the subfile are displayed and the cursor is positioned under the customer number in the first record. The work station user moves the cursor to the third record, updates the address for that customer, and moves the cursor to the customer number of the fourth record:

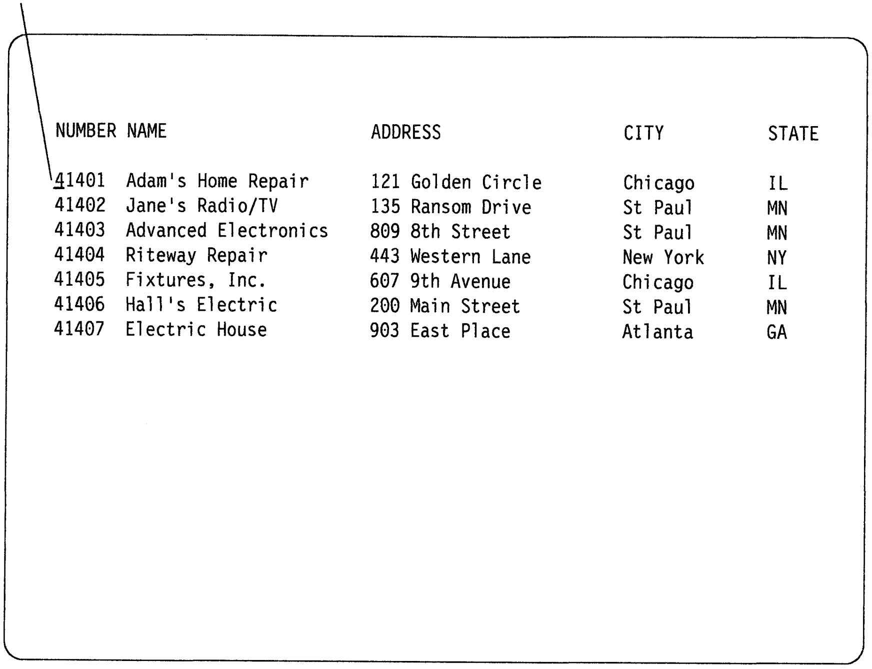


```
NUMBER NAME                ADDRESS              CITY        STATE

41394  Sorensen and Walton 500 5th Avenue       New York    NY
41395  Charland, Inc.      200 Madison Avenue   New York    NY
41396  Anderson's Electric 950 2nd Avenue       Atlanta     GA
41397  Morem Motors        1300 Pine Street     Atlanta     GA
41398  Polt Electronics    240 Walters Place    Chicago     IL
41399  Clark's TV          560 3rd Street       Chicago     IL
41400  Jim's Repair        700 4th Avenue       Chicago     IL
```

RSLH183-0

Now the work station user presses the Roll Up key to display the next seven records. The cursor is positioned under the customer number in the first record:

Cursor

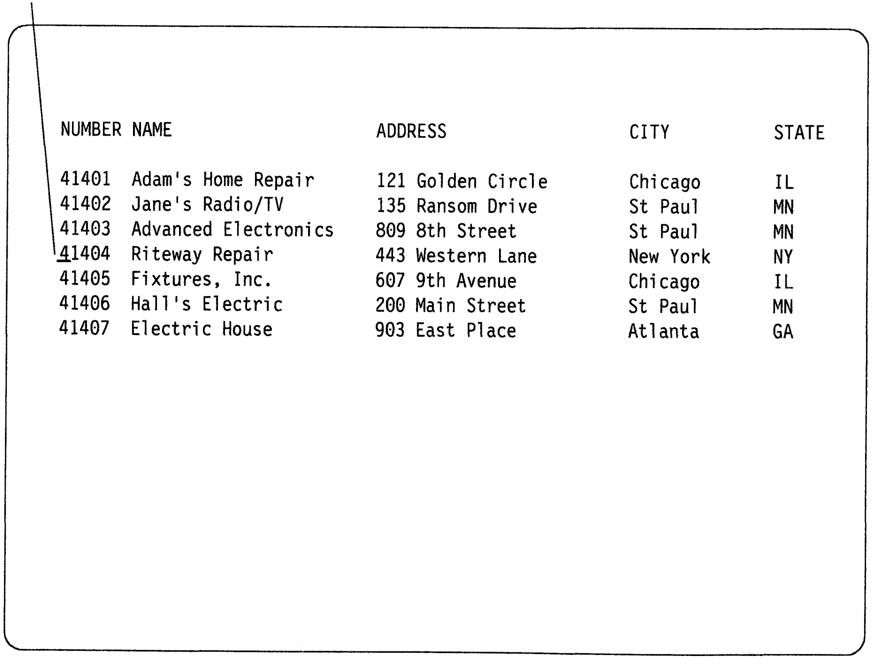


```
NUMBER NAME                 ADDRESS               CITY          STATE

41401  Adam's Home Repair   121 Golden Circle     Chicago       IL
41402  Jane's Radio/TV      135 Ransom Drive      St Paul       MN
41403  Advanced Electronics 809 8th Street        St Paul       MN
41404  Riteway Repair       443 Western Lane      New York      NY
41405  Fixtures, Inc.       607 9th Avenue        Chicago       IL
41406  Hall's Electric      200 Main Street       St Paul       MN
41407  Electric House       903 East Place        Atlanta       GA
```

RSLH184-0

If the DSPATR(PC) keyword had not been specified and the work station user pressed the Roll Up key, the cursor would have been positioned at the fourth record under customer number:

Cursor



```
NUMBER NAME                 ADDRESS               CITY          STATE

41401  Adam's Home Repair   121 Golden Circle     Chicago       IL
41402  Jane's Radio/TV      135 Ransom Drive      St Paul       MN
41403  Advanced Electronics 809 8th Street        St Paul       MN
41404  Riteway Repair       443 Western Lane      New York      NY
41405  Fixtures, Inc.       607 9th Avenue        Chicago       IL
41406  Hall's Electric      200 Main Street       St Paul       MN
41407  Electric House       903 East Place        Atlanta       GA
```

RSLH185-0

Cursor Positioning When a Fold or Truncate Key Is Used: When you use the SFLDROP keyword to assign a CF or CA key as a fold key, cursor positioning is handled the same way as described under "Cursor Positioning When a Roll Key Is Used" on page 4-56. The cursor is positioned as specified for all displayed records, including those folded. When you use the SFLDROP keyword to assign a CF or CA key as a truncate key, the cursor is positioned only for those fields displayed. Cursor positioning specifications for fields in the folded portion of the record are ignored.

Cursor Positioning and Rolling When Two or More Records Are Displayed: When you display two or more records at the same time, the position of the cursor determines the action taken when the work station user presses a roll key, regardless of which record was written to the display last.

The cursor can be positioned in the *rollable area* of the display or in the *nonrollable area* of the display. A *rollable area* is:

- A nonsubfile record with the ROLLUP/ROLLDOWN keyword in effect
- A subfile control record with the ROLLUP/ROLLDOWN keyword in effect
- A rollable subfile, which is an active subfile with subfile size greater than page size
- An active subfile with subfile size equal to page size and the ROLLUP/ROLLDOWN keyword in effect for its subfile control record

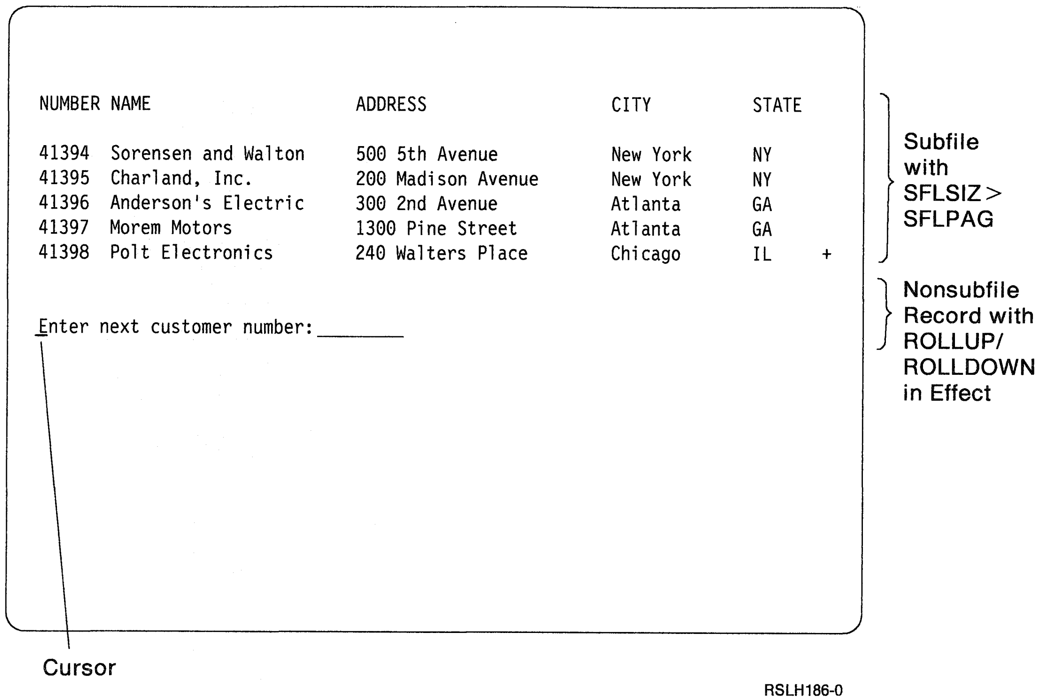
Based on the location of the cursor, the action taken when the work station user presses a roll key is as follows:

- If the cursor is positioned in the rollable area at a rollable subfile or at the subfile control record for a rollable subfile, the subfile is rolled. If the subfile is at the end of the subfile and the corresponding ROLLUP/ROLLDOWN keyword is not in effect, the end-of-subfile message is sent to the work station user. If the ROLLUP/ROLLDOWN keyword is in effect at the end of the subfile, control returns to the program.
- If the cursor is positioned in the rollable area at a nonsubfile record with the ROLLUP/ROLLDOWN keyword in effect, at a subfile control record with the ROLLUP/ROLLDOWN keyword in effect, or at an active subfile with subfile size equal to page size and the ROLLUP/ROLLDOWN keyword in effect for the subfile control record, control returns to the program.
- If the cursor is not positioned in the rollable area, the system attempts to find the uppermost rollable area on the display and perform the action indicated, as listed previously. If there is no rollable area on the display, the command-key-not-valid message is sent to the work station user.

  Note: Records that do not occupy display space (record formats with no fields, with hidden, program-to-system, or message fields only, or with the CLRL keyword specified and no input-capable fields; and message subfiles) are assumed to be at line 0. Therefore, these records are considered to be uppermost on the display, and the system attempts to roll them first.

The following examples illustrate what action is taken, based on the location of the cursor, when two records are displayed and the work station user presses a roll key.

In the following example, control returns to the program if the corresponding ROLLUP/ROLLDOWN keyword is in effect. This occurs because the cursor is positioned at a nonsubfile record with the ROLLUP/ROLLDOWN keyword in effect.

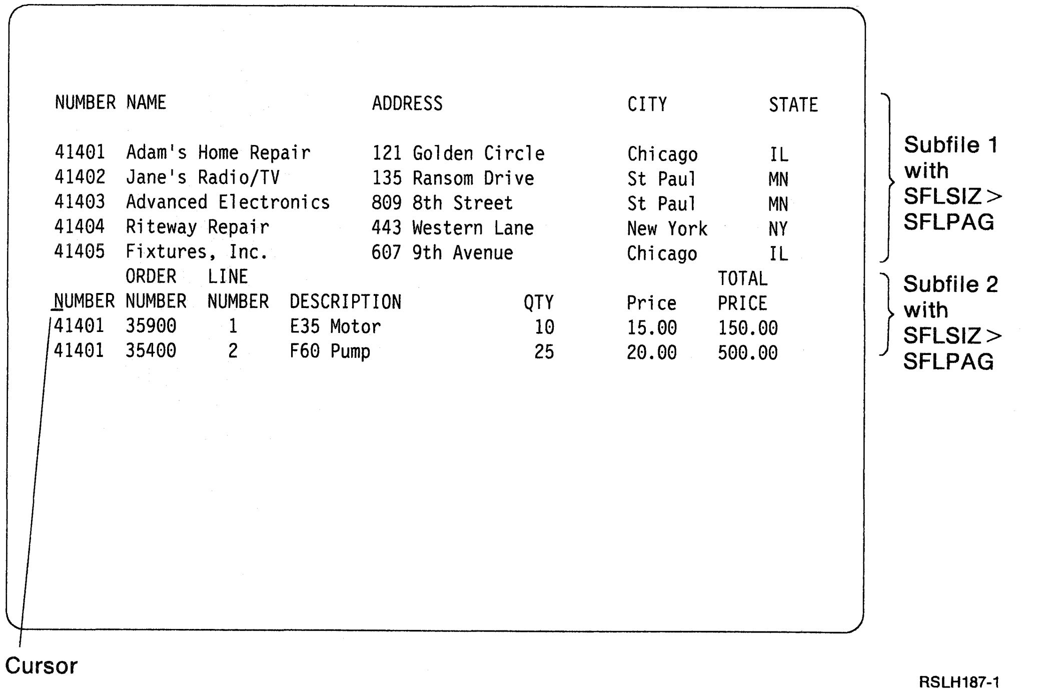


```
NUMBER NAME                 ADDRESS              CITY        STATE

41394  Sorensen and Walton  500 5th Avenue       New York    NY
41395  Charland, Inc.       200 Madison Avenue   New York    NY
41396  Anderson's Electric  300 2nd Avenue       Atlanta     GA
41397  Morem Motors         1300 Pine Street     Atlanta     GA
41398  Polt Electronics     240 Walters Place    Chicago     IL     +


Enter next customer number:________
```

Subfile with SFLSIZ> SFLPAG
Nonsubfile Record with ROLLUP/ ROLLDOWN in Effect
Cursor

RSLH186-0

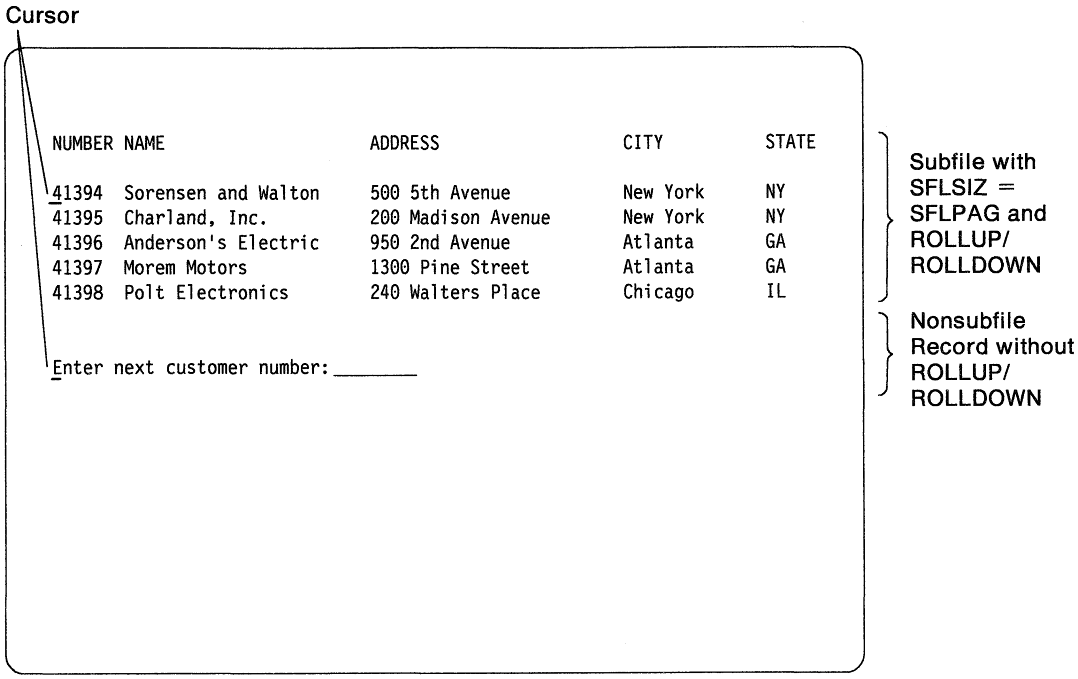
In the following example, two subfiles with the subfile size greater than the page size and their control records are displayed. The work station user positions the cursor in the bottom subfile control record. The bottom subfile is rolled because the cursor is positioned at a rollable subfile in the rollable area of the display.



```
NUMBER NAME                 ADDRESS              CITY        STATE

41401  Adam's Home Repair   121 Golden Circle    Chicago     IL
41402  Jane's Radio/TV      135 Ransom Drive     St Paul     MN
41403  Advanced Electronics 809 8th Street       St Paul     MN
41404  Riteway Repair       443 Western Lane     New York    NY
41405  Fixtures, Inc.       607 9th Avenue       Chicago     IL
       ORDER  LINE                                           TOTAL
NUMBER NUMBER NUMBER  DESCRIPTION          QTY   Price       PRICE
41401  35900    1     E35 Motor             10   15.00       150.00
41401  35400    2     F60 Pump              25   20.00       500.00
```

RSLH187-1

In the following example, a subfile with the subfile size equal to the page size and the ROLLUP/ROLLDOWN keyword in effect is written to the display first. A nonsubfile record without the ROLLUP/ROLLDOWN keyword in effect is then written to the display. If the cursor is positioned at either the subfile record or the second record, control returns to the program.

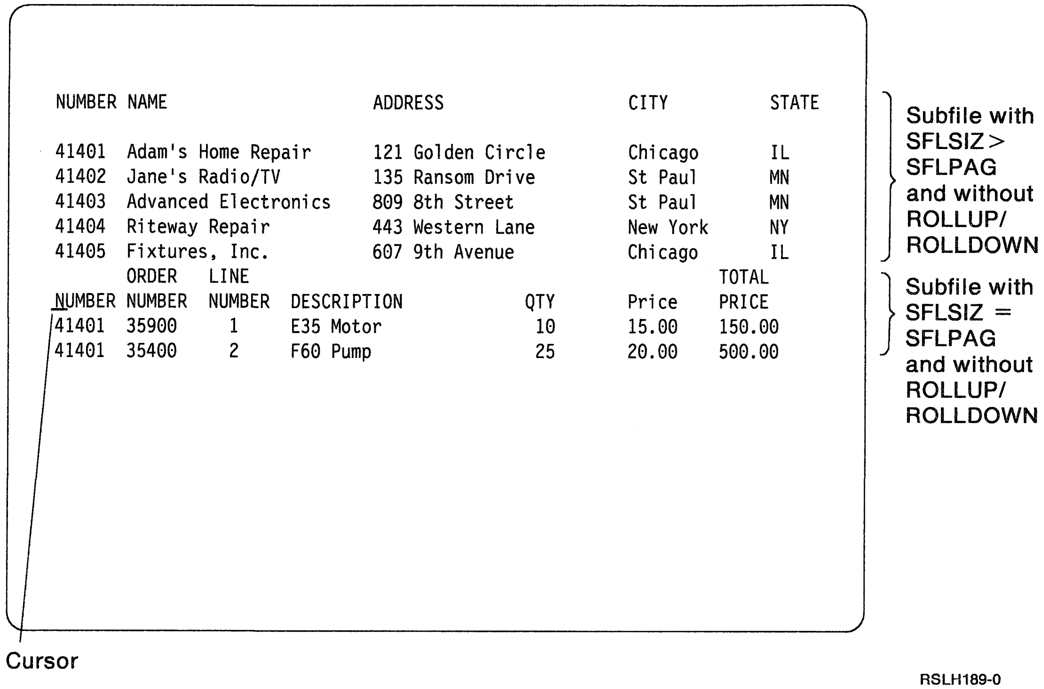


```
NUMBER NAME                   ADDRESS                 CITY          STATE

41394  Sorensen and Walton    500 5th Avenue          New York      NY
41395  Charland, Inc.         200 Madison Avenue      New York      NY
41396  Anderson's Electric    950 2nd Avenue          Atlanta       GA
41397  Morem Motors           1300 Pine Street        Atlanta       GA
41398  Polt Electronics       240 Walters Place       Chicago       IL


Enter next customer number: ________
```

Cursor
Subfile with SFLSIZ = SFLPAG and ROLLUP/ ROLLDOWN
Nonsubfile Record without ROLLUP/ ROLLDOWN

RSLH188-0

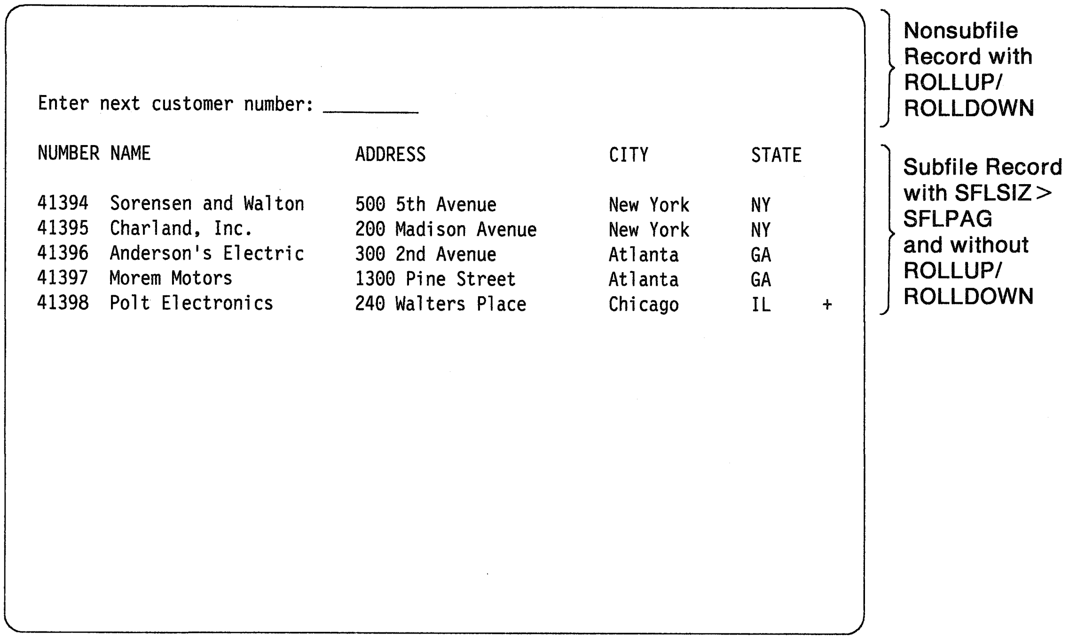
In the following example, the first subfile has a subfile size greater than the page size but the ROLLUP/ROLLDOWN keyword is not specified. The second subfile has a subfile size equal to the page size but the ROLLUP/ROLLDOWN keyword is not specified. If the cursor is positioned at the second subfile, the first subfile is rolled. In this position, the cursor is not in a rollable area; therefore, the system finds the uppermost rollable area in the display and performs the roll function.



```
NUMBER NAME                   ADDRESS              CITY         STATE

41401  Adam's Home Repair     121 Golden Circle    Chicago       IL
41402  Jane's Radio/TV        135 Ransom Drive     St Paul       MN
41403  Advanced Electronics   809 8th Street       St Paul       MN
41404  Riteway Repair         443 Western Lane     New York      NY
41405  Fixtures, Inc.         607 9th Avenue       Chicago       IL
       ORDER   LINE                                        TOTAL
NUMBER NUMBER  NUMBER  DESCRIPTION          QTY    Price   PRICE
41401  35900     1     E35 Motor             10    15.00   150.00
41401  35400     2     F60 Pump              25    20.00   500.00
```

Subfile with SFLSIZ> SFLPAG and without ROLLUP/ ROLLDOWN
Subfile with SFLSIZ = SFLPAG and without ROLLUP/ ROLLDOWN
Cursor
RSLH189-0

In the following example, the subfile with a subfile size greater than the page size and without ROLLUP/ROLLDOWN keyword is written to the display first. The nonsubfile record with ROLLUP/ROLLDOWN keyword in effect is then written to the display above the subfile. If the cursor is positioned within the subfile record, the subfile is rolled. If the cursor is not positioned within the subfile record, the subfile is not rolled and control returns to the program.



```
Enter next customer number: _________

NUMBER NAME                     ADDRESS                 CITY          STATE

41394  Sorensen and Walton      500 5th Avenue          New York      NY
41395  Charland, Inc.           200 Madison Avenue      New York      NY
41396  Anderson's Electric      300 2nd Avenue          Atlanta       GA
41397  Morem Motors             1300 Pine Street        Atlanta       GA
41398  Polt Electronics         240 Walters Place       Chicago       IL      +
```

Nonsubfile Record with ROLLUP/ ROLLDOWN

Subfile Record with SFLSIZ> SFLPAG and without ROLLUP/ ROLLDOWN

RSLH190-1

File Definition Considerations

The DFRWRT (Defer Write) and RSTDSP (Restore Display) parameters can be specified on the CRTDSPF command when you create the file. These parameters control the way the system processes the display file I/O and what happens to the file data when records from other files are written to the display.

DFRWRT (Defer Write) Parameter

The DFRWRT parameter on the Create Display File (CRTDSPF) or Change Display File (CHGDSPF) command allows you to specify how the system is to handle write operations. If you specify DFRWRT(*NO), the program does not regain control until the write operation has displayed the data and updated the input/output feedback area.

If you specify the default of DFRWRT(*YES) for the file, the program regains control after the output record is processed. The program can then use the record area where the output was stored to start processing the next write or write-read operation. The data is actually sent to the display only when a read or write-read operation is issued or when the FRCDTA DDS keyword is in effect for a write-only operation. For a description and example of the FRCDTA keyword, see the *DDS Reference*.

Using DFRWRT(*YES) on a display file improves systems performance; however, DFRWRT(*YES) should not be used in the following circumstances:

- If you want to find out immediately if the write operation was successful. An error associated with a write operation for a file with DFRWRT(*YES) specified is issued only when the data is actually sent to the display.
- If the time between the write operation and the read or write-read is long. For example, if the program does several database operations after the write operation (before it issues a read or write-read operation), the user will not see the data while the database operations are being performed.
- If the file is closed after the write-only operation and the KEEP keyword is not specified. If the display file has the DDS keyword KEEP specified in any of its records, the data accumulated from the write-only operation is displayed when the file is closed. However, if the KEEP keyword is not specified, the data may never be displayed.

The DFRWRT parameter has no effect on the following:

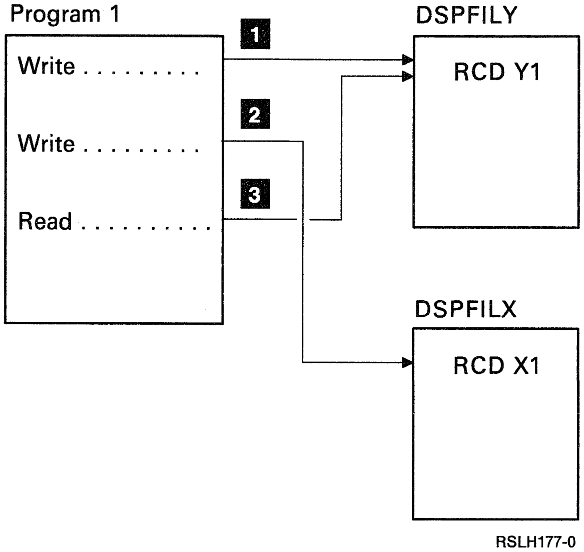
- Write operations using user-defined data streams
- Write operations to display files that use program-described data
- Record formats for which the FRCDTA DDS keyword is in effect

RSTDSP (Restore Display) Parameter

The RSTDSP parameter on the Create Display File (CRTDSPF) or Change Display File (CHGDSPF) command allows you to control whether the system saves information that was previously displayed. A display file may be opened to a device even when another display file is already using that device. When an I/O operation is performed to the second display file, the first display file is *suspended*. When a display file is suspended, the information that is on the display can be saved automatically by the system. You control this by using the RSTDSP parameter on the CRTDSPF command when you create the display file. You can change this parameter after the file is created by using the CHGDSPF command. By using

RSTDSP(*YES), the system saves the contents of a display file when it is suspended so that they may be restored when an I/O operation is later performed to that display file. By specifying RSTDSP(*NO), the system does not save the contents; the application program using that display file will rewrite the display if it needs to show more data with that display file.

These functions let you overlap a program call, keyboard input, and file I/O processing. The following example illustrates this:



1 Program 1 issues a write operation to record format RCD Y1 in file DSPFILY, activating DSPFILY.

2 Program 1 issues a write operation to record format RCD X1 in file DSPFILX, suspending files DSPFILY and activating DSPFILX. Assuming RSTDSP(*YES) parameter is specified for DSPFILY, the data displayed on the device is saved. If RSTDSP(*NO) had been specified instead, the data would be lost and the program would have to write the information in RCD Y1 again if it needed to show it.

3 Program 1 issues a read operation to record format RCD Y1, suspending file DSPFILX and restoring file DSPFILY. If the RSTDSP(*YES) parameter is specified for DSPFILY, then the data displayed on the device when DSPFILY is suspended will be restored.

When RSTDSP(*YES) is specified for a file, and you are suspending and restoring that file because of operations to another display file, some displays may appear to flash on the screen briefly. If a file has a record on the display and an I/O operation is done to a second file, the first file is suspended and its screen contents saved. When returning to the first file, the file and its screen contents are restored. If a write operation is being done to a different record format in that file, the restored display will flash briefly before the output operation is complete. If you are going to completely rewrite the display contents from your program when going back to the first file, you can use RSTDSP(*NO).

However, certain DDS keywords are used only when there is data already on the display from a display file. These keywords are CLRL, OVERLAY, PUTOVR, PUTRETAIN, ERRMSG, and ERRMSGID. If you are writing a record that has these keywords in effect, you must be sure that the records that are on the display are the records to which these keywords apply. If you have done anything that would have caused the display file to be suspended, that data must be restored to the screen so that the write operations to record formats using these keywords make sense. Use the RSTDSP(*YES) parameter to ensure this.

Another time that RSTDSP(*YES) should be used is when you perform multiple read operations to a record format on the display without intervening write operations. If you should call a program while processing the data that has been read and that program presents a display of its own, the subsequent read operation done by your program will restore the display properly.

The saving and redisplay of the display data requires significant system and data transmission overhead. For a 1920-character 5250 display station, approximately 3000 characters are transmitted for each save and redisplay of display data. To avoid this overhead, applications can be written so that the programs in the application either:

- Share the same copy of the display file among themselves by specifying SHARE(*YES) on the display file, or
- Perform complete display rewrites each time they write to the display.

A complete display rewrite occurs when a record is written to the screen and the OVERLAY keyword is not used or implied.

Note: If complete display rewrites are not performed and if new input fields, occupying positions on the screen above the currently displayed fields, are sent to the display, the program receives a message (CPF5192). This occurs because the 5250 display station requires that new input fields sent to the display appear in lower positions than input fields currently on the display. In normal operations, data management performs field processing to satisfy the 5250 requirement. See “Positioning Record Formats” on page 4-87.

When two programs use the same display file and that file has the SHARE(*YES) parameter specified, and one program calls the other, the display file that they have in common will not be suspended even though both programs have opened the file. If the file is not shared, then the system maintains separate copies of the file for each program and will suspend and restore the files as if they were not the same.

Since system programs never specify file sharing, you should specify RSTDSP(*YES) on the CRTDSPF or CHGDSPF command if the program containing the file calls system functions that present displays. System functions that break into the normal path of an application, such as the System Request Menu or the presentation of break messages, will always restore the display to the state that the application program expects; so you do not need to use RSTDSP(*YES) to protect your display file from these functions.

When a close operation is issued to a suspended file, the saved display data (RSTDSP(*YES) parameter) is redisplayed only if the KEEP keyword was specified for a record format in the saved display data.

Field Definition Considerations

Field Attribute Characters

Each field displayed has a beginning attribute character and an ending attribute character associated with it that define the displayed field. The beginning character precedes the first character of a field and is displayed as a blank. The ending attribute character follows the last character of a field and is also displayed as a blank. For example, if you specify a field for positions 2 through 8, the beginning attribute character is in position 1 and the ending attribute character is in position 9. These characters are not included in the field length you specify in DDS. A beginning attribute character can overlap an ending attribute character; that is, they can occupy the same position on the display. However, nothing else can overlap the beginning attribute character. Therefore, when you design a display, you must allow one space for the beginning attribute character of each field. You can use the blank attribute character to space between fields when they are displayed.

When a record is displayed so that the last field in the record ends in the last position on the line, the ending attribute character for that field is in the first position of the next line. The beginning attribute character of the first field in the next record can be superimposed on the ending attribute character. For example, if the ending attribute character for the last field in record 1 is in position 1 of line 5, the beginning attribute character for record 2 can also be in position 1 of line 5. In this case, the first record is not considered to be overlapped. However, if the first field in a record begins in position 1, which means that the beginning attribute character is in the last position of the preceding line, the previous record is overlapped and is cleared from the display.

Specifying Display Attributes

You can emphasize a field of a record on the display by specifying certain display attributes in the DDS for the file. However, any function not supported for your display device is ignored. The functions of the display attributes are:

- Underlining a field (DSPATR(UL) keyword), which is the default for input fields.
- Placing vertical separators between the characters in a field (DSPATR(CS) keyword).
- Highlighting a field by displaying it with greater intensity than is normally used on the display (DSPATR(HI) keyword).
- Reversing the image of a field from light on dark to dark on light or from dark on light to light on dark (DSPATR(RI) keyword).
- Making the data in the field invisible to the work station user (DSPATR(ND) keyword).
- Placing the cursor at a specific field (DSPATR(PC) keyword).
- Blinking a field when it is displayed (DSPATR(BL) keyword).

Note: You can use the PUTOVR, OVRDTA, and OVRATR keywords or the PUTRETAIN keyword to change the attributes of a field without resending the field. If you use the PUTOVR, OVRDTA, and OVRATR keywords, you cannot use the PUTRETAIN keyword.

Using Color Attributes

You can design your displays for use on display stations that show color. The DDS keyword COLOR allows you to specify the following colors for fields: green, white, red, turquoise, yellow, pink, and blue. This keyword is ignored if it is selected for a field displayed on a noncolor display station. See the *DDS Reference* for more information on the COLOR keyword and on the use of display attributes with the COLOR keyword.

If the COLOR keyword is not specified in the DDS for the display file but the device is specified in the device description as a color display station, displays that you have designed for noncolor display devices can also be used for the color display station. The keywords DSPATR(UL) and DSPATR(RI), if specified on separate fields, function the same as they do for the 5250 display station. However, the keywords DSPATR(CS), DSPATR(HI), and DSPATR(BL) produce the following colors on a color display station (the specified display attributes CS, HI, and BL are suppressed).

Color Produced on the Color Display Station when No COLOR Keyword is Specified	**Display Attribute Selected: DSPATR(CS)**	**Display Attribute Selected: DSPATR(HI)**	**Display Attribute Selected: DSPATR(BL)**
Green (normal)			
Turquoise	X		
White		X	
Red, no blinking			X
Red, blinking		X	X
Yellow	X	X	
Pink	X		X
Blue	X	X	X

Specifying Validity-Checking Functions

You can have the system check the validity of data entered by the work station user. If errors are found, a message is issued so that the work station user can correct the input before the record is passed to the application program. The validity-checking functions you can specify in DDS are:

- Detecting fields in which at least one character must be entered (CHECK(ME) keyword). Blanks are valid characters. This is referred to as *mandatory enter*.
- Detecting fields in which every position must contain a character (CHECK(MF) keyword). Blanks are valid characters. This is referred to as *mandatory fill*.
- Detecting incorrect data types where character, numeric, or signed numeric data is required.
- Detecting data that is not in the range specified for the field (RANGE keyword).

- Performing comparison checking between data entered and specified constant value (COMP keyword).
- Comparing the data entered to a specific list of valid entries (VALUES keyword).
- Detecting if a valid field or record name was entered in a character field (CHECK(VN) keyword).
- Detecting if a valid object name was entered in a character field (CHECK(VNE) keyword).
- Performing modulus 10 or 11 check digit verification (CHECK(M10) or CHECK(M11) keyword). (Only one can be specified.)
- Allowing blank-key entries to be processed as if no entry had been made (CHECK(AB) keyword). CHECK(AB) – Allow Blanks – is ignored if the subfile keyword SFLROLVAL or SFLRCDNBR is also specified for the field.

When you specify the RANGE, COMP/CMP, VALUES, CHECK(VN), CHECK(VNE), CHECK(M10), or CHECK(M11) keyword for validity checking and an error is detected by one of these validity checking functions, the following happens:

1. The keyboard is locked.
2. All fields in error are displayed in reverse image. If a field in error has both the underline (UL) display attribute and the highlight attribute (HI), its image is not reversed, as this combination of attributes has the same effect as DSPATR(ND).
3. The cursor is positioned at the beginning of the first field in error.
4. An error message for the first field in error is displayed on the error line.

If the RANGE, COMP, VALUES, CHECK(VN), or CHECK(VNE) keyword is specified for a field, and data is entered into that field, the field indicates that it has been changed regardless of attempts by the work station user to restore the field after an error. If blanks (for character fields) or zeros (for numeric fields) will fail the validity checking function, use the CHECK(AB) keyword. This will satisfy the validity checking function.

When you specify validity checking for records that are part of a subfile, each field in the record is validity checked before it is placed in the subfile from the display. You cannot roll the records until all fields in error are corrected.

Device support only performs validity checking on a field if the field is changed by the work station user or if its modified data tag (MDT) is set on using DSPATR(MDT).

Notes:

1. If the work station user presses the Dup key, any validity checking for a field is ignored. The DUP keyword lets the work station user use the Dup key.

2. The value for a numeric field for which the COMP, VALUES, or RANGE keyword is specified is aligned based on the decimal positions specified for the field and filled with zeros where necessary. If decimal positions were not entered for the field, the decimal point is assumed to be to the right of the digit to the extreme right in the value. For example, for a numeric field with length of 5 and decimal positions of 2, the value 1.2 is interpreted as 001.20 and the value 100 is interpreted as 100.00.

3. When you use the RANGE keyword for validity checking an input field and blanks are entered in the input field, the value for the input field may not meet the range requirements. Blanks are converted to zeros for numeric fields and are passed as blanks for character fields. Use the field level keyword BLANKS to determine when a field is displayed as all blanks. The response indicator on the BLANKS keyword is set on if the work station user enters blanks. For more information on the BLANKS keyword, see the *DDS Reference*.

Limitations on the Number of Input-Capable Fields

For a remote 5250 display station (a work station attached through a remote controller), you can specify as many as 126 or 256 input fields on one display, depending on the controller model. (The 5294 controller supports 126 input fields; the 5394 controller supports 256 input fields.) Additionally, if either DSPATR(OID) or DSPATR(SP) is specified, this maximum is reduced by 1 for each three instances of these keywords. If fewer than three instances occur, it is still reduced by one.

For a local 5250 display station (a work station attached through the local work station controller), you can specify as many as 256 input fields. Also, if either DSPATR(OID) or DSPATR(SP) is specified, this maximum is reduced by 1 for each three instances of these keywords. In addition, any use of the magnetic stripe reader on a local 5250 display station also reduces the maximum number of fields. The maximum number of fields is calculated as follows:

$$256 - \left[\frac{3+B}{6} + \frac{A}{3}\right] \text{ rounded up to the next whole number}$$

RSLH131-2

A is the number of DSPATR(OID) and DSPATR(SP) fields on the display and B is the length of the longest expected magnetic stripe input where 125 data characters is the maximum allowed. Magnetic stripe data not specified as DSPATR(OID) can be entered into any input field.

If the maximum number of input fields is exceeded in any of the preceding cases, message CPF5192 is issued to the using program.

No maximum-number-of-fields diagnostic is provided during display file creation because the number of fields and record formats are not known until the program is run.

When a subfile record is displayed, the actual number of input-capable fields sent to the display is the number defined in the record multiplied by the number of subfile records that are displayed.

For remotely attached 3270 displays, the limitation is 126 input fields.

For ASCII displays attached through a protocol converter, the limitations are the same as the controller to which they are attached.

Negative Numeric Input Data Considerations

The negative sign in numeric input data can appear in three forms:

- Hex 60 if the sign is entered using the - (minus) key
- Hex D if the sign is entered using the Field Minus key
- Hex Dn if the sign is entered as an alphameric character with a D zone

The hex 60 is treated as a true minus sign if it is to the right of the least significant digit.

The hex D zone is treated as a minus sign if it is the least significant digit. In addition, it is treated as a significant digit with a value equal to the numeric portion.

Embedded blanks (between significant digits) are changed to zeros before decimal alignment.

Refer to the *DDS Reference* for input data examples.

Right-to-Left Cursor Support

The cursor can be made to move from right to left on the display between fields and in input fields. Two parameters for the DDS CHECK keyword can be used to do this:

- CHECK (RL): Moves the cursor from right to left in specified nonnumeric input fields or all nonnumeric input fields on the display.
- CHECK (RLTB): Moves the cursor from right to left between fields.

When using these parameters, remember the following:

- Modulus check digit verification is supported, but the check digit is still the byte to the extreme right of the field.
- A field for which right-to-left cursor movement is specified can occupy more than one line on the display. However, the cursor still moves from the top of the display to the bottom.
- You cannot use right-to-left cursor movement with user-defined data streams.

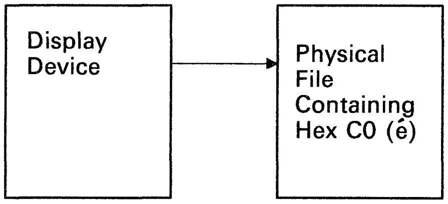
Note: If no cursor positioning is specified in the device file or by the program, the cursor is placed in the input-capable field to the extreme left of the top line.

Using Alternative Character Sets and Code Pages

In multinational environments, data in one national character set may need to be displayed or entered on display devices that support another national character set. This is particularly true of characters with accents and other characters with diacritical marks (such as c with a cedilla, n with a tilde, and u with an umlaut). In this section, these characters are called *extended alphabetics*.

For example, assume that a physical file on the system contains data in the Basic French character set, and includes an é (e with an acute accent). In the Basic French character set, this character is hex C0. The data could have been entered on a display device that can handle the character or could have been sent to this system from another system over a communications line.

Note: On some devices, extended alphabetics can be entered on the keyboard (by pressing Cmd, then the key to its right with two diacritical marks on it, then the 2 hexadecimal digits that represent that character).

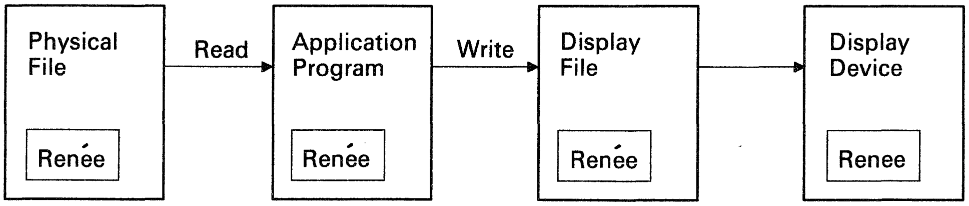


P7730016-0

When the hex C0 is sent to a display device that does not recognize hex C0 as é, the hex C0 is not displayed as é. For example, a display device that recognizes the U.S. basic character set displays hex C0 as { (a left brace). A reasonable character to substitute for é would be e, which is code point 85. To do this, the hex C0 must be translated to hex 85 for display on a U.S. basic display device.

You can use character translation to cause the hex C0 to be translated to an unaccented e (hex 85). With the IBM-supplied translate tables, the system can translate most extended alphabetics, but not all. If an extended alphabetic character does not have a clearly preferable equivalent, the system translates the character to a (-) hyphen.

System Has Characters Not Normally Displayed on the Device: The case of an output/input field shows how character translation occurs at a display device.



P7730015-0

Assume that a record in a physical file contains a field with the value Renée. An application program reads the record from the physical file, and writes a record containing the data to the display file. In the DDS for the display file, the output/input field in which Renée appears has the CHRID keyword specified. This keyword asks the system to perform character translation if needed. Character translation is needed if the CHRID parameter value for the display file differs from the CHRID parameter value for the display device. In this case, character translation is needed because the display file has CHRID(288 297), which is the basic French code page and character set, and the display device description has CHRID(101 37), which is the basic U.S. code page and character set.

When displaying the data, the system translates the hex C0 to hex 85, and Renee appears on the display.

On input, one of the following occurs:

- If the modified data tag (MDT) is turned on for the field (this happens when the work station user keys into the field or if the DSPATR(MDT) keyword is in effect for the field) the contents of the field (with the translated character hex 85) is returned to the program.
- If the modified data tag (MDT) is not turned on for the field, the saved contents of the field (with original, untranslated, data) is returned to the program.

Device Passes Characters Not Displayed on the System: If the work station user can enter data not normally displayed on the system (for instance, a user on a French-language device in Montreal dials up a remote line to a system in Toronto), no change occurs when the data is sent to the system. The program can read data from the display device and write it to a physical file. However, when another user on the Toronto system displays data from the physical file, the system attempts to translate the data for the more limited device.

Using Character Translation: Specify the CHRID keyword in the DDS for display file fields that will need translation.

Character translation occurs only if the CHRID parameter value in the display file is different from the CHRID parameter value in the device description.

1. The CHRID parameter on the display file is set by the CHRID parameter on the CRTDSPF, CHGDSPF, or OVRDSPF command. The CHRID parameter can have one of the following values:

   *DEVD (the default): Use the CHRID parameter specified on the device description.

   *SYSVAL: Use the character set and code page specified in the system value QCHRID.

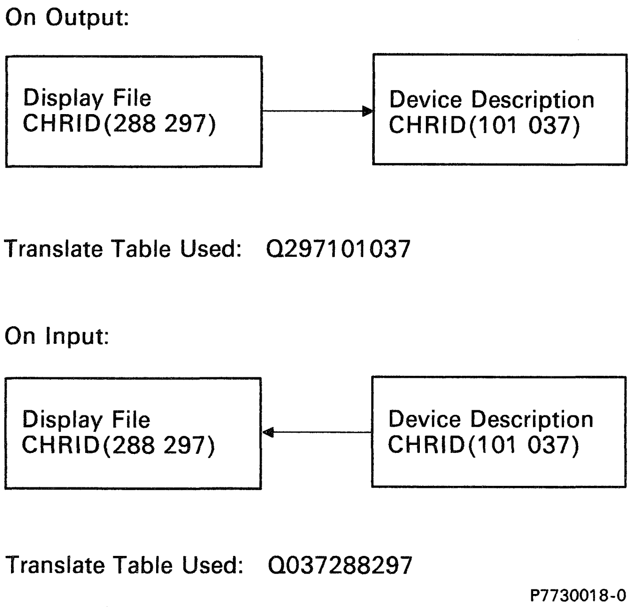
   Character set and code page: Use the character set and code page specified. See Table 4-3 on page 4-74 for the list of valid values.

2. The CHRID parameter on the device description is set by the CHRID parameter on the Create Device Description Display (CRTDEVDSP) or Change Device Description Display (CHGDEVDSP) command. This parameter can have one of the following values:

   *SYSVAL: Use the character set and code page specified in the system value QCHRID.

   Character set and code page: Use the character set and code page specified. See Table 4-3 on page 4-74 for the list of valid values.

If the CHRID parameter values are different, the system uses a translation table in library QUSRSYS to convert the data. Different translation tables are used for output and input. The name of the translation table used is derived from parts of the CHRID parameter values for the display file and device description, as follows:



On Output:

Display File
CHRID(288 297) → Device Description
CHRID(101 037)

Translate Table Used: Q297101037

On Input:

Display File
CHRID(288 297) ← Device Description
CHRID(101 037)

Translate Table Used: Q037288297

P7730018-0

IBM supplies a number of tables in library QUSRSYS to handle translation among the most frequently used combinations of character sets and code pages. If you need to tailor the way characters are translated, you can create your own table in library QUSRSYS to do the translation. To determine what translation tables are available to use, look in the QUSRSYS library for object type *TBL.

Character Identifier (CHRID) Values Supported for Each Display: The CHRID value specified should be based on the attributes of the display device. The table below shows CHRID values that are appropriate for each work station display keyboard type. For some display devices, you do not need to specify the KBDTYPE parameter, but the KBDTYPE value for the equivalent keyboard can be used to determine the CHRID value for the device:

Table 4-3 (Page 1 of 2). CHRID Values

Language/Country	Keyboard Type (KBDTYPE)	Limited CHRID	Full CHRID
International and US ASCII	INB	103 038	697 500
Multinational	AGI BLI CAI DMI FAI FNI FQI ICI INI ITI JEI NEI NWI PRI SFI SGI SPI SSI SWI UKI USI		697 500
Arabic	CLB		235 420
Austria/Germany	AGB	265 273	697 273
Belgium Multinational	BLI		697 500
Canada/French	CAB	277 260	341 260
Cyrillic	CYB		960 880
Denmark/Norway	DMB NWB	281 277	697 277
Finland/Sweden	FNB SWB	285 278	697 278
France	FAB FQB	288 297	697 297

Table 4-3 (Page 2 of 2). CHRID Values

Language/Country	Keyboard Type (KBDTYPE)	Limited CHRID	Full CHRID
Greece	GKB		218 423
Hebrew	NCB		941 424
Iceland	ICB		697 871
Italy	ITB	293 280	697 280
Japan/English	JEB	297 281	697 281
Japan/Kanji	JKB (For Personal System/55[1] and 5295 display stations)		332 290
Japan/Katakana	KAB (For 5251, 5291, 5292, and 3180 Katakana display stations)		332 290
Korea	KOB		933 833
Latin 2	ROB		959 870
Netherlands	NEB		697 037
Portugal	PRB	301 037	697 037
Simplified Chinese	RCB		936 836
Spain	SPB	305 284	697 284
Spanish Speaking	SSB	309 284	697 284
Switzerland/French Multinational	SFI		697 500
Switzerland/German Multinational	SGI		697 500
Thailand	THB		938 838
Traditional Chinese	TAB		101 037
Turkey	TKB		965 905
United Kingdom/English	UKB	313 285	697 285
United States/English	USB	101 037	697 037
Yugoslavia	YGI		959 870

[1] Personal System/55 and PS/55 are trademarks of the International Business Machines Corporation in the U.S.A. and/or other countries. IBM Personal System/55 is a family of double-byte capable personal computers that are used to process Chinese, Korean, and Japanese languages.

Editing Output Fields

The system provides editing support that makes fields more readable when they are displayed. With the system editing support, you can do the following:

- Suppress leading zeros
- Punctuate a field with commas and periods to show decimal column and to group digits by threes
- Print negative values with a minus sign or CR to the right
- Print zero values as zeros or blanks
- Print asterisks to the left of significant digits to provide asterisk protection
- Print a currency symbol corresponding to the system value QCURSYM

The system provides this editing support with edit codes and edit words. Edit codes are a defined set of editing patterns. You identify these by name, and the system edits a field according to the pattern defined by the named edit code. Edit words are edit patterns that you define to produce the desired results. Edit codes cover most commonly used editing requirements. You need to use the edit word support only for those editing needs not covered by edit codes.

There are two methods of using edit codes and edit words. Which one you use depends on how you define the display file and how it is used in an application program. If your application is using program-described data, your high-level language may allow you to identify edit codes or create your own edit words. If your application is using externally described data, the edit code (EDTCDE) DDS keyword allows you to identify an edit code, and the edit word (EDTWRD) DDS keyword allows you to define your own editing pattern.

The system provides several edit codes. They are:

- 1 through 4
- A through D
- J through M
- X through Z

The editing patterns defined by these codes are contained in Appendix E, "Edit Codes."

User-Defined Edit Codes

You can also define five edit codes to provide more editing function than is available with the OS/400 edit codes, and to handle common editing functions that would otherwise require the use of an edit word. These are called user-defined edit codes. For example, you may need to edit numbers that include hyphens (like some telephone numbers), or more than one decimal point. You can use user-defined edit codes for these functions. These edit codes are named QEDIT5, QEDIT6, QEDIT7, QEDIT8, and QEDIT9 and can be referred to in DDS or a high-level language program by number (5, 6, 7, 8, or 9).

These edit codes are created by using the Create Edit Description (CRTEDTD) command. Edit descriptions are always placed in library QSYS. They cannot be moved or renamed; only one occurrence of each is allowed. Edit descriptions have an object type of *EDTD.

Even though they are user-defined edit codes, your system comes with a version of each one of them. You can use these edit descriptions as they are, or you can delete them and create your own. The editing performed by the IBM-supplied versions of these edit descriptions as well as a definition of the contents of and the rules for using any user-defined edit code are described in Appendix E, "Edit Codes."

Before using any of the user-defined edit codes, you should check its contents on your system, since it may have been changed from the IBM-supplied version. The Display Edit Description (DSPEDTD) command will display the contents of a user-defined edit code.

Changing a user-defined edit code description does not affect any application or display file that has already been created using that edit description. If you want your application to use the changed edit description, you must either create the high-level language program again (if the edit code is referred to in the program) or create the file again (if the application is using an externally described file that contains EDTCDE keywords).

Indicators and Condition Names

Both option and response indicators can be specified at the file level, record format level, and field level. Indicators specified at the file level apply to all record formats within the file.

Each indicator occupies a one-character position in the input or output buffer for the appropriate operation (input response indicators and output option indicators only) unless a separate indicator area is specified by the INDARA keyword. A value of 1 for an indicator means that the indicator is on; a value of 0 means that the indicator is off.

You can use the INDARA keyword to separate the option and response indicators from the input and output records used by the program. If you use the INDARA keyword, the indicators are placed in a separate 99-character area; see your appropriate high-level language manual for information on how this 99-character area is defined.

If you use the same indicator number as both a response indicator and as an option indicator, you can use the status of the response indicator to set the option indicator for a subsequent output operation. For example, indicator 15 is used as both a response indicator and an option indicator. If the response indicator is on when an input operation is performed on the record format, option indicator 15 is set on and will be on when an output operation is performed for that record format.

The maximum number of record formats that you can define for a display file is 1024. If you do not use the INDARA keyword, the maximum number of fields that you can specify depends on the number of indicators (1 character each) you use and the length of each field you describe. The total combined length of all fields and indicators in a record format cannot exceed 32 763 characters. If you use the INDARA keyword to specify a separate indicator area, the maximum number of fields that you can specify depends only on the length of each field. The total number of all fields cannot exceed 32 763.

It is possible to have different response indicators for the ROLLUP/ROLLDOWN keywords on record formats displayed at the same time. For example, record A has specified a roll-up indicator of 52 and record B has specified a roll-up indicator of 25 and both records are displayed. When a read operation is requested to record A in your program, the operator presses the Roll Up key which returns control to your program. Record A is passed to your program with response indicator 52 set on; response indicator 25 is not set. Your program can then do a read operation to record B. When record B is passed to your program, response indicator 25 is set on; response indicator 52 is not set. Only the response indicator specified on the record format for which the read operation is done is set. The record format in which the cursor was located when the Roll Up key was pressed does not affect the setting of the response indicator associated with the ROLL keyword.

An indicator specified with the SETOF or SETOFF keyword becomes a response indicator that is set off and returned to the program. A corresponding indicator in the program is not set off until an input operation is performed. If the same indicator is specified elsewhere in the record format as a response indicator, the indicator is returned to the program based on the status of the associated keyword condition. For example, if response indicator 01 is specified both for the SETOF/SETOFF keyword and the CF5 key, indicator 01 is returned in the on condition when the CF5 key is pressed. If the indicator is specified elsewhere as a response indicator, there is no need to use the SETOF/SETOFF keyword.

See the *DDS Reference* for the keywords for which an option indicator is required or for which an option indicator can be used, and for more information on the INDARA keyword.

Specifying Screen Size Condition Names

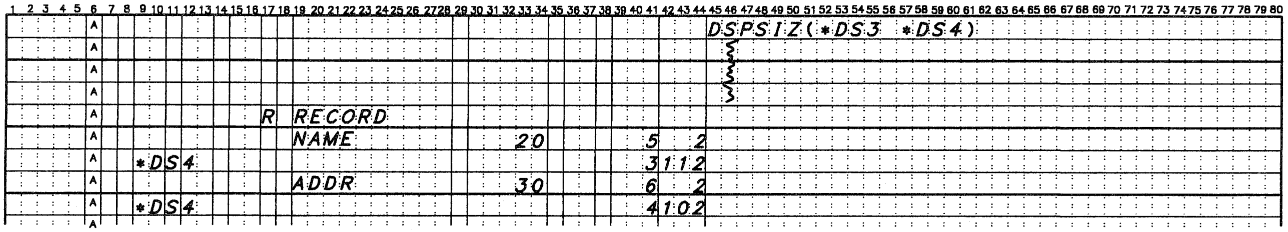
In some cases, you can use the following screen size condition names to select keywords and display locations based on screen size:

- *DS3, 24 by 80 (5251 Models 11 and 12, 5291, 5292, 3179 Model 2, 3180-2, 3196, and 3197)
- *DS4, 27 by 132 (3180-2; 3197 Models D1, D2, W1, W2; and 3477)

Note: The capability to display in 27 by 132 mode is allowed on 3180-2 and 3197 devices attached to a 6040 or 6041 local work station controller, or remotely attached to a 5294 or 5394 controller. The display size for 27 by 132 mode is ignored for the DSPSIZ keyword unless these controllers are specified.

You can also define your own screen size condition names. See the *DDS Reference* for more information.

These condition names can be used to place fields in different locations on different sizes of screens. However, the fields must still be specified in the same order on each size of screen. For example, the following DDS formats a display for both a 24 by 80 screen and a 27 by 132 screen:



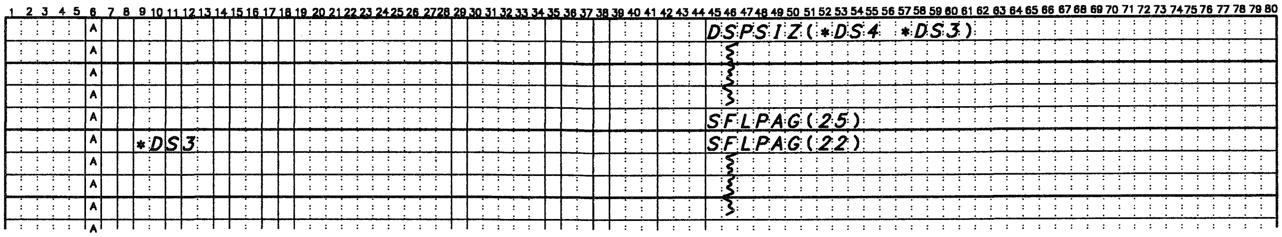
```
     A                                      DSPSIZ(*DS3  *DS4)
     A
     A
     A
     A          R RECORD
     A            NAME          20         5  2
     A  *DS4                               3112
     A            ADDR          30         6  2
     A  *DS4                               4102
     A
```

RSLH132-3

Normally, the display files are set up for a 24 by 80 screen (default size). The DSPSIZ keyword specifies which display sizes are valid for a file and indicates which sizes are the primary and secondary screen sizes. (The primary screen size is the first or only DSPSIZ value.) On the DSPSIZ keyword, the screen size can be specified as *DS3, *DS4, 24 80, or 27 132. For example, DSPSIZ (24 80) specifies a screen size of 24 by 80.

The screen size designated as the primary screen size should be the one with which the display file will most often be used. A performance benefit will be realized by coding the DSPSIZ keyword in this manner. Additional processing is performed when the actual screen size is the secondary screen size.

The screen size condition names let you improve the use of a single display file for any size screen. For example, when you are using subfiles, you can specify 22 records per page for a 24 by 80 screen or 25 records per page for a 27 by 132 screen.



```
     A                                      DSPSIZ(*DS4 *DS3)
     A
     A
     A
     A                                      SFLPAG(25)
     A *DS3                                 SFLPAG(22)
     A
     A
     A
     A
```

RSLH133-2

Specifying Function Keys

The Enter key can always be used by the work station user. However, most other function keys cannot be used unless you specify the corresponding DDS keywords in the display file. In most cases, you must specify the following keywords in order for the work station user to use the corresponding function key:

- CAnn
- CFnn
- CLEAR
- HELP (not required if you only need the Help key to retrieve the second-level text for a message on the display)
- HOME
- PRINT
- ROLLDOWN or PAGEUP (not required to be able to roll a subfile when the subfile page is not equal to the subfile size)
- ROLLUP or PAGEDOWN (not required to be able to roll a subfile when the subfile page is not equal to the subfile size)

CA and CF Keywords

The command function (CF) keys and command attention (CA) keys are numbered 1 through 24 and are the same physical set of keys on the keyboard. They can be used to set a response indicator or to tell device support to perform a certain function. When a command key is pressed, a record is returned to a program similar to when the Enter key is pressed. The difference between the two types of command keys is that CF keys return changed fields to a program while CA keys do not. If a CA key is used, the record returned to the program does not contain the data entered by the work station user and no field validation is performed.

If a key is specified as a CF key in a file, it cannot also be specified as a CA key in the file. Likewise, if it is specified as a CA key, it cannot also be a CF key. For example, if function key 01 is specified as a CA key (CA01), you cannot specify CF01 anywhere in the same file.

If a response indicator is specified for a CF key and the key is pressed, the response indicator is set on and passed to the program with the input data. If a response indicator is not specified for a CF key, only the input data is passed.

Note: The input/output feedback area contains the 1-character attention identifier (AID), which also identifies the key pressed. See Appendix A, "Feedback Area Layouts" on page A-1 for a description of the input/output feedback area.

If a response indicator is specified for a CA key and the key is pressed, the response indicator is set on and passed to the program. Fields sent to the display and hidden fields are returned to the program. If a CF key or the Enter key was pressed previously, the input-only field is returned as previously keyed data. If data was never entered into an input-only field, the field is returned as blanks (character field) or zeros (numeric field). Fields changed by the user since the last time a CF key or Enter key was pressed are not returned.

The use of CA keys can cause the input buffer of the program to contain user-entered data that does not meet the validation specified in the display file. For example, the work station user enters data and presses a CF key or the Enter key, and the data causes a validation error. Input data is processed one field at a time with data manipulation taking place before the validity checking. If a validity checking error occurs, a message is selected and all the other input data is processed. After all input data is processed and one or more errors have occurred, a message is sent to the work station user. Then, if the work station user presses a valid CA key, no changed data is sent from the display. The data is moved from the input buffer save area to the input buffer. The input buffer now contains the data that is in error. If your program is not going to process this data when the CA key is pressed, you do not have a problem. If this is a problem, avoid using CA keys; only use CF keys so that invalid data can be detected. If you want to use CA keys and expect to process data from the display, you should not specify the following validity checking DDS keywords:

CHECK(M10)
CHECK(M11)
CHECK(VN)
CHECK(VNE)
COMP/CMP
RANGE
VALUES

The Print, Help, Clear, and Home keys operate in the same manner as the CA keys. The Roll Up, Roll Down, Page Up, and Page Down keys operate in the same manner as CF keys.

Specifying Alternative Help, Page Up, and Page Down Keywords

You can also define command attention or command function keys to perform the functions of the Help, Page Up (or Roll Down) and Page Down (or Roll Up) keys. The function key specified on the keyword identifies the alternative key to be used.

The DDS keywords are:

- **ALTHELP**: Indicates that the help function will be started when either the Help key or the key specified on the ALTHELP keyword is pressed. If the ALTHELP keyword is specified but an alternative key is not specified, the default is CA01. Note that the Help key is an attention key, not a function key, because it does not return input.
- **ALTPAGEUP** and **ALTPAGEDWN**: Indicate that the paging functions will be started when the page keys or the keys specified on the keywords are pressed. If alternative keys are not specified on the ALTPAGEUP or ALTPAGEDWN keywords, the defaults are CF07 and CF08, respectively. Note that the page keys are function keys, because they return input.

The alternative keys specified on the ALTHELP, ALTPAGEUP, and ALTPAGEDWN keywords provide the same function as the actual keys. For example, if pressing the Help key starts the help function, then pressing the alternative key defined by the ALTHELP keyword will also start the help function. Likewise, if pressing the Page Up or Page Down key returns control to the application program, then pressing the alternative key will also return control to the application program. Both of these examples appear to the program as if the actual key was pressed.

The user profile option for scrolling (USROPT(*ROLLKEY)) applies to the PAGEUP, PAGEDOWN, ALTPAGEUP, ALTPAGEDWN, ROLLUP, and ROLLDOWN keywords.

The alternative help key function does not work when the keyboard is locked. For example, if you type information into a field that is not input-capable, a controller-detected error occurs and flashing numbers appear. The Help key can be used to get more information about the error. The function key specified as the alternative help key will not be valid until the Reset key is pressed, and then the help information will no longer be available.

Command Key Validity Considerations

The following lists command key considerations:

- A write or write-read operation for which no input or output is sent to the display does not affect which keys are valid. Examples of such operations are a write operation or an update operation to a subfile record format.
- If a write or write-read operation displays a message by selecting either the ERRMSG or ERRMSGID keyword, the command keys in effect on that output operation are valid. Therefore, you can specify a different set of command keys to be valid if an error occurs.
- If only one subfile record format is displayed and the subfile control record format specifies a CA or CF key for the SFLDROP keyword, that key remains valid for that function as long as the subfile is still on the display. In addition, the key specified for the SFLENTER keyword remains valid until another write or write-read operation does not specify the key as being valid.
- If two subfile record formats are displayed and both specify the SFLDROP keyword, only the last SFLDROP keyword is used. There can only be one drop key at a time.
- The following keys are valid even though they might not be specified in the display file:
  - Roll Up and Roll Down keys, which are valid any time device support can roll a displayed subfile
  - Help key, which is valid any time device support recognizes a message on the display for which there is help

Active Record Format Table

In the following discussion about some of the DDS keywords it helps to keep in mind that the system maintains a table of which record formats are currently on the display. Read operations may take place against only those record formats that are in this table. Certain DDS keywords such as OVERLAY, ERASE, and CLRL cause records in this table to altered.

A record format is added to the table when a write operation with this record format is performed. A record format will be removed from this table when a read operation would no longer be correctly performed for this record. For example, a record may be erased from the display when another record is written to it that has both the ERASE and OVERLAY keywords specified.

This table is cleared whenever the display is cleared, for example, when a write of a record that does not specify OVERLAY is performed.

Placing Records on the Display

One record format can occupy an entire screen or the screen can be divided to display more than one record format. If a record is displayed on more than one line, the lines must be consecutive lines on the display. For example, if one record occupies two lines and the record begins on line 2, the record must continue on line 3. In addition, when the CLRL keyword is not used, two records from different record formats cannot be displayed on the same line. However, if only the ending attribute character for the last field in a record is in position 1 of the next line, another record format can begin on that same line. (See "Field Attribute Characters" on page 4-67 for more information.) Only a beginning attribute character can occupy line 1, position 1.

Figure 4-7 shows how a screen can be divided. Figure 4-8 on page 4-85 shows invalid divisions of the screen.

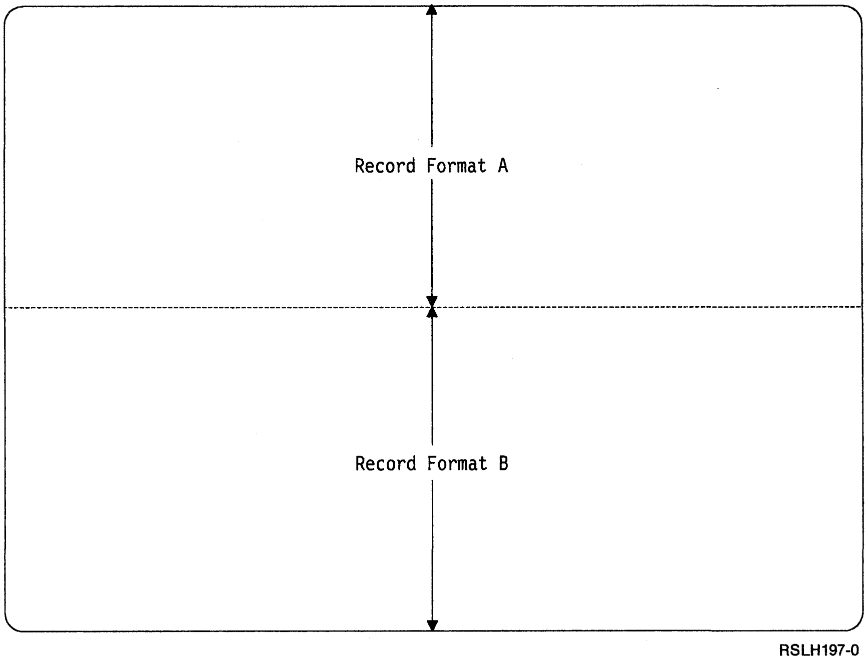


Figure 4-7 (Part 1 of 3). Valid Placement of Records on a Screen when the CLRL Keyword Is Not Used

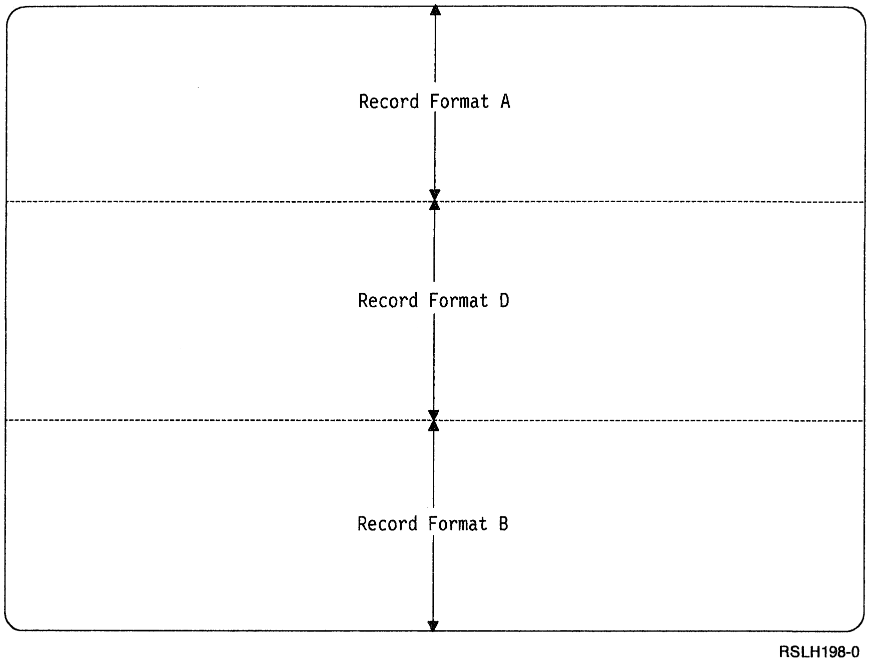


Figure 4-7 (Part 2 of 3). Valid Placement of Records on a Screen when the CLRL Keyword Is Not Used

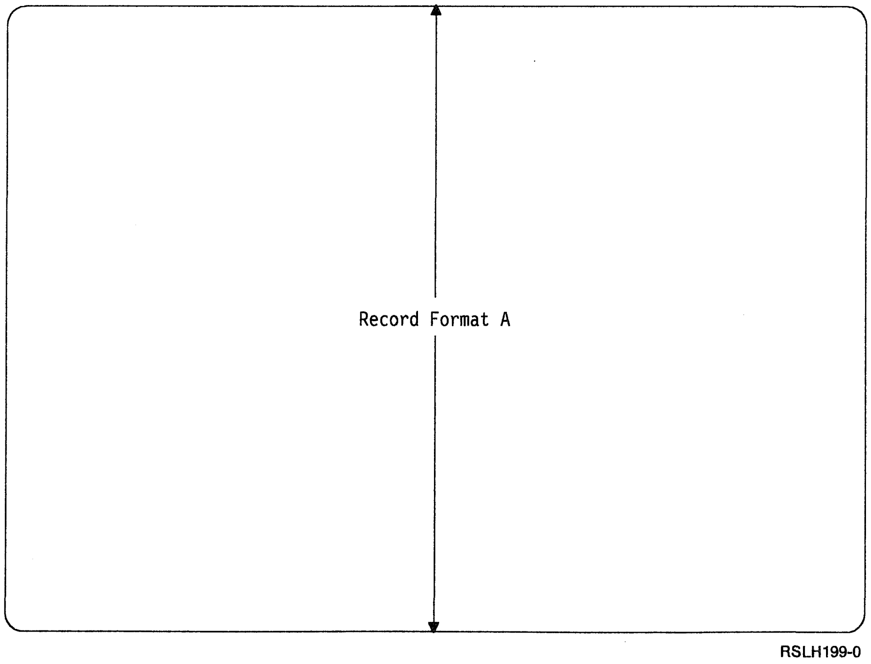


Figure 4-7 (Part 3 of 3). Valid Placement of Records on a Screen when the CLRL Keyword Is Not Used

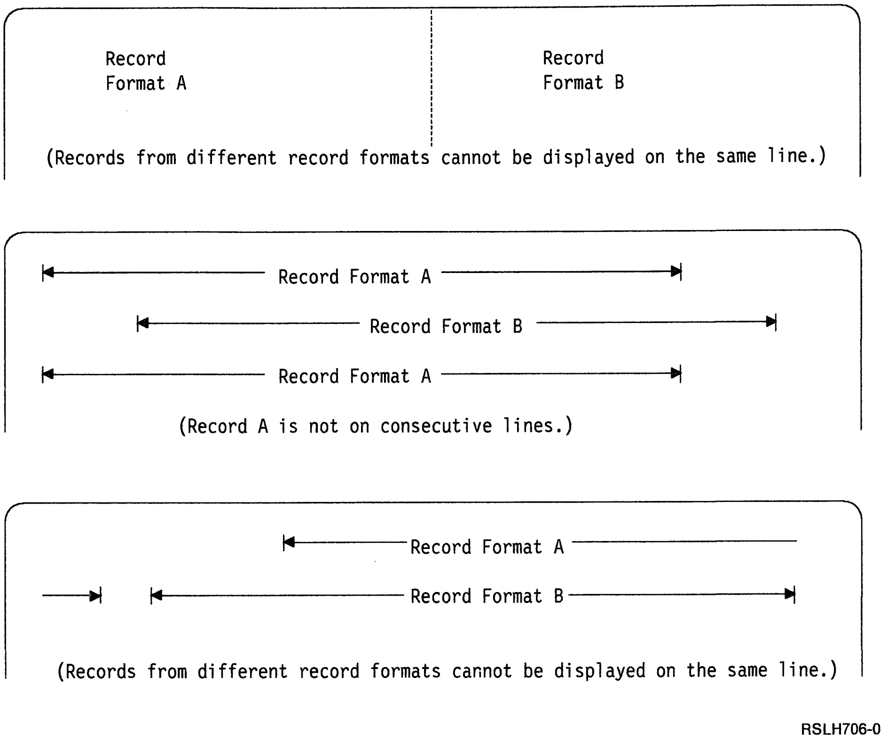

Record
Format A
Record
Format B
(Records from different record formats cannot be displayed on the same line.)
Record Format A
Record Format B
Record Format A
(Record A is not on consecutive lines.)
Record Format A
Record Format B
(Records from different record formats cannot be displayed on the same line.)
RSLH706-0

Figure 4-8. Invalid Placement of Records on a Screen when the CLRL Keyword Is Not Used

The following types of records do not occupy space on a display:

- Records with no fields defined
- Records with only hidden, program-to-system, or message fields
- Records with the CLRL keyword specified and with no input-capable fields

These records are assumed to be at line 0. At any one time, the system only keeps track of one such record. If an output operation for a record assumed to be at line 0 replaces another record assumed to be at line 0, you can no longer issue an input operation for the replaced record.

The formats displayed can change while a file is being processed because information on a display can be deleted when new formats are displayed. For example, fields from record format A occupy lines 1 through 4, fields from record format B occupy lines 5 through 7, and fields from record format C occupy lines 8 through 10. Figure 4-9 on page 4-86 shows what happens when record format D, which occupies lines 5 through 9, is displayed when record formats A, B, and C were previously displayed. (The OVERLAY, PUTOVR, or CLRL keyword must have been specified for record format D; otherwise, record format A would be deleted also.) As a special case, if record format D had the CLRL(*NO) keyword in effect and occupied lines 5 through 9, lines 5 through 7 of record format B would remain on the display (see "Using the Clear Line (CLRL) Keyword" on page 4-95 for more information). Any data in the fields of record format B would be changed if those fields are overlaid by fields in record format D.

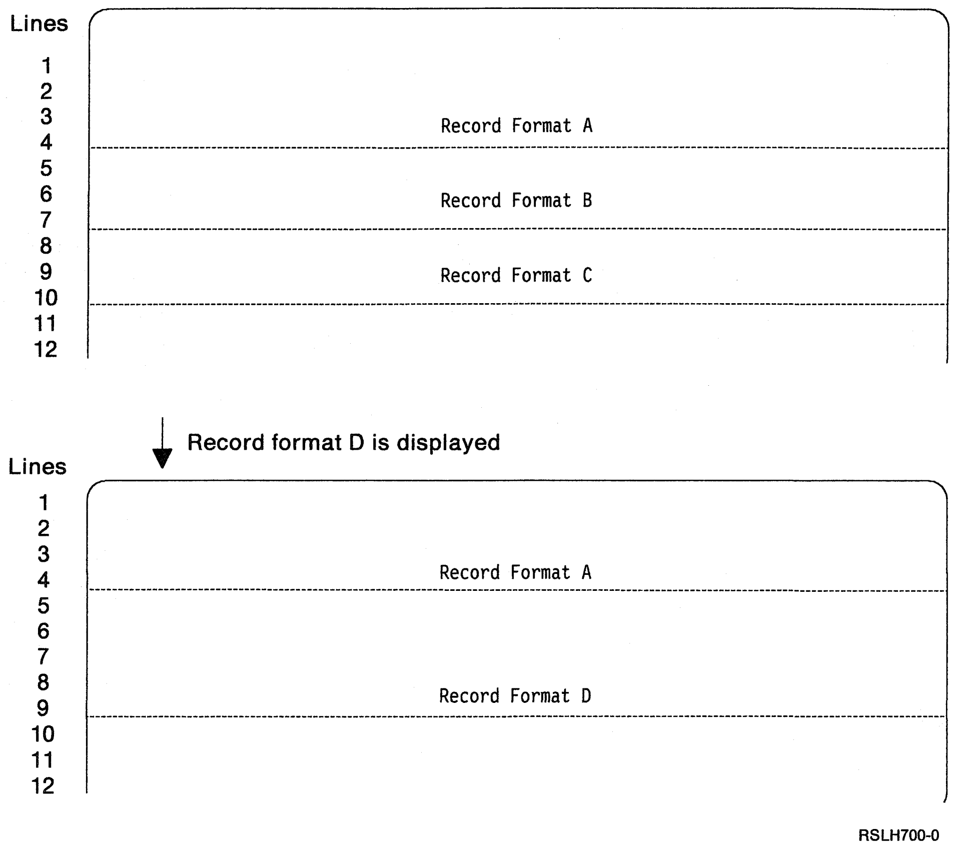


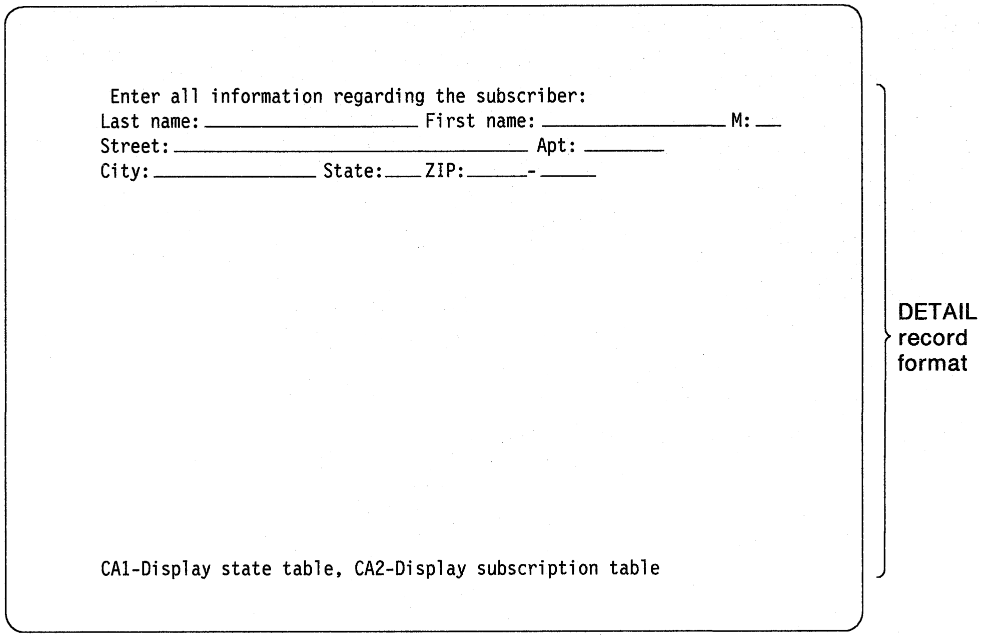
Figure 4-9. Replacing Record Formats

Records containing input fields should be sent to the display device in the order in which they appear on the display. For example, in Figure 4-9 both record formats A and B contain input fields and appear on the same display. Record format A is displayed on lines 1 through 4, and record format B is displayed on lines 5 through 7. Record format A should be sent to the display first. This technique provides better performance than if record formats with input fields are sent randomly or in some other order.

Positioning Record Formats

Because of the characteristics of the 5250 display stations, certain record format positioning and operational combinations can produce undesirable results. The following example illustrates a combination that can cause undesirable results and explains how to avoid these results.

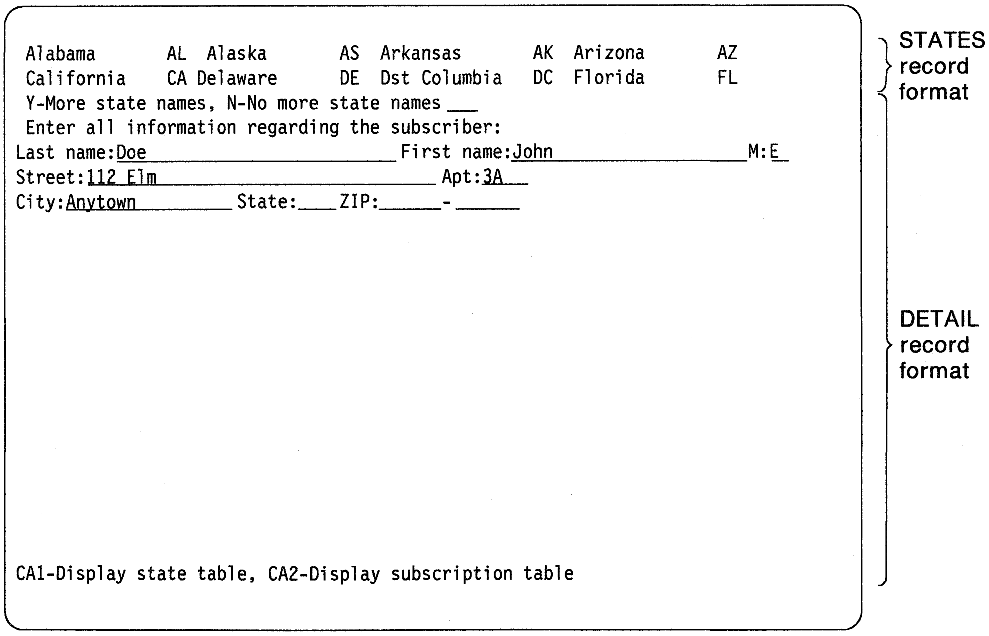
The displays produced and the DDS for the record formats follow:



```
  Enter all information regarding the subscriber:
 Last name:____________________ First name:_________________ M:__
 Street:____________________________________ Apt: _______
 City:_______________ State:___ZIP:______-_____




















 CA1-Display state table, CA2-Display subscription table
```

DETAIL record format

RSLH166-0

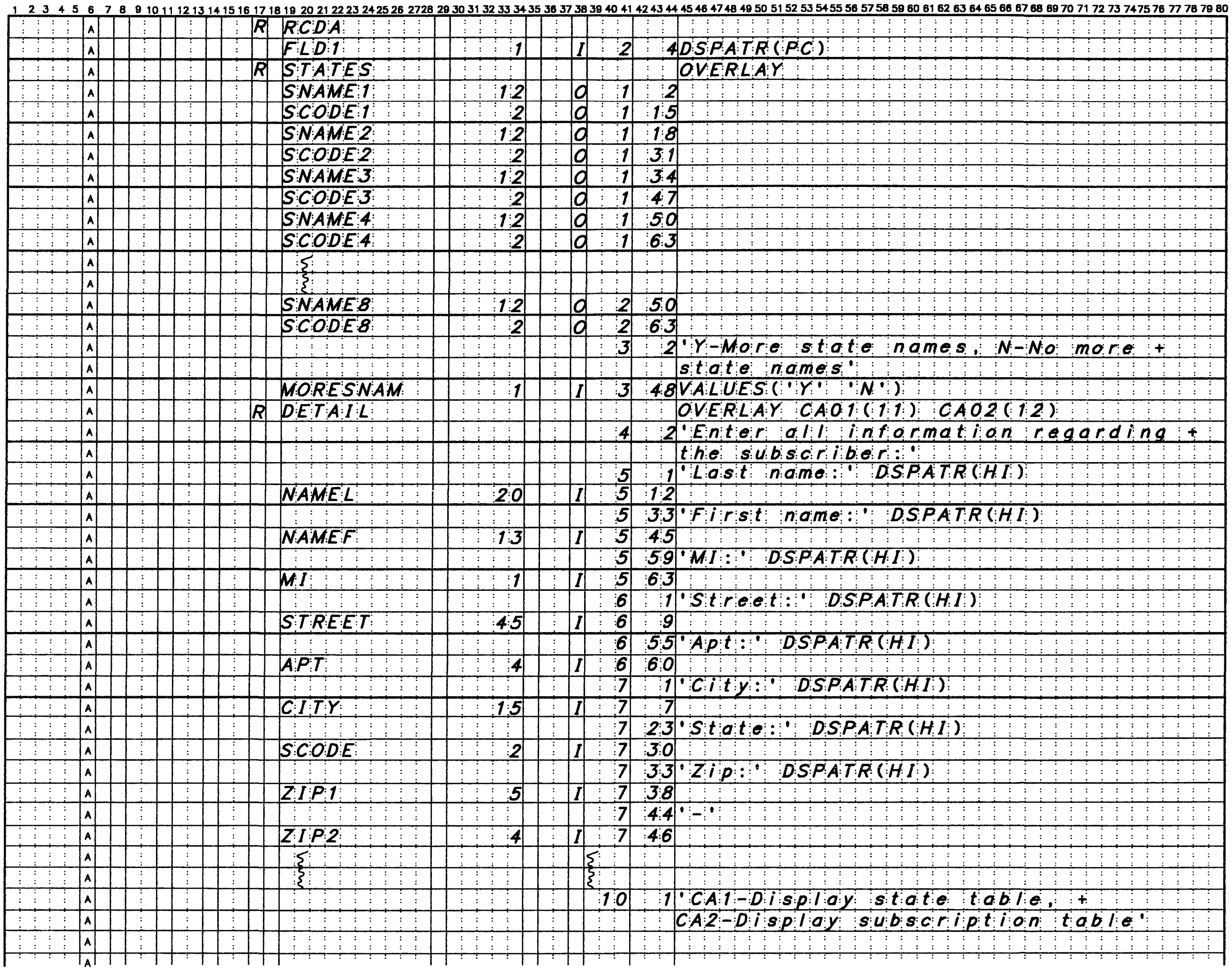


```
 Alabama       AL  Alaska       AS  Arkansas      AK  Arizona      AZ
 California    CA Delaware      DE  Dst Columbia  DC  Florida      FL
 Y-More state names, N-No more state names ___
 Enter all information regarding the subscriber:
Last name:Doe__________________First name:John_________________M:E_
Street:112 Elm______________________ Apt:3A__
City:Anytown________ State:___ZIP:______-______

















CA1-Display state table, CA2-Display subscription table
```

STATES record format

DETAIL record format

RSLH169-0



```
     A          R RCDA
     A            FLD1           1   I  2  4DSPATR(PC)
     A          R STATES                    OVERLAY
     A            SNAME1        12   O  1  2
     A            SCODE1         2   O  1 15
     A            SNAME2        12   O  1 18
     A            SCODE2         2   O  1 31
     A            SNAME3        12   O  1 34
     A            SCODE3         2   O  1 47
     A            SNAME4        12   O  1 50
     A            SCODE4         2   O  1 63
     A
     A
     A            SNAME8        12   O  2 50
     A            SCODE8         2   O  2 63
     A                                  3  2'Y-More state names, N-No more +
     A                                        state names'
     A            MORESNAM       1   I  3 48VALUES('Y' 'N')
     A          R DETAIL                    OVERLAY CA01(11) CA02(12)
     A                                  4  2'Enter all information regarding +
     A                                        the subscriber:'
     A                                  5  1'Last name:' DSPATR(HI)
     A            NAMEL         20   I  5 12
     A                                  5 33'First name:' DSPATR(HI)
     A            NAMEF         13   I  5 45
     A                                  5 59'MI:' DSPATR(HI)
     A            MI             1   I  5 63
     A                                  6  1'Street:' DSPATR(HI)
     A            STREET        45   I  6  9
     A                                  6 55'Apt:' DSPATR(HI)
     A            APT            4   I  6 60
     A                                  7  1'City:' DSPATR(HI)
     A            CITY          15   I  7  7
     A                                  7 23'State:' DSPATR(HI)
     A            SCODE          2   I  7 30
     A                                  7 33'Zip:' DSPATR(HI)
     A            ZIP1           5   I  7 38
     A                                  7 44'-'
     A            ZIP2           4   I  7 46
     A
     A
     A                                 10  1'CA1-Display state table, +
     A                                        CA2-Display subscription table'
     A
     A
```

RSLH125-1

Assume that the DETAIL record format is on the screen, and the work station user is entering data for a subscriber. Because the user does not know the state code for the state to be entered, he presses the CA01 key.

Because CA01 is defined as a CA key, no data is transmitted to display device support when the CA01 key is pressed. The data, however, remains on the screen. The program detects that the CA key was pressed because response indicator 11 is set on. The program then displays the STATES record format.

Because the STATES record format is physically above the DETAIL record format, the system must resend the field attributes for the input fields in the DETAIL record format. (The system would also resend the field attributes if the STATES record format contained only output-only fields and was replacing the RCDA record format. In this case, the field attributes are resent because a record format (RCDA) with a specific cursor location is being removed.)

The following problems occur because display device support resends the field attributes for the DETAIL record format:

1. All the input fields in the DETAIL record format lose their modified data tags (MDTs). When the program does the next read to the DETAIL record format (for example, when the work station user presses the Enter key), none of the fields typed in before the user pressed the CA01 key are returned to the program. The program cannot retrieve that typed data even though the data still remains on the screen.
2. The highlight attribute for the constant fields (except Last name) is lost. Display device support does not resend the field attributes for output-only fields. However, if an output-only field immediately follows an input-capable field so that the leading attribute character for the output field is in the same position as the ending attribute character for the input-capable field, the attribute of the output field reverts to normal.

Note: Display device support needs to resend the attributes for the input-capable fields when a subfile is rolled from a full page to a partial page, a partial page to a full page, or a partial page to a partial page. The two problems mentioned above may also occur when resending the field attributes.

To avoid problem 1:

- Avoid using a CA key.
- If you must use a CA key, avoid writing the format containing the CA key to the screen and then writing another format that is physically placed above the first format if both formats contain input-capable fields.
- Avoid writing a format to the screen that causes the removal of a format containing a cursor location specification.

To avoid problem 2:

- Do not specify an output-only field with special display attributes immediately following an input-capable field.
- If you must specify an output-only field with special display attributes immediately following an input-capable field, avoid writing that format to the screen and then writing another format that is physically placed above the first format if both formats contain input-capable fields.
- Avoid writing a format to the screen that causes the removal of a format containing a cursor location specification.

Controlling Display Functions on Output Operations

When your program displays a new record format for output or to allow input, the existing display is usually erased before the new record format is displayed. For example, if three record formats are on the display at the same time and you write another record to the display, those three would be erased. DDS provides you with several keywords that allow you to control the displaying of records and fields on input and output operations.

On an output operation, you can control certain functions of the display before a new record format is displayed. The following lists those functions (see "Write Operation" on page 4-24 for more information on output operations).

- Overlaying a display but not erasing the entire display (OVERLAY keyword). Only records that are completely or partially overlapped are erased; all other records remain on the display.
- Preventing an overlapped record format from being erased when the overlapping record format is written to the display (CLRL(*NO)) or clearing a specified number of lines (CLRL(nn), CLRL(*END), and CLRL(*ALL)). When you used the CLRL(*NO) keyword, only those lines of the display that are used by the overlapping record format are erased; the other lines remain on the display. With the CLRL(nn) keyword, the number of lines you specify are cleared before the record is written to the screen. If you use the CLRL(*END) keyword, all lines below the starting line number are cleared. If you use CLRL(*ALL), all lines are cleared. (For more information on the CLRL keyword, see "Using the Clear Line (CLRL) Keyword" on page 4-95.)
- Changing the attributes or the content of a field in a record that is currently displayed on a subsequent write operation for the same record. When the PUTOVR keyword is specified, the display attributes are overridden for those fields with the OVRATR keyword in effect. Similarly, the data content is overridden for the fields for which the OVRDTA keyword is in effect. When the PUTOVR keyword is in effect, the output operation functions as if the OVERLAY keyword were also in effect, even if the OVERLAY keyword is not specified. The PUTOVR keyword cannot be specified in a record format that contains the PUTRETAIN keyword nor can it be used for subfile records.

  The display attributes that can be overridden by the OVRATR keyword are:

CHECK(ER)	End of Record
CHECK(ME)	Mandatory enter
DSPATR(MDT)	Set on modified data tag
DSPATR(PR)	Protect
DSPATR(BL)	Blink
DSPATR(CS)	Column separator
DSPATR(HI)	High intensity
DSPATR(ND)	Nondisplay
DSPATR(PC)	Position cursor
DSPATR(RI)	Reverse image
DSPATR(UL)	Underline
DUP	Dup key capable

- Erasing specific records on a display before displaying the next record (ERASE keyword), when the OVERLAY keyword is in effect. The ERASE keyword can only be used with the OVERLAY keyword.

- Erasing all unprotected input and output/input fields on the display before displaying the next record (ERASEINP keyword). (Fields are protected using the DSPATR(PR) keyword.) The ERASEINP keyword can only be used with the OVERLAY keyword. ERASEINP(*MDTON) will erase all unprotected input-capable fields that have their modified data tags on. ERASEINP(*ALL), will erase all unprotected input-capable fields whether their modified data tags are on or not.

  You can use ERASEINP to improve response time because it causes the display to clear fields instead of requiring blanks to be sent to the display. If the fields erased at the display do not have their modified data tags set on for the next read operation, data is returned for those fields from the input save area. This is data saved by the system from the previous return of the field from the device.

  You can use the INZINP keyword at the record level with ERASEINP(*ALL) and PUTOVR to initialize the input save area without sending data for the cleared fields to the display.

- Resetting the modified data tags associated with records already on the display before displaying this next record (MDTOFF keyword). The MDTOFF keyword can only be used with the OVERLAY keyword. MDTOFF is processed before this next record is displayed. If you specify MDTOFF(*UNPR), only the modified data tags of the unprotected fields are reset. If you specify MDTOFF(*ALL), the modified data tags of all input-capable fields are reset.

- Protecting all input-capable fields remaining on the display before displaying the next record (PROTECT keyword). All input-capable fields are changed to output-only fields. The records containing these fields cannot be read by the program until they are written by the program to the display again. The PROTECT keyword can only be used with the OVERLAY keyword.

- Retaining a record or field on a display (PUTRETAIN keyword) so that it does not have to be retransmitted to the display. The PUTRETAIN keyword can only be used with the OVERLAY keyword. The PUTRETAIN keyword can be used to change only the display attributes of a field. Except for not sending data, all other functions are supported when the PUTRETAIN keyword is specified.

  Using the PUTRETAIN keyword at either the record format level or the field level can cause fields from this record which were previously written to the display to remain on the display even if they are not selected for this write operation. To avoid this, you can use the PUTRETAIN keyword at the field level and define the field twice: once with option indicators as you want it to appear in the display, and once with no option indicators and as a constant with a value of blanks. If the first field is not selected, the second field is. The second field is displayed so the blanks erase the contents of the unselected field.

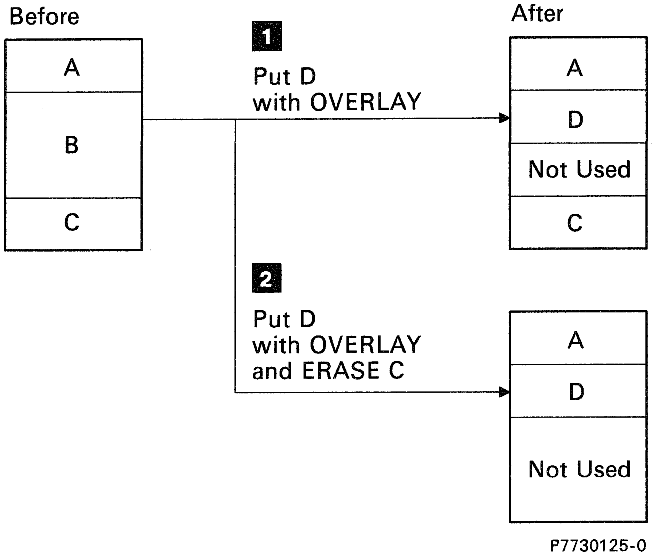
  Note: The ERRMSG and ERRMSGID keywords function as if the PUTRETAIN keyword were specified at the record format level. That is, no fields are sent to the display, no field attributes for other fields are changed, and no command keys are changed when the ERRMSG and ERRMSGID keywords are in effect.

- Locking the keyboard until the last record is displayed so that data cannot be entered into input fields (LOCK keyword).

Note: When you use the CLRL, OVERLAY, PUTOVR, PUTRETAIN, ERRMSG, or ERRMSGID keyword, you should specify RSTDSP(*YES) on the Create Display File or Change Display File command; otherwise, data on the display can be lost if the file is suspended.

Using the OVERLAY and ERASE Keywords

The following diagram shows the effect of the OVERLAY and ERASE keywords on an output operation:



1 Record format B is erased because record format D overlaps it, and record format D is displayed. Record format D did not use all of the space record format B previously used so it does not overlap record format C.

2 Record format B is erased because record format D overlaps it and record format C is erased because ERASE C is specified. Record format D is displayed, and part of the display is no longer being used.

Using the SLNO Keyword

You can specify a starting line number for your record formats by using the DDS record-level keyword SLNO. On the SLNO keyword, you can specify the actual starting line number for the record format (a value from 1 to 27), or you can specify a variable starting line number (*VAR) and the starting line number value in your high-level language program at run time.

When you use SLNO(nn), the system adjusts the line numbers for all fields in a record by the specified value minus 1.

If you specify SLNO(*VAR), you must specify the value for the starting line number in your high-level language program. If this value is 0 or if the value is not specified, a starting line number of 1 is assumed. If the value exceeds the number of lines on the screen or if it is a negative value, the system sends a message to the program and the I/O request is not performed. If you specify SLNO(*VAR) and the starting location for the field for at least one display size is row 1, column 1, a warning message (severity 10) is issued at file creation time. At run time, an error message is issued if the screen size being displayed contains a field starting in row 1, column 1, and the variable starting line number is set to 1 by the program.

If you specify SLNO(*VAR), the system adjusts the line numbers for each field in the record format by that value minus 1. If the resulting line number exceeds the screen size, the field is not displayed. In addition, if any part of a field goes beyond the last line on the screen, the field is not displayed.

If the CLRL(nn) keyword is also specified, clearing starts with the starting line number. The starting line number for a record format is determined as follows:

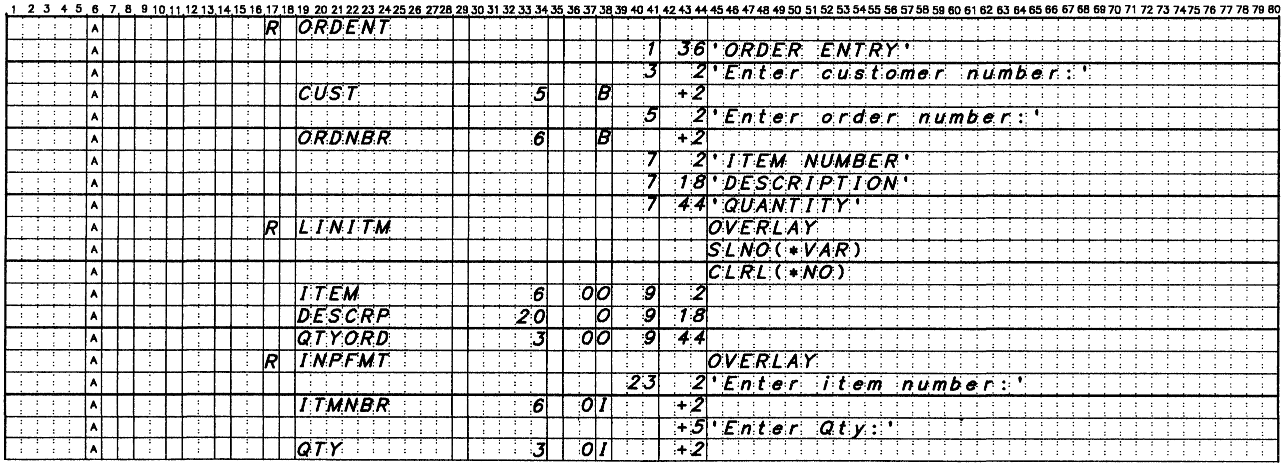
- If the SLNO keyword is not specified, the field locations determine the starting line number.
- If the SLNO(nn) keyword is specified, nn is the starting line number.
- If the SLNO(*VAR) keyword is specified, the starting line number defaults to 1 at creation time and is changed at run time by the value specified in the program.

If you use the SLNO(*VAR) keyword with the OVERLAY keyword but without the CLRL keyword and then write the record several times, each time with a different starting line number, the previous record is deleted before the new record is displayed.

If the SLNO keyword is used with the PUTOVR, PUTRETAIN, ERRMSG, or ERRMSGID keyword in effect, the system checks the starting line number to determine if the previous output operation to the record had the same starting line number. If it did, the action specified by the PUTOVR, PUTRETAIN, ERRMSG, or ERRMSGID keyword is performed. If the starting line numbers are not the same, the PUTOVR, PUTRETAIN, ERRMSG, or ERRMSGID keyword is ignored, and the record format is displayed with the lines adjusted by the new value.

The SLNO keyword cannot be used in a record format that contains the record-level keywords ASSUME, KEEP, USRDFN, SFL, or SFLCTL, or in a display file that contains the file level keyword PASSRCD.

The following DDS shows an example of using the SLNO(*VAR) keyword:



```
     A          R ORDENT
     A                                  1 36'ORDER ENTRY'
     A                                  3  2'Enter customer number:'
     A            CUST           5   B    +2
     A                                  5  2'Enter order number:'
     A            ORDNBR         6   B    +2
     A                                  7  2'ITEM NUMBER'
     A                                  7 18'DESCRIPTION'
     A                                  7 44'QUANTITY'
     A          R LINITM                    OVERLAY
     A                                      SLNO(*VAR)
     A                                      CLRL(*NO)
     A            ITEM           6  0O  9  2
     A            DESCRP        20   O  9 18
     A            QTYORD         3  0O  9 44
     A          R INPFMT                    OVERLAY
     A                                 23  2'Enter item number:'
     A            ITMNBR         6  0I    +2
     A                                    +5'Enter Qty:'
     A            QTY            3  0I    +2
```

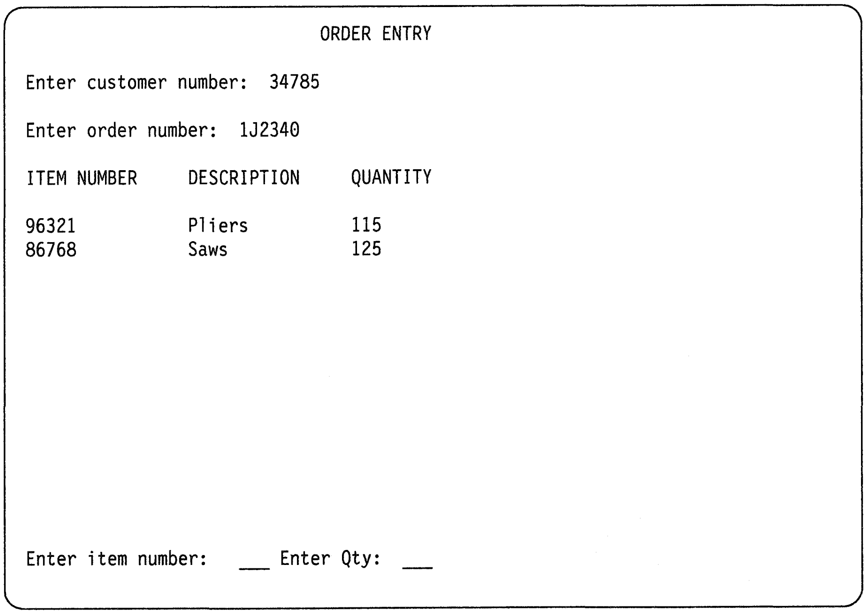
RSLH126-0

In this example, the record format ORDENT contains the prompt for an Order Entry display. The work station user enters a customer number and an order number. The program then writes the record format INPFMT to the display, which allows the work station user to enter an item number and quantity ordered. After the work station user enters the item number and the quantity, the program retrieves the description of the item from a file and writes the record format LINITM to the display. The program then writes the INPFMT record format to the display to allow the work station user to enter another item number.

The design of this display allows the work station user to enter the item number and quantity on the same line. As a line item is entered, the program uses the LINITM record format to build the order on the display. The SLNO(*VAR) keyword is used so the program can add a line to the display each time the LINITM record format is written. The CLRL(*NO) keyword has to be specified on the LINITM record format so that the previous record is not deleted when a new record is written.

When the LINITM record format is first written to the display, the value of *VAR is 1 so the fields are displayed on line 9. On each successive output operation to this record format, the program adds 1 to the starting line number so that a new line item is added to the display.

After the work station user enters two item numbers and quantities, the display looks like this:



```
                             ORDER ENTRY

Enter customer number:  34785

Enter order number:  1J2340

ITEM NUMBER     DESCRIPTION     QUANTITY

96321           Pliers          115
86768           Saws            125











Enter item number:   ___ Enter Qty:  ___
```

The use of the SLNO keyword is most efficient when you want the work station user to always enter data on the same line and yet build a display of previously entered records, as shown in the preceding example. However, for a typical inquiry function where you want to display more than one record at a time, the use of a subfile is more efficient.

Using the Clear Line (CLRL) Keyword

You can use the DDS keyword CLRL(nn) to specify that a certain number of lines be cleared on the screen before you write a record format to the screen. You can also specify CLRL even when the record contains no fields that are displayed. Clearing begins with the starting line number, and the value specified on the CLRL keyword determines the number of lines to be cleared (nn can be any value from 1 to 27). The starting line number is determined as follows:

- If the SLNO keyword is not specified, the field locations determine the starting line number.
- If the SLNO(nn) keyword is specified, nn is the starting line number.
- If the SLNO(*VAR) keyword is specified, the starting line number defaults to 1 at the time the display file is created and can be changed by the application program at the time it is run.

You can use the CLRL(*END) keyword to clear all lines from the starting line to the end of the display. You can also use the CLRL(*NO) keyword to prevent an overlapped record from being erased at all. You use the CLRL(*ALL) keyword to clear all the lines of the display. Since the default action is to clear all the lines of the screen, you do not normally have to specify CLRL(*ALL) unless you also specify a DDS keyword, such as USRDSPMGT, that changes this default.

Note: When you use the CLRL keyword, you should specify RSTDSP(*YES) on the CRTDSPF or CHGDSPF command; otherwise, data on the display may be lost if the file is suspended.

If the CLRL(nn) keyword is specified in a record format with input-capable fields, any input-capable fields in the overlapped records are no longer input-capable. Fields in all other record formats that are not overlapped remain input-capable. If you do not want these fields to remain input-capable, you should use the PROTECT keyword on the record format along with the CLRL(nn) keyword.

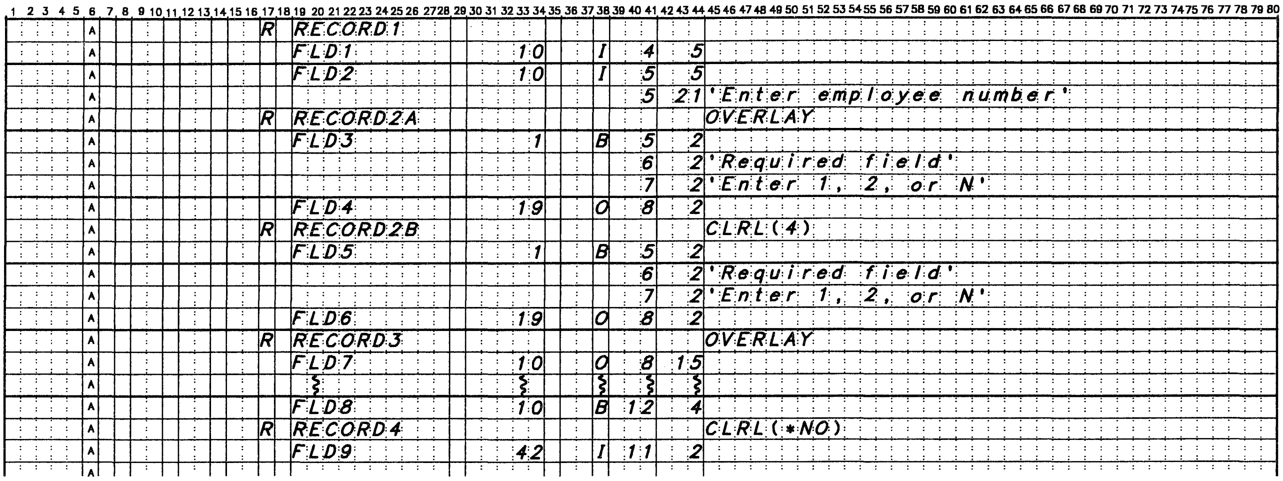
Records with the CLRL keyword and with no input-capable fields are assumed to be at line 0. Thus, if the CLRL(nn) keyword is specified in a record format that has no input-capable fields, all records already on the display remain on the display and their input-capable fields remain input-capable. Because records which start at line 0 are not known to the system, the ROLLUP and ROLLDOWN keywords do not work for these records. Also, these records may not be cleared completely when they are overlapped by other records that have the OVERLAY keyword specified. The lines needed for the overlapping record are cleared whereas the lines not needed for the overlapping record remain on the screen.

The CLRL(nn) keyword is not allowed in a record format with the record-level keywords ASSUME, KEEP, USRDFN, SFL, or SFLCTL, or in a display file with the file level keyword PASSRCD.

The CLRL(nn) keyword is ignored if either the ERRMSG or ERRMSGID keyword is in effect.

If the CLRL(nn) keyword is used and the PUTOVR or PUTRETAIN keyword is in effect, the clearing of lines may conflict with the PUTOVR or PUTRETAIN function. The PUTOVR or PUTRETAIN keyword requires that the fields being overridden be on the display whereas the CLRL(nn) keyword may clear those fields first. If a record becomes unavailable for input because of the CLRL(nn) keyword, the input-capable fields remain input-capable if the PUTOVR keyword is in effect. However, the system issues a message if the program attempts to read such a record.

Although the CLRL(nn), CLRL(*NO), and CLRL(*END) keywords imply the OVERLAY keyword, the following example illustrates the differences between the CLRL and OVERLAY keywords:



```
     A          R RECORD1
     A            FLD1          10   I  4  5
     A            FLD2          10   I  5  5
     A                                  5 21'Enter employee number'
     A          R RECORD2A                  OVERLAY
     A            FLD3           1   B  5  2
     A                                  6  2'Required field'
     A                                  7  2'Enter 1, 2, or N'
     A            FLD4          19   O  8  2
     A          R RECORD2B                  CLRL(4)
     A            FLD5           1   B  5  2
     A                                  6  2'Required field'
     A                                  7  2'Enter 1, 2, or N'
     A            FLD6          19   O  8  2
     A          R RECORD3                   OVERLAY
     A            FLD7          10   O  8 15
     A             ≀             ≀   ≀  ≀  ≀
     A            FLD8          10   B 12  4
     A          R RECORD4                   CLRL(*NO)
     A            FLD9          42   I 11  2
     A
```

RSLH127-0

1. If the program performs the output operations on the record formats in the following order:

   RECORD1
   RECORD3
   RECORD2A

   Lines 4 through 12 are deleted when RECORD2A is written to the display because RECORD2A overlaps RECORD1 and RECORD3, and only the OVERLAY keyword is specified for RECORD2A.

2. If the program performs the output operations on the record formats in the following order:

   RECORD1
   RECORD3
   RECORD2B

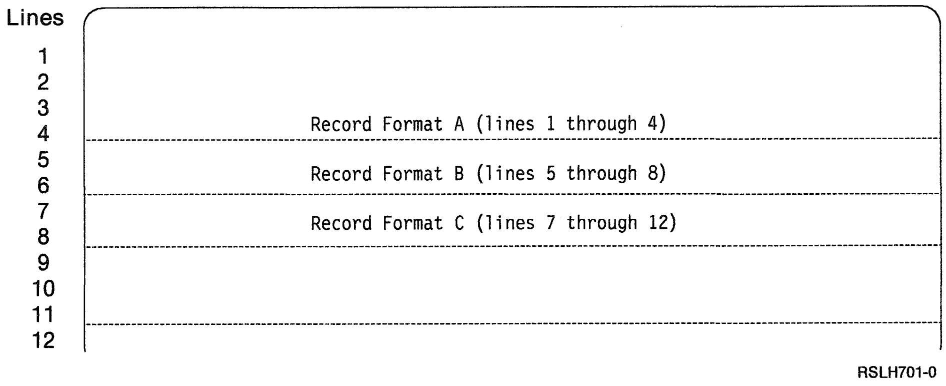
   Lines 5 through 8 are cleared before RECORD2B is written to the display because the CLRL(4) keyword is specified. FLD1 in RECORD1 and any input-capable fields in RECORD3 (lines 9 through 12) remain on the screen but are no longer input-capable because part of RECORD1 and RECORD3 is overlapped by RECORD2B.

3. If the program performs the output operations on the record formats in the following order:

   RECORD1
   RECORD3
   RECORD4
   RECORD2A

   RECORD1 remains on the screen when RECORD3 is written to the screen because the OVERLAY keyword is specified in RECORD3. When RECORD4 is written to the screen, it uses part of line 11, which is also used by RECORD3, and because CLRL(*NO) is specified in RECORD4, RECORD3 remains on the screen. However, the system is no longer aware that RECORD3 is on the screen so when RECORD2A is written, only lines 4 through 8 are cleared; the part of RECORD3 below line 8 remains on the screen.

You can use the CLRL(*NO) keyword to prevent an overlapped record from being deleted when the overlapping record is written to the display. If you use this keyword, any records being displayed that are to be overlapped are not deleted from the screen; the new record overlays them entirely or partially. There is a performance advantage to using CLRL(*NO) if you have a display that contains constants and data that is repeatedly sent to the screen. By sending the constants as a separate format and using CLRL(*NO) for the format containing only the data, you can reduce the time required to send the record format to the display. For example:


Lines
1
2
3
4
5
6
7
8
9
10
11
12
Record Format A (lines 1 through 4)
Record Format B (lines 5 through 8)
Record Format C (lines 7 through 12)
RSLH701-0

If CLRL(*NO) is specified on record format C, all fields of record format B not overlapped by C remain on the screen when record format C is written to the screen. If the OVERLAY or PUTOVR keyword were used for this same situation, record format B would be deleted when record format C is written to the screen because record format C overlaps record format B.

Using the ALWROL Keyword

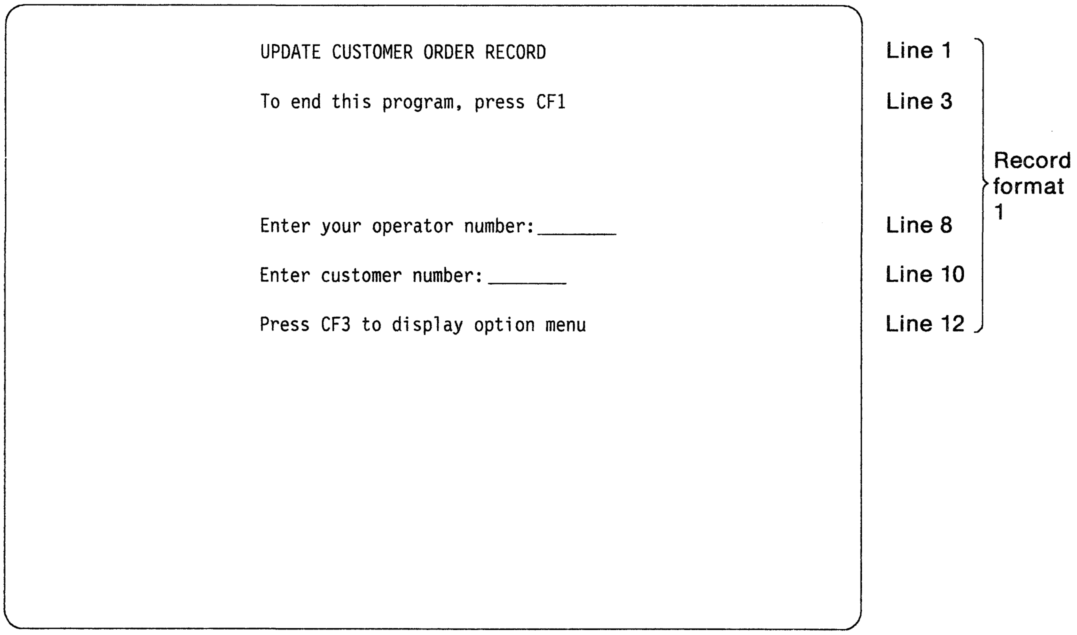
The allow roll (ALWROL) keyword provides the support for high-level language programs to roll the data between two lines on the display up or down. The lines vacated by the rolled data are set to nulls and another record format can be written to those lines.

In your program, you must specify the starting line number and the ending line number of the lines to be rolled, the number of lines to be rolled, and whether the roll is to be up or down. If the number of lines to be rolled is positive, the data is rolled up. If the number of lines to be rolled is negative, the data is rolled down. The start and end line numbers define a window on the screen.

In this window, the lines of data are rolled up (or down) by the number of lines you specified in your program. The data rolled off the window is gone. The input-capable fields of any record format partially or completely within the window are no longer input-capable. After the roll, your program cannot issue an input operation to any record format within the window. If you attempt to issue an input operation to one of these record formats, the system issues message CPF5022.

The following example shows a display before a program-controlled roll occurs, and the same display after a program-controlled roll occurs. In the program, the starting line number is specified as 8, the ending line number as 18, and the number of lines to be rolled down as 6.

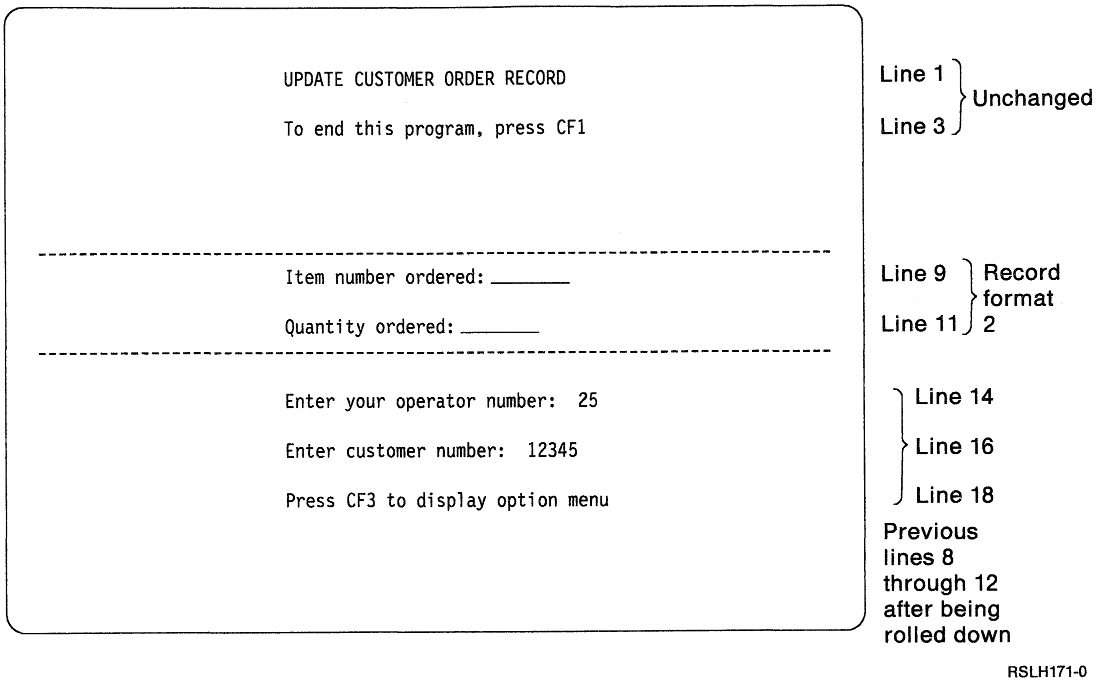
Display before the Roll Operation



```
UPDATE CUSTOMER ORDER RECORD

To end this program, press CF1




Enter your operator number:_______

Enter customer number:_______

Press CF3 to display option menu
```

RSLH165-0

Display after a Roll Down Operation



```
UPDATE CUSTOMER ORDER RECORD

To end this program, press CF1




--------------------------------------------------------------------------
Item number ordered: _______

Quantity ordered: _______
--------------------------------------------------------------------------

Enter your operator number:  25

Enter customer number:  12345

Press CF3 to display option menu
```

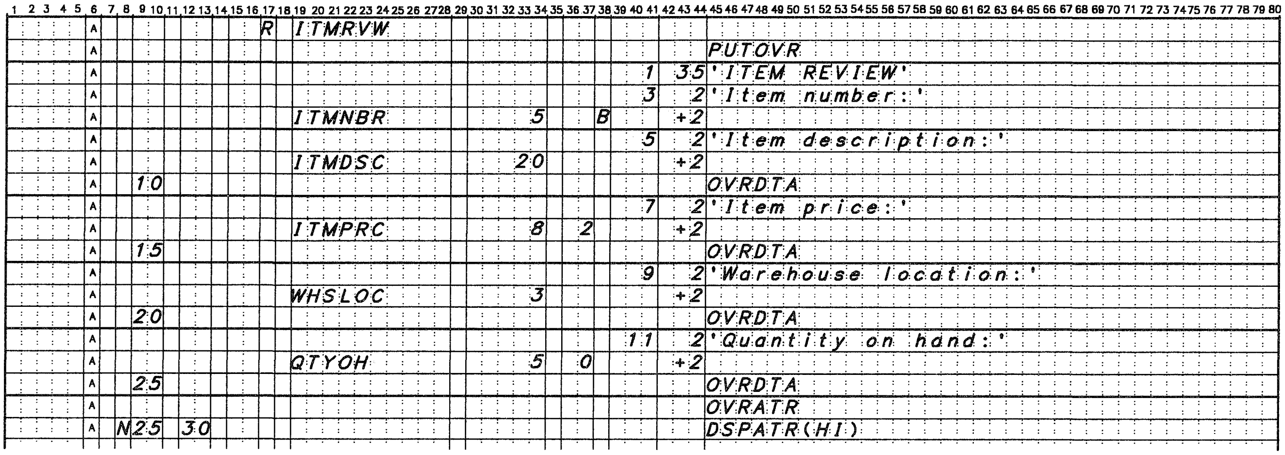
The ALWROL keyword cannot be used with the file level keyword PASSRCD or with the following record-level keywords: KEEP, ASSUME, USRDFN, SFL, or SFLCTL.

If the ERRMSG, ERRMSGID, PUTOVR, or PUTRETAIN keyword is in effect for the same output operation in which the ALWROL keyword is in effect, the system issues message CPF5014. If an ERRMSG, ERRMSGID, PUTOVR, or record level PUTRETAIN keyword is not in effect, the message is not issued. However, if the PUTRETAIN keyword is specified at the field level with option indicators, the message (CPF5014) is issued if the option indicators for the PUTRETAIN keyword are on or off.

Using the PUTOVR, OVRDTA, and OVRATR Keywords

You can use the PUTOVR, OVRDTA, and OVRATR keywords to cause the system to send only some of the data and attributes of a record to the display. This can be used to shorten the response time at the display, especially for remotely attached displays.

The following is an example of the PUTOVR keyword. The DDS describes a display that allows the work station user to enter an item number, and review the item description, the item price, the warehouse location, and the quantity on hand.



```
     A          R ITMRVW
     A                                      PUTOVR
     A                                  1 35'ITEM REVIEW'
     A                                  3  2'Item number:'
     A            ITMNBR         5   B    +2
     A                                  5  2'Item description:'
     A            ITMDSC        20        +2
     A  10                                  OVRDTA
     A                                  7  2'Item price:'
     A            ITMPRC         8  2     +2
     A  15                                  OVRDTA
     A                                  9  2'Warehouse location:'
     A            WHSLOC         3        +2
     A  20                                  OVRDTA
     A                                 11  2'Quantity on hand:'
     A            QTYOH          5  0     +2
     A  25                                  OVRDTA
     A                                      OVRATR
     A N25 30                               DSPATR(HI)
```

RSLH136-1

The following happens:

1. On the first output operation, all fields are sent to the display, and all option indicators are off. The PUTOVR keyword is ignored because the record is not already on the screen. On the first output operation, the current field values in the program are displayed for the output fields. If your program has not set any of these fields, the values will be whatever the high-level language used to initialize the output buffer.

   If an output-capable field must always have a specific value on the first output operation, you can use the DFT or DFTVAL keywords to initialize the field to that value. When used on an output-capable field with the PUTOVR and OVRDTA keywords the DFT keyword causes the system to place the default value on the display rather than the program value when the record is first placed on the display.

2. The work station user enters an item number. The program sets on indicators 10, 15, 20, and 25 and issues a write-read operation to display the output fields. On the write operation, the PUTOVR keyword is in effect because the record is already on the screen. Because the OVRDTA keyword is specified on the ITMDSC, ITMPRC, WHSLOC, and QTYOH fields and because their option indicators are on, these fields are the only data sent to the display.

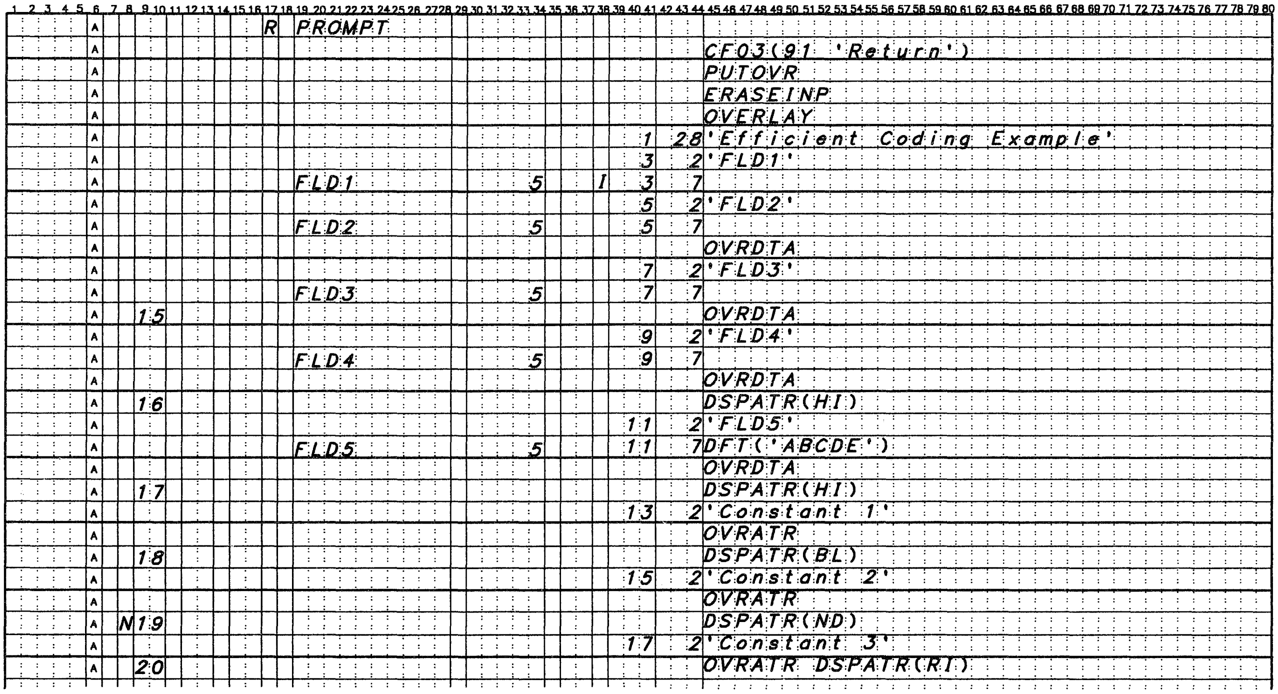
   If the work station user enters another item number and the data for a field already displayed does not change, the program sets off the option indicator and does not display that field again. For example, assume that for the second item number, the WHSLOC is the same as for the first item number. On the output operation to display the information for the second item number, the program sets off indicator 20. Therefore, the only fields sent to the display are ITMDSC, ITMPRC, and QTYOH because indicators 10, 15, and 25 are on.

For the QTYOH field, the program can change the attributes for the field without changing the data by setting off indicator 25 and setting on indicator 30 before the output operation.

You can use the option indicators on the OVRDTA keyword to control which fields are sent to the display. If no option indicators are used, all fields with the OVRDTA keyword specified are sent to the display on each output operation because the OVRDTA keyword is in effect when the PUTOVR keyword is in effect. In the preceding example, if no option indicators were used, all four fields would be sent to the display on each output operation. You can also use the same indicator to control more than one field.

An alternative design for this same application is to use two record formats and send the constants to the display in one record format and the variables in the other record format. You would have to use the CLRL(*NO) keyword to prevent the record format containing the constants from being erased. However, if the record format is already on the display, the use of the PUTOVR keyword provides the most efficient approach.

The following examples illustrate how to use the PUTOVR keyword for efficient coding:



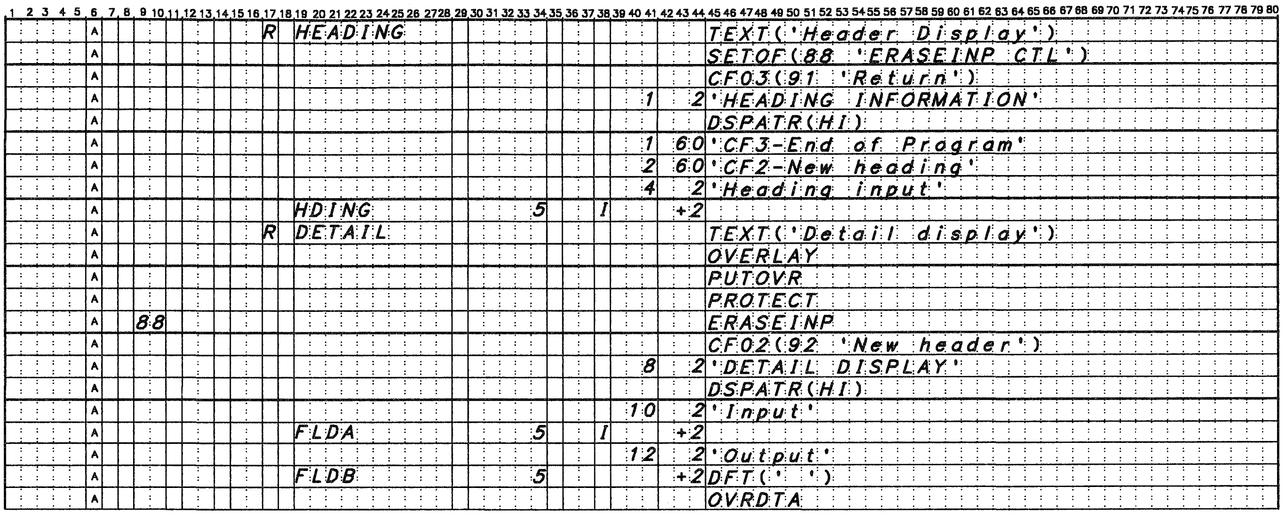
```
     A          R PROMPT
     A                                      CF03(91 'Return')
     A                                      PUTOVR
     A                                      ERASEINP
     A                                      OVERLAY
     A                                  1 28'Efficient Coding Example'
     A                                  3  2'FLD1'
     A            FLD1           5   I  3  7
     A                                  5  2'FLD2'
     A            FLD2           5      5  7
     A                                      OVRDTA
     A                                  7  2'FLD3'
     A            FLD3           5      7  7
     A 15                                   OVRDTA
     A                                  9  2'FLD4'
     A            FLD4           5      9  7
     A                                      OVRDTA
     A 16                                   DSPATR(HI)
     A                                 11  2'FLD5'
     A            FLD5           5     11  7DFT('ABCDE')
     A                                      OVRDTA
     A 17                                   DSPATR(HI)
     A                                 13  2'Constant 1'
     A                                      OVRATR
     A 18                                   DSPATR(BL)
     A                                 15  2'Constant 2'
     A                                      OVRATR
     AN19                                   DSPATR(ND)
     A                                 17  2'Constant 3'
     A 20                                   OVRATR DSPATR(RI)
```

RSLH137-1

In the preceding example, the following happens:

1. If the record format is not currently on the display, the PUTOVR, OVRATR, and OVRDTA keywords are ignored when the record format is displayed. On subsequent output operations when the record format is already on the display and the PUTOVR keyword is in effect, only the fields or constants defined with the OVRATR or OVRDTA keyword are sent to the display. The ERASEINP keyword is used because it is the most efficient way to clear all input fields, and the OVERLAY keyword is used because it is required with the ERASEINP keyword.
2. FLD1 is an input field that is cleared each time the record format is displayed.
3. FLD2 is sent to the display each time the record format is displayed because its associated OVRDTA keyword is unconditionally specified.
4. FLD3 is sent to the display on the first output operation. On subsequent output operations, FLD3 is not sent to the display unless indicator 15, which is used to condition the OVRDTA keyword, is on.
5. FLD4 is sent to the display on each output operation because its associated OVRDTA keyword is unconditionally specified. When the OVRDTA keyword is in effect, the attributes for the field are always sent to the display. Indicator 16 is used to control the DSPATR(HI) keyword for FLD4.
6. On the first output operation, the default value of ABCDE appears in FLD5. On subsequent output operations, a value from the program is displayed in FLD5 because its associated OVRDTA keyword is unconditionally specified. Indicator 17 is used to control the DSPATR(HI) keyword for FLD5.
7. *Constant 1* is always displayed, but it is only sent to the display on the first output operation. However, the attributes for the field are sent to the display each time the record format is written, and option indicator 18 is used to control whether the field blinks.
8. *Constant 2* is sent to the display only on the first output operation. However, the attributes for the field are sent to the display each time the record format is written, and if option indicator 19 is off, *Constant 2* will not be displayed.
9. *Constant 3* is sent to the display only on the first output operation. However, the attributes for this field are not sent to the display on subsequent output operations unless indicator 20 is on. If option indicator 20 is on when an output operation is done, *Constant 3* is displayed in reverse image, and it will continue to appear in reverse image regardless of the status of indicator 20 on subsequent output operations.

The following example shows how the PUTOVR keyword can be used for an application in which the work station user enters some information common to a group of records and then repeatedly enters detailed information relating to specific records in the group.



```
     A          R HEADING                   TEXT('Header Display')
     A                                      SETOF(88 'ERASEINP CTL')
     A                                      CF03(91 'Return')
     A                                  1  2'HEADING INFORMATION'
     A                                      DSPATR(HI)
     A                                  1 60'CF3-End of Program'
     A                                  2 60'CF2-New heading'
     A                                  4  2'Heading input'
     A            HDING          5   I     +2
     A          R DETAIL                    TEXT('Detail display')
     A                                      OVERLAY
     A                                      PUTOVR
     A                                      PROTECT
     A 88                                   ERASEINP
     A                                      CF02(92 'New header')
     A                                  8  2'DETAIL DISPLAY'
     A                                      DSPATR(HI)
     A                                 10  2'Input'
     A            FLDA           5   I     +2
     A                                 12  2'Output'
     A            FLDB           5         +2DFT(' ')
     A                                      OVRDTA
```

RSLH138-2

In the preceding example, the following happens:

1. The program displays the HEADING record format, and then performs an input operation to the record format to receive the HDING field as input. The SETOF keyword in the HEADING record format sets off indicator 88, which is used to condition the ERASEINP keyword in the DETAIL record format.

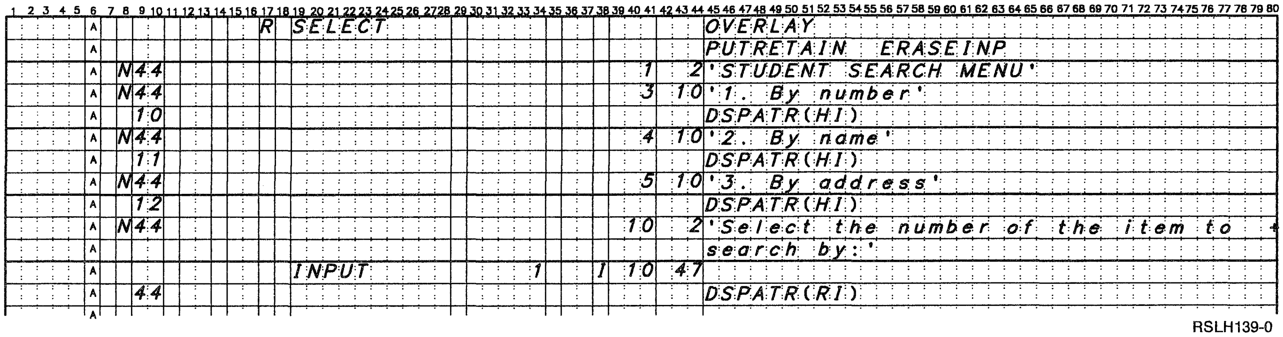
2. The program then displays the DETAIL record format. Because the OVERLAY keyword is in effect, the HEADING record format remains on the display. The PROTECT keyword is also in effect so the input field (HDING) in the HEADING record format is protected. Therefore, the work station user cannot change this field when the DETAIL record format is displayed.

3. The ERASEINP keyword is conditioned by option indicator 88. Because indicator 88 is off the first time the DETAIL record format is displayed, the ERASEINP keyword is not in effect. On subsequent output operations, indicator 88 is set on and the ERASEINP keyword is in effect. Therefore, FLDA is cleared on subsequent output operations. The option indicator is used on the ERASEINP keyword so that it is not in effect the first time the DETAIL record format is displayed. Because the ERASEINP keyword is processed before the PROTECT keyword, it would clear the HDING field in the HEADING record format if it were in effect the first time the DETAIL record format is written.

4. FLDB is an output field that is sent to the display on each output operation because the OVRDTA keyword is specified unconditionally. The DFT keyword with a value of blanks is used so the field will not contain any data the first time the DETAIL record is displayed for a group.

Using the PUTRETAIN Keyword

The PUTRETAIN keyword is used to reduce the number of characters sent to the display. The following is an example of the PUTRETAIN keyword used at the record format level. The following DDS describes a student search menu having three options 1, 2, and 3. The option selected is highlighted. For example, if option 1 is selected, the character string *1. By number* is highlighted.



```
     A          R SELECT                    OVERLAY
     A                                      PUTRETAIN  ERASEINP
     A N44                                1  2'STUDENT SEARCH MENU'
     A N44                                3 10'1. By number'
     A  10                                  DSPATR(HI)
     A N44                                4 10'2. By name'
     A  11                                  DSPATR(HI)
     A N44                                5 10'3. By address'
     A  12                                  DSPATR(HI)
     A N44                               10  2'Select the number of the item to +
     A                                      search by:'
     A            INPUT          1   I 10 47
     A  44                                  DSPATR(RI)
     A
```

RSLH139-0

| *Figure 4-10. Use of the PUTRETAIN Keyword at the Record Level*

The following happens:

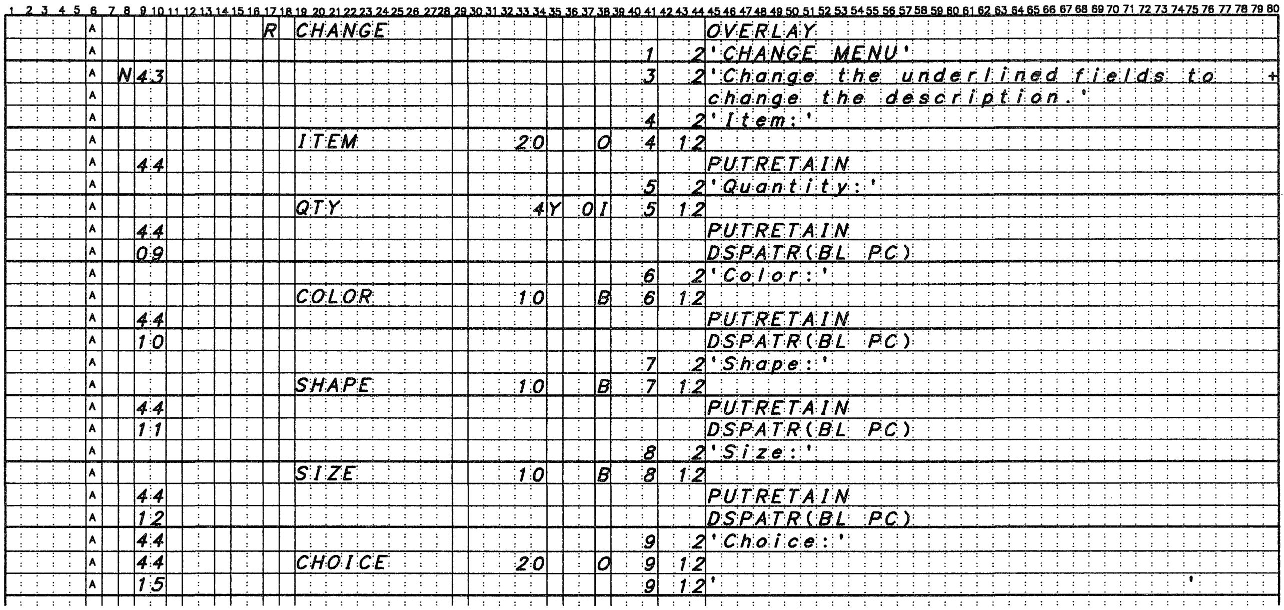
1. On the first output operation, all fields are sent to the display, and all option indicators are off. The PUTRETAIN keyword is ignored because the record is not already on the display.
2. The work station user selects item 1, 2, or 3. When the program receives the input, it sets on indicator 10, 11, or 12, depending on which item is chosen. If anything other than item 1, 2, or 3, the program sets on indicator 44.

   On the next output operation, field 1, 2, or 3 is highlighted, or the input field is in reverse image, depending on which indicator is on.

   The data for all fields is not resent to the display but the field attributes are resent. No data is sent for constants. To resend attributes for each output field or constant, 4 bytes are needed. To resend attributes for each input-capable field, 9 bytes are needed. By using the PUTRETAIN keyword, you reduce the number of characters sent to the display by 96, from 138 to 42. (These numbers do not include protocol control characters needed to frame data.)

   The ERASEINP keyword causes the work station user's selection to be erased.

The following is an example of the PUTRETAIN keyword used at the field level. Here, the PUTRETAIN keyword is used to retain invalid input and to reduce the number of characters sent to the display. The following DDS describes a display containing an item's name, color, shape, and size, and asks for quantity. The work station user can change the values for color, shape, and size.



```
     A          R CHANGE                    OVERLAY
     A                                  1  2'CHANGE MENU'
     A N43                              3  2'Change the underlined fields to   +
     A                                      change the description.'
     A                                  4  2'Item:'
     A            ITEM          20   O  4 12
     A  44                                  PUTRETAIN
     A                                  5  2'Quantity:'
     A            QTY            4Y 0I  5 12
     A  44                                  PUTRETAIN
     A  09                                  DSPATR(BL PC)
     A                                  6  2'Color:'
     A            COLOR         10   B  6 12
     A  44                                  PUTRETAIN
     A  10                                  DSPATR(BL PC)
     A                                  7  2'Shape:'
     A            SHAPE         10   B  7 12
     A  44                                  PUTRETAIN
     A  11                                  DSPATR(BL PC)
     A                                  8  2'Size:'
     A            SIZE          10   B  8 12
     A  44                                  PUTRETAIN
     A  12                                  DSPATR(BL PC)
     A  44                              9  2'Choice:'
     A  44        CHOICE        20   O  9 12
     A  15                              9 12'                              '
```

RSLH140-2

Figure 4-11. Use of the PUTRETAIN Keyword at the Field Level

The following happens:

1. On the first output operation, all indicators are off, so all the constants and the fields except CHOICE and the constant field following CHOICE are sent to the display.

2. The work station user enters a quantity. The program sets on indicator 43. When the next output operation occurs, indicator 43 prevents the second constant field from being resent.

3. When the work station user is to enter the quantity for another item, the program issues another output operation. The attributes for the fields QTY, ITEM, COLOR, SHAPE, and SIZE are sent to the display. Field selection prevents the CHOICE field from being sent to the display.

   At least one field, in this case QTY, must be retained to prevent the entire record area from being erased.

4. If the work station user enters an invalid quantity, color, shape, or size, indicator 44 is set on so that the input fields (QTY, COLOR, SHAPE,˙and SIZE) are not erased and so that the output field CHOICE is sent to the display. In addition, the appropriate indicator, 9, 10, 11, or 12, is set on so that the input field in error blinks and the cursor position is below the field. (The CHOICE field would show the user valid choices for the field in error.)

5. The CHOICE field and a constant field of blank are defined for the same location. After the work station user enters valid data, indicator 15 is set on, indicator 44 is set off, and the constant field initializes the CHOICE field to all blanks.

Using the DFT and DFTVAL Keywords

Both DFT and DFTVAL keywords are used to specify the default values to be displayed for fields. However, there are differences between the way the two are used.

The DFT keyword can be used with constant, input, output, and output/input fields and cannot be optioned. When it is used with output or output/input fields the OVRDTA and PUTOVR keywords must also be specified. If the record is not on the display, this combination of keywords will cause the default value to be placed on the screen. If the record is already on the display, the PUTOVR keyword is in effect and the data from the program appears on the display rather than the default value.

The DFTVAL keyword can be used only on output and output/input fields and can be optioned. If it is in effect on an output operation, the value from the keyword is placed in the field, rather than the value from the program. If the record is on the display and PUTOVR and OVRDTA keywords are in effect, the program value is used rather than the default value.

The DFT and DFTVAL keywords may not be specified on the same field.

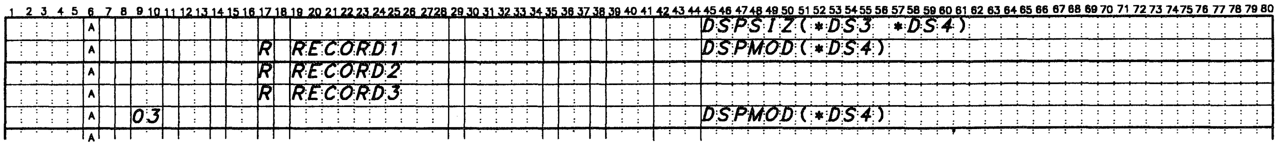
Using the DSPMOD Keyword

Some devices, for example the 3180-2 work station, support an alternate screen size. You can specify this alternate size using the DSPMOD keyword. The DSPMOD keyword indicates, for a particular record, which mode is used to display the record. Any record that does not have the DSPMOD keyword specified for it is displayed in the default display mode. The default display mode is the first of the *DS3 or *DS4 display sizes on the DSPSIZ keyword.

The DSPMOD keyword is only valid when both *DS3 and *DS4 are specified on the DSPSIZ keyword. This keyword is valid only at the record level. Option indicators are allowed. The DSPMOD keyword may not be duplicated in a record.

Note: The capability to display in 27 by 132 mode is allowed on 3180-2 and 3197 devices attached to a local work station controller, or remotely attached to a 5294 or 5394 controller. The DSPMOD keyword is ignored unless these controllers are used.

For example, the following DDS would display RECORD1 in 27 by 132 mode, and RECORD2 in 24 by 80 mode (the default mode set up by the DSPSIZ keyword). RECORD3 will be displayed in 27 by 132 mode if option indicator 03 is on, or in 24 by 80 mode if option indicator 03 is not on.



```
     A                                      DSPSIZ(*DS3  *DS4)
     A          R RECORD1                   DSPMOD(*DS4)
     A          R RECORD2
     A          R RECORD3
     A  03                                  DSPMOD(*DS4)
     A
```

RSLH134-1

Figure 4-12. Using the DSPMOD Keyword

The use of the DSPMOD keyword can cause the display mode to be changed dramatically. Caution should be used when specifying the DSPMOD keyword. When a record with DSPMOD active causes the mode to be changed, all records currently on the display are cleared and deleted from the active record table. The record with DSPMOD active is then sent to the display. The mode for this record is maintained on the display as long as the DSPMOD keyword is active. Setting DSPMOD off or a write operation to another record without DSPMOD causes the display mode to be placed back in the primary display screen size for the device.

Using the DDS in Figure 4-12 on page 4-106, the DSPMOD keyword gives the following results if records are written to the screen in the following order:

- RECORD1 is displayed in *DS4 mode.
- The display screen is cleared and RECORD2 is displayed in *DS3 mode.
- If indicator 03 is off, RECORD3 is displayed in *DS3 mode. RECORD2 remains on the display.
- If indicator 03 is on, RECORD2 is cleared and RECORD3 is displayed in *DS4 mode.

Note: When changing display modes, the displayed subfile data is removed from the display. However, the subfile data is not cleared from the subfile table.

The following keywords are ignored if the display modes have changed:

- ALWROL

  When a record is not on the screen, it cannot be rolled.
- ASSUME

  The records with the ASSUME keyword remain on the screen when the file is opened. When the display modes change, the records on the screen are cleared. This is similar to specifying the ASSUME keyword without the OVERLAY keyword. The display size of the file with the KEEP keyword must equal the display size of the file with the ASSUME keyword.
- CLRL

  All lines will be cleared by a change in display mode.
- ERASEINP/INZINP
- ERRMSG
- ERRMSGID
- KEEP
- OVERLAY
- PROTECT
- PUTOVR

  When the display modes change, the record is displayed with PUTOVR not in effect, even if the record was on the screen before the display modes changed.
- PUTRETAIN
- SFLMSG
- SFLMSGID

Using the CSRLOC and DSPATR(PC) Keywords

You can specify where you want the cursor positioned after an output operation by using the CSRLOC or DSPATR(PC) keyword.

On the record-level keyword CSRLOC, you can specify the names of two 3-character hidden fields that contain the exact line and position for the cursor location. With the CSRLOC keyword, you can position the cursor outside the record you are displaying.

With the field level keyword DSPATR(PC), you can position the cursor at the first position of that field.

Note: The cursor is not positioned if the keyboard is unlocked before the output operation.

If both the CSRLOC and DSPATR(PC) keywords are specified, the cursor is positioned by the CSRLOC keyword. If several fields have DSPATR(PC) keyword specified, the cursor is positioned at the first field for which the DSPATR(PC) keyword is specified.

If a cursor location is not specified on an output operation and there is no other record on the display with a cursor location specified, the cursor is positioned at the first input-capable field on the screen. If there was a record on the display that had the cursor position specified then the cursor remains in the specified position. If there is no input-capable field, the cursor is positioned in the upper-left corner of the screen.

Controlling Display Functions on Input Operations

On an input operation, you can control certain functions of the display (see "Read Operation" on page 4-25 for more information on input operations).

- Initializing a record on the display before reading it (INZRCD keyword). That is, the record does not have to be written to the display before it can be read by the program. The system does this the same way an application program performs an output operation with the exception of the following:

  - For an output-only field, no user data is available so the field is initialized to blanks. If the field is edited, the editing is ignored. If the BLKFOLD keyword is specified, it is ignored.
  - For an output/input field, no user data is available so the field is initialized to blanks. If the field is edited, the editing is ignored. The field actually contains null characters (hexadecimal zeros), which appear as blanks.
  - For a constant or input-only field, the data does not normally come from the output buffer so the field appears the same as when the program displays it using a write operation.
  - For a hidden field, the field is returned on a read operation as blanks (hex 40) if the field is a character field or zeros (hex F0) if the field is a numeric field.
  - For a message field, there is no message data so the field is ignored.
  - The LOGOUT keyword is ignored.
  - The ERRMSG and ERRMSGID keywords are ignored because the record is not already on the display.
  - The SFLMSG and SFLMSGID keywords are ignored.

  All other fields or keywords are processed as if they were selected on an output operation.

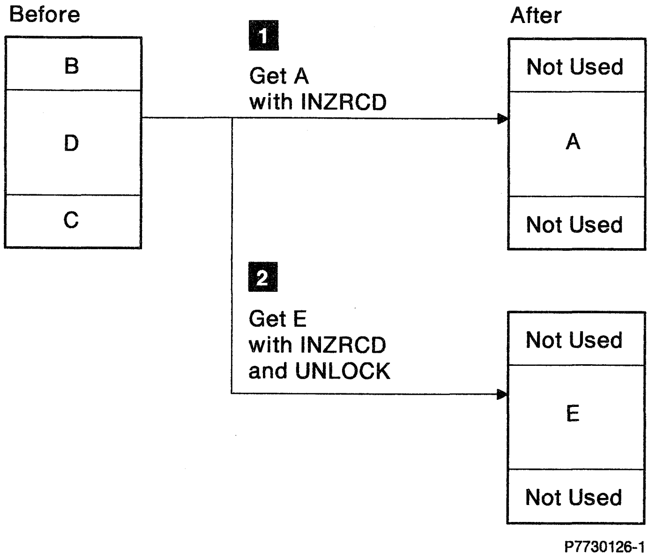
- Unlocking the keyboard so that data can be entered into input fields while the program is processing previously entered input data (UNLOCK keyword). Normally, input fields are not erased until after the keyboard is unlocked. On a read operation, input fields are erased after the keyboard is unlocked only if the UNLOCK keyword is specified and the GETRETAIN keyword is not specified.

- Retaining input data on a display after the work station user presses the Enter key (GETRETAIN keyword). The GETRETAIN keyword can only be used with the UNLOCK keyword.

- Setting on a response indicator when data is entered into an input field or when data is changed in an output/input field (CHANGE keyword).

Using the INZRCD and UNLOCK Keywords

The following diagram shows the effect of INZRCD and UNLOCK keywords on an input operation:



Note: Record formats A, D, and E occupy the same lines.

1 Record formats B, D, and C are erased if the OVERLAY keyword is not specified for record format A. Record format A is displayed with constants and initialized input fields. The keyboard is unlocked. The keyboard is locked after the work station user satisfies the get operation.

2 Record formats B, D, and C are erased if the OVERLAY keyword is not specified for record format E. Record format E is displayed with constants and initialized input fields. The keyboard is unlocked. After the work station user satisfies the read operation, the contents of the input fields are erased and the keyboard is unlocked again.

Note: Even though the UNLOCK keyword is specified, field validity checking, if specified, and command key verification are performed. Therefore, a work station user could be keying into the next record when an error message is sent to the display.

Messages on the Display

You can specify the following message handling functions for display files:

- Specify that a message is to be displayed on the message line with the ERRMSG or ERRMSGID keywords. For a message defined as a constant, use the ERRMSG keyword; second-level text is not supported for these kinds of messages. For a message defined in a message file, use the ERRMSGID keyword. Substitution text is not supported for the ERRMSGID keyword but second-level text can be seen by pressing the Help key. When you use these keywords, the keyboard locks and the work station user must press the Reset key to clear the message from the display and continue.

  Note: When you use the ERRMSG or ERRMSGID keyword, you should specify RSTDSP(*YES) on the CRTDSPF or CHGDSPF command; otherwise, data may be lost if the display file is suspended.

- Specify that messages are to be displayed on the message line when a subfile control record is written using the SFLMSG or SFLMSGID keywords. If the message is a constant, use the SFLMSG keyword; if it is defined in a message file, use the SFLMSGID keyword. The restrictions on these keywords are the same as ERRMSG and ERRMSGID

- Specify a message field (M in position 38). The value from this output field will appear on the message line. The value in the field is specified by the application program in the output buffer. The length of this field should not exceed 78 positions if the message is to be displayed on a 24 by 80 screen, or 130 positions if the message is to be displayed on a 27 by 132 screen. Second-level text and substitution variables are not supported for a message field.

- Specify that messages are to be displayed in a field on the display using the MSGID keyword. When the MSGID keyword is used, the keyboard is not locked because the field on which it is specified is a normal output-capable field. Second-level text and substitution variables are not supported for the MSGID keyword.

- Specify that messages that are contained on a program message queue are to be displayed by using the SFLMSGRCD, SFLMSGKEY, and SFLPGMQ keywords. When you use subfile support for messages, you can display more than one message at a time and the keyboard does not lock while the message is being displayed. Since the messages specified here are from the program message queue, both second-level text and substitution text are supported. See "Message Subfile" on page 4-54 for more information. See also the *CL Programmer's Guide* for information on message queues and sending messages.

The message displayed on the message line is determined by the following order of priority (1 is the highest):

1. ERRMSG
2. ERRMSGID
3. SFLMSG
4. SFLMSGID
5. Position 38 in DDS is M

In addition, you can change the line on which messages are displayed by using the MSGLOC keyword. If not specified otherwise, the message line is the last line on the display. If you use the MSGLOC keyword, the new message line applies for messages that are displayed for validity check errors and invalid keys as well as for user-defined messages.

Notes:

1. If the MSGLOC keyword is not specified, and a device capable of displaying 27 lines is attached to a local work station controller or attached to a remote 5294 or 5394 Controller, the default values are:
   - Line 28 for the 27 by 132 screen size
   - Line 25 for the 24 by 80 screen size
2. If line 25 is specified for the 24 by 80 mode, either because the default was used or line 25 was specified in the MSGLOC keyword, the message actually appears on line 24 unless the display is capable of displaying the message on the 25th line.
3. The normal system display for message help gives the work station user access to extended help (by pressing the Help key again), and may allow the user to use F10 to display all messages in the job log. The message help display used for a work station other than the job requester device or for a device associated with a multi-device file does not provide these functions.

Using the PRINT Keyword

You can specify the PRINT keyword in the following ways in DDS:

- With no parameters
- With a printer file name and, optionally, a library name
- With a response indicator
- With the special value *PGM

Table 4-4 summarizes the action taken for the forms of the PRINT keyword when the work station user presses the Print key.

Table 4-4. PRINT Keyword Considerations Using Print Key

Parameter	Action
None	The output goes to the work station printer associated with this display (the PRINTER parameter on the Create Device Description for Display [CRTDEVDSP] or Change Device Description for Display, [CHGDEVDSP]). If the operation to the work station printer fails or if there is no work station printer specified, the output goes to the printer file specified on the PRTFILE parameter on the display device description. The default for the PRTFILE parameter is QSYSPRT.
File name and, optionally, library name	The print operation is directed to the specified printer file. If the operation fails, it is directed to the default printer file, which is specified on the PRTFILE parameter on the CRTDEVDSP or CHGDEVDSP command.
Response indicator	Control returns to the program with the response indicator set on.
*PGM	Control returns to the program which must check the attention identifier in the input/output feedback area to determine which key was pressed.

The PRINT keyword can be used at the file level and also at the record level. When PRINT is specified at the record level, several records with different forms of the PRINT keyword (or with no PRINT keyword) can be active at the same time. The last record format written to the screen controls the use of the Print key for the entire screen. If the last record format written to the screen has the PRINT keyword specified, then that specification of the keyword is used when the Print key is pressed. If the last record format does not have the PRINT keyword specified, the Print key is not active for the entire screen.

If you specify the PRINT keyword in any form, the work station user can print a display containing the second-level text for a message. In this case, the print operation is performed as if the PRINT keyword were specified with no parameters.

When you change the device description of a work station printer (by using the CHGDEVPRT command or the DLTDEVD and CRTDEVPRT commands), you should also change the device description of the associated display device, using the PRINTER parameter on the CHGDEVDSP command. You should do this even if the name of the printer, whose device description you changed, remains the same.

Keyboard-Locking Considerations

The following lists what happens to the keyboard when a write, write-read, or read operation is run:

Operation	Keyboard
Write	The keyboard is unlocked by default. If the LOCK keyword is specified, the keyboard is not unlocked.
Write-Read	The keyboard is unlocked.
Read	The keyboard is unlocked (if locked) before work station user action. After work station user action, the keyboard is locked by default. If the UNLOCK keyword is specified, the keyboard is left unlocked.

Every time the keyboard is unlocked, the cursor is repositioned. In some cases, many write operations between read operations can cause erratic cursor movement. If the work station user starts keying before the last write operation, the cursor is repositioned when the keyboard is unlocked and this can cause confusion for the work station user. You can prevent this by using the LOCK keyword. By using the LOCK keyword on each write operation but the last, the keyboard remains locked until the last write operation. This avoids erratic cursor movement, but prevents the work station user from starting to key data.

Normally, a work station user action, such as pressing a valid command key, locks the keyboard.

To position the cursor with the DSPATR(PC), CSRLOC, or SFLRCDNBR(CURSOR) keyword, the keyboard must be locked. Only the following conditions on an output operation cause the keyboard to be locked and must be present for the device to position the cursor. (An output operation normally unlocks the keyboard before it ends unless the LOCK keyword is specified so these conditions lock the keyboard only momentarily.)

- Input-capable fields are erased (ERASEINP keyword).
- Modified data tags are reset (MDTOFF keyword).
- Any input-capable field is written to the display.
- The complete display is erased (a write operation without an OVERLAY keyword).
- The 5250 format buffer is reset, which can be the result of:
  - A record format with an input-capable field is overlaid or erased.
  - A record format with a cursor location specification is overlaid or erased.
  - The PROTECT keyword is specified on the record being written.

For the cursor positioning keyword to take effect, the keyboard must go from the lock condition to an unlocked condition. That is, if the keyboard is unlocked prior to the write operation, the cursor positioning keyword does not take effect immediately on the write operation. However, there is one exception. If the keyboard is temporarily locked during an output operation, the cursor positioning keyword will be in effect if the output operation unlocked the keyboard at the end.

In addition, if any of these preceding conditions happen on a write operation, the keyboard must be unlocked before any work station user action either by the same operation or a following operation (it should be the last write operation).

A write operation to a subfile never unlocks the keyboard because no input or output is sent to the device.

Using Multiple Devices

A multiple-device display file is a display file which can be used to access more than one display device in an application program. Multiple-device display files can be used to request input from one or more of the devices without waiting for the input to arrive. This is called the invite-device operation, and the devices from which input is requested are called invited devices. This allows the program to continue processing while the devices are sending data. The application program can then perform a read-from-invited-devices operation to wait for input from any one of the invited devices. The other invited devices can be sending input while the program processes the data from the responding device. After a device has responded and the data has been processed, it can be invited again by another invite-device operation.

Multiple-device display files are supported in RPG/400, COBOL/400, C/400, and CL. The support provided by CL is described in the *CL Programmer's Guide.* The information on multiple-device display files in this chapter is intended to provide a general outline of the support provided by the other high-level languages. For more detailed information on how to handle a display file as a multiple-device display file and how to call the operations described in this chapter, see the appropriate high-level language manual.

Before a device can be used for I/O operations in a multiple-device display file, it must be acquired to the file. At least one device is implicitly acquired when the file is opened and additional devices can be acquired by the acquire-device operation. For more information on the open and acquire-device operations, see their descriptions under "Display File Operations" on page 4-20. A program can direct the invite-device operation to any device currently acquired for the file. This operation requests input from the device without waiting for the input to arrive.

When the program is ready to process input from one of the invited devices, it can issue a read-from-invited-devices operation. This operation waits for a specified time for input to arrive from one of the invited devices. The time limit can be specified when the display file is created and can subsequently be changed or overridden. If no invited devices respond within the time limit, the program receives an indication that the time limit expired and can continue processing. If an invited device responds within the time limit, the program can determine which device responded and the record format used to process the data. The other invited devices remain invited and can be sending data. The responding device can also be invited again by another invite-device operation.

A read operation can also be directed to a specific display device. This operation will not complete until the specified device responds with data. The device need not be invited for the read operation, but, if it is, the program will wait for input and the device is no longer invited.

The program can cancel the request for input from an invited device by performing a cancel-invite operation on the invited device.

If the program no longer needs a device, it can perform a release-device operation to make the device available to other users.

When the program no longer needs the file or any of the file resources, it can make them available to other users by performing a close operation. All devices that are acquired to the file when it is closed are implicitly released.

Using Application Help

Application help is a function provided by the system that lets you define online help information for the records in a display file and then presents that help to the work station user when the Help key is pressed. You can define the display file so that you supply online help information for an entire record on the display, or you can define smaller rectangular areas within a record to have their own online help information. You can use this rectangular area support to get help down to the individual field level in a record. You can also define file-level help for the areas on the display that are not part of these help areas.

If you specify that you want application help used on a record, you define the areas you want it on with H specifications in your DDS and the HLPARA keyword. Then, when the Help key is pressed with the cursor on that area, the display file is suspended and the online help information is displayed. You can exit the help display by pressing the End function key or the Enter key. The display file that was suspended is displayed again.

You can define this online help information either as a document using the word processing functions in AS/400 Office or as record formats in this or another display file.

Note: Caution is advised when using application help with multiple devices acquired to the file, as all devices wait for the completion of the application help display.

To provide help to your application users, you define rectangular help areas on the display. Each help area is defined by an H specification level. In this H specification level, a help area is defined by specifying the upper-left row column and lower-right row column of the rectangular area. These coordinates must be located in the screen area, but are not required to be in the record area. The cursor location is determined when the Help key is pressed. If the Help key is enabled and the cursor is in any active help area, the help record or help document associated with that help area is displayed. If a help record is displayed, this becomes the *primary help record*.

The HELP keyword, which enables the Help key on the keyboard, must be specified at the file or record level if H specifications are used.

You can use the HLPRTN keyword to specify that your program is to receive control when the Help key is pressed instead of displaying help.

Help List

The system maintains a list of all H specifications from record formats that are on the display. When a record format is added to the display, the H specifications for that record format are placed on the front of this list. Thus, the H specifications from the records most recently written to the display are the first to be searched when the user presses the Help key. If the record format contains more than one H specification, they are added to the help list in the order in which they are defined in the display file. Thus, the order of the H specifications in the file also determines which help area is used. The help list is cleared either when a record format is written that clears the display, or when one is written that has the HLPCLR keyword enabled.

The HLPBDY keyword partitions this list of H specifications into sublists by defining help boundaries. A help sublist contains all those H specifications defined between help boundaries. (The H specification that has the HLPBDY keyword coded is considered to be before the boundary.) Sublists are important when using the roll keys to look at more online help information. See "Using Secondary Help Records" on page 4-121 for more information.

Pressing the Help key in a help area on the display does the following:

- Suspends the display file.
- Searches the list of active help areas to find the first one in the list that contains the cursor location when the Help key is pressed.
- As specified for that help area, either opens the display file that contains the help record and displays the desired help record, or displays a help document.
- Allows the work station user to roll forward or backward through the online help information.

Updating the Help List

As stated in the previous section, display data management updates the help list whenever a program writes a record to the display. How the help list is updated depends on whether the record:

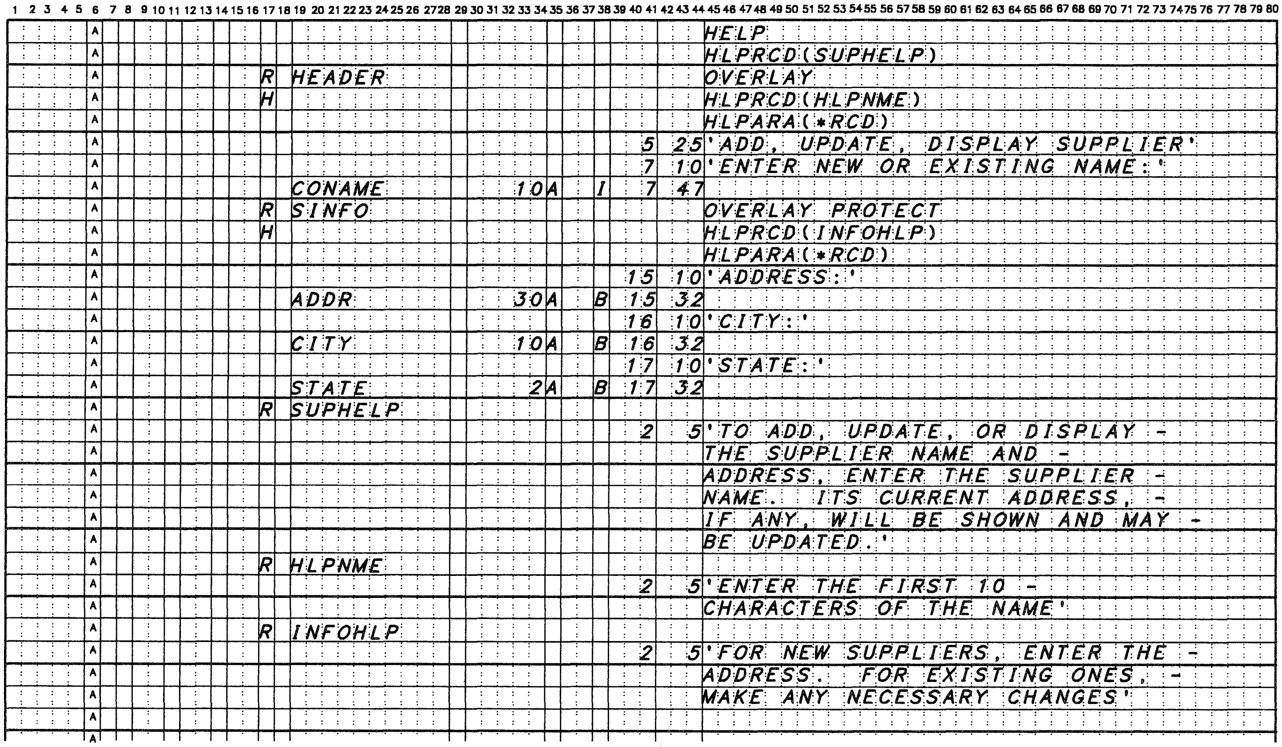
- Has the OVERLAY keyword in effect,
- Has H specifications,
- Completely or partially overlaps another record already on the display.

The following table documents the types of help-list updating for records for each combination of the three items listed previously:

Overlay	H Specifications	Overlaps	Help List Update
No	No	N/A	Help list is cleared.
No	Yes	N/A	Help list is cleared and H specifications for the record are added to the help list.
Yes	No	No	Help list is not changed.
Yes	No	Yes	H specifications are removed from the help list if they are within the boundaries of the record being written.
Yes	Yes	No	H specifications for the record are added to the help list. H specifications are removed from the help list if they are within the boundaries of any H specification in the record being added.
Yes	Yes	Yes	H specifications are removed from the help list if they are within the boundaries of the record being written. Then the H specifications for the record are added to the list. Finally, H specifications are removed from the help list if they are within the boundaries of any H specification in the record being added.

Defining One Help Area per Record

Only one file level help keyword (either HLPDOC or HLPRCD) can be defined, but multiple H specifications can be specified for a record. In the following example, one H specification is defined. The file level help is used to provide help when the cursor is not located in the record area for either HEADER or SINFO since HLPARA(*RCD) is the location specified on the H specification.



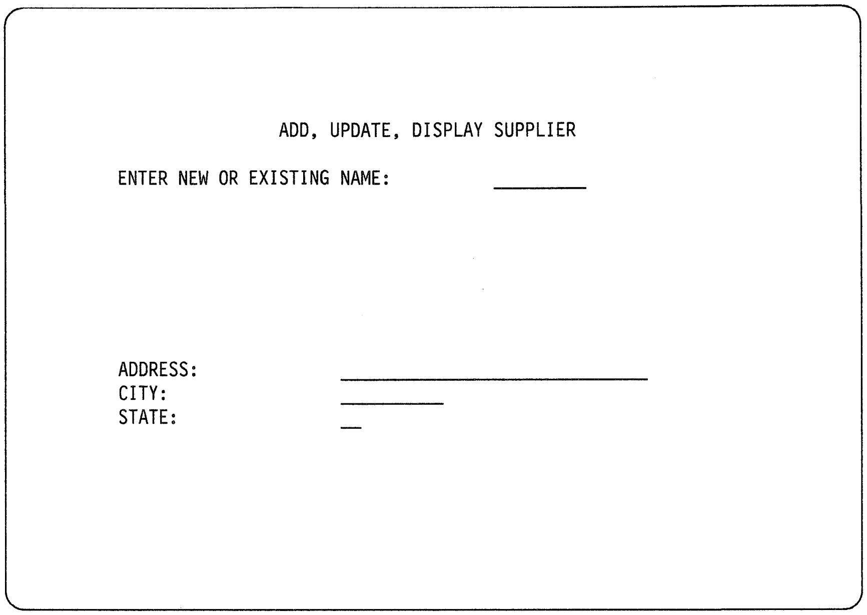
```
     A                                      HELP
     A                                      HLPRCD(SUPHELP)
     A          R HEADER                    OVERLAY
     A          H                           HLPRCD(HLPNME)
     A                                      HLPARA(*RCD)
     A                                  5 25'ADD, UPDATE, DISPLAY SUPPLIER'
     A                                  7 10'ENTER NEW OR EXISTING NAME:'
     A            CONAME        10A  I  7 47
     A          R SINFO                     OVERLAY PROTECT
     A          H                           HLPRCD(INFOHLP)
     A                                      HLPARA(*RCD)
     A                                 15 10'ADDRESS:'
     A            ADDR          30A  B 15 32
     A                                 16 10'CITY:'
     A            CITY          10A  B 16 32
     A                                 17 10'STATE:'
     A            STATE          2A  B 17 32
     A          R SUPHELP
     A                                  2  5'TO ADD, UPDATE, OR DISPLAY -
     A                                      THE SUPPLIER NAME AND -
     A                                      ADDRESS, ENTER THE SUPPLIER -
     A                                      NAME.  ITS CURRENT ADDRESS, -
     A                                      IF ANY, WILL BE SHOWN AND MAY -
     A                                      BE UPDATED.'
     A          R HLPNME
     A                                  2  5'ENTER THE FIRST 10 -
     A                                      CHARACTERS OF THE NAME'
     A          R INFOHLP
     A                                  2  5'FOR NEW SUPPLIERS, ENTER THE -
     A                                      ADDRESS.  FOR EXISTING ONES, -
     A                                      MAKE ANY NECESSARY CHANGES'
```

RSLH100-2

The H in column 17 indicates a help specification level. This is used to define help for the containing record. The help specifications are defined before the first field in the record. Multiple help specifications can be specified on a record (see the next example).

The HLPARA(*RCD) keyword indicates that the help is associated with the record area. The record area includes all columns in the lines occupied by the record. When the cursor is located in the record area and the Help key is pressed, the online help information from the record on the HLPRCD keyword is displayed. Refer to the *DDS Reference* for more information on these keywords.

The cursor location determines which of the three help screens are displayed. On the following screen example, both the HEADER and SINFO records are displayed.



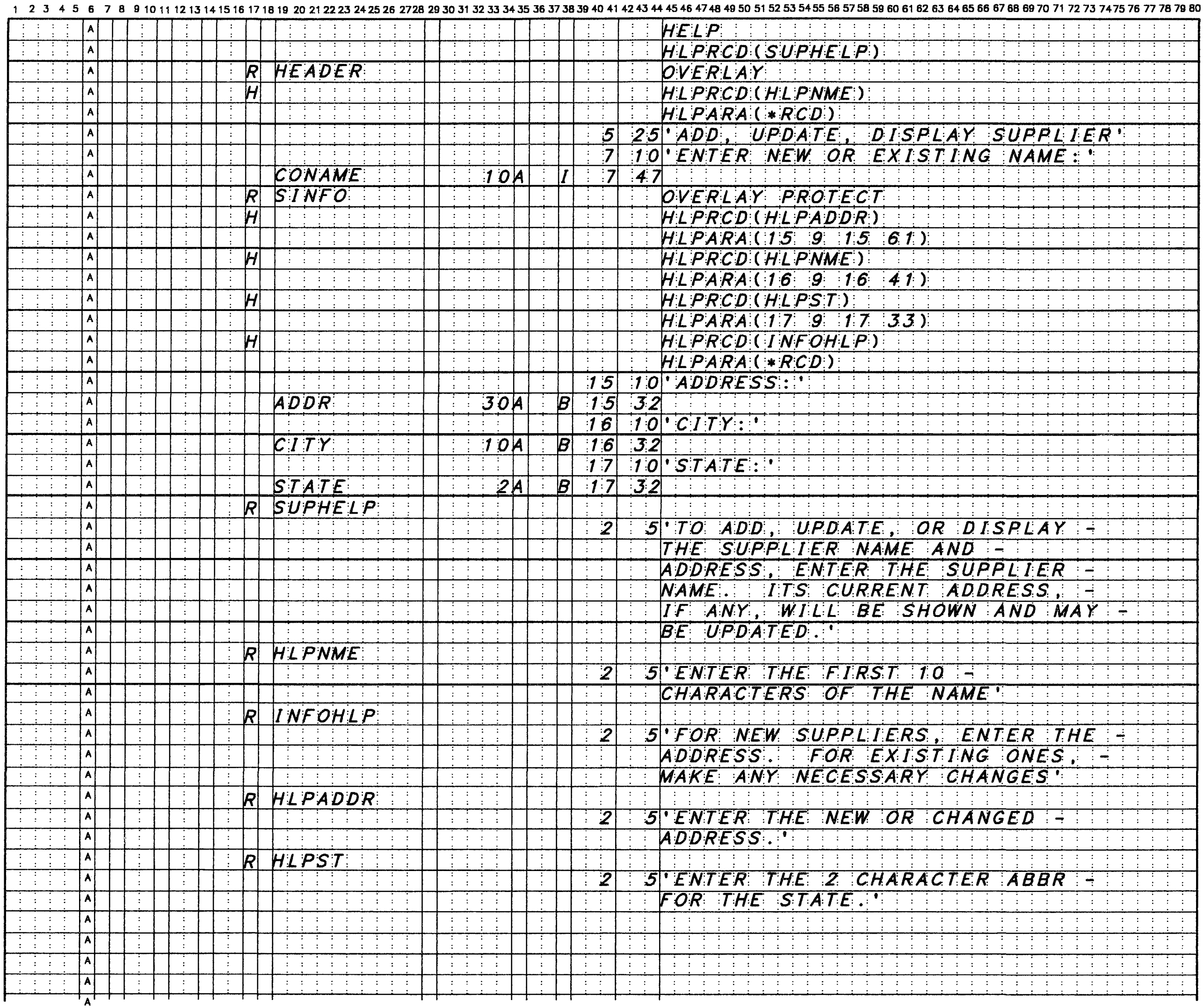
```
                    ADD, UPDATE, DISPLAY SUPPLIER

     ENTER NEW OR EXISTING NAME:          ________




     ADDRESS:               ____________________________
     CITY:                  _________
     STATE:                 __
```

For this screen, the record area for HEADER is line 5 to line 7, so HLPNME is displayed when the Help key is pressed and the cursor is located in this area. The record area for SINFO is line 15 to line 17, and the INFOHLP record is displayed. When the Help key is pressed and the cursor is not located in either of these areas, the SUPHELP record is displayed.

Defining Multiple Help Areas in a Record

Several help screens can be associated with a record, and selection is done by the position of the cursor. In this example, each field for the SINFO record has a separate help screen. Note that the help screen for the city name is the same as for the HEADER record.

The DDS for this file is:



```
     A                                      HELP
     A                                      HLPRCD(SUPHELP)
     A          R HEADER                    OVERLAY
     A          H                           HLPRCD(HLPNME)
     A                                      HLPARA(*RCD)
     A                                    5 25'ADD, UPDATE, DISPLAY SUPPLIER'
     A                                    7 10'ENTER NEW OR EXISTING NAME:'
     A            CONAME        10A  I    7 47
     A          R SINFO                     OVERLAY PROTECT
     A          H                           HLPRCD(HLPADDR)
     A                                      HLPARA(15 9 15 61)
     A          H                           HLPRCD(HLPNME)
     A                                      HLPARA(16 9 16 41)
     A          H                           HLPRCD(HLPST)
     A                                      HLPARA(17 9 17 33)
     A          H                           HLPRCD(INFOHLP)
     A                                      HLPARA(*RCD)
     A                                   15 10'ADDRESS:'
     A            ADDR          30A  B   15 32
     A                                   16 10'CITY:'
     A            CITY          10A  B   16 32
     A                                   17 10'STATE:'
     A            STATE          2A  B   17 32
     A          R SUPHELP
     A                                    2  5'TO ADD, UPDATE, OR DISPLAY -
     A                                      THE SUPPLIER NAME AND -
     A                                      ADDRESS, ENTER THE SUPPLIER -
     A                                      NAME.  ITS CURRENT ADDRESS, -
     A                                      IF ANY, WILL BE SHOWN AND MAY -
     A                                      BE UPDATED.'
     A          R HLPNME
     A                                    2  5'ENTER THE FIRST 10 -
     A                                      CHARACTERS OF THE NAME'
     A          R INFOHLP
     A                                    2  5'FOR NEW SUPPLIERS, ENTER THE -
     A                                      ADDRESS.  FOR EXISTING ONES, -
     A                                      MAKE ANY NECESSARY CHANGES'
     A          R HLPADDR
     A                                    2  5'ENTER THE NEW OR CHANGED -
     A                                      ADDRESS.'
     A          R HLPST
     A                                    2  5'ENTER THE 2 CHARACTER ABBR -
     A                                      FOR THE STATE.'
```

RSLH113-1

The order of H specifications is important since the first match found is selected. Therefore, the more specific HLPARA locations should be listed first. In this example, HLPARA(*RCD) includes the other HLPARAs, so it is listed last.

Using Secondary Help Records

When a Roll Up or Roll Down key is pressed on a primary help record, a *secondary help* record is displayed. The record formats that are displayed come from the same help group, or if none are available there, from the same help sublist as the help record which is currently displayed.

A help group is defined with the HLPSEQ keyword and consists of those records that have the same group name specified. The HLPSEQ keyword allows you to specify the help group name and the help sequencing number. The help sequencing number specifies the order in which the help will be displayed. If two help record formats that are in different display files happen to have the same group name, they are still considered to be in separate help groups. Help records that do not have a HLPSEQ keyword specified are considered to be groups of one.

Secondary help is selected in one of three ways (listed in order of priority):

Method 1

If the Roll Up key is pressed, a record format which is in the same help group as the current help record format and has the next highest help sequencing number is selected. If the current help record format already has the highest help sequencing number in the group, method 2 is used to select the secondary help.

If the Roll Down key is pressed the process is the same, except that the next-lowest help sequence number is used and method 2 is used when the lowest help sequence number is reached.

Method 2

If the Roll Up key is pressed, the help sublist is searched for the next H specification that does not refer to the same help group as the currently displayed help record. Pressing the Roll Down key works similarly, looking for the previous entry in the sublist that does not refer to the same help group.

Searching in the sublist continues until the boundary of the sublist is reached. The search then wraps to the other end of the sublist and continues until the current H specification being displayed is reached. If no H specification that has a satisfactory HLPRCD is found, the next method is used.

Notes:

1. To prevent rolling to an unexpected help record, it is recommended that the HLPBDY keyword be specified on the last H specification in each record of the application display file. This will define one sublist for each record that has help. However, if multiple records are on the screen, this may not be desirable. In this case, the HLPBDY keyword should be in effect only on the last record put to the screen.
2. Since this second method works only when a sublist can be identified, it is not used if the primary help was the default help format for the file (for example, if it was selected from the file level HLPRCD keyword).

Method 3

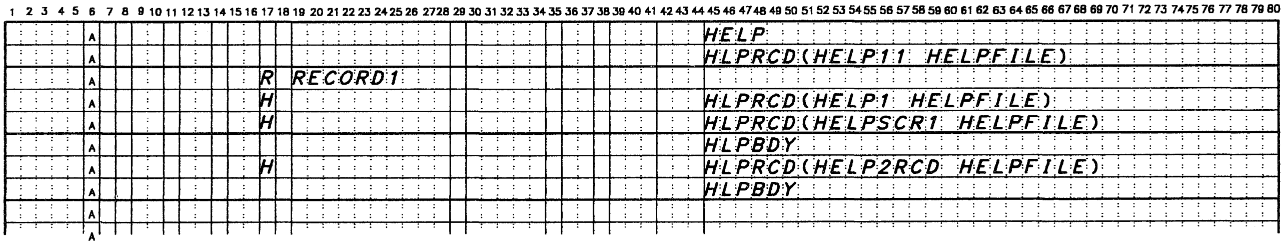
In the help display file, the record format which has the same help group name and the lowest (for Roll Down, highest) help sequence number is selected. This method always finds a match since the current help format will always meet this criteria if no other format does.

The following chart is a checklist for secondary help when Roll Up is pressed (listed in order of priority).

Characteristics (all must apply)	**Method Used**	**Resulting Secondary Help for Roll Up key**
In help file: – Current record help sequence number is not highest in help group	1	In help file: Next highest help sequencing number in help group
In application file: – Primary help was not file level help – At least one record on H-spec in the sublist is from different help group	2	In application file: HLPRCD from next H-spec in sublist (wrapping to top of sublist if necessary) whose record is from a different help group
Any	3	In help file: Record in help group with lowest help sequencing number

Example: Using Secondary Help Records

In the following example, the fields and HLPARA keywords are not specified since neither has an effect on the order of secondary text:



```
     A                                      HELP
     A                                      HLPRCD(HELP11 HELPFILE)
     A          R RECORD1
     A          H                           HLPRCD(HELP1 HELPFILE)
     A          H                           HLPRCD(HELPSCR1 HELPFILE)
     A                                      HLPBDY
     A          H                           HLPRCD(HELP2RCD HELPFILE)
     A                                      HLPBDY
     A
     A
```

RSLH108-0

File HELPFILE contains the following help record formats with HLPSEQ keywords coded as shown:

Record	Keyword Specification
HELP	HLPSEQ(GROUPA 1)
HELP1	HLPSEQ(GROUPA 2)
HELP11	HLPSEQ(GROUPA 3)
HELPSCR1	HLPSEQ(GROUPB 1)
HELPSCR2	HLPSEQ(GROUPB 2)
HELP2RCD	

Note that the help record formats HELP and HELPSCR2 are not referred to in the application display file. Since they are not referred to in this way, they will not be primary help records, but they will be displayed during secondary help processing as follows:

```
Sequence 1:
  Primary help format       HELP11
  Press Roll Down           HELP1         (using method 1)
  Press Roll Down           HELP          (using method 1)
  Press Roll Down           HELP11        (using method 3)
```

In this first sequence, HELP1 is shown when the Roll Down key is pressed since this is the previous help record format in GROUPA. Similarly, HELP is shown when Roll Down is pressed the second time. When Roll Down is pressed the third time, the end of the help group is reached, so method 2 comes into play. However, method 2 is not applicable since HELP11 was specified at the file level. Method 3 then selects HELP11 again since it is the last help record format in the original help group.

```
Sequence 2:
    Primary help format     HELPSCR1
    Press Roll Up           HELPSCR2      (using method 1)
    Press Roll Up           HELP1         (using method 2)
    Press Roll Down         HELP          (using method 1)
```

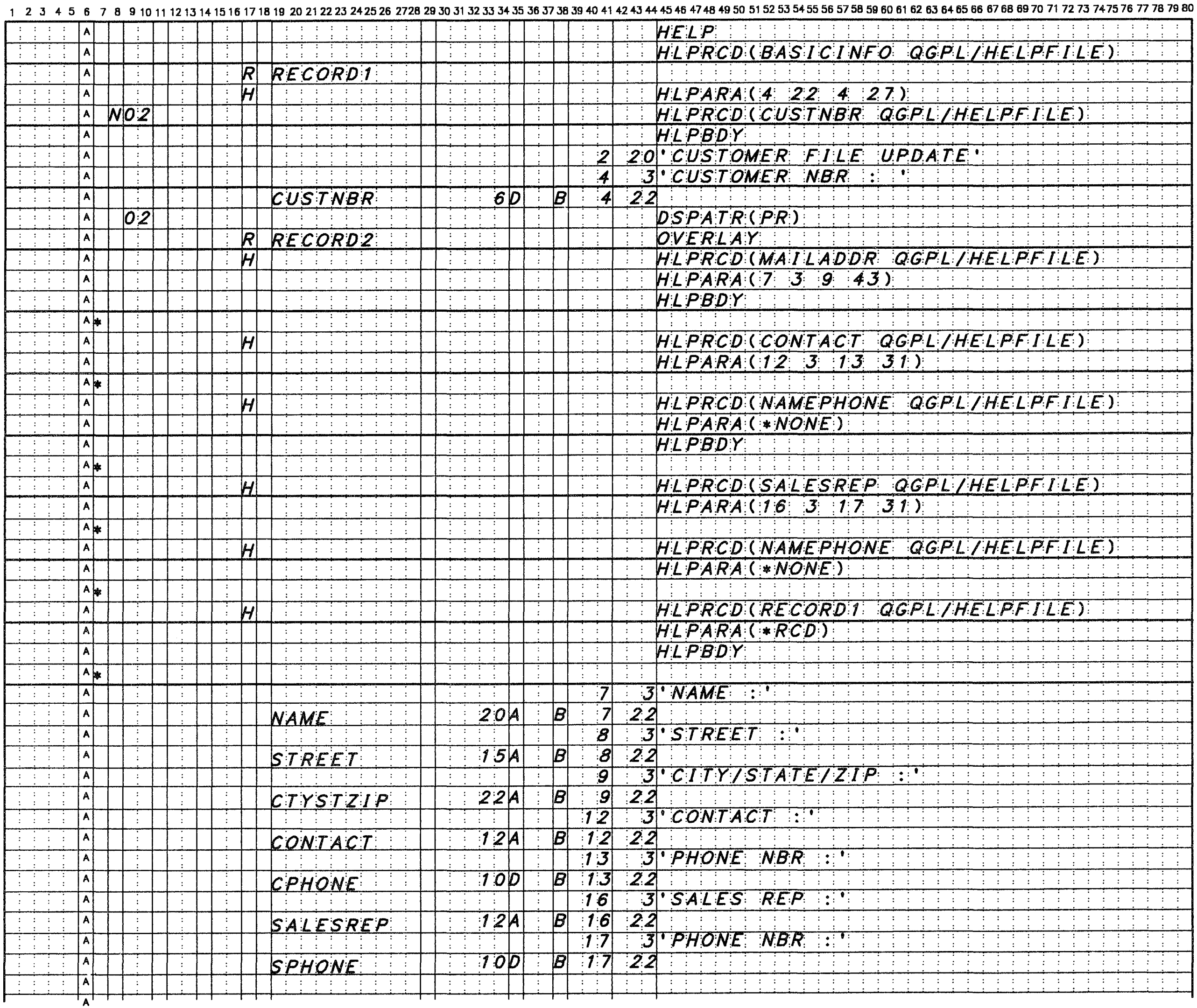
The second sequence starts with HELPSCR1 being displayed as the primary help format. Pressing Roll Up causes HELPSCR2 to be displayed since it is the next help format in GROUPB. Rolling up again runs off the end of the group and so method 2 is used. This finds HELP1 since it is the next format found, after wrapping, in the sublist that contains HELPSCR1. Pressing Roll Down now causes HELP to be displayed since it is the previous entry in GROUPA.

```
Sequence 3:
    Primary help format     HELP2RCD
    Press Roll Up or Down   HELP2RCD      (using method 3)
```

HELP2RCD is not in a help group and method 1 does not apply. Method 2 fails because HELP2RCD is the only one in its sublist. Thus, method 3 specifies that HELP2RCD is to be shown when rolling in either direction.

Examples of Application Help Using HLPRCD

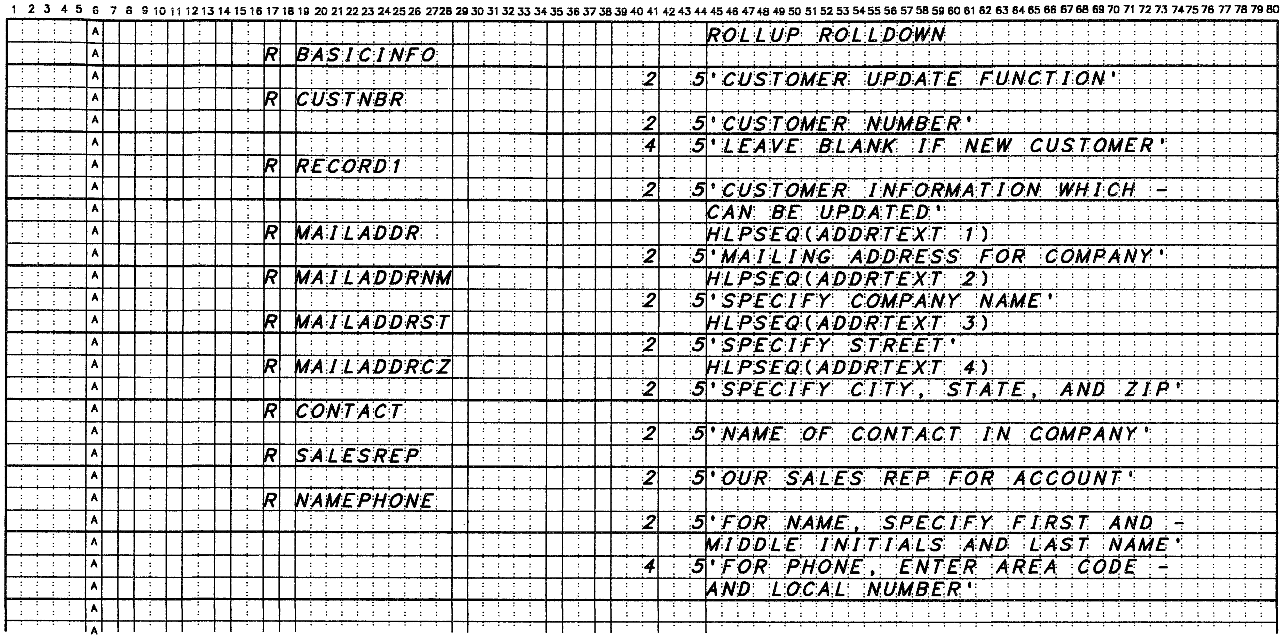
The following DDS describes an application display file which refers to another help display file QGPL/HELPFILE.



```
     A                                      HELP
     A                                      HLPRCD(BASICINFO QGPL/HELPFILE)
     A          R  RECORD1
     A          H                           HLPARA(4 22 4 27)
     A N02                                  HLPRCD(CUSTNBR QGPL/HELPFILE)
     A                                      HLPBDY
     A                                  2 20'CUSTOMER FILE UPDATE'
     A                                  4  3'CUSTOMER NBR :'
     A            CUSTNBR        6D  B  4 22
     A  02                                  DSPATR(PR)
     A          R  RECORD2                  OVERLAY
     A          H                           HLPRCD(MAILADDR QGPL/HELPFILE)
     A                                      HLPARA(7 3 9 43)
     A                                      HLPBDY
     A*
     A          H                           HLPRCD(CONTACT QGPL/HELPFILE)
     A                                      HLPARA(12 3 13 31)
     A*
     A          H                           HLPRCD(NAMEPHONE QGPL/HELPFILE)
     A                                      HLPARA(*NONE)
     A                                      HLPBDY
     A*
     A          H                           HLPRCD(SALESREP QGPL/HELPFILE)
     A                                      HLPARA(16 3 17 31)
     A*
     A          H                           HLPRCD(NAMEPHONE QGPL/HELPFILE)
     A                                      HLPARA(*NONE)
     A*
     A          H                           HLPRCD(RECORD1 QGPL/HELPFILE)
     A                                      HLPARA(*RCD)
     A                                      HLPBDY
     A*
     A                                  7  3'NAME :'
     A            NAME          20A  B  7 22
     A                                  8  3'STREET :'
     A            STREET        15A  B  8 22
     A                                  9  3'CITY/STATE/ZIP :'
     A            CTYSTZIP      22A  B  9 22
     A                                 12  3'CONTACT :'
     A            CONTACT       12A  B 12 22
     A                                 13  3'PHONE NBR :'
     A            CPHONE        10D  B 13 22
     A                                 16  3'SALES REP :'
     A            SALESREP      12A  B 16 22
     A                                 17  3'PHONE NBR :'
     A            SPHONE        10D  B 17 22
```

RSLH110-1

File QGPL/HELPFILE contains the following record formats:



```
     A                                            ROLLUP ROLLDOWN
     A          R BASICINFO
     A                                      2  5'CUSTOMER UPDATE FUNCTION'
     A          R CUSTNBR
     A                                      2  5'CUSTOMER NUMBER'
     A                                      4  5'LEAVE BLANK IF NEW CUSTOMER'
     A          R RECORD1
     A                                      2  5'CUSTOMER INFORMATION WHICH -
     A                                            CAN BE UPDATED'
     A          R MAILADDR                        HLPSEQ(ADDRTEXT 1)
     A                                      2  5'MAILING ADDRESS FOR COMPANY'
     A          R MAILADDRNM                      HLPSEQ(ADDRTEXT 2)
     A                                      2  5'SPECIFY COMPANY NAME'
     A          R MAILADDRST                      HLPSEQ(ADDRTEXT 3)
     A                                      2  5'SPECIFY STREET'
     A          R MAILADDRCZ                      HLPSEQ(ADDRTEXT 4)
     A                                      2  5'SPECIFY CITY, STATE, AND ZIP'
     A          R CONTACT
     A                                      2  5'NAME OF CONTACT IN COMPANY'
     A          R SALESREP
     A                                      2  5'OUR SALES REP FOR ACCOUNT'
     A          R NAMEPHONE
     A                                      2  5'FOR NAME, SPECIFY FIRST AND -
     A                                            MIDDLE INITIALS AND LAST NAME'
     A                                      4  5'FOR PHONE, ENTER AREA CODE -
     A                                            AND LOCAL NUMBER'
     A
     A
```

RSLH111-1

QGPL/HELPFILE can be divided into the following help groups:

Group Name	Elements
None	BASICINFO
None	CUSTNBR
None	CONTACT
None	NAMEPHONE
None	SALESREP
ADDRTEXT	MAILADDR, MAILADDRNM, MAILADDRST, MAILADDRCZ
None	RECORD1

Note: Stand-alone help records have no group name or sequencing. They are considered to be groups of one.

When RECORD1 is written to the screen, indicator 02 is off allowing input to the customer number field and enabling its help to be requested.

Cursor Location (Row/Column)	Primary Help	Roll Up Secondary Help	Roll Down Secondary Help
1/1 to 4/21	BASICINFO	BASICINFO	BASICINFO
4/22 to 4/27	CUSTNBR	CUSTNBR	CUSTNBR
4/28 to 24/80	BASICINFO	BASICINFO	BASICINFO

After the work station user presses the Enter key, the application program retrieves (if a customer number was entered) the current information for that customer and sets on indicator 02. This protects the customer number field and disables its help area. RECORD1 and RECORD2 are sent to the screen. Because both records are active, the help areas of both records are available for use, as shown in the following tables:

Cursor Location (Row/Column)	**Primary Help Record Format**	**Where Specified**
1/1 to 4/21	BASICINFO	File level HLPRCD keyword
4/22 to 4/27		
(ind 02 off)	CUSTNBR	First H-spec of RECORD1
(ind 02 on)	BASICINFO	File level HLPRCD keyword
4/28 to 7/2	BASICINFO	File level HLPRCD keyword
7/3 to 9/43	MAILADDR	First H-spec of RECORD2
9/44 to 12/2	RECORD1	Sixth H-spec of RECORD2
12/3 to 13/31	CONTACT	Second H-spec of RECORD2
13/32 to 16/2	RECORD1	Sixth H-spec of RECORD2
16/3 to 17/31	SALESREP	Fourth H-spec of RECORD2
17/32 to 24/80	BASICINFO	File level HLPRCD keyword

The secondary help is shown as follows:

Current Help Record Format	**Key Pressed**	**Secondary Help Record Format**	**Method Used**
BASICINFO	Either	BASICINFO	3
CUSTNBR	Either	CUSTNBR	3
MAILADDR	Roll Up	MAILADDRNM	1
	Roll Down	MAILADDRCZ	3
MAILADDRNM	Roll Up	MAILADDRST	1
	Roll Down	MAILADDR	1
MAILADDRST	Roll Up	MAILADDRCZ	1
	Roll Down	MAILADDRNM	1
MAILADDRCZ	Roll Up	MAILADDR	3
	Roll Down	MAILADDRST	1
RECORD1	Roll Up	SALESREP	2
	Roll Down	NAMEPHONE	2
SALESREP	Roll Up	NAMEPHONE	2
	Roll Down	RECORD1	2
NAMEPHONE	Roll Up	RECORD1	2
	Roll Down	SALESREP	2

Restrictions on Using Help Records

The following restrictions apply when using the help record form of application help:

1. The application program does not control the displaying of the help records. When the Help key is pressed, the system controls the displaying of help. Since the system does not control the buffers or hidden message field areas of the application program, the following takes place:

   - Output fields in the help record formats are displayed as blanks.
   - The ALIAS keyword is allowed but ignored.
   - CSRLOC and MSGID are processed by the system, but the hidden and program-to-system fields are not passed from the application program.

2. Application help controls all function keys when a help record is displayed:

   - The Roll Up key is enabled and causes the next help record to be displayed according to the rules for displaying secondary help.
   - The Roll Down key is enabled and causes the previous help record to be displayed according to the rules for secondary help.
   - The Enter key is enabled and causes a return to the application program. Any CA or CF key enabled for the help record will also cause a return to the application program. All other function keys, including the Help key, are ignored.

3. No input is returned to the user program:

   - Input-capable fields on a help record are displayed with underlines; however, no input is returned to the application program. Any input keyed in an input field is lost when application help is finished.
   - Response indicators are not returned.

4. Help records can contain H specifications, but they are ignored.

5. Option indicators are assumed to be off.

6. Screen size conditioning can be used.

7. Records with the USRDFN, SFL, and SFLCTL keywords may not be used as help records. When a display file is created, a diagnostic is issued if the HLPSEQ keyword is found on a record with one of these keywords. When the application is running, error reset message CPD4050 is issued if a record with one of these keywords is used as help. The help record is not displayed.

   The KEEP and ASSUME keywords should be avoided on help records as they cause results that cannot be predicted.

8. Only the following keywords are active when specified for help records:

   COLOR
   DATE
   DFT
   DSPATR
   DSPSIZ
   MSGCON
   MSGLOC
   TEXT
   SLNO (constant value). If SLNO(*VAR) is specified, 1 is used for the starting line number.
   TIME

   All other display file keywords, though allowed, do not make sense for help records and may or may not be processed while displaying the help.

Document Support for Application Help

In addition to the HLPRCD-specified help, you can use online documents for online help information. These documents are created using the word processing functions in the AS/400 Office licensed program. The document and folder do not have to exist when you create the display file. You do not need the AS/400 Office on your system to run online help information, but you do need the licensed program to create an online help document. See the *Office Services Concepts and Programmer's Guide* for information on creating these documents.

The DDS H specification keyword, HLPDOC, defines a document to be used as online help information. The parameters for HLPDOC (all required) are help information label name, document name, and folder name. Because it is stored in the DIA document library, the online document name is not library-qualified. Option indicators are allowed on the HLPDOC keyword. Any valid document and folder name can be specified on the HLPDOC keyword.

The folder-name parameter identifies the folder containing the document specified. Because folders can reside inside other folders, and because any given folder or document name is only unique within its containing folder, you may be required to concatenate several folder names together to identify a document or folder.

The folder name you specify on the HLPDOC keyword can be a simple folder name (which follows the same syntax rules as the document name), or you can specify the folder name as a concatenated name. If the folder name is concatenated, each simple folder name is separated by a forward slash (/). The total length of the folder name cannot exceed 63 characters.

In DDS, a help information label name, document name, or folder name can be enclosed in apostrophes. The enclosing apostrophes are required when the name contains an opening or a closing parenthesis character. When the name is enclosed in apostrophes, specify two apostrophes for each apostrophe character within the name. If the folder name is concatenated, the enclosing apostrophes (if specified) must be around the entire concatenated name.

If you use SDA to create your display file, SDA will always enclose the help information label name, document name, and folder name in apostrophes (and double any apostrophes contained in the names). SDA was not used in the examples in this section, and the help information label name, document name, and folder name are not enclosed in apostrophes.

When specified at the file level, the HLPDOC keyword defines a default help document for the file. This is displayed when the Help key is pressed and no HLPARA is defined for the cursor location.

When HLPDOC is specified on the H specification level, the HLPARA keyword must also be specified. If option indicators are specified on the HLPDOC keyword, they determine whether that H specification is selected.

When an online document is used to display help, the document is displayed starting from a specified location in the document (help-text-label-name parameter). You can page through the online information using the roll keys and command keys.

You can specify more than one help document in a file. Each help document must be specified on a separate H specification. The order of the H specifications is important. The H specification used is the first H specification with HPLARA containing the current cursor location and option indicators on the HLPDOC keyword in effect. Because of this sequencing, the most specific help area locations should be specified first.

Both help records and help documents can be specified in the same display file, but there are some considerations to be aware of when both are used:

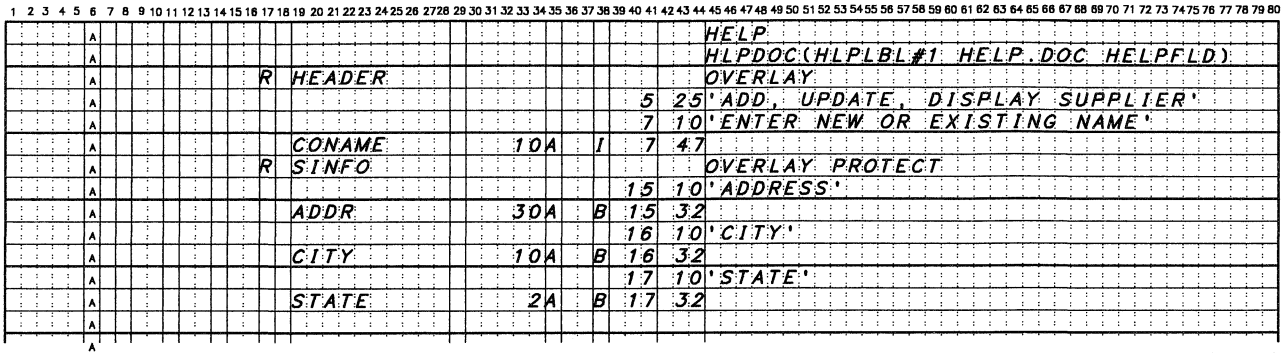
- When the Help key is pressed with the cursor location in a help area described by a help document, that document is the only help displayed when the roll keys are used. No other document or help records are displayed.
- When the Help key is pressed with the cursor location in a help area described by a help record, the normal sequencing used for help records is followed with help documents being ignored.
- The HLPDOC and HLPRCD keywords cannot both be specified at the file level or on the same H specification.
- The HLPBDY keyword cannot be specified with the HLPDOC keyword.

Defining File Level Help Document

Help documents can be specified either on the H specification or file level. When specified on the H specification, the help document is displayed when the cursor is located in the help area at the time the Help key was pressed. File level help is displayed when the Help key is pressed and the cursor is not located in a help area.

In this example, the help document displayed is in document HELP.DOC in folder HELPFLD, starting at help label HLPLBL#1. The HLPDOC keyword provides the name of the document containing the online help and the help label in the online help document that is to be displayed when the Help key is pressed. The roll keys and command keys can be used to page through the help document.

The DDS for this file is:



```
     A                                      HELP
     A                                      HLPDOC(HLPLBL#1 HELP.DOC HELPFLD)
     A          R HEADER                    OVERLAY
     A                                  5 25'ADD, UPDATE, DISPLAY SUPPLIER'
     A                                  7 10'ENTER NEW OR EXISTING NAME'
     A            CONAME        10A  I  7 47
     A          R SINFO                     OVERLAY PROTECT
     A                                 15 10'ADDRESS'
     A            ADDR          30A  B 15 32
     A                                 16 10'CITY'
     A            CITY          10A  B 16 32
     A                                 17 10'STATE'
     A            STATE          2A  B 17 32
     A
     A
```

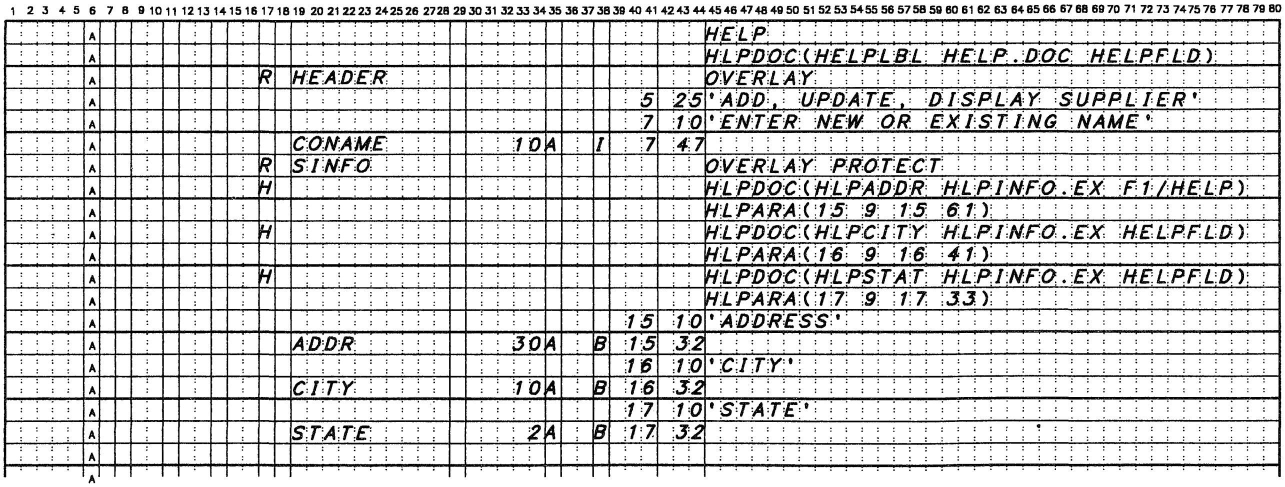
RSLH105-2

Defining H Specification Level Help Document

Only one file level help document can be defined, but multiple H specification level help documents can be specified. In the following example, the help document displayed for the ADDR field is in document HLPINFO.EX in folder F1/HELP, starting at help label HLPADDR. The help document displayed for the CITY field is in the HLPFLD document starting at help text label HLPCITY. The help document displayed for the STATE field is in the same document starting at help text label HLPSTAT.

If the cursor is not located in any of these areas when the Help key is pressed, the document HELP.DOC in folder HELPFLD, starting at help label HELPLBL, is displayed. The roll keys and command keys can be used to page through the help document.

The DDS for this file is:



```
     A                                      HELP
     A                                      HLPDOC(HELPLBL HELP.DOC HELPFLD)
     A          R HEADER                    OVERLAY
     A                                  5 25'ADD, UPDATE, DISPLAY SUPPLIER'
     A                                  7 10'ENTER NEW OR EXISTING NAME'
     A            CONAME        10A  I  7 47
     A          R SINFO                     OVERLAY PROTECT
     A          H                           HLPDOC(HLPADDR HLPINFO.EX F1/HELP)
     A                                      HLPARA(15 9 15 61)
     A          H                           HLPDOC(HLPCITY HLPINFO.EX HELPFLD)
     A                                      HLPARA(16 9 16 41)
     A          H                           HLPDOC(HLPSTAT HLPINFO.EX HELPFLD)
     A                                      HLPARA(17 9 17 33)
     A                                 15 10'ADDRESS'
     A            ADDR          30A  B 15 32
     A                                 16 10'CITY'
     A            CITY          10A  B 16 32
     A                                 17 10'STATE'
     A            STATE          2A  B 17 32
     A
     A
```

RSLH104-2

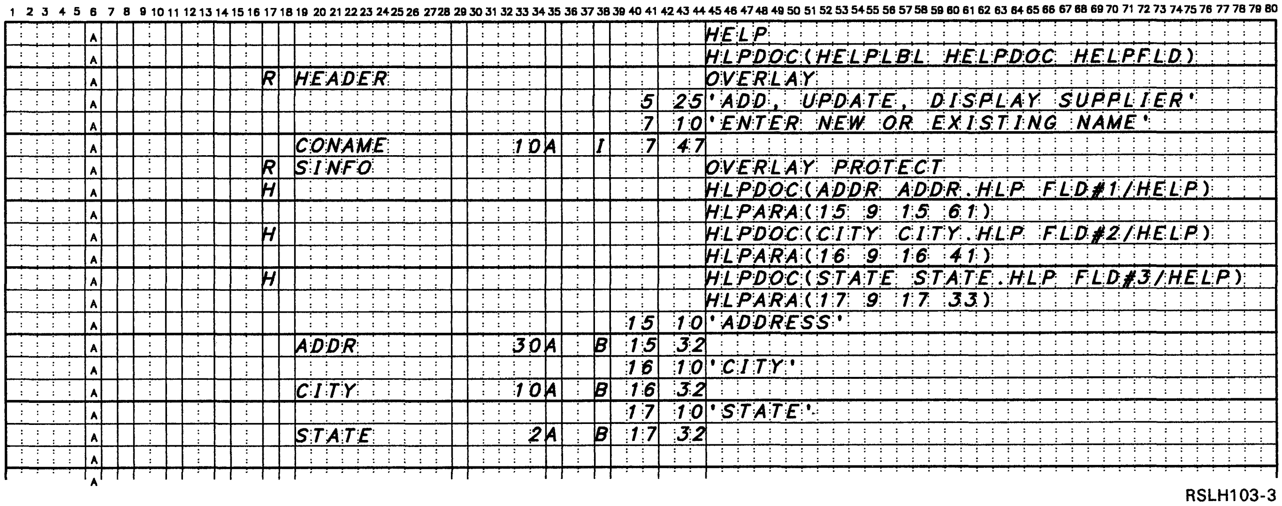
Defining Multiple H Specification Level Help Documents

In this example, the help document to be displayed for the ADDR field is in document ADDR.HLP in folder FLD#1/HELP, starting at help label ADDR. The help document displayed for the CITY field is in document CITY.HLP in folder FLD#2/HELP, starting at help label CITY. The help document to be displayed for the STATE field is in document STATE.HLP in folder FLD#3/HELP starting at help label STATE.

If the cursor is not located in any of these areas, the document HELPDOC in folder HELPFLD, starting at help label HELPLBL, is displayed.

When an online help document is displayed, the roll keys and command keys can be used to page through that document. You cannot page between documents.

The DDS for this file is:



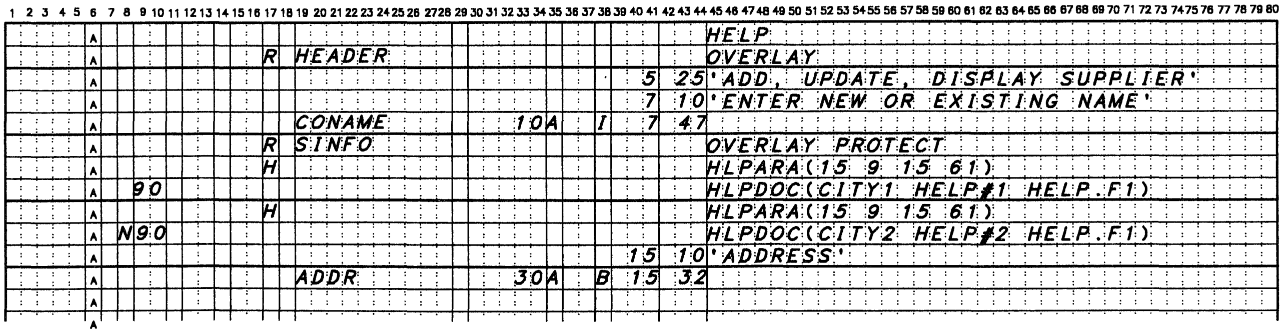
```
     A                                      HELP
     A                                      HLPDOC(HELPLBL HELPDOC HELPFLD)
     A          R HEADER                    OVERLAY
     A                                  5 25'ADD, UPDATE, DISPLAY SUPPLIER'
     A                                  7 10'ENTER NEW OR EXISTING NAME'
     A            CONAME        10A  I  7 47
     A          R SINFO                     OVERLAY PROTECT
     A          H                           HLPDOC(ADDR ADDR.HLP FLD#1/HELP)
     A                                      HLPARA(15 9 15 61)
     A          H                           HLPDOC(CITY CITY.HLP FLD#2/HELP)
     A                                      HLPARA(16 9 16 41)
     A          H                           HLPDOC(STATE STATE.HLP FLD#3/HELP)
     A                                      HLPARA(17 9 17 33)
     A                                 15 10'ADDRESS'
     A            ADDR          30A  B 15 32
     A                                 16 10'CITY'
     A            CITY          10A  B 16 32
     A                                 17 10'STATE'
     A            STATE          2A  B 17 32
     A
     A
```

RSLH103-3

Defining Help Documents with Option Indicators

The HLPDOC keyword can be specified with option indicators. In this example, assume the SINFO record is put with indicator 90 on and the cursor is in the help area defined for the H specification. When the Help key is pressed, the HELP#1 document in folder HELP.F1, starting at location CITY1, is displayed. If indicator 90 is off, the HELP#2 document in folder HELP.F1, starting at location CITY2, is displayed.

The DDS for this file is:



```
     A                                      HELP
     A          R HEADER                    OVERLAY
     A                                  5 25'ADD, UPDATE, DISPLAY SUPPLIER'
     A                                  7 10'ENTER NEW OR EXISTING NAME'
     A            CONAME        10A  I  7 47
     A          R SINFO                     OVERLAY PROTECT
     A          H                           HLPARA(15 9 15 61)
     A 90                                   HLPDOC(CITY1 HELP#1 HELP.F1)
     A          H                           HLPARA(15 9 15 61)
     AN90                                   HLPDOC(CITY2 HELP#2 HELP.F1)
     A                                 15 10'ADDRESS'
     A            ADDR          30A  B 15 32
     A
     A
```

RSLH102-1

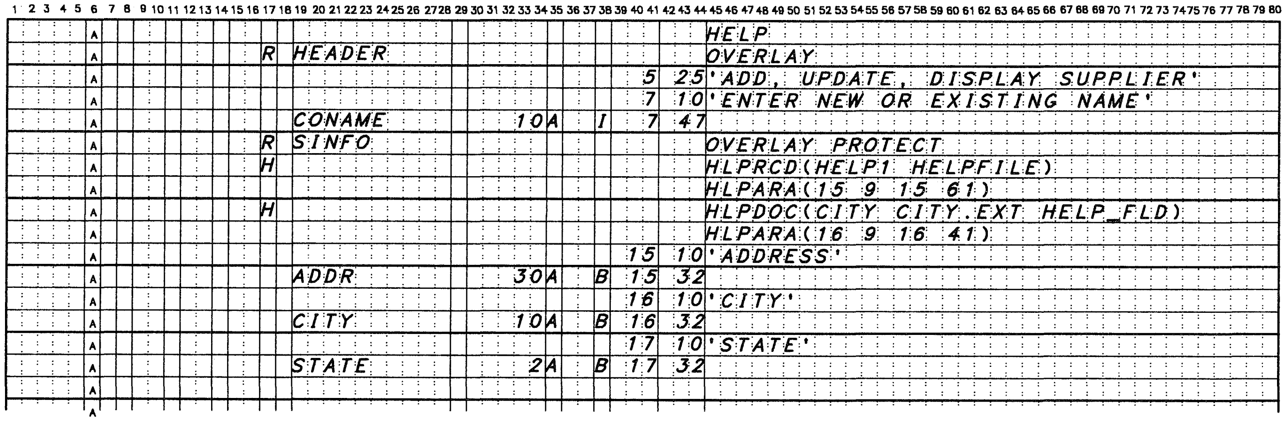
Defining Help Records and Help Documents in the Same File

Both help records and help documents can be specified in the same display file. If secondary help is defined for the help records, paging through the help records is the same as if no help documents were specified.

In the following example:

- If the cursor is in the area defined by the first H specification, the HELP1 help record from file HELPFILE is displayed. If the roll keys are pressed while looking at the help, the records in HELPFILE that are in the same help sequencing group are displayed.
- If the cursor is in the area defined by the second H specification, the help document CITY.EXT in the folder HELP_FLD, starting at the help label CITY, is displayed. If the roll keys or command keys are pressed, the remainder of the document is displayed.

The DDS for this file is :



```
     A                                      HELP
     A          R HEADER                    OVERLAY
     A                                  5 25'ADD, UPDATE, DISPLAY SUPPLIER'
     A                                  7 10'ENTER NEW OR EXISTING NAME'
     A            CONAME        10A  I  7 47
     A          R SINFO                     OVERLAY PROTECT
     A          H                           HLPRCD(HELP1 HELPFILE)
     A                                      HLPARA(15 9 15 61)
     A          H                           HLPDOC(CITY CITY.EXT HELP_FLD)
     A                                      HLPARA(16 9 16 41)
     A                                 15 10'ADDRESS'
     A            ADDR          30A  B 15 32
     A                                 16 10'CITY'
     A            CITY          10A  B 16 32
     A                                 17 10'STATE'
     A            STATE          2A  B 17 32
     A
     A
```

RSLH101-2

Automatic Recovery for Display Errors

Permanent I/O errors on display devices can be handled automatically when you use the device-recovery-action job attribute. The Device Recovery Action (DEVRCYACN) parameter of the Change Job (CHGJOB) command indicates a system action for jobs that are about to receive a permanent I/O error (such as a device being powered off). You can use the following parameter values:

*DSCMSG When the I/O error occurs, the system runs a Disconnect Job (DSCJOB) command to suspend the job. The user remains disconnected until signing on again to the same display. The job is resumed at the point just after the I/O operation. Display data management sets the return code to 83E1 and sends a CPF509F message to the program message queue. These indicate that the display has been cleared and is again available for use.

*DSCENDRQS This option also runs the DSCJOB command when an I/O error occurs, but when the job is resumed, the system runs the End Request (ENDRQS) command to give control to the most recent request-processing program.

*ENDJOB When an I/O error occurs, the job ends automatically and its priorities are lowered to conserve system resources.

*ENDJOBNOLIST Again, the job ends automatically and its priorities are lowered to conserve system resources. However, no job log is produced for the job.

When an I/O error occurs (especially in the case of a communications line failing for several active jobs at once), using the device-recovery-action job attribute can save system resources. Also, the device-recovery-action job attribute makes the job of coding for I/O errors easier. Once the device is recovered, the user application can continue at the point the error occurred by using the *DSCMSG and *DSCENDRQS values. For *DSCMSG, if the return code is 83E1 after an I/O operation, the program needs to assume that the display is blank. The program must branch back to the point at which the first I/O operation to the display is done. For *DSCENDRQS, the request-processing program receives control upon connecting to the job again. This is similar to selecting option 2 on the System Request Menu.

The device-recovery-action job attribute works only for *REQUESTER devices and does not apply to batch jobs, pass-through jobs, or work-station-function jobs.

Passing Data between Programs Using Display Functions

You can use display device support I/O operations to pass data between programs both in the same routing step of a job or across routing steps.

Passing Data in the Same Routing Step in a Job

To pass data between programs in the same routing step, you can either share the file between the programs, or you can use the KEEP and ASSUME DDS keywords to share a file between programs. The programs must open the same file and the file must be designated as a shared file. (See "Sharing Files in the Same Job" on page 1-21.) Because the programs share the file, the read operations are performed as described under "Read Operation" on page 4-25 and "Read Operation Following a Read Operation" on page 4-30.

You can also use the KEEP and ASSUME keywords to pass data between programs. The data is written to the display by the first program, and used from the display by the second.

You can use the KEEP keyword to keep data on the display for review after the program has ended, or you can use it to pass data between programs.

Normally, when a file is closed, the current display is cleared. However, you can control this by using the KEEP keyword. If a record that is on the display when the file is closed has the KEEP keyword specified, the system saves the name of the first such record to support passing data. Using this keyword alone, you cannot support processing of passed data, you must also use the ASSUME keyword in the display file opened by the next program.

The ASSUME keyword causes a read operation to a specific record format name to be valid when no preceding write operation to that record format name (or any other record format) has occurred since the device file was opened. The following shows a typical example of what happens when the ASSUME keyword is specified:

1. Program PGM1 issues a write operation to the record format PGM1ANY in the display file DSPFIL1 and calls program PGM2. PGM1ANY specifies the LOCK keyword.
2. PGM2 opens DSPFIL2 and issues a read operation to record format PGM2ANY. The ASSUME keyword is specified. Input data is read from the display and processed by record format PGM2ANY.

Records from record formats having the ASSUME keyword specified cannot overlay one another on the display. In addition, all records must have at least one field that is displayed.

When the system reads assumed records from the display, only fields whose modified data tags (MDTs) are on are returned. (This assumes that either the DSPATR(MDT) keyword was specified for the field on the last output operation or the work station user has keyed in the field.) When the ASSUME keyword is in effect, the data returned to the program is as follows:

- For input-only and output/input fields with their MDTS on, the modified data is returned.
- For other input-capable fields, blanks are returned for character fields and zeros are returned for numeric fields.

In addition, only those fields received whose line and position on the display match the field of an assumed record are processed. The data is processed using the field descriptions in the assumed record, independent of how the fields were written to the display.

For the 5250 display station, the first write operation or write-read operation after the file is opened sends a CA and CF key specification to the device. Because there is no write operation after an open when using the ASSUME keyword, the CF key specification remains as it was left by the last application. The CA and CF key specification in this file will not be used until after the first write operation to the file.

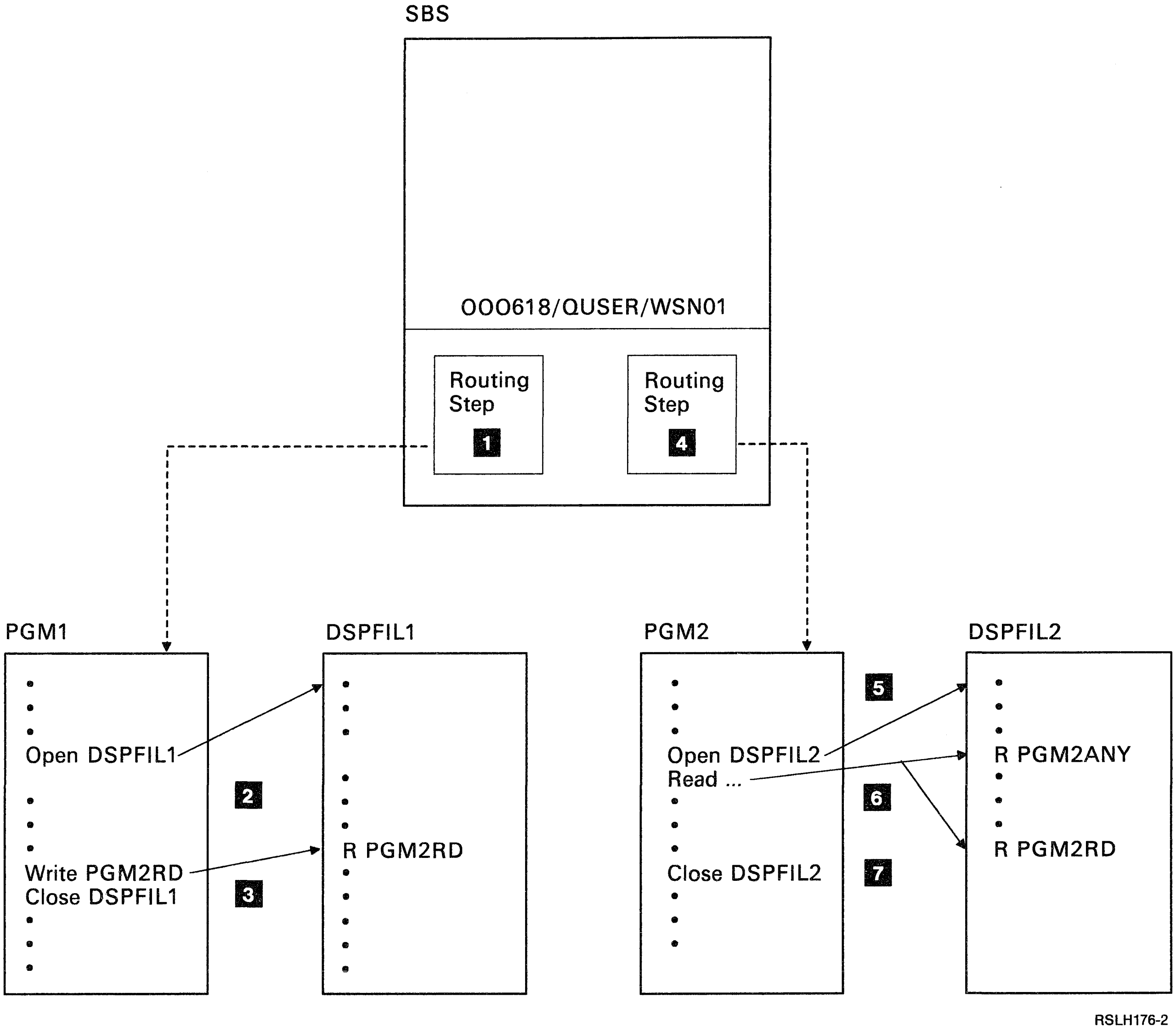
After the file is opened and the first write operation is issued, the display will be erased if the OVERLAY keyword is not specified. When the OVERLAY keyword is specified, all input-capable fields on the display become output-only fields. After the first write operation, assumed records cannot be read by a program.

Fields not received from the assumed records on the display are returned to the user program as follows:

Field	**Initialized To**
Field with DFT keyword used	Value specified in DFT keyword
Character field (no DFT keyword)	Blanks
Numeric field (no DFT keyword)	Zero
Hidden field	Zero

Passing Data between Routing Steps in a Job

The following example shows the steps used in passing display data between programs in different routing steps. Note the use of the KEEP, ASSUME, and PASSRCD keywords. Programs PGM1 and PGM2 are user programs started by the subsystem SBS using different routing entries. PGM1 and PGM2 run in different routing steps but are both contained in the job 000618/QUSER/WSN01.



1 The user signs on and SBS starts a routing step based on the routing data. The first program in the routing step is PGM1.

2 PGM1 opens the display file DSPFIL1.

3 PGM1 interacts with the work station user and issues the following before ending:

- A write operation to DSPFIL1 with the record format name PGM2RD. The KEEP keyword is specified in the record format PGM2RD.
- A close operation to DSPFIL1. The information displayed on WSN01 is not cleared because the KEEP keyword is specified in record format PGM2RD.

PGM1 then ends by issuing a RRTJOB command specifying routing data that will cause SBS to call PGM2.

4 SBS starts a new routing step based on the data supplied by PGM1 in the RRTJOB command.

5 PGM2 opens the display file DSPFIL2.

6 PGM2 performs a read operation to DSPFIL2 with or without the record format name PGM2ANY. If the record format name is not specified, the system tries to use the record format PGM2RD to process the data because the KEEP keyword was specified (in step 5). To do so, record format PGM2RD must exist in DSPFIL2, but it does not have to be identical to record format PGM2RD in DSPFIL1; only the fields that are required by PGM2 must be identical. (If a field is returned from the device for which no field description exists in PGM2ANY, the field is ignored. If the field description requires validation and the data received fails the validity check, PGM2 reissues a read operation to allow the current work station user to correct the data.) If a record format is not specified or if record format PGM2RD does not exist in DPSFIL2 or does not have the ASSUME keyword specified, the data passed to the device via the KEEP keyword cannot be processed. (The ASSUME keyword must be specified for the record format used to process passed data.)

7 PGM2 processes the data and issues a close operation to DSPFIL2, which clears the display.

When PGM2 ends, its routing step ends and the display returns to the control of SBS which shows the sign-on display.

User-Defined Data Streams

Instead of having device support control and process the 5250 display data stream, you can control and process it. To do so, you must use the USRDFN keyword in the DDS for a device file. The data is sent to the device using a normal output operation that uses the name of the record format containing the USRDFN keyword. When input data is received after the output operation, your program must determine through the input record area what was received from the device.

Note: You should be careful when using this support because error conditions can cause an apparent OS/400 malfunction.

When you use the USRDFN keyword at the record level, the format does not contain any fields. Therefore, the buffer length of the file defaults to the length of the longest normal record or 100 (whichever is greater). If the user-defined data stream is longer than this default buffer length, you should perform the following steps to obtain a larger buffer length:

- Define an externally described file, and create the program using this file and a record format in that file. The format should not have response indicators defined for it. This includes file level indicators that are propagated to all record formats. Any fields in the format should be defined with a field use of *both* (input and output).
- Create a second file and specify LVLCHK(*NO). The second file should have two record formats:
  - A format with the same name as the format in the first file and which contains the USRDFN keyword.
  - A format with one field in it. The length of this field must be as long as the longest user-defined data stream that is to be sent to the screen.
- When you run the program, override the first file with the second display file.

Device Differences

If you use user-defined data streams, you should be aware of the following differences between local work stations (those attached to the work station controller) and remote work stations (those attached to remote work station controllers).

- For local work stations, a start-of-header order length greater than seven causes the negative response hex 1005012B to be sent. No negative response is sent for remote work stations.
- For local work stations, do not accept more than four field control words for each input-field definition. More than four causes the negative response hex 10050130 to be sent. No negative response is sent for remote work stations.
- For local work stations, self-check fields can be as long as 33 bytes for signed numeric fields and 32 bytes for all other fields. If the length is exceeded, the status response hex 00000287 is sent to the work station user when he tries to exit the self-check field. For remote work stations, all fields can be 33 bytes.
- The ASCII data translation feature is supported only for remote work stations.
- Two forms of the Request Maintenance Statistics command are supported for remote work stations: (1) requests that the error log area be reset, and (2) requests that the error log area not be reset. For local work stations, the error log area is always reset.

User-Defined Data Stream Considerations

The output record area for a user-defined record format must contain the following in the order specified:

1. Device-specific information (required by device support), which causes device support to send the device-specific data stream to the device.
2. Device-specific data stream, which is sent to the device.

The USRDFN keyword is specified at the record format level and excludes for that record format most other functions such as the use of indicators and all field-related functions. However, the display file can contain other record formats not containing the USRDFN keyword, and the record formats can be used in any order by a program. When you use such a display file, you should be aware of the following:

- When the display device support write routine recognizes a user-defined request, it disregards all previous requests to the display. At the completion of the user-defined request, device support assumes that a single record format is on the screen and that this is the record format containing the USRDFN keyword. All erasing, all resetting, and the unlocking of the keyboard is your responsibility. The next I/O request can be another user-defined request or a normal field-level request.
- A normal field-level request after a user-defined request is handled as follows:
  - If the OVERLAY keyword is not specified, the screen is erased before the request is run.
  - If the OVERLAY keyword is specified, only the portion of the display needed (entire lines) is erased.
  - The 5250 format buffer is reset, which means that all input fields are changed to output-only fields.

  Device support assumes that only this record format exists on the screen. All previous requests are disregarded.
- Help specifications are allowed in user-defined record formats.
- When the USRDFN keyword is specified in a record format, no fields can be defined for that record format. The only valid keywords are:

File Level	Record Format Level
KEEP	ALTNAME
OPENPRT	HELP
PASSRCD	HLPCLR
PRINT	HLPRTN
	INVITE
	PRINT
	TEXT

- The user-defined data stream can alter the CF and CA keys and the location of the message line in the display. However, device support assumes they are the same as when they were last set.
- All display files that contain user-defined data streams should be opened as both input and output files. This is because read and write commands in the data stream are not dependent on write and read requests.

The output buffer must include information needed to send and receive the appropriate line controls. The buffer format is:

Device	Byte	Contents
5250 display station	0-1	Send data length (in hexadecimal), defines the output data stream length
	2-3	Receive data length (in hexadecimal), defines the maximum input data length
	4	Requested function Hex 61 Send (3270 data stream with no translate) Hex 63 Send/Receive (3270 data stream with no translate) Hex 71 Send Hex 73 Send/Receive
	5-n	Output data stream

The output data stream should start with an escape character hex 04 and be followed by a clear unit hex 40 or write to display command hex 11.

For details about the 5250 Display Data Stream see the *IBM 5250 Information Display System Functions Reference Manual.*

This support prohibits the use of a read operation except after a write-read-with-nowait operation. A write-read operation that sends a read command and specifies the receive data length performs the operation normally performed with a read request. The write-read function can be performed by doing one of the following:

Operation	RPG/400	COBOL/400	Control Language
Write-read wait	EXFMT		SNDRCVF WAIT(*YES)
Write-read with nowait	WRITE with INVITE, READ	WRITE with INVITE, READ	SNDRCVF WAIT(*NO)

For each I/O request, you must make sure that the function byte and the send length actually reflect the data stream sent. The receive length must be long enough to accommodate all data returned by the device. If any of these are wrong, unexpected or unacceptable functions may happen. For example, if the function byte indicates a send to the work station and the data stream specifies a read modified command, device support sends the data stream to the work station and no read is performed.

On input operations, input buffer of the program contains the data received from the device. For example, when a read modified completes, the input buffer contains an aid identification (AID) byte and the cursor address followed by each changed field. Each changed field is preceded by the buffer address order and field location on the screen.

All AID bytes are accepted by device support for user-defined data streams. If the Print key is pressed, device support attempts to perform the print function.

All write and write-read operations must specify the record format name. A read operation that specifies the name of a record format that contains the USRDFN keyword and is not one of the appropriate read operations for the device causes an exception to be issued.

Each request to a 5250 display station can contain more than one command. Each command must be requested using the appropriate device support operation identified as follows:

Operation	Output Buffer Send Length (Bytes 0 – 1)	Output Buffer Receive Length (Bytes 2 – 3)	Output Buffer Request Function (Byte 4)	Output Buffer Command (Bytes 5 – n)
Write	nn	0	71 (Send)	Clear unit Clear format table Write to display Write error line Restore Roll Copy
Write-Read	nn	nn	73 (Send/Receive)	Clear unit Clear format table Write to display Write error line Restore Roll Read input fields Read MDT fields Read immediate Save

Using Windows

There are applications that could make use of a window on the display to assist the user in entering data.

A window is information that overlays part of the current display and allows the user to read the information inside the window. The remainder of the display is not overlaid by the window and still can be read by the user.

Windows can be done using standard DDS or with user-defined data streams. An example, using two methods of standard DDS, is part of library QUSRTOOL, which is an optionally installed part of the OS/400 program.

This window example is part of the Programming and System Management Tips and Techniques section of QUSRTOOL. See the member AAAMAP in file QATTINFO of library QUSRTOOL for information on where the documentation and example source is located.

Using Display Files with Program-Described Data

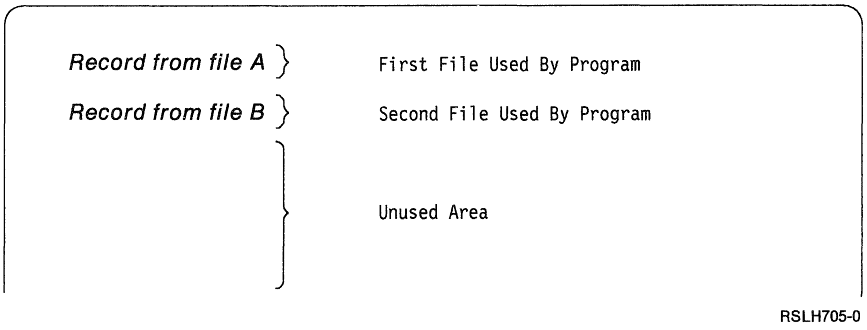
You can create a display file without using data description specifications. Such a display file then uses program-described data, and has no record or field level descriptions of its own.

When you are using program-described data with a display file to communicate with one or more display devices, only simple display formatting can be performed, and that formatting must be specified in the high-level language program that is using the file. All field descriptions are defined and all processing is performed in the program that uses the file. More than one display file can be opened to the same device at the same time within the same program, but only two can be used on the same device at the same time: one for input and one for output.

When a display file that uses program-described data is opened, the system treats the input or output area on the display as a single field. That is, the field length is the same as the record length. The record length is defined by the program that is using the file, and stays the same from the time the file is opened until it is closed. Indicators cannot be passed when records are passed from the program to a device, or from the device to a program. Also, command keys cannot be used for program-described device files.

The space on the display is assigned to program-described device files as shown in the following example.

Records for the first file used by the program appear on the first (top) part of the display. Records for the second file appear on the display immediately following the area used by records in the first file.



In program-described display files, the maximum record lengths are:

- For an input file, the screen size minus 2
- For an output file, the screen size minus 2
- For an output/input file, the screen size minus 2
- For two files (one output and one input), the screen size minus 3

When a program-described device file is opened, it can be defined as:

- Input only
- Output only
- Input and output

Input-Only Files

When an input-only file is opened, the record is initialized to a single blank field on the screen. The cursor is positioned at the first position of the field and the work station user can type in any type of data.

When the program reads the record, the input is passed to the program. The record is not erased from the screen. The cursor is again positioned at the first position of the record (field) and the keyboard is unlocked when the program reads the next record. The work station user can then type in the next record over the previous record.

Output-Only Files

When an output-only file is opened, the record is initialized to a single blank field on the screen. When the program writes a record to the file, the record is displayed and the keyboard is locked. The work station user must press the Enter key before another record can be written to the file. Subsequent records written to the file erase the currently displayed record because only one record can be displayed for the output file.

Input and Output Files

When an input and output file is opened, the record is initialized to a single blank field on the screen. The cursor is positioned at the first position of the file and the work station user can type in any kind of data.

The program that is using the file can read records or write records in any sequence. Whenever a record is written to the file, the modified data tag is set off (to indicate that data was not entered into the field) and the keyboard is unlocked. If the work station user then enters data into the field, the modified data tag is set on. If the next operation is a write operation instead of a read operation, the data keyed in by the work station user is written over and the modified data tag is set off again.

Chapter 5. Printer Support

The printers attached to the AS/400 system are supported by the operating system through printer files. Some of the values specified by printer files are the:

- Page size, in lines per page and columns per line
- Overflow line number for the page
- Number of characters printed per inch (CPI)
- Name of the output queue that will hold a spooled printer file
- Print quality used by the printer

The following are the main elements of printer support:

Application program A high-level language program that creates the data to be printed.

Printer device description
Describes the printer device to the system, including device name, type of printer, and device address.

Printer device file Describes how the printed output will look.

Device files describe how the system is to operate on data as it passes between your program and a device. Printer device files perform this function for printers. In this chapter printer device files are referred to as *printer files.*

The printer file to be used to print a report is specified in the program creating the report. The printer file contains information used to create a spooled file and to direct the file to the correct output queue or printer in the specified format.

The printer file must not be confused with the spooled file. The printer file describes what the printed output will look like and contains instructions to the system about how to build the spooled file. The spooled file contains the output from the user's program.

Note: If direct output is being used instead of spooling, no spooled file is created. The output from the program is printed directly from the application program.

Related CL Commands

The following CL commands are used to control printer support functions.

Printer Device Description Commands

CHGDEVPRT Change Device Description for Printer: Changes some attributes of a printer device description, such as the message queue used by the printer or the font type.

CRTDEVPRT Create Device Description for Printer: Creates a device description to describe a printer to the system.

Printer File Commands

CHGPRTF Change Printer File: Changes some attributes of the printer file, such as page size and lines per inch.

CRTPRTF Create Printer File: Creates a printer file containing the file description. The printer file contains no data.

DLTF Delete File: Deletes files.

DSPFD Display File Description: Displays the current characteristics of a file.

DSPFFD Display File Field Description: Displays the structure of a file that was created using DDS.

OVRPRTF Override with Printer File: Replaces the printer file or certain parameters of the printer file specified for the current application.

Detailed information about these commands can be found in the *CL Reference.*

Supported Printer Families

The printers that can be attached to the AS/400 system use one of the following printer data streams:

- SNA Character Stream (SCS)

  The Systems Network Architecture (SNA) Character Stream uses EBCDIC controls and formatted data. SCS printers are the 3287, 3812, 4214, 4234, 4245, 5219, 5224, 5225, 5256, 5262, and 6262.

- Intelligent Printer Data Stream (IPDS)

  IPDS is an all-points-addressable data stream that allows the user to position text, images, and graphics at any defined point on a printed page. IPDS printers include the 3812, 3816, 3820, 3825, 3827, 3835, 4224, and the 4234 Model 12.

- ASCII Data Stream

  ASCII data streams use ASCII control characters and data. ASCII printers include the Proprinter[1] X2. This data stream is generated only through PC Support.

One of the attributes specified by the printer file is the data stream language to be used by the system when that printer file is used. A printer understands only one

[1] Proprinter is a registered trademark of the International Business Machines Corporation.

data stream. If a spooled file created with one data stream (such as SCS) is printed on a printer that understands a different data stream (such as IPDS), the system will translate the printer commands into the correct language. If all commands cannot be translated, the system will send the operator a message.

Formatting Printer Output

The following factors affect what printed output will look like. Factors that affect only certain printers are noted with the model numbers of the affected printers.

- Printer or output queue to use.
- Page size (length and width).
- Overflow line. When this position on a page is reached, the application program may perform special end-of-page processing, such as a footings routine.
- Number of lines per inch (LPI).
- Number of characters per inch (CPI) (4214, 4234 SCS, 5224, and 5225).
- Font identifier to be used with the file (for printers that support fonts: 3812, 3816, 5219, and all IPDS printers).
- Data stream language (SCS or IPDS).
- Column positioning of text on a line.
- Control of line spacing.
- Folding or truncation. Whether a line of text is to be folded when it does not fit on one line of the form. If a line of text is folded, it continues on the next line of the form. If it is not folded, the line of text is truncated at the page width (all SCS printers).
- Page numbering.
- System date and time stamps.
- System assistance in form alignment.
- Data editing.
- Underlining.
- Highlighting.
- Print quality (4214, 4224, 4234, 5219).
- Graphic character set and code page used in printing output.
- Paper drawer for printers with automatic feeding of cut sheets (3812, 3816, 3820, 3825, 3827, 4214, 4224, and 5219).
- First character forms control (FCFC).
- Number of copies to print.
- Number of spooled file separators between spooled files.
- Character size, including height and width (all IPDS printers).
- Color (4224).
- Bar code. Many different types of machine-readable bar codes can be printed (3812, 3816, 4224, and 4234 IPDS printers).
- Page rotation. Page rotation specifies the degree to which the output page should be rotated (3812, 3816, 3820, 3825, 3827, and 3835).
- Unprintable characters. The replace unprintable character option specifies whether unprintable characters are to be replaced and what substitution character is to be used.
- Duplex printing. Whether you print on one side or both sides of the paper (3820, 3825, and 3827).

Producing Printer Output

Two methods can be used to produce printed output: spooling and direct output. When spooling is selected (by specifying SPOOL(*YES) on the CHGPRTF, CRTPRTF, or OVRPRTF commands), the user's data to be printed is put into a spooled file in an output queue. The OUTQ parameter determines which output queue is used. Many users can put their printed output into the same output queue. A printer writer program then takes the files from the output queue and prints them on the printer. This allows sharing of the printer by multiple users. See Chapter 6, "Spool Support" for more information about spooling.

Direct output does not involve spooling; the user's data is sent directly to the printer and only one program can use the printer. Direct output is selected by specifying SPOOL(*NO) on the CRTPRTF, CHGPRTF or OVRPRTF commands, and specifying the desired printer on the DEV parameter. Once the application program has run, the system no longer has a copy of the user's output.

Printer Files

Printer files provide information about how the user's data will be positioned on a page when printed. Two types of printer files can be used:

- Program-described printer files
- Externally described printer files

Program-Described Printer Files

You can create your own printer file with the CRTPRTF command, or you can use one of the printer files supplied by IBM.

QPRINT	A general purpose printer file. Data that is printed using printer file QPRINT is spooled to the output queue indicated by *JOB.
QPRINT2	A printer file for printed output requiring 2 copies. Data that is printed using QPRINT2 is spooled to output queue QPRINT2.
QPRINTS	A printer file for printing special forms. Data that is printed using QPRINTS is spooled to output queue QPRINTS.

When you use program-described data with a printer file, all formatting and control information must be specified in the high-level language program that uses the file. This control information is created by the compiler based on the output or printer specifications in the high-level language program.

A program-described printer file is created by specifying SRCFILE(*NONE) on the CRTPRTF command.

Externally Described Printer Files

For externally described printer files, the information that describes where to position data on the page is contained in the printer file. Data description specifications (DDS) are used to describe how the printed output will appear on the page. An externally described printer file is created by specifying the name of the source file and source member on the CRTPRTF command.

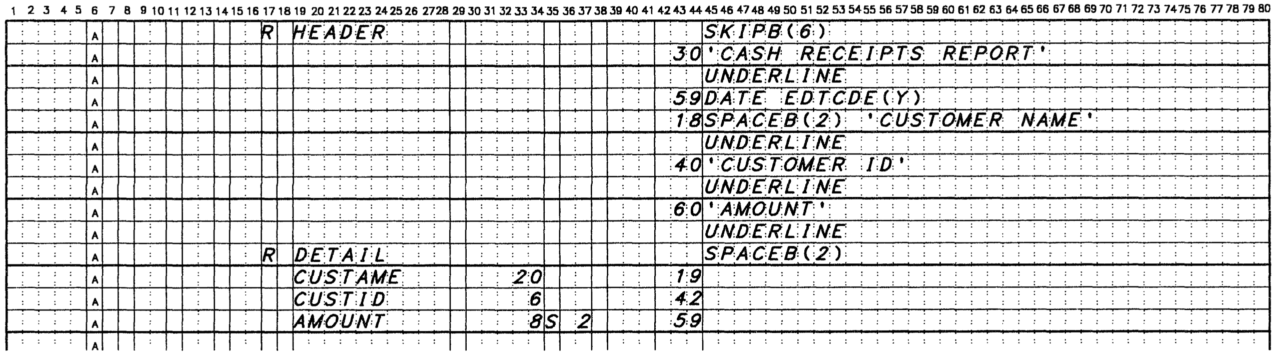
To focus on the contents of DDS input, headings for DDS forms within the chapters of this manual are purposely omitted. For your reference, Appendix G, "DDS Coding Form" shows a sample DDS coding form with the section headings.

Externally Described Printer File Example

In the following discussion, a cash receipt report is used as an example. The cash receipt report will be designed to:

- Have a page length of 66 lines
- Have a page width of 132 columns
- Be printed at 6 lines per inch

The following DDS example could be used to format the printed output for the cash receipt report:



```
     A          R HEADER                    SKIPB(6)
     A                                    30'CASH RECEIPTS REPORT'
     A                                      UNDERLINE
     A                                    59DATE EDTCDE(Y)
     A                                    18SPACEB(2) 'CUSTOMER NAME'
     A                                      UNDERLINE
     A                                    40'CUSTOMER ID'
     A                                      UNDERLINE
     A                                    60'AMOUNT'
     A                                      UNDERLINE
     A          R DETAIL                    SPACEB(2)
     A            CUSTAME       20        19
     A            CUSTID         6        42
     A            AMOUNT         8S 2     59
     A
```

RSLH191-0

Assuming that the source for the DDS is in a file named MYFILE, and the source member is named CASH, a printer file using these specifications can be created using the following command:

```
CRTPRTF  FILE(CASHRPT) SRCFILE(MYFILE) SRCMBR(CASH)
```

The externally described printer file, CASHRPT, can be used to print the cash receipt report as shown:

```
                 CASH RECEIPTS REPORT      06/02/89

CUSTOMER NAME        CUSTOMER ID           AMOUNT

 P G LOCHNER           480189             16562.57
 C A BATEMAN           350197             11104.67
 A D VANCE             341706              9570.31
 S P RABBITT           976726             14570.15
 T C DEAN              106293             21570.50
```

RSLH192-1

The cash receipt report uses both constant and variable data passed from the user's program. The program will do a WRITE operation for each entry in the report, passing as printable data the customer name, customer ID, and the cash amount. The system will automatically add the current date and the underlining where specified. The printer file controls line spacing and column positioning of the data.

Note: The cash receipt report could be printed using a program-described printer file such as QPRINT. If a program-described printer file is used, information describing where to position the data on the page must be included in the user's program.

DDS for Externally Described Printer Files

You can use DDS for an externally described printer file to define records and fields and control the printing of information through spacing and skipping. One record format can control the printing of more than one line. When you enter the DDS, you can specify which line each field should be printed on as well as where the field should be printed on that line. You can use DDS to specify:

- Indicators to condition the printing of a field.
- Option indicators can be placed in a separate buffer (INDARA).
- Documentation for an indicator's purpose can be done (INDTXT).
- Edit words (EDTWRD) and edit codes (EDTCDE).
- Constant data may be included from the printer file (DFT).
- Constant data may be included from an already existing message (MSGCON).
- Underlining of fields (UNDERLINE).
- Bold printing (HIGHLIGHT).
- Colors for printing (COLOR) can be selected.
- Vertical relative line spacing (SPACEA, SPACEB).
- Vertical absolute line skipping (SKIPA, SKIPB).
- Folding onto a new line (BLKFOLD) at a blank in the user's data.
- Characters per inch (CPI) can be varied within a file.
- Lines per inch (LPI) can be varied within a file.
- Fonts (FONT) can be varied within a file. Graphic fonts can also be selected.
- User-defined characters (DFNCHR and TRNSPY) allow creation of characters not supplied by the printer.
- Paper source drawer (DRAWER) can be varied within a file.
- Page rotation (PAGRTT) can be varied within a file.
- The height and width of characters (CHRSIZ) can be expanded for regular and graphic fonts.
- Bar codes (BARCODE) can be printed.
- Print quality (PRTQLTY) can be draft, near-letter, or standard for fields that are bar codes or fields for which a character size has been specified.
- A hex string to be printed (CVTDTA). This allows printing of characters not on the keyboard.
- A graphic character set and code page other than the default for the device (CHRID) can be selected for parts of the file.
- An alternative name for a field can be specified (ALIAS).
- For documentation purposes, a text string can be specified (TEXT).
- A field description can be duplicated from a database file (REF, REFFLD, and DLTEDT).
- A floating point field can be printed in fixed decimal notation (FLTFIXDEC).
- The precision of a floating point field can be specified (FLTPCN).

You can also specify system functions that should be performed when a file is printed. These functions print page numbers, the system date, and the system time (PAGNBR, DATE, and TIME).

The order in which fields are specified for the record format determines the order in which they must be used in the record for a using program.

The fields do not need to be specified in the order in which they appear, but print speed will be best if they are.

For complete information about DDS for printer files, see the *DDS Reference*.

Controlling Spacing and Skipping Lines

The printing of lines can be controlled by spacing and skipping through an externally described printer file. Spacing means to advance one line at a time; skipping means to move to a specific line number.

In the DDS for a printer file, you specify spacing and skipping through keywords:

SPACEA Spaces a specified number of lines (1 through 255) *after* printing.

SPACEB Spaces a specified number of lines (1 through 255) *before* printing.

SKIPA Skips to a specified line number (0 through 255) *after* printing.

SKIPB Skips to a specified line number (0 through 255) *before* printing.

DDS Keywords

Table 5-1 on page 5-8 shows which printer DDS keywords are supported on each AS/400 printer. Not all printers supported by the AS/400 system support the same printer DDS keywords.

Table 5-1. Printer DDS Keyword Support

DDS Keyword	3287, 4245 5256, 5262, 6262 SCS Printers	3812 SCS Printer	4214 SCS Printer	4234 SCS Printer	5224, 5225 SCS Printers	5219 SCS Printer	3812, 3816, 4224, 4234 IPDS Printers	3820, 3825, 3827, 3835 IPDS Printers
ALIAS	x	x	x	x	x	x	x	x
BARCODE							x	
BLKFOLD	x	x	x	x	x	x		
CHRID		x	x	x	x	x	x	x
CHRSIZ							x	
COLOR							x[1]	
CPI			x	x	x			
CVTDTA			x	x	x		x	
DATE	x	x	x	x	x	x	x	x
DFNCHR			x[2]	x[2]	x			
DFT	x	x	x	x	x	x	x	x
DLTEDT	x	x	x	x	x	x	x	x
DRAWER		x	x			x	x[3]	x
EDTCDE	x	x	x	x	x	x	x	x
EDTWRD	x	x	x	x	x	x	x	x
FLTFIXDEC	x	x	x	x	x	x	x	x
FLTPCN	x	x	x	x	x	x	x	x
FONT							x	x
INDTXT	x	x	x	x	x	x	x	x
INDARA	x	x	x	x	x	x	x	x
HIGHLIGHT	x	x	x	x	x	x	x	x
LPI							x	x
MSGCON	x	x	x	x	x	x	x	x
PAGNBR	x	x	x	x	x	x	x	x
PAGRTT		x					x[3]	x
PRTQLTY							x[4]	
REF	x	x	x	x	x	x	x	x
REFFLD	x	x	x	x	x	x	x	x
SKIPA	x	x	x	x	x	x	x	x
SKIPB	x	x	x	x	x	x	x	x
SPACEA	x	x	x	x	x	x	x	x
SPACEB	x	x	x	x	x	x	x	x
TEXT	x	x	x	x	x	x	x	x
TIME	x	x	x	x	x	x	x	x
TRNSPY			x	x	x			
UNDERLINE	x	x	x	x	x	x	x	x

[1] Keyword is accepted but ignored by the 3812, 3816, and 4234 IPDS printers.

[2] The 4214 and 4234 Printers will accept the DFNCHR DDS keyword. However, the DFNCHR keyword only supports an 8 by 9 matrix pattern (5224 and 5225 Printers). The 4214 Printer supports an 8 by 10 matrix pattern while the 4234 Printer supports an 8 by 9 matrix pattern at 10 characters per inch and an 8 by 12 matrix pattern at 15 characters per inch. For the 4214 Printer, the ninth bit is propagated to the tenth bit to form an 8 by 10 matrix pattern. For the 4234 Printer at 15 characters per inch, the tenth, eleventh, and twelfth bit positions are always off. Characters defined with the DFNCHR DDS keyword should be tested on the 4214 and 4234 Printers to determine if print quality is acceptable.

[3] Keyword is accepted but ignored by the 4224 Printer and 4234 IPDS Printer.

[4] Applies only to the 4224-1E3 Printer and 4234 IPDS Printer.

When the printer file is created (using the CRTPRTF command), conflicting DDS keywords can be specified in the same printer file, but conflicting DDS keywords cannot be specified on the same record format. (Any combination of printer DDS keywords that are not supported by a single printer is considered conflicting.)

For example, BARCODE can be specified on record format A, and DFNCHR on record format B; but BARCODE and DFNCHR cannot both be specified on record format A, since no printer supports both DDS keywords. If both keywords are specified on the same record format, then a level 30 diagnostic message is issued and the printer file is not created.

Open Processing for Printers

Purpose for Open Processing

A printer file is opened to prepare the system so that the application can put data into a spooled file or print it out directly to a printer. Information from the high-level language application program, the printer file, and any printer file overrides is combined. The system takes the final results, and determines such factors as:

- The printer data stream language to use (SCS or IPDS)
- Whether folding is to be done
- How unprintable characters are to be treated
- The LPI, CPI, font and page size

These are referred to as attributes of the printer file.

In addition, open processing includes the following:

- Validity checking. For example, the system determines if the page size selected is supported.
- Building internal control blocks. These will be used to build the spooled file correctly.
- Filling in the open feedback area. This provides the high-level language program with the status of the open printer file.

Merging the Factors Controlling Open Processing

The printer file open is controlled by parameters specified in the printer file (from the CRTPRTF and CHGPRTF commands), in the high-level language program, and in printer file overrides (on the OVRPRTF command). These controls are processed in the following order:

1. From the printer file
2. From the high-level language application
3. From the overrides

As an example, if the CRTPRTF command specified an LPI of 8, and an OVRPRTF command specified an LPI of 6, the LPI of 6 would be used since the override value takes precedence over the CRTPRTF value.

Open Considerations

The following should be considered for opening printer files:

Record Length

The record length is required for program-described data files. It specifies the number of bytes of data to be printed for each output operation. This record length is specified through the high-level language application. See the appropriate HLL manual for instructions on setting this value.

Page Size, Lines Per Inch, and Characters Per Inch

The page size (PAGESIZE Parameter) is specified in length (number of lines) and width (number of characters). If the record length is longer than the page width, the records are either truncated (*NO specified for FOLD parameter) or continued on the next line (*YES specified for FOLD parameter). Folding is never done for files opened with DEVTYPE(*IPDS) specified.

Two other factors that control the size of the printed output are the lines per inch (LPI) and the pitch, or characters per inch (CPI). LPI and CPI are parameters that can be specified on the printer file. For correct printing, the page size must be compatible with the LPI and the CPI. For example, if the page to be printed is 11 inches long, and 66 lines are needed on each page (page length is 66), the LPI value should be 6 (66/6=11).

For printers that support fonts, the value for CPI in the printer file is not used to determine the pitch of the printed output if a value for the FONT parameter is specified in the printer file. If FONT is specified, the pitch is determined by the CPI value associated with the font. Table 5-5 on page 5-27 lists the CPI values associated with each font. See "Printer Font Support" on page 5-23 for more information on the relationship between fonts and CPI.

The page size should also take into account the page rotation (PAGRTT) value. The page size should match the PAGRTT value. For example, if the page is rotated so that it will be 11 inches wide and 8-1/2 inches long, the page size (PAGESIZE) parameter should indicate a page 11 inches wide by 8-1/2 inches long. A length of 51, a width of 110, with 6 LPI, 10 CPI, and FONT(*CPI) would be correct.

Each entry in the following table shows the valid range of values for lines per page for each printer type and for each value of lines per inch (LPI) valid for the printer.

Table 5-2. Lines per Page (PAGESIZE Parameter)

Printer	4 Lines per Inch	6 Lines per Inch	8 Lines per Inch	9 Lines per Inch
3287	1-104	1-104	1-104	–
3812 SCS	1-56	1-84	1-112	1-126
3812 IPDS	2-56	2-84	2-112	2-112
3816 SCS	1-56	1-84	1-112	1-126
3816 IPDS	2-56	2-84	2-112	2-112
3820	1-56	1-84	1-112	1-126
3825	1-56	1-84	1-112	1-126
3827	1-56	1-84	1-112	1-126
3835	2-91	2-136	2-182	2-204
4214	1-255	1-255	1-255	1-255
4224	2-91	2-136	2-182	2-204
4234	1-255	1-255	1-255	1-255
4245 Models 12 and 20	–	2-144	2-192	–
4245 Models T12 and T20	–	1-255	1-255	–
5211	–	2-84	2-112	–
5219 Continuous Forms	2-255	2-255	2-255	–
5219 Cut Sheet	57	86	114	–
5224	1-255	1-255	1-255	1-255
5225	1-255	1-255	1-255	1-255
5256 (set manually)	–	1-255	1-255	–
5262	–	1-255	1-255	–

Each entry in the following table shows the valid range of values for characters per line for each printer type and for each value of characters per inch (CPI) for the printer.

Table 5-3. Characters per Line (PAGESIZE Parameter)

Printer	5 Characters per Inch	10 Characters per Inch	12 Characters per Inch	13.3 Characters per Inch	15 Characters per Inch	16.7 Characters per Inch
3287	–	1-132	–	–	–	–
3812[1]	1-42	1-85	1-102	–	1-127	–
3812[1] Rotated Form	1-70	1-140	1-168	–	1-210	–
–						
3816[1]	1-42	1-85	1-102	–	1-127	–
3816[1] Rotated Form	1-70	1-140	1-168	–	1-210	–
3820[1]	–	1-85	1-102	–	1-127	–
3825[1]	–	1-85	1-102	–	1-127	–
3827[1]	–	1-85	1-102	–	1-127	–
3835[1]	–	1-132	1-158	–	1-198	–
4214 Continuous Forms	1-66	1-132	1-158	–	1-198	1-220
4214 Cut Sheet	1-60	1-120	1-144	–	1-180	1-200
4224[1]	–	1-132	1-158	–	1-198	1-220
4234[1]	–	1-132	–	–	1-198	1-238
4245	–	1-132	–	–	–	–
5219	–	1-132	1-158	–	1-198	–
5224	–	1-132	–	–	1-198	–
5225	–	1-132	–	–	1-198	–
5256 Model 3	–	1-132	–	–	–	–
5262	–	1-132	–	–	–	–

[1] Many character-per-inch values (implied by the pitch of the font), are supported in addition to the ones listed here. To find the maximum characters per line, multiply the implied characters per inch value listed in the Table 5-5 on page 5-27 by the maximum page width supported (in inches). (The maximum page width supported by the 3812 and 3816 Printers is 8.5 inches for non-rotated forms and 14.0 inches for rotated forms.)

Nonspooled Input/Output

Only one printer file can be opened to a printer at a time when using nonspooled I/O.

Level Checking

Level checking consists of comparing the level identifiers of the record formats in the file the program was compiled with and the identifiers of the current file. If the level identifiers do not match, an error message is sent to your program. If you are sure that the changes in the record format do not affect your program, you can specify LVLCHK(*NO) on an OVRPRTF command so that your program can run. You can specify LVLCHK(*NO) initially when you create the file. More information on level checking is provided under "Effect of Changing Fields in a File Description" on page 5-38.

Sharing Printer Files

A printer can be shared by two or more programs that run in the same routing step if each program opens the same printer file with SHARE(*YES) specified. This lets you produce one continuous output list from two or more programs.

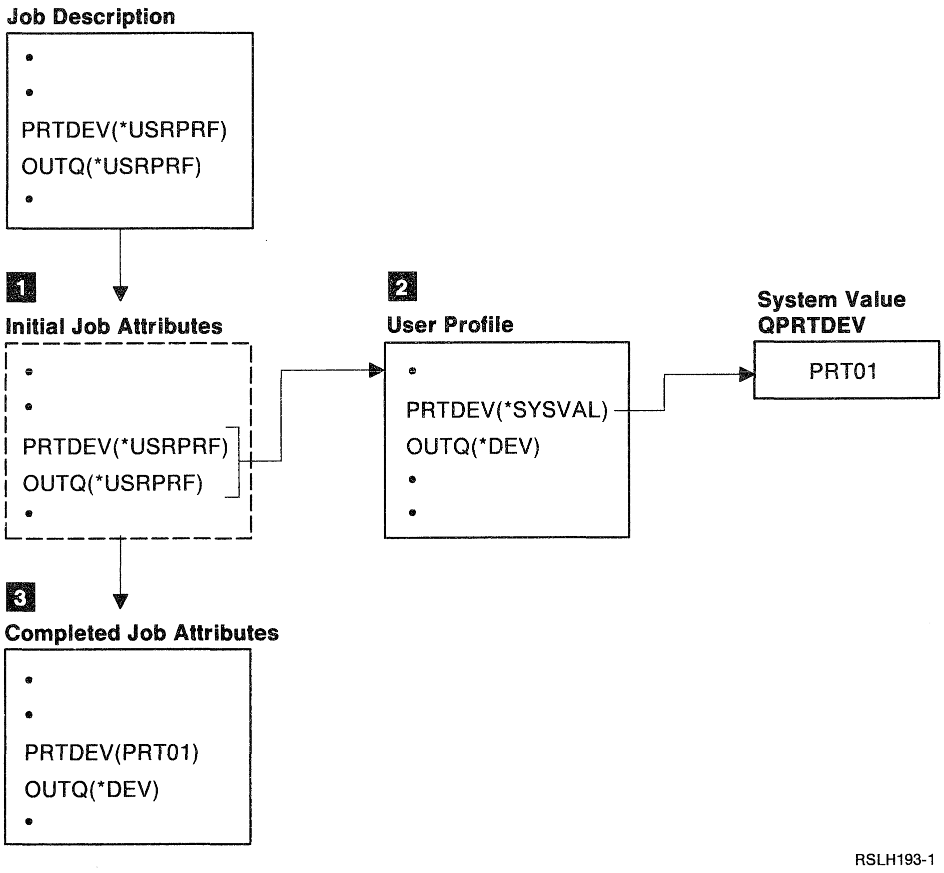
Specifying Output Queue and Printer Device Values

When a job starts, it always has both a job description and a user profile. (See the following illustration.)

1 The job description is copied into the job and provides the initial values for the job attributes. Some of the job description parameters have values that default to the user profile. The two that are important to this discussion (PRTDEV and OUTQ) both have a default value of *USRPRF.

2 If the defaults are used, the real values are accessed from the user profile. The user profile value for PRTDEV defaults to the QPRTDEV system value.

3 When job initialization is complete, the two attributes are filled.

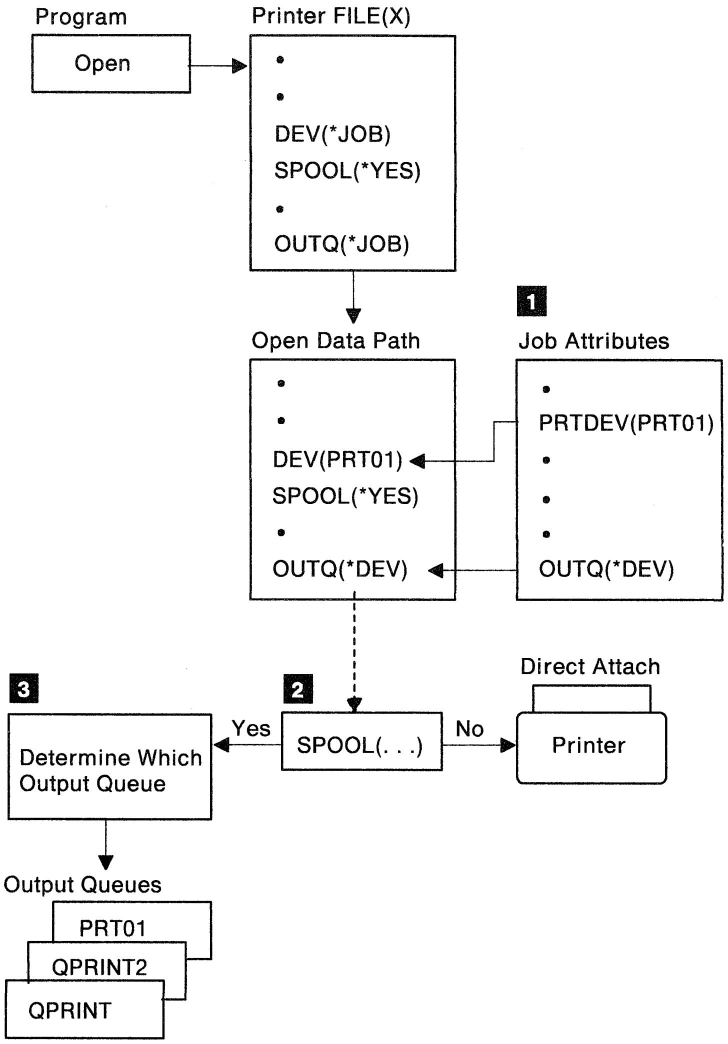


These attributes provide the base values for the job. The printer file also has the DEV and OUTQ parameters that may specify to use these job attributes. The default is *JOB.

Note: The OUTQ parameter can specify *DEV. If the printer file is directed to use the job attributes, the output queue to be used is the queue associated with the DEV parameter specified in the printer file. The DEV attribute contains a name of a printer. The OUTQ parameter contains either an output queue name or the value *DEV.

Once the job starts, the values in the job description, the user profile, or the system value are no longer used to determine the job attributes. Any changes are ignored for active jobs. The CHGJOB command can be used to change the PRTDEV or OUTQ job attributes while the job is active.

When a program opens a printer file, the open routine goes through a series of steps to determine the output queue and device.



RSLH144-1

1 If OUTQ(*DEV) is specified for the device file, the device is determined by the DEV parameter as follows:

DEV(*JOB)	Use the job attribute PRTDEV
DEV(*SYSVAL)	Use the name in the QPRTDEV system value
DEV(xxx)	Use the xxx printer device

In the previous illustration, the open data path DEV attribute of *JOB causes the job attribute of PRT01 to be used.

2 The device file is checked to see if it should be directly attached to a printer. Attached directly means that you are not using spooling, but will specifically allocate a physical printer device. If SPOOL(*NO) is specified in the device file, direct attach printing is requested and step 3 would not occur.

3 The OUTQ to use is determined. Assuming SPOOL(*YES) in the open data path, the OUTQ parameter in the open data path is tested.

OUTQ(*JOB) Use the job attribute OUTQ
OUTQ(*DEV) Use the DEV name determined in step 2
OUTQ(xxx) Use the xxx output queue

Note: If *DEV is specified, the output queue selected is the queue associated with the device.

While it appears that the keyword values are pointing at a device, they are really pointing at the output queue associated with the device.

FORMTYPE Parameter

The FORMTYPE parameter specifies the type of form used in the printer when the output is printed.

If your program is sending output directly to the printer (SPOOL(*NO)), the form type specified in the printer file is compared to the form type loaded on the printer. If the form types are not the same, a message is sent to the system operator with the following options:

- Continue to print on the currently loaded form type.
- Cancel the processing.
- Load the correct form type and continue to print.

If your program is sending output to a spooled file, the comparison of form types occurs when the spool writer processes the spooled file.

Page Rotation

When DEVTYPE(*IPDS) is specified for the printer file, and printing is to be done in computer output reduction mode (PAGRTT(*COR)) on an IPDS printer, the printer file cannot contain any graphic, bar codes, variable LPI, or variable font. If the printer file contains these functions, the output will not be rotated and reduced. See the PAGRTT parameter on the CRTPRTF command in the *CL Reference* for a description of *COR mode.

Fonts

The FONT parameter (3812, 3816, 3820, 3825, 3827, 3835, 4224, 4234 IPDS, and 5219 Printers only) specifies the font to be used with the device. Either *CPI, *DEVD or a 3- or 4-digit identifier may be used to specify a font ID. This identifier corresponds to a particular font style. For a chart showing these identifiers, see Table 5-5 on page 5-27. The value *DEVD indicates that the font ID specified in the device description is to be used. The value *CPI indicates the system will select a font using the pitch of the CPI parameter.

Proportionally spaced fonts are approximately equal to 12 characters per inch. A proportionally spaced font has characters that are not of equal width. For example, an uppercase `W` is a wide character, a lowercase `i` is a narrow character. If proportionally spaced fonts are sent to the printer and if the data attempts to print beyond the form width, the printer does not fold data to a new line. An error message indicating invalid data results.

You should not use proportionally spaced fonts with externally described printer files, because column positions are not as meaningful, and overprinting and gaps can occur in the output.

Paper Source Drawers and Form-feed Mode

The DRAWER parameter (3812, 3816, 3820, 3825, 3827, 4214, 4224-1E3, and 5219 Printers) specifies the paper source drawer to use when in automatic cut sheet mode. Drawer 1 (the default), drawer 2, drawer 3, or the envelope slot may be selected. Specify DRAWER(1), DRAWER(2), or DRAWER(3) for cut sheets; or DRAWER(*E1) for envelopes. This parameter is only valid when FORMFEED(*AUTOCUT) is specified. (For the 3812 and 3816 Printers, FORMFEED(*AUTOCUT) is always assumed.)

Table 5-4 shows which DRAWER parameter values are supported on each printer.

Table 5-4. Printer Drawer Support

Drawer	**3812 Printer**	**3816 Printer**	**3820, 3825, 3827 Printers**	**4214 Printer**	**4224-1E3 Printer**	**5219 Printer**
1	x	x	x	x	x	x
2	x	x	x	x	x	x
3				x	x	
*E1				x	x	x

The FORMFEED parameter allows selection of one of three form-feed modes for the 4214, 4224-1E3, or 5219 Printer. The three modes are:

*CONT Indicates that continuous forms will be used. The tractor attachment must be put on when using this mode.

*CUT Indicates that single cut sheets will be manually fed into the printer.

*AUTOCUT Indicates that automatic form feeding of single cut sheets will be done. The automatic sheet feed attachment must be put on when this value is specified.

The value *DEVD (the default) may be used to indicate that the form feed value specified in the device description is to be used. Whenever there is a change of form feed type (*CUT or *CONT or *AUTOCUT), an inquiry message is sent to the message queue associated with the device requesting the operator to put on the appropriate attachment. The forms alignment message (CPA4002) and the end-of-forms message are not issued when *CUT or *AUTOCUT mode is used.

For manual feeding of envelopes, specify FORMFEED(*CUT). The end-of-forms messages will be suppressed for each sheet manually fed. Note that the automatic sheet feed may be left attached for manual feeding of envelopes or cut sheets. Refer to your printer operator's guide for more information.

FORMFEED(*AUTOCUT) DRAWER(*E1) may also be specified for manual feeding of envelopes when the 2-drawer automatic sheet feed attachment is put on. However, the end-of-forms message is not suppressed for each sheet manually fed.

Print Quality

The print quality (PRTQLTY) parameter controls the entire printer file. (The DDS print quality keyword applies only to the field or record on which it is specified.) PRTQLTY on the CHGPRTF, CRTPRTF, and OVRPRTF commands specifies the following:

For the 5219 Printer, different print qualities are produced by varying the speed at which the print ribbon advances. Quality mode (*STD or *NLQ) results in normal advancement of the print ribbon. In draft mode (*DRAFT), the ribbon advances at a rate of one-third the distance it advances in quality mode. The 5219 Printer has a converse ribbon switch that overrides the value of *DRAFT specified by this parameter.

For the 4214, 4224, and 4234 Printers, print quality is produced by varying the density of the dot matrix pattern used to create printable characters. Quality mode (*STD) is the normal mode (high density), and requires two passes by the printer to produce a line of data. Draft mode (*DRAFT) results in high speed printing; the entire character set can be printed in draft mode. Quality mode (*NLQ) requires multiple passes by the printer to produce a line of data.

For the 4214 Printer, standard quality mode (*STD) is only supported for 10 and 12 characters per inch. If PRTQLTY(*STD) and 5, 15, or 16.7 characters per inch are specified, data is printed in draft mode.

For the 4234 Printer, only a limited character set (62 characters) is supported when PRTQLTY(*DRAFT) is specified. See the *4234 Printer Operator's Guide* for a description of the character set supported with draft print quality.

For the 4224 Printer, the fonts supported are not available for all three print qualities. The OCR-A and OCR-B fonts are only supported with PRTQLTY(*NLQ). The Courier and Essay fonts are only available with PRTQLTY(*NLQ) and PRTQLTY(*STD). The Gothic font is only available with PRTQLTY(*DRAFT). If there is a mismatch between the print quality and font selected, the font will be changed to match the print quality. However, for OCR-A and OCR-B, the print quality will be changed to match the font.

For the 3812 SCS and 3816 SCS Printers, the automatic hardware selection of Computer Output Reduction printing selected through soft switches on the printers occur only when *DRAFT is specified for PRTQLTY and PAGRTT is *DEVD. If PAGRTT(*COR) is specified, the PRTQLTY parameter does not affect the printed output.

Duplex Printing

The DUPLEX parameter (3820, 3825, and 3827 Printers only) allows you to specify whether output should be printed on one or two sides of the paper. The following are valid values:

- *NO, print on one side only.
- *YES, print on both sides, where the top of each printed page is at the same end of the sheet of paper. This is usually used for output bound at the side.
- *TUMBLE, print on both sides, where the top of one printed page is at the opposite end from the top of the other printed page on the back side. This is usually used for output bound at the top.

For an SCS printer, or when DEVTYPE(*SCS) is specified for the printer file, the following DDS keywords are not supported. If any of these keywords are used, the keyword is ignored, a diagnostic message is issued, and processing continues.

BARCODE	Bar codes
COLOR	Color printing
FONT	Font selection
LPI	Lines per inch
PRTQLTY	Print quality

For an IPDS printer, or when DEVTYPE(*IPDS) is specified for the printer file, the following DDS keywords are not supported. If any of these keywords are used, the keyword is ignored, a diagnostic message is issued, and processing continues.

BLKFOLD	Blank fold
CPI	Characters per inch
DFNCHR	Define character

For direct output, DDS keywords that apply to the printer data stream may still be ignored on a write operation if the particular printer does not support the function represented by the keyword. For example, if the printer file specifies the CHRID keyword, it is valid for printers using the SCS data stream. However, if the printer being used is a 5256 Printer, the CHRID keyword will be ignored, since the 5256 does not support the CHRID function.

For spooling, this condition is detected when the writer is printing the spooled file. See “Printer File Redirection” on page 5-39 for more information.

- Forms alignment requires operator intervention and should be specified only on an as-needed basis. When forms alignment is specified, the first print line of the first record is printed and the system operator is sent a message requesting forms alignment. Be sure that enough data is printed to allow the operator to align the forms.

Force-End-of-Data Considerations: The force-end-of-data function is available to ensure that all the records are printed before you continue processing the next group of records. The force-end-of-data function is ignored on the 3812, 3816, 3820, 3825, 3827, and 3835 Printers because they buffer the entire page prior to putting the printed image on the paper. Once printed, the page is ejected, and nothing more can be added to the page.

Status on Output Completion

When the output operation has completed, control returns to the high-level language application program. The user's data has been merged with the necessary print commands based on the control information passed with the user's program or found in the externally described printer file. The data and controls are put into the spooled file. If direct printing is used, the data and controls are buffered up into full pages. As each page is finished, it is sent to the printer. The current information is stored in the I/O feedback area for printer files. This includes:

- Current line number
- Current page number
- Number of records that have been processed
- Major or minor return code of an error that has been detected

Close Processing for Printers

The close operation is done by the high-level language program when all the printing is finished for a printer file. This operation prints any last partial page for direct output. The temporary control blocks created at open time are deleted. The spooled file (or printer file for direct output) is deallocated.

Close Considerations

The 4210 Printer, configured as a 4214, may report that printing is complete when the data is in the buffer of the printer. For short printouts, the writer or job may end while the printer has data in the buffers and is still printing. In this case, no condition message (that states it is out of forms) is sent once the writer or job ends. Printing will not be completed until the printer is loaded with forms and made ready again, at which point any data in the buffers will continue to print.

If the Allocate Object (ALCOBJ) command was used to preallocate the printer, the printer remains allocated to the job.

Error Handling

Errors that can occur fall into two categories: Those that can be recovered from, and those that can not. Recoverable errors include such conditions as end of forms, end of ribbon, and printer cover open. There are others as well. Non-recoverable errors include such conditions as the printer signaling a device error. Appendix D, "Printer File Return Codes" contains a list of printer return codes and indicates which errors can occur when creating the spooled file.

When a non-recoverable error occurs, an escape message is signalled to the program doing the printing. If this is a printer writer printing spooled data, the printer writer will end, and the spooled file will remain on the output queue.

When a recoverable error occurs, the system action will depend on how the printer is configured. The PRTERRMSG parameter on the CRTDEVPRT and CHGDEVPRT commands allow the user to select one of two types of error recovery. If *INFO is selected, an informational message is signalled to the operator when an error is detected. The operator can correct the error condition, and allow the printer to continue from where it stopped. If *INQ is selected, an inquiry message is signalled to the operator when an error is detected. After the error condition is corrected, the operator is given several choices on how printing should continue. Depending on the particular error, the operator may select the page number to begin printing on, may continue printing right where the printer stopped, or may choose to hold the spooled file on the output queue and continue with a different file. The second level text of the inquiry message explains the possible options and the steps to be followed for each option.

For simpler error recovery, specify PRTERRMSG(*INFO) on the CRTDEVPRT or CHGDEVPRT CL commands. Less operator work is required to start printing again. The level of error recovery provided by the system is less sophisticated.

Setting PRTERRMSG(*INQ) will allow the system to provide more error recovery function, but more work is required from the operator. It is suggested that PRTERRMSG(*INFO) be used for customers who are less experienced with the printers attached to their AS/400 system.

When PRTERRMSG(*INQ) has been selected, many recoverable errors allow the user to enter the word PAGE as a reply to the inquiry message. This instructs the printer writer to restart printing on the current page. The Work with Writer (WRKWTR) command can be used to see which page the printer writer is on.

Because some printers, most notably IPDS printers, buffer large amounts of data prior to printing, the WRKWTR command may show a different page number than the one currently being printed. This is especially true if the pages are very small, or contain only a small amount of printable data. If this occurs, the operator may wish to enter a specific page number to restart on, rather than entering the word PAGE.

In a few cases, there is no informational message available for a recoverable error condition. For these cases an inquiry message will be sent regardless of the value of the PRTERRMSG parameter.

For some errors, the 4224 requires the operator to press the CANCEL key to clear the printer of an error. If this happens when PRTERRMSG(*INFO) has been specified, a message will be sent to the operator indicating that the cancel key has been pressed and some data will not be printed.

Special Printer Considerations

First Character Forms Control Data

For program-described printer files, you can also specify the print control information in the data itself. You can do this by including an ANSI first-character forms-control code in position 1 of each data record in the printer file. (You cannot use first-character forms control and DDS on the same file.)

To include the print control information in the data, you specify one of the following ANSI first-character forms-control codes in the first position of each data record:

ANSI Control Code	Action before Printing a Line
' '	Space one line (blank code)
0	Space two lines
-	Space three lines
+	Suppress space
1	Skip to channel 1
2	Skip to channel 2
3	Skip to channel 3
4	Skip to channel 4
5	Skip to channel 5
6	Skip to channel 6
7	Skip to channel 7
8	Skip to channel 8
9	Skip to channel 9
A	Skip to channel 10
B	Skip to channel 11
C	Skip to channel 12

Any other character in position 1 of a record defaults to a blank (the ANSI code for spacing one line). If this occurs, the notify message CPF4916 is sent to the high-level language program once per file.

When you use first character forms control data for a printer file, the print control information created by the high-level language compiler is ignored. The character in position 1 of the record is used as the print control character for that record.

To create a program-described printer file that uses first-character forms-control data, specify the CTLCHAR parameter and, optionally, the CHLVAL parameter on the Create Printer File (CRTPRTF) command. CTLCHAR(*FCFC) specifies that the first character in every record is an ANSI forms-control code.

The CHLVAL parameter allows you to associate a specific skip-to line number with an ANSI channel identifier. For example, if you specify CHLVAL(2 20), channel identifier 2 is allocated with line number 20; therefore, if you place the forms-control 2 in the first position of a record, the printer skips to line 20 before printing the line.

Each control identifier can be specified only once on the parameter. If no line number is defined for a channel identifier and that channel identifier is encountered in the data, the printer takes the default of spacing one line before printing.

In the following example, a file, PRTFCFC, that uses first-character forms-control data, is created:

```
CRTPRTF  FILE(QGPL/PRTFCFC) OUTQ(PRINT) CTLCHAR(*FCFC) +
CHLVAL((1 1) (2 10) (12 60))
```

The printer output is spooled to the output queue PRINT. Channel identifier 1 is associated with line 1, channel identifier 2 is associated with line 10, and channel identifier 12 is associated with line 60.

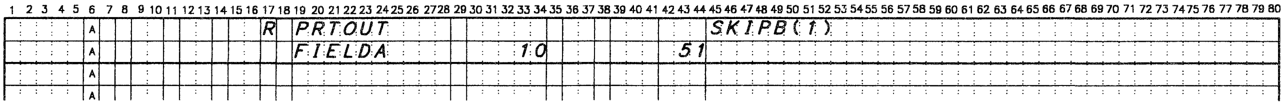
Printer Font Support

Characters Per Inch Versus Font

Most printers use the CPI parameter on the CRTPRTF, CHGPRTF, or OVRPRTF command to determine the pitch (characters per inch) of the printed output. However, the 3812 SCS, 3816, 5219, and all IPDS printers use the FONT parameter to select both an implied pitch (characters per inch) and a font style. The CPI parameter is ignored when using these printers. Conversely, the FONT parameter is ignored on printers where it is not applicable.

To provide flexibility in your print job, where possible, the pitch of the font identifier (FONT parameter) should match the value specified on the CPI parameter. Thus a print job intended for a printer that supports fonts can then be printed on another printer without significant change in the appearance of the printed output. For example, a printer file which has FONT(222), Gothic font with 15 pitch, and CPI(15) could print on a 5219, 3812, or 4224 Printer (which use the FONT parameter) or also could print on a 4214, 4234, 5224, or 5225 Printer (which support 15 characters per inch). If this print job were directed to a printer that only supports 10 characters per inch, then printer file redirection would be used. By setting the FONT parameter to FONT(*CPI) the system will select a font of the same pitch as the CPI parameter value. See “Printer File Redirection” on page 5-39 for more information.

When using an SCS externally described printer file, normally the value specified in the CPI parameter is used to position fields on the printed page. For example, if a printer file has 10 characters per inch specified, and FIELDA is specified to start in column 51, then there would be 50 blanks to the left of FIELDA (50 blanks at 10 characters per inch is 5 inches).



```
     A          R PRTOUT                    SKIPB(1)
     A            FIELDA        10        51
     A
     A
```

RSLH114-0

For the IPDS externally described printer files, the pitch implied by the FONT parameter on the CRTPRTF, CHGPRTF, and OVRPRTF commands is used instead of the CPI parameter or the value of the FONT DDS keyword to determine the starting column of fields on a printed page. In the example above, if CPI(10) and FONT(087) had been specified, and the printer specified was one that supports fonts, then FIELDA specified to start in column 51 would be 50 blanks at 12 characters per inch (implied CPI value for font 087) or 50/12 inches, which is 4.167 inches in from the left margin of the paper.

Proportionally Spaced Fonts

The 3812, 5219, and all IPDS printers support proportionally spaced fonts. For proportionally spaced fonts, characters vary in width depending on the character being printed (for example, i is a narrow character, and W is a wide character). See Table 5-5 on page 5-27 for a list of all supported fonts. The implied characters-per-inch column in this table lists the value of the width of a blank character for the font selected.

When using proportionally spaced fonts, fold and truncation (FOLD parameter) may not work as intended. This is because the system does not keep track of the width of each individual character. See “Folding Records in a Printer File” on page 5-43 for more information.

The 3812, 3816, 3820, 3825, 3827, and 3835 Printers also support typographic fonts. A typographic font is specified by point size (height of font). A point measures 1/72 of an inch, so an 8-point font would be 1/9 of an inch high and a 24-point font would be 1/3 of an inch high. When using fonts that contain tall characters, double or triple spacing may be needed to avoid having lines overlap when the page is printed.

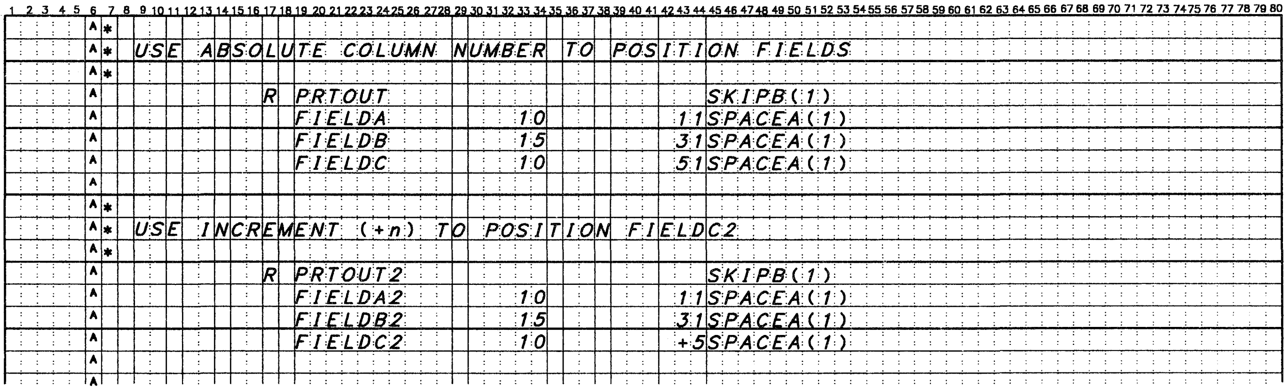
Because proportionally spaced and typographic fonts have characters of variable widths, care should be taken with the use of underlining and overstriking. The highlighting or underlining method of printing a line with a space after (SPACEA) value of 0 followed by printing another line may not work correctly.

Proportionally spaced and typographic fonts may be specified on the CRTPRTF, CHGPRTF, or OVRPRTF command when using an externally described printer file. As noted above, the implied characters-per-inch value of the font identifier specified on the FONT parameter is used to position fields on a printed page. The same rule for positioning fields on a printed page is used with proportionally spaced and typographic fonts. The width of a blank character is used to position fields on a page. Output should be tested to see that using externally described printer files with proportionally spaced fonts produces satisfactory results because overprinting and gaps can occur in the output.

Note: The amount of printed space for a field varies depending on which characters are in a field. Enough space should be left between fields to allow for the widest characters (uppercase characters) expected in that field.

In the previous example, if CPI(10) and FONT(1351) had been specified, and the printer used were a 3812 Printer, then FIELDA specified to start in column 51 would be 50 blanks at 17.14 characters per inch (implied CPI value for font 1351) or 50/17.14 inches, which is 2.975 inches from the left margin of the paper.

In this example, FIELDA would start 2.975 inches from the left margin of the paper regardless of how many fields were defined to its left. When using an increment value (+n) instead of a column number (positions 42 through 44 in DDS specification), fields are positioned the same for proportionally spaced fonts as they are for fixed pitch fonts. That is, the field is positioned based on the width of the blank for the specified font. The following example illustrates that using either absolute column numbers or relative increment numbers (+n) will supply the same result.



```
     A*
     A*  USE ABSOLUTE COLUMN NUMBER TO POSITION FIELDS
     A*
     A          R PRTOUT                    SKIPB(1)
     A            FIELDA        10        11SPACEA(1)
     A            FIELDB        15        31SPACEA(1)
     A            FIELDC        10        51SPACEA(1)
     A
     A*
     A*  USE INCREMENT (+n) TO POSITION FIELDC2
     A*
     A          R PRTOUT2                   SKIPB(1)
     A            FIELDA2       10        11SPACEA(1)
     A            FIELDB2       15        31SPACEA(1)
     A            FIELDC2       10        +5SPACEA(1)
     A
     A
```

RSLH115-3

Font Substitution

If the font identifier specified on the FONT parameter is not supported by the printer being used, printer data management selects a substitute font that is supported by the printer (if possible). A substitute font is always at the same pitch or a higher pitch to ensure that as much data as possible fits on the printed page. A complete list of fonts supported and the substitute font selected for each printer that supports the FONT parameter is shown in Table 5-5 on page 5-27.

If a substitute font cannot be used, spool redirection is done. An inquiry message is sent to the message queue associated with the device or printer writer. The inquiry message gives you the option of holding or printing the file. If the print option is used, then the spooled file will be reformatted with the print attributes of file QPSPLPRT. Output may not look as intended. Refer to the printer reference manual for information on what conditions cause the substitute font to not be used.

For the SCS 3812 and 5219 Printers, font substitution can be made only at the file level. For font changes made in the document, a substitute font cannot be specified. In this case, spool redirection (described previously) is used.

IPDS Printer FONT Parameter Considerations

When FONT(*DEVD) is specified on the CRTPRTF, CHGPRTF, and OVRPRTF commands, the following limitations are imposed:

- Bar codes are positioned on the page assuming a 10-pitch font is specified in the device description.
- The data stream created may be longer than if a specific font were selected. Fields are positioned with spaces (hex 40) between them instead of using commands to specify the location where a field is to be placed.
- If a proportionally spaced font is used with a field in the file, any following fields may not be positioned in the column used if a specific font were specified at the file level. This is caused by the variable width of the characters in the proportionally spaced font, which are followed by spaces (hex 40) to position the next field.

In order to prevent the IPDS printers from reporting position check errors due to part of a character printing off of the top or bottom of the page, the system will make a slight adjustment downward if printing is done on the very first line of the page, and a slight adjustment upward if printing is done too close to the end of the form. This adjustment will be about 1/72nd of an inch. All other lines on the page will not have an adjustment made to them.

Fonts Supported for Each Printer

The following table lists the valid font identifiers, the characters-per-inch (CPI) value for each font style, a description of each font style, and if the font is supported on a particular printer. A value of Y indicates that the font is supported. A value of N indicates that the font is not supported, and the value in parentheses following the N indicates the substitute font ID used by the system. See "Font Substitution" on page 5-25 for more information on font substitution.

Please note the following characteristics in the *CPI* column:

- A number preceding a P indicates a typographic font describing the font height with a point size, for example, 8P. A point means 1/72nd of an inch. So, an 8-point font would be 1/9th of an inch high. A 24-point font would be 1/3rd of an inch high. Any value listed in parentheses directly beneath the P value is the approximate characters per inch. This is because each typographic font family has a different number of CPI.
- A number preceding a (PS) indicates a proportionally spaced font. The characters vary in width depending on the character being printed. For example, *i* is a narrow character, *w* is a wide character. The CPI value listed is the width of a blank character for that font.

Table 5-5 (Page 1 of 4). Printer Font Table (FONT Parameter)

			Can Be Downloaded and/or Is Resident		**Can Be Downloaded and Is Not Resident**	**Cannot Be Downloaded and Is Resident**		
Font ID	**CPI**	**Fonts**	**ASCII 3812 ASCII 3816**	**IPDS 3812 IPDS 3816**	**3820 3825 3827 3835**	**IPDS 4224[1]**	**IPDS 4234**	**5219[2]**
003	10	OCR-B[3]	Y	Y	Y	Y	Y	N(011)
005	10	Rhetoric/Orator	Y	Y	Y	N(011)	N(026)	Y
011	10	Courier	Y	Y	Y	Y	Y	Y
012	10	Prestige Pica	Y	Y	Y	N(011)	N(026)	Y
013	10	Artisan	Y	Y	N(011)	N(011)	N(011)	Y
018	10	Courier Italic	Y	Y	Y	N(011)[4]	N(026)	N(011)
019	10	OCR-A[3]	Y	Y	Y	Y	Y	N(011)
020	10	Pica	Y	Y	N(011)	N(011)	N(026)	Y
026	10	Gothic	Y	Y	N(011)	Y	Y	Y
030	10	Symbol	Y	Y	Y	N(011)	N(011)	N(011)
038	10	Orator Bold	Y	Y	Y	N(011)[4]	N(011)	N(011)
039	10	Gothic Bold	Y	Y	Y	N(026)[4]	N(026)	N(011)
040	10	Gothic Text	Y	Y	Y	N(026)	N(026)	N(011)
041	10	Roman Text	Y	Y	Y	N(011)	N(026)	N(011)
042	10	Serif Text	Y	Y	Y	N(011)	N(026)	N(011)
043	10	Serif Italic	Y	Y	Y	N(011)[4]	N(026)	N(011)
044	10	Katakana	Y	Y	Y	N(011)	N(011)	N(011)
046	10	Courier Bold	Y	Y	Y	N(011)[4]	N(026)	N(011)
051	10	Gothic	Y	Y	N(011)	N(026)	Y	N(026)
052	10	Courier	Y	Y	N(011)	N(011)	Y	N(011)
066	12	Gothic Text	Y	Y	Y	N(087)	N(087)	N(087)
068	12	Gothic Italic	Y	Y	Y	N(087)[4]	N(087)	N(087)

Table 5-5 (Page 2 of 4). Printer Font Table (FONT Parameter)

			Can Be Downloaded and/or Is Resident		Can Be Downloaded and Is Not Resident	Cannot Be Downloaded and Is Resident		
Font ID	**CPI**	**Fonts**	**ASCII 3812 ASCII 3816**	**IPDS 3812 IPDS 3816**	**3820 3825 3827 3835**	**IPDS 4224[1]**	**IPDS 4234**	**5219[2]**
069	12	Gothic Bold	Y	Y	Y	N(087)	N(087)	N(087)
070	12	Serif Text	Y	Y	Y	N(087)	N(087)	N(087)
071	12	Serif Italic	Y	Y	Y	N(087)[4]	N(087)	N(087)
072	12	Serif Bold	Y	Y	Y	N(087)[4]	N(087)	N(087)
074	12	Gothic	N(087)	N(087)	N(085)	N(087)	Y	N(087)
075	12	Courier	N(085)	N(085)	N(085)	N(085)	Y	Y
080	12	Symbol Diplomat Script	Y	Y	N(085)	N(087)	N(087)	Y
084	12	Script	Y	Y	Y	N(087)	N(087)	Y
085	12	Courier	Y	Y	Y	Y	Y	Y
086	12	Prestige Elite	Y	Y	Y	N(087)	N(087)	Y
087	12	Letter Gothic	Y	Y	Y	Y	Y	Y
091	12	Light Italic	Y	Y	N(085)	N(087)[4]	N(087)	Y
110	12	Letter Gothic Bold	Y	Y	Y	N(087)[4]	N(087)	N(087)
111	12	Prestige Elite Bold	Y	Y	Y	N(087)[4]	N(087)	N(086)
112	12	Prestige Elite Italic	Y	Y	Y	N(087)[4]	N(087)	N(086)
154	12 (PS)	Essay	N(160)	N(160)	N(160)	N(085)	Y	N(085)
155	12 (PS)	Boldface Italic Mixed	Y	Y	Y	N(160)[4]	N(160)	N(160)
158	12 (PS)	Modern Mixed	Y	Y	N(160)	N(160)	N(160)	Y
159	12 (PS)	Boldface Mixed	Y	Y	Y	N(160)[4]	N(160)	Y
160	12 (PS)	Essay Mixed	Y	Y	Y	Y	Y	Y
162	12 (PS)	Essay Italic Mixed	Y	Y	Y	N(160)[4]	N(160)	Y
163	12 (PS)	Essay Bold Mixed	Y	Y	Y	N(160)	N(160)	N(160)
173	12 (PS)	Essay Light Mixed	Y	Y	Y	N(160)	N(160)	N(160)
174	12 (PS)	Gothic Mixed Pitch	N(160)	N(160)	Y	N(160)	N(160)	N(160)
175	12 (PS)	Document Mixed	Y	Y	Y	N(160)	N(160)	N(160)
204	13.3	Gothic Text	Y	Y	N(223)	N(222)	Y	N(222)
205	13.3	Gothic	N(204)	N(204)	N(223)	N(222)	Y	N(222)
221	15	Prestige	Y	Y	N(223)	N(222)	N(222)	Y
222	15	Gothic	N(230)	N(230)	N(223)	Y	Y	Y
223	15	Courier	Y	Y	Y	Y	Y	Y
225	15	Symbol Diplomat Script	Y	Y	N(223)	N(222)	N(222)	Y
229	15	Serif Text	Y	Y	Y	N(222)	N(222)	N(222)

Table 5-5 (Page 3 of 4). Printer Font Table (FONT Parameter)

			Can Be Downloaded and/or Is Resident		Can Be Downloaded and Is Not Resident	Cannot Be Downloaded and Is Resident		
Font ID	**CPI**	**Fonts**	**ASCII 3812 ASCII 3816**	**IPDS 3812 IPDS 3816**	**3820 3825 3827 3835**	**IPDS 4224[1]**	**IPDS 4234**	**5219[2]**
230	15	Gothic Text	Y	Y	Y	N(222)	N(222)	N(222)
232	15	Gothic	N(230)	N(230)	Y	N(222)	Y	N(222)
233	15	Courier	N(230)	N(230)	Y	N(223)	Y	N(223)
244	5	Courier Ultra Expanded	Y	Y	N(011)	N(011)	N(026)	N(011)
245	5	Courier Ultra Expanded Bold[5]	Y	Y	N(011)	N(011)[4]	N(026)	N(011)
252	17	Courier Ultra Condensed	Y	Y	N(223)	N(223)	N(400)	N(223)
253	17	Courier Ultra Condensed Bold	Y	Y	N(223)	N(223)	N(400)	N(223)
254	17	Courier (sub/superscript)	Y	Y	N(223)	N(223)	N(400)	N(223)
258	18	Gothic	N(252)	N(252)	Y	N(222)	Y	N(222)
259	18	Gothic	N(252)	N(252)	N(222)	N(222)	Y	N(222)
281	20	Gothic Text	Y	Y	Y	N(222)	N(400)	N(222)
290	27	Gothic Text	Y	Y	Y	N(222)	N(400)	N(222)
300	16.7	Gothic	N(252)	N(252)	N(290)	N(400)	Y	N(222)
400	16.7	Gothic	N(252)	N(252)	N(290)	Y	Y	N(222)
751	8P (27)	Sonoran Serif Roman Medium	Y	N(281)	Y	N(222)	N(258)	N(222)
1051	10P (21.8)	Sonoran Serif Roman Medium	Y	N(252)	Y	N(222)	N(222)	N(222)
1053	10P (21.8)	Sonoran Serif Roman Bold	Y	N(253)	Y	N(222)	N(222)	N(222)
1056	10P (21.8)	Sonoran Serif Italic Medium	Y	N(252)	Y	N(222)	N(222)	N(222)
1351	12P (17.1)	Sonoran Serif Roman Medium	Y	N(252)	Y	N(222)	N(222)	N(222)
1653	16P (13.3)	Sonoran Serif Roman Bold	Y	Y	Y	N(011)	N(011)	N(011)

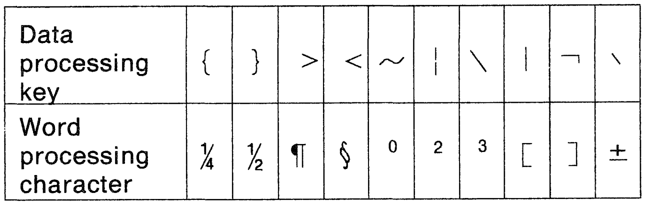
Table 5-5 (Page 4 of 4). Printer Font Table (FONT Parameter)

			Can Be Downloaded and/or Is Resident		**Can Be Downloaded and Is Not Resident**	**Cannot Be Downloaded and Is Resident**		
Font ID	**CPI**	**Fonts**	**ASCII 3812 ASCII 3816**	**IPDS 3812 IPDS 3816**	**3820 3825 3827 3835**	**IPDS 4224[1]**	**IPDS 4234**	**5219[2]**
2103	24P (10)	Sonoran Serif Roman Bold	Y	Y	Y	N(011)	N(011)	N(011)

[1] For the 4224 Printer, the fonts supported are not available for all print qualities. The OCR-A and OCR-B fonts are only supported with PRTQLTY(*NLQ). The Courier and Essay fonts are available only with PRTQLTY(*NLQ) and PRTQLTY(*STD). The Gothic font is only available with PRTQLTY(*DRAFT). If there is a mismatch between the print quality and the font selected, the font changes to match the print quality.

[2] For the 5219 Printer, each font style identifier has an associated printwheel cartridge. The description on the printwheel cartridge contains the font style identifier, the font style name, and the pitch indicator. When a new font is specified (one that differs from the previously-used font), the printer indicates Not Ready, the Change Font light is lit, and a ***Device Not Ready*** message is sent to the message queue associated with the printer. The light display indicates the hexadecimal printwheel value to be on the device. Refer to the printer device operator's guide for more information on changing printwheels.

All fonts available on the 5219 Printer are supported on the FONT parameter. However, only the Courier, Artisan, and Letter Gothic are data processing printwheels. The characters on these printwheels match the data processing keyboards on the IBM AS/400 work station displays (5251, 5252, 5291, and 5292). The rest of the 5219 printwheels are multinational word processing printwheels (code page 256) whose graphic content is not identical to the full set of characters available on the IBM AS/400 work station display keyboards. If you press the following special character keys on the work station display keyboard, the special characters from the multinational word processing printwheel appear:



Data processing key	{	}	>	<	~	¦	\	|	¬	`
Word processing character	¼	½	¶	§	^{0}	^{2}	^{3}	[	]	±

RSLH594-0

[3] OCR-A and OCR-B have special code page requirements. See "Using Alternative Character Sets and Code Pages for Printer Output" on page 5-32 to find the appropriate code page.

[4] The 4224 Printer can simulate certain italic and bold fonts. For example, the 4224 Printer does not support font identifier 018 (Courier Italic), but the font is simulated by italicizing font identifier 011 (Courier). Font identifier 046 (Courier Bold) is simulated by printing font identifier 011 (Courier) in bold print.

[5] The Courier Ultra Expanded Bold (font identifier 245), which has a pitch of 5 characters per inch, is imitated on the 4224 Printer by printing the Courier font (font identifier 011) with characters expanded to double width.

Replacing Unprintable Characters

You can replace unprintable characters in your data before the data is written to the printer or to a spooled file by specifying RPLUNPRT(*YES) on the CRTPRTF, CHGPRTF, or OVRPRTF commands. The replacement of an unprintable character depends on the hexadecimal value of the unprintable character and the printer type specified for the printer file.

The RPLUNPRT value must be selected before the spooled file is created. Once a spooled file is in an output queue, changing the RPLUNPRT value has no effect on that particular spooled file.

- When RPLUNPRT(*YES) is specified, any characters in the range hex 00 through hex 3F, and hex FF, are replaced with a blank. Any characters in the range hex 40 through hex FE that the printer cannot print will be replaced with the replacement character specified by the user. The default replacement character is a blank. The character which a printer cannot print varies depending on the type of printer.
- When RPLUNPRT(*NO) is specified, no translation of the data stream is made. Any characters in the range hex 00 through hex 3F, or hex FF, may cause undesirable results. These characters are in the range used by printer control characters.

  For most characters in this range, the printer signals an unrecoverable error and the file will either be held in the output queue or will not be processed. Some characters in this range control forms action and character representation on the printer and, as a result, additional skipping or spacing may occur. If control characters are placed in the data, system functions such as displaying or copying spooled files and restarting or backing up a printer writer may produce results that cannot be predicted.

  If the hexadecimal value of the unprintable character is hex 40 through hex FE, a message is sent to the message queue associated with the device. The message gives you the option to end the writer, hold the file (spooled output only), ignore the error and continue printing, or select a page number where printing should be restarted. If the ignore option is taken, then unprintable characters continue to be reported. If the option to start again (specify page number) is taken, all unprintable characters are replaced with blanks and you will receive no more notification for unprintable characters.

Considerations for the 5262 and 4245 Printers

The printer translates lowercase characters to uppercase characters when using a print band which does not contain lowercase characters. If your print job contains other characters which are not on the print band, they can be translated to blanks by specifying RPLUNPRT(*YES) for the printer file.

A print band is selected by switches on the 5262 operator's panel. The operator must select both a language ID and a band image using these switches.

The 4245 Printer detects the print band the printer is using.

When a print band changes for a print job, no inquiry message is sent to the message queue associated with the printer writer. You can specify a different forms type for that job. A message to change the form type is sent to the printer writer message queue that you can use to notify the operator of the change to the print band.

Using Alternative Character Sets and Code Pages for Printer Output

In multinational environments, data in one national graphics character set may need to be printed on devices that support another national character set. This is particularly true of characters with accents and other characters with diacritical marks (such as ç, ñ, and ü). In this section, these characters are called *extended alphabetics.*

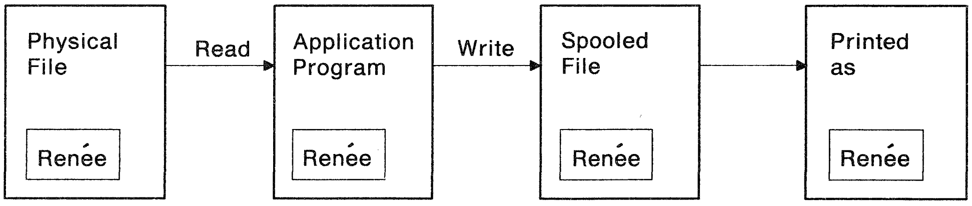
For example, assume that a physical file on the system contains data in the Basic French character set, and includes the character é. In the Basic French character set, this character is hex C0. The data could have been entered on a display device that can handle the character or could have been sent to the system from another system over a communications line. When hex C0 is sent to a printer that is set up for the United States Basic character set, the hex C0 is printed as {. Depending on the printer and the hexadecimal value sent, the hexadecimal value could be an unprintable character. The way the printer handles a specific hex code point (for example, hex C0) depends on the current value of the CHRID parameter in the printer file. You can specify three parameter values for the CHRID parameter:

- With an explicit value specified for the CHRID parameter, the printer will interpret the data as if the data were in the character set and code page specified.
- With CHRID(*SYSVAL) specified, the printer file takes the value specified in the QCHRID system value when the output is created.
- With CHRID(*DEVD) specified, the printer uses the CHRID that was set with the device control panel or that was specified when the printer device description was created.

Not all printers can handle all CHRID parameter values. If a CHRID is specified for a printer on which that CHRID is not supported, a message will be sent to the operator. See Table 5-6 on page 5-34 for a description of which printers support which extensions.

For program-described printer files, the value of the CHRID parameter determines the code page and character set used to print the data. However, for externally described printer files, the CHRID parameter is used only for fields that also have the CHRID DDS keyword specified. Fields that do not have the CHRID DDS keyword use the code page and character as if CHRID(*DEVD) had been specified for the CHRID parameter on the printer file.

The following shows how extended alphabetics are handled in printer output:



RSLH174-1

Assume that a record in a physical file contains a field with the value Renée. An application program reads the record from the physical file, and writes a record containing the data to the spooled file. The output field in the printer file that describes how Renée is to be printed has the CHRID DDS keyword specified, indicating that the printer is to interpret extended alphabetics. (The graphic character set 288 and code page 297 are specified for the interpretation in either the printer file or the QCHRID system value. Code page 297 is used for French language.)

When printing the data, the printer interprets the hex C0 as specified in character set 288 and code page 297. If character set 101 and code page 037 had been selected, hex C0 (é) would have been printed as {.

The following CHRID values (graphic character set and code page) must be specified to print fonts OCR-A and OCR-B on the 4224 and 3812 Printers:

580 340
590 340
647 892
697 893

Character Identifier (CHRID) Values Supported for Each Printer

Table 5-6 (Page 1 of 3). Character Identifier Values and Applicable Printers (CHRID Parameter)

	Code Pages		**Printers[1]**					
Language Groups	**CHRID Code Page xxx yyy[2,3]**	**Sub-stitute Code Page yyy[2,4]**	**3812[5] 3816[5] 3820 3825 3827 3835**	**4214[5]**	**4224[5]**	**4234[5]**	**5219**	**5224 5225**
Major Groups								
International (and US ASCII)	103 038	500	Yes	n/a	n/a	n/a	Yes	n/a
Multinational	697 500		Yes	Yes	Yes	Yes	n/a	n/a
	337 256	500	Yes	n/a	n/a	n/a	n/a	Yes
	697 256	500	Yes	n/a	Yes	n/a	n/a	Yes
United States	101 037		Yes	Yes	Yes	Yes	Yes	Yes
	697 037		Yes	Yes	n/a	Yes	n/a	n/a
Individual Languages/Countries								
Arabic	697 361		Yes	n/a	n/a	n/a	n/a	n/a
Arabic X/B	235 420		Yes	n/a	Yes	n/a	n/a	n/a
Austria/Germany[6]	265 273		Yes	Yes	Yes	Yes	Yes	Yes
	697 273		Yes	Yes	Yes	Yes	n/a	n/a
Austria/Germany	697 286	273	Yes	n/a	Yes	n/a	n/a	n/a
Belgium[6]	269 274		Yes	Yes	Yes	Yes	Yes	Yes
	697 274		Yes	Yes	Yes	Yes	n/a	n/a
Brazil[6]	273 275		Yes	Yes	Yes	Yes	Yes	Yes
	697 275		Yes	Yes	Yes	Yes	n/a	n/a
Canadian French[6]	277 276	297	Yes	n/a	n/a	n/a	Yes	Yes
	341 260	037	Yes	n/a	Yes	n/a	n/a	n/a
	697 260		Yes	n/a	n/a	n/a	n/a	n/a
Canada-Bilingual	038 256		Yes	n/a	n/a	n/a	Yes	n/a
	039 256		Yes	n/a	n/a	n/a	Yes	n/a
Canada-English	037 256		Yes	n/a	n/a	n/a	Yes	n/a
Cyrillic	960 880		n/a	n/a	Yes	n/a	n/a	n/a
Czechoslovakia/Czech	083 257		n/a	n/a	n/a	n/a	Yes	n/a
Czechoslovakia/Slovak	085 257		n/a	n/a	n/a	n/a	Yes	n/a
Denmark/Norway[6]	281 277		Yes	Yes	Yes	Yes	Yes	Yes
	697 277		Yes	Yes	Yes	Yes	n/a	n/a
Denmark/Norway	697 287	277	Yes	n/a	Yes	n/a	n/a	n/a
Finland/Sweden[6]	285 278		Yes	Yes	Yes	Yes	Yes	Yes
	697 278		Yes	Yes	Yes	Yes	n/a	n/a
Finland/Sweden	697 288	278	Yes	n/a	Yes	n/a	n/a	n/a
France (1977)[6]	289 279	297	Yes	n/a	n/a	n/a	n/a	Yes
France (1980)[6]	288 297		Yes	Yes	Yes	Yes	Yes	n/a
	697 297		Yes	Yes	Yes	Yes	n/a	n/a
France	251 256		Yes	n/a	n/a	n/a	Yes	n/a
France/Belgium	031 256		Yes	n/a	n/a	n/a	Yes	n/a
Germany/Austria	028 256		Yes	n/a	n/a	n/a	Yes	n/a
	029 256		Yes	n/a	n/a	n/a	Yes	n/a
Greek	218 423		n/a	n/a	Yes	n/a	n/a	n/a
Hebrew	941 424		Yes	n/a	Yes	n/a	n/a	n/a
	697 424		Yes	n/a	n/a	n/a	n/a	n/a

Table 5-6 (Page 2 of 3). Character Identifier Values and Applicable Printers (CHRID Parameter)

	Code Pages		Printers[1]					
Language Groups	CHRID Code Page xxx yyy[2,3]	Sub-stitute Code Page yyy[2,4]	3812[5] 3816[5] 3820 3825 3827 3835	4214[5]	4224[5]	4234[5]	5219	5224 5225
Hong Kong	119 256		Yes	n/a	n/a	n/a	Yes	n/a
Hungary	091 257		n/a	n/a	n/a	n/a	Yes	n/a
Icelandic	697 871		Yes	n/a	Yes	n/a	n/a	n/a
	697 029		Yes	n/a	n/a	n/a	n/a	n/a
Italy[6]	293 280		Yes	Yes	Yes	Yes	Yes	Yes
	697 280		Yes	Yes	Yes	Yes	n/a	n/a
Italy	041 256		Yes	n/a	n/a	n/a	Yes	n/a
Japan-English[6]	297 281		Yes	Yes	Yes	Yes	Yes	Yes
	697 281		Yes	n/a	Yes	Yes	n/a	n/a
	068 256		Yes	n/a	n/a	n/a	Yes	n/a
	069 256		Yes	n/a	n/a	n/a	Yes	n/a
Japan-Katakana[6]	332 290		Yes	n/a	Yes	Yes	n/a	Yes
	697 290		Yes	n/a	n/a	n/a	n/a	n/a
Latin	959 870		n/a	n/a	Yes	n/a	n/a	n/a
Latin America/Puerto Rico	025 256		Yes	n/a	n/a	n/a	Yes	n/a
Netherlands	043 256		Yes	n/a	n/a	n/a	Yes	n/a
Norway/Denmark	055 256		Yes	n/a	n/a	n/a	Yes	n/a
Poland	093 257		n/a	n/a	n/a	n/a	Yes	n/a
Portugal[6]	301 282		Yes	Yes	Yes	Yes	Yes	Yes
	697 282		Yes	Yes	Yes	Yes	n/a	n/a
Portugal	697 831	282	Yes	n/a	Yes	n/a	n/a	n/a
	063 256		Yes	n/a	n/a	n/a	Yes	n/a
Romania	087 258		n/a	n/a	n/a	n/a	Yes	n/a
South Africa	081 258		n/a	n/a	n/a	n/a	Yes	n/a
Spain[6]	305 283	284	Yes	n/a	Yes	Yes	Yes	Yes
	697 283	284	Yes	n/a	Yes	n/a	n/a	n/a
	697 289	284	Yes	n/a	Yes	n/a	n/a	n/a
	045 256		Yes	n/a	n/a	n/a	Yes	n/a
Spanish Speaking[6]	309 284		Yes	Yes	Yes	Yes	Yes	Yes
	697 284		Yes	Yes	Yes	Yes	n/a	n/a
Sweden/Finland	052 256		Yes	n/a	n/a	n/a	Yes	n/a
	053 256		Yes	n/a	n/a	n/a	Yes	n/a
Switzerland/French	048 256		Yes	n/a	n/a	n/a	Yes	n/a
Switzerland/German	049 256		Yes	n/a	n/a	n/a	Yes	n/a
Thai	1102 889		n/a	n/a	Yes	n/a	n/a	n/a
Turkish	965 905		n/a	n/a	Yes	n/a	n/a	n/a
United Kingdom[6]	313 285		Yes	Yes	Yes	Yes	Yes	Yes
	697 285		Yes	Yes	Yes	Yes	n/a	n/a
U.K./Israel	066 256		Yes	n/a	n/a	n/a	Yes	n/a
U.K./Israel-Latin	067 256		Yes	n/a	n/a	n/a	Yes	n/a
USA-Accounting	017 256		Yes	n/a	n/a	n/a	Yes	n/a
USA/Australia	001 256		Yes	n/a	n/a	n/a	Yes	n/a

Table 5-6 (Page 3 of 3). Character Identifier Values and Applicable Printers (CHRID Parameter)

	Code Pages		Printers[1]					
Language Groups	**CHRID Code Page xxx yyy[2,3]**	**Sub-stitute Code Page yyy[2,4]**	**3812[5] 3816[5] 3820 3825 3827 3835**	**4214[5]**	**4224[5]**	**4234[5]**	**5219**	**5224 5225**
Yugoslavia	410 890		n/a	n/a	Yes	n/a	n/a	n/a
Yugoslavia-Latin	095 257		n/a	n/a	n/a	n/a	Yes	n/a
Noncountry Languages								
APL	697 293		Yes	n/a	n/a	n/a	n/a	n/a
APL Alternate	697 310		Yes	n/a	n/a	n/a	n/a	n/a
ASCII	103 256		Yes	n/a	n/a	n/a	Yes	n/a
EBCDIC	101 256		Yes	n/a	n/a	n/a	Yes	n/a
International Typographic	697 361		Yes	n/a	n/a	n/a	n/a	n/a
OCR (unregistered)	697 340		Yes	n/a	Yes	n/a	n/a	n/a
OCR A	697 892		Yes	n/a	n/a	n/a	n/a	n/a
OCR A (unregistered)	580 340		Yes	n/a	Yes	n/a	n/a	n/a
OCR B	697 893		Yes	n/a	n/a	n/a	n/a	n/a
OCR B (unregistered)	590 340		Yes	n/a	Yes	n/a	n/a	n/a
Personal Computer	697 361		Yes	n/a	n/a	n/a	n/a	n/a
Symbol-Selectric	201 259		Yes	n/a	n/a	n/a	Yes	n/a
Symbol-6640	202 259		Yes	n/a	n/a	n/a	Yes	n/a
Symbol-6670	203 259		Yes	n/a	n/a	n/a	Yes	n/a
Symbols Set 7	697 259		Yes	n/a	n/a	n/a	n/a	n/a

[1] The 5256, 5262, and 4245 (Models T12 and T20) work station printers do not support the hardware function required for alternative CHRID processing. If a nondefault character set and code page is selected for these printers, a diagnostic message is sent and processing continues using the default character set.

[2] If the printer supports the code page specified (the second part *(yyy)* of the CHRID parameter) but not the character set *(xxx)*, then the character set supported by the printer is used along with the specified code page. For example, if 337 037 (extended character set for displays) is specified for the 5224 and 5225 Printers, the print file is printed with character set 101, code page 037.

[3] In some cases, the printer will substitute a supported code page for an unsupported code page. Consult the various printer reference guides for defaults on the code page mapping.

[4] If the printer does not support or map the code page specified, an attempt is made by the system to find a satisfactory substitute. This column shows the code page substitutes that are made if the specified printer supports the substitute.

[5] The 3812, 3816, 4214, 4224, and 4234 Printers support character set 697 (full character set). This character set contains all the characters in the limited character sets. For example, 697 037 would contain all the characters in 101 037 or 337 037 (extended character set for displays).

[6] Indicates that this language is considered a primary language group. All other entries, if any, under the primary language group, are considered as alternate language groups.

Print Text

The system allows users to specify on the printer file a line of text that is to be printed at the bottom of every page. This line of text is called the print text and is set by using the PRTTXT parameter on the CRTPRTF, CHGPRTF, or OVRPRTF commands. Up to 30 characters are allowed in the line of print text. The 30 characters are centered at the bottom of the page, 2 lines below the overflow line. If the user already has data to print on the line that the print text goes on, the print text is bumped down to the next blank line on the page. If no lines are blank, the print text is printed on the last line of the page.

A system value, QPRTTXT, can be used to specify the print text so that the same text can appear on all files printed on the system. Also, the print text can be taken from the job description, so that all files created from a particular job can have the same print text.

Print text is useful for printing a security classification on each page. It can also be used to print a company name or slogan on each page.

Editing Output Fields

The system provides editing support that makes fields more readable when they are printed. With the system editing support, you can do the following:

- Suppress leading zeros
- Punctuate a field with commas and periods to show decimal column and to group digits by threes
- Print negative values with a minus sign or CR to the right
- Print zero values as zeros or blanks
- Print asterisks to the left of significant digits to provide asterisk protection
- Print a currency symbol corresponding to the system value QCURSYM

The system provides this editing support with edit codes and edit words. Edit codes are a defined set of editing patterns. You identify these by name, and the system edits a field according to the pattern defined by the named edit code. Edit words are edit patterns that you define to produce the desired results. Edit codes cover most commonly used editing requirements. You need to use the edit word support only for those editing needs not covered by edit codes.

There are two methods of using edit codes and edit words. Which one you use depends on how you define the printer file and how it is used in an application program. If your application is using program-described data, your high-level language may allow you to identify edit codes or create your own edit words. If your application is using externally described data, the edit code (EDTCDE) DDS keyword allows you to identify an edit code; the edit word (EDTWRD) DDS keyword allows you to define your own editing pattern.

| The system provides several edit codes:

- 1 through 4
- A through D
- J through M
- X through Z

The editing patterns defined by these codes are described in Appendix E, "Edit Codes."

User-Defined Edit Codes

You can also define five edit codes to provide more editing function than is available with the OS/400 edit codes, and to handle common editing functions that would otherwise require the use of an edit word. These are called user-defined edit codes. For example, you may need to edit numbers that include hyphens (like some telephone numbers), or more than one decimal point. You can use user-defined edit codes for these functions. These edit codes are named QEDIT5, QEDIT6, QEDIT7, QEDIT8, and QEDIT9 and can be referred to in DDS or a high-level language program by number (5, 6, 7, 8, or 9).

These edit codes are created by using the Create Edit Description (CRTEDTD) command. Edit descriptions are always placed in library QSYS. They cannot be moved or renamed; only one occurrence of each is allowed. Edit descriptions have an object type of *EDTD.

IBM supplies a version of each of the QEDITX edit codes. You can use these edit descriptions as they are, or you can delete them and create your own. See Appendix E, "Edit Codes" for more information about using these edit descriptions.

Before using any of the user-defined edit codes, you should check its contents on your system, since it may have been changed from the IBM-supplied version. The Display Edit Description (DSPEDTD) command can be used to display the contents of a user-defined edit code.

Changing a user-defined edit code description does not affect any application or printer file that has already been created using that edit description. If you want your application to use the changed edit description, you must either create the high-level language program again (if the edit code is used in the program) or create the file again (if the application is using an externally described file that contains EDTCDE keywords).

Effect of Changing Fields in a File Description

When a program using externally described printer files is compiled, the compiler extracts the file descriptions for the files referred to in the program and makes these file descriptions part of the compiled program. When you run the program, you can verify that the record formats with which the program was compiled are the current record formats. To do this, you use the LVLCHK parameter on the create file command when the file is created.

The system assigns a unique level identifier for each record format when the file it is associated with is created. The system uses the information in the record format description to determine the level identifier. This information includes the name of the record format, the names, attributes, and order of the fields in the format, the indicators used, and the names and the order of the indicators in the record format. If you use the INDARA keyword to remove the indicator from the output buffer, the indicators used are not included in the level identifier information.

When the file is opened, if level checking is specified (LVLCHK parameter), the system does a format-by-format comparison of the level-checking values specified in the program to the level-checking values specified in the printer file. If any of the formats specified in the program do not exist in the file, or if any of the level checking values are different, an error occurs. Formats can be added to or removed from a printer file without affecting existing application programs that do not use the added or deleted formats.

You should display the file description to determine if the changes affect your program. You can use the Display File Field Description (DSPFFD) command to display the file description or, if you have the source entry utility (SEU), you can display the source file. Not every change in a file necessarily affects your program. You may not have to recompile your program. If you do not have to recompile your program, you should specify LVLCHK(*NO) for the file (CHGPRTF or OVRPRTF command).

You can add a field to the end of a printer file record format without having to recompile your program as long as you do not want to use the field in your program. If you delete a field from the end of the record format, you do not have to recompile your program if you are not using the field. However, if you add a field to or delete a field from a record format anywhere other than at the end, you must recompile your program. Otherwise, the field offsets in the record passed to and from the program are wrong for processing.

In general, anything that changes the length or position of any fields in the record format used by the program will require that the program be recompiled.

Printer File Redirection

Spooled or nonspooled output intended for a printer can be redirected to another printer. However, each file is checked to ensure that the file attributes (device type, number of lines per inch, number of characters per inch, page length, and page width) and any advanced functions used by the file (such as variable LPI, variable font, or defined characters) are valid on the new printer.

Nonspooled Output

When a nonspooled file is redirected, and the printer file attributes do not match the new printer, one of the following occurs:

- If the printer file specifies a characters-per-inch value not supported by the device being used, a diagnostic message (CPF4057) is sent to the program message queue and the data is printed at 10 characters per inch. If the page width is greater than 132 characters, the records are folded.

  Note: Folding is not supported on IPDS printers.

- If the printer file specifies a lines-per-inch value not supported by the device being used, a diagnostic message (CPF4056) is sent to the program message queue, the data is printed at 8 lines per inch.

- If the page length is greater than the maximum length allowed for the printer being used, the printing ends with an escape message (CPF4138).

- If the printer file specifies special device requirements (such as use of certain DDS keywords) that are not supported by the device being used, then a diagnostic message is sent to the program message queue and the special function is ignored.

Spooled Output

When a spooled file is redirected to another printer, the spooled file cannot be printed without change if any of the following are true:

- The spooled file attributes are not supported by the printer.
- The spooled file has special device requirements not supported by the printer.
- The spooled file was created on another system and specifies page sizes, output drawers, print qualities, lines per inch, characters per inch, character

identifiers, or justification values in the spooled file that are not supported by the printer.

Redirecting Spooled Files to SCS Printers

The following section describes the actions taken when a spooled file is redirected to an SCS printer and cannot be printed without change (SCS printers include the 3812, 3816, 4214, 4234, 4245, 5219, 5224, 5225, 5256, and 5262 Printers):

- If the spooled file uses the IPDS data stream (DEVTYPE(*IPDS)), the spooled file attributes are not supported by the printer, or the special device requirements used by the spooled file are not supported by the printer, an inquiry message is sent to the message queue of the writer with the options to:

  - End the writer
  - Print the spooled file with lines folded when the lines are longer than the width of IBM-supplied printer file QPSPLPRT
  - Print the spooled file with lines truncated when the lines are longer than the width of IBM-supplied printer file QPSPLPRT
  - Hold the spooled file and process the next file on the output queue

  If the spooled file is printed, results may be unpredictable because the file is printed using the printer attributes specified in the IBM-supplied printer file QPSPLPRT, and all advanced functions used by the spooled file are removed. Functions removed include:

  DDS keywords:

BARCODE	Bar codes
CHRID	Graphic character set and code page
CHRSIZ	Character size (width and height)
COLOR	Color printing
CPI	Characters per inch
DFNCHR	Define character
DRAWER	Paper drawer selection
FONT	Font selection
LPI	Lines per inch
PAGRTT	Page rotation
PRTQLTY	Print quality
TRNSPY	Transparency

  Other print functions:

  Drawer change in document
  Font change in document
  Lines-per-inch change in document
  Page rotation in document
  Subscript and superscript

- If the spooled file specifies a characters-per-inch value not supported by the printer, an inquiry message is sent to the message queue of the writer with the option to:

  - End the writer
  - Print the spooled file at 10 characters per inch with lines folded when the lines are longer than the width of IBM-supplied printer file QPSPLPRT
  - Hold the spooled file and process the next file on the output queue

- The 5219 Printer is an exception to the above cases if the only mismatch between the spooled file and printer is the highlight special device requirement that the file contains. When this occurs, an inquiry message is sent to the message queue of the writer with the option to:

  – End the writer.
  – Print the spooled file without highlighting but keep all other advanced functions used by the file.
  – Attempt to print the spooled file without changing. (If this is not successful, the file will be held on the output queue.)
  – Hold the spooled file and process the next file on the output queue.

  If the spooled file is printed, the resulting output closely resembles how the file was intended to look. This is because the attributes specified by the spooled file were used and advanced functions were kept.

- Documents created on other systems may contain print controls that are not supported by the 5219 or 3812 Printers. These controls may include variable form size, output drawer, print quality, lines per inch, characters per inch, character identifier, or justification. If this occurs, an inquiry message is sent to the message queue of the writer with the options to:

  – End the writer.
  – Print the spooled file with unsupported values changed to values which are supported by the printer.
  – Attempt to print the spooled file without changing. (If this is not successful, the file will be held on the output queue).
  – Hold the spooled file and process the next file on the output queue.

  If the spooled file is printed, the file attributes from the spooled file are used and all advanced functions that are valid for the device are kept. The resulting output should closely resemble what the file was intended to look like, though it still may not print exactly as intended because of the unsupported values.

Redirecting Spooled Files to IPDS Printers

The following describes the actions taken when a spooled file is redirected to an IPDS printer (3812, 3816, 3820, 3825, 3827, 3835, 4224, or 4234) and cannot be printed without changing:

- If the spooled file uses the SCS data stream (DEVTYPE(*SCS)) and contains DBCS (double-byte character set) data or has a page length greater than that supported by the printer (for both SCS and IPDS files), an inquiry message is sent to the message queue of the writer with the option to:

  – End the writer
  – Print the spooled file with lines truncated when the lines are longer than the width of IBM-supplied printer file QPSPLPRT
  – Hold the spooled file and process the next file on the output queue

  If the spooled file is printed, results may be unpredictable because the file is printed using the printer attributes specified in the IBM-supplied printer file QPSPLPRT, and all advanced functions used by the spooled file are removed. Functions removed include:

DDS keywords:

BARCODE	Bar codes
CHRSIZ	Character size (width and height)
COLOR	Color printing
CPI	Characters per inch
DFNCHR	Define character
DRAWER	Paper drawer selection
FONT	Font selection
LPI	Lines per inch
PAGRTT	Page rotation
PRTQLTY	Print quality
TRNSPY	Transparency

Other print functions:

Drawer change in document
Font change in document
Lines-per-inch change in document
Page rotation in document
Subscript and superscript

- If the spooled file uses the SCS data stream (DEVTYPE(*SCS)), does not contain DBCS data, and uses special device requirements (graphics, defined characters, transparencies, variable font, or enhanced 3812 fonts) or uses a proportionally spaced font for the FONT parameter of the file, an inquiry message is sent to the message queue of the writer with the option to:

  - End the writer
  - Transform the spooled file to IPDS format and print
  - Hold the spooled file and process the next file on the output queue

  If the spooled file is printed, the attributes from the spooled file are used and most advanced functions are kept. Advanced functions not kept are graphics, defined characters, justification, and transparencies. The transformation to IPDS format should substantially keep the integrity of the text data. However, the file may still not print exactly as intended because unsupported font pitches, font spacings, and character identifiers are changed to the closest approximation valid on the printer.

- If the spooled file uses DEVTYPE(*SCS), does not contain DBCS data, does not use defined characters, graphics, transparency, variable fonts, or enhanced 3812 fonts, and does not use a proportionally spaced font for the FONT parameter of the printer file, the writer automatically transforms the file to an IPDS file and prints it. The spooled file may not print exactly as intended because unsupported font pitches, font spacings, and character identifiers are changed to the closest approximation available on the printer.

- If the spooled file uses the IPDS data stream (DEVTYPE(*IPDS)) but uses advanced functions not supported by the printer, an inquiry message is sent to the message queue of the writer with the option to:

  - End the writer
  - Print the spooled file with unsupported advanced functions dropped from the file
  - Hold the spooled file and process the next file on the output queue

Folding Records in a Printer File

The FOLD parameter on the CRTPRTF, CHGPRTF, and OVRPRTF commands allows you to specify if all positions in a record are to be printed when the record length exceeds the page width (as specified by the PAGESIZE parameter). The FOLD parameter is not supported for DEVTYPE(*IPDS).

When DEVTYPE(*IPDS) is specified, or if you are creating a printer file for use with AS/400 Office, the FOLD parameter is ignored. If the data length is longer than the page width, the data is truncated by the printer and a message is sent to the message queue associated with the device. You may end the writer, hold the spooled file (spooled output only), or continue printing by typing a page to start again. If the option to start again is taken, you receive no further notification when the printer truncates data for the spooled file.

When you specify FOLD(*NO) for a program-described printer file, the data records are truncated at the page width. For an externally described printer file, a field that ends outside the page width is truncated at the page width, and fields that start outside the page width are not printed. The first time data is truncated, a message is sent. When you specify FOLD(*YES) for a program-described printer file, the data outside the page width to the end of the record is printed on the next line. For an externally described printer file, a field that ends outside the page width is folded to the next line. The fields in the record that start outside the page width are printed on a separate line. The first time a print record is folded, a message is sent to your program.

When a proportionally spaced or typographic font is used for SCS printers, folding may not produce the desired results. For these types of fonts, characters vary in width depending on the character being printed (for example, `i` is a narrow character, and `W` is a wide character). The system determines the number of characters in the page width based on the width of a blank for the specified font. Depending on the width of the characters in the line being folded, data may be truncated. These widths are listed in the implied CPI column of the font table listed in Table 5-5 on page 5-27.

Printer File Parameters

Table 5-7 on page 5-44 lists the parameters that apply to printers and where the parameters can be specified. Note that some values (for example, LPI(7.5)) and some parameters (for example, IGCDTA) are valid only on DBCS systems.

For specific information on how these parameters can be specified on the CRTPRTF and OVRPRTF commands, see the *CL Reference*.

Table 5-7 (Page 1 of 2). Printer Support Parameters

CL Parameter Name	Description	Specified on CRTPRTF Command	Specified on OVRPRTF Command	Specified in HLL Program
FILE	File	Qualified file name	*PRTF or file name	RPG/400, COBOL/400, PL/I, C/400, and BASIC
TOFILE	Overriding file		*FILE or qualified name	
SRCFILE	Source file	*NONE or qualified name		
SRCMBR	Source member	*FILE or qualified name		
GENLVL	Generation severity level	0, 10, 20, or 30		
OPTION	Type of output list	*SRC, *SOURCE, *NOSRC, *NOSOURCE, *LIST or *NOLIST, *SECLVL or *NOSECLVL		
DEV	Device name	*JOB, *SYSVAL, or device name	*JOB, *SYSVAL, or device name	
DEVTYPE	Device type	*SCS or *IPDS	*SCS or *IPDS	
PAGESIZE	Page length and width	Page length and width	Page length and width	RPG/400 and BASIC
LPI	Lines per inch	3, 4, 6, 7.5, 8, or 9	3, 4, 6, 7.5, 8, or 9	
CPI	Characters per inch	5, 10, 12, 13.3, 15, 16.7, 18 or 20	5, 10, 12, 13.3, 15, 16.7, 18 or 20	
OVRFLOW	Overflow line number	Line number	Line number	RPG/400 and BASIC
FOLD	Fold records	*NO, or *YES	*NO, or *YES	
RPLUNPRT	Replace unprintable characters	*YES and replacement character, or *NO	*YES and replacement character, or *NO	
ALIGN	Align forms	*NO, or *YES	*NO, or *YES	RPG/400
CTLCHAR	Control character	*NONE or *FCFC	*NONE or *FCFC	PL/I and C/400
CHLVAL	Channel value	*NORMAL or channel value and line number	*NORMAL or channel value and line number	
PRTQLTY	Print quality	*STD, *DRAFT, *NLQ, or *DEVD	*STD, *DRAFT, *NLQ, or *DEVD	
FORMFEED	Form feed mode	*DEVD, *CONT, *CUT, or *AUTOCUT	*DEVD, *CONT, *CUT, or *AUTOCUT	COBOL/400
DRAWER	Paper source	1, 2, 3 or *E1	1, 2, 3 or *E1	
FONT	Font identifier	*CPI, *DEVD, or font identifier	*CPI, *DEVD, or font identifier	
CHRID	Character set code page	*DEVD, *SYSVAL, or graphic character set and code page	*DEVD, *SYSVAL, or graphic character set and code page	
PAGRTT	Page rotate	*DEVD, *COR, 0, 90, 180, or 270	*DEVD, *COR, 0, 90, 180, or 270	
PRTTXT	Print text	*JOB, *SYSVAL, *BLANK, or print text	*JOB, *SYSVAL, *BLANK, or print text	
JUSTIFY	Justify	0, 50, or 100	0, 50, or 100	
DUPLEX	Output printed on one or both sides of the paper	*NO, *YES, or *TUMBLE	*NO, *YES, or *TUMBLE	
SPOOL	Spool data	*YES, or *NO	*YES, or *NO	
OUTQ	Output queue	*JOB, *DEV, or qualified name	*JOB, *DEV, or qualified name	
FORMTYPE	Form type	*STD or form type	*STD or form type	RPG/400

Table 5-7 (Page 2 of 2). Printer Support Parameters

CL Parameter Name	Description	Specified on CRTPRTF Command	Specified on OVRPRTF Command	Specified in HLL Program
COPIES	Number of copies	Number of copies	Number of copies	
MAXRCDS	Maximum records	*NOMAX, or maximum records	*NOMAX, or maximum records	
FILESEP	Separator pages	Number of file separators	Number of file separators	
SCHEDULE	Schedule	*FILEEND, *JOBEND, or *IMMED	*FILEEND, *JOBEND, or *IMMED	
HOLD	Hold	*NO or *YES	*NO or *YES	
SAVE	Save	*NO or *YES	*NO or *YES	
OUTPTY	Output priority	*JOB or output priority	*JOB or output priority	
USRDTA	User data	*SOURCE or user data	*SOURCE or user data	
IGCDTA	Double-byte data	*NO or *YES	*NO or *YES	
IGCEXNCHR	Double-byte extension characters	*YES, or *NO	*YES, or *NO	
IGCCHRRTT	Rotate double-byte characters	*NO, or *YES	*NO, or *YES	
IGCCPI	Double-byte characters per inch	*CPI, 5, 6, 10, or *CONDENSED	*CPI, 5, 6, 10, or *CONDENSED	
IGCSOSI	Shift control characters	*YES, *NO, or *RIGHT	*YES, *NO, or *RIGHT	
WAITFILE	File wait time	*IMMED, *CLS or number of seconds	*IMMED, *CLS or number of seconds	RPG/400 and PL/I
SHARE	Shared file	*NO or *YES	*NO or *YES	PL/I
LVLCHK	Level check	*YES, or *NO	*YES, or *NO	RPG/400, COBOL/400, PL/I, C/400, and BASIC
AUT	Authority	*CHANGE, *ALL, *USE, *EXCLUDE or authorization list name	*CHANGE, *ALL, *USE, *EXCLUDE or authorization list name	
TEXT	Text	*SRCMBRTXT, *BLANK, or text description		
	Record length			RPG/400, COBOL/400, PL/I, C/400, and BASIC
	End-of-page indicator			RPG/400, PL/I and BASIC
	Skip before	Line number		RPG/400
	Skip after	Line number		RPG/400
	Space before	Number of lines		RPG/400, COBOL/400
	Space after	Number of lines		RPG/400, COBOL/400

Performance Considerations

For externally described printer files, the fewer the number of fields in a record, the faster the processing of that record. Also, by putting several lines of text within a record instead of each line being a separate record, system overhead involved with the processing of each record is reduced.

When coding the DDS for externally described printer files, define the fields in sequential order. The output will not be changed if fields are not defined in sequential order, but the extra travel time of the printer head may be noticeable.

For externally described printer files, specify a specific font or FONT(*CPI) on the CRTPRTF, CHGPRTF, or OVRPRTF command instead of FONT(*DEVD). This helps keep the data stream as small as possible.

If a spooled file is intended to be printed on an IPDS printer, specify DEVTYPE(*IPDS) on the CRTPRTF, CHGPRTF, or OVRPRTF command to avoid the extra system processing required to transform the data stream from SCS to IPDS.

Chapter 6. Spool Support

Spooling functions help system users to manage input and output operations more efficiently. The system supports two types of spooling:

- *Output spooling* sends job output to disk storage, rather than directly to a printer or diskette output device. Output spooling allows the job producing the output to continue processing independently of the speed or availability of output devices.
- *Input spooling* accepts job input, stores the input data in disk storage to await processing, and allows the input device to be used independently of when the job is actually processed.

Output spooling may be used for both printer and diskette devices; input spooling applies to diskette and database file input.

This chapter discusses both output and input spooling, including advanced output spooling support, such as using multiple output queues and redirecting files. For more information about spooling support for printer and diskette devices, see Chapter 5, "Printer Support" and Chapter 8, "Diskette Support."

Output Spooling Overview

Output spooling allows the system to produce output on multiple output devices, such as printer and diskette devices, in an efficient manner. It does this by sending the output of a job destined for a printer or diskette to disk storage. This process breaks a potential job limitation imposed by the availability or speed of the output devices.

Spooling is especially important in a multiple-user environment where the number of jobs running often exceeds the number of available output devices. Using output spooling, the output can be easily redirected from one device to another.

The main elements of output spooling are:

Device description A description of the printer or diskette device

Spooled output file A file containing spooled output records that are to be produced on an output device

Output queue An ordered list of spooled output files

Writer A program that takes spooled output files from an output queue and produces them on an output device

Application program A high-level language program that creates a spooled output file using a device file with the spooling attribute specified

Device file A description of the format of the output, and a list of attributes that describe how the system should process the spooled output file

Figure 6-1 on page 6-2 shows the relationship of these spooling elements.

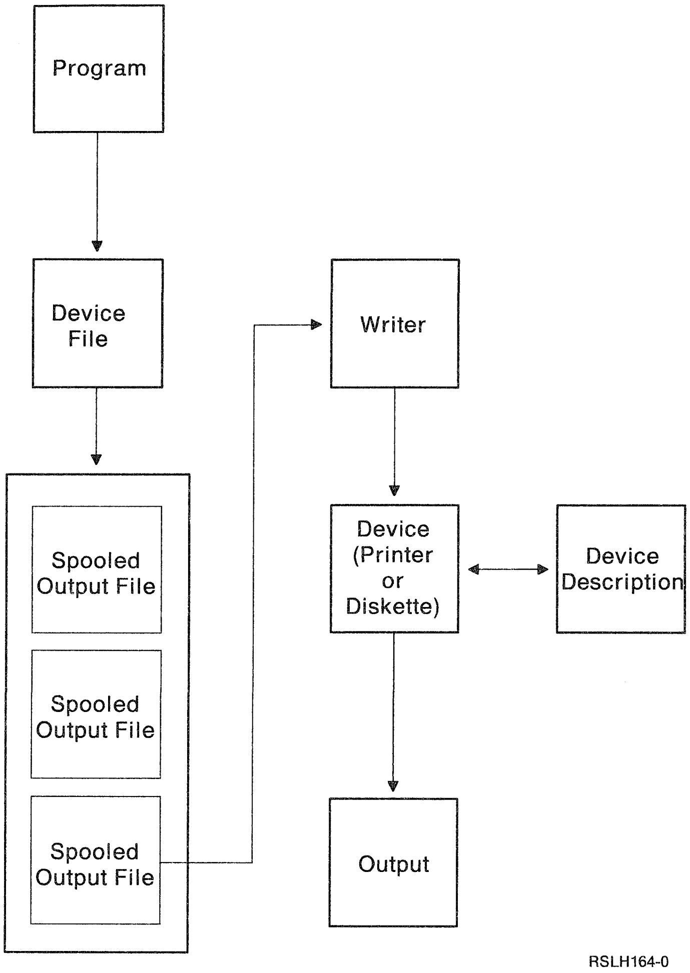


Figure 6-1. Relationship of Output Spooling Elements

Output spooling functions are performed by the system without requiring any special operations by the program that produces the output. When a device file is opened by a program, the operating system determines whether the output is to be spooled. When a printer or diskette file specifying spooling is opened, the spooled file containing the output of the program is placed on the appropriate output queue in the system.

A spooled file can be made available for printing when the printer file is opened, when the printer file is closed, or at the end of the job. A printer writer is started in the spooling subsystem to write the records to the printer. The spooled output is selected from an output queue. The same general description applies for spooled diskette files.

Device Descriptions

Device descriptions must be created for each printer and diskette device to define that device to the system. Printer device descriptions are created using the Create Device Description for Printer (CRTDEVPRT) command; diskette device descriptions are creating using the Create Device Description for Diskette (CRTDEVDKT) command. See the *Device Configuration Guide* for more information about specifying device descriptions.

Device File Considerations for Spooled Output

Some attributes of the printer or diskette file used by the application program apply to the spooled output file and how the spooled file is processed by the system when the application program ends. These attributes can be specified on the following commands:

CHGDKTF	Change Diskette File
CRTDKTF	Create Diskette File
OVRDKTF	Override with Diskette File
CHGPRTF	Change Printer File
CRTPRTF	Create Printer File
OVRPRTF	Override with Printer File

The parameters used to specify these attributes are:

Parameter	**Description**
OUTQ	The output queue for the file
FORMTYPE	The type of forms to be used
COPIES	The number of copies to be produced
MAXRCDS	The maximum number of records
SCHEDULE	When the spooled file is available for the writer
SAVE	Whether the file should be saved on the queue after it has been written to the device
FILESEP	The number of file separator pages to be printed before each file
HOLD	Whether the spooled file should be held on the queue until it is released
OUTPTY	The output priority of the file
USRDTA	The user data tag describing the file

Spooled Output Files

Summary of Spooled Output File Commands

The following commands may be used to work with spooled output files. For detailed descriptions of the commands, see the *CL Reference*.

CHGSPLFA Change Spooled File Attributes: Allows you to change some attributes of a spooled file, such as the output queue name or the number of copies requested, while the spooled file is on an output queue.

CPYSPLF Copy Spooled File: Copies a spooled output file to a specified database file. The database file may then be used for other applications, such as those using microfiche or data communications.

DLTSPLF Delete Spooled File: Deletes a spooled file from an output queue.

DSPSPLF Display Spooled File: Allows you to display data records of a spooled output file.

HLDSPLF Hold Spooled File: Stops the processing of an output file by a spool writer. The next output file in line will be processed.

RLSSPLF	Release Spooled File: Releases a previously held output file for processing by the spool writer.
WRKSPLF	Work with Spooled Files: Allows you to display or print a list of all spooled output files on the system.
WRKSPLFA	Work with Spooled File Attributes: Shows the current attributes of a spooled output file.

Locating Your Spooled Output

The Work with Spooled Files (WRKSPLF) command can be used to display or print all the spooled files that you have created. This is the easiest way to find your output if you do not know the name of the output queue where it has been placed.

File Redirection

File redirection occurs when a spooled file is sent to an output device other than the one for which it was originally intended. File redirection may involve devices that process different media (such as printer output sent to a diskette device) or devices that process the same type of media but are of different device types (such as 5219 Printer output sent to a 4224 Printer).

Depending on the new output device for the spooled file, the file may be produced just as it would have been on the originally specified device. However, differences in devices often cause the output to be formatted differently. In these cases, the system sends an inquiry message to the writer's message queue to inform you of the situation and allow you to specify whether you want printing to continue. For more information about print file redirection, see Chapter 5, "Printer Support."

Output Queues

Batch and interactive job processing may result in spooled output records that are to be produced on an output device, such as a printer or diskette drive. These output records are stored in spooled output files until they can be produced. There may be many spooled output files for a single job.

When a spooled output file is created, the file is placed on an output queue. Each output queue contains an ordered list of spooled output files. A job can have spooled files on one or more output queues. All spooled files on a particular output queue should have a common set of output attributes, such as device, form type, and lines per inch. Using common attributes on an output queue reduces the amount of intervention required and increases the device throughput.

The following lists the parameters on the Create Output Queue (CRTOUTQ) command and what they specify:

- DSPDTA: Whether users without any special authority but who do have *USE authority to the output queue can display, copy, or send the contents of spooled files other than their own.
- JOBSEP: How many, if any, job separator pages are to be produced between the output of each job when the output is printed.
- OPRCTL: Whether a user having job control authority can control the output queue (for example, if the user can hold the output queue).
- SEQ: Controls the order in which files will be sorted on the output queue. See "Order of Spooled Files on an Output Queue" on page 6-6 for more information.

- AUTCHK: Specifies what type of authority to the output queue will enable a user to control the output queue (for example, enable the user to hold the output queue).
- AUT: Public authority.
- TEXT: Text description.

Summary of Output Queue Commands

The following commands may be used to create and control output queues. For detailed descriptions of the commands, see the *CL Reference*.

CHGOUTQ Change Output Queue: Allows you to change certain attributes of an output queue, such as the sequence of the spooled files on the output queue.

CLROUTQ Clear Output Queue: Removes all spooled files from an output queue.

CRTOUTQ Create Output Queue: Allows you to create a new output queue.

DLTOUTQ Delete Output Queue: Deletes an output queue from the system.

HLDOUTQ Hold Output Queue: Prevents all spooled files from being processed by the spool writer.

RLSOUTQ Release Output Queue: Releases a previously held output queue for processing by the spool writer.

WRKOUTQ Work with Output Queue: Shows the overall status of all output queues, or the detailed status of a specific output queue.

WRKOUTQD Work with Output Queue Description: Shows descriptive information for an output queue.

Default Printer Output Queues

When a printer is configured to the system, the system automatically creates the printer's default output queue in library QUSRSYS. The output queue is given a text description of `'Default output queue for printer xxxxxxxxxx'`, where `xxxxxxxxxx` is the name of the printer.

The AUT parameter for the output queue is assigned the same value as that specified by the AUT parameter for the printer device description. All other parameters are assigned their default values. Use the Change Command Default (CHGCMDDFT) command to change the default values used when creating output queues with the CRTOUTQ command.

The default output queue for a printer is owned by the user who created the printer device description. In the case of automatic configuration, both the printer and the output queue are owned by the system profile QPGMR.

Default System Output Queues

The system is shipped with the defaults on commands to use the default output queue for the system printer as the default output queue for all spooled output. The system printer is defined by the QPRTDEV system value.

When a spooled file is created by opening a device file and the output queue specified for the file cannot be found, the system will attempt to place the spooled file on output queue QPRINT in library QGPL. If for any reason the spooled file cannot be placed on output queue QPRINT, an error message will be sent and the output will not be spooled.

The following output queues are supplied with the system:

QDKT	Default diskette output queue
QPRINT	Default printer output queue
QPRINTS	Printer output queue for special forms
QPRINT2	Printer output queue for 2-part paper

Creating Your Own Output Queues

You can create output queues for each user of the system. For example:

```
CRTOUTQ OUTQ(QGPL/JONES) TEXT('Output queue for Mike Jones')
```

Order of Spooled Files on an Output Queue

The order of spooled files on an output queue is mainly determined by the status of the spooled file. A file that is being processed by a writer (WTR status) is placed at the top of the queue. The status of a file being processed is WTR unless another user is holding it and the writer has not yet finished processing the file. (In that case, the file has a status of HLD). All other files with a status of RDY are listed on the output queue after the file being processed by a writer, followed by files with statuses other than RDY.

Within each type of spooled file (RDY and non-RDY files) the following information causes a further ordering of the files. The items are listed in sequence based on the amount of importance they have on the ordering of spooled files, with the first item having the most importance.

1. The output priority of the spooled file.
2. A date and time field (time stamp).

   For output queues with SEQ(*JOBNBR) specified, the date and time that the job which created the spooled file entered the system are the date and time field. (A sequential job number is also assigned to the job when it enters the system.)

   For output queues with SEQ(*FIFO) specified, the date and time field is set to the current system date and time when any of the following occur:

   - A spooled file is created by opening a device file.
   - The output priority of the job which created the spooled file is changed.
   - The status of the spooled file changes from RDY to HLD, SAV, OPN, or CLO; or the status changes from HLD, SAV, OPN, or CLO to RDY.
   - A spooled file is moved to another output queue which has SEQ(*FIFO) specified.
3. The SCHEDULE parameter value of the spooled file. Files with SCHEDULE(*JOBEND) specified are grouped together and placed after other files of the same job that have SCHEDULE(*IMMED) or SCHEDULE(*FILEEND) specified.
4. The spool number of the file.

Because of the automatic sorting of spooled files, different results occur when SEQ(*JOBNBR) is specified for an output queue than when SEQ(*FIFO) is specified. For example, when a spooled file is held and then immediately released on an output queue with SEQ(*JOBNBR) specified, the file will end up where it started; but if the same file were held and then immediately released on an output queue with SEQ(*FIFO) specified, the file would be placed at the end of the spooled files which have the same priority and a status of RDY.

For first-in-first-out output queues, the date and time change to the current date and time when:

- A spooled file moves from one queue to a SEQ(*FIFO) queue.
- The status changes to RDY or from RDY.

  Note: The date and time do not change when the reason the status changes from RDY to WTR or from WTR to RDY is because the writer was canceled.

- The priority of the spooled file changes.

Using Multiple Output Queues

You may want to create multiple output queues for:

- Special forms printing
- Output to be printed after normal working hours
- Output that is not printed

  An output queue can be created to handle spooled files that need only to be displayed or copied to a database file. Care should be taken to remove unneeded spooled output files.

- Special uses

  For example, each programmer could be given a separate output queue.

- Output of special IBM files

  You may want to consider separate queues for the following IBM-supplied files:

  - QPJOBLOG: You may want all job logs sent to a separate queue.
  - QPPGMDMP: You may want all program dumps sent to a separate queue so you can review and print them if needed or clear them daily.
  - QPSRVDMP: You may want all service dumps sent to a separate queue so the service representative can review them if needed.

Controlling Multiple Output Queues

Controlling multiple output queues requires both finding where your output is and determining how to print it if a writer is not started to the queue.

The Work with Spooled Files (WRKSPLF) command can be used to display all the spooled files that you have created. This is the easiest way to find your output if you do not know the name of the output queue where it has been placed. The files are listed in the same order as they would be on a SEQ(*FIFO) output queue. (See "Order of Spooled Files on an Output Queue" on page 6-6 for more information.)

If you know the name of the output queue that contains the spooled file, the Work with Output Queue (WRKOUTQ) command can be used to display the queue to determine its position on the queue. The WRKOUTQ command also provides the option to display all the output queues that exist on your system.

If a writer is not started to the output queue that contains the spooled file you wish to print, you have three options in order to print the file:

- Use the Change Spooled File Attributes (CHGSPLFA) command to move the spooled file to an output queue that has a writer started to it.
- Select a printer that is not being used and use the Start Printer Writer (STRPRTWTR) command to start a writer that will print the spooled files on your output queue.
- Select a writer that is started to a different output queue and use the Change Writer (CHGWTR) command to change the writer to produce the spooled files on your output queue.

The Work with Writers (WRKWTR) command can be used to find a list of printers attached to your system and to determine whether a writer is started to a printer.

Output Queue Recovery

If a job with spooled output is running when the job or system stops abnormally, the files remain on the output queue. Some number of records written by active programs may still be in main storage when the job ends and will be lost. You should check these files to ensure that they are complete before you decide to continue using the files.

You can use the SPLFILE parameter on the End Job (ENDJOB) command to specify if all spooled output files (except QPJOBLOG) created by the job are to be kept for normal processing by the writer, or if these files are to be deleted.

If an abnormal end occurs, the spooled output file QPJOBLOG will be written at the next IPL of the system.

If a writer fails while a spooled file is being printed, the spooled file remains on the output queue intact.

If an output queue becomes damaged such that it cannot be used, you will be notified by a message sent to the system operator message queue. The message will come from a system function when a writer or a job tries to put or take spooled files from the damaged queue.

A damaged output queue is automatically deleted by the system during the next IPL. You cannot delete a damaged job or output queue yourself. After the system is restarted, the deleted queue can be recreated by entering the Create Output Queue (CRTOUTQ) command.

Note: If the output queue that was damaged was the default output queue associated with a printer, the system will automatically re-create the output queue using the same public authority as specified for the device and the CRTOUTQ command defaults for the other parameters. When this happens, you should verify that these attributes are the correct ones for the output queue. To change the default attributes used when re-creating output queues, use the Change Command Default (CHGCMDDFT) command to change the defaults for the parameters on the CRTOUTQ command. See the *CL Reference* for more information about the CHGCMDDFT command.

Job information that was on a damaged output queue will be lost when the queue is deleted. You will also receive messages during the IPL that indicate which files were not completed. You or system users can also determine the status of a particular job by using the Display Job (DSPJOB) command. Jobs whose spooled files were not produced may have to be submitted again.

Spool Writers

A writer is an OS/400 program that takes spooled output files from an output queue and produces them on an output device. The spooled output files that have been placed on a particular output queue will remain stored in the system until a writer is started to the output queue.

The writer takes spooled files one at a time from the output queue, based on their priority. The writer produces a spooled output file only if its entry on the output queue indicates that it has a ready (RDY) status. You can display the status of a particular output file using the Work with Output Queue (WRKOUTQ) command.

If the spooled output file has a ready status, the writer takes the entry from the output queue and produces the specified job and/or file separators, followed by the output data in the file. If the output file does not have a ready status, the writer leaves the entry on the output queue and goes on to the next entry. In most cases the writer will continue to produce output files (preceded by job and file separators) until all files with a ready status have been taken from the output queue.

The AUTOEND parameter on the start writer commands determines whether the writer continues to wait for new files to become available to be written, end after producing one file, or end after all spooled files with ready status have been taken from the output queue.

Summary of Spool Writer Commands

The following commands may be used to control spool writers. For detailed descriptions of the commands, see the *CL Reference*.

STRDKTWTR Start Diskette Writer: Starts a spooling writer to a specified diskette device to produce spooled output files on that device.

STRPRTWTR Start Printer Writer: Starts a spooling writer to a specified printer device to produce spooled output files on that device.

CHGWTR Change Writer: Allows you to change some writer attributes, such as form type, number of file separator pages, or output queue attributes.

HLDWTR Hold Writer: Stops a writer at the end of a record, at the end of a file, or at the end of a page.

RLSWTR Release Writer: Releases a previously held writer for additional processing.

ENDWTR End Writer: Ends a spool writer and makes the associated output device available to the system.

Restarting and Controlling Printing

Several cases exist where you may wish to restart printing or control a file while it is being printed.

- The system ended while a spooled file was being printed.
- You wish to print only selected portions of a large spooled file.
- A spooled file needs to be printed immediately.

To finish printing a file that was partially printed when the system ended, use the WRKOUTQ command to determine the name of the file that was partially printed. The complete file name is composed of file, file number, job, user, and job number. Check your printed output to find the last page that was printed. Use the Start Printer Writer (STRPRTWTR) to start a writer to the correct printer and output queue, specifying the complete file name and the starting page in order to resume printing at the correct location.

To print selected pages of a spooled file that has already started printing, the Hold Writer (HLDWTR) and Release Writer (RLSWTR) commands can be used to skip through the spooled file. Whenever a section of the spooled file is reached that is not needed, use the HLDWTR command to hold the writer. Then, release the writer to the next section of the spooled file to be printed by using the PAGE parameter on the RLSWTR command.

To print a spooled file right away when another spooled file is being printed, move the spooled file to be printed to the top of the output queue and then bypass the remaining output in the spooled file that is now printing. To move a spooled file to the top of the output queue, specify *NEXT for the PRTSEQ parameter on the Change Spooled File Attributes (CHGSPLFA) command. To bypass the remaining output in the spooled file being printed, hold the writer with the HLDWTR command and then specify *BYPASS for the OPTION parameter on the RLSWTR command. The writer will stop printing the current spooled file and begin printing the spooled file you moved to the top of the output queue.

Note: When you requested that the printing spooled file be bypassed, the writer held the file. After the high priority spooled file has printed, you probably want to resume printing this file. Use the Release Spooled File (RLSSPLF) command to release the held file so that it is eligible to print.

Spooled File Security

Spooled file security is primarily controlled through the output queue which contains the spooled file. In general, there are four ways that a user can become authorized to control a spooled file (for example, hold or release the spooled file):

- User is assigned spool control authority (SPCAUT(*SPLCTL)) in the user's user profile.

- User is assigned job control authority (SPCAUT(*JOBCTL)) in the user's user profile and the output queue is operator controllable (OPRCTL(*YES)).

- User has the required object authority for the output queue. The required object authority is specified by the AUTCHK keyword on the CRTOUTQ command. A value of *OWNER indicates that only the owner of the output queue is authorized via the object authority for the output queue. A value of *DTAAUT indicates that users with *CHANGE authority to the output queue are authorized to control the output queue.

Note: The specific authority required for *DTAAUT are *READ, *ADD, and *DLT data authorities.

- A user is always allowed to control the spooled files created by that user.

For the Copy Spooled File (CPYSPLF), Display Spooled File (DSPSPLF), and Send Network Spooled File (SNDNETSPLF) commands, in addition to the four ways already listed, there is an additional way a user can be authorized. If DSPDTA(*YES) was specified when the output queue was created, any user with *USE authority to the output queue will be allowed to run the command. The specific authority required is *READ data authority.

See the *CL Reference* for details about the authority requirements for individual commands.

To place a spooled file on an output queue, one of the following authorities is required:

- User is assigned spool control authority (SPCAUT(*SPLCTL)) in the user's user profile.
- User is assigned job control authority (SPCAUT(*JOBCTL)) in the user's user profile and the output queue is operator controllable (OPRCTL(*YES)).
- User has *READ authority to the output queue. This authority can be given to the public by specifying (AUT(*USE)) on the CRTOUTQ command.

Producing Spooled Output Files

The following section describes the considerations that apply for producing spooled output. These include some of the values that can be specified in a device file used by an application program, and considerations for opening, closing, and performing output operations using the device file.

Using the SCHEDULE Parameter

The SCHEDULE parameter can be specified with one of three values to control when the spooled file is available for a writer to produce the file.

*FILEEND Specifies that the file is made available to the writer when the file is closed

*JOBEND Specifies that the file is made available to the writer when the job is ended

*IMMED Specifies that the file is made available to the writer when the file is opened

*JOBEND Considerations

The SCHEDULE(*JOBEND) spooled output files of a job are grouped together on their output queues when the job completes. All SCHEDULE(*JOBEND) spooled files of the same job which are grouped together on a queue are produced together by the writer. Another spooled file can be added to the top of the queue while the writer is producing a SCHEDULE(*JOBEND) spooled file. After a writer produces one file of a job which is SCHEDULE(*JOBEND), it checks the following file on the queue. If this file is from the same job and is also SCHEDULE(*JOBEND), the writer produces it next. However, if the file is from a different job or is not SCHEDULE(*JOBEND), the first file on the queue is produced next.

If you want your SCHEDULE(*JOBEND) spooled files grouped together on a SEQ(*FIFO) output queue, you must be careful not to separate the spooled files. File operations such as HLDSPLF, CHGSPLFA, and RLSSPLF are performed one at a time (even from a Work with Output Queue (WRKOUTQ) screen. If operations to other files on the queue are done at the same time, your spooled files can be separated on a SEQ(*FIFO) queue. If your SCHEDULE(*JOBEND) spooled files separate, you can regroup them by changing their output priority with the Change Job (CHGJOB) command.

*IMMED Considerations

When a writer is producing a SCHEDULE(*IMMED) spooled output file, it may catch up to the program producing the output. When this happens, the writer must wait for the program to produce more output. Because of this, you should be careful using *IMMED for the schedule option. When this happens, the writer cannot process other spooled files. Moreover, the device cannot be used for other productive work.

Normally, spooled files that are created with SCHEDULE(*IMMED) specified are assigned a smaller internal buffer than spooled files which are created with SCHEDULE(*JOBEND) or SCHEDULE(*FILEEND) specified. Using SCHEDULE(*IMMED) to assign a smaller internal buffer may allow the spooled data to be produced sooner, but can also adversely affect performance because more disk operations are needed for the same amount of spooled data.

A large internal buffer is always used for spooled print files that use certain special device requirements or are created by AS/400 PC Support. Special device requirements include:

- Defined characters
- Graphics 4214
- Graphics 4234
- Graphics 522X

Changing the SCHEDULE parameter of a spooled file with the CHGSPLFA command does not affect the internal buffers used for that file.

Using the OUTPTY Parameter

Once a file is available for a writer to produce, the OUTPTY parameter determines the order in which the files will be produced. The OUTPTY parameter supports the value *JOB (use the default output priority for the job) and a range of values from 1 to 9. All available files that have an output priority of 1 are positioned at the top of the output queue and are the first files to be printed. Next are the priority 2 files and so on. Priority 9 files are the last available files to be printed. By selecting the appropriate output priority for spooled files, you may insure that the spooled files needed right away will be printed first.

The priority of a spooled file when it is created is set from the printer device file. Use the Create, Change, or Override with Print File commands to set the OUTPTY parameter to the desired value before the file is opened. After the file is opened, the output priority of the spooled file can be changed by using the CHGSPLFA command.

Open Considerations

The following considerations apply to opening device files for spooled output:

- The output queue should be created for the type of output your program produces so that system operator intervention can be kept to a minimum while a writer is producing output. You should consider the following when creating an output queue:
  - What form of output is being produced (printer or diskette).
  - What print forms the output is produced on.
  - What kind of protection you want to place on your data. (Do you want someone else to be able to display your data?)
  - How many job separators you want.
- The SCHEDULE parameter specifies when output is to be made available to a writer. When a writer processes a specific file, it is dependent on things such as:
  - When the writer is started
  - Other output files on the queue
  - If the writer or the output queue is being held
- The parameters specified to produce output are saved until they are used by the writer.

Output Considerations

The following considerations apply to output operations performed on spooled files:

Force-End-of-Data Considerations: The force-end-of-data (FEOD) operation can make part of the spooled output available to the writer unless SCHEDULE(*JOBEND) or HOLD(*YES) are specified for the file. This operation lets you write parts of a spooled output file, such as writing one order at a time. You should not use the force-end-of-data function for normal output.

Note: For more spooled output file I/O considerations, see the I/O considerations sections for using the devices discussed in Chapter 5, "Printer Support" and Chapter 8, "Diskette Support."

Close Considerations

When the schedule option is *FILEEND, the output file is made available to the writer. The file resources used by the program are deallocated.

If an application program is writing data when the system ends abnormally, the spooled file will be shown containing 0 pages on spool displays such as the WRKOUTQ, WRKSPLFA, and WRKJOB displays. Records that were stored in internal system buffers will be lost.

Spooled output files that contain no records (OPEN and CLOSE, but no I/O) are automatically deleted by the system when the application closes the device file. Writers will not select these files for printing unless SCHEDULE(*IMMED) is specified and the writer selects the file before it is closed.

Controlling the Number of Spooled Files in Your System

The number of spooled files in your system should be limited. When a job is completed, spooled files and internal job control information are kept until the spooled files are printed or canceled. The number of jobs on the system and the number of spooled files known to the system increase the amount of time needed to perform IPL and internal searches, and increases the amount of temporary storage required.

The number of jobs known to the system can be displayed using the Work with System Status (WRKSYSSTS) command.

You can use the Work with Spooled Files (WRKSPLF) command to identify spooled files that are no longer needed. By periodically entering the command:

```
WRKSPLF SELECT(*ALL)
```

you can determine which spooled files are older than 2 or 3 days, then delete the spooled files or contact the users who created them.

For detailed information on minimizing the number of job logs (for example, by using LOG(4 0 *NOLIST)), see the *CL Programmer's Guide*. For information regarding the use of system values to control the amount of storage associated with jobs and spooled files, refer to the *Work Management Guide*.

Copying Spooled Output Files

You can use the Copy Spooled File (CPYSPLF) command to copy a spooled output file to a physical file. The original spooled file is not affected by the copy operation and can still be produced by a spool writer. You may want to copy a spooled file to a database file for the following reasons:

- You can save paper by having reports that are normally printed produced on microfiche.
- You can copy a report to a database file and send it over communications lines for printing at another location.
- You have collected information in a spooled file by running one of the system display commands with OUTPUT(*PRINT) specified. The spooled file created by this operation can be copied to a database file so that it can be read and processed by the application program using it.

  Note: If you use this method of gathering information, remember that the system displays that you are spooling may change when new functions are added to the system.
- You can copy the file to a spooled file so you can direct the same output to a different output queue. For more information, refer to "Duplicate Spooled File (DUPSPLF) Sample Command" on page 6-18.

Specifying CPYSPLF Parameters

The CTLCHAR and CHLVAL parameters control the format of the copied output.

The Control Character (CTLCHAR) Parameter

The CTLCHAR parameter determines which control code is produced by the Copy Spooled File (CPYSPLF) command. You can select one of the following control codes:

- *NONE: No print control characters are created. You can use this code, for example, when printed displays (produced with the OUTPUT(*PRINT) parameter) are to be read by an application program.
- *FCFC: The first character of every record will contain one of the following ANSI control codes:

Code	Action before Printing a Line
' '	Space one line (blank code)
0	Space two lines
-	Space three lines
+	Suppress space
1	Skip to next channel 1
2	Skip to next channel 2
3	Skip to next channel 3
4	Skip to next channel 4
5	Skip to next channel 5
6	Skip to next channel 6
7	Skip to next channel 7
8	Skip to next channel 8
9	Skip to next channel 9
A	Skip to next channel 10
B	Skip to next channel 11
C	Skip to next channel 12

- *PRTCTL: The first 4 characters of every record will contain skip-before and space-before values. This code can be viewed as sssl where sss is the skip-before line value (001 to 255) and l is the space-before value (0, 1, 2, or 3). When one part of the code is created by the CPYSPLF command, the other part will be blank. In the following examples, ƀ represents a blank:

'001ƀ'	Skip to line 1 before printing
'099ƀ'	Skip to line 99 before printing
'ƀƀƀ1'	Space 1 line before printing
'ƀƀƀ0'	Do not space (or skip) before printing

  You can use this control code when printing with an RPG/400 program if the page size is not more than 99 lines long. To do this, move the control characters into the RPG/400 program's PRTCTL data structure space before and skip before fields and then print line.

- *S36FMT: Specifies that the format of the records copied to a database file is the same as created by $UASF on System/36 for COPYPRT. Only spooled print files can be copied when *S36FMT is specified. You can use this option when you plan to send the spooled file to a System/36.

  The first record placed in the database file for each spooled file to be copied is a heading record. Columns which are not defined will be blank.

Figures 6-1 and 6-2 describe the formats of the header and data records.

Table 6-1. Header Record Format

Beginning Column	Field Length	Description
1	1	The letter H (to indicate the heading record).
4	6	The spool ID of the entry. Valid spool IDs range from SP0000 to SP9999, and from A00000 to J99999.
12	8	The procedure name. This will be blank if the file was not created by a procedure.
22	8	The job name. (The last 2 characters of the name are truncated.)
32	8	The user ID of the spooled file creator. (The last 2 characters of the ID are truncated.)
42	8	The printer device file name. (The last 2 characters of the name are truncated.)
52	2	The System/36 printer ID that corresponds to the device the file will be printed on. The printer ID shown will be the ID for the System/36 operating environment that copies the file, not the environment that created the spooled file.
56	4	The forms identification. (This will be the first 4 characters of the form type of the spooled file.)
61	2	The number of copies (in binary).
65	2	The number of pages (in binary).
69	4	The number of records (in binary). This is the number of data records which follow this heading record.
74	2	The number of lines per page (in binary).
78	1	The letter I if this entry contains print records with double byte character set data.
81	1	The letter M if this entry contains print records with a length greater than 132.
84	1	Lines per inch (in binary).
85	1	Characters per inch (in binary).
86	1	Font ID (in binary). The AS/400 FONT print file parameter is converted to a binary 1 field. The maximum font ID on System/36 OCL is 255; the AS/400 system supports font IDs above 255. Any time an AS/400 font ID above 255 is used, this field is set to X'0B' for Courier 11 font. If you specify FONT(*CPI), the field contains X'00'.
87	1	Justify. Valid values are X'00' (0%), X'32' (50%), and X'64' (100%).
88	1	Align. (Y means to align forms, N means not to align forms.)
89	2	The maximum length of the print lines in the spooled file that was copied.
92	10	The nontruncated user ID of the spooled file creator.
102	10	The nontruncated printer device file name.
112	10	The nontruncated form type.
113	7	(Used internally by the system.)

The data records placed in the disk file for each copied spooled file have the following format:

Table 6-2. Data Record Format

Beginning Column	Field Length	Description
1	2	The page number (in binary).
3	2	The line number (in binary).
5	4	The record number (in binary).
9	1	The letter I if this print record contains double byte character set data.
10	1	A double byte character set shift-out character (hex 0E) if this print record starts with double byte character set data.
11	nnn	The data to be printed. (The field length will be the file record length minus 10. If the print data is longer than the field length, it is truncated; and if it is shorter than the field length, it is blank padded at the end.) The format of data may not exactly match the format which would be produced on System/36 when multiple prints are used to construct a single print line.

The 2-byte binary numbers are unsigned which means that a page number of 65 535 is the largest page number in a heading record or data record. When the actual number is larger, it will wrap beyond 65 535 to 0, then 1, 2, 3, and so on.

The Channel Value (CHLVAL) Parameter

You can use the CHLVAL parameter on the CPYSPLF command to assign line numbers for the different channels described above. You can use the first character forms control code to create microfiche with the database file or print the file with the Copy File (CPYF) command (which allows files to be printed using a first-character forms-control print file).

CPYSPLF Command Control Codes: Examples

To copy a spooled file named ORDERS in job NEWORDERS to database file PRTORDERS so that you can use the CPYF command to print the database file type:

```
CPYSPLF   FILE(ORDERS) JOB(NEWORDERS) TOFILE(PRTORDERS) +
          CTLCHAR(*FCFC)
```

To print the data from the physical file, you type the following:

```
OVRPRTF   FILE(QSYSPRT) CTLCHAR(*FCFC) CHLVAL(*NORMAL)
CPYF      FROMFILE(PRTORDERS) TOFILE(QSYSPRT)
DLTOVR    FILE(QSYSPRT)
```

For an additional example using the CPYSPLF command, see “Duplicate Spooled File (DUPSPLF) Sample Command” on page 6-18.

Sample Commands for Additional Spooling Support

You can define some functions to provide additional spooling support. Example source and documentation for the commands, files, and programs for these functions are part of library QUSRTOOL, which is an optionally installed part of the OS/400 program.

The spooling support is part of the Programming and System Management Tips and Techniques section of QUSRTOOL. See the member AAAMAP in file QATTINFO of library QUSRTOOL for information on where the documentation and example source is located for each of these functions.

Examples of spooling functions in QUSRTOOL are:

Duplicate Spooled File (DUPSPLF) Sample Command: The DUPSPLF command can be created to duplicate a spooled file and place the duplicate output in a different output queue. You can do this when you want the same spooled output to be printed on multiple printers and each printer normally works with a specific output queue.

Convert Output Queue (CVTOUTQ) Command Example: In some environments, you may want a function to place the information displayed by the WRKOUTQ command into a database file for processing. Each database record will contain some of the attributes of a spooled file and can be manipulated with other processing techniques, such as assigning a different output queue to all the spooled files.

Move Spooled File (MOVSPLF) Sample Command: A typical use of the previously described CVTOUTQ sample command is to move all spooled output files from one queue to another. The Move Spooled File (MOVSPLF) sample command performs this function.

Input Spooling Support

Input spooling takes the information from the input device, prepares the job for scheduling, and places an entry in a job queue. Using input spooling, you can usually shorten job run time, increase the number of jobs that can be run sequentially, and improve device throughput.

The main elements of input spooling are:

Job queue An ordered list of batch jobs submitted to the system for running and from which batch jobs are selected to run.

Reader A function that takes jobs from an input device or a database file and places them on a job queue.

When a batch job is read from an input source by a reader, the commands in the input stream are stored in the system as requests for the job, the inline data is spooled as inline data files, and an entry for the job is placed on a job queue. The job information remains stored in the system where it was placed by the reader until the job entry is selected from the job queue for processing by a subsystem.

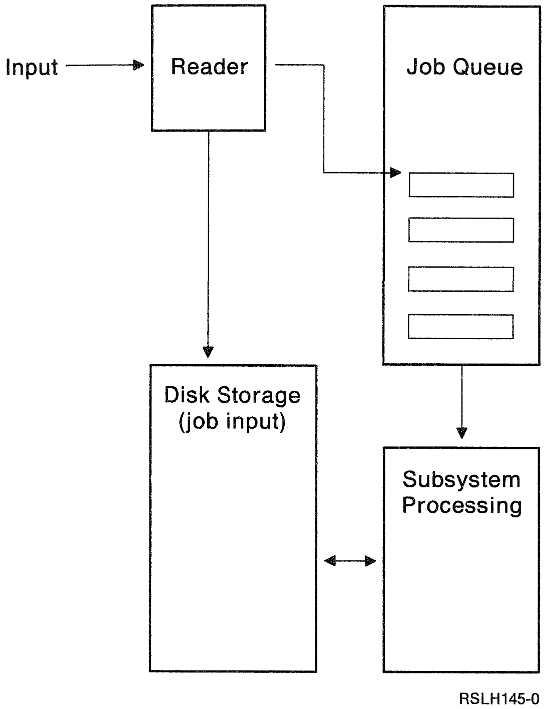
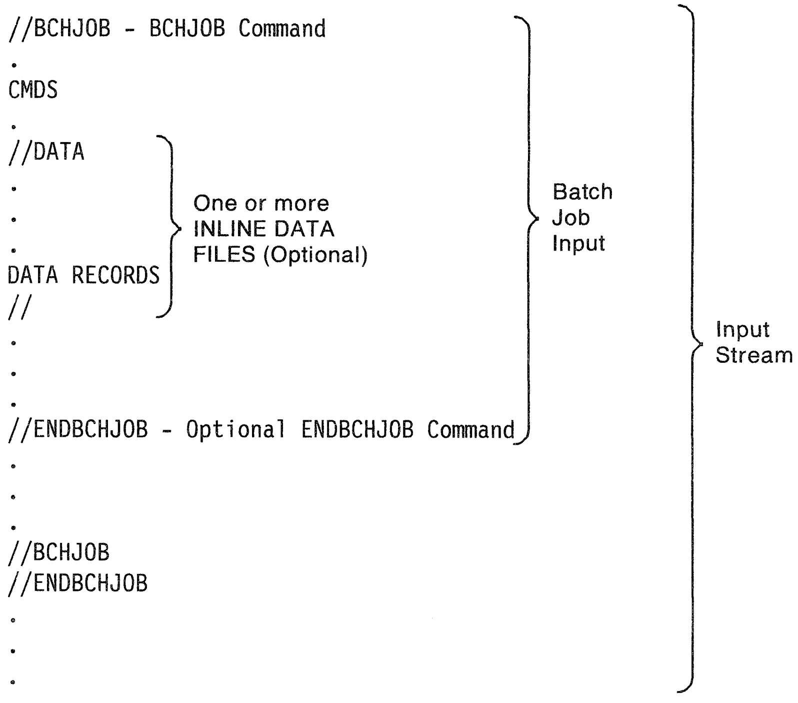


Figure 6-2. Relationship of Input Spooling Elements

You can use the reader functions to read an input stream from diskette or database files. The following example shows the typical organization of an input stream:



```
//BCHJOB - BCHJOB Command
.
CMDS
.
//DATA
.
.
.
DATA RECORDS
//
.
.
.
//ENDBCHJOB - Optional ENDBCHJOB Command
.
.
.
//BCHJOB
//ENDBCHJOB
.
.
.
```

RSLH116-2

The job queue on which the job is placed is specified on the JOBQ parameter on the BCHJOB command, on the start reader command, or in the job description. If the JOBQ parameter on the BCHJOB command is:

- *RDR: The job queue is selected from the JOBQ parameter on the start reader command.
- *JOBD: The job queue is selected from the JOBQ parameter in the job description.
- A specific job queue: The specified queue is used.

For jobs with small input streams, you may improve system performance by not using input spooling. The submit job commands (SBMDBJOB and SBMDKTJOB) read the input stream and place the job on the job queue in the appropriate subsystem, bypassing the spooling subsystem and reader operations.

If your job requires a large input stream to be read, you should use input spooling (STRDKTRDR or STRDBRDR command) to allow the job to be input independent of when the job is actually processed.

Summary of Job Input Commands

The following commands may be used when submitting jobs to the system. The start reader commands may be used for spooling job input; the submit job commands do not use spooling. For detailed descriptions of these commands, see the *CL Reference*.

BCHJOB Batch Job: Marks the start of a job in a batch input stream and defines the operating characteristics of the job.

DATA Data: Marks the start of an inline data file.

ENDBCHJOB	End Batch Job: Marks the end of a job in a batch input stream.
ENDINP	End Input: Marks the end of the batch input stream.
SBMDBJOB	Submit Database Jobs: Reads an input stream from a database file and places the jobs in the input stream on the appropriate job queues.
SBMDKTJOB	Submit Diskette Jobs: Reads an input stream from diskette and places the jobs in the input stream on the appropriate job queues.
STRDBRDR	Start Database Reader: Starts a reader to read an input stream from a database file and places the job in the input stream on the appropriate job queue.
STRDKTRDR	Start Diskette Reader: Starts a reader to read an input stream from diskette and places the job in the input stream on the appropriate job queue.

Job Queues

A job queue is an ordered list of jobs waiting to be processed by a particular subsystem. Jobs will not be selected from a job queue by a subsystem unless the subsystem is active and the job queue is not held. You can use job queues to control the order in which jobs are run.

A base set of job queues is provided with your system. In addition, you may create additional job queues that you need.

IBM-Supplied Job Queues

Several job queues are provided by IBM when your system is shipped. IBM supplies job queues for each IBM-supplied subsystem.

QCTL	Controlling subsystem queue
QBASE	QBASE subsystem job queue
QBATCH	Batch subsystem queue
QINTER	Interactive subsystem queue
QPGMR	Programmer subsystem queue
QSPL	Spooling subsystem queue
QSYSSBSD	QSYSSBSD subsystem job queue
QS36MRT	QS36MRT job queue
QS36EVOKE	QS36EVOKE job queue
QFNC	Finance subsystem job queue
QSNADS	QSNADS subsystem job queue

Using Multiple Job Queues

In many cases, using QBATCH as the only job queue with the default of one active job will be adequate for your needs. If this is not adequate, you may want to have multiple job queues so that some job queues are active during normal working hours, some are for special purposes, and some are active after normal working hours. For example, you could designate different job queues for:

- Long-running jobs so you can control how many jobs are active at the same time.

  You may also want these jobs to use a lower priority than the other batch jobs.

- Overnight jobs that are inconvenient to run during normal working hours.

  For example, to run a Reorganize Physical File Member (RGZPFM) command on a large database file requires an exclusive lock on the file. This means that other users cannot access the file while this operation is taking place. Additionally, this operation could take a long time. It would be more efficient to place this job on a job queue for jobs which run during off-shift hours.

- High-priority jobs.

  You may want to have a job queue to which all high-priority work is sent. You could then ensure that this work is completed rapidly and is not delayed by lower-priority jobs.

- Jobs that are directed to particular resource requirement such as diskette or tape.

  Such a job queue would need a MAXACT parameter of 1 in the job queue entry of the subsystem description so that only one job at a time uses the resource.

  For example, if a tape is used for several jobs, all jobs using tape would be placed on a single job queue. One job at a time would then be selected from the job queue. This would ensure that no two jobs compete for the same device at the same time. If this happened, it would cause one of the jobs to end with an allocation error.

  Note: Tape output cannot be spooled.

- Programmer work.

  You may want to have a job queue to handle programmer work or types of work that could be held while production work is being run.

- Sequential running of a series of jobs.

  You may have an application in which one job is dependent on the completion of another job. If you place these jobs on a job queue that selects and runs one job at a time, this would ensure the running sequence of these jobs.

  If a job requires exclusive control of a file, you may want to place it on a job queue when the queue is the only one active on the system, such as during the night or on a weekend.

If you use multiple job queues, you will find that control of the various job queues is a main consideration. You will usually want to control:

- How many job queues exist.
- How many job queues are active in a particular subsystem at the same time.
- How many active jobs can be selected from a particular job queue at a particular time.
- How many jobs can be active in a subsystem at a particular time.

Creating Your Own Job Queues

There are numerous reasons why you may decide that you need job queues in addition to the ones supplied by IBM. Additional job queues can be created by using the Create Job Queue (CRTJOBQ) command:

```
CRTJOBQ QGPL/QNIGHT TEXT('Job queue for night-time jobs')
```

The following lists the parameters on the Create Job Queue (CRTJOBQ) command and what they specify:

- OPRCTL: Whether a user having job control authority can control the job queue (for example, if the user can hold the job queue)
- AUTCHK: Specifies what type of authority to the job queue will enable a user to control the job queue (for example, enable the user to hold the job queue)

- AUT: Public authority
- TEXT: Text description

Multiple Job Queues for a Subsystem

If the priority and sequence of the next job queue to be used is important, you may want to assign and control multiple job queues for each subsystem. One use of multiple job queues is to establish a high-priority and a normal-priority job queue within a subsystem, allowing only one active job on each queue at any time.

Another example: If your production batch jobs need to be completed before a special after-hours job queue can be made active, you could have the last job in the normal batch job queue release the after-hours job queue.

Refer to the SEQNBR parameter in the Add Job Queue Entry (ADDJOBQE) command in the *CL Reference* to determine how to set priorities for jobs on job queues. For more information, refer to the *Work Management Guide*.

Using the WRKJOBQ Command

Jobs already on the job queue can be controlled using the Work with Job Queue (WRKJOBQ) command.

The WRKJOBQ command lists either:

- All the job queues on the system
- All the jobs on a specific job queue

The ability to list all the job queues is important when you are not sure what job queue was used for a job. From the list of all job queues, you can look at each job queue to find the job. The display of a specific job queue provides a list of all the jobs on the queue in the order in which they will become active.

Transferring Jobs

If a job is on a job queue and is not yet active, you can change the job to a different job queue using the JOBQ parameter on the Change Job (CHGJOB) command.

If a job becomes active, it is possible to place it back on a job queue. See the *Work Management Guide* for a discussion of the Transfer Job (TFRJOB) and Transfer Batch Job (TFRBCHJOB) commands.

Job Queue Security

You can maintain a level of security with your job queue by authorizing only certain persons (user profiles) to that job queue. In general, there are three ways that a user can become authorized to control a job queue (for example, hold or release the job queue):

- User is assigned spool control authority (SPCAUT(*SPLCTL)) in the user's user profile.
- User is assigned job control authority (SPCAUT(*JOBCTL)) in the user's user profile and the job queue can be controlled by the operator (OPRCTL(*YES)).
- User has the required object authority to the job queue. The required object authority is specified by the AUTCHK parameter on the CRTJOBQ command. A value of *OWNER indicates that only the owner of the job queue is authorized via the object authority for the job queue. A value of *DTAAUT indicates that users with *CHANGE authority for the job queue are authorized to control the job queue.

Note: The specific authority required for *DTAAUT are *READ, *ADD, and *DLT data authority.

See *CL Reference* for more information about authority requirements for individual commands.

These three methods of authorization apply only to the job queue, not to the jobs on the job queue. The normal authority rules for controlling jobs apply whether the job is on a job queue or whether it is currently running. See the *Work Management Guide* for details on the authority rules for jobs.

Job Queue Recovery

If a command fails or the system stops abnormally while a reader or a submit jobs command is running and a partial job (not all the input stream has been read) is placed on the queue, the entire job must be resubmitted to the job queue.

If a job is on a job queue when the system stops abnormally without damaging that job queue, the job remains intact on the job queue and is ready to run when the system becomes active again.

If the system stops abnormally while a job is running, the job is lost and must be resubmitted to the job queue.

If a job queue becomes damaged such that it cannot be used, you will be notified by a message sent to the system operator message queue. The message will come from a system function when a reader, Submit Jobs command, or a job tries to put or take jobs from the damaged queue.

A damaged job queue is automatically deleted by the system during the next IPL. You cannot delete a damaged job queue yourself. After the system is restarted, the deleted queue can be recreated by entering the Create Job Queue (CRTJOBQ) command.

Job information that was on a damaged job queue will be lost when the queue is deleted. You will also receive messages during the IPL that indicate which jobs were not completed. You or system users can also determine the status of a particular job by using the Display Job (DSPJOB) command. Jobs that were not processed may have to be resubmitted.

Using an Inline Data File

An inline data file is a data file that is included as part of a batch job when the job is read by a reader or a submit jobs command. An inline data file is delimited in the job by a //DATA command at the start of the file and by an end-of-data delimiter at the end of the file. The end-of-data delimiter can be a user-defined character string or the default of //.

The // must appear in positions 1 and 2. If your data contains a // in positions 1 and 2, you should use a unique set of characters such as:

```
// *** END OF DATA
```

To specify this as a unique end-of-data delimiter, the ENDCHAR parameter on the //DATA command should be coded as:

```
ENDCHAR('// *** END OF DATA')
```

Note: Spooled inline data files can be accessed only during the first routing step of a batch job. If a batch job contains a Transfer Job (TFRJOB), a Reroute Job (RRTJOB), or a Transfer Batch Job (TFRBCHJOB) command, the spooled inline data files cannot be accessed in the new routing step.

An inline data file can be either named or unnamed. For an unnamed inline data file, either QINLINE is specified as the file name in the //DATA command or no name is specified. For a named inline data file, a file name is specified.

A named inline data file has the following characteristics:

- It has a unique name in a job; no other inline file can have the same name.
- It can be used more than once in a job.
- Each time it is opened, it is positioned to the first record.

To use a named inline data file, you must either specify the file name in the program or use an override command to change the file name specified in the program to the name of the inline data file. The file must be opened for input only.

An unnamed inline data file has the following characteristics:

- Its name is QINLINE. (In a batch job, all unnamed inline data files are given the same name.)
- It can only be used once in a job.
- When more than one unnamed inline data file is included in a job, the files must be in the input stream in the same order as when the files are opened.

To use an unnamed inline data file, do one of the following:

- Specify QINLINE in the program.
- Use an override file command to change the file name specified in the program to QINLINE.

If your high-level language requires unique file names within one program, you can use QINLINE as a file name only once. If you need to use more than one unnamed inline data file, you can use an override file command in the program to specify QINLINE for additional unnamed inline data files.

Note: If you run commands conditionally and process more than one unnamed inline data file, the results cannot be predicted if the wrong unnamed inline data file is used.

Open Considerations for Inline Data Files

The following considerations apply to opening inline data files:

- Record length specifies the length of the input records. (Record length is optional.) When the record length exceeds the length of the data, a message is sent to your program. The data is padded with blanks. When the record length is less than the data length, the records are truncated.

- When a file is specified in a program, the system searches for the file as a named inline data file before it searches for the file in a library. Therefore, if a named inline file has the same name as a file that is not an inline file, the inline file is always used, even if the file name is qualified by a library name.

- Named inline data files *can* be shared between programs in the same job by specifying SHARE(*YES) on a create file or override file command.

  For example, if an override file command specifying a file named INPUT and SHARE(*YES) is in a batch job with an inline data file named INPUT, any programs running in the job that specify the file name INPUT will share the same named inline data file.

Unnamed inline data files *cannot* be shared between programs in the same job.

- When you use inline files, you should make sure the correct file type is specified on the //DATA command. For example, if the file is to be used as a source file, the file type on the //DATA command must be source.
- Inline data files must be opened for input only.

Spooling Subsystem

The spooling subsystem, QSPL, is used for processing the spooling readers and writers. The subsystem needs to be active only when readers or writers are active. The spooling subsystem and the individual readers and writers can be controlled from jobs that run in other subsystems.

The start reader and start writer commands submit jobs to the job queue of the spooling subsystem.

Requests for reader or writer jobs are placed on the QSPL job queue, and the next entry on the QSPL job queue is selected to run if:

- The number of active jobs is less than the QSPL subsystem attribute of MAXJOBS.
- The number of active jobs from the QSPL job queue is less than the MAXACT attribute for the job queue.

Work management associated with the QSPL subsystem is similar to that for other subsystems as described in the *Work Management Guide*.

Spooling Library

The spooling library (QSPL) contains database files that are used to store data for spooled input and output files. Each file in library QSPL can have several members. Each member contains all the data for one spooled input or output file.

When the spooled file is printed or canceled, its associated database member in the spooling library is cleared of records, but not removed, so that it can be used for another spooled input or output file. If no database members are available in library QSPL, then a member is automatically created.

The system keeps a record of the number of times a database member in the spool library is used. During a normal initial program load (IPL), those members that have not been used for the past seven consecutive normal IPLs are removed, and their disk space is freed for other uses. While the database members are being removed, the following status message (CPI3301) is displayed:

```
Normal cleanup of excess spool members in progress
```

A maximum of 200 database members can be removed during a single initial program load (IPL).

Printing a spooled file or clearing an output queue does not reduce the number of associated database members. If an excessive number of associated database members were created on your system (for example, if a program went into a loop and created thousands of spooled files), the spool database members use storage on the system even if you clear the output queue. To remove these spool database members, install the operating system again with a cold start and abbreviated installation. The cold start removes all database files associated with job queues and output queues. For more information, see the *Device Configuration Guide*.

The procedure previously described is the only allowable way to remove files from the QSPL library. Any other way can cause severe problems. It is best to keep the QSPL library small by periodically deleting old spooled files with the DLTSPLF or CLROUTQ commands. This procedure allows database members to be used again, rather than having to increase the size of the the spooling library to accommodate new database members.

Displaying the data in the QSPL library may also prevent the data from being cleared, wasting storage space. Any command or program used to look at a database file in the QSPL library must allocate the database file and member; if a writer tries to remove an allocated member after printing is completed, it will not be able to clear the member. Because the member is not cleared, it cannot be used for another spooled file and it will not be removed after seven IPLs.

Saving a database file in the QSPL library can cause more problems than displaying the data in one member of the file because all members will be allocated a much longer time when a database file is saved. Because restoring these files destroys present and future spooled file data, there is no reason to save one of these files.

Chapter 7. Tape Support

The AS/400 system supports 1/2-inch reel and 1/4-inch cartridge tape for creating backup copies and offline storage of information, or for transferring information to other devices or systems. Tape storage capacities range from 41 to 161 megabytes, using densities of 1600, 3200, 6250, or 10,000 bits per inch (bpi), depending on the tape device used.

Tape is particularly useful for storing large amounts of data:

- Reading and writing information with tape is faster than with diskette.
- More information can be stored on a single tape than on diskette.

For information about saving and restoring files and libraries, see the *Backup and Recovery Guide*.

Related CL Commands

The following commands are available to help maintain and use tapes. The *CL Reference* contains detailed descriptions of these commands.

Tape Configuration Description Commands:

CHGCTLTAP Change Controller Description for Tape: Changes a controller description for a tape controller.

CHGDEVTAP Change Device Description for Tape: Changes a device description for a tape device.

CRTCTLTAP Create Controller Description for Tape: Creates a controller description for a tape controller.

CRTDEVTAP Create Device Description for Tape: Creates a device description for a tape device.

DLTCTLD Delete Controller Description: Deletes a controller description.

DLTDEVD Delete Device Description: Deletes a device description.

DSPCTLD Display Controller Description: Displays a controller description.

DSPDEVD Display Device Description: Displays a device description.

Tape Device File Commands:

CHGTAPF Change Tape File: Changes certain attributes of a tape device file.

CRTTAPF Create Tape File: Creates a tape device file used to read and write records on tape.

DLTF Delete File: Deletes files.

DSPFD Display File Description: Displays the current characteristics of a file.

OVRTAPF Override with Tape File: Temporarily changes a tape file or tape file attributes specified in a program.

Other Tape Support Commands:

CHKTAP Check Tape: Searches a tape volume for a specific volume identifier or field label.

CPYFRMTAP Copy from Tape: Copies records from a tape file to an output file or a printer.

CPYTOTAP Copy to Tape: Copies records to a tape file from a physical, logical, tape, diskette, or spooled inline file.

DMPTAP Dump Tape: Dumps label information, data blocks, or both, from a tape with or without a label.

DSPTAP Display Tape: Displays volume and file label information, saved objects information, or the volume type (*NL or *LTM) and density for volumes without labels.

INZTAP Initialize Tape: Initializes tapes, with or without labels, or can be used to clear all data on the tape from the load point to the end-of-tape marker.

Tape Device Characteristics

Initializing Tapes

All tapes must be initialized before they can be used by the system. The Initialize Tape (INZTAP) command is used to initialize tapes, with or without labels, and to clear all data on the tape from the load point to the end-of-tape marker.

In the following example, a tape volume loaded on device TAP01 is initialized to standard label format.

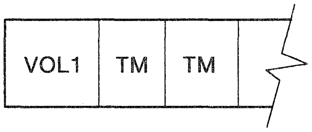
```
INZTAP  DEV(TAP01)
        NEWVOL(BACKUP)
        DENSITY(1600)
```

The tape volume will be initialized with the volume ID BACKUP. EBCDIC code is specified (by default) and the tape density is set at 1600 bits per inch.

Tape Labeling

The following series of diagrams provides a basic description of standard tape labeling used for the AS/400 system.

In the following diagram, the tape is given a volume label (marked VOL1). Two tape marks (TM) are written when the tape is initialized using the INZTAP command.

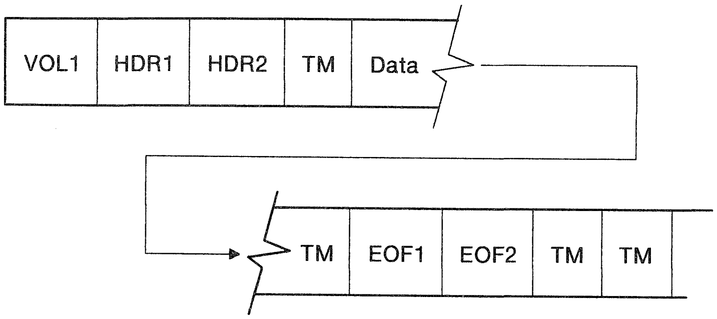


RSLH146-0

When the tape is used as an output device for a high-level language program, the two tape marks following the VOL label are written over with header labels (HDR1 and HDR2). Each header label is 80 bytes long, the first containing such information as the file name and date, the second specifying information such as record and block lengths, record block format, and buffer offset (for ASCII files). A single tape mark is added following the header labels.

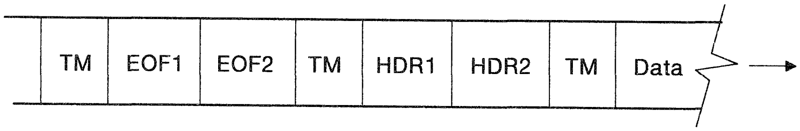
Data written to the tape is added following the tape mark. When the end of the file is reached, a tape mark and two end-of-file labels are written on the tape. The end-of-file labels contain the same information included in the header labels except that the second end-of-file label (EOF2) includes the nonzero block count for the file.

Two tape marks follow the end-of-file labels as shown in the following diagram.



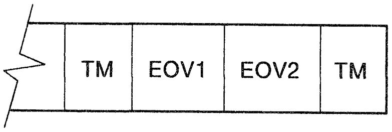
RSLH147-0

If a second file is added to the tape, the second tape mark following the end-of-file labels is written over by a header label (HDR1) for the new file, followed by the second header label, another tape mark, and the file data as shown in the following diagram.



RSLH148-0

When the end of the physical tape is reached, a tape mark and two end-of-volume labels are written, followed by a final tape mark. If the file is not complete, it will be continued on a second volume, starting with a volume label specifying the tape as volume 2 of the file.



RSLH149-0

Multivolume Tape Files

A single tape file may be contained on a single tape volume. You may also use:

- Multifile volumes: Volumes of tape containing more than one tape file.
- Multivolume files: Files contained on more than one reel (volume) of tape.

If you use multivolume files, the following conventions must be followed:

- The labels on each volume must be consistent. All volumes of a labeled multivolume file must be labeled; all volumes of an unlabeled multivolume file must be unlabeled.
- All volumes must be written in the same character code (EBCDIC or ASCII) and density.
- All sections of a particular data file (on all volumes that it spans) must have the same record format, block length, and record length.
- If more than one tape unit is specified, the volumes must be put on the devices in the sequence specified in the tape file. For example, if a multivolume file is contained on four volumes (VOL01, VOL02, VOL03, and VOL04) and the tape units specified, in order, are TAPE01, TAPE02, and TAPE03, the volumes must be placed on a tape drive as follows: VOL01 on TAPE01, VOL02 on TAPE02, VOL03 on TAPE03, and VOL04 on TAPE01.

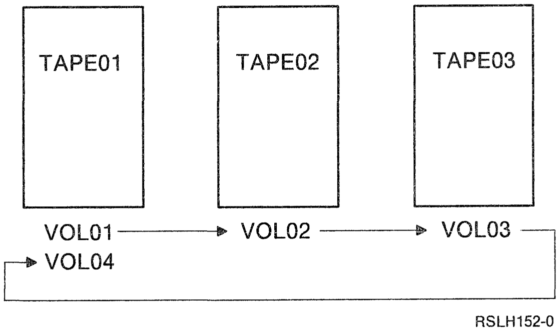


Figure 7-1. Multivolume Tape File Sequence Using Three Tape Devices

If volumes are used in reverse order to be read backward, VOL04 is on TAPE01, VOL03 on TAPE02, VOL02 on TAPE03, and VOL01 on TAPE01.

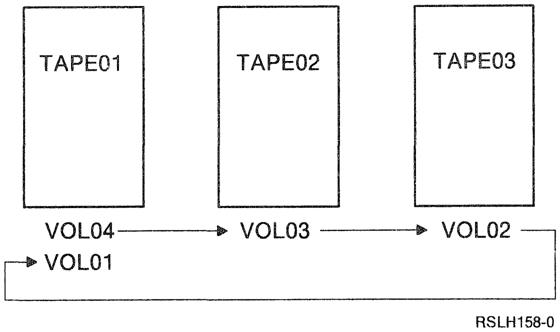


Figure 7-2. Read Backwards for Multivolume File Sequence

Extending Tape Files

Tape data files on 1/2-inch reel tapes can be extended using the EXTEND parameter on the CRTTAPF, CHGTAPF, and OVRTAPF commands. Extending tape files is not supported for cartridge tape devices.

When you create a new file on tape or extend a file, any previous tape data following the new data on the tape is no longer accessible using tape device support.

In the following example, a tape contains four files, FILE1, FILE2, FILE3, and FILE4. If FILE2 is extended, FILE3 and FILE4 will no longer be accessible, even though some data may actually remain on the tape following FILE2.

Note: In this example, if you specify EXTEND(*YES *CHECK) on the OVRTAPF command, the expiration date of the file following the file to be extended (FILE3) will be checked *before* extending the file (FILE2). However, the expiration dates of any remaining files (FILE4) are not checked, even if EXTEND(*YES *CHECK) is specified.

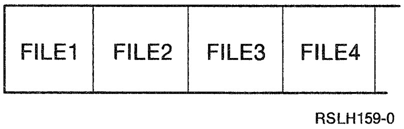


Figure 7-3. Tape Containing Four Files

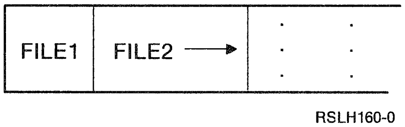


Figure 7-4. Tape after Extending FILE2

Tape Configuration Descriptions and Device Files

To access data from a tape device on the system, the following objects must exist:

- First, a device description must exist for each tape device to describe the device to the system. The tape device description is specified using the Create Device Description for Tape (CRTDEVTAP) command. The tape device description contains information such as device address, device name, device type, model number, and features. For some tape devices, a tape controller description must also exist.
- Second, a device file must exist for the tape device. Tape device files are created using the Create Tape File (CRTTAPF) command and describe how input data is to be presented to a program from a device, or how output data is to be presented to the device from a program. Tape device files must not be confused with the actual data files on the tape volumes: the tape device file provides a link between the application program and the tape device for processing a volume containing data files.

It is not necessary to have a separate device file for each tape device; you can use a single device file for several different tape devices using an Override Tape File

(OVRTAPF) command. Any number of device files can be associated with one device.

Note: The configuration descriptions must be varied on before they can be used. See the *Device Configuration Guide* for information about varying on configuration descriptions.

IBM-Supplied Tape Files

The following tape device files are shipped with the operating system for your use:

- QTAPE (tape file)
- QTAPSRC (tape source file)

These files are all program-described data files in library QGPL. The record format names are the same as the file names. The files contain default values for most parameters.

You can create additional tape device files to fit your needs. For example, you can create an additional tape device file to contain the specific volume and label information for a tape data file that can be used by several programs.

Example of Creating a Tape Device File

In the following example, a tape device file, TAP05 in library QGPL, is created for output that is written to tape:

```
CRTTAPF  FILE(QGPL/TAP05)
         DEV(TAP01)
         REELS(*SL)
         SEQNBR(3)
         CODE(*EBCDIC)
         ENDOPT(*UNLOAD)
```

The tape REELS parameter is specified with the value *SL, indicating that the tape uses standard labels. The device name is TAP01. The file is written at sequence number three on the tape (SEQNBR parameter), in EBCDIC code (CODE parameter), and will be unloaded after it has processed (ENDOPT parameter).

Specifying Tape Device File Parameters

Tape device file records are described in the application program that uses the tape information. The system views each record as one field with a length equal to the record length.

The following section lists considerations for parameters that may be specified on the CRTTAPF, CHGTAPF, and OVRTAPF commands.

Parameter **Description**

DEV The name of the device description identifying the tape devices that the tape file can access.

VOL The volume identifiers (from 1 through 6 alphameric characters) of the tapes to be used for the file may be specified using the VOL parameter on the CRTTAPF, CHGTAPF, and OVRTAPF commands.

REELS

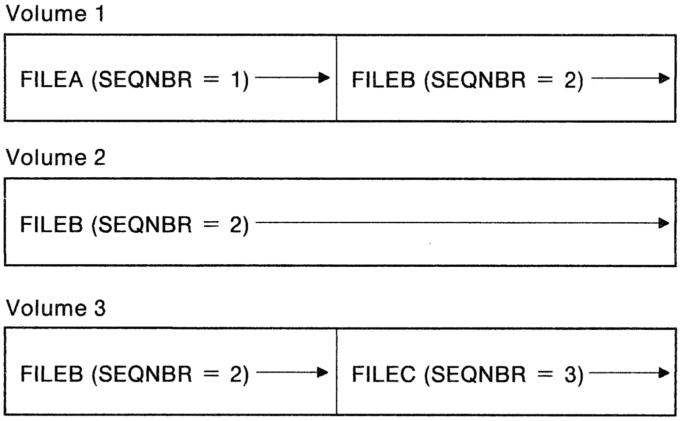
The type of label processing used for the tape on which the file is to reside and the number of tapes (reel number) that are to contain the file are specified using the REELS parameter. The reel number is ignored for output processing or if a volume list is specified. The reel number is also ignored if standard label processing is specified (using *SL on the REELS parameter). For cartridge tape devices, *SL must be specified for the REELS parameter.

If some of the file labels are incorrect, bypass label processing (*BLP) should be specified. Each reel is checked for a standard label; however, most other volume label information and the file labels on the tape are ignored. Refer to the *CL Reference* for more information on label processing.

For bypass label processing, each file on the tape must contain a header label and either an end-of-file or end-of-volume trailer label.

SEQNBR

The sequence number of the file on tape is specified by the SEQNBR parameter. The files are numbered consecutively across all the volumes they occupy, starting with sequence number 1 for the first file on the first volume, 2 for the second file, and so on. The following shows how files are numbered for labeled volumes containing more than one file and multivolume files (FILEB is contained on three volumes):



RSLH161-0

The sequence number specified for new standard labeled tape files (SEQNBR parameter on CRTTAPF, CHGTAPF, and OVRTAPF commands) must correspond to the physical sequence number of files on the tape. This means that if files 1 and 2 exist on the tape, the next file created must have a sequence number of 3. An exception to this rule occurs when a new file is created on a tape that contains the last volume of a multivolume file. In this case, the sequence number of the new file must be the sequence number of the last file on the multivolume tape plus 1. As shown in the previous diagram, the sequence number of FILEC must be 3.

The location of a file on tape is always specified on the SEQNBR parameter. The information specified on the LABEL parameter is used to verify that the correct file is to be processed only after the file is located using the information specified on the SEQNBR parameter. A tape file cannot be located by label name. If the Check Tape (CHKTAP) command is used, the sequence number of the file is returned in the completion message.

Some special values can be used in place of an actual sequence number:

- *NEXT: The next sequential file on the tape is processed. If the tape is currently positioned prior to the first file, the first file on the tape is processed. *NEXT is useful for applications that need to read all files on a tape. This value can only be specified for tape device files used to read from tape. An error message is issued when a tape device file is used to write to a tape and *NEXT is specified.
- *END: The file will be written to the end of the tape. This value can only be specified in tape device files used to write to tape. An error message is issued when a tape device file is used to read from a tape and *END is specified.

The SEQNBR parameter for an output file, for which EXTEND(*NO) is specified, must be one of the following:

- SEQNBR(1). This overwrites the first file on the volume, regardless of the sequence number in the labels of the first draft file already on the volume.
- A value 1 greater than the value for a file that already exists on the volume. This either overwrites an existing file on the volume or adds a file at the end of the volume.
- *END.

LABEL

The data file label on the tape is specified using the LABEL parameter.

The information specified on the LABEL parameter is used for new labels created for an output file for which EXTEND(*NO) is specified. It is also used for an input or output file for which EXTEND(*YES) is specified to verify that the correct file is processed.

RCDLEN

The length of records to be used by a program using this file is specified using the RCDLEN parameter. If *CALC is specified, the system attempts to calculate record length from the file header labels. The maximum record length is 32 767 bytes for fixed-length or undefined format records, and 32 759 for variable-length or spanned records. Fixed-length and undefined format output records cannot be less than 18 bytes in length.

BLKLEN

The length of the block of data to be transferred on each input or output operation is specified using the BLKLEN parameter. If *CALC is specified, the system attempts to calculate block length from the file header labels. The block length must be between 18 and 32 767 bytes.

RCDBLKFMT

The format of the input or output records and blocks is specified using the RCDBLKFMT parameter. Records can be:

- Fixed-length, unblocked (*F)
- Fixed-length, blocked (*FB)

- Variable-length, unblocked, unspanned (*V)
- Variable-length, blocked, unspanned (*VB)
- Variable-length, unblocked, spanned (*VS)
- Variable-length, blocked, spanned (*VBS)
- Undefined format (variable length) (*U)

The record length, block length, and record block format may not need to be specified for standard labeled input tape files or standard labeled output tape files with EXTEND(*YES) specified because this information is taken from the tape labels. If a block length or record block format is specified that does not match that specified in the tape label, the specification from the tape label is assumed. If the record length specified in the program does not match the length of the data, the data is padded or truncated to the length specified in the program.

EXTEND

New records may be added to the end of the data on the tape by specifying the EXTEND parameter. If the file is not the last file on the tape, all remaining files are destroyed. (This is also true when an existing file is written over.) The record and block length specified in the label are used for the extension. EXTEND is valid only for 1/2-inch reel tape devices.

By specifying EXTEND(*CHECK) on the OVRTAPF, the system will check the expiration date of the first file following the file to be extended.

DENSITY

The tape density expressed as the number of bits per inch (1600, 3200, 6250, or 10000 bits per inch) is specified using the DENSITY parameter. All files on a volume are recorded in the same density. The DENSITY parameter is used only to set the output volume density when the first file on a volume not labeled is created. The volume label on a labeled tape is used to determine the number of bits per inch.

CODE

The character code (EBCDIC or ASCII) for the data on tapes not labeled is specified using the CODE parameter. For standard label tapes, the volume label determines the character code. When ASCII is the interchange code, volume labels, header labels and data written by the system conform to "American National Standard" X3.27-1978, "Magnetic Tape and File Structure for Information Interchange."

CRTDATE

The creation date of an input data file on a labeled tape is specified using the CRTDATE parameter. If the creation date written on the tape does not match the date specified in the file description, a message is sent to the system operator, who determines what should be done.

EXPDATE

The expiration date of an output data file on a labeled tape is specified using the EXPDATE parameter. The data file cannot be written over until the date has expired. The file is considered to be protected.

When an output data file is created on a labeled volume (instead of extending an existing file), the expiration date of the new file is compared to the date of the file preceding it on the volume. If the expiration date of the new file is later than the expiration date of the file preceding it on the tape, an inquiry message (CPA4036) is sent. The system operator can:

- Allow the file to be created
- Put on a new tape and try again
- End processing

Note: Creating the file could produce a volume for which CHECK(*FIRST) on INZTAP is unreliable.

If you do not want the data file to be written over, specify *PERM on the EXPDATE parameter.

ENDOPT

When you use a multivolume data file and specify ENDOPT(*LEAVE), the first volume must be put on the first tape unit specified in the DEV parameter (except for a read backward, in which case the last volume must be put on the first tape unit specified). If a user opens the file again (using the same device list), and the tape is left on a tape unit other than the first one specified in the DEV parameter, the tape reel must be moved to the first tape unit specified in the DEV parameter before the next file on that tape reel is opened.

The ENDOPT parameter can be used to specify where magnetic tape is to be positioned when the tape file is closed. The magnetic tape can be rewound to the load point, left as it is, or unloaded.

USRLBLPGM

User header and trailer labels are supported through the use of the USRLBLPGM parameter. USRLBLPGM identifies the user program that is used during open and close processing. See "Processing User Labels" on page 7-15 for more information.

BUFOFSET

The buffer offset length for an ASCII file is specified using the BUFOFSET parameter. You can specify a buffer offset length for any ASCII input tape file. You can specify a buffer offset value of *BLKDSC for an input or output ASCII format *D or *DB file to process block with 4-digit block descriptors.

For additional tape information, and information about using tape for save and restore operations, see the *Backup and Recovery Guide*.

Table 7-1 lists parameters that apply to magnetic tape and where the parameters can be specified. Some parameters are valid only on DBCS systems. The *CL Reference* contains detailed information about how to specify these parameters on the CRTTAPF, CHGTAPF, and OVRTAPF commands.

Table 7-1. Tape Device File Parameters

CL Parameter	Description	Specified on CRTTAPF Command	Specified on OVRTAPF Command	Specified in HLL Programs
FILE	File name	Qualified file name	File name	RPG/400, COBOL/400, BASIC, PL/I, or C/400
DEV	Device name	*NONE or device name	Device name	
VOL	Volume	*NONE or volume identifier	*NONE or volume identifier	
REELS	Number of labeled tapes (reels)	1 or number of reels	1 or number of reels	
REELS	Volume label type (reels)	*SL, *NL, *NS, *BLP, or *LTM	*SL, *NL, *NS, *BLP, or *LTM	
SEQNBR	Sequence number	1, *NEXT, *END, or sequence number of file	*NEXT, *END or sequence number of file	
LABEL	Label	*NONE or file label	*NONE or file label	BASIC
FILETYPE	File type	*DATA or *SRC		
RCDLEN	Record length	*CALC or record length	*CALC or record length	RPG/400, COBOL/400, BASIC, PL/I, or C/400
BLKLEN	Block length	*CALC or block length	*CALC or block length	COBOL/400
BUFOFSET	Buffer offset	*BLKDSC or buffer offset	*BLKDSC or buffer offset	
RCDBLKFMT	Record block format	*F, *FB, *V, *VB, *D, *DB, *VS, *VBS, or *U	*F, *FB, *V, *VB, *D, *DB, *VS, *VBS, or *U	COBOL/400, C/400
EXTEND	Extend	*NO or *YES	*NO, *YES *CHECK, or *YES *NOCHECK	COBOL/400, C/400
DENSITY	Density	1600, 3200, 6250, or 10000		
CODE	Character code	*EBCDIC or *ASCII	*EBCDIC or *ASCII	COBOL/400
CRTDATE	Creation date	*NONE or date	*NONE or date	
EXPDATE	Expiration date	*NONE, date, or *PERM	*NONE, date, or *PERM	
ENDOPT	End option	*REWIND, *LEAVE or *UNLOAD	*REWIND, *LEAVE or *UNLOAD	COBOL/400
USRLBLPGM	User label program	*NONE or qualified program name	*NONE or qualified program name	
IGCDTA	Double-byte data	*NO or *YES	*NO or *YES	
WAITFILE	File wait time	*IMMED, *CLS, or number of seconds	*IMMED, *CLS, or number of seconds	
SHARE	Shared file	*NO or *YES	*NO or *YES	
AUT	Authority	*CHANGE, *ALL, *USE, *EXCLUDE, or authorization list name		
TEXT	Text	*BLANK or text		

Using Tape Files in High-Level Language Programs

A magnetic tape device can be accessed with a program-described device file. To use a tape file with a program, you must either specify the tape file name in the program or use an Override with Tape File (OVRTAPF) command. The high-level language you use determines what tape parameters can be specified in the program.

Open Processing for Tape

The following considerations apply to opening tape files:

- When a tape file is opened, any parameters specified in the file are merged (overridden) with the parameters specified in the program. These parameters are then merged (overridden) with the parameters specified on an OVRTAPF command, if specified.

- The device names must be specified when the tape file is opened. If DEV(*NONE) is specified in the tape file, one or more device names must be specified on an OVRTAPF command. You can specify as many as four device names for a single tape file (depending on how many magnetic tape units you have).

  The record length, block length, record block format, and buffer offset (for an ASCII file) specified for the file are always returned to the program in the data management open feedback area. They are returned in the format in which they are written in the HDR2 file header label. This information is available regardless of the type of label processing used for the file.

- Read backwards is supported for fixed (*F), fixed block (*FB), and undefined format (*U) tape files and for both multifile volumes and multivolume files. It can be requested only through a high-level language when the file is opened. An escape message is signaled if read backwards is attempted for variable-length (spanned or not spanned) or source records.

  Note: Cartridge tape devices do not have read backwards capabilities.

  If a device and volume list are specified for a file that is to be read backwards, volume positioning on devices is the same as though the file were to be read forward. For example, a file with DEV(QTAPE1 QTAPE2) VOL(VOL01 VOL02 VOL03) expects VOL03 on QTAPE1, VOL02 on QTAPE2, and then VOL01 on QTAPE1.

  For read backwards, end of file occurs for standard label processing (*SL) or bypass label processing (*BLP) if the system can recognize the first volume of the file from the header labels. If the header labels are not recognized for the first volume of the file or if this is a *BLP file, end of file is signaled when the number of reels specified on the REELS parameter or the number of identifiers on the VOL parameter has been processed.

- Some high-level languages allow you to specify where the tape is to be positioned when an input tape file is opened. This indicates if the tape is to be processed in the forward or the backward direction.

  The rules for determining the first volume of a file from the labels are:

  – HDR1 label multivolume sequence field = 1 (ASCII or EBCDIC with no HDR2 label)

    or

  – HDR2 label volume switch indicator field = 0 (EBCDIC)

- The record length must be specified according to the information shown in Table 7-2 on page 7-13.

Table 7-2. Specifying Record Lengths by Record and Format Type

Record and Format Type	Minimum Record Length for *DATA	Minimum Record Length for *SRC	Maximum Record Length for *DATA	Maximum Record Length for *SRC	Block Length
Fixed blocked, *F, *FB, *U	18	30	32 767	32 767	Multiple of *DATA record length
Variable unblocked, *V	1	13	32 759 (See Note)	32 767	Equal to maximum *DATA record length plus 8
D-type ASCII unblocked, *D	1	13	9 995 (See Note)	10 007 (See Note)	Equal to maximum *DATA record length plus 4, plus buffer offset
Variable blocked, *VB	1	13	32 759	32 767	At least maximum *DATA record length plus 8
D-type ASCII blocked, *DB	1	13	9 995 (See Note)	10 007 (See Note)	At least maximum *DATA record length plus 4, plus buffer offset
*VS, *VBS	1	13	32 759	32 759	

Note: This is the maximum record length for a record being written to a tape. Input records can be padded to 32 767.

- For source files, the record length used to determine the block length is the actual data length, not the data length plus 12 bytes (for sequence number and date).

- You must supply either a RCDLEN or BLKLEN parameter value for unspanned, unblocked records (*F, *V, *D, *U).

- You must supply both RCDLEN and BLKLEN parameter values for spanned or blocked records (*FB, *VB, *DB, *VS, *VBS).

- When the file type specified in the tape file is a source file, a date and sequence number are appended to each record on input operations and are removed on output operations. (The date field is always 0.) The program can check (if the high-level language you are using allows it) to determine if the input or output file is a source file. The record length must include 12 bytes for the date and sequence number. The block length and record length for a source file have the same ratio as the block length and record length for data records minus the 12 bytes needed for source files. For example, if the actual type data record length is 80 and block length is 800, to use the file as a source file the record length becomes 92 and the block length is unchanged at 800.

- For standard labeled, input file processing, the block length in the file label is used regardless of what is specified when the file is opened or what is specified in the device file for block length.

- Variable-length (spanned or unspanned) records and undefined format records can be used for output files. If your high-level language does not support variable-length records, then all records for a variable format, output tape file are maximum length.

- The sequence number *must* be specified so the file can be found. Tape files cannot be located by label name.

- If both the VOL and REELS parameters are specified, the volume list is used to control the volumes processed by the file, even if the REELS parameter is specified in a previous override (at a higher call level than the oldest VOL parameter specification). If you want to use the REELS parameter (number of reels) to limit the number of input volumes processed, specify *NONE for the VOL parameter.

Input/Output Processing for Tape

The following considerations apply to I/O operations performed on tape files:

Read and Write Considerations:

- Record length must be specified in your program when writing variable-length records (*D, *DB, *V, *VB, *VS, *VBS, or *U specified on the RCDBLKFMT parameter on the CRTTAPF command).

  If the maximum record length on tape is shorter than the output record length (because of overrides or existing file labels), the records are truncated to the maximum length allowed. A diagnostic message is sent when the file is opened to indicate that output records may be truncated.

- If the record length specified by the program differs from the actual length of the data, the data is padded or truncated as necessary to match the program specification.

Read Considerations:

- If no end-of-file label is found, tapes are processed until the specified volume identifiers have been used. If VOL(*NONE) is specified, tapes are processed until the specified number of reels (REELS parameter) is reached. Message CPA5230 is sent to the operator message queue when all the identifiers in the VOL list have been used. When you receive this message, you can:

  - End the device support function immediately: The file is closed.
  - Continue: Other volumes are processed.

- When an invalid length tape block is read, a CPF5036 notify message is sent to the program. If your high-level language reports this condition to your program, it can continue processing by attempting to read another record. When you continue processing this way, the invalid block is skipped so your program does not receive any records from the invalid block.

Force-End-of-Data Considerations: The force-end-of-data function is valid for both input and output. The force-end-of-data function for an output file forces all buffered records to be written to tape. When this occurs, a short block may be written on the volume. The force-end-of-data function for an input file positions the tape at the last volume for the file and signals end-of-file to the using program.

Force-End-of-Volume Considerations: The force-end-of-volume function is valid for both input and output files. It causes volumes to be switched immediately, or signals end-of-file if there are no more continuation volumes for an input file.

Close Processing for Tape

When a tape file is closed, one of the following functions is performed according to what you specify in the tape file, using the ENDOPT parameter on a CRTTAPF, CHGTAPF, or an OVRTAPF command.

- The tape is rewound to its load point.
- The tape is left as it is.
- The tape is rewound and unloaded so that it can be removed from the magnetic tape unit.

If the tape operation ends abnormally, the position of the tape when the file is closed may be unchanged, or the tape may be rewound regardless of what has been specified.

Handling Tape Processing Errors and Damaged Tapes

Tape data files can be damaged when, due to an error condition or a system failure, no tape marks or labels are written at the end of the data file on the tape.

If this happens using a 1/2-inch reel tape device, the following will occur when you try to read the data file:

- The new and existing data may appear to be concatenated when processed for input. If the tape is labeled, an error message may be sent to the system operator when the trailer labels are read. No error will be detected for unlabeled tape.
- If the new data and the existing data do not appear to be concatenated, an error message is sent to the system operator.
- If the tape contains no existing data or tape marks beyond the location of the last data block (the tape may be new or completely deleted), the tape advances until it runs off the end of the reel when used as input.

Note: Whenever an output file is closed, the system attempts to write an end-of-tape marker and label at the end of the file even if a previous error that signaled an escape message was reported. If a file close is attempted and the system cannot write closing tape marks and labels, a message is sent to your job log.

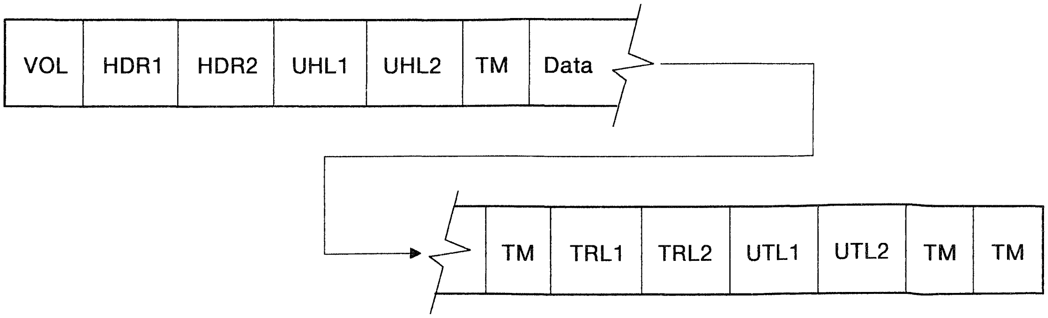
If damage occurs while using a cartridge tape device, blank tape will be encountered and an error message will be sent to the system operator.

Processing User Labels

The USRLBLPGM parameter can be used to specify the name of a program used to process user header and trailer labels. This parameter is not valid for save and restore functions.

The system calls the program specified on the USRLBLPGM parameter during open and close processing for every label processed, plus one additional time to indicate when no more labels are being passed.

The following diagram shows the format of a tape with user labels. A user label program used to process the labels in the diagram would be called three times for open processing (for UHL1, UHL2, and a final time to indicate no more labels being passed). The program would be called three more times for close processing.



RSLH162-0

Three variables are passed to the program by the system. The variables have the following lengths:

- Parameter 1: 80 characters
- Parameter 2: 1 character
- Parameter 3: 244 characters

The following list shows the content of the variables passed to the user label program:

Parameter 1

Position	Description
1 - 80	User header or trailer label

- For output files, this parameter indicates the next user label to be written, set by the user label program.
- For input files, this parameter indicates the user label most recently read from the tape, set by the system.

Parameter 2

Position	Description
1	End-of-labels indication

Parameter 2 contains a character 0 or 1, indicating whether or not the label read is the last label to be read. For output files, the value is set by the user label program; for input files, the value is set by the system.

- 0 indicates that Parameter 1 contains a label.
- 1 indicates that Parameter 1 does not contain a label. All labels have been processed.

Parameter 3

Position **Description**

1- 80 Current volume label

81-160 Last processed HDR1 or TRL1 label, whichever was encountered last

161-240 Last processed HDR2 or TRL2 label, whichever was encountered last

241-242 User label number

- Output files: The number of the next user label to be written in the current header or trailer group.
- Input files: The count of the number of user labels read in the current header or trailer group.

243 Open file option

The open file option field contains a character indicating whether the file was opened as an input or output file.

- I indicates that the file was opened as an input file.
- 0 indicates that the file was opened as an output file.

244 Expect labels

The expect labels field contains an integer indicating whether the call is returning or expecting labels from the user program.

- 0 indicates that the user program is returning labels.
- 1 indicates that the user program is expecting labels.

Other Tape Support Commands

Several tape support functions are provided by the following CL commands.

CHKTAP Check Tape: Searches a tape volume for a specific volume identifier or field label.

The following command checks a tape volume loaded on device TAP01 to see if a file named TEST is located at sequence number 3.

```
CHKTAP  DEV(TAP01) SEQNBR(3) LABEL(TEST)
```

DMPTAP Dump Tape: Dumps label information, data blocks, or both, from a tape with or without a label.

The following command dumps the second file, including all label information, from a tape volume loaded on device TAP02.

```
DMPTAP  DEV(TAP02) SEQNBR(2) TYPE(*ALL)
```

DSPTAP Display Tape: Displays volume and file label information, saved objects information, or the volume type (*NL or *LTM) and density for volumes without labels.

The following command displays the volume and file label information of a tape volume loaded on device TAP01.

```
DSPTAP  DEV(TAP01)
```

Performance Considerations for Tape

The 6346 and 9346 tape devices are intended primarily for save and restore operations. These devices are not designed for operations that cannot process data fast enough to keep the tape moving. Performance and reliability will suffer if the tape stops and starts too often.

If you use the 6346 or 9346 devices for operations other than save and restore, the tape will more likely keep moving when you do any of the following:

- Process large tape blocks (BLKLEN parameter). If the records are small, use a blocked format (specify *FB, *DB, *VB, or *VBS on the RCDBLKFMT parameter).
- Use fixed-length records, because they are processed more efficiently than variable-length records.
- Design the application to do as little processing as possible between tape read or write operations. The best application design may be one that uses a system copy file command (CPYF, CPYFRMTP, or CPYTOTAP) to copy records between tape and a database file, with application programs that only process records in the database file.

Chapter 8. Diskette Support

Diskettes can be used to create backup copies of information, provide offline storage of files and libraries, or to transfer information to other systems or devices.

The AS/400 system supports both 5 1/4-inch and 8-inch diskettes, with the following restrictions:

- 5 1/4-inch diskettes must be double-sided, high capacity (2HC) diskettes. These diskettes have a capacity equivalent to 8-inch, double density diskettes. When initializing these diskettes, use 2D format and sector sizes 256, 512, or 1024.
- 8-inch diskettes can be any of the following three types:
  - Diskette 1 is a single-sided, single-density diskette.
  - Diskette 2 is a double-sided, single-density diskette.
  - Diskette 2D is a double-sided, double-density diskette.

Double-density means that a diskette contains information on both sides and with two times the amount of information stored in the same space as a diskette 1. A diskette 2D holds approximately four times the amount of information as a diskette 1.

For information about how to use the diskette devices for save/restore operations, see the *Backup and Recovery Guide*.

Related CL Commands

The following commands are available to help maintain and use diskettes. The *CL Reference* contains detailed descriptions of these commands.

Diskette Device Description Commands:

CHGDEVDKT Change Device Description for Diskette: Changes a device description for a diskette device.

CRTDEVDKT Create Device Description for Diskette: Creates a device description for a diskette device.

DLTDEVD Delete Device Description: Deletes a device description.

Diskette Device File Commands:

CHGDKTF Change Diskette File: Changes certain attributes of a diskette device file.

CRTDKTF Create Diskette File: Creates a diskette device file used to read and write records on diskette.

DLTF Delete File: Deletes files.

DSPFD Display File Description: Displays the current characteristics of a file.

OVRDKTF Override with Diskette File: Overrides a diskette file or file attributes specified in a program.

Other Diskette Support Commands:

CHKDKT Check Diskette: Checks a diskette for a specific volume identifier, file label, or both.

CLRDKT Clear Diskette: Clears all files on the diskette by deleting the labels and creates an expired file, labeled DATA, that includes the entire diskette. The data is not deleted.

CPYFRMDKT Copy from Diskette: Copies records from a diskette file to an output file or a printer.

CPYTODKT Copy to Diskette: Copies records to a diskette file from a physical, logical, diskette, or spooled inline file.

DLTDKTLBL Delete Diskette Label: Deletes a file label from a diskette.

DSPDKT Display Diskette: Displays the volume and file labels, or save and restore information on a diskette.

DUPDKT Duplicate Diskette: Copies information from one diskette to one or more diskettes.

INZDKT Initialize Diskette: Initializes a diskette, to write identification information and to format the diskette for use by the system.

RNMDKT Rename Diskette: Changes the volume and owner identifiers in the volume label.

Diskette Device Characteristics

Diskette Exchange Types

Diskette data must use one of the following exchange types, and the diskettes must have the attributes (diskette type, sector size, and record length) appropriate for that exchange type. The exchange types are:

- Basic Exchange

  Basic exchange data sets may be exchanged between systems capable of reading and writing both the IBM Diskette 1 and the IBM Diskette 2. The sector size must be 128 bytes. The length of the records can be from 1 to 128 bytes, with one record per sector.

- H Exchange

  Type H exchange data sets may be exchanged between systems capable of reading and writing the IBM Diskette 2D. The sector size must be 256 bytes. The length of the records can be from 1 to 256 bytes, with one record per sector.

- E Exchange

  Type E exchange data sets force the using system to examine each field in the header label. E exchange data sets are created by save/restore operations.

- I Exchange

  Type I exchange data sets may be used for:

  - Diskette 1 or Diskette 2 with sector sizes of 128, 256, or 512 bytes
  - Diskette 2D with sector sizes of 256, 512, or 1024 bytes

The record length can be from 1 to 4096 bytes. One, more than one, or part of a record may be contained in one sector, or one record may span across sectors, depending on sector size and record length.

For these exchange types, each record in the file is fixed-length (each record contains the same number of bytes).

Initializing Diskettes

Before you can use a diskette on the system it must be initialized and have a volume label written on it. Diskettes are normally initialized and ready for use when you receive them.

The Initialize Diskette (INZDKT) command initializes a diskette to a specified format. You may want to use the INZDKT command to reinitialize a diskette in the following situations:

- If you want to change the sector size on a diskette to a size that is acceptable for a specified exchange type (basic, H, or I) or to accommodate save/restore.
- If you are encountering errors while reading or writing a diskette. Initializing the diskette may correct the errors or assign an alternate cylinder to be used in place of a defective cylinder. If you encounter more than two defective cylinders during the process of preparing, you will receive a message stating that the diskette is unusable. If you receive this message, you should discard the diskette.

Initializing a diskette will delete the data contained on the diskette. If you want to save the data, you must do so before initializing the diskette. You can use the Duplicate Diskette (DUPDKT) command to copy the entire diskette to another diskette, or the Copy File (CPYF) command to copy each file on the diskette to a database file or to another media. You may also use the CPYFRMDKT and CPYTODKT commands to copy data from or to a diskette.

Multivolume Diskette Files

Depending on the size of your data files, you may place several files on one diskette, called a multifile volume. If a single file occupies more than one volume, it is called a multivolume file.

If you use multivolume files, the following conventions must be followed:

- All volumes must be contained on the same format (1, 2, or 2D) of diskette that contains the first volume of the file.
- All volumes must have the same sector size (128, 256, 512, or 1024 bytes).
- All volumes must be written in the same character code (EBCDIC or ASCII).
- For an input file, the file exchange types must be the same on each volume of the file.
- For an input file, the length of the records in the file must be the same on each volume of the file.
- Except for the first volume, multivolume files cannot be written to diskettes that contain active files. (An active file has an expiration date greater than the system date.) If you attempt to do this, a message is sent to the system operator that allows the active files to be ignored (written over).

- A file cannot be written to a diskette that has an extended label area. (Extended label areas – cylinders that have been allocated to contain additional data set labels – are not supported by the system.)

When you use a diskette for spooled output, make sure the output is written to the right diskette. You can do this by specifying the VOL parameter when the output is spooled. The advantage of spooling output is that more than one job can be running at the same time while output is produced.

Note: The volume sequence number field in the diskette file label for multivolume files has only two positions. Volume 100 is indicated by 00, after which the numbers return (wrap around) to 01. An informational message is sent to the system operator each time the numbers return to 01.

To use the diskette device for other than saving and restoring, the following conventions must be observed:

- Each volume identifier can be from 1 through 6 alphameric characters.
- If you have multivolume files, a volume identifier can apply to more than one volume. A multivolume file occupies more than one volume (diskette). However, you are not required to use the same volume identifier for every volume in a multivolume file.
- Files are not written to diskettes with extended label areas.

Diskette Device Descriptions and Device Files

To access data from a diskette device on the system, two objects must exist:

- First, a device description must exist for each diskette device to describe the device to the system. The diskette device description is created using the Create Device Description for Diskette (CRTDEVDKT) command. The diskette device description contains information such as device address, device name, device type, model number, and features.
- Second, a device file must exist for the diskette device. The device file describes how input data is presented to a program from the device, or how output data is to be presented to the device from a program. Diskette device files must not be confused with the actual data files on the diskette volumes: the diskette device file provides a link between the application program and the diskette device for processing the diskettes containing the data files.

Diskette files must be program-described, meaning that fields and records are described in the program that processes the file, rather than in the file itself. It is not necessary to have a separate device file for each diskette device; you can use a single device file for several different diskette devices by using an override command. Any number of diskette files can be associated with one diskette device.

Note: The device must be varied on before it can be used. This function can be performed automatically when the system is started or you can use the appropriate vary command. See the *Device Configuration Guide* for information about varying on device descriptions.

Diskette Device Files

IBM-Supplied Diskette Files

The following diskette files are shipped with the operating system for your use:

- QDKT (diskette file)
- QDKTSRC (diskette source file)

These files are all program-described data files in library QGPL. The record format names are the same as the file names. The files contain default values for most parameters.

Note: The device (DEV) for these diskette device files uses the default value *NONE. You must use the CHGDKTF or OVRDKTF commands to specify a diskette device before using these files.

Example of Creating a Diskette Device File

You can create additional device files to fit your needs. For example, you can create additional device files to:

- Direct output to a special output queue
- Contain the specific volume and label information for a diskette file that can be used by several programs
- Spool output or not spool output

In the following example, a diskette device file, DKFILE, is created for output that is written to diskette.

```
CRTDKTF  FILE(DKFILE) DEV(DKT01) CODE(*ASCII)
```

The diskette device (specified on the DEV parameter) is DKT01, and the diskette data file will be written in ASCII code. All other parameters on the CRTDKTF command use their default values.

Because the default values were assumed in this example, the diskette volume, file label, and the creation date of the data file on the diskette must be specified in another CL command or in each program that uses the device file.

Specifying Diskette Device File Parameters

In diskette device files, the data in each record is described in the application program. The system views each record as one field with a length equal to the record length.

The following diskette file attributes may be specified for the CRTDKTF, CHGDKTF, or OVRDKT commands:

- Spooling information for output files, including:
  - Output queue name
  - Maximum number of records that can be spooled for the file
  - When output is made available to a writer program (at job end, at file end, or immediately as it is spooled)
  - Whether the output is to be held on the output queue until the system operator releases it
  - Whether the output is to be saved after it is produced

  Note: If you do not spool output, you must ensure that the device which the output is to be written to is assigned to your device file.

- The device with which the file can be used. You can specify a device name on the create device file command. The device name can also be specified using the OVRDKTF or CHGDKTF command.
- The wait time for file allocation. The number of seconds the system is to wait for the file resources to be allocated when the file is opened.
- Whether or not the open data path (ODP) for the file can be shared.
- The volume identifiers of the diskettes to be used for the file (specifying VOL(*NONE) causes volume checking to be ignored).
- The data file label on the diskette.
- The character code (EBCDIC or ASCII) for the data on the diskettes.
- The creation date of an input data file. If the creation date written on the diskette does not match the date specified in the diskette file, a message is sent to the system operator, who determines what should be done. The date must be specified in the format specified in the system value QDATFMT. However, the specified date is put in the diskette label in the format, yymmdd.
- The exchange type (basic, H, or I) to be used when creating an output file on the diskette. This attribute is not used when processing an input file.
- The expiration date of an output data file on diskette. The expiration date means that the data file cannot be written over until the date has expired. The file is considered to be protected. The date must be specified in the format specified in the system value QDATFMT. However, the specified date is stored in the diskette label as yymmdd.

  If you do not want the data file to expire, specify *PERM on the EXPDATE parameter.

Table 8-1 lists the parameters that apply to diskettes and where the parameters can be specified. The *CL Reference* contains detailed information about how to specify these parameters on the CRTDKTF, CHGDKTF and OVRDKTF commands.

Table 8-1. Diskette File Parameters

CL Parameter	Description	Specified on CRTDKTF Command	Specified on OVRDKTF Command	Specified in HLL Programs
FILE	File name	Qualified device file name	Device file name	RPG/400, COBOL/400, BASIC, PL/I, and C/400
DEV	Device name	*NONE or device name	Device name	
VOL	Volume	*NONE or volume identifier	*NONE or volume identifier	
LABEL	Label	*NONE or file label	File label	PL/I
FILETYPE	File type	*SRC or *DATA		
EXCHTYPE	Exchange type	*STD, *BASIC, *H, or *I	*STD, *BASIC, *H, or *I	
CODE	Code	*EBCDIC or *ASCII	*EBCDIC or *ASCII	COBOL/400
CRTDATE	Creation date	*NONE or date	*NONE or date	
EXPDATE	Expiration date	*NONE, *PERM, or date	*NONE, *PERM, or date	
SPOOL	Spool data	*YES or *NO	*YES or *NO	
OUTQ	Output queue	Qualified name	Qualified name	
MAXRCDS	Maximum records	*NOMAX or maximum records	*NOMAX or maximum records	
SCHEDULE	Schedule	*FILEEND, *JOBEND or *IMMED	*FILEEND, *JOBEND or *IMMED	
HOLD	Hold	*NO or *YES	*NO or *YES	
SAVE	Save	*NO or *YES	*NO or *YES	
OUTPTY	Output priority	*JOB or output priority	*JOB or output priority	
USRDTA	User data	*BLANK or user data	*BLANK or user data	
WAITFILE	File wait time	*IMMED, *CLS or number of seconds	*IMMED, *CLS or number of seconds	
SHARE	Shared file	*YES or *NO	*YES or *NO	PL/I
AUT	Authority	*CHANGE, *ALL, *USE, *EXCLUDE, or authorization list name		
TEXT	Text	*BLANK or text		
	Record length			RPG/400, COBOL/400, BASIC, PL/I, and C/400

Using Diskette Files in High-Level Language Programs

A diskette device can be accessed from a program using a program-described diskette device file. To use the diskette file with a program, you must either specify the diskette file name in the program or use an Override with Diskette File (OVRDKTF) command.

To use the diskette device directly, you must specify the device name and the SPOOL(*NO) parameter. The high-level language you use determines what diskette parameters can be specified in the program. If the parameters are not specified in the file description or program, they can be specified in an OVRDKTF command.

Open Processing for Diskette

The following considerations apply to opening diskette files:

- When a diskette file is opened, the parameters specified in the file are merged (overridden) with the parameters specified in the program. These parameters are then merged (overridden) with the parameters specified on an OVRDKTF command, if specified.
- The device name must be specified when the diskette file is opened if the file is not to be spooled; that is, SPOOL(*NO) is specified. If DEV(*NONE) is specified in the diskette file, the device name must be specified on an OVRDKTF command.
- For an input file, the program may specify the record length, but it is not a requirement. If record length is not specified or is specified with a length of 0, the system will determine the record length from the data file label on the diskette. If the program specifies a record length that is not equal to the length of the records in the data file, the records are padded or truncated to the length specified in the program. A diagnostic message is sent stating that the records are padded if the program record length is greater than the length of the records in the data file.
- For output files, record length must be specified in the program. When the record length specified in the program exceeds that for which the diskette is formatted, a diagnostic message is sent to your program, and the records are truncated.

  Maximum record lengths supported by exchange types are:

Basic exchange	128 bytes
H exchange	256 bytes
I exchange	4096 bytes

- The file label name is required when the file is opened. If the diskette file was created with LABEL(*NONE) specified, the label must be supplied with the OVRDKTF command. The label name must not exceed 8 characters for files in exchange type Basic, H, or I. The first character must be alphabetic (A through Z, #, $, or @). The rest of the characters can be any character (A through Z, 0 through 9, #, $, or @) except a blank.

  A file will not be written with an invalid label name. If an invalid label name is encountered for an input file, a diagnostic message is sent and an attempt is made to read the file.
- When the file type specified in the diskette file is a source file, a date and sequence number are appended to each record on input operations and are removed on output operations. (The date field is always 0.) The program can

specify (if the high-level language you are using allows it) that the system should check to determine if the input or output file is a source file. The record length specified in a program that uses a source file must include 12 bytes for source data. For example, if the data is 80 bytes long, a record length of 92 must be specified in the program.

- A volume identifier specifies the diskette to process. If no volume identifier is specified, the diskette is processed without checking the volume identifier. When a volume identifier is specified, the correct volume must be loaded. If not, a message that requests that the correct diskette be put in is sent to the system operator.
- For multivolume files, all volumes must be on the same type of diskette, have the same sector size, have the same record length, and must be written in the same character code. For output files, all volumes after the first volume must be written to diskettes that do not contain active files. If active files exist, a message is sent to the system operator that allows the active files to be ignored (overwritten).
- When an output file is created, all expired files on a diskette are written over; no messages are sent to your program. An expired file is a file having an end date less than or equal to the system date.

I/O Processing

The following considerations apply to input/output (I/O) operations performed on diskette files:

Read and Write Considerations

- When a volume switch occurs while processing multivolume files, a message identifying the current volume identifier is sent to your program.
- If the record length specified in the program does not match the length of the data, the data is padded or truncated to match the record length specified in the program.

Force-End-of-Data Considerations

- The force-end-of-data function is valid for both input and output.
- The force-end-of-data function for an output file does not write any data on diskette.
- The force-end-of-data function for an input file positions the diskette at the last volume of the file and signals end-of-file to the program using the file. See Chapter 6, “Spool Support” for information about how this function is handled for a SPOOL(*YES) output file.

Close Processing

When a diskette file is closed, the labels for output data are written on the diskette before the file resources are deallocated.

Handling Diskette Errors

If you encounter an error on the input diskette while using the DUPDKT command, the operation is ended without any data being written to the output diskette.

If you encounter an error on the input diskette while using the CPYF, CPYTODKT, or CPYFRMDKT commands, all the data in the file before the record that caused the error is copied. The remaining data in the file is not copied.

Part 4. Appendixes

Appendix A. Feedback Area Layouts

Tables in this section describe the open and I/O feedback areas associated with any opened file. The following information is presented for each item in these feedback areas:

- Offset, which is the number of bytes from the start of the feedback area to the location of each item.
- Data Type.
- Length, which is given in number of bytes.
- Contents, which is the description of the item and the valid values for it.
- File type, which is an indication of what file types each item is valid for.

The support provided by the high-level language you are using determines how to access this information and how the data types are represented. See your high-level language manual for more information.

Open Feedback Area

The open feedback area is the part of the open data path (ODP) that contains general information about the file after it has been opened. It also contains file-specific information, depending on the file type, plus information about each device or communications session defined for the file. This information is set during open processing and may be updated as other operations are performed.

Table A-1 (Page 1 of 5). Open Feedback Area

Offset	Data Type	Length	Contents	File Type
0	Character	2	Open data path (ODP) type: DS Display, tape, ICF, save, printer file not being spooled, or diskette file not being spooled. DB Database member. SP Printer or diskette file being spooled or inline data file.	All
2	Character	10	Name of the file being opened. If the ODP type is DS, this is the name of the device file or save file. If the ODP type is SP, this is the name of the device file or the inline data file. If the ODP type is DB, this is the name of the database file that the member belongs to.	All
12	Character	10	Name of the library containing the file. For an inline data file, the value is *N.	All
22	Character	10	Name of the spooled file. The name of a database file containing the spooled input or output records.	Printer or diskette being spooled or inline data
32	Character	10	Name of the library in which the spooled file is located.	Printer or diskette being spooled or inline data
42	Binary	2	Spooled file number.	Printer or diskette being spooled
44	Binary	2	Maximum record length.	All

Table A-1 (Page 2 of 5). Open Feedback Area

Offset	Data Type	Length	Contents	File Type
46	Character	2	Reserved.	
48	Character	10	Member name: • If ODP type DB, the member name in the file named at offset 2. If file is overridden to MBR(*ALL), the member name that supplied the last record. • If ODP type SP, the member name in the file named at offset 22.	Database, printer, diskette, and inline data
58	Binary	4	Reserved.	
62	Binary	4	Reserved.	
66	Binary	2	File type: 1 Display 2 Printer 4 Diskette 5 Tape 9 Save 11 ICF	Display, printer, diskette, tape, save, and ICF
68	Character	3	Reserved.	
71	Binary	2	Number of lines on a display screen or number of lines on a printed page.	Display, printer
73	Binary	2	Number of positions on a display screen or number of characters on a printed line.	Display, printer
75	Binary	4	Number of records in the member at open time. For a joined file, the number of records in the primary. Supplied only if the file is being opened for input.	Database, inline data
79	Character	2	Access type: AR Arrival sequence. KC Keyed with duplicate keys allowed. Duplicate keys are accessed in first-changed-first-out (FCFO) order. KF Keyed with duplicate keys allowed. Duplicate keys are accessed in first-in-first-out (FIFO) order. KL Keyed with duplicate keys allowed. Duplicate keys are accessed in last-in-first-out (LIFO) order. KN Keyed with duplicate keys allowed. The order in which duplicate keys are accessed can be one of the following: • First-in-first-out (FIFO) • Last-in-first-out (LIFO) • First-changed-first-out (FCFO) KU Keyed, unique.	Database

Table A-1 (Page 3 of 5). Open Feedback Area

Offset	Data Type	Length	Contents	File Type
81	Character	1	Duplicate key indication. Set only if the access path is KC, KF, KL, KN, or KU: D Duplicate keys allowed if the access path is KF or KL. U Duplicate keys are not allowed; all keys are unique and the access path is KU.	Database
82	Character	1	Source file indication. Y File is a source file. N File is not a source file.	Database, tape, diskette, and inline data
83	Character	10	Reserved.	
93	Character	10	Reserved.	
103	Binary	2	Offset to volume label fields of open feedback area.	Diskette, tape
105	Binary	2	Maximum number of records that can be read or written in a block when using blocked record I/O.	All
107	Binary	2	Overflow line number.	Printer
109	Binary	2	Blocked record I/O record increment. Number of bytes that must be added to the start of each record in a block to address the next record in the block.	All
111	Binary	4	Reserved.	
115	Character	1	Miscellaneous flags. Bit 1: Reserved.	
			Bit 2: File shareable 0 File was not opened shareable. 1 File was opened shareable (SHARE(*YES)).	All
			Bit 3: Commitment control 0 File is not under commitment control. 1 File is under commitment control.	Database
			Bit 4: Commitment lock level 0 Only changed records are locked. 1 All records accessed are locked.	Database
			Bit 5: Member type 0 Member is a physical file member. 1 Member is a logical file member.	Database

Table A-1 (Page 4 of 5). Open Feedback Area

Offset	Data Type	Length	Contents	File Type
			Bit 6: Field-level descriptions 0 File does not contain field-level descriptions. 1 File contains field-level descriptions.	All, except database
			Bit 7: DBCS-capable file 0 File is not DBCS-capable. 1 File is DBCS-capable.	Database, display, printer, tape, diskette, and ICF
			Bit 8: End-of-file delay 0 End-of-file delay processing is not being done. 1 End-of-file delay processing is being done.	Database
116	Character	10	Name of the requester device. For display files, this is the name of the display device description that is the requester device. For ICF files, this is the program device name associated with the remote location of *REQUESTER. This field is supplied only when either a device or remote location name of *REQUESTER is being attached to the file by an open or acquire operation. Otherwise, this field contains *N.	Display, ICF
126	Binary	2	File open count. If the file has not been opened shareable, this field contains a 1. If the file has been opened shareable, this field contains the number of programs currently attached to this file.	All
128	Binary	2	Reserved.	
130	Binary	2	Number of based-on physical members opened. For logical members, this is the number of physical members over which the logical member was opened. For physical members, this field is always set to 1.	Database
132	Character	1	Miscellaneous flags.	
			Bit 1: Multiple member processing 0 Only the member specified will be processed. 1 All members will be processed.	Database
			Bit 2: Join logical file 0 File is not a join logical file. 1 File is a join logical file.	Database
			Bit 3: Local or remote data 0 Data is stored on local system. 1 Data is stored on remote system.	Database

Table A-1 (Page 5 of 5). Open Feedback Area

Offset	Data Type	Length	Contents	File Type
			Bit 4: Remote System/38 or AS/400 data. Applicable only if the value of Bit 3 is 1. 0 Data is on a remote System/38 or AS/400 system. 1 Data is not on a remote System/38 or AS/400 system.	Database
			Bit 5: Separate indicator area 0 Indicators are in the I/O buffer of the program. 1 Indicators are not in the I/O buffer of the program. The DDS keyword, INDARA, was used when the file was created.	Printer, display, and ICF
			Bit 6: User buffers 0 System creates I/O buffers for the program. 1 User program supplies I/O buffers.	All
			Bits 7-8: Reserved.	
133	Character	13	Reserved.	
146	Binary	2	Number of devices defined for this ODP. For displays, this is determined by the number of devices defined on the DEV parameter of the Create Display File (CRTDSPF) command. For ICF, this is determined by the number of program devices defined or acquired with the Add ICF Device Entry (ADDICFDEVE) or the Override ICF Device Entry (OVRICFDEVE) command. For all other files, it has the value of 1.	All
148	Character		Device name definition list. See "Device Definition List" on page A-6 for a description of this array.	All

Device Definition List

The device definition list part of the open feedback area is an array structure. Each entry in the array contains information about each device or communications session attached to the file. The number of entries in this array is determined by the number at offset 146 of the open feedback area. The device definition list begins at offset 148 of the open feedback area. The offsets shown for it are from the start of the device definition list rather than the start of the open feedback area.

Table A-2 (Page 1 of 4). Device Definition List

Offset	Data Type	Length	Contents	File Type
0	Character	10	Program device name. For database files, the value is DATABASE. For printer or diskette files being spooled, the value is *N. For save files, the value is *NONE. For ICF files, the value is the name of the program device from the ADDICFDEVE or OVRICFDEVE command. For all other files, the value is the name of the device description.	All, except inline data
10	Character	50	Reserved.	
60	Character	10	Device description name. For printer or diskette files being spooled, the value is *N. For save files, the value is *NONE. For all other files, the value is the name of the device description.	All, except database and inline data
70	Character	1	Device class. hex 01 Display hex 02 Printer hex 04 Diskette hex 05 Tape hex 09 Save hex 0B ICF	All, except database and inline data

Table A-2 (Page 2 of 4). Device Definition List

Offset	Data Type	Length	Contents	File Type
71	Character	1	Device type. hex 02 5256 Printer hex 07 5251 Display Station hex 08 Spooled hex 0A BSCEL hex 0B 5291 Display Station hex 0C 5224/5225 printers hex 0D 5292 Display Station hex 0E APPC hex 0F 5219 Printer hex 10 5583 Printer (DBCS) hex 11 5553 Printer hex 12 5555-B01 Display Station hex 13 3270 Display Station hex 14 3270 Printer hex 15 Graphic-capable device hex 16 Financial terminal hex 17 3180 Display Station hex 18 Save file hex 19 3277 DHCF Device hex 1A 9347 Tape Unit hex 1B 9348 Tape Unit hex 1C 9331-1 Diskette Unit hex 1D 9331-2 Diskette Unit hex 1E Intrasystem communications support hex 1F Asynchronous communications support hex 20 SNUF hex 21 4234 (SCS) Printer hex 22 3812 (SCS) Printer hex 23 4214 Printer hex 24 4224 (IPDS) Printer hex 25 4245 Printer hex 26 3179-2 Display Station hex 27 3196-A Display Station hex 28 3196-B Display Station hex 29 5262 Printer hex 2A 6346 Tape Unit hex 2B 2440 Tape Unit hex 2C 9346 Tape Unit hex 2D 6331 Diskette Unit hex 2E 6332 Diskette Unit hex 30 3812 (IPDS) Printer hex 31 4234 (IPDS) Printer hex 32 IPDS printer, model unknown	All, except database and inline data

Table A-2 (Page 3 of 4). Device Definition List

Offset	Data Type	Length	Contents	File Type
			hex 33 3197-C1 Display Station	
			hex 34 3197-C2 Display Station	
			hex 35 3197-D1 Display Station	
			hex 36 3197-D2 Display Station	
			hex 37 3197-W1 Display Station	
			hex 38 3197-W2 Display Station	
			hex 39 5555-E01 Display Station	
			hex 3A 3430 Tape Unit	
			hex 3B 3422 Tape Unit	
			hex 3E 3476-EA Display Station	
			hex 40 3278 DHCF device	
			hex 41 3279 DHCF device	
			hex 42 ICF finance device	
			hex 43 Retail communications device	
72	Binary	2	Number of lines on the display screen.	Display
74	Binary	2	Number of positions in each line of the display screen.	Display
76	Character	2	Bit flags.	Display
			Bit 1: Blinking capability. 0 Display is not capable of blinking. 1 Display is capable of blinking.	
			Bit 2: Device location. 0 Local device. 1 Remote device.	Display
			Bit 3: Acquire status. This bit is set even if the device is implicitly acquired at open time. 0 Device is not acquired. 1 Device is acquired.	Display, ICF
			Bit 4: Invite status. 0 Device is not invited. 1 Device is invited.	Display, ICF
			Bit 5: Data available status (only if device is invited). 0 Data is not available. 1 Data is available.	Display, ICF
			Bit 6: Transaction status. 0 Transaction is not started. An evoke request has not been sent, a detach request has been sent or received, or the transaction has completed. 1 Transaction is started. The transaction is active. An evoke request has been sent or received and the transaction has not ended.	ICF
			Bit 7: Requester device. 0 Not a requester device. 1 A requester device.	Display, ICF

Table A-2 (Page 4 of 4). Device Definition List

Offset	Data Type	Length	Contents	File Type
			Bit 8: DBCS device. 0 Device is not capable of processing double-byte data. 1 Device is capable of processing double-byte data.	Display
			Bits 9-10: Reserved.	
			Bit 11: DBCS keyboard. 0 Keyboard is not capable of entering double-byte data. 1 Keyboard is capable of entering double-byte data.	Display
			Bits 12-16: Reserved.	
78	Character	1	Synchronization level. hex 00 The transaction was built with SYNLVL(*NONE). Confirm processing is not allowed. hex 01 The transaction was built with SYNLVL(*CONFIRM). Confirm processing is allowed.	ICF
79	Character	1	Conversation type. hex D0 Basic conversation (CNVTYPE(*USER)). hex D1 Mapped conversation (CNVTYPE(*SYS)).	ICF
80	Character	50	Reserved.	

Volume Label Fields

Table A-3. Volume Label Fields

Offset	Data Type	Length	Contents	File Type
0	Character	128	Volume label of current volume.	Diskette, tape
128	Character	128	Header label 1 of the opened file.	Diskette, tape
256	Character	128	Header label 2 of the opened file.	Tape

I/O Feedback Area

The results of I/O operations are communicated to the program using OS/400 messages and I/O feedback information. The I/O feedback area is updated for every I/O operation unless your program is using blocked record I/O. In that case, the feedback area is updated only when a block of records is read or written. Some of the information reflects the last record in the block. Other information, such as the count of I/O operations, reflects the number of operations on blocks of records and not the number of records. See your high-level language manual to determine if your program uses blocked record I/O.

The I/O feedback area consists of two parts: a common area and a file-dependent area. The file-dependent area varies by the file type.

Common I/O Feedback Area

Table A-4 (Page 1 of 3). Common I/O Feedback Area

Offset	Data Type	Length	Contents
0	Binary	2	Offset to file-dependent feedback area.
2	Binary	4	Write operation count. Updated only when a write operation completes successfully. For blocked record I/O operations, this count is the number of blocks, not the number of records.
6	Binary	4	Read operation count. Updated only when a read operation completes successfully. For blocked record I/O operations, this count is the number of blocks, not the number of records.
10	Binary	4	Write-read operation count. Updated only when a write-read operation completes successfully.
14	Binary	4	Other operation count. Number of successful operations other than write, read, or write-read. Updated only when the operation completes successfully. This count includes update, delete, force-end-of-data, force-end-of-volume, change-end-of-data, release record lock, and acquire/release device operations.
18	Character	1	Reserved.
19	Character	1	Current operation. hex 01 Read or read block or read from invited devices hex 02 Read direct hex 03 Read by key hex 05 Write or write block hex 06 Write-read hex 07 Update hex 08 Delete hex 09 Force-end-of-data hex 0A Force-end-of-volume hex 0D Release record lock hex 0E Change end-of-data hex 11 Release device hex 12 Acquire device

Table A-4 (Page 2 of 3). Common I/O Feedback Area

Offset	Data Type	Length	Contents
20	Character	10	Name of the record format just processed, which is either: • Specified on the I/O request, or • Determined by default or format selection processing For display files, the default name is either the name of the only record format in the file or the previous record format name for the record written to the display that contains input-capable fields. Because a display file may have multiple formats on the display at the same time, this format may not represent the format where the last cursor position was typed. For ICF files, the format name is determined by the system, based on the format selection option used. Refer to the *Communications Programmer's Guide* for more information.
30	Character	2	Device class: Byte 1: hex 00 Database hex 01 Display hex 02 Printer hex 04 Diskette hex 05 Tape hex 09 Save hex 0B ICF Byte 2 (if byte 1 contains hex 00): hex 00 Nonkeyed file hex 01 Keyed file Byte 2 (if byte 1 does not contain hex 00): hex 02 5256 Printer hex 07 5251 Display Station hex 08 Spooled hex 0A BSCEL hex 0B 5291 Display Station hex 0C 5224/5225 printers hex 0D 5292 Display Station hex 0E APPC hex 0F 5219 Printer hex 10 5583 Printer (DBCS) hex 11 5553 Printer hex 12 5555-B01 Display Station hex 13 3270 Display Station hex 14 3270 Printer hex 15 Graphic-capable device hex 16 Financial terminal hex 17 3180 Display Station hex 18 Save file hex 19 3277 DHCF device hex 1A 9347 Tape Unit hex 1B 9348 Tape Unit

Table A-4 (Page 3 of 3). Common I/O Feedback Area

Offset	Data Type	Length	Contents
			hex 1C 9331-1 Diskette Unit
			hex 1D 9331-2 Diskette Unit
			hex 1E Intrasystem communications support
			hex 1F Asynchronous communications support
			hex 20 SNUF
			hex 21 4234 (SCS) Printer
			hex 22 3812 (SCS) Printer
			hex 23 4214 Printer
			hex 24 4224 (IPDS) Printer
			hex 25 4245 Printer
			hex 26 3179-2 Display Station
			hex 27 3196-A Display Station
			hex 28 3196-B Display Station
			hex 29 5262 Printer
			hex 2A 6346 Tape Unit
			hex 2B 2440 Tape Unit
			hex 2C 9346 Tape Unit
			hex 2D 6331 Diskette Unit
			hex 2E 6332 Diskette Unit
			hex 30 3812 (IPDS) Printer
			hex 31 4234 (IPDS) Printer
			hex 32 IPDS printer, model unknown
			hex 33 3197-C1 Display Station
			hex 34 3197-C2 Display Station
			hex 35 3197-D1 Display Station
			hex 36 3197-D2 Display Station
			hex 37 3197-W1 Display Station
			hex 38 3197-W2 Display Station
			hex 39 5555-E01 Display Station
			hex 3A 3430 Tape Unit
			hex 3B 3422 Tape Unit
			hex 3E 3476-EA Display Station
			hex 40 3278 DHCF device
			hex 41 3279 DHCF device
			hex 42 ICF finance device
			hex 43 Retail communications device
32	Character	10	Device name. The name of the device for which the operation just completed. Supplied only for display, printer, tape, diskette, and ICF files. For printer or diskette files being spooled, the value is *N. For ICF files, the value is the program device name. For other files, the value is the device description name.
42	Binary	4	Length of the record processed by the last I/O operation (supplied only for an ICF, display, tape, or database file). On ICF write operations, this is the record length of the data. On ICF read operations, it is the record length of the record associated with the last read operation.
46	Character	80	Reserved.
126	Binary	2	Number of records retrieved on a read request for blocked records or sent on a write or force-end-of-data or force-end-of-volume request for blocked records. Supplied only for database, diskette, and tape files.
128	Character	4	Reserved.
132	Binary	4	Current block count. The number of blocks of the tape data file already written or read. For tape files only.
136	Character	8	Reserved.

I/O Feedback Area for ICF and Display Files

Table A-5 (Page 1 of 4). I/O Feedback Area for ICF and Display Files

Offset	Data Type	Length	Contents	File Type
0	Character	2	Flag bits. Bit 1: Cancel-read indicator. 0 The cancel-read operation did not cancel the read request. 1 The cancel-read operation canceled the read request. Bit 2: Data-returned indicator. 0 The cancel-read operation did not change the contents of the input buffer. 1 The cancel-read operation placed the data from the read-with-no-wait operation into the input buffer. Bit 3: Command key indicator. 0 Conditions for setting this indicator did not occur. 1 The Print, Help, Home, Roll Up, Roll Down, or Clear key was pressed. The key is enabled with a DDS keyword, but without a response indicator specified. Bits 4-16: Reserved.	Display

Table A-5 (Page 2 of 4). I/O Feedback Area for ICF and Display Files

<table>
<tr><th>Offset</th><th>Data Type</th><th>Length</th><th>Contents</th><th>File Type</th></tr>
<tr><td>2</td><td>Character</td><td>1</td><td>Attention indicator byte (AID). This field identifies which function key was pressed.
For ICF files, this field will always contain the value hex F1 to imitate the Enter key being pressed on a display device.
For display files, this field will contain the 1-byte hexadecimal value returned from the device.
<table>
<tr><th>Hex Codes</th><th>Function Keys</th></tr>
<tr><td>hex 31</td><td>1</td></tr>
<tr><td>hex 32</td><td>2</td></tr>
<tr><td>hex 33</td><td>3</td></tr>
<tr><td>hex 34</td><td>4</td></tr>
<tr><td>hex 35</td><td>5</td></tr>
<tr><td>hex 36</td><td>6</td></tr>
<tr><td>hex 37</td><td>7</td></tr>
<tr><td>hex 38</td><td>8</td></tr>
<tr><td>hex 39</td><td>9</td></tr>
<tr><td>hex 3A</td><td>10</td></tr>
<tr><td>hex 3B</td><td>11</td></tr>
<tr><td>hex 3C</td><td>12</td></tr>
<tr><td>hex B1</td><td>13</td></tr>
<tr><td>hex B2</td><td>14</td></tr>
<tr><td>hex B3</td><td>15</td></tr>
<tr><td>hex B4</td><td>16</td></tr>
<tr><td>hex B5</td><td>17</td></tr>
<tr><td>hex B6</td><td>18</td></tr>
<tr><td>hex B7</td><td>19</td></tr>
<tr><td>hex B8</td><td>20</td></tr>
<tr><td>hex B9</td><td>21</td></tr>
<tr><td>hex BA</td><td>22</td></tr>
<tr><td>hex BB</td><td>23</td></tr>
<tr><td>hex BC</td><td>24</td></tr>
<tr><td>hex BD</td><td>Clear</td></tr>
<tr><td>hex F1</td><td>Enter/Rec Adv</td></tr>
<tr><td>hex F3</td><td>Help (not in operator-error mode)</td></tr>
<tr><td>hex F4</td><td>Roll Down</td></tr>
<tr><td>hex F5</td><td>Roll Up</td></tr>
<tr><td>hex F6</td><td>Print</td></tr>
<tr><td>hex F8</td><td>Record Backspace</td></tr>
<tr><td>hex 3F</td><td>Auto Enter (for Selector Light Pen)</td></tr>
</table></td><td>Display, ICF</td></tr>
<tr><td>3</td><td>Character</td><td>2</td><td>Cursor location (line and position). Updated on input operations that are not subfile operations that return data to the program. For example, hex 0102 means line 1, position 2. Line 10, position 33 would be hex 0A21.</td><td>Display</td></tr>
<tr><td>5</td><td>Binary</td><td>4</td><td>Actual data length. For an ICF file, see the <i>Communications Programmer's Guide</i> for additional information. For a display file, this is the length of the record format processed by the I/O operation.</td><td>Display, ICF</td></tr>
<tr><td>9</td><td>Binary</td><td>2</td><td>Relative record number of a subfile record. Updated for a subfile record operation. For input operations, updated only if data is returned to the program.</td><td>Display</td></tr>
</table>

Table A-5 (Page 3 of 4). I/O Feedback Area for ICF and Display Files

Offset	Data Type	Length	Contents	File Type
11	Binary	2	Indicates the lowest subfile relative record number currently displayed in the uppermost subfile display area if the last write operation was done to the subfile control record with SFLDSP specified. Updated for roll up and roll down operations. Reset to 0 on a write operation to another record. Not set for message subfiles.	Display
13	Binary	2	Total number of records in a subfile. Updated on a put-relative operation to a subfile record, or a write or write-read operation to a subfile control record that creates the subfile (SFLINZ keyword).	Display
15	Character	19	Reserved.	
34	Character	2	Major return code. 00 Operation completed successfully. 02 Input operation completed successfully, but job is being canceled (controlled). 03 Input operation completed successfully, but no data received. 04 Output exception. 08 Device already acquired. 11 Read from invited devices was not successful. 34 Input exception. 80 Permanent system or file error. 81 Permanent session or device error. 82 Acquire or open operation failed. 83 Recoverable session or device error.	Display, ICF
36	Character	2	Minor return code. For the values for a display file, see Appendix C, "Display File Return Codes." For the values for an ICF file, see the *Communications Programmer's Guide* and the appropriate communications-type programmer's guide.	Display, ICF
38	Character	8	Systems Network Architecture (SNA) sense return code. For some return codes, this field may contain more detailed information about the reason for the error. For a description of the SNA sense codes, see the appropriate SNA manual.	ICF
46	Character	1	Safe indicator: 0 An end-of-text (ETX) control character has not been received. 1 An ETX control character has been received.	ICF
47	Character	1	Reserved.	
48	Character	1	Request Write (RQSWRT) command from remote system/application. 0 RQSWRT not received 1 RQSWRT received	ICF

Table A-5 (Page 4 of 4). I/O Feedback Area for ICF and Display Files

Offset	Data Type	Length	Contents	File Type
49	Character	10	Record format name received from the remote system.	ICF
59	Character	4	Reserved.	
63	Character	8	Mode name.	ICF
71	Character	9	Reserved.	

I/O Feedback Area for Printer Files

Table A-6. I/O Feedback Area for Printer Files

Offset	Data Type	Length	Contents
0	Binary	2	Current line number in a page.
2	Binary	4	Current page count.
6	Character	28	Reserved.
34	Character	2	Major return code. 00 Operation completed successfully. 80 Permanent system or file error. 81 Permanent device error. 82 Open operation failed. 83 Recoverable device error occurred.
36	Character	2	Minor return code. For the values for a printer file, refer to Appendix D, "Printer File Return Codes."

I/O Feedback Area for Database Files

Table A-7. I/O Feedback Area for Database Files

Offset	Data Type	Length	Contents
0	Binary	4	Size of the database feedback area, including the key.
4	Character	4	Bits 1-32: Each bit represents a join file in JFILE keyword. 0 JDFTVAL not supplied for file 1 JDFTVAL supplied for file
8	Character	2	Reserved.
10	Binary	2	Number of locked records.
12	Character	6	Reserved.
18	Character	1	Reserved.
19	Character	1	Current record deleted indication:
			Bits 1-3: Reserved.
			Bit 4: Deleted record indicator. 0 Current file position is at an active record. 1 Current file position is at a deleted record.
			Bit 5: Write operation key feedback indicator. 0 Key feedback is not provided by last write operation. 1 Key feedback is provided by last write operation.
			Bit 6: File position changed indicator. Set only for read and positioning I/O operations. Not set for write, update, and delete I/O operations. 0 File position did not change. 1 File position did change.
			Bit 7: Reserved.
			Bit 8: Duplicate key indicator. 0 The key of the last read or write operation was not a duplicate key. 1 The key of the last read or write operation was a duplicate key.
20	Binary	2	Number of key fields. Use this offset for binary operations. Use the next offset (offset 21) for character operations. These offsets overlap and provide the same value (there can be no more than 32 key fields, and only the low-order byte of offset 20 is used).
21	Character	1	Number of key fields.
22	Character	4	Reserved.
26	Binary	2	Key length.
28	Binary	2	Data member number.
30	Binary	4	Relative record number in data member.
34	Character	*	Key value.

Get Attributes

Performing the get attributes operation allows you to obtain certain information about a specific display device or ICF session.

Table A-8 (Page 1 of 3). Get Attributes

Offset	Data Type	Length	Contents	File Type
0	Character	10	Program device name.	Display, ICF
10	Character	10	Device description name: Name of the device description associated with this entry.	Display, ICF
20	Character	10	User ID.	Display, ICF
30	Character	1	Device class: D Display I ICF U Unknown	Display, ICF
31	Character	6	Device type: 3179 3179 Display Station 317902 3179-2 Display Station 3180 3180 Display Station 3196A 3196-A1/A2 Display Station 3196B 3196-B1/B2 Display Station 3197C1 3197-C1 Display Station 3197C2 3197-C2 Display Station 3197D1 3197-D1 Display Station 3197D2 3197-D2 Display Station 3197W1 3197-W1 Display Station 3197W2 3197-W2 Display Station 3270 3270 Display Station 3476EA 3476-EA Display Station 525111 5251 Display Station 5291 5291 Display Station 5292 5292 Display Station 529202 5292-2 Display Station 5555B1 5555-B01 Display Station 5555E1 5555-E01 Display Station DHCF77 3277 DHCF device DHCF78 3278 DHCF device DHCF79 3279 DHCF device APPC Advance program-to-program communications device ASYNC Asynchronous communications device BSC Bisynchronous communications device BSCEL BSCEL communications device FINANC ICF Finance communications device INTRA Intrasystem communications device LU1 LU1 communications device RETAIL RETAIL communications device SNUF SNA upline facility communications device	Display, ICF

Table A-8 (Page 2 of 3). Get Attributes

Offset	Data Type	Length	Contents	File Type
37	Character	1	Requester device. This flag indicates whether this entry is defining a *REQUESTER device. N Not a *REQUESTER device (communications source device). Y A *REQUESTER device (communications target device).	Display, ICF
38	Character	1	Acquire status. Set even if device is implicitly acquired at open time. N Device is not acquired. Y Device is acquired.	Display, ICF
39	Character	1	Invite status. Y Device is invited. N Device is not invited.	Display, ICF
40	Character	1	Data available. Y Invited data is available. N Invited data is not available.	Display, ICF
41	Binary	2	Number of rows on display.	Display
43	Binary	2	Number of columns on display.	Display
45	Character	1	Display allow blink. Y Display is capable of blinking. N Display is not capable of blinking.	Display
46	Character	1	Online/offline status. O Display is online. F Display is offline.	Display
47	Character	1	Display location. L Local display. R Remote display.	Display
48	Character	1	Display type. A Alphanumeric or Katakana. I DBCS.	Display
49	Character	1	Keyboard type of display. A Alphanumeric or Katakana keyboard. I DBCS keyboard.	Display
50	Character	1	Transaction status. All communication types. N Transaction is not started. An evoke request has not been sent, a detach request has been sent or received, or the transaction has completed. Y Transaction is started. The transaction is active. An evoke request has been sent or received and the transaction has not ended.	ICF

Table A-8 (Page 3 of 3). Get Attributes

Offset	Data Type	Length	Contents	File Type
51	Character	1	Synchronization level. APPC only. 0 Synchronization level 0 (SYNLVL(*NONE)). 1 Synchronization level 1 (SYNLVL(*CONFIRM)).	ICF
52	Character	1	Conversation being used. APPC only. M Mapped conversation. B Basic conversation.	ICF
53	Character	8	Remote location name. All communication types.	ICF
61	Character	8	Local LU name. APPC only.	ICF
69	Character	8	Local network ID. APPC only.	ICF
77	Character	8	Remote LU name. APPC only.	ICF
85	Character	8	Remote network ID. APPC only.	ICF
93	Character	8	Mode. APPC only.	ICF
101	Character	43	Reserved.	Display, ICF

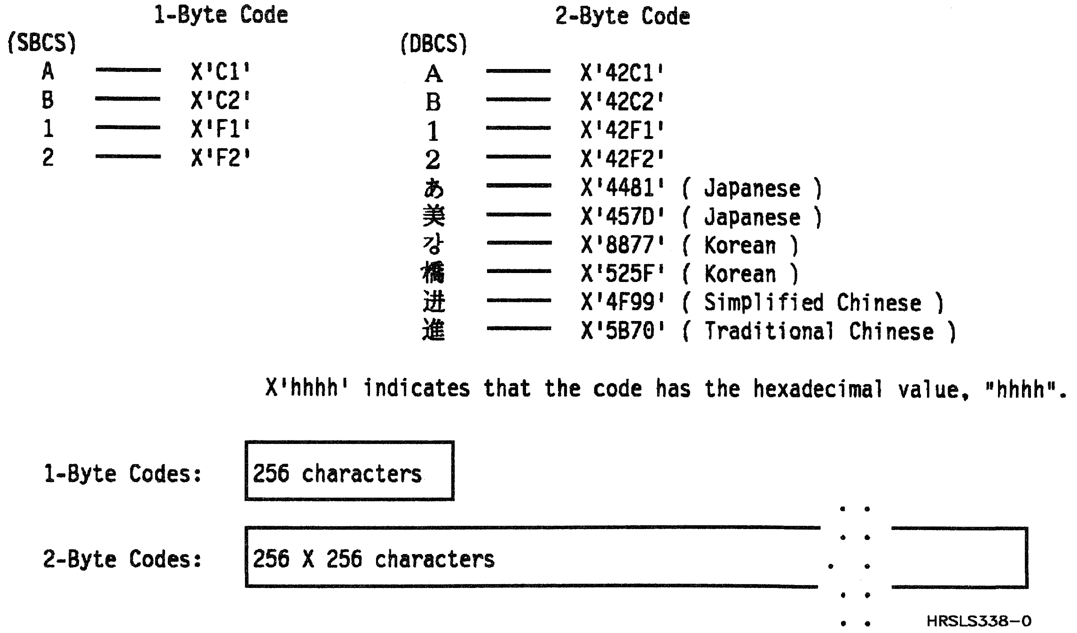
Appendix B. Double-Byte Character Set Support

This appendix contains information that you need if you use double-byte data. This includes the following topics:

- Double-byte character set (DBCS) fundamentals
- Device file support
- Printer support
- Spooling
- Display support
- Copying files that contain double-byte data
- Writing application programs that process double-byte data
- Using DBCS font tables
- Processing double-byte characters
- Using DBCS sort tables
- Using system-supplied objects.
- Using DBCS conversion dictionaries
- Using DBCS conversion

Double-Byte Character Set Fundamentals

Some languages, such as Chinese, Japanese, and Korean, have a writing scheme that uses many different characters that cannot be represented with single-byte codes. To create coded character sets for such languages, the system uses two bytes to represent each character.



(SBCS)	1-Byte Code
A	X'C1'
B	X'C2'
1	X'F1'
2	X'F2'

(DBCS)	2-Byte Code
Ａ	X'42C1'
Ｂ	X'42C2'
１	X'42F1'
２	X'42F2'
あ	X'4481' (Japanese)
美	X'457D' (Japanese)
강	X'8877' (Korean)
橋	X'525F' (Korean)
进	X'4F99' (Simplified Chinese)
進	X'5B70' (Traditional Chinese)

X'hhhh' indicates that the code has the hexadecimal value, "hhhh".

You can use double-byte characters as well as single-byte characters in one application. For instance, you may want to store double-byte data and single-byte data in your database, create your display screens with double-byte text and fields, or print reports with double-byte characters on your printer.

The AS/400 system supports the following double-byte character sets:

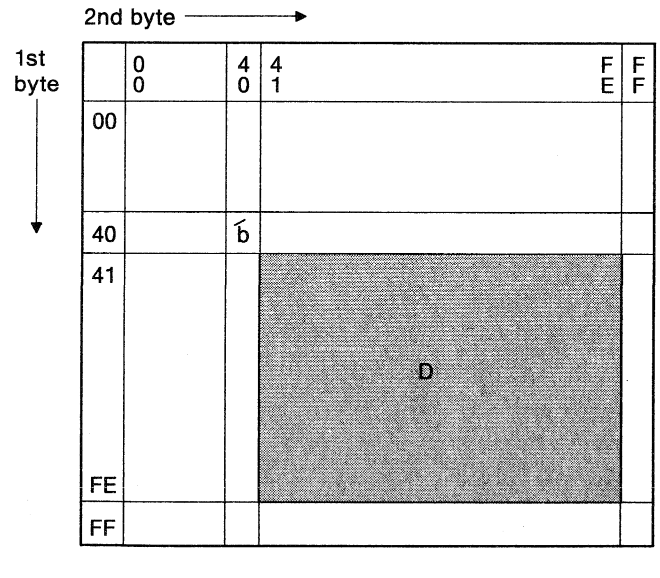
- IBM Japanese Character Set. Includes Kanji, Hiragana, Katakana.
- IBM Korean Character Set. Includes Hangeul, Hanja.
- IBM Simplified Chinese Character Set.
- IBM Traditional Chinese Character Set.

DBCS Code Scheme

DBCS is a graphic character set with the following code-range characteristics:

First byte	hex 41 to hex FE
Second byte	hex 41 to hex FE
Double-byte blank	hex 4040

In the following figure, using the first byte as the vertical axis and the second byte as the horizontal axis, the code points of 256 x 256 intersections are expressed. The lower-right code area is designated as the valid double-byte code area and ƀ is assigned to the double-byte blank.



ƀ: double-byte blank
D: double-byte code area

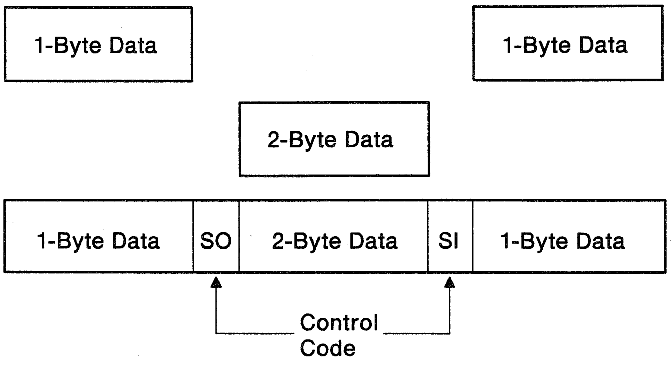
RSLH712-1

By assigning the values hex 41 to hex FE in the first and second bytes as the DBCS character codes, the codes can be grouped in wards. For example, the code group starting with hex 41 is called *ward 41* and hex 42 is called *ward 42*. The characters of ward 42 are double-byte alphanumeric characters which represent the same alphanumeric characters as single-byte characters, but with double-byte codes.

This coding scheme applies to the AS/400 system, System/36, System/38, as well as the System/370. A different DBCS code scheme is used on IBM Personal System/55

Shift Control Characters

The system uses shift control characters to identify the beginning and end of a string of double-byte characters. The shift-out character (SO), hex 0E, indicates the beginning of a double-byte character string. The shift-in character (SI), hex 0F, indicates the end of a double-byte character string.



RSLH713-0

Each shift control character occupies the same amount of space as one alphameric character. By contrast, double-byte characters occupy the same amount of space as two alphameric characters.

Using Double-Byte Data

This section tells you where you can use double-byte data and discusses the limitations to its use.

Where You Can Use

You can use double-byte data in the following ways:

- As data in files:
  - Data in database files.
  - Data entered in input-capable and data displayed in output-capable fields of display files.
  - Data used as literals (strings of characters enclosed in apostrophes [']) in display files.
  - Data printed in output-capable fields and literals in printer files.
- As the text of messages and as the text of object descriptions on CL commands. A text description written in double-byte characters cannot be longer than 24 characters.
- As literals and constants, and as data to be processed by high-level language programs.

Double-byte data can be displayed only at DBCS display stations and printed only on DBCS printers. Double-byte data can be written onto diskettes and tape, as well as onto disk storage.

Where You Cannot Use

You cannot use double-byte data in the following ways:

- As AS/400 object names.
- As command names or variable names in control language (CL) and other high-level languages.
- As displayed or printed output on alphameric devices.

Device File Support

The following sections describe DBCS device files and special considerations for working with DBCS device files. Data description specifications (DDS), a language used to describe files, can be used with DBCS device files. For information about using DDS, refer to the *DDS Reference* manual.

What a DBCS File Is

A *DBCS file* is a file that contains double-byte data or is used to process double-byte data. Other files are called *alphameric files*.

The following types of device files can be DBCS device files:

- Display
- Printer
- Tape
- Diskette
- ICF

When to Indicate a DBCS File

You should indicate that a file is DBCS in one or more of the following situations:

- The file receives input, or displays or prints output, which has double-byte characters.
- The file contains double-byte literals.
- The file has double-byte literals in the DDS that are used in the file at processing time (such as constant fields and error messages).
- The DDS of the file, if any, includes DBCS keywords. See the *DDS Reference* for information on these keywords.
- The file stores double-byte data (database files).

How to Indicate a DBCS File

You must indicate that a device file is a DBCS file in order for the system to process it properly. You can do this in one of the following ways:

- Through DDS
  - DDS provides the following data types.
    - DBCS-Only Field

      DBCS-only fields display and accept only double-byte characters. In these fields, double-byte characters are always enclosed in shift-out and shift-in characters, which have to be paired. One double-byte character occupies two single-byte character positions, whereas the shift-in and shift-out characters each occupy one single-byte character position.

    - Open Field

      Open fields display and accept both single-byte and double-byte characters. Double-byte characters are enclosed in shift-out and shift-in characters that have to be paired.

    - Either Field

      Either fields display and accept *either* single-byte or double-byte characters, but not *both*. Double-byte characters are enclosed in shift-out and shift-in character pairs.

  - In printer and ICF files, by defining fields with DBCS open data type (type O).
  - In display files, by defining fields with DBCS-only data type (J), either data type (E), or open data type (O).
  - By using a double-byte literal that is used with the file at processing time, such as literals specified with the Default (DFT) and Error Message (ERRMSG) DDS keywords.

    Note: You may also use double-byte literals as text and comments in a file, such as with the DDS keyword TEXT. However, the system does not consider a file, whose only DBCS attribute is that it has double-byte documentation, to be a DBCS file.

  - By specifying the Alternative Data Type (IGCALTTYP) DDS keyword on display and printer files. This keyword lets you use display and printer files with both alphameric and double-byte applications. When you put the IGCALTTYP keyword into effect, you can use double-byte data with the file.

    Put the IGCALTTYP keyword into effect by changing or overriding display and printer files with the IGCDTA(*YES) value using the Change Display File (CHGDSPF), Change Printer File (CHGPRTF), Override with Display File (OVRDSPF), and Override with Printer File (OVRPRTF) commands. When you specify IGCDTA(*NO), the IGCALTTYP keyword is not in effect and you can use only alphameric data with the file. Changing or overriding the file to put the IGCALTTYP keyword into effect does not change the DDS of the file.

- By specifying IGCDTA(*YES) on the file creation command. You can enter this value on the following file creation commands:

  - Create Diskette File (CRTDKTF)
  - Create Display File (CRTDSPF)
  - Create Printer File (CRTPRTF)
  - Create Tape File (CRTTAPF)

  Except when using the IGCALTTYP function, you do not need to specify IGCDTA(*YES) on the file creation command if you have already specified DBCS functions in the DDS. Instead, specify IGCDTA(*YES) when the file has DBCS functions that are not indicated in the DDS. For example, specify IGCDTA(*YES) on the file creation command if the file issues messages with DBCS text.

Improperly Indicated DBCS Files

If you do not properly indicate that a file is a DBCS file, one of the following happens:

- For printer files, printer data management assumes the output data to the printer does not contain double-byte data. The end result depends on the type of printer the data is printed on and the status of the replace unprintable character parameter for the printer file you are using.

  If the printer is an alphameric printer and the replace-unprintable-character option has not been chosen, the double-byte data, including the control characters, is sent as-is to the printer. On most alphameric printers, the shift control characters are invalid, and an error will occur at the printer.

  If the printer is a DBCS printer and the replace-unprintable-character is not selected, the double-byte data is sent as-is to the printer. In this case, since the printer is a DBCS printer, the shift control characters are interpreted correctly, and the double-byte data is printed with the exception of extended characters. Because the file was not indicated as a DBCS file, the system will not perform extended character processing. The extended characters are printed with the default symbol for undefined characters.

  If the replace-unprintable-character option is selected, printer data management support interprets shift control characters as unprintable characters and replaces them with blanks. The double-byte data itself is interpreted as alphameric data, and the printer attempts to print it as such. The end result probably will not make sense.

- For display files, display data management assumes that the output data to the display does not contain double-byte data. The end result depends on whether the display is an alphameric or DBCS display.

  If the display is an alphameric display, the double-byte data is interpreted as alphameric data. The shift control characters appear as blanks. A portion of the double-byte data may cause alphameric characters to appear, while the remaining portion of the double-byte data is displayed as blanks. In general, the end result does not make sense.

  If the display is a DBCS display, the double-byte data is displayed properly with the exception of extended characters. The system does not perform extended character processing on the data. Therefore, extended characters are displayed with the default symbol for undefined double-byte characters.

- The system does not recognize literals with DBCS text as double-byte literals if the source file is not specified as a DBCS file.

Handling DBCS Object Descriptions

In many cases, printer files are used by the system to produce data that will eventually be displayed or printed. In these cases, the data is first placed into a spooled file using one of the IBM-supplied printer files. The data is then taken from the spooled file and is displayed or printed based on the request of the user.

When the data involved contains DBCS characters, the printer file that is used to place the data into the spooled file must be capable of processing DBCS data. A printer file is capable of processing DBCS data when *YES is specified on the IGCDTA parameter for the file. In most cases, the system recognizes the occurrence of DBCS data and takes appropriate measures to ensure the printer file that is used is capable of processing DBCS data.

There is one case, however, where the system cannot recognize the occurrence of DBCS data and may attempt to use a printer file that is not capable of processing DBCS data. If this occurs, the output at the display or printer may not be readable. This can happen when object descriptions containing DBCS characters are to be displayed or printed. The system does not recognize that DBCS characters are contained in an object description and does not take special action to ensure the correct printer file is used.

To ensure that you do not receive incorrect results when you are processing an object description that contains DBCS characters, some recommendations should be followed. Action is required on your part only if you have single-byte versions of the OS/400 product installed or if you have a single-byte NLS feature installed as a secondary language. Printer files that are received as part of the DBCS version of a product or as part of a DBCS NLS feature are always capable of processing DBCS data.

The following recommended actions should be performed after the product or feature has been installed:

1. If all printers and display devices attached to your system are DBCS-capable, you can enable all printer files for DBCS data. For IBM-supplied printer files that are received as part of a single-byte version of the product or as part of a single-byte NLS feature, you can enable all printer files by issuing the following command:

   ```
   CHGPRTF FILE(*ALL/*ALL) IGCDTA(*YES)
   ```

   After this command has been completed, all printer files in all libraries will be enabled for DBCS data. The change will be a permanent change.

2. If all printer and display devices attached to your system are not DBCS-capable, it is recommended that you do not enable all IBM-supplied printer files.

   Instead, use the library search capabilities of the system to control which printer files will be used for any particular job. When the potential exists that DBCS data will be encountered, the library list for the job should be such that the printer files that are DBCS-enabled will be found first in the library list. Conversely, if only single-byte data is expected to be encountered, the library list should be set up so the printer files that are not enabled will be found first. In this way, the printer file capabilities will match the type of data that will be processed. The decision as to what type of printer file to use is made on the basis of what type of data will be processed. The device that will be used to actually display or print the data should not influence this decision.

   In some cases it may be desirable to use a printer file that is not DBCS-enabled to process DBCS data. In this case, it is recommended that you make the printer file only temporarily DBCS-capable instead of making a permanent change.

   For a specific job, you can make this temporary change by using the OVRPRTF command. By using this command, you can temporarily enable a specific printer file. You can issue this command several times for a job to enable several printer files.

   To temporarily enable a specific printer file, you can use the following command:

   ```
   OVRPRTF FILE(filename) IGCDTA(*YES)
   ```

   Where filename is the name of the printer file you want to enable.

Special Functions and Considerations

Special functions are available for working with DBCS device files:

- The DDS keyword IGCALTTYP lets you change the data type of fields in display and printer files. Refer to the *DDS Reference* for more information about using the IGCALTTYP keyword and when you might use it.
- The DBCS printers offer special printing functions, such as character rotation and condensed printing. See "Printer Support" for a description of these functions.
- The use of DBCS input fields affects the total number of input fields allowed on a display. For a local 5250 display station, you can specify as many as 256 input fields. However, each three instances of a DBCS field reduces the maximum number of fields by one. For example, if there are 9 DBCS fields on a display, then the maximum is 256-(9/3)=253 input fields.

Printer Support

You should be familiar with both the DBCS file support and DDS for DBCS printer files before reading this section. See "What a DBCS File Is" on page B-4 for DBCS file support. See "Special Functions and Considerations" for information about DDS for DBCS printer files.

Using DBCS Printer Functions

The DBCS printers offer the following functions:

- Character rotation
- Expanded character printing
- Condensed printing
- Shift control character printing

Character Rotation

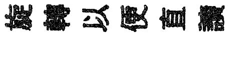
The DBCS printers can rotate double-byte characters 90 degrees counterclockwise before printing so that the printed output can be read vertically.

For example, the character rotation function takes characters as shown:

旋轉以便直讀

HRSLS302-1

and rotates them so that you can read them vertically:



HRSLS303-1

Specify character rotation with the IGCCHRRTT parameter on the Create Printer File (CRTPRTF), Change Printer File (CHGPRTF) and Override with Printer File (OVRPRTF) commands, or with the IGCCHRRTT keyword in the DDS for the file you are printing.

Note: This function rotates only double-byte characters. It does not rotate alphameric characters.

Expanded Character Printing

The DBCS printers can expand characters to twice their normal width, their normal height, or their normal size. Specify the character expansion with the DDS character size (CHRSIZ) keyword. For example, if you specify the value CHRSIZ(2), the following characters:

橫向放大兩倍

HRSLS304-1

are printed twice as wide:

橫　向　放　大　兩　倍

HRSLS305-1

Condensed Printing

The DBCS printers can print 20 double-byte characters per 3 inches so that more double-byte characters fit on a printed line. For example, the following characters shown:

密集印刷節省空間

HRSLS306-1

when condensed, are printed as:

密集印刷節省空間

HRSLS307-1

Specify condensed character printing with the IGCCPI parameter on the CRTPRTF, CHGPRTF, and OVRPRTF commands.

Defining a Line

The record-level define line (DFNLIN) keyword in DDS can be used to draw a horizontal or vertical line (also known as a grid line). A horizontal line is drawn at the bottom of the character spaces. A vertical line is drawn on the left edge of the character spaces.

The DFNLIN keyword is valid for SCS printers.

The maximum number of lines that can be printed at one time is 200. The maximum number of active vertical (vertical lines currently being printed on the page) is 150. More than 200 DFNLIN keywords may be used per page if all the define lines from the previous records have been printed.

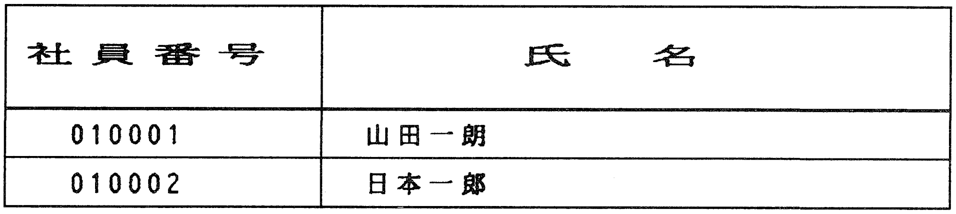
Output considerations at run time:

- Spacing and skipping are processed before the DFNLIN keyword. If you space or skip past the start of a line, that line will be truncated (or not printed if the end of the line is passed also).
- A horizontal line cannot extend over a page boundary. A horizontal or vertical line cannot be started over a page boundary.

- The start line value specified on the DFNLIN keyword cannot be larger than the page length value specified on the PAGESIZE parameter on the printer.
- The start position value specified on the DFNLIN keyword cannot be larger than the page width value specified on the PAGESIZE parameter.
- The sum of the length and the start line value for a vertical line (specified on the DFNLIN keyword) cannot be larger than the page length specified on the PAGESIZE parameter.
- The sum of the length and the start position value for a horizontal line (specified on the DFNLIN keyword) cannot be larger than the page width specified on the PAGESIZE parameter.

A diagnostic message is sent whenever the PAGESIZE and DFNLIN values together cannot correctly process a request.

The following is an example of using DFNLIN to produce lines in a table:



社員番号	氏　名
010001	山田一朗
010002	日本一郎

HRSLS308-1

Shift Control Character Printing

The DBCS printers can print shift control characters in one of the following ways:

- Suppress the shift control characters so that these characters do not occupy any space on printed output.
- Print one blank in the space occupied by each shift control character.
- Print two blanks in the space occupied by the shift-in character and suppress the shift-out character.

Specify how to print shift control characters on the DBCS printers with the IGCSOSI parameter on the CRTPRTF, CHGPRTF, and OVRPRTF commands.

Double-Byte Character Printing Restrictions

When you print double-byte data, consider certain restrictions for the following:

- Extended character printing
- Condensed printing
- Unprintable double-byte characters
- Invalid double-byte characters
- Double-byte data in an alphameric field
- Spanned lines
- Use of the Print key
- Spanned pages
- Cut sheets with the 5553 Printer
- End-of-forms on the 5553 Printer
- Double-byte data printed on alphameric printers

Extended Character Printing

Specify extended character processing to make sure that extended characters are processed. Otherwise, the system prints only basic double-byte characters. See "Processing Double-Byte Characters" on page B-22 for instructions on specifying extended character processing and for information on the effects of such processing.

Condensed Printing

When specifying condensed printing on DBCS printers (by specifying IGCCPI(*CONDENSED) on the CRTPRTF, CHGPRTF, or OVRPRTF command) consider the following:

- Specify the page width in alphameric print positions with the CPI parameter. Although the record to be printed may contain 88 double-byte characters (which would use 176 print positions in normal printing) and the page width is 132 print positions, the double-byte data should print properly in condensed mode.
- For program-described printer files, data might not be printed in the proper position on the page. The system does not perform boundary alignment for alphameric data in printed records. When double-byte and alphameric data are printed on the same line, the printer begins printing alphameric data in the first space following the double-byte data. As a result, characters might not be printed on the proper position on the page.
- For DDS files, the printer begins printing alphameric data in the first position following the double-byte data, when double-byte and alphameric characters are mixed in a field defined with data type O (double-byte-capable). As a result, data might not be printed on the proper position on the page. This situation does not arise when the field contains only double-byte data or when alphameric data is printed in a field defined with an alphameric data type.

Selecting the Appropriate Page Width

Page width is specified as the second value of the PAGESIZE parameter on the CRTPRTF, CHGPRTF, or OVRPRTF commands. The correct page width depends on the printer being used and the characters per inch (CPI) specified for the printer file.

When describing printer files used with printers configured as a 5553 Printer, select a page size in the range based on characters per inch:

CPI	**Page-Width Range**
10	1 through 136
12	1 through 163
13.3	1 through 181
15	1 through 204
18	1 through 244
20	1 through 272

Choose one of the following (depending on the CPI selected) when describing printer files used with printers configured as a 5583 Printer:

CPI	**Page-Width Range**
10	1 through 132
12	1 through 158
13.3	1 through 176
15	1 through 198
18	1 through 236
20	1 through 264

Unprintable Double-Byte Characters

A double-byte character is considered unprintable if its DBCS code is invalid or it does not have a character image defined.

You can specify that the system replace unprintable double-byte characters when printing double-byte characters by specifying the Replace Unprintable Character parameter (RPLUNPRT(*YES)) on the CRTPRTF, CHGPRTF, or OVRPRTF command, but you cannot choose the replacement character.

Although you cannot choose the replacement character for unprintable double-byte characters, you can choose the replacement character for unprintable alphameric characters. To improve system performance, select a blank ( ) as the replacement character for unprintable alphameric characters.

When the system finds an unprintable double-byte character during printing, the following happens:

- If you specify RPLUNPRT(*YES), the system does not send a message when it finds unprintable characters. Instead, the system prints unprintable extended characters as either the DBCS underline (__) when you specify extended character processing, or as the undefined character when you do not specify extended character processing.

  The system prints unprintable basic double-byte characters as DBCS blanks.

- If you specify RPLUNPRT(*NO), the system sends an inquiry message when it finds unprintable characters. You have the following options:

  - Hold the printer file, if it is a spooled printer file.
  - Continue printing where the unprintable character was encountered. If you continue printing, the system sends the inquiry message that you just received. It is sent each time the system finds an unprintable character, regardless of your response to the first message.
  - Continue printing by specifying a page number where printing should continue. When the system finds subsequent unprintable characters, it processes the characters as if the file was specified with RPLUNPRT(*YES). See the item in this list about RPLUNPRT(*YES) for a description of how the system processes these characters.

Invalid Double-Byte Characters

If the system finds an invalid double-byte character, it stops processing double-byte extended characters and prints them as the undefined character.

An invalid double-byte character has an invalid DBCS code. See the *CGU User's Guide* for a list of valid DBCS codes for each DBCS language. This is in contrast to an undefined DBCS character where the DBCS code is valid, but no graphic symbol has been defined for the code.

Double-Byte Data in an Alphameric Field

If you try to print double-byte data in a field that is described in DDS as being alphameric, the system interprets the data to be alphameric. What happens depends on the actual printer being used, whether it is an alphameric or DBCS printer, and the status of the replace-unprintable-characters option. This condition is a special case which is described under "Improperly Indicated DBCS Files" on page B-6.

Spanned Lines

If a printed line of double-byte data exceeds its specified page width (line length), the system tries to continue printing the data. To do this, the system ignores the FOLD parameter on the CRTPRTF, CHGPRTF, and OVRPRTF commands. As a result, the system might not print the double-byte data as you expected and the following occurs:

- If a record to be printed exceeds the page width, the printer *wraps* the data (continues printing the record on the next line). Because the system is not aware that the data is wrapped, the system does not skip lines and start new pages properly. A new page might start in the middle of a record.
- The printer does not split double-byte characters when there is not enough room at the end of a line and a field of double-byte data is continued on a second printed line, even if you specified the CHRSIZ keyword. Instead, the system leaves a blank space on the first line where the character would have been printed and continues printing the complete character on the next line.

Spanned Pages

If data from a printed DBCS field spans to a second page, the system inserts a shift-in character at the beginning of each printed page of double-byte data, shifting the data out of DBCS mode. The printed data that follows does not make sense unless the data on the second page begins with a shift-out character.

To avoid this problem, break double-byte data fields that might span pages into several smaller fields.

Using the Print Key

If you want to print a display containing double-byte data by pressing the Print key, make sure that the associated display file or printer file is a DBCS file. If neither is a DBCS file, the display will not print properly.

One way to make sure that either the display or printer file is a DBCS file is to override the file using the OVRDSPF or the OVRPRTF commands. For example, to override the system-supplied default printer file (the printer file which is used to print displays that are printed by pressing the Print key), enter:

```
OVRPRTF FILE(QSYSPRT) ICGDTA(*YES)
```

Note: Do not change the printer file QSYSPRT to a DBCS file with a CHGPRTF command. This printer file is used to print a variety of system data, including alphameric data. A system performance degradation results if QSYSPRT is a DBCS file and it is processing only alphameric data.

Refer to Chapter 2, “Overrides and File Redirection” for more information on overrides.

5553 Printer with Cut Sheets

If you use cut sheets with the 5553 Printer, leave a one-inch margin at the top and bottom of the page. Without these margins, data may not be printed on the correct page.

5553 Printer End-of-Forms

If you send the ignore (I) reply to the end-of-forms message that you receive when using continuous forms on the 5553 Printer, and if the printer has already printed within 2-1/2 inches of the bottom of the page, the system might not start printing subsequent pages where expected.

To avoid this problem, do the following when you receive the end-of-forms message:

1. Remove the current form from the tractor feed.
2. Insert new forms.
3. Align the first form to the first line.
4. Press the CANCEL button on the printer.
5. Press the SELECT button on the printer.
6. Respond to the end-of-forms message:

   a. For spooled printer files, specify the page on which you want to continue printing when you enter a response to the message. Determine which page to continue printing as follows:

      1) If no data was printed on the last 2-1/2 inches of the last form, enter the number of the next page to be printed.
      2) If data was printed on the last 2-1/2 inches of the last form, enter the number of the last page printed. Reprinting the page ensures that all of the data is printed.

      Use the Work with Writer (WRKWTR) command to find out approximately which page was last printed. The WRKWTR command displays the number of pages that the writer has currently printed.

   b. For direct printer files, enter RETRY to reprint the last page printed. This ensures that all of the data is printed.

Effects of Printing Double-Byte Data on Alphameric Printers

Printing DBCS output on an alphameric printer can result in extensive degradation of system performance.

In addition, the following occurs:

- For direct printer output, the system prints the file and sends a diagnostic message describing the situation to your program message queue.

  Instead of printing double-byte data, the system prints double-byte characters as underlines (__) and prints shift control characters as blanks (   ). Although the system does not print the individual double-byte characters, they are correctly stored in the system.

- For spooled printer output, the system sends an inquiry message to the message queue named on the Start Printer Writer (STRPRTWTR) command. This message lets you do the following:

  – Continue printing. When you continue printing, the system prints the file but does not print double-byte characters within it. Instead, the system prints double-byte characters as underlines (__) and prints shift control characters as blanks (   ). Although the system does not print the individual double-byte characters, they are correctly stored in the system.
  – Hold the spooled printer file so that you can transfer it to an output queue used only for DBCS output. See Chapter 6, "Spool Support" for instructions on transferring a spooled printer file.
  – Cancel printing altogether.

Spooling Support

Create separate output queues for double-byte and alphameric data. This may improve throughput (the rate at which the system processes work) because the system can process alphameric data more quickly than it can process double-byte data.

See the *Work Management Guide* for instructions on how to create output queues.

Applying Overrides in Printing

When starting a job, consider adding the OVRPRTF command to the initial program of the job:

```
OVRPRTF FILE(QSYSPRT) IGCDTA(*YES)
```

Override the printer file (QSYSPRT) to make it capable of printing double-byte data and to ensure that DBCS output printed as the result of pressing the Print key is printed properly. Refer to Chapter 2, "Overrides and File Redirection" for more information on overrides.

Display Support

The following sections describe information on displaying double-byte characters.

Inserting-Shift Control Characters

The system inserts shift-control characters into DBCS-only fields automatically.

To insert shift-control characters into open fields or either fields, do the following:

1. Position the cursor in the field in which you want to insert double-byte data.
2. Press the Insert Shift Control Character key (according to your 5250 personal computer user's guide or 5295 display station user's guide).

The system inserts both shift control characters at the same time, as follows (where 0_E represents the shift-out character and 0_F represents the shift-in character):

$$0_E0_F$$

The system leaves the cursor under the shift-in character and puts the keyboard in insert mode. Insert double-byte characters between the shift control characters. To insert double-byte characters, start keying in double-byte characters at the cursor position. For example, enter the double-byte character string D1D2D3 as follows (where 0_E represents the shift-out character, 0_F represents the shift-in character, and D1, D2, and D3 represent three double-byte characters):

$$0_E\mathrm{D1D2D3}0_F$$

To find out if a field already has the shift control characters, press the Display Shift Control Character key.

Number of Displayed Extended Characters

The system can display only 512 distinct (different) extended characters on a Japanese display screen at one time. When you try to display the 513th extended character, the system displays the undefined character even if you specified extended character processing.

However, the additional extended characters are stored correctly in the system.

Effects of Displaying Double-Byte Data at Alphameric Work Stations

Alphameric display stations cannot display double-byte data correctly. If you try to display double-byte data at an alphameric display station, the following happens:

- The system sends an inquiry message to that display station, asking whether you want to continue using the program with double-byte data or to cancel it.
- If you continue using the program, displayed double-byte data does not make sense. The system ignores the shift control characters and interprets the double-byte characters as though they were alphameric.

Copying Files

You can copy both spooled and nonspooled DBCS files.

Copying Spooled Files

Copy spooled files that have double-byte data by using the Copy Spooled File (CPYSPLF) command. However, the database file to which the file is being copied must have been created with the IGCDTA(*YES) value specified.

When copying spooled files to a database file that contains double-byte data, an extra column is reserved for the shift-out character. This shift-out character is placed between the control information for the record and the user data. The following table shows the shift-out character column number, based on the value specified for the Control Character (CTLCHAR) keyword:

CTLCHAR Value	Column for Shift-Out Character
*NONE	1
*FCFC	2
*PRTCTL	5
*S36FMT	10

Copying Nonspooled Files

You can use the Copy File (CPYF) command to copy double-byte data from one file to another.

When copying data from a double-byte database file to an alphameric database file, specify one of the following on the CPYF command:

- If both files are source files or if both files are database files, you can specify either the FMTOPT(*MAP) parameter or the FMTOPT(*NOCHK) parameter.
- If one file is a source file and the other file is a database file, specify the FMT(*CVTSRC) parameter.

When you copy DBCS files to alphameric files, the system sends you an informational message describing the difference in file types.

Either the FMTOPT(*MAP) or FMTOPT(*NOCHK) options of the copy file function must be specified for copies from a physical or logical file to a physical file when there are fields with the same name in the from-file and to-file, but the data type for fields is as follows:

From-File Field Data Type	To-File Field Data Type
A (character)	J (DBCS-only)
O (DBCS-open)	J (DBCS-only)
O (DBCS-open)	E (DBCS-either)
E (DBCS-either)	J (DBCS-only)

When you copy double-byte data from one database file to another with the FMTOPT(*MAP) parameter specified, double-byte data will be copied correctly. The system will perform correct padding and truncation of double-byte data to ensure data integrity.

Application Program Considerations

The following sections describe considerations for writing applications that process double-byte data, including:

- Designing
- Changing alphameric application programs to DBCS application programs.

Designing Application Programs That Process Double-Byte Data

Design your application programs for processing double-byte data in the same way you design application programs for processing alphameric data, with the following additional considerations:

- Identify double-byte data used in the database files (if any).
- Design display and printer formats that can be used with double-byte data.
- If needed, provide DBCS conversion as a means of entering data for interactive applications. Use the DDS keyword for DBCS conversion (IGCCNV) to specify DBCS conversion in display files.
- Write DBCS error messages to be displayed by the program.
- Specify extended character processing so that the system prints and displays all double-byte data. See "Extended Characters" on page B-22 for instructions.
- Determine which double-byte characters, if any, must be defined.

When you write application programs to process double-byte data, consider how the system processes double-byte data and any restrictions on using double-byte data.

Changing Alphameric Application Programs to DBCS Application Programs

If an alphameric application program uses externally described display files, you can change that application program to a DBCS application program by changing only the files. To convert an application program, do the following:

1. Create a duplicate copy of the source statements for the alphameric file that you want to change.
2. Change alphameric constants and literals to double-byte constants and literals.
3. Change fields in the file to the open (O) keyboard shift so that you can enter both double-byte and alphameric data in these fields. You do not have to change the length of the fields.
4. Store the converted display file in a separate library. Give the file the same name as its alphameric version.
5. When you want to use the changed file in a job, change the library list, using the Change Library List (CHGLIBL) command, for the job in which the file will be used. The library in which the DBCS display file is stored is then checked before the library in which the alphameric version of the file is stored.

DBCS Font Tables

DBCS font tables contain the images of the double-byte extended characters used on the system. The system uses these images to display and print extended characters. The following sections describe information needed to manage DBCS font tables:

- Using commands to manage DBCS font tables
- Using DBCS font tables on the system
- Finding out if a DBCS font table exists
- Copying a DBCS font table onto tape or diskette
- Copying a DBCS font table from tape or diskette to the system
- Deleting a DBCS font table

You can find instructions for adding user-defined characters (characters created using the Character Generator Utility) to DBCS font tables in the *CGU User's Guide*.

Commands for DBCS Font Tables

The following commands perform operations related to the DBCS font tables:

- Check DBCS Font Table (CHKIGCTBL)
- Copy DBCS Font Table (CPYIGCTBL)
- Delete DBCS Font Table (DLTIGCTBL)
- Start Character Generator Utility (STRCGU)
- Start Font Management Aid (STRFMA)

Finding Out if a DBCS Font Table Exists

Use the Check DBCS Font Table (CHKIGCTBL) command to find out if a DBCS font table exists in your system.

For example, to find out if the table QIGC2424 exists, enter:

```
CHKIGCTBL IGCTBL(QIGC2424)
```

If the table does not exist, the system responds with a message. If the table does exist, however, the system simply returns without a message.

Check for the existence of a table when installing a new type of DBCS device to make sure that the table used by the device exists in the system.

Copying a DBCS Font Table onto Tape or Diskette

Use the Copy DBCS Font Table (CPYIGCTBL) command to copy a DBCS font table onto tape or diskette.

The DBCS font tables are saved when you use the Save System (SAVSYS) command so you do not have to use the CPYIGCTBL command when performing normal system backup.

When to Copy a Table onto Tape or Diskette

Copy a DBCS font table onto tape or diskette in the following instances:

- Before deleting that table.
- After new user-defined characters are added to the tables.
- When planning to use the tables on another system that uses diskettes in a format compatible with the I-format, such as System/36 and System/38.

How to Copy a Table onto Tape or Diskette

To copy a DBCS font table onto a tape or diskettes, do the following:

1. Make sure that you have a tape or two diskettes initialized to the *DATA format. If necessary, initialize the tape or diskettes in the *DATA format by specifying the FMT(*DATA) parameter on the Initialize Diskette (INZDKT) command. See Chapter 7, “Tape Support” for complete instructions on initializing tapes and Chapter 8, “Diskette Support” for diskettes.

2. Load the initialized tape or diskette onto the system.

3. Enter the CPYIGCTBL command as follows:

   a. Choose the value OPTION(*OUT).

   b. Use the DEV parameter to select the device to which you want to copy the table.

   c. Use the SELECT and RANGE parameters to specify which portion of the table you want copied from the system. See the description of the CPYIGCTBL command in the *CL Reference* for instructions on choosing SELECT and RANGE parameter values.

The following are two examples of the CPYIGCTBL command used to copy a DBCS font table onto removable media.

- To copy the DBCS font table QIGC2424 onto diskettes, enter:

  ```
  CPYIGCTBL IGCTBL(QIGC2424) OPTION(*OUT) DEV(QDKT)
  ```

- To copy just the user-defined characters from DBCS font table QIGC2424 onto tape, enter:

  ```
  CPYIGCTBL IGCTBL(QIGC2424) OPTION(*OUT)  +
    DEV(QTAP01) SELECT(*USER)
  ```

4. Press the Enter key. The system copies the DBCS font table onto the specified media.
5. Remove the tape or diskette after the system finishes copying the table.

Copying a DBCS Font Table from Tape or Diskette

Use the Copy DBCS Font Table (CPYIGCTBL) command to copy a DBCS font table from a tape or a diskette onto the system. The system automatically creates the DBCS font table again when copying its contents if the following are true:

- The specified table does not already exist in the system.
- The media from which you are copying the table contains all of the double-byte characters supplied with your system.
- SELECT(*ALL) or SELECT(*SYS) is specified on the CPYIGCTBL command.

How to Copy a Table from a Tape or Diskette

To copy a DBCS font table from tape or diskette onto the system:

1. Load the removable media from which the table will be copied onto the system.
2. Enter the CPYIGCTBL command as follows:

   a. Choose the OPTION(*IN) value.

   b. Use the DEV parameter to select the device from which to copy the DBCS font table.

   c. Use the SELECT and RANGE parameters to specify which portion of the table will be copied from the tape or diskette. See the *CL Reference* for a description of the CPYIGCTBL command and for instructions on choosing SELECT and RANGE parameter values.

   Following are two examples of commands used to copy a DBCS font table on the system.

   - To copy the DBCS font table QIGC2424 from diskette, enter:

     ```
     CPYIGCTBL IGCTBL(QIGC2424) OPTION(*IN) DEV(QDKT)
     ```

   - To copy just the user-defined characters from DBCS font table QIGC2424 from tape and to replace the user-defined characters in the table with the ones from the tape, enter:

     ```
     CPYIGCTBL IGCTBL(QIGC2424) OPTION(*IN) DEV(QTAP01)  +
       SELECT(*USER) RPLIMG(*YES)
     ```

3. Press the Enter key. The system copies the DBCS font table from the tape or diskette onto the system.
4. Remove the tape or diskette after the system finishes copying the table.

Deleting a DBCS Font Table

Use the Delete DBCS Font Table (DLTIGCTBL) command to delete a DBCS font table from the system.

When to Delete a DBCS Font Table

Delete an unused DBCS font table to free storage space. For example, if you do not plan to use Japanese printer 5583 or 5337 with your system, font table QIGC3232 is not needed and can be deleted.

How to Delete a DBCS Font Table

When deleting a table, do the following:

1. If desired, copy the table onto tape or diskettes. See “Copying a DBCS Font Table onto Tape or Diskette” on page B-19 for instructions. If you do not copy the table to removable media before deleting it, you will not have a copy of the table for future use.

2. Vary off all devices using that table.

   Warning: Do not delete a DBCS font table if any device attached to the system, and currently varied on, uses that table. Also, make sure that the affected controller is not varied on. If you try to delete the table while the device and controller are varied on, the system reports any devices attached to the same controller(s) as those devices, and the controller(s) as damaged the next time you try to print or display extended characters on an affected device. If such damage is reported, do the following:

   a. Vary off the affected devices, using the Vary Configuration (VRYCFG) command.
   b. Vary off the affected controller.
   c. Vary on the affected controller.
   d. Vary on the affected devices.
   e. Continue normal system work.

3. Enter the DLTIGCTBL command.

   For example, to delete the DBCS font table QIGC3232, enter:

   ```
   DLTIGCTBL IGCTBL(QIGC3232)
   ```

4. Press the Enter key. The system sends inquiry message CPA8424 to the system operator message queue for you to confirm your intention to delete a DBCS table.

5. Respond to the inquiry message. The system sends you a message when it has deleted the table.

Starting the Character Generator Utility

Use the STRCGU command to call the CGU main menu or specify a specific CGU function, depending on the parameter used. Refer to the *CGU User's Guide* for more information.

Copying User-Defined Double-Byte Characters

Use the STRFMA command to copy user-defined double-byte characters between an AS/400 DBCS font table and a user font file at a Personal System/55 or a Traditional Chinese 5295 Display Station. Refer to the *AS/400 Font Management Aid User's Guide*, SC18-2216, on how to use this command.

System Use of DBCS Font Tables

The system processes double-byte extended characters by first referring to their *DBCS code* in storage, and then displaying or printing the image stored in a DBCS font table that is associated with the code. The table used depends on the dot matrix used by the device on which the character will appear. IBM supplies DBCS font tables that contain images for the system-supplied double-byte extended characters.

Processing Double-Byte Characters

All double-byte data requires special handling and processing. There are two types of double-byte characters: basic and extended. These characters are usually processed by the device on which the characters are displayed or printed.

Basic Characters

The device can process basic double-byte characters without any instructions from the system. The device knows about the characters (graphic) without system intervention because they are stored in the hardware of a DBCS device. The number of double-byte characters that are stored in the device varies with the language supported and the storage size of the device. A DBCS device can display or print basic characters without using the extended character processing function of the operating system.

Extended Characters

When processing extended characters, the device requires the assistance of the system. The system must tell the device what the character looks like before the device can display or print the character. Extended characters are stored in a DBCS font file, not in the hardware of a DBCS device. When displaying or printing extended characters, the device receives them from the DBCS font table under control of the extended character processing function of the operating system.

Extended character processing is defined as a function of the operating system that is required to make characters stored in a DBCS font file available to a DBCS device.

To specify extended character processing, specify the double-byte extended character parameter, IGCEXNCHR(*YES), on the file creation command when you create a display (CRTDSPF command) or printer file (CRTPRTF command) that processes double-byte data. Because IGCEXNCHR(*YES) is the default value, the system automatically processes extended characters unless you instruct it otherwise.

If IGCEXNCHR(*NO) was specified on a file creation command, you can change this file attribute by using a file change (CHGDSPF or CHGPRTF) or override (OVRDSPF or OVRPRTF) command. For example, to override the display file DBCSDSPF so that extended characters are processed, enter:

```
OVRDSPF DSPF(DBCSDSPF) IGCEXNCHR(*YES)
```

Notes:

1. The system ignores the IGCEXNCHR parameter when processing alphameric files.

2. When you use the Japanese 5583 Printer to print extended characters, you must use the Kanji print function of the Advanced DBCS Printer Support licensed program. Refer to *AS/400 Utilities: Kanji Print Function User's Guide and Reference*, SH18-2179, for how to use this utility.

What Happens when Extended Characters Are Not Processed

When extended characters are not processed (when IGCEXNCHR(*NO) is specified on a file creation, change, or override command), the following happens:

- Basic double-byte characters are displayed and printed.
- On displays, the system displays the undefined character where it would otherwise display extended characters.
- On printed output, the system prints the undefined character where it would otherwise print extended characters.
- The extended characters, though not displayed or printed, are stored exactly as you enter them.

Why Double-Byte Characters Require Special Processing

Because of the large number of double-byte characters, the system needs more information to identify each double-byte character than is needed to identify each alphameric character. This additional information uses more storage. A double-byte character uses two bytes of storage, an alphameric character uses only one byte. Also, the system must assist the device when necessary for extended characters.

Double-Byte Character Size

Double-byte characters are physically larger than alphameric characters. In system displays, double-byte characters are twice as wide as alphameric characters.

Consider the width of double-byte characters when you calculate the length of a double-byte data field, because field lengths are still identified as the number of alphameric character positions used. For more information on calculating the length of fields containing double-byte data, refer to the *DDS Reference.*

DBCS Sort Tables

DBCS sort tables contain the sort information and collating sequences of all the double-byte characters used on the system. The system uses these tables to sort double-byte characters using the Sort Utility.

You may sort Japanese, Korean, and Traditional Chinese double-byte characters. Each of these languages have two DBCS sort tables, a DBCS master sort table and a DBCS active sort table, except for Korean which has only a DBCS active sort table. See "DBCS Sort Tables" on page B-28 for a complete list of the tables. The DBCS master sort table contains sort information for all defined DBCS characters. The DBCS active sort table for Japanese and Traditional Chinese is created using the master sort table information and contains the collating sequences of the double-byte characters. These collating sequences have a purpose similar to the EBCDIC and ASCII collating sequences for the single-byte alphanumeric character set. For Korean characters, the Hangeul characters are assigned DBCS codes according to their pronunciation order, so a separate collating sequence is not required. However, each of the Hanja characters is mapped to a Hangeul character of the same pronunciation using the DBCS active sort table QCGACTVK. Refer to the *Sort User's Guide/Reference* for more information.

The following sections describe information needed to manage DBCS sort tables:

- Using commands to manage DBCS sort tables
- Using DBCS sort tables on the system
- Finding out if a DBCS sort table exists
- Saving a DBCS sort table onto tape or diskette
- Restoring a DBCS sort table from tape or diskette
- Copying a DBCS sort table (Japanese only) to a data file
- Copying a DBCS sort table (Japanese only) from a data file
- Deleting a DBCS sort table.

User-defined characters can be defined and maintained using the Character Generator Utility (CGU). Information on CGU can be found in the *CGU User's Guide*.

Commands for DBCS Sort Tables

All DBCS sort tables have an operating system object type of *IGCSRT. The following commands perform operations related to the DBCS sort tables:

- Copy DBCS Sort Table (CPYIGCSRT) command (for Japanese table only)
- Delete DBCS Sort Table (DLTIGCSRT) command
- Start Character Generator Utility (STRCGU) command.

Using DBCS Sort Tables on the System

The tables are loaded into the system when you install the OS/400 program and are considered OS/400 objects. You can save the tables onto tape or diskette, delete them from the system, and restore them onto the system. The Japanese DBCS master sort table can also be copied to a data file and copied from a data file.

Note: For the Japanese tables (QCGMSTR and QCGACTV), the Korean table (QCGACTVK) and the Traditional Chinese tables (QCGMSTRC and QCGACTVC), you can also add sort information for each user-defined character, and add that character to the DBCS collating sequence, as you create it using the Character Generator Utility.

Finding Out if a DBCS Sort Table Exists

Use the Check Object (CHKOBJ) command to find out if a DBCS sort table exists in your system.

For example, to find out if the table QCGMSTR exists, enter:

```
CHKOBJ OBJ(QSYS/QCGMSTR) OBJTYPE(*IGCSRT)
```

If the table does not exist, the system responds with a message. If the table does exist, however, the system does not respond at all.

Check for the existence of a DBCS active sort table when you want to sort double-byte characters for the first time. The DBCS active table for the DBCS language must exist to sort the characters.

Saving a DBCS Sort Table onto Tape or Diskette

Use the Save Object (SAVOBJ) command to save a DBCS sort table onto tape or diskette. Specify *IGCSRT for the object type.

The DBCS sort tables are saved when you use the SAVSYS command so you do not have to use the SAVOBJ command when performing normal system backup.

When to Save a DBCS Sort Table onto Tape or Diskette

Save a DBCS sort table onto tape or diskette in the following instances:

- Before deleting that table
- After information is added, updated, or changed in the tables using the Character Generator Utility
- When planning to use the tables on another AS/400 system

Restoring a DBCS Sort Table from Tape or Diskette

Use the RSTOBJ command to restore a DBCS sort table from a tape or a diskette onto the system. The tables on the tape or diskette must previously have been saved using the SAVOBJ command. Specify *IGCSRT for the object type. The system automatically recreates the DBCS sort table when the specified table does not already exist in the system.

These tables must be restored to the QSYS library for the system to know they exist. For that reason, RSTOBJ restores *IGCSRT objects only to the QSYS library and only if the objects do not already exist there.

Copying a Japanese DBCS Master Sort Table to a Data File

Through the Character Generator Utility, use the CPYIGCSRT command to copy the Japanese DBCS master sort table (QCGMSTR) to a data file. This data file can then be moved to a System/36 DBCS system to replace the Japanese master sort table there.

When to Copy the Japanese DBCS Master Sort Table to a Data File

Copy the Japanese DBCS master sort table to a data file in the following instances:

- When planning to move the table to the System/36 for use there. You should always transport the Japanese DBCS master sort table together with the Japanese DBCS font tables.
- Before deleting that table, as an alternative to the SAVOBJ command. You can then keep the file or save it on diskette or tape.

How to Copy the Japanese DBCS Master Sort Table to a Data File

To copy the Japanese DBCS master sort table to a data file, do the following:

1. Decide what data file you want to copy it to. The file need not exist, it will be automatically created.
2. Enter the CPYIGCSRT command as follows:
   a. Choose the value OPTION(*OUT).
   b. Use the FILE parameter to specify the name of the data file to which you want to copy the master table. If you are transporting the master table to the System/36 for use there, you should specify a file name of #KAMAST, or you will have to rename the file when you get it to the System/36. Use the

AS/400 CPYF command for copying the file onto diskette, and the System/36 TRANSFER command for copying the file from diskette to the System/36.

   c. Use the MBR parameter to specify the name of the data file member to which you want to copy the master table. If you are transporting the master table to the System/36 for use there, you should specify *FILE for the MBR parameter.

3. Press the Enter key. The system creates the file and member if they do not exist, and overwrites the existing member if they do exist.

4. If you now transport this file to your System/36 to replace the #KAMAST file there, you should also use the SRTXBLD procedure to update the active table to reflect the new master table.

Copying a Japanese DBCS Master Sort Table from a Data File

Use the CPYIGCSRT command to copy the Japanese DBCS master sort table (QCGMSTR) from a data file.

When to Copy the Japanese DBCS Master Sort Table from a Data File

Copy the Japanese DBCS master sort table from a data file in the following instances:

- When you have migrated your System/36 master file #KAMAST from your System/36 to a AS/400 data file. This is not necessary if you use the Migration Utility. **You should always migrate the Japanese DBCS master sort table together with the Japanese DBCS font tables.**
- When you have copied a version you wanted of the master table to a file and you now want to restore that version.

How to Copy the Japanese DBCS Master Sort Table from a Data File

To copy the Japanese DBCS master sort table from a data file, do the following:

1. Enter the CPYIGCSRT command as follows:

   a. Choose the value OPTION(*IN).

   b. Use the FILE parameter to specify the name of the data file which contains a migrated System/36 master file or a AS/400 master table previously copied in to the file using OPTION(*OUT) with the CPYIGCSRT command. To migrate your System/36 master file without using the Migration Utility, use the TRANSFER command with the IFORMAT parameter on the System/36 to save the #KAMAST master file on diskette. Use the AS/400 Copy File (CPYF) command to copy the master file #KAMAST from diskette. Use the CPYIGCSRT command as described here to copy data from the file to the AS/400 Japanese DBCS master sort table.

   c. Use the MBR parameter to specify the name of the data file member from which you want to copy the master table data.

2. Press the Enter key. Even though the information in the existing Japanese DBCS master sort table is overridden, that table must exist before you can use this command.

3. To update the Japanese DBCS active table to reflect the new copied information, use the SRTXBLD procedure in the System/36 environment, or the STRCGU command specifying OPTION(5). This must be done before you can use the Sort Utility to sort Japanese double-byte characters.

Deleting a DBCS Sort Table

Use the DLTIGCSRT command to delete a DBCS sort table from the system.

When to Delete a DBCS Sort Table

Delete an unused DBCS sort table to free storage space, but you should always first save a copy of the table using the SAVOBJ command. You should delete the DBCS master sort table for a DBCS language if any of the following are true:

1. You will not be creating any new characters for that language using the Character Generator Utility.
2. You will not be using the Sort Utility to sort characters for that language.

You should delete the DBCS active sort table for a DBCS language if you will not be using the Sort Utility to sort characters for that language. The DBCS active sort table must be on the system to use the Sort Utility for this language.

How to Delete a DBCS Sort Table

When deleting a table, do the following:

1. If desired, save the table onto tape or diskettes. See "Saving a DBCS Sort Table onto Tape or Diskette" on page B-25 for instructions. If you do not save the table onto removable media before deleting it, you will not have a copy of the table for future use.
2. Enter the DLTIGCSRT command.

   For example, to delete the DBCS sort table QCGACTV, enter:

   ```
   DLTIGCSRT IGCSRT(QCGACTV)
   ```
3. Press the Enter key. The system sends you a message when it has deleted the table.

System-Supplied Objects

This section discusses system-supplied DBCS objects that are distributed with each DBCS national language:

- DBCS Font Tables
- DBCS Font Files
- DBCS Sort Tables
- DBCS Conversion Dictionary
- DBCS Device Files

DBCS Font Tables

The following DBCS font tables are objects that you can save or restore. You can also copy the tables onto tape or diskette, copy them from tape or diskette into the system, and delete them. See "DBCS Font Tables" on page B-18 for instructions on these procedures. For each DBCS language, you can also add, delete, and change characters in the tables using the Character Generator Utility.

QIGC2424 A Japanese DBCS font table used to display and print DBCS extended characters in a 24 by 24 dot matrix image. The system uses the table with Japanese display stations and display-station-attached printers, 5227 Model 1 Printer and 5327 Model 1 Printer.

QIGC2424C A Traditional Chinese DBCS font table used to print DBCS extended characters in a 24 by 24 dot matrix image. The system uses the table with the 5227 Model 3 Printer.

QIGC2424K A Korean DBCS font table used to print DBCS extended characters in a 24 by 24 dot matrix image. The system uses the table with the 5227 Model 2 Printer.

QICC2424S A Simplified Chinese DBCS font table used to print DBCS extended characters in a 24 by 24 dot matrix image. The system uses the table with the 5227 Model 5 Printer.

QIGC3232 A Japanese DBCS font table in a 32 by 32 dot matrix image. The system uses the table with the 5583 Printer and the 5337 Printer.

DBCS Font Files

In addition to the system-supplied DBCS font tables, the system also provides DBCS font files. These DBCS font files are physical files which contain frequently used double-byte characters. When using the Character Generator Utility, you can use the characters in these files as the base for a new user-defined character. These files are supplied with read-only authority as they are not to be changed. If you do not use Character Generator Utility or the Advanced DBCS Printer Support licensed program, you may delete these files to save space. They all exist in the QSYS library.

QCGF2424 A Japanese DBCS font file used to store a copy of the Japanese DBCS basic character images to be used as a reference for the CGU and Advanced Print Writer Utility.

QCGF2424K A Korean DBCS font file used to store a copy of the Korean DBCS basic character images to be used as a reference for the CGU and Advanced Print Writer Utility.

QCGF2424C A Traditional Chinese DBCS font file used to store a copy of the Traditional Chinese DBCS basic character images to be used as a reference for the CGU and Advanced Print Writer Utility.

QCGF2424S A Simplified Chinese DBCS font file used to store a copy of the Simplified Chinese DBCS basic character images to be used as a reference for the CGU and Advanced Print Writer Utility.

DBCS Sort Tables

The following DBCS sort tables are objects that you can save, restore and even delete. Using the Character Generator Utility you can also add, delete and change entries in these tables corresponding to the image entries in the DBCS font tables, for Japanese, Korean and Traditional Chinese languages. For Japanese use only, you can also copy the DBCS master sort table to and from a data file. See "DBCS Sort Tables" on page B-23 for instructions on these procedures.

QCGMSTR A Japanese DBCS master sort table used to store the sort information for the Japanese double-byte character set.

QCGACTV A Japanese DBCS active sort table used to store the sort collating sequences for the Japanese double-byte character set.

QCGMSTRC A Traditional Chinese DBCS master sort table used to store the sort information for the Traditional Chinese double-byte character set.

QCGACTVC A Traditional Chinese DBCS active sort table used to store the sort collating sequences for the Traditional Chinese double-byte character set.

QCGACTVK A Korean DBCS active sort table used to map Hanja characters to Hangeul characters with equivalent pronunciation.

DBCS Conversion Dictionary

The system-supplied DBCS conversion dictionary, QSYSIGCDCT, contains Japanese words and phrases and is intended for Japanese data entry.

QSYSIGCDCT contains these entries:

- Personal names
  - Family names
  - First names
- Organization names
  - Private enterprises registered in the security market
  - Public corporations
  - Typical organizations in the central and local governments
  - Most universities and colleges
- Addresses
  - Public administration units within the prefectures
  - Towns and streets in 11 major cities
- Business terms, such as department names and position titles commonly used in enterprises
- Individual double-byte characters, including basic double-byte characters, as defined by IBM

You cannot add or delete entries from this dictionary. However, you may rearrange the related DBCS words so that the words used most frequently are displayed first during DBCS conversion. See "Editing a DBCS Conversion Dictionary" on page B-32 for instructions on rearranging terms.

You can use the following commands to perform object management functions with the DBCS conversion dictionary. Specify the object type (*IGCDCT) on the OBJTYPE(*IGCDCT) parameter when entering these commands:

- CHGOBJOWN: Change the owner of a DBCS conversion dictionary
- CHKOBJ: Check a DBCS conversion dictionary
- CRTDUPOBJ: Create a duplicate object of the dictionary
- DMPOBJ: Dump a DBCS conversion dictionary
- DMPSYSOBJ: Dump the system-supplied dictionary
- DSPOBJAUT: Display a user's authority to the dictionary
- GRTOBJAUT: Grant authority to use the dictionary
- MOVOBJ: Move the dictionary to another library
- RNMOBJ: Rename the dictionary
- RSTOBJ: Restore the dictionary
- RVKOBJAUT: Revoke authority to use the dictionary
- SAVOBJ: Save the dictionary
- SAVCHGOBJ: Save a changed dictionary

The system saves or restores DBCS conversion dictionaries when you use these commands:

- RSTLIB: Restore a library in which the dictionary is stored
- SAVLIB: Save a library in which the dictionary is stored
- SAVSYS: Save QSYSIGCDCT, the system DBCS conversion dictionary, when saving the system

DBCS Device Files

The following chart contains the command used to create the printer device file used by the operating system to format DBCS lists. This object is shipped in addition to those device files named in this guide and is stored in the library QSYS. You should *never* override this file with another file.

File Name	Create Command	Parameters	Name of Associated Commands
QPDSPDCT	CRTPRTF	FILE(QSYS/QPDSPDCT) RPLUNPRT(*NO) DEV(*JOB) MAXRCDS(*NOMAX) SCHEDULE(*FILEEND) LVLCHK(*NO) FOLD(*YES) IGCDTA(*YES) IGCEXNCHR(*YES) IGCSOSI(*YES) TEXT('Printer device file for DSPIGCDCT command')	DSPIGCDCT, EDTIGCDCT

DBCS Conversion Dictionaries

The DBCS conversion dictionary is a collection of alphameric entries and their related DBCS words. The system refers to the dictionary when performing DBCS conversion. See "How DBCS Conversion Works" on page B-39 for information on how the system uses the DBCS conversion dictionary during DBCS conversion.

A system-supplied and a user-created dictionary are used with DBCS conversion. This section describes the following information about DBCS conversion dictionaries:

- Using the system-supplied dictionary
- Using a user-created dictionary
- Using DBCS conversion dictionary commands
- Creating a dictionary
- Editing a dictionary
- Displaying and printing a dictionary
- Deleting a dictionary

System-Supplied Dictionary (for Japanese Use Only)

QSYSIGCDCT, the system-supplied dictionary that is stored in the library, QSYS, is a collection of entries with a Japanese pronunciation expressed in alphameric characters and the DBCS words related to those entries. The system checks this dictionary second when performing DBCS conversion. Refer to "DBCS Conversion Dictionary" on page B-29 for detailed information.

User-Created Dictionary

A user-created dictionary contains any alphameric entries and related DBCS words that you choose to include. You might create a user dictionary to contain words unique to your business or words that you use regularly but that are not included in the system-supplied dictionary.

You can create one or more DBCS conversion dictionaries with any name and store them in any library. When performing DBCS conversion, however, the system only refers to the first user dictionary named QUSRIGCDCT in the user's library list, no matter how many dictionaries you have or what they are named. Make sure that the library list is properly specified so that the system checks the correct dictionary.

During DBCS conversion, the system checks QUSRIGCDCT before checking QSYSIGCDCT.

DBCS Conversion Dictionary Commands

The following commands perform operations related to the DBCS conversion dictionary:

- Create DBCS Conversion Dictionary (CRTIGCDCT)
- Edit DBCS Conversion Dictionary (EDTIGCDCT)
- Display DBCS Conversion Dictionary (DSPIGCDCT)
- Delete DBCS Conversion Dictionary (DLTIGCDCT)

Creating a DBCS Conversion Dictionary

To create a DBCS conversion dictionary, do the following:

1. Use the Create DBCS Conversion Dictionary (CRTIGCDCT) command.
2. Name the dictionary, QUSRIGCDCT, so it can be used during DBCS conversion. The system uses the dictionary if it is the first user-created dictionary found when searching a user's library list.

   You might call the dictionary by another name while it is being created to prevent application programs from using it for conversion. Later, change the dictionary name using the Rename Object (RNMOBJ) command.

   For example, to create a user DBCS conversion dictionary to be stored in the library DBCSLIB, enter:

   ```
   CRTIGCDCT IGCDCT(DBCSLIB/QUSRIGCDCT)
   ```
3. Use the EDTIGCDCT command to put entries and related words into the dictionary after creating it. See "Editing a DBCS Conversion Dictionary" on page B-32 for instructions on putting entries in the dictionary.

Editing a DBCS Conversion Dictionary

Use the Edit DBCS Conversion Dictionary (EDTIGCDCT) command to edit the DBCS conversion dictionary. Use editing to add user-defined characters to the dictionary, so that users can enter characters using DBCS conversion, and rearrange terms in a DBCS conversion dictionary to suit individual needs.

Requirements for a DBCS Conversion Dictionary: The display station needed for use while editing the DBCS conversion dictionary depends on the value that you entered for the ENTRY parameter on the EDTIGCDCT command:

- If you specified a specific string with the ENTRY parameter or if you want to display double-byte characters, you must use a DBCS display station.
- If you did not specify a specific string with the ENTRY parameter, or if you do not want to display double-byte characters, use either a DBCS display station, or a 24-row by 80-column alphameric display station.

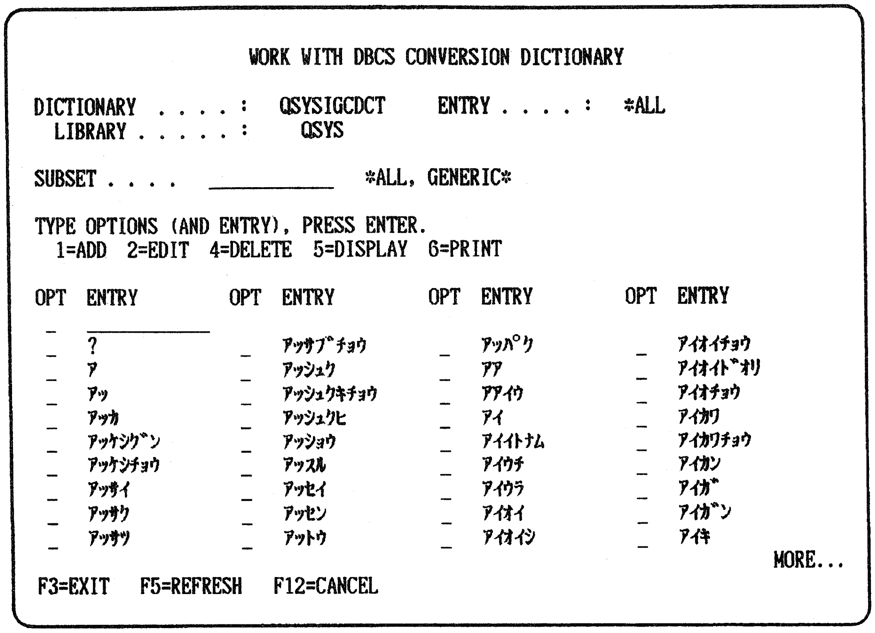
DBCS Conversion Dictionary Operations: You may perform the following editing operations on a user-created DBCS conversion dictionary:

- Add entries to the dictionary (including adding the first entries to the dictionary after it is created). The dictionary can contain as many as 99,999 entries.
- Delete entries from the dictionary.
- Change entries in the dictionary, such as replacing the DBCS words related to an alphameric entry.
- Move the DBCS words related to an alphameric entry to rearrange the order in which they appear during DBCS conversion.

The only editing function that you can perform with QSYSIGCDCT, the system-supplied dictionary, is to move DBCS words related to an alphameric entry. Move words in order to rearrange the order in which they appear during DBCS conversion.

Displays Used for Editing a DBCS Conversion Dictionary: After you enter the EDTIGCDCT command, the system presents either the Work With DBCS Conversion Dictionary display or the Edit Related Words display, depending on the value entered for the ENTRY parameter on the command.

Work with DBCS Conversion Dictionary Display: Use this display to work with alphameric entries, such as choosing an entry to edit or deleting an entry. The system displays the Work with DBCS Conversion Dictionary display if you enter *ALL or a generic string for the ENTRY parameter of the EDTIGCDCT command.

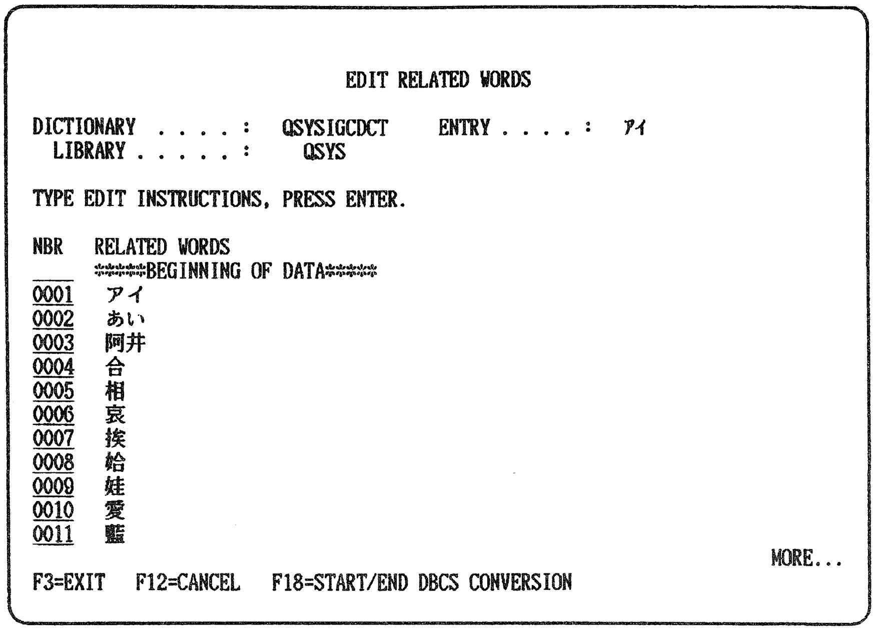


```
                    WORK WITH DBCS CONVERSION DICTIONARY

DICTIONARY . . . . :   QSYSIGCDCT     ENTRY . . . . :   *ALL
  LIBRARY . . . . . :    QSYS

SUBSET . . . .     __________       *ALL, GENERIC*

TYPE OPTIONS (AND ENTRY), PRESS ENTER.
  1=ADD  2=EDIT  4=DELETE  5=DISPLAY  6=PRINT

OPT  ENTRY          OPT  ENTRY          OPT  ENTRY          OPT  ENTRY

 _   __________
 _   ?               _   ｱｯｻﾌﾞﾁｮｳ        _   ｱｯﾊﾟﾘ           _   ｱｲｵｲﾁｮｳ
 _   ｱ               _   ｱｯｼｭｸ           _   ｱｱ              _   ｱｲｵｲﾄﾞｵﾘ
 _   ｱｯ              _   ｱｯｼｭｸｷﾁｮｳ       _   ｱｱｲｳ            _   ｱｲｵﾁｮｳ
 _   ｱｯｶ             _   ｱｯｼｭｸﾋ          _   ｱｲ              _   ｱｲｶﾜ
 _   ｱｯｹｼｸﾞﾝ         _   ｱｯｼｮｳ           _   ｱｲｲﾄﾅﾑ          _   ｱｲｶﾜﾁｮｳ
 _   ｱｯｹｼﾁｮｳ         _   ｱｯｽﾙ            _   ｱｲｳﾁ            _   ｱｲｶﾝ
 _   ｱｯｻｲ            _   ｱｯｾｲ            _   ｱｲｳﾗ            _   ｱｲｶﾞ
 _   ｱｯｻｸ            _   ｱｯｾﾝ            _   ｱｲｵｲ            _   ｱｲｶﾞﾝ
 _   ｱｯｻﾂ            _   ｱｯﾄｳ            _   ｱｲｵｲｼ           _   ｱｲｷ
                                                                        MORE...
F3=EXIT    F5=REFRESH    F12=CANCEL
```

HRSLS332-0

See the discussion of the EDTIGCDCT command in the *CL Reference* for a complete description of the Work with DBCS Conversion Dictionary display.

Edit Related Words Display: Use this display to work with the DBCS words related to an alphameric entry. The system displays the Edit Related Words display if you enter a specific string for the ENTRY parameter. The system also displays the Edit Related Words display if you choose an entry to edit from the Work with DBCS Conversion Dictionary display. Following is an example of the Edit Related Words display:



```
                              EDIT RELATED WORDS

DICTIONARY . . . . :   QSYSIGCDCT    ENTRY . . . . :  ｱｲ
  LIBRARY . . . . . :    QSYS

TYPE EDIT INSTRUCTIONS, PRESS ENTER.

NBR   RELATED WORDS
      *****BEGINNING OF DATA*****
____
0001   アイ
0002   あい
0003   阿井
0004   合
0005   相
0006   哀
0007   挨
0008   姶
0009   娃
0010   愛
0011   藍
                                                                MORE...
F3=EXIT   F12=CANCEL   F18=START/END DBCS CONVERSION
```

HRSLS328-1

See the discussion of the EDTIGCDCT command in the *CL Reference* for a complete description of the Edit Related Words display.

Examples of Editing Operations: The following sections give examples of the editing operations that you can perform using the EDTIGCDCT displays:

- Beginning to edit a dictionary
- Adding the first entries in a dictionary
- Deleting an entry
- Moving a related word
- Ending editing the dictionary

Beginning to Edit a Dictionary: Enter the EDTIGCDCT command to start editing the dictionary for any type of editing operation. For example, to put the first entry in the dictionary, enter:

```
EDTIGCDCT IGCDCT(DBCSLIB/QUSRIGCDCT) ENTRY(*ALL)
```

Or, to edit the entries beginning with the string ABC enter:

```
EDTIGCDCT IGCDCT(DBCSLIB/QUSRIGCDCT) ENTRY('ABC*')
```

Adding the First Entries in a Dictionary: To add the first entries into a dictionary, do the following:

1. Specify ENTRY(*ALL) when entering the EDTIGCDCT command. For example, to edit the dictionary QUSRIGCDCT stored in the library DBCSLIB, enter:

   ```
   EDTIGCDCT IGCDCT(DBCSLIB/QUSRIGCDCT) ENTRY(*ALL)
   ```

   The system displays the Work with DBCS Conversion Dictionary display.

2. Enter a 1 in the first option field in the list and enter an alphameric entry to be added to the dictionary in the entry field.

   The system then displays the Edit Related Words display showing only two lines of data: `BEGINNING OF DATA` and `END OF DATA`.

3. Enter an I in the *NBR* field beside the `BEGINNING OF DATA` line to insert a line.

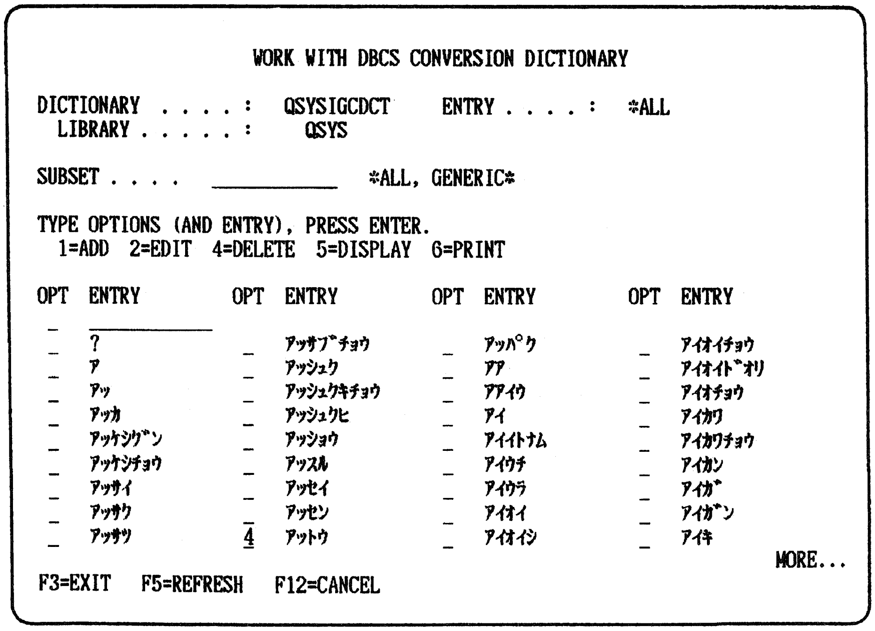
4. Press the Enter key. The system displays a blank line.

5. On the blank line, enter a DBCS word to be related to the new alphameric entry.

   If you enter data on the inserted line and leave the cursor on that line, another new line appears below when you press the Enter key. You can enter another DBCS word on this line, or delete it by leaving it blank, and pressing the Enter key.

6. When you finish adding this first entry, press F12 to take you to the Exit Entry display. Take the Y option to save the entry and then return to the Work With DBCS Conversion Dictionary display. Enter option 1 again and enter another alphameric entry in the entry field to continue adding entries to the dictionary, or press F3 to end editing the dictionary.

Deleting an Entry: Enter a 4 in the input field beside the entry to be deleted as follows:

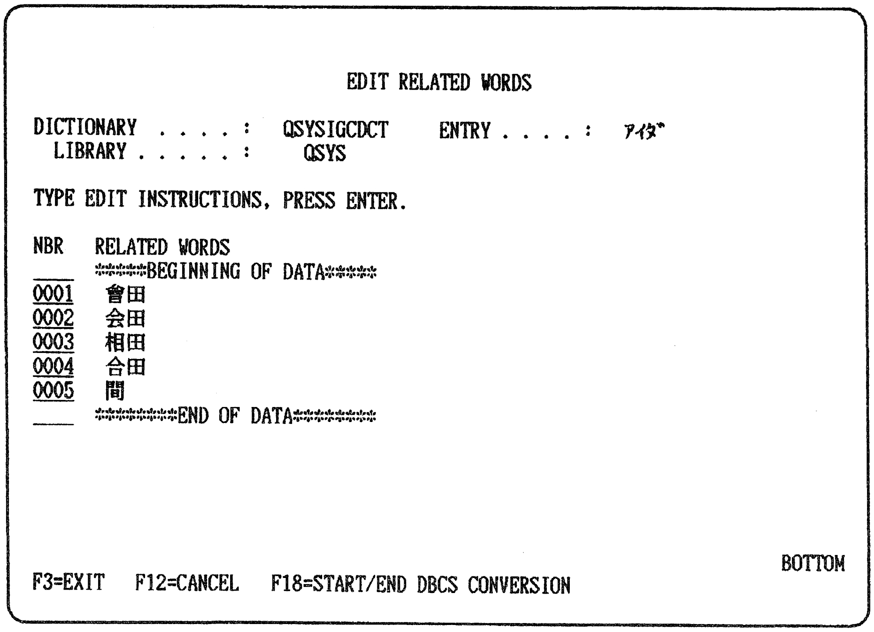


```
                     WORK WITH DBCS CONVERSION DICTIONARY

DICTIONARY . . . . :   QSYSIGCDCT    ENTRY . . . . :   *ALL
  LIBRARY . . . . . :    QSYS

SUBSET . . . .     __________     *ALL, GENERIC*

TYPE OPTIONS (AND ENTRY), PRESS ENTER.
  1=ADD  2=EDIT  4=DELETE  5=DISPLAY  6=PRINT

OPT  ENTRY          OPT  ENTRY          OPT  ENTRY          OPT  ENTRY
 _   __________
 _   ?               _   ｱｯｻﾌﾞﾁｮｳ        _   ｱｯﾊﾟｸ           _   ｱｲｵｲﾁｮｳ
 _   ｱ               _   ｱｯｼｭｸ           _   ｱｱ              _   ｱｲｵｲﾄﾞｵﾘ
 _   ｱｯ              _   ｱｯｼｭｸｷﾁｮｳ       _   ｱｱｲｳ            _   ｱｲｵﾁｮｳ
 _   ｱｯｶ             _   ｱｯｼｭｸﾋ          _   ｱｲ              _   ｱｲｶﾜ
 _   ｱｯｹｼｸﾞﾝ         _   ｱｯｼｮｳ           _   ｱｲｲﾄﾅﾑ          _   ｱｲｶﾜﾁｮｳ
 _   ｱｯｹｼﾁｮｳ         _   ｱｯｽﾙ            _   ｱｲｳﾁ            _   ｱｲｶﾝ
 _   ｱｯｻｲ            _   ｱｯｾｲ            _   ｱｲｳﾗ            _   ｱｲｶﾞ
 _   ｱｯｻｸ            _   ｱｯｾﾝ            _   ｱｲｵｲ            _   ｱｲｶﾞﾝ
 _   ｱｯｻﾂ            4   ｱｯﾄｳ            _   ｱｲｵｲｼ           _   ｱｲｷ
                                                                     MORE...
F3=EXIT   F5=REFRESH   F12=CANCEL
```

HRSLS331-1

Moving a Related Word: Moving the words related to an alphameric entry changes the order in which the words appear during DBCS conversion. To move a word, do the following:

1. Display the Edit Related Words display for the entry in which you want to move DBCS words, either by entering a specific entry with the EDTIGCDCT command, or by choosing an entry to edit from the Work with DBCS Conversion Dictionary display.
2. When the display appears, enter an M in the *NBR* field beside the DBCS word to be moved.
3. Enter an A in the *NBR* field of the line after which the word will be moved.
4. Press the Enter key. The system moves the word, as follows:



```
                         EDIT RELATED WORDS

DICTIONARY . . . . :   QSYSIGCDCT    ENTRY . . . . :   ｱｲﾀﾞ
  LIBRARY . . . . . :    QSYS

TYPE EDIT INSTRUCTIONS, PRESS ENTER.

NBR   RELATED WORDS
____  *****BEGINNING OF DATA*****
0001   會田
0002   会田
0003   相田
0004   合田
0005   間
____  *********END OF DATA*********




                                                               BOTTOM
F3=EXIT   F12=CANCEL   F18=START/END DBCS CONVERSION
```

HRSL326-1

Ending the Editing Process: To end the editing operation, press F3. The Exit Entry display is displayed, and you can choose to save the entry or not. The system then returns you to your basic working display, such as the Command Entry display.

Suggestions for Editing the Dictionary: When editing the DBCS conversion dictionary, consider the following:

- You can use DBCS conversion with the Edit Related Words display to enter related words into a user-created dictionary. See "DBCS Conversion (for Japanese Use Only)" on page B-38 for information on this procedure.
- Place the most commonly used DBCS words at the beginning of the list of related words on the Edit Related Words display. This simplifies DBCS conversion because the system displays the related words in the same order in which those words are listed in the dictionary.

Displaying and Printing the DBCS Conversion Dictionary

Use the Display DBCS Conversion Dictionary (DSPIGCDCT) command to display and print the DBCS conversion dictionary. You can display or print the entire dictionary or just a certain part of it, depending on the value you specify for the ENTRY parameter.

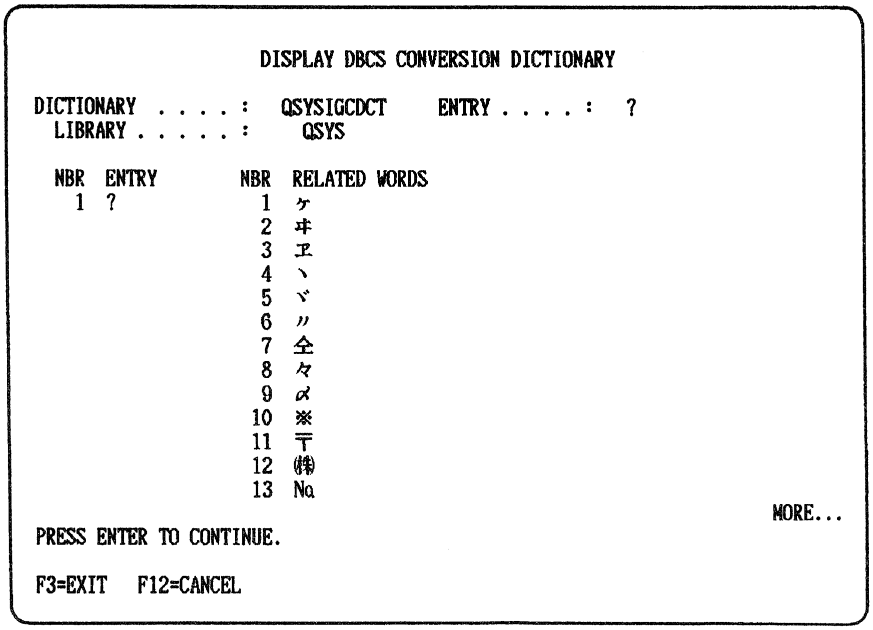
For example, to print the entry ABC from the dictionary QUSRIGCDCT and its related words, enter:

```
DSPIGCDCT IGCDCT(DBCSLIB/QUSRIGCDCT)  +
  ENTRY(ABC) OUTPUT(*PRINT)
```

To display all of the entries from the system-supplied dictionary QSYSIGCDCT and their related words, enter:

```
DSPIGCDCT IGCDCT(QSYS/QSYSIGCDCT)
```

Following is an example of the display produced by the DSPIGCDCT command. It shows alphameric entries and their related words.



```
                        DISPLAY DBCS CONVERSION DICTIONARY

DICTIONARY . . . . :   QSYSIGCDCT      ENTRY . . . . :   ?
  LIBRARY . . . . . :    QSYS

  NBR  ENTRY          NBR  RELATED WORDS
    1  ?                1  ヶ
                        2  ヰ
                        3  ヱ
                        4  ヽ
                        5  ヾ
                        6  〃
                        7  仝
                        8  々
                        9  〆
                       10  ※
                       11  〒
                       12  ㈱
                       13  №
                                                                  MORE...
PRESS ENTER TO CONTINUE.

F3=EXIT    F12=CANCEL
```

HRSLS337-1

See the discussion of the DSPIGCDCT command in the *CL Reference* for a complete description of the command and the display it produces.

Deleting a DBCS Conversion Dictionary

Use the Delete DBCS Conversion Dictionary (DLTIGCDCT) command to delete a DBCS conversion dictionary from the system.

In order to delete the dictionary, you must have object existence authority to the dictionary and object operational authorities to the library in which the dictionary is stored.

When you delete a dictionary, make sure that you specify the correct library name. It is possible that many users have their own dictionaries, each named QUSRIGCDCT, stored in their libraries. If you do not specify any library name, the system deletes the first DBCS conversion dictionary in your library list.

For example, to delete the DBCS conversion dictionary QUSRIGCDCT in the library DBCSLIB, enter:

```
DLTIGCDCT IGCDCT(DBCSLIB/QUSRIGCDCT)
```

DBCS Conversion (for Japanese Use Only)

DBCS conversion is an alternative to directly keying in double-byte data. It lets you enter an alphameric entry or DBCS code and convert the entry or code to its related DBCS word. DBCS conversion is intended for Japanese character sets and its use is limited for application to other double-byte character sets.

Specifically, DBCS conversion lets you convert the following:

- A string of alphameric characters to a DBCS word
- English alphameric characters to double-byte alphameric characters
- Alphameric Katakana to double-byte Hiragana and Katakana letters
- A *DBCS code* to its corresponding double-byte character
- A *DBCS number* to its corresponding double-byte character

Where You Can Use DBCS Conversion

You can use DBCS conversion in the following instances:

- When entering data into input fields of certain SEU displays. For information about which fields you can use with DBCS conversion, refer to the *SEU User's Guide/Reference*.

- When prompting for double-byte data using QCMDEXEC. For instructions on this procedure, see the *CL Reference*.

- When entering data into input fields of DBCS display files in user-written applications. Specify DBCS conversion with the DDS keyword IGCCNV. See the *DDS Reference* for information on this keyword.

- When editing the related words on the Edit Related Words display, which is displayed when editing the DBCS conversion dictionary (EDTIGCDCT command). See "Editing a DBCS Conversion Dictionary" on page B-32 for information on the Edit Related Words display.

How DBCS Conversion Works

DBCS conversion is an interactive function between you and the system in which you enter an alphameric entry. The system displays related DBCS words, and you choose which word to use.

The system determines which words are related to an alphameric entry by checking DBCS conversion dictionaries. The system checks two DBCS conversion dictionaries when performing DBCS conversion. It checks the first user-created dictionary named QUSRIGCDCT found when searching a user's library list. Then, it checks the system-supplied dictionary, QSYSIGCDCT, stored in the library QSYS. (QSYSIGCDCT contains only Japanese double-byte characters.) You can create other user dictionaries, and you can give them names other than QUSRIGCDCT, but the system only refers to the first user-created dictionary named QUSRIGCDCT found in your library list when performing DBCS conversion.

After checking the dictionaries, the system displays words related to the alphameric entry. You then position the cursor under the word of your choice, and press the Enter key. The system enters that word where the cursor was positioned when you began DBCS conversion.

Using DBCS Conversion

You can change the user-defined dictionary used during DBCS conversion. Before you change the user-defined dictionary, end your application program or end the command that the system is performing. Then change the dictionary that is used by changing the library list (using the CHGLIBL command).

You can create your own DBCS conversion dictionary for DBCS conversion. The system-supplied dictionary is a collection of entries with a Japanese pronunciation expressed in alphameric characters and Japanese DBCS words related to the entry. See "Creating a DBCS Conversion Dictionary" on page B-31 for instructions on this procedure.

If no user-created dictionary is found, the system refers only to QSYSIGCDCT. See "DBCS Conversion Dictionaries" on page B-30 for more information on creating and using DBCS conversion dictionaries.

Performing DBCS Conversion

The following procedure describes how to convert one alphameric entry to its related DBCS word using DBCS conversion. You must start DBCS conversion separately for each field in which you want to enter double-byte data.

Note: DBCS conversion is intended for Japanese data entry. Its use with other languages is limited.

While performing DBCS conversion, you can display information about the function by pressing the Help key. Help is available until you end DBCS conversion.

1. Position the cursor in the field in which you want to enter double-byte characters. Insert shift control characters into the field if they have not yet been inserted. To find out how to insert shift characters, see "Inserting-Shift Control Characters" on page B-15.
2. Position the cursor under the shift-in character, in a blank area between the shift control characters, or under a double-byte character.
3. Press the function key used to start DBCS conversion.

In SEU, as well as from the Edit Related Words display (displayed when using the EDTIGCDCT command), press F18. The system displays the following prompt line:

```
_ __________ _
A     B      C
```

4. Enter the following values:

   a. In the field marked A, enter one of the following:

      I Inserts the converted word before the character under which you positioned the cursor in step 2 on page B-39.

      R Replaces the character under which you positioned the cursor in step 2 on page B-39 with the converted word.

   b. In the field marked B, enter one of the following:

      1) A string of alphameric characters to be converted. The string can have as many as 12 characters.

      2) The 4-character DBCS code of a double-byte character.

      3) The 2- to 5-digit DBCS number of a double-byte character.

   c. In the field marked C enter one of the following conversion codes:

      No entry Converts the entry in field B from alphameric to double-byte by referring to the DBCS conversion dictionaries.

      G Converts the 2- to 5-digit DBCS number in field B to the character it represents.

      H Converts the entry in field B to double-byte Hiragana, uppercase alphabetic, numeric, or special characters.

      K Converts the entry in field B to double-byte Hiragana, lowercase alphabetic, numeric, or special characters.

      X Converts the 4-character DBCS code to the character it represents.

5. Press the Enter key. The system displays the following prompt line:

```
_ __________ _ ______________________________+
A     B      C               D
```

6. In the field marked D, the system displays words related to the entry in field B.

   If you see a plus (+) sign following the last displayed word, the system has additional words to display. Press the Roll Up key to see these entries. Then, to return to a word displayed earlier, press the Roll Down key.

   If a word is shown in a reverse image, the word contains an embedded blank.

7. Choose the DBCS word from field D that best suits your needs by positioning the cursor under that DBCS word.

8. Press the Enter key. The system enters the word where the cursor was positioned in step 2, either by inserting the word or by replacing another word, depending on what you entered in field A.

9. Do one of the following:

   a. Continue using DBCS conversion. Repeat 4 on page B-40 through 8 on page B-40 until you finish entering data into the field.

   b. End DBCS conversion by pressing the *same* function key used to start conversion. The system automatically ends conversion when you reach the end of the field.

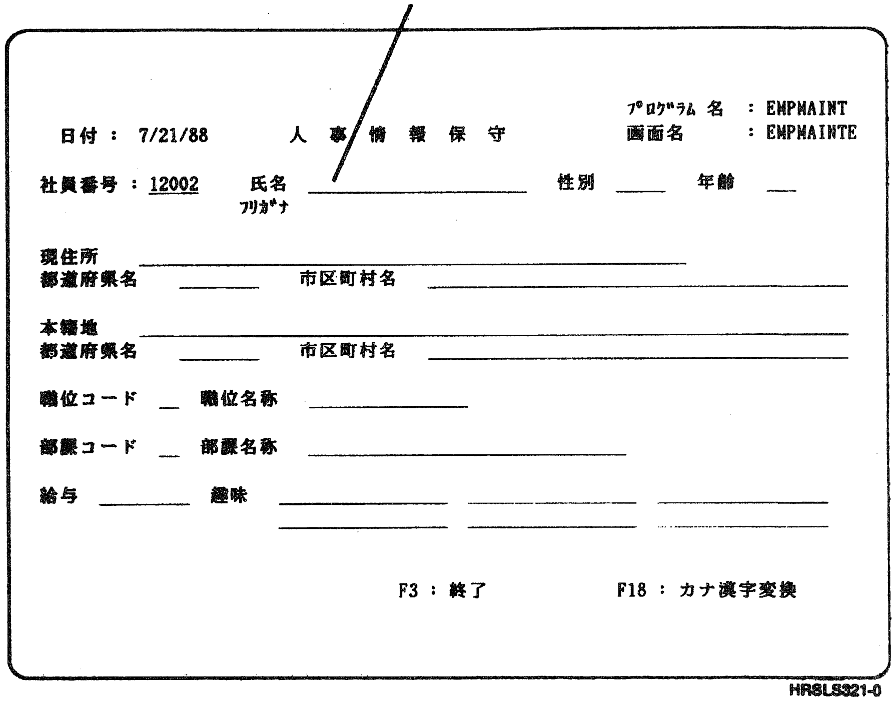
      In SEU, as well as from the Edit Related Words display (displayed when using the EDTIGCDCT command), press F18.

      Note: Until DBCS conversion is ended, you cannot perform any other system function. For example, the F3 key cannot be used to exit an SEU display.

Examples

Converting One Alphameric Entry to a Double-Byte Entry: The following example shows how to convert one entry and enter it into a field.

1. Position the cursor in the field in which you want to enter double-byte data.



Position the cursor here.

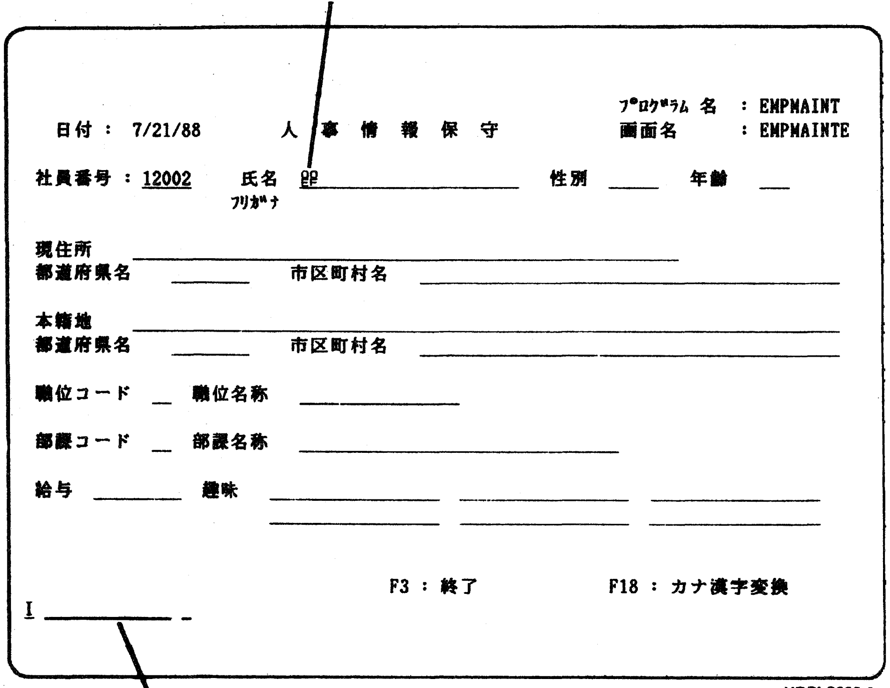
```
                                                    ﾌﾟﾛｸﾞﾗﾑ 名   : EMPMAINT
 日付 :  7/21/88       人 事 情 報 保 守            画面名       : EMPMAINTE

社員番号 : 12002    氏名 ____________________   性別 ____   年齢 __
                   ﾌﾘｶﾞﾅ

現住所      ______________________________________________
都道府県名   _______     市区町村名 _______________________________________

本籍地      _____________________________________________________________
都道府県名   _______     市区町村名 _______________________________________

職位コード __  職位名称  _______________

部課コード __  部課名称  ______________________________

給与 ________   趣味  _______________  _______________  _______________
                      _______________  _______________  _______________

                                  F3 : 終了          F18 : カナ漢字変換
```

HRSLS321-0

2. Insert shift control characters into the field. See “Inserting-Shift Control Characters” on page B-15 for instructions on inserting shift control characters.

3. Press the function key used to start DBCS conversion. For the display just shown, the function key is F18. The system displays a prompt line.

Notice that shift control characters have been inserted into the field.



```
                                                    ﾌﾟﾛｸﾞﾗﾑ 名   : EMPMAINT
  日付 :  7/21/88       人 事 情 報 保 守          画面名       : EMPMAINTE

 社員番号 : 12002     氏名 ____________________   性別 _____   年齢 ___
                 ﾌﾘｶﾞﾅ

 現住所   __________________________________________________
 都道府県名   _______     市区町村名   _______________________________________

 本籍地   ____________________________________________________________________
 都道府県名   _______     市区町村名   _______________________________________

 職位コード __  職位名称   _______________

 部課コード __  部課名称   ______________________________

 給与 ________  趣味  _______________  _______________  _______________
                      _______________  _______________  _______________

                          F3 : 終了              F18 : カナ漢字変換
 I ____________ _
```

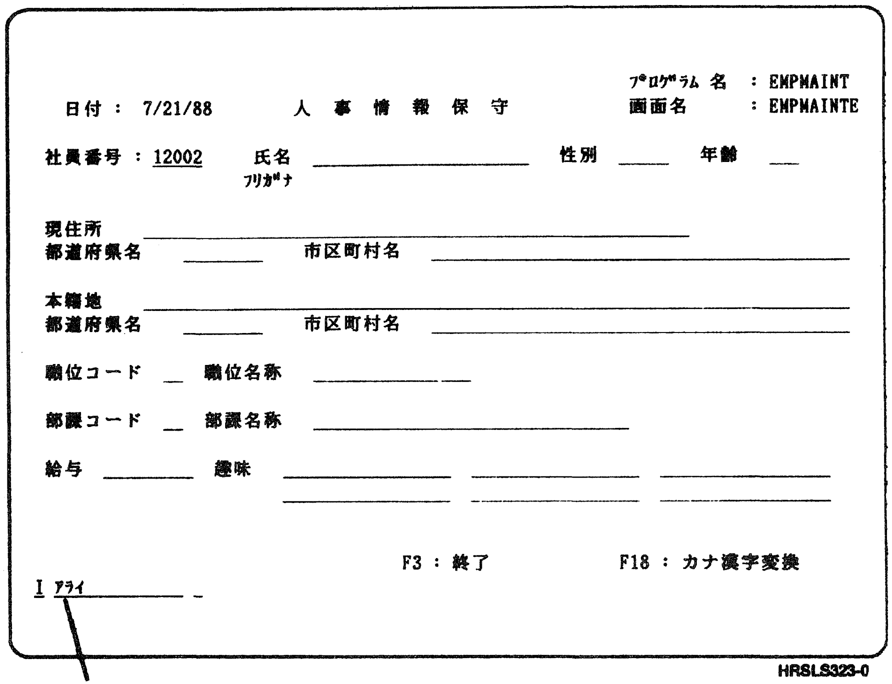
The Prompt Line.

HRSLS322-0

Because the cursor was placed under a shift-in character when conversion was started, conversion automatically is set to I (inserting the converted word).

4. Enter an alphameric entry to be converted in the second field.

   Leave the third field blank.



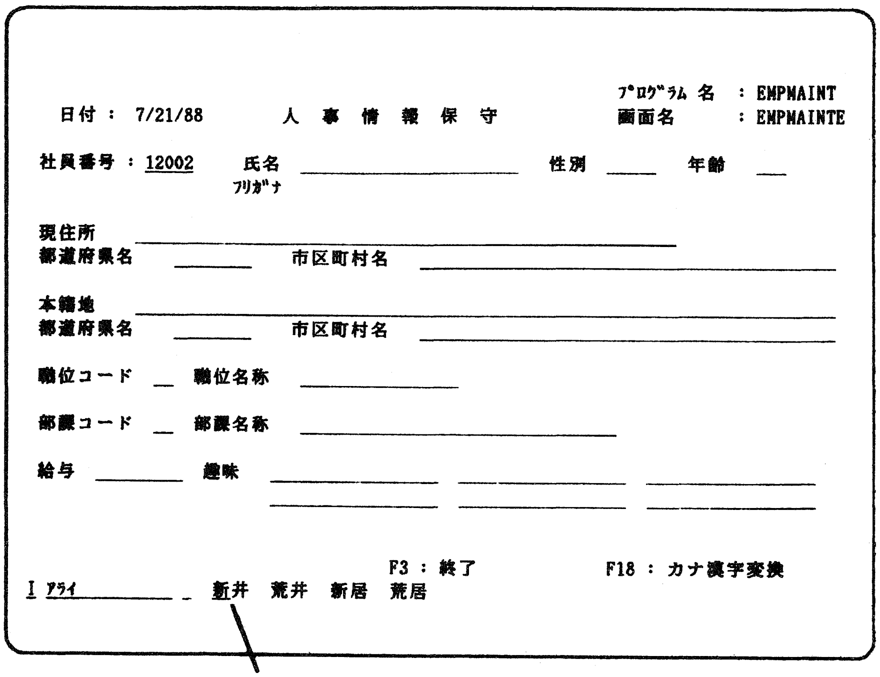
```
                                                      ﾌﾟﾛｸﾞﾗﾑ 名  : EMPMAINT
  日付 :  7/21/88        人 事 情 報 保 守            画面名      : EMPMAINTE

社員番号 : 12002    氏名 ____________________   性別 _____   年齢 __
                  ﾌﾘｶﾞﾅ

現住所 ____________________________________________________
都道府県名    _______     市区町村名  _______________________________________

本籍地 ____________________________________________________________________
都道府県名    _______     市区町村名  _______________________________________

職位コード __  職位名称   _______________

部課コード __  部課名称   ______________________________

給与 ________   趣味   ________________  ________________  ________________
                      ________________  ________________  ________________


                                  F3 : 終了             F18 : カナ漢字変換
I ｱﾗｲ__________ _
```

HRSLS323-0

Enter an alphameric entry here.

5. Press the Enter key. The system displays related DBCS words.

6. Position the cursor under the DBCS word that you want to enter, if that word is not the first DBCS word shown. In the following example, the first word is the one to be entered.



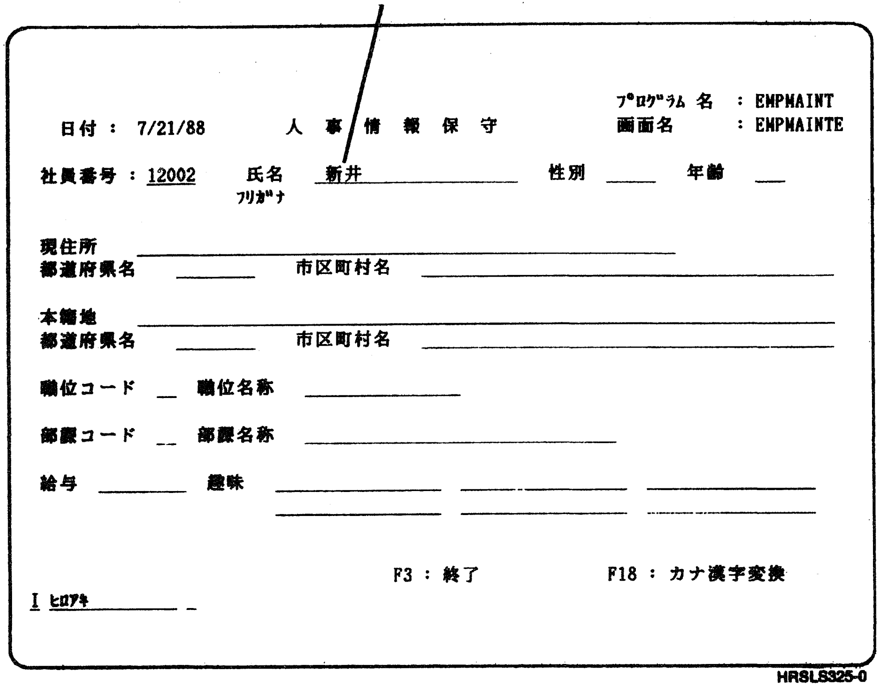
```
                                                        ﾌﾟﾛｸﾞﾗﾑ 名  : EMPMAINT
  日付 :  7/21/88        人 事 情 報 保 守              画面名      : EMPMAINTE

社員番号 : 12002     氏名 ________________        性別 ____     年齢 __
                    ﾌﾘｶﾞﾅ

現住所
部道府県名     ______     市区町村名   ________________________________

本籍地
部道府県名     ______     市区町村名   ________________________________

職位コード __ 職位名称   ____________

部課コード __ 部課名称   ________________________

給与 ______   趣味   _____________  _____________  _____________
                     _____________  _____________  _____________

                                  F3 : 終了              F18 : カナ漢字変換
I ｱﾗｲ__________ _  新井  荒井  新居  荒居
```

Position the cursor here.

HRSLS324-0

7. Press the Enter key. The DBCS word is entered into the field.

The system enters the word into the field.



```
                                                        ﾌﾟﾛｸﾞﾗﾑ 名   : EMPMAINT
 日付 :  7/21/88      人  事  情  報  保  守           画面名       : EMPMAINTE

社員番号 : 12002    氏名   新井___________    性別 ____   年齢 __
                  ﾌﾘｶﾞﾅ

現住所   ________________________________________
都道府県名   ______     市区町村名  ____________________________

本籍地   ____________________________________________________
都道府県名   ______     市区町村名  ____________________________

職位コード __  職位名称   ____________

部課コード __  部課名称   ________________________

給与 _______   趣味   ____________  ____________  ____________
                     ____________  ____________  ____________


                          F3 : 終了            F18 : カナ漢字変換
I ﾋﾛｱｷ__________ _
```

HRSLS325-0

Converting Many Alphameric Entries at One Time: You do not have to continually start DBCS conversion for each alphameric entry. Instead, you can do the following:

1. Enter as many alphameric entries as will fit into field B. Separate each entry by a blank. Field B contains space for 12 alphameric characters:

```
These are the entries to be converted.
  |   |   |
I XXX_YYY_ZZZ_ _

A      B       C                  D
```

   The system converts the entries one at a time, in the order entered. When the system converts an entry, the system displays the DBCS words related to that entry in field D.

2. Position the cursor under the DBCS word that you want to use.

3. Press the Enter key. Then, the system adjusts field B; the next entry is moved to the position farthest left of the field. The DBCS words related to that entry are displayed in field D.

   At this time, you can enter additional alphameric entries to be converted at the end of field B.

Converting Alphameric Blanks to DBCS Blanks: You can convert alphameric blanks (one position wide) to DBCS blanks (two positions wide, the same width as double-byte characters) using DBCS conversion.

To convert blanks, do the following:

1. Enter one or more blanks in field B.

```
_ __________ _

A     B      C              D
```

2. Press the Enter key. The system displays in field D the same number of DBCS blanks as the alphameric blanks that you entered in field B. The DBCS blanks are displayed in reverse image.
3. Press the Enter key again. The system enters the DBCS blanks into the field where you started DBCS conversion.

Changing Alphameric Entries or Conversion Code: If none of the related words shown during conversion are suitable candidates for the alphameric entry, and you would like to try a conversion again (by using a different type of conversion or a different alphameric entry), do the following:

1. Move the cursor to field B. For example:

```
Move the cursor here.
  |
  XXXXXX
_ __________ _ ______________________

A     B      C          D
```

2. Do one of the following:
   a. Position the cursor under the first position of the field in which you want to enter alphameric entries.
   b. Enter a different alphameric entry.
   c. Change the conversion code in field C, such as from H to K.
3. Press the Enter key.
4. Continue DBCS conversion.

Using DBCS Conversion to Enter Words in the DBCS Conversion Dictionary: You can use DBCS conversion when entering DBCS words on the Edit Related Words display.

To start DBCS conversion, do the following:

1. Position the cursor at the position where the DBCS word is to be entered.
2. Press F18. The system displays the conversion prompt line at the bottom of the display.

Perform DBCS conversion according to the instructions described in "Performing DBCS Conversion" on page B-39.

Note: You must start and end DBCS conversion separately for each line of data.

Considerations for Using DBCS Conversion

Consider the following when performing DBCS conversion:

- You can only perform DBCS conversion at a DBCS display station, using the 5556 Katakana keyboard.
- You may use DBCS conversion to insert or replace characters only if the line in which double-byte characters are to be inserted has sufficient space.
  - The space available for inserting characters is equal to the number of characters from the last character on the line that is not blank to the right edge of the display.
  - The space available for replacing characters is equal to the number of characters from the cursor position (including the character marked by the cursor) to the end of the DBCS portion of the field.

  The following happens when you do not have enough space:

  - If you try to insert or replace a string of characters where there is no space available, the system sends a message.
  - If you ignore the message and press the Enter key again, the system truncates the characters in excess of the limit from the right side of the string to be inserted or replaced.

Appendix C. Display File Return Codes

This section contains descriptions of all major and minor return codes for display files. These return codes are set in the I/O feedback area of the display file. The return codes report the results of each operation. The appropriate return code is available to the application program that issued the operation. The program then checks the return code and acts accordingly. Refer to your high-level language manual for information about how to access these return codes.

The return code is a four-digit value: the first two digits contain the major code, and the last two digits contain the minor code. With some return codes, a message is also sent to the job log or the operator message queue (QSYSOPR). You can refer to the message for additional information.

Major Code 00

Major Code 00 – Operation completed successfully.

Description: The operation issued by your program completed successfully.

Action: Continue with the next operation.

Code **Description/Action**

0000 **Description:** For input operations performed by your program, 0000 indicates that some data was received on a successful input operation.

For output operations performed by your program, 0000 indicates that the last output operation completed successfully.

Action: Your program may continue. One of the messages listed below may have been issued to warn of an unusual condition that may be significant to your program even though it is not an error.

Messages:

```
CPF4018 (Status)
CPF4019 (Diagnostic)
CPF4054 (Diagnostic)
CPF4082 (Diagnostic)
CPF4410 (Diagnostic)
CPF5003 (Status)
CPF5508 (Diagnostic)
```

Major Code 02

Major Code 02 – Input operation completed successfully, but your job is being ended (controlled).

Description: The input operation issued by your program was completed successfully. However, your job is being ended (controlled).

Action: Your program should complete its display processing as soon as possible to allow your program to complete in an orderly manner. The system eventually changes a job ended (controlled) to a job ended (immediate) and forces all processing to stop for your job.

Code	Description/Action
0200	**Description:** On a successful input operation, an indication was received that a job ended (controlled) request is pending. Also, 0200 indicates that some data was received. **Action:** Your program may continue. However, the recommended action is to complete the display processing and end the program, because the system will eventually cancel your job and cause all processing to stop for your job.

Major Code 03

Major Code 03 – Input operation completed successfully, but no data received.

Description: The input operation was completed successfully, but no data was received.

Action: Check the minor return code for additional information, and continue with the next operation.

Code **Description/Action**

0300 **Description:** No data was received on a successful input operation. Examples of conditions causing this are no data being available on a get-relative or a get-next-changed operation to a subfile record format.

Action: Continue with whatever processing is appropriate. For example, if your program issued a get-next-changed operation to a subfile record format, then a 0300 indicates that there are no more subfile records changed by the user and no more user input data to process.

Messages:

```
CPF5017 (Notify)
CPF5020 (Notify)
CPF5037 (Notify)
```

| **0309** **Description:** Your program is being ended (controlled). No data was received.

This return code is only applicable to the read-from-invited-devices operation.

Action: Your program can continue processing. However, the recommended action is to complete the display processing and end the program, because the system will eventually cancel your job and cause all processing to stop for your job.

Messages:

```
CPF4741 (Notify)
```

0310 **Description:** The time interval specified by the WAITRCD value for the display file has ended.

This return code is only applicable to the read-from-invited-devices operation.

Note: Because no device is associated with the completion of this operation, the device name in the I/O feedback area contains an *N.

Action: Issue the operation to perform the intended function after the specified time interval has ended. For example, if you are using the time interval to control the length of time to wait for data, you can then issue another read-from-invited-devices operation to receive the data.

Messages:

```
| CPF4742 (Status)
| CPF4743 (Status)
```

Major Code 04

Major Code 04 – Output exception occurred.

Description: An output exception occurred because your program attempted to send output when it should have been receiving the data that had already been sent by the display. Your data, associated with this output operation, was not sent to the display. Your program can attempt to send its output later.

Action: Issue an input operation to receive the data.

Code **Description/Action**

0412 **Description:** An output exception occurred because your program attempted to send data when it should have been receiving the data that had already been sent by the display. Your program data was not sent and should be sent later, after the data from the display has been received.

Action: Issue an input operation to receive the data.

Messages:

```
CPF4737 (Notify)
```

Major Codes 08-11

Major Codes 08 and 11 – Miscellaneous program errors occurred.

Description: The operation just attempted by your program was not successful. The operation may have failed because it was issued at the wrong time.

Action: Refer to the individual return code descriptions for the appropriate recovery actions.

Code **Description/Action**

0800 **Description:** The acquire operation just performed was not successful. It tried to acquire a device that had already been acquired.

Action: If the display device requested by the original acquire operation is the one needed, your program can begin using it because it is already available. If a different device is required, issue another acquire operation for a different device name.

Messages:

```
CPD4077 (Diagnostic)
CPF50A0 (Status)
```

1100 **Description:** The read-from-invited-devices operation was not successful because no devices were invited.

Action: Issue an invite function followed by a read-from-invited-devices operation.

Messages:

```
CPF4740 (Notify)
```

Major Code 34

Major Code 34 – Input exception occurred.

Description: The input operation attempted by your program was not successful. The data received was too long for the record format specified on the input operation.

Action: Refer to the individual return code descriptions for the appropriate recovery actions.

Code **Description/Action**

3431 **Description:** The input operation issued by your program was not successful because the length of the data received from the display exceeds the receive data length specified in the user-defined data stream. The data received is truncated.

This return code is only applicable to input operations using a record format specifying a user-defined data stream (USRDFN DDS keyword).

Action: Close the device file and end your program. Then, change your program so that the input record length is at least as long as the data record to be received.

Messages:

```
CPF5062 (Notify)
```

Major Code 80

Major Code 80 – Permanent system or file error (nonrecoverable).

Description: A nonrecoverable file or system error has occurred. Recovery is unlikely until the problem causing the error has been corrected.

Action: The following general actions can be taken by your program for each 80xx return code. Other specific actions are given in each return code description.

- Continue processing without the display.
- Close the device file and open the file again. If the operation is still unsuccessful, try it again only a limited number of times. (The number of times should be specified in your program.)
- End.

Code Description/Action

8081 **Description:** The operation was not successful because a system error condition was detected.

Action: Your display device may need to be varied off and then on again. Your program can either:

- Continue processing without the display device.
- Close the device file and open the file again.
- End.

Messages:

CPF4182 (Escape)
CPF4510 (Escape)
CPF5192 (Escape)
CPF5257 (Escape)
CPF5403 (Escape)
CPF5404 (Escape)
CPF5405 (Escape)
CPF5408 (Escape)
CPF5409 (Escape)
CPF5410 (Escape)
CPF5411 (Escape)
CPF5414 (Escape)
CPF5415 (Escape)
CPF5416 (Escape)
CPF5418 (Escape)
CPF5423 (Escape)
CPF5429 (Escape)
CPF5431 (Escape)
CPF5433 (Escape)
CPF5434 (Escape)
CPF5441 (Escape)
CPF5447 (Escape)
CPF5455 (Escape)
CPF5456 (Escape)
CPF5507 (Escape)

8082 **Description:** The operation was not successful because the display device is unusable. This may occur because a cancel reply has been taken to an error recovery message for the device or because the display has been held by a Hold Communications Device (HLDCMNDEV) command. No operations should be issued to the device.

Action: Communications with the display cannot be resumed until the device has been reset to a varied-on state. If the device has been held, use the Release Communications Device (RLSCMNDEV) command to reset the device. If the device is in an error state, vary the device off and then on again. Once the device is reset, normal operation can be started by opening the display device file again. Your program can either:

- Continue processing without the display device.
- Close the device file and open the file again.
- End.

Messages:

```
CPF4354 (Escape)
CPF5269 (Escape)
```

80A6 **Description:** A Systems Network Architecture (SNA) unbind operation was not successful on a close or release operation. This may be the result of a device configuration error. The device may be unusable. No operations should be issued to the device.

Action: Refer to the device response code in the accompanying error message to determine the cause of the failure. Vary the device off and then on again to reset the error. Correct the error and try your program again.

Messages:

```
CPF4527 (Escape)
```

80B3 **Description:** The open operation was not successful because the display file is not available. The file cannot be opened again until the necessary resources are available.

Action: Your program can wait for the display file to become available, then issue another open operation. Otherwise, you may continue other processing or end the program.

Consider increasing the WAITFILE parameter with the Change Display File (CHGDSPF) command or Override Display File (OVRDSPF) command to allow more time for the file to become available.

Messages:

```
CPF4128 (Escape)
```

80C0 **Description:** A nonrecoverable error has occurred on the display device.

Action: Your display devices may need to be varied off and then on again. Your program can either:

- Continue processing without the display station.
- Close the device file and open the file again.
- End.

Messages:

CPF5103 (Escape)
CPF5192 (Escape)
CPF5412 (Escape)
CPF5413 (Escape)
CPF5419 (Escape)
CPF5420 (Escape)
CPF5421 (Escape)
CPF5430 (Escape)
CPF5437 (Escape)
CPF5439 (Escape)

80EB **Description:** An open operation was not successful because an open option that was not valid or an invalid combination of options was specified in your program, in the display file, or in an override command.

Action: Close the display file, correct the problem, and open the file again. See the individual messages to determine what options are not valid.

Messages:

CPF4062 (Escape)
CPF4129 (Escape)
CPF4148 (Escape)
CPF4156 (Escape)
CPF4163 (Escape)
CPF4169 (Escape)
CPF4191 (Escape)
CPF4238 (Escape)
CPF4345 (Escape)
CPF5151 (Escape)
CPF5510 (Escape)
CPF5511 (Escape)
CPF5512 (Escape)
CPF5513 (Escape)
CPF5552 (Escape)

80ED **Description:** The open operation was not successful because the record format descriptions in the file have changed since your program was compiled.

Action: Close the file and end the program. Determine whether the changes affect your application program. If they do, recompile the program. If the changes do not affect your program, the file should be changed or overridden to LVLCHK(*NO). When LVLCHK(*NO) is specified, the system does not compare the record format descriptions.

Messages:

CPF4131 (Escape)

80F8 **Description:** An operation to a file was not successful because the file is marked in error.

Action: Close the file. Refer to messages in the job log to determine what errors occurred. Take the appropriate action for those errors.

Messages:

CPF4132 (Escape)
CPF4213 (Escape)
CPF4550 (Escape)
CPF5129 (Escape)
CPF5144 (Escape)
CPF5427 (Escape)

Major Code 81

Major Code 81 – Permanent device error (nonrecoverable).

Description: A nonrecoverable device-related error occurred during an I/O operation. Any attempt to continue using this display device will probably fail again until the cause of the problem is found and corrected, but operations directed to other display devices associated with the file should be expected to work.

Action: The following general actions can be taken for each 81xx return code. Other specific actions are given in each return code description.

- Continue processing without the display device.
- Release the device or close the file, correct the problem, and acquire the device again or open the file. If the operation is still unsuccessful, try it again only a limited number of times. (The number of times should be specified in your program.)
- End.

Several return codes indicate that an error condition must be corrected by varying the device off and on again.

Code **Description/Action**

8181 **Description:** A system error condition was detected during the I/O operation to the device.

Action: Release the device in error or close the file. You may need to vary the device off and on again to clear the error. Determine the cause of the failure from the accompanying message. Check for any system operator messages indicating that additional corrective action must be performed. Open the file again or acquire the device to continue.

Messages:

```
CPF4553 (Escape)
CPF4725 (Escape)
CPF5105 (Escape)
CPF5189 (Escape)
CPF5254 (Escape)
```

8191 **Description:** The operation was not successful because a permanent line error occurred, and the system operator took a recovery option in response to the line error message. (You can find out what type of line error occurred by asking the system operator.) The device has been marked unusable.

Action: Release the device in error or close the file. Vary the device off and on again to clear the error. Open the file again or acquire the device to continue.

Messages:

```
CPF4526 (Escape)
CPF4542 (Escape)
CPF4551 (Escape)
CPF5128 (Escape)
CPF5143 (Escape)
CPF5198 (Escape)
```

8197 **Description:** A nonrecoverable error condition was detected at the device. An example of such an error is the user turning off the display device.

Action: Release the device in error or close the file. The display device may need to be varied off and then on again to clear the error. Refer to the accompanying error message for additional information regarding the source of the specific error detected. Open the file or acquire the device again to continue.

Messages:

CPF4149 (Escape)
CPF4197 (Escape)
CPF4524 (Escape)
CPF4533 (Escape)
CPF4538 (Escape)
CPF5047 (Escape)
CPF5106 (Escape)
CPF5140 (Escape)
CPF5143 (Escape)
CPF5199 (Escape)
CPF5265 (Escape)

81C2 **Description:** The operation issued by your program was not successful because the Systems Network Architecture (SNA) session with the display is not active.

Action: Release the device or close the file. Vary the device off and on again to clear the error. Open the file or acquire the device again to continue.

Messages:

CPF5170 (Escape)
CPF5422 (Escape)

Major Code 82

Major Code 82 – Open or acquire operation failed.

Description: An attempt to open the display file or acquire the display device was not successful. The error may be recoverable or permanent, but is limited to the specific display device. Recovery is unlikely until the problem causing the error has been corrected.

Action: The following general actions can be taken for each 82xx return code. Other specific actions are given in each return code description. You can either:

- Continue processing without the device.
- Release the device or close the file, correct the problem, and acquire the device or open the file again. A subsequent operation could be successful if the error occurred because of some temporary condition such as the device being in use at the time.

  If the operation is still unsuccessful, try it again only a limited number of times. (The number of times should be specified in your program.)

  - If the attempted operation was an acquire operation, release the device and issue the acquire operation again.
  - If the attempted operation was an open operation, close the file and issue the open operation again.
- End.

Several return codes indicate that an error condition must be corrected by changing a value in the file. To change a parameter value for the file, use the Change Display File (CHGDSPF) or Override with Display File (OVRDSPF) command.

Code **Description/Action**

8281 **Description:** A system error condition was detected on an open or acquire operation. The file may previously have been in error, or the file could not be opened due to a system error.

Action: Determine the cause of the failure from the accompanying message. Check for any system operator messages indicating that additional corrective action must be performed.

The display device may need to be varied off and then on again to clear the error. Your program can either:

- Continue processing without the display device.
- Release the device or close the file, correct the problem, and acquire the device or open the file again.
- End.

Messages:

```
CPF4168 (Escape)
CPF4182 (Escape)
CPF4221 (Escape)
CPF5105 (Escape)
CPF5254 (Escape)
CPF5257 (Escape)
CPF5410 (Escape)
CPF5411 (Escape)
CPF5424 (Escape)
CPF5447 (Escape)
CPF5455 (Escape)
```

8282 **Description:** The open or acquire operation was not successful because the display device is unusable. This may occur because a cancel reply has been taken to an error recovery message for the device or because the display has been held by a Hold Communications Device (HLDCMNDEV) command. No operations should be issued to the device. Communications with the display cannot be resumed until the device has been reset to a varied-on state.

Action: Close the display device file. If the device has been held, use the Release Communications Device (RLSCMNDEV) command to reset the device. If the device is in an error state, vary the device off and then on again. Once the device is reset, start normal operation by opening the display device file again.

Messages:

```
CPF4171 (Escape)
CPF4354 (Escape)
CPF5548 (Escape)
```

8291 **Description:** A permanent line error occurred on an open or acquire operation. The device has been marked unusable.

Action: Release the device in error or close the file. Vary the device off and on again to clear the error. Open the file again to continue.

Messages:

```
CPF4146 (Escape)
CPF4179 (Escape)
CPF4193 (Escape)
CPF4291 (Escape)
CPF5198 (Escape)
CPF5260 (Escape)
```

8297 **Description:** The open or acquire operation has ended abnormally due to a nonrecoverable error condition detected at the display device. An example of such an error is the user turning off the display device.

Action: Release the device in error or close the file. The display device may need to be varied off and then on again to clear the error. Refer to the accompanying error message for additional information regarding the source of the specific error detected. Open the file or acquire the device again to continue.

Messages:

```
CPF4192 (Escape)
CPF5047 (Escape)
CPF5106 (Escape)
CPF5140 (Escape)
CPF5143 (Escape)
CPF5199 (Escape)
```

82A6 **Description:** The open or acquire operation failed because the Systems Network Architecture (SNA) bind command was not successful.

Action: Ensure that the display device with which your program is communicating is configured properly. Refer to the device response codes in the accompanying error message for additional information regarding the specific error detected.

Messages:

```
CPF4124 (Escape)
CPF4190 (Escape)
CPF5103 (Escape)
CPF5517 (Escape)
```

82A8 **Description:** The acquire operation was not successful because the maximum number of devices allowed for the display file has been reached.

Action: Your program can recover by releasing a different device and issuing the acquire operation again. If more devices are needed, close your file and increase the MAXDEV value in the display file.

Messages:

```
CPF4745 (Escape)
CPF5041 (Status)
CPD4757 (Diagnostic)
```

82A9 **Description:** The acquire operation was not successful because the requester device is not available. The requester device may not be available because your program is not running in an interactive job.

Action: Your program can continue without the display, attempt to use a different display device, or end.

If your program needs to use the requester device, make sure that it runs in an interactive job.

Messages:

```
CPF4366 (Escape)
CPF5381 (Escape)
```

82AA **Description:** The open or acquire operation was not successful because the display device description was not found.

Action: Your program can continue without the display, attempt to use a different display device, or end.

Verify that the name of the display device was correctly specified in the DEV parameter on the CRTDSPF, CHGDSPF, CRTDEVDSP, or OVRDSPF command.

Messages:

```
CPF4103 (Escape)
CPF4747 (Escape)
```

82AB **Description:** The open or acquire operation was not successful because the display device was not varied on.

Action: Your program can continue without the display, attempt to acquire a different display device, or end. Vary on the display device and attempt the open or acquire operation again.

Messages:

```
CPF4285 (Escape)
CPF5333 (Escape)
```

82B3 **Description:** The open or acquire operation was not successful because the display device you are acquiring is in use in another process.

Action: Wait for the display device to become available, then issue the acquire operation again. Otherwise, you may continue other processing without the display, or end the program.

You can use the Work with Configuration Status (WRKCFGSTS) command to determine which job is using the display device.

Consider increasing the WAITFILE parameter of the CHGDSPF or OVRDSPF command to allow more time for the device to become available.

Messages:

```
CPF4109 (Escape)
CPF4130 (Escape)
CPF4282 (Escape)
CPF5217 (Escape)
CPF5332 (Escape)
```

82EE **Description:** An open or acquire operation was attempted to a device that is not supported for a display file.

Your program is attempting to acquire a device that is not a valid display device; or it is trying to acquire the requester device, but the requester device for the job is a communications device, not a display device.

Action: Your program can continue without the display, attempt to acquire a different display device, or close the file and end.

Verify that the name of the display device was specified correctly on the CHGDSPF, CRTDEVDSP, CRTDSPF, or OVRDSPF command.

If your program was attempting to acquire the requester device, verify that your program is running in an interactive job so that the requester device is a display device.

Messages:

```
CPF4105 (Escape)
CPF4223 (Escape)
CPF4760 (Escape)
CPF5038 (Escape)
```

82EF **Description:** An open or acquire operation was attempted for a device that the user is not authorized to or that is in service mode.

Action: Your program can continue without the display, attempt to acquire a different display device, or end.

If the operation was an open operation, close the file, correct the problem, and then issue the open operation again. If the operation was an acquire operation, correct the problem and issue the acquire operation again.

For authority errors, obtain authority to the device from your security officer or device owner. If the device is in service mode, the system service tools (SST) function is currently using the device. Wait until the device is available to issue the operation again.

Messages:

```
CPF4104 (Escape)
CPF4186 (Escape)
CPF5278 (Escape)
CPF5279 (Escape)
```

82F8 **Description:** Your program attempted an acquire operation to a device that is marked in error due to a previous error during an I/O or acquire operation.

Action: Release the device or close the file, correct the previous error, and acquire the device or open the file again.

Messages:

```
CPF5293 (Escape)
```

Major Code 83

Major Code 83 – Device error occurred (recoverable).

Description: An error occurred during an I/O operation, but the display device is still usable. Recovery within your program might be possible.

Action: The following general actions can be taken for each 83xx return code. Other specific actions are given in each return code description.

- Continue processing without the display device.
- Correct the problem and continue processing with the display device. If the attempt to recover from the operation is unsuccessful, try it again only a limited number of times. (The number of times should be specified in your program.)
- End.

Several return codes indicate that an error condition must be corrected by changing a value in the display file. To change a parameter value for the file, use the Change Display File (CHGDSPF) or Override with Display File (OVRDSPF) command.

Code Description/Action

830B **Description:** Your program has attempted to issue an input or output operation either before the device was acquired or after it was released.

Your program may have improperly handled a *permanent device* or *acquire failed* error.

Action: Verify that your program tries no input or output operation with a display device that is not connected to the file and that return codes from acquire or I/O operations are handled properly.

Messages:

CPD4079 (Diagnostic)
CPF4739 (Status)
CPF5067 (Escape)
CPF5068 (Escape)
CPF5070 (Escape)
CPF5170 (Escape)
CPF5217 (Escape)

831D **Description:** The operation just attempted by your program was rejected because a parameter was not valid, out of limits, or missing.

Action: Your program can bypass the failing step and continue, or close the file and end. Refer to the accompanying message to determine what parameter was incorrect. Correct the error in your program before attempting to try the operation again.

Messages:

CPF4912 (Notify)
CPF5002 (Notify)
CPF5008 (Notify)
CPF5012 (Notify)
CPF5014 (Notify)
CPF5021 (Notify)
CPF5218 (Escape)
CPF5302 (Escape)
CPF5303 (Escape)
CPF5398 (Escape)

831E **Description:** The operation just issued by your program was not valid or an invalid combination of operations was specified. The error may have been caused by one of the following:

- Either your program issued an operation with an unrecognized code, or the operation specified by the code or DDS keyword is not supported by the display.
- An invalid combination of operations or keywords was requested.
- A user-defined data stream contained a command that was not valid for the display.

Action: Your program can bypass the invalid operation and continue, or close the file and end. Refer to the accompanying message to determine why the operation was rejected. Correct the error in your program before attempting to try the failing operation again.

Messages:

CPF4564 (Escape)
CPF5005 (Notify)
CPF5011 (Notify)
CPF5039 (Notify)
CPF5045 (Notify)
CPF5051 (Notify)
CPF5055 (Notify)
CPF5056 (Notify)
CPF5059 (Notify)
CPF5066 (Notify)
CPF5149 (Escape)

831F **Description:** A length that is not valid was specified on the operation.

On an output operation, your program has tried to send a data record having a length that exceeds the maximum record length allowed for the display device. The data has been truncated.

Action: Issue the output operation again with a smaller output length. The record length for a non-field-level display file cannot exceed the display size. The record length for any display file must be no greater than 32 763 characters.

Messages:

CPF4010 (Diagnostic)
CPF4078 (Diagnostic)

8322 **Description:** The attempted operation is not valid in the current state. Either a write operation was attempted while your program was not in the send state or a subfile operation was attempted when the subfile was not active.

Action: Your program can bypass the operation that is not valid and continue, or close the file and end. Correct the sequence of operations in your program before attempting to run the job again.

Messages:

CPF5013 (Notify)
CPF5060 (Notify)

832D **Description:** Your program attempted an operation that is not valid when an invite operation is outstanding. Once you have issued an invite or get-no-wait operation, another invite operation cannot be issued for the same display device until the first invite has been completed by a read or read-from-invited-devices operation.

Action: Issue an input operation to receive the data that was invited before issuing another invite operation. Otherwise, close the file and end. If a coding error in your program caused the error, correct the sequence of operations in your program.

Messages:

```
CPF5052 (Notify)
```

8343 **Description:** An attempt was made to add another record to a subfile after the subfile was full.

Action: Your program can clear the subfile, or continue without adding more records to the subfile. Otherwise, close the file and end.

Increase the subfile length in the DDS statements. If a coding error in your program caused the error, correct your program.

Messages:

```
CPF5043 (Notify)
```

83E0 **Description:** Your program attempted to issue an operation using a record format that was not defined for the display file, or omitted the record format name.

Action: Check the name of the record format in your program to be sure it is correct. Then check that the record format is defined properly in the DDS for the file.

Messages:

```
CPF5022 (Notify)
CPF5023 (Notify)
CPF5053 (Notify)
CPF5054 (Notify)
```

83E1 **Description:** An error occurred during an I/O operation. The requester device was set to automatically disconnect your program. The same user signed on to the same display device, so the program was started up again. The display was cleared of any data present when the error occurred.

Action: Branch to a normal starting point in your program and rewrite the display. There is no need to perform a close or open operation on the display files that were active when the error occurred.

Messages:

```
CPF509F (Notify)
```

83E8 **Description:** Your CL program issued an End Receive (ENDRCV) command when there is no outstanding read-with-no-wait operation.

Action: Your program can issue an output operation to continue sending, issue an input operation to begin receiving, or close the file and end. Correct the error in your program before attempting to repeat the failing operation.

Messages:

```
CPF4910 (Notify)
```

83F6 **Description:** Your program sent data to the display that is not valid. The data type may not be correct for the field in which it is used.

Action: Check the name of the record format in your program to be sure it is correct. Verify that the data definition statements in your program match the output record defined in the DDS for the file. Correct the error in your program before attempting to repeat the failing operation.

Messages:

```
CPF5063 (Notify)
CPF5216 (Escape)
CPF5301 (Escape)
```

83F8 **Description:** Your program attempted an I/O operation to a device that is marked in error due to a previous error during an I/O or acquire operation.

Action: Release the device or close the file, correct the previous error, and acquire the device or open the file again.

Messages:

```
CPF5293 (Escape)
```

Appendix D. Printer File Return Codes

This section contains descriptions of all major and minor return codes for printer files. These return codes are set in the I/O feedback area of the printer file. Return codes report the results of each operation. The appropriate return code is available to the application program that issued the operation. The program then checks the return code and acts appropriately. Refer to your high-level language manual for information about how to access these return codes.

The return code is a four-digit value: the first two digits contain the major code, and the last two digits contain the minor code. With some return codes, a message is also sent to the job log or the operator message queue (QSYSOPR). You can refer to the message for additional information. Message IDs followed by an asterisk (*) may be received by applications while spooling output.

Major Code 00

Major Code 00 – Operation completed successfully.

Description: The operation issued by your program completed successfully.

Action: Continue with the next operation.

Code **Description/Action**

0000 **Description:** For output operations performed by your program, 0000 indicates that the last output operation completed successfully.

The notify messages are used after certain error conditions to give the operator the choice of continuing or canceling the printing of that file. If the reply is CANCEL, another message is issued with a nonzero return code.

Action: Your program may continue. One of the following diagnostic messages may have been issued to warn of an unusual condition that may be significant to your program even though it is not an error.

Messages:

CPA4001 (Inquiry)
CPA4003 (Inquiry)
CPA4004 (Inquiry)
CPA4005 (Inquiry)
CPA4007 (Inquiry)
CPA4008 (Inquiry)
CPA4009 (Inquiry)
CPA4010 (Inquiry)
CPA4011 (Inquiry)
CPA4012 (Inquiry)
CPA4013 (Inquiry)
CPA4014 (Inquiry)
CPA4015 (Inquiry)
CPA4017 (Inquiry)
CPA4019 (Inquiry)
CPA4037 (Inquiry)
CPA4038 (Inquiry)
CPA4039 (Inquiry)
CPA4040 (Inquiry)
CPA4042 (Inquiry)
CPA4043 (Inquiry)
CPA4046 (Inquiry)
CPA4047 (Inquiry)
CPA4048 (Inquiry)
CPA4065 (Inquiry)
CPA4066 (Inquiry)
CPA4072 (Inquiry)*
CPA4073 (Inquiry)
CPA4074 (Inquiry)
CPA4075 (Inquiry)
CPA4076 (Inquiry)
CPA4251 (Inquiry)
CPA4256 (Inquiry)
CPA5335 (Inquiry)
CPA5339 (Inquiry)
CPA5340 (Inquiry)
CPA5341 (Inquiry)
CPA5342 (Inquiry)
CPA5343 (Inquiry)
CPA5344 (Inquiry)
CPA5347 (Inquiry)
CPA5348 (Inquiry)
CPD4005 (Diagnostic)
CPD4006 (Diagnostic)
CPD4007 (Diagnostic)
CPD4008 (Diagnostic)
CPD4069 (Diagnostic)
CPD4071 (Diagnostic)*
CPD4072 (Diagnostic)
CPF4032 (Diagnostic)
CPF4033 (Diagnostic)
CPF4056 (Diagnostic)
CPF4057 (Diagnostic)
CPF4239 (Escape)
CPF4245 (Escape)
CPF4249 (Escape)
CPF4260 (Escape)*
CPF4420 (Diagnostic)
CPF4421 (Diagnostic)
CPF4905 (Notify)*
CPF4913 (Diagnostic)
CPF4914 (Diagnostic)
CPF4916 (Notify)*
CPF4918 (Notify)*
CPF4919 (Notify)*
CPI4015 (Informational)
CPI4016 (Informational)
CPI4017 (Informational)
CPI4018 (Informational)
CPI4019 (Informational)
CPI4020 (Informational)
CPI4024 (Informational)

Major Code 80

Major Code 80 – Permanent system or file error (nonrecoverable).

Description: A nonrecoverable file or system error occurred. Recovery is unlikely until the problem causing the error has been corrected.

Action: The following general actions can be taken by your program for each 80xx return code. Other specific actions are given in each return code description.

- Continue processing without the printer.
- Close the printer file and open the file again.
- End.

Code **Description/Action**

8081 **Description:** The operation was not successful because a system error condition was detected.

Action: Your printer may need to be varied off and then on again. Your program can either:

- Continue processing without the printer.
- Close the device file and open the file again.
- End.

Messages:

CPF4182 (Escape)*
CPF4289 (Escape)
CPF4510 (Escape)*
CPF4516 (Escape)
CPF4552 (Escape)
CPF4591 (Escape)
CPF5159 (Escape)
CPF5196 (Escape)
CPF5246 (Escape)
CPF5257 (Escape)*
CPF5261 (Escape)
CPF5262 (Escape)*
CPF5401 (Escape)
CPF5408 (Escape)
CPF5409 (Escape)
CPF5410 (Escape)
CPF5414 (Escape)
CPF5416 (Escape)
CPF5418 (Escape)
CPF5423 (Escape)
CPF5429 (Escape)
CPF5431 (Escape)*
CPF5433 (Escape)
CPF5434 (Escape)
CPF5447 (Escape)
CPF5453 (Escape)
CPF5507 (Escape)

8082 **Description:** The operation attempted was not successful because the printer is unusable. This may occur because:

- A cancel reply has been taken to an error recovery message for the device.
- A cancel reply was returned to a maximum records reached inquiry message.
- The printer has been held by a Hold Communications Device (HLDCMNDEV) command.

No operations should be issued to the device.

Action: Communications with the printer cannot be resumed until the device has been reset to a varied-on state. If the device has been held, use the Release Communications Device (RLSCMNDEV) command to reset the device. If the device is in an error state, vary the device off and then on again. Once the device is reset, normal operation can be started again by reopening the printer file.

Messages:

```
CPF4502 (Escape)
CPF5104 (Escape)
CPF5116 (Escape)*
CPF5269 (Escape)
```

80B3 **Description**: The open operation was not successful because the printer file or printer device is not available.

Action: The printer file cannot be opened again until the necessary resources are available. Your program can wait for the resources to become available, then issue another open operation. Otherwise, you may continue other processing or end the program. The Work with Configuration Status (WRKCFGSTS) command may be used to determine whether the printer device is in use or not varied on. If the device is in use, the WRKCFGSTS command will also identify the job that is using it.

Consider increasing the WAITFILE parameter with the Change Printer File (CHGPRTF) or Override with Printer File (OVRPRTF) command to allow more time for the file resources to become available.

Messages:

```
CPF4128 (Escape)*
CPF9808 (Diagnostic)*
```

80C0 **Description**: A nonrecoverable error has occurred on the printer device.

Action: Your printer may need to be varied off and then on again. Your program can either:

- Continue processing without the printer.
- Close the printer file and open the file again.
- End.

Messages:

CPF4262 (Escape)
CPF4509 (Escape)
CPF5103 (Escape)
CPF5247 (Escape)
CPF5412 (Escape)
CPF5413 (Escape)
CPF5419 (Escape)
CPF5420 (Escape)
CPF5430 (Escape)
CPF5437 (Escape)

80EB **Description**: An open operation was not successful because an open option that was not valid or an invalid combination of options was specified in your program, in the printer file, or in an override command.

Action: Close the printer file, correct the problem, and issue the open operation again. See the individual messages to determine what options are not valid.

Messages:

CPD4012 (Diagnostic)
CPD4013 (Diagnostic)*
CPD4020 (Diagnostic)
CPD4021 (Diagnostic)*
CPD4024 (Diagnostic)
CPD4025 (Diagnostic)
CPD4033 (Diagnostic)
CPD4034 (Diagnostic)*
CPD4036 (Diagnostic)*
CPD4037 (Diagnostic)*
CPD4038 (Diagnostic)*
CPF4133 (Escape)
CPF4138 (Escape)*
CPF4139 (Escape)*
CPF4148 (Escape)
CPF4156 (Escape)
CPF4157 (Escape)*
CPF4159 (Escape)*
CPF4162 (Escape)
CPF4181 (Escape)*
CPF4196 (Escape)*
CPF4206 (Escape)*
CPF4209 (Escape)
CPF4214 (Escape)*
CPF4217 (Escape)
CPF4219 (Escape)
CPF4224 (Escape)
CPF4237 (Escape)*
CPF4238 (Escape)
CPF4263 (Escape)*
CPF4264 (Escape)*
CPF4295 (Escape)*
CPF4296 (Escape)*
CPF4335 (Escape)
CPF4336 (Escape)
CPF4337 (Escape)
CPF4338 (Escape)
CPF4339 (Escape)*
CPF4340 (Escape)
CPF4345 (Escape)
CPF4352 (Escape)
CPF4637 (Escape)
CPF5370 (Escape)

80ED **Description**: An open operation was not successful because the record format descriptions in the printer file have changed since your program was compiled.

Action: Close the printer file and end the program. Determine whether the changes affect your application program. If they do, then recompile the program. If the changes do not affect your program, the file should be changed or overridden to LVLCHK(*NO). When LVLCHK(*NO) is specified, the system does not compare the record format descriptions.

Messages:

```
CPF4131 (Escape)*
```

80EF **Description**: An open operation was not successful because your program is not authorized to the printer device.

Action: Close the file, correct the problem, then issue the open operation again. Obtain authority to the device from your security officer or the device owner.

Messages:

```
CPF4104 (Escape)*
```

80F8 **Description:** An operation was not successful because the file is marked in error.

Action: Close the file. Refer to messages in the job log to determine what errors occurred. Take the appropriate recovery action for those errors.

Messages:

```
CPF4132 (Escape)*
CPF5129 (Escape)*
CPF5293 (Escape)*
CPF5427 (Escape)*
```

Major Code 81

Major Code 81 – Permanent device error (nonrecoverable).

Description: A nonrecoverable device-related error occurred during an I/O operation. Any attempt to continue using this printer device will probably fail again until the cause of the problem is found and corrected.

Action: The following general actions can be taken for each 81xx return code. Other specific actions are given in each return code description.

- Continue processing without the printer device.
- Close the file, correct the problem, and open the file again. If the operation is still unsuccessful, try it again only a limited number of times. (The number of times should be specified in your program.)
- End.

Several return codes indicate that an error condition must be corrected by varying the device off and on again.

Code **Description/Action**

8181 **Description**: A system error condition was detected during an I/O operation to the printer device.

Action: Close the file. You may need to vary the device off and on again to clear the error. Determine the cause of the failure from the accompanying message. Check for any system operator messages indicating that additional corrective action must be performed. Open the file again to continue.

Messages:

```
CPF4289 (Escape)
CPF4552 (Escape)
CPF4553 (Escape)
CPF5105 (Escape)
CPF5159 (Escape)
CPF5507 (Escape)
```

8191 **Description:** The operation was not successful because a permanent line error occurred, and the system operator took a recovery option in response to the line error message. (You can find out what type of line error occurred by asking the system operator.) The device has been marked unusable.

Action: Close the file. Vary the device off and on again to clear the error. Open the file again to continue.

Messages:

```
CPF4146 (Escape)
CPF4193 (Escape)
CPF4526 (Escape)
CPF4542 (Escape)
CPF5128 (Escape)
CPF5198 (Escape)
```

8197 **Description:** A nonrecoverable error condition was detected at the device.

Action: Close the file. Vary the device off and on again to clear the error. Refer to the accompanying error message for additional information regarding the source of the specific error detected. Open the file again to continue.

Messages:

CPF4149 (Escape)
CPF4192 (Escape)
CPF4197 (Escape)
CPF4216 (Escape)
CPF4524 (Escape)
CPF4533 (Escape)
CPF4538 (Escape)
CPF4583 (Escape)
CPF5106 (Escape)
CPF5143 (Escape)
CPF5199 (Escape)
CPF5201 (Escape)
CPF5268 (Escape)
CPF5360 (Escape)

81C2 **Description:** The operation issued by your program was not successful because the Systems Network Architecture (SNA) session with the printer is not active.

Action: Close the file. Vary the device off and on again to clear the error. Open the file again to continue.

Messages:

CPF5422 (Escape)

Major Code 82

Major Code 82 – Open operation failed.

Description: An attempt to open the printer file was not successful. The error may be recoverable or permanent, but is limited to the printer device. Recovery is unlikely until the problem causing the error has been corrected.

Action: The following general actions can be taken for each 82xx return code. Other specific actions are given in each return code description. You can either:

- Continue processing without the device.
- Close the file, correct the problem, and open the file again. A subsequent operation could be successful if the error occurred because of some temporary condition such as the device being in use at the time.

  If the operation is still unsuccessful, try it again only a limited number of times. (The number of times should be specified in your program.)
- End.

Several return codes indicate that an error condition must be corrected by changing a value in the file. To change a parameter value for the file, use the Change Printer File (CHGPRTF) or the Override with Printer File (OVRPRTF) command.

Code **Description/Action**

8281 **Description:** A system error condition was detected on an open operation that was not successful. The printer file may previously have been in error, or the printer file could not be opened due to a system error.

Action: Your printer may need to be varied off and then on again to clear the error. Your program can either:

- Continue processing without the printer.
- Close the file, correct the problem, and open the file again.
- End.

Determine the cause of the failure from the accompanying message.

Messages:

```
CPF4168 (Escape)*
```

8282 **Description:** The open operation was not successful because the printer device is unusable. This may occur because a cancel reply has been taken to an error recovery message for the printer or because the printer has been held by a Hold Communications Device (HLDCMNDEV) command. No operations should be issued to the device.

Action: Close the file. Communications with the printer cannot be resumed until the device has been reset to a varied-on state. If the device has been held, use the Release Communications Device (RLSCMNDEV) command to reset the device. If the device is in an error state, vary the device off and then on again. Once the device is reset, normal operation can be started by opening the printer device file again.

Messages:

CPF4110 (Escape)
CPF4298 (Escape)
CPF4354 (Escape)

8291 **Description:** A permanent line error occurred on an open operation. The printer device has been marked unusable.

Action: Close the file. Vary the device off and on again to clear the error. Open the file again to continue.

Messages:

CPF4179 (Escape)
CPF4291 (Escape)

82A6 **Description:** The open operation failed because of a Systems Network Architecture (SNA) protocol violation.

Action: Ensure that the printer with which your program is communicating is configured properly. Refer to the device response codes in the accompanying error message for additional information regarding the specific error detected.

Messages:

CPF4124 (Escape)
CPF4190 (Escape)
CPF4192 (Escape)
CPF4527 (Escape)
CPF4533 (Escape)
CPF5103 (Escape)
CPF5143 (Escape)
CPF5453 (Escape)

82AA **Description:** The open operation was not successful because the printer device description was not found.

Action: Your program can continue without the printer, attempt to use a different printer, or end.

Verify that the name of the printer was correctly specified in the DEV parameter on the CRTPRTF, CHGPRTF, OVRPRTF, or CRTPRTDEV command.

Messages:

CPF4103 (Escape)*

82B3 **Description:** The open operation was not successful because the printer you requested is in use in another file in your job.

Action: Close both of the printer device files, then open the one that you want to use again.

Messages:

```
CPF4106 (Escape)
```

82EE **Description:** An open operation was attempted to a device that is not supported for a printer file.

Your program is attempting to open a device that is not a valid printer.

Action: Your program can continue without the printer, attempt to use a different printer, or close the file and end.

Verify that the name of the printer was specified correctly on the CHGPRTF or OVRPRTF command.

Messages:

```
CPF4105 (Escape)
```

82EF **Description:** An open operation was attempted for a device that the user is not authorized to, or that is in service mode.

Action: Your program can continue without the printer, attempt to use a different printer, or end.

Close the file, correct the problem, and then issue the open operation again.

For authority errors, obtain authority to the device from your security officer or device owner. If the device is in service mode, the system service tools (SST) function is currently using the device. Wait until the device is available to issue the operation again.

Messages:

```
CPF4104 (Escape)*
CPF4186 (Escape)
CPF9802 (Diagnostic)*
```

Major Code 83

Major Code 83 – Device error occurred (recoverable).

Description: An error occurred during an I/O operation, but the printer device is still usable. Recovery within your program might be possible.

Action: The following general actions can be taken for each 83xx return code. Other specific actions are given in each return code description.

- Continue processing without the printer device.
- Correct the problem and continue processing with the printer device. If the attempt to recover from the operation is unsuccessful, try it again only a limited number of times. (The number of times should be specified in your program.)
- End.

Several return codes indicate that an error condition must be corrected by changing a value in the file. To change a parameter value for the file, use the Change Printer File (CHGPRTF) or Override with Printer File (OVRPRTF) command.

Code Description/Action

8319 **Description:** A negative response was received to the last printer operation attempted by your program. The error may have been caused by the user pressing the Cancel key on the printer.

Action: Your program can try a different operation, or close the file and end. Refer to the device response code in the accompanying message to determine why the operation was rejected. Correct the error in your program before attempting to try the operation again.

Messages:

```
CPF4158 (Escape)
CPF4531 (Escape)
CPF5050 (Escape)
```

831D **Description:** The operation just attempted by your program was rejected because a parameter was not valid, out of limits, or missing.

Action: Your program can bypass the failing step and continue, or close the file and end. Refer to the accompanying message to determine what parameter was incorrect. Correct the error in your program before attempting to try the operation again.

Messages:

CPD4016 (Diagnostic)*

CPD4017 (Diagnostic)*

CPD4027 (Diagnostic)*

CPD4028 (Diagnostic)*

CPD4029 (Diagnostic)*

CPD4030 (Diagnostic)*

CPD4041 (Diagnostic)*

CPF4909 (Notify)*

CPF5108 (Escape)*

CPF5148 (Escape)*

CPF5273 (Escape)*

CPF5275 (Escape)

CPF5276 (Escape)*

CPF5288 (Escape)*

CPF5289 (Escape)*

CPF5324 (Escape)*

CPF5359 (Escape)*

CPF5363 (Escape)

CPF5366 (Escape)*

CPF5367 (Escape)*

CPF5368 (Escape)*

831E **Description:** The operation just issued by your program was not valid or an invalid combination of operations was specified.

Action: Your program can bypass the invalid operation and continue, or close the file and end. Refer to the accompanying message to determine why the operation was rejected. Correct the error in your program before attempting to try the failing operation again.

Messages:

CPD4015 (Diagnostic)*

CPD4018 (Diagnostic)*

CPD4031 (Diagnostic)

CPF4915 (Notify)*

CPF5149 (Escape)*

CPF5185 (Escape)*

CPF5245 (Escape)*

CPF5290 (Escape)*

CPF5320 (Escape)*

CPF5321 (Escape)*

CPF5322 (Escape)*

CPF5323 (Escape)*

CPF5325 (Escape)*

CPF5362 (Escape)*

831F **Description:** A length that is not valid was specified on the operation.

On an output operation, your program has tried to send a data record having a length that exceeds the maximum record length allowed for the printer or the page size defined for the file. If you are using direct I/O, you have exceeded the maximum number of bytes allowed per page. The data has been truncated.

Action: Issue the output operation again with a smaller output length. The record length for a program-described printer file cannot exceed the page size. The record length for any printer file must be no greater than 32 767 characters.

Messages:

```
CPF4906 (Notify)*
CPF5160 (Escape)
```

8343 **Description:** The designated page overflow line number has been reached.

Action: Your program should take whatever application dependent action is appropriate. This may include printing page totals or a running foot line.

Messages:

```
CPF5004 (Status)*
```

83E0 **Description:** Your program attempted to issue an operation using a record format that was not defined for the printer file, or omitted the record format name.

Action: Check the name of the record format in your program to be sure it is correct. Then check that the record format is defined properly in the DDS for the file.

Messages:

```
CPF5186 (Escape)*
CPF5187 (Escape)*
```

83F6 **Description:** Your program sent invalid data to the printer. The data type may be incorrect for the field in which it is used.

Action: Check the name of the record format in your program to be sure it is correct. Verify that the data definition statements in your program match the output record defined in the DDS for the file. Correct the error in your program before attempting to repeat the failing operation.

Messages:

CPD4014 (Diagnostic)*
CPD4022 (Diagnostic)*
CPD4026 (Diagnostic)*
CPD4035 (Diagnostic)*
CPD4516 (Informational)
CPD4591 (Escape)
CPF4634 (Escape)
CPF4635 (Escape)
CPF4636 (Escape)
CPF4642 (Escape)
CPF4643 (Escape)
CPF4644 (Escape)
CPF4645 (Escape)
CPF4646 (Escape)
CPF4647 (Escape)
CPF5075 (Notify)*
CPF5234 (Escape)*
CPF5246 (Escape)
CPF5261 (Escape)
CPF5297 (Escape)*
CPF5364 (Escape)
CPF5365 (Escape)
CPF5369 (Escape)
CPF5372 (Escape)
CPF5373 (Escape)
CPF5374 (Escape)
CPF5375 (Escape)
CPF5376 (Escape)
CPF5377 (Escape)
CPF5411 (Escape)

Appendix E. Edit Codes

OS/400 Edit Codes

Table E-1 summarizes the functions provided by OS/400 edit codes.

Table E-1. Summary Chart for OS/400 Edit Codes

Edit Code	Commas[1] Printed	Decimal Points[1] Printed	Signs Printed When Negative Number	Blank Value of QDECFMT System Value	I Value of QDECFMT System Value	J Value of QDECFMT System Value	Leading Zero Suppressed
1	Yes	Yes	No sign	.00 or 0	,00 or 0	0,00 or 0	Yes
2	Yes	Yes	No sign	Blanks	Blanks	Blanks	Yes
3		Yes	No sign	.00 or 0	,00 or 0	0,00 or 0	Yes
4		Yes	No sign	Blanks	Blanks	Blanks	Yes
A	Yes	Yes	CR	.00 or 0	,00 or 0	0,00 or 0	Yes
B	Yes	Yes	CR	Blanks	Blanks	Blanks	Yes
C		Yes	CR	.00 or 0	,00 or 0	0,00 or 0	Yes
D		Yes	CR	Blanks	Blanks	Blanks	Yes
J	Yes	Yes	− (Minus)	.00 or 0	,00 or 0	0,00 or 0	Yes
K	Yes	Yes	− (Minus)	Blanks	Blanks	Blanks	Yes
L		Yes	− (Minus)	.00 or 0	,00 or 0	0,00 or 0	Yes
M		Yes	− (Minus)	Blanks	Blanks	Blanks	Yes
Y[2]							Yes
Z[3]							Yes

[1] The QDECFMT system value determines the decimal point character (period as used in the U.S.), the character used to separate groups of three digits (comma as used in the U.S.), and the type of zero suppression (depending on comma and period placement).

[2] The Y edit code suppresses the farthest left zero of a date field that is three to six digits long, and it suppresses the two farthest left zeros of a field that is seven positions long. The Y edit code also inserts slashes (/) between the month, day, and year according to the following pattern:

nn/n
nn/nn
n/nn/n
nn/nn/nn
nnn/nn/nn

If the DATE keyword is specified with EDTCDE(Y), the separator character used is the job attribute, DATSEP. If not specified on the DATSEP job attribute, the system value, QDATSEP is used (which defaults to [/]). At file creation time, DATFMT is *JUL (Julian), the date is normally formatted as nn/nnn and EDTCDE(Y) is not valid.

[3] The Z edit code removes the sign (plus and minus) from a numeric field. The sign of the units column is changed to a hexadecimal F before the field is written.

Examples of Editing Using OS/400 Edit Codes

Table E-2 shows valid edit codes with examples of unedited source data and edited output. Zero suppression and decimal characters are determined by the system value QDECFMT. The date separator character is determined by the job attribute DATSEP. In this figure, QDECFMT is assumed to equal ƀ, and DATSEP is assumed to equal /.

Table E-2. Valid Edit Codes, Source Data, and Edited Output

Edit Codes	Positive Number — Two Decimal Positions	Positive Number — No Decimal Positions	Negative Number — Three Decimal Positions[1]	Negative Number — No Decimal Positions[1]	Zero Balance— Two Decimal Positions[1]	Zero Balance — No Decimal Positions[1]
Unedited	1234567	1234567	ƀƀƀƀ125–	ƀƀƀƀ125–	ƀƀƀƀƀƀ	ƀƀƀƀƀƀ
1	12,345.67	1,234,567	.125	125	.00	0
2	12,345.67	1,234,567	.125	125		
3	12345.67	1234567	.125	125	.00	0
4	12345.67	1234567	.125	125		
A	12,345.67	1,234,567	.125CR	125CR	.00	0
B	12,345.67	1,234,567	.125CR	125CR		
C	12345.67	1234567	.125CR	125CR	.00	0
D	12345.67	1234567	.125CR	125CR		
J	12,345.67	1,234,567	.125–	125–	.00	0
K	12,345.67	1,234,567	.125–	125–		
L	12345.67	1234567	.125–	125–	.00	0
M	12345.67	1234567	.125–	125–		
Y[2]	123/45/67	123/45/67	0/01/25	0/01/25	0/00/00	0/00/00
Z[3]	1234567	1234567	125	125		

[1] The ƀ represents a blank.
[2] The Y edit code suppresses the farthest left zero of a date field that is three to six digits long, and it suppresses the two farthest left zeros of a field that is seven positions long.
[3] The Z edit code removes the sign (plus or minus) and suppresses leading zeros.

User-Defined Edit Codes

Table E-3. IBM-Supplied Edit Descriptions

Description	QEDIT5 Edit Codes	QEDIT6 Edit Codes	QEDIT7 Edit Codes	QEDIT8 Edit Codes	QEDIT9 Edit Codes
Integer mask	ƀƀƀ,ƀƀƀ,ƀƀƀ,ƀƀƀ,ƀƀ0	ƀƀƀ,ƀƀƀ,ƀƀƀ,ƀƀƀ,ƀƀ0	ƀƀƀ,ƀƀƀ,ƀƀƀ,ƀƀƀ,ƀ0ƀ	ƀƀƀ,ƀƀƀ,ƀƀƀ,ƀƀƀ,ƀ0ƀ	0ƀƀ-ƀƀ-ƀƀ
Decimal point	. (period)	. (period)	. (period)	. (period)	. (period)
Fraction mask	ƀƀƀƀƀƀƀƀƀ	ƀƀƀƀƀƀƀƀƀ	ƀƀƀƀƀƀƀƀƀ	ƀƀƀƀƀƀƀƀƀ	None
Fill character	ƀ	ƀ	ƀ	ƀ	ƀ
Floating currency symbol	None	None	None	$	None
Zero balance	Replace with fill character	Replace with fill character	Normal editing rules	Normal editing rules	Normal editing rules
Negative status	CR	- (minus sign)	- (minus sign)	- (minus sign)	None
Positive status	DR	None	None	None	None
Left constant	None	None	$	None	None
Right constant	None	ƀ*	None	None	None

An edit description contains:

- Integer mask. Describes the editing of the integer portion of a field. All characters except blank, zero, and ampersand (&) are treated as constants:
  - Blank means to replace the blank with a digit if zero suppression has ended; otherwise, replace the blank with the fill character.
  - The zero to the farthest left means to replace the zero with a digit and end zero suppression. All other zeros are treated as constants.
  - Ampersand means to replace the & with a blank.
- Decimal point. Defines what character is used as the decimal point. By default, a period (.) is used.
- Fraction mask. Describes the editing of the fraction portion of a field. Ampersand is the same as for the integer mask. All zeros are treated as constants and all blanks are replaced with digits (no fill character is used).
- Fill character. Defines what character is used in each position of a result that is zero-suppressed. By default, a blank is used.
- Floating currency symbol. Defines the floating currency symbol to be used to edit the field.
- Zero balance. Specifies how zero values are to be edited. They can be edited using the fill character or the integer and fraction masks.
- Negative status. Defines the characters that are to follow the edited result of a field if the field is negative.
- Positive status. Defines the characters that are to follow the edited result of a field if the field is positive or zero.

- Left constant. Defines a constant that is to be the portion to the farthest left of the edited result of a field.
- Right constant. Defines a constant that is to be the portion to the farthest right of the edited result of a field.

Using User-Defined Edit Codes

The following are rules that apply to using edit descriptions. These rules are affected by the length and decimal positions of the field being edited.

- The field to be edited is aligned by the integer and fraction masks.
- The entire integer mask is not always used. The integer mask is truncated immediately to the digit replace character to the farthest left that could be used, which is by the number of integers in the field to be edited. Zero suppression termination is remembered from the truncated position.
- The decimal point immediately follows the integer mask and the fraction mask immediately follows the decimal point. If no decimal point is used, the fraction mask immediately follows the integer mask.
- The entire fraction mask is not always used. The fraction mask is truncated immediately to the digit replace character to the farthest right that could be used, which is by the number of decimal positions in the field to be edited.
- The width of the edited result is equal to the total of the following:
  - Length of left constant
  - Length of floating currency symbol
  - Length of truncated integer mask
  - Length of decimal point (which is always 1 unless no decimal point is used)
  - Length of truncated fraction mask
  - Length of negative or positive status value
  - Length of right constant
- If either the integer mask or fraction mask does not contain enough digit replace characters for the field to be edited, the field is not edited and is ignored.

Example of a User-Defined Edit Code

The following Create Edit Description (CRTEDTD) command shows how to create an edit description to edit a numeric field and indicate if the value is a credit or debit:

```
CRTEDTD  EDTD(5) INTMASK('ƀƀƀ,ƀƀƀ,ƀƀ')
         FRACMASK('ƀƀƀƀ') NEGSTS('DEBITƀ')
         POSSTS('CREDIT')  LFTCNS('$')  RGTCNS('ƀB**')
```

The field that uses the previous edit description contains the value 001234 and has two decimal positions. The edited field looks like this:

```
$ƀƀ12.34CREDITƀ**
```

Note that when you create an edit description, you specify only the number (5, 6, 7, 8, or 9) that is associated with the edit code. The system automatically affixes QEDIT to the number you specify. (In the preceding example, EDTD(5) was specified. This would then be QEDIT5.)

Appendix F. System/36-Compatible Display Data Management

This appendix describes how to use OS/400 display data management to provide System/36-compatible functions. Additional information on System/36 compatibility can be found in the following manuals:

- *Concepts and Programmer's Guide for the System/36 Environment*
- *System/36-Compatible RPG II User's Guide/Reference*
- *System/36-Compatible COBOL User's Guide/Reference*
- *DDS Reference* manual

In order to migrate System/36 applications that use display devices, OS/400 display data management has functions that allow the operating system to work like System/36 work station data management. The level of compatibility with the System/36 depends on the following:

- The User Display Management (USRDSPMGT) keyword

  This keyword is automatically put in the DDS source when the $SFGR utility or the Create System/36 Display File (CRTS36DSPF) CL command converts SFGR source to DDS source. Specifying this keyword indicates that the System/36 work station data management function is to be used instead of OS/400 display data management functions.

  For example, the USRDSPMGT keyword specifies that the cursor is positioned as it is on the System/36. If the USRDSPMGT keyword is not specified, the cursor is positioned according to the OS/400 display data management rules.

- The programming language used to write the program

  To get application-level System/36 compatibility, the programs must be written in System/36-compatible RPG II or System/36-compatible COBOL.

  For example, if a program is written in a System/36-compatible language, the file status codes set by the high-level language and return codes set by data management are the same as the System/36. If a program is not written in a System/36-compatible language, the file status codes and return codes are the AS/400 values.

- The environment where the application runs

  To get environment-level System/36 compatibility, the programs must run in the System/36 environment.

  For example, if an application running in the System/36 environment uses the MSGID DDS keyword to display the first-level text message from a user message file, the message file used is the message file specified on the USER1 parameter of the MEMBER OCL statement. If this same application is not run in the System/36 environment, the message file used is a message file named USR1 in the job's library list.

The remainder of this appendix contains a section for each display data management topic you may need to know about if you are interested in S/36 compatibility. Where appropriate, each topic contains the following information:

- How the System/36 supports the function. More information on the System/36 functions can be found in the following manuals:
  - *System/36 Creating Displays: Screen Design Aid and System Support Program*, SC21-7902
  - *System/36 Concepts and Programmer's Guide*, SC21-9019
- How the default OS/400 display data management supports the function.
- How to get the System/36-compatible function with OS/400 display data management, including which of the three level-of-compatibility items previously listed is required to get the compatible function.

Clearing Lines on the Display

When a System/36 application writes the first record in a job, System/36 work station data management clears the display before displaying the application data. The System/36 function is provided on the AS/400 system for the first write operation in a System/36-environment job if the program is written in a System/36-compatible language or the display file contains the USRDSPMGT keyword. The *Concepts and Programmer's Guide for the System/36 Environment* describes starting and ending a System/36-environment job.

OS/400 display data management normally clears the display when a display file is opened; System/36 work station data management does not. The System/36 function is provided by the OS/400 operating system for programs written in a System/36-compatible language or that open a display file that specifies the USRDSPMGT keyword.

Applications can clear all or just parts of the display by using the clear line (CLRL) keyword. The CLRL keyword is automatically put in the DDS source when the System/36 environment converts the SFGR source to DDS. The value generated for the number of lines to clear is based on the value in columns 19 and 20 of the SFGR S specification.

Because some of the OS/400 and System/36-compatible display-clearing functions are different, the following DDS keywords cannot be used in display files that specify the USRDSPMGT keyword:

- ERASE
- PUTRETAIN
- KEEP
- ASSUME

Also, the OVERLAY keyword is ignored for programs written in a System/36-compatible language and for display files that specify the USRDSPMGT keyword.

Input Data for Display File Records

On the System/36, only one record with input fields can be displayed. For example, if an application on the System/36 writes record REC1 and then record REC2, only the last record written (REC2) has input-capable fields. When REC1 is written, the input fields defined by REC1 are input-capable. When REC2 is written, REC1 fields are then no longer input-capable, and the input fields defined by REC2 are input-capable.

With the default OS/400 display data management support, many different records with input fields can be displayed at the same time. For example, record REC1 (which occupies lines 1 through 12) and record REC2 (which occupies lines 21 through 24) can be displayed at the same time. The application program can read data from either record. Only the data for the input-capable fields defined by the specific record is returned. In the previous example, if the application program needs to get all the data from all the input-capable fields on the display, the application has to issue two read operations: one for REC1 and one for REC2.

OS/400 display data management allows only one record with input fields for programs written in a System/36-compatible language or for display files that specify the USRDSPMGT keyword.

Input Data from the Work Station Controller

The System/36 work station controller returns the data for all input-capable fields. With the default OS/400 display data management support, the work station controller returns only data from the display for the input-capable fields that the display station operator has changed.

OS/400 display data management provides the System/36 function with the Modified Data Tag (MDT) value of the DSPATR DDS keyword. The DSPATR(MDT) keyword is automatically put in the DDS source for every input-capable field when the System/36 environment converts SFGR source to DDS. This keyword sets on an indicator in the work station controller that makes the work station controller act as if the field with DSPATR(MDT) was changed by the display station operator. Because this keyword is specified for all input-capable fields, all the input data is returned to the system and to the program when the program issues a read operation.

Self-Check

On System/36, if a field specifies modulus 10 or modulus 11 self-checking (column 30 of SFGR D specification), the work station controller verifies that valid data is entered into the field. If a display station operator enters an incorrect modulus 10 or modulus 11 number for a field, the work station controller displays a blinking four-digit error code, and the user must enter data into the field again.

The default OS/400 display data management support for self-check verifies that the field has a valid modulus 10 or modulus 11 number when the Enter key or a function key is pressed instead of issuing the error message as the user types the data into the field.

OS/400 display data management provides the System/36 function of issuing the error message as the user types the data if the display file containing the modulus 10 or modulus 11 field specifies the USRDSPMGT keyword.

Return Input

System/36 work station data management has support to indicate to an application program whether any fields on the display have been modified by the display station operator. To use this support, the SFGR source must specify N for return input (column 22) in the SFGR S specification. If N is specified and the user does not change any fields on the display, all blanks are received as input data for a read operation.

OS/400 display data management provides the System/36 support. If the CHANGE keyword is specified at the record level without a response indicator in a display file that specifies the USRDSPMGT keyword, and the program doing the read operation is written in a System/36-compatible language, the program receives a blank record when the display station operator does not change any data on the display. The CHANGE keyword is automatically put in the DDS source when SFGR source is converted to DDS by the System/36 environment if N is specified for return input.

Erase Input Fields

On System/36, if a program writes a record to the display and the erase-input-fields function (columns 31 and 32 of the SFGR S specification) is enabled for the record, the input fields on the display are erased, but the specified record is not sent to the display. For example, if a program writes record REC1 to the display and then writes record REC2 to the display (where REC2 has the erase-input-fields function enabled), all the input fields defined by REC1 are erased and REC2 is not sent to the display.

OS/400 display data management follows the System/36 handling of erase-input-fields when the program is written in a System/36-compatible language, the USRDSPMGT keyword is specified, and the ERASEINP keyword is enabled for the record.

On System/36, if a program writes a record to the display and the erase-input-fields and put-override functions are both enabled for the record, the input fields on the display are erased, and the put-override data for the specified record is sent to the display.

OS/400 display data management follows the System/36 handling of erase-input-fields when the program is written in a System/36-compatible language, the USRDSPMGT keyword is specified, and the ERASEINP and PUTOVR keywords are enabled for the record.

Display Attributes

Some fields do not have beginning display attributes on the System/36. A field has a beginning display attribute only if one or more of the following is true:

- The field is input-capable.
- The output data is based on an indicator (column 23-24 of SFGR D specification).
- The field specifies a display attribute (high intensity, blink, nondisplay, reverse image, underline, column separators).
- The field appears on a row that is not cleared by the record.

To provide application compatibility with the System/36, OS/400 display data management follows the System/36 rules to determine if a field should have a beginning attribute when the USRDSPMGT keyword is specified in a display file. Also, if a field does not have a beginning attribute and the OVRATR keyword is specified on the field, the OVRATR keyword is ignored (no field attribute is sent).

On the System/36, some fields do not have ending display attributes. A field has an ending attribute only if one or more of the following is true:

- The field is input-capable.
- The output data is based on an indicator (columns 23-24 of SFGR D specification).
- The field specifies a display attribute (high intensity, blink, nondisplay, reverse image, underline, column separators).

OS/400 display data management follows the System/36 rules to determine if a field should have an ending attribute when the USRDSPMGT keyword is specified in a display file.

Positioning the Cursor

System/36 uses the following rules to determine where the cursor should be positioned when a record is displayed:

1. The cursor is positioned to the first input field (defined by the SFGR source statements) that has an indicator specified for the position-cursor attribute, has this indicator set on, and where one of the following conditions is true:
   - This field does not have the protect attribute specified (columns 37 and 38 of the SFGR D specification).
   - This field has an indicator specified for the protect attribute, but the indicator is off.
   - This field has an unoptioned-protect attribute specified (columns 37 or 38 of the SFGR D specification is Y or N).
2. If the cursor is not positioned by rule 1, a check is made for fields with an unoptioned-position-cursor attribute (columns 32 or 33 of the SFGR D specification is Y). If there is a field with an unoptioned-position-cursor attribute, the cursor is positioned to this field.

3. If the cursor is not positioned by rule 1 or 2, the cursor is positioned to the first input field (as defined by the SFGR source statements) that does not have an unoptioned-protect attribute (columns 37 or 38 of the SFGR D specification are blank).
4. If the cursor is not positioned by any of the previous rules, the cursor is positioned to the upper-left-hand corner of the display.

OS/400 display data management positions the cursor following the System/36 rules for display files that have the USRDSPMGT keyword specified.

Displaying Messages

On System/36, fields can be defined that have message text automatically inserted into the field when a record is displayed. System/36 work station data management retrieves the message text defined for the field and supplies the message text as output data for the field. To display message text, System/36 applications must specify the following:

- M for constant type (column 56 of SFGR D specification).
- The message identification code (MIC) (columns 57 through 60 of SFGR D specification or output data supplied by the program).
- The message member identifier (columns 61 and 62 of SFGR D specification or output data supplied by the program).

OS/400 applications can display messages in a variety of different ways. For information on displaying messages, see "Messages on the Display" on page 4-110.

To provide application compatibility with System/36, the MSGID keyword can be used to display messages. The MSGID keyword is automatically generated by the System/36 environment if M is specified for constant type in the SFGR D specification.

Either of the following formats can be used to display messages that follow the System/36 conventions for sending messages:

```
MSGID(message-identifier message-file)
```

```
MSGID(USR message-identification-code message-file)
```

Message-identifier consists of two parts: a message prefix and a message-identification-code (MIC). If an application uses the first format of the MSGID keyword, the three-character message prefix should be the first three characters of the seven-character message-identifier. If an application uses the second format of the MSGID keyword, the prefix does not need to be provided. The prefix is already specified as USR in the MSGID keyword.

The message-identification-code parameter specifies the four-character message ID of the message to be displayed.

For more information on specifying display file message-identifiers, see the *DDS Reference*.

The message-file parameter identifies the message file that contains the message to be displayed. For System/36 compatibility, this parameter can be specified in one of two formats.

The first format for specifying the message-file parameter is in a two-character field-name. The field-name must exist in the same record as the MSGID field, and the field must be defined as a character field with usage H (hidden), P (program-to-system), B (both), or O (output-only). The field identified by field-name indicates the message file that the System/36 environment uses to display a message. The values allowed for the field are:

Table F-1. Message Files for MSGID

Value of Field-Name	Message File Used
U1	First-level message text from the message file specified on USER1 parameter on MEMBER OCL statement.
U2	Second-level message text from the message file specified on USER2 parameter on MEMBER OCL statement.
P1	First-level message text from the message file specified on PROGRAM1 parameter on MEMBER OCL statement.
P2	Second-level message text from the message file specified on PROGRAM2 parameter on MEMBER OCL statement.
M1	First-level message text from IBM-supplied message file ##MSG1.
M2	Second-level message text from IBM-supplied message file ##MSG1.

The second format for specifying the message-file parameter is a special value. The following special values can be specified for the message file:

Table F-2. Message Files for MSGID

Special Value	Message File Used
*USR1	First-level message text from message file specified on USER1 parameter on MEMBER OCL statement.
*USR2	Second-level message text from message file specified on USER2 parameter on MEMBER OCL statement.
*PGM1	First-level message text from message file specified on PROGRAM1 parameter on MEMBER OCL statement.
*PGM2	Second-level message text from message file specified on PROGRAM2 parameter on MEMBER OCL statement.
*SYS1	First-level message text from IBM-supplied message file ##MSG1.
*SYS2	Second-level message text from IBM-supplied message file ##MSG1.

The following are considerations you should be aware of when using the System/36-compatible functions of the MSGID keyword:

- If a field name or a special value is specified, the processing done by OS/400 display data management depends on the environment where the application runs. For example, if the application is run in the System/36 environment, *USR1 indicates to use the message file identified by the USER1 parameter on the MEMBER OCL statement. If the same application is not run in the System/36 environment, *USR1 indicates to use message file USR1.
- If the first-level text is to be displayed and the message has only second-level text, the second-level text is displayed.
- If the second-level text is to be displayed and the message has only first-level text, the first-level text is displayed.
- If the message text to be displayed is longer than the length of the output field, the message text is truncated to the length of the output field. If the message text to be displayed is shorter than the length of the output field, the message text is padded with blanks.
- If the message identifier contains any characters that are not valid (valid characters are 0-9 and A-F), the message text displayed is the message identifier followed by the two-character message file identifier.
- If the message identifier or the message file is not found, the message text displayed is the message identification code followed by two question marks (??).
- The following DDS keywords cannot be specified on a field with the MSGID keyword:

  DFT
  DFTVAL
  FLTFIXDEC
  FLTPCN
  MSGCON

Put Override

On System/36, if the put-override option is enabled (columns 33 and 34 of SFGR S specification) and the number of lines to clear (columns 19 and 20 of SFGR S specification) is specified, the number of lines to clear is ignored. System/36 work station data management does not clear the display when the put-override option is enabled. OS/400 display data management also has support to ignore the clear function (CLRL keyword) for records that have the put-override option (PUTOVR keyword) enabled. This support is used for programs written in a System/36-compatible language or using a display file that specifies the USRDSPMGT keyword.

On System/36, if the put-override option is enabled, no input-field definitions are sent to the display. Input-field definitions contain information such as length of the input-field, mandatory-fill attribute, mandatory-enter attribute, modified-data-tag attribute, and protected input-field attribute. If there are input fields currently on the display, these fields remain input-capable. The default OS/400 display data management support sends input-field definitions when the put override option is enabled. OS/400 display data management provides the System/36 function of not sending input-field definitions to the display when the put-override function is enabled and the record to be displayed is in a display file that specifies the USRDSPMGT keyword. If the application uses the System/36 function of not sending input-field definitions to the display when the put-override function is enabled, attributes that are defined by the input-field definitions cannot be changed when put-override is enabled. For example, the protected input-field attribute (DSPATR(PR)) is ignored when put-override is enabled, because the input-field definitions are not sent to the display.

On System/36, the put-override option can be specified for records that are currently displayed and for records that are not currently on the display. For example, if record REC1 is currently displayed and the application program issues a write operation with the put-override function enabled for record REC2, the request to display record REC2 is processed as a put-override request. The default support in OS/400 display data management supports the put-override function only for records that are currently displayed. For example, if record REC1 is currently displayed and the application program issues a write operation with the put-override function enabled for record REC2, the put-override function is ignored and the request to display record REC2 is processed as if the PUTOVR keyword was not enabled. If the display file specifies the USRDSPMGT keyword, OS/400 display data management provides the System/36 function of allowing the put-override function even if the record is not currently displayed.

Handling Signed Numeric Data

System/36 does no checking when incorrect data (data other than 0 through 9) is specified for a signed numeric field on a write operation. For example, if the data in a signed numeric field is character data, the character data is displayed. The default support in OS/400 display data management is to replace any incorrect signed numeric data with nulls when the field is displayed. OS/400 display data management supports the System/36 function when the USRDSPMGT keyword is specified in the display file.

System/36 also does no checking when a read operation returns incorrect data for a signed numeric field. For example, if the data returned for a signed numeric field is character data, the character data is passed to the application program. The default support in OS/400 display data management replaces any incorrect signed numeric data with zeros before the data is given to the application program. OS/400 display data management supports the System/36 function for programs written in a System/36-compatible language.

On System/36, if zeros are entered in a signed numeric field and the field-minus key is used to exit the field, a value of negative zero is returned to the application program. The default support in OS/400 display data management returns zeros to the application instead of negative zero. OS/400 display data management supports the System/36 function for programs written in a System/36-compatible language or using a display file that specifies the USRDSPMGT keyword.

Function Keys

On System/36, application programs can receive control when a function key is pressed. To use this support, the application must enable the desired function key in the SFGR source and the program must indicate that it can process the function key.

The default OS/400 display data management support does not allow a program not written in a System/36 compatible language to indicate which function keys the program can process. The function keys that are enabled in the DDS for the display file are the function keys that OS/400 display data management passes to the program to handle.

OS/400 display data management supports the System/36 function of a program indicating which function keys it can process for programs written in a System/36-compatible language. See the appropriate System/36-compatible language manual for information on enabling function-key handling in a program.

Help Key Considerations

The HELP and HLPRTN DDS keywords are used to indicate what processing should be done by the system when the Help key is pressed. The HELP keyword indicates that the application wants the Help key enabled. The HELP keyword is automatically put in the DDS source when SFGR source is converted to DDS by the System/36 environment.

The HLPRTN keyword indicates that the application program wants to process the Help key when the display station operator presses the Help key. When the Help key is enabled in the SFGR key mask (columns 64 through 79 of the SFGR S specification) the HLPRTN keyword is put in the DDS source when the System/36 environment converts SFGR source to DDS.

If the HELP and HLPRTN keywords are not specified and the display station operator presses the Help key, a message is displayed by OS/400 display data management indicating that the Help key is not allowed.

If the USRDSPMGT and HELP keywords are specified, the HLPRTN keyword is not specified, and the display station operator presses the Help key, OS/400 display data management uses the following rules to determine how the Help key is be processed:

1. If the Help key is pressed when a message is being displayed, the second-level text for the message is displayed.
2. If the Help key is not processed by rule 1 and there is application help defined for the record, the application help for the record is displayed.
3. If the Help key is not processed by rules 1 or 2, an error message is displayed indicating that the key is not allowed.

If the USRDSPMGT, HELP, and HLPRTN keywords are specified and the display station operator presses the Help key, OS/400 display data management uses the following rules to determine how the Help key is be processed:

1. If the application program has indicated that it can process the Help key, an indication is returned to the application program that the Help key was pressed.
2. If the Help key is not processed by rule 1 and the Help key is pressed when a message is being displayed, the second-level text for the message is displayed.
3. If the Help key is not processed by rules 1 or 2 and there is application help defined for the record, the application help for the record is displayed.
4. If the Help key is not processed by rules 1, 2 or 3, an error message is displayed indicating that the key is not allowed.

Using Command Keys to Exit Application Help

On System/36, if application help is displayed and the display station operator presses a command key to exit application help, an indication of what command key was pressed is given to the application program. To use this support, the application must enable the desired command key on the display where the Help key is pressed and the key must be enabled on the application help display. If the display file containing application help specifies the USRDSPMGT keyword, OS/400 display data management returns the indication of what command key was pressed to exit application help.

On System/36, if application help is displayed and the display station operator presses a command key to exit application help, the data from the display where the Help key is pressed is returned to the application program. To use this support, the application must indicate *restore yes* (columns 47 and 48 of the SFGR H specification) on the display where the Help key is pressed. OS/400 display data management returns the data from the display where the Help key is pressed if both the display file where the Help key is pressed and the display file containing application help specify the USRDSPMGT keyword.

Cancel-Invite Operation

On System/36, if a program issues a cancel-invite operation and data has already been received by the system (for example, the display station operator pressed the Enter key before the cancel-invite was issued), the data is lost and the application is not notified.

The default OS/400 display data management support causes the cancel-invite operation to fail if the data is available. Message CPF4737 and return code 0412 are sent to the application program. The application can check the return code and issue a read operation to receive the data that was returned from the display station.

OS/400 display data management supports the System/36 function and does not send message CPF4737 or set the return code for programs written in a System/36-compatible language or for display files that have the USRDSPMGT keyword specified.

Retain Command and Function Keys

On System/36, when a program issues a write operation to display a record, the application can indicate that the command and function keys currently active for the display station should be the command and function keys that are used for the new record. For example, if an application writes record REC1 (where REC1 enables command keys 1 through 12 and all of the function keys), and then the application writes record REC2 (where REC2 retains the command and function keys from REC1), then command keys 1 through 12 and all of the function keys are enabled when REC2 is displayed. System/36 applications indicate that the command keys should be retained by specifying an R for the enable command keys option in the SFGR source (column 28 of the SFGR S specification). System/36 applications indicate that the function keys should be retained by specifying an R for the enable function keys option in the SFGR source (column 27 of the SFGR S specification).

AS/400 application programs can specify either a list of command and function keys that are valid when a record is displayed or an indication that the command and function keys that are currently active should remain active when a record is displayed.

OS/400 display data management allows an application to retain the current command keys for a record if the RETCMDKEY keyword is specified for the record. The RETCMDKEY keyword is automatically generated by the System/36 environment if R is specified for command keys in the SFGR S specification.

Also, OS/400 display data management allows an application to retain the current function keys for a record if the RETKEY keyword is specified for the record. The RETKEY keyword is automatically generated by the System/36 environment if R is specified for function keys in the SFGR S specification.

System/36 Functions Not Supported

This section describes the System/36 work station data management functions that are not supported by OS/400 display data management.

If one program issues a write operation that invites the display and another program issues a write operation with the put-override function enabled for the same display, all the write and read operations must use record formats from the same display file. For example, if program A sets up a read-under-format (RUF) request by writing record REC1 from file FILE1, the second program must also use FILE1 for any write operations with the put-override function enabled. If a different display file is used, all the input fields from REC1 are changed to output-only fields when the put-override function is handled by OS/400 display data management.

The put-override function can be used only across different jobs (for example, a single-requester-terminal (SRT) program that gives control to a multiple-requester-terminal (MRT) program) when the application programs are using RUF. If RUF is not in progress and the second job issues a write operation with the put-override function enabled, all the input fields on the display are changed to output-only fields when the put-override function is handled by OS/400 display data management.

On System/36, application help can return the data from input fields on an application help display. This is not supported by OS/400 display data management.

Restricted DDS Keywords/Functions

If the USRDSPMGT keyword is specified in the DDS source, the following DDS functions cannot be used:

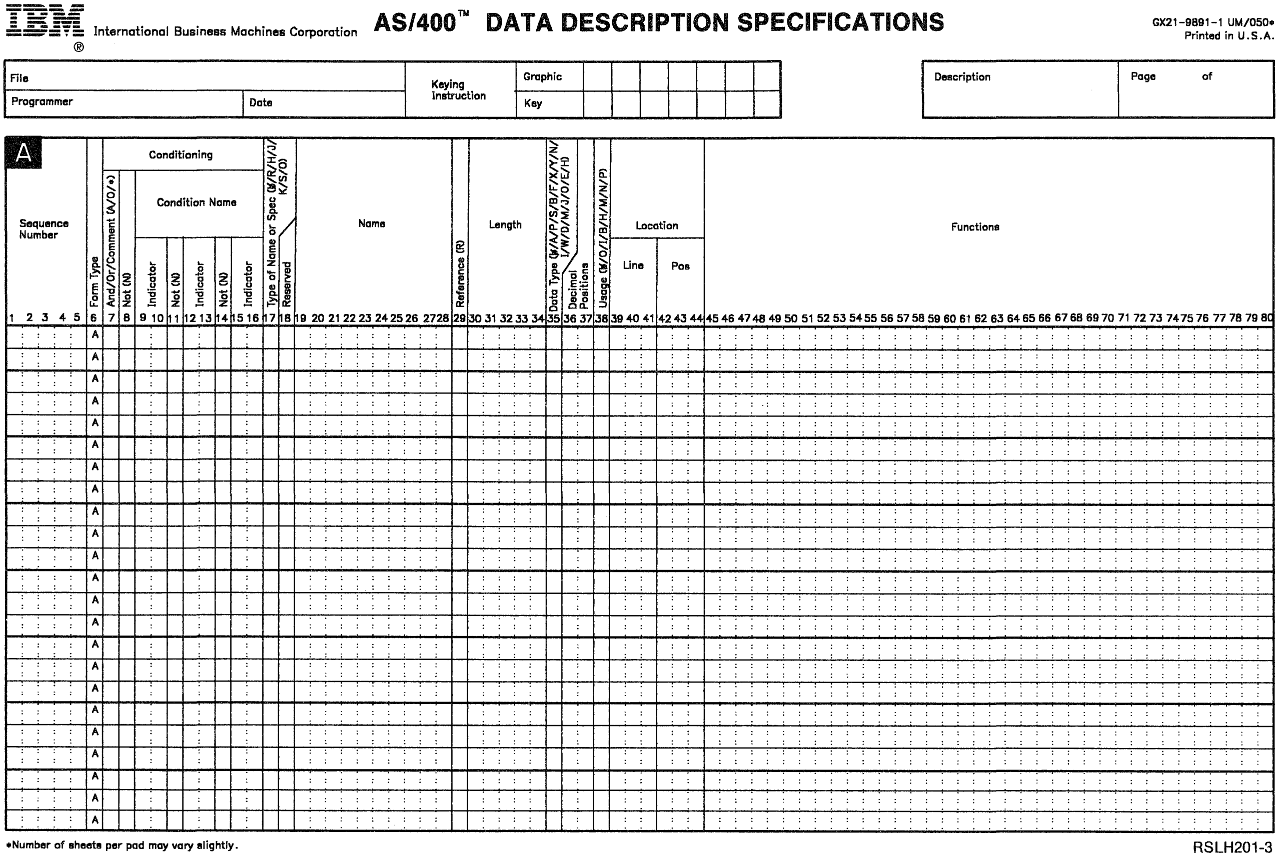
- DDS keywords that control clearing the display:
  - ASSUME
  - ERASE
  - KEEP
  - PUTRETAIN
- Subfile DDS keywords:
  - SFL
  - SFLCTL

  Because System/36-compatible languages do not support subfiles, subfiles cannot be used in a display file if USRDSPMGT is specified.
- Response indicators

  Because System/36-compatible languages do not return indicators from the read operation to a display file, response indicators cannot be specified if USRDSPMGT is used.

Appendix G. DDS Coding Form

The *DDS Coding Form* provides a common format for describing data externally.



International Business Machines Corporation

AS/400™ DATA DESCRIPTION SPECIFICATIONS

GX21-9891-1 UM/050*
Printed in U.S.A.

File			Keying Instruction	Graphic							
Programmer		Date		Key							

Description	Page of

A

Sequence Number	Form Type	Conditioning							Type of Name or Spec (b̸/R/H/J/K/S/O)	Reserved	Name	Reference (R)	Length	Data Type (b̸/A/P/S/B/F/X/Y/N/I/W/D/M/J/O/E/H)	Decimal Positions	Usage (b̸/O/I/B/H/M/N/P)	Location		Functions
		And/Or/Comment (A/O/*)	Not (N)	Condition Name													Line	Pos	
				Indicator	Not (N)	Indicator	Not (N)	Indicator											
1 2 3 4 5	6	7	8	9 10	11	12 13	14	15 16	17	18	19 20 21 22 23 24 25 26 27 28	29	30 31 32 33 34	35	36 37	38	39 40 41	42 43 44	45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80
	A																		
	A																		
	A																		
	A																		
	A																		
	A																		
	A																		
	A																		
	A																		
	A																		
	A																		
	A																		
	A																		
	A																		
	A																		
	A																		
	A																		
	A																		
	A																		
	A																		
	A																		
	A																		

*Number of sheets per pad may vary slightly.

RSLH201-3

Figure G-1. Sample DDS Form

Glossary

access. To read; the ability to use or read.

access path. The order in which records in a database file are organized for processing by a program.

acquire. To assign a display station or session to a program.

active file. A tape or diskette file with an expiration date greater than the system date.

active record. Any record format that is currently displayed or an active subfile record. Contrast with *inactive record*.

active sort table. A system-supplied DBCS sort table that contains the collating sequences for all defined double-byte characters in a double-byte character set. These tables are maintained by the character generator utility function of the AS/400 Application Development Tools licensed program.

active subfile. A subfile in which a write operation was issued to the subfile record format or to the subfile control record format with the DDS keyword SFLINZ in effect.

address. The location in the storage of a computer where particular data is stored. Also, the numbers that identify such a location.

all-points-addressable. Pertaining to the capability to address, refer to, and position text, overlays, and images at any defined point on the printable area of the paper.

allocate. To reserve a resource for use in performing a specific task. Contrast with *deallocate*.

alphabetic character. Any one of the letters A through Z or a through z or one of the characters #, $, or @.

alphameric. Pertaining to the letters, A through Z or a through z; numbers, 0-9; and special symbols, $, #, @, ., or _. Synonymous with *alphanumeric*.

alphanumeric. Pertaining to the letters, A through Z or a through z; numbers, 0-9; and special symbols, $, #, @, ., or _. Synonymous with *alphameric*.

application. A particular business task, such as inventory control or accounts receivable.

application program. A program used to perform a particular data processing task such as inventory control or payroll.

attribute character. A character associated with a field in a display file record format that defines how the field is displayed.

backup. Pertaining to an alternative copy used as a substitute if the original is lost or destroyed.

BASIC (beginner's all-purpose symbolic instruction code). A programming language with a small list of commands and a simple syntax, primarily designed for numeric applications.

basic characters. Frequently used double-byte characters that are stored in the hardware of a DBCS device. The number of double-byte characters that are stored in the device varies with the language supported and the storage size of the device. A DBCS device can display or print basic characters without using the extended character processing function of the operating system. Contrast with *extended characters*. See also *extended character processing*.

batch. Pertaining to a group of jobs to be run on a computer sequentially with the same program with little or no operator action. Contrast with *interactive*.

batch job. A predefined group of processing actions submitted to the system to be performed with little or no interaction between the user and the system. Contrast with *interactive job*.

beginning attribute character. For a display file, the character that precedes the first position in a field and that defines how the data in the field is displayed.

binary. A numbering system with a base of two (0 and 1).

bit. Either of the binary digits, 0 or 1. Compare with *byte*.

block. (1) A group of records that are recorded or processed as a unit. (2) A set of adjacent records stored as a unit on a disk, diskette, or magnetic tape.

buffer. A portion of storage used to hold input or output data temporarily.

byte. A group of 8 adjacent bits. In the EBCDIC coding system, 1 byte can represent a character. In the double-byte coding system, 2 bytes represent a character.

C/400. The IBM licensed program that is the SAA C programming language available on the AS/400 system, including system-specific functions.

call level. The position of a program in a nest of programs called explicitly by the CALL instruction or implicitly by some event. The first program has a call level of 1. Any program called by a level 1 program has a call level of 2, and so on.

character. Any letter, number, or other symbol in the data character set that is part of the organization, control, or representation of data.

character field. An area that is reserved for information that can contain any of the characters in the character set. Contrast with *numeric field*.

character generator utility (CGU). A function of the AS/400 Application Development Tools licensed program that is used to define and maintain user-defined double-byte characters and related sort information.

character set. A group of characters used for a specific reason; for example, the set of characters the computer can display, the set of characters a printer can print, or a particular set of graphic characters in a code page; for example, the 256 EBCDIC characters.

character string. A sequence of consecutive characters that are used as a value.

characters per inch (cpi). The number of characters printed horizontally within an inch across a page.

check digit. The far right number of a self-check field used to verify the accuracy of the field.

CL. See *control language (CL)*.

close. The function that ends the connection between a file and a program, and ends the processing. Contrast with *open*.

code point. One of the 256 values assigned to a character in a character set. On the AS/400 system, a code point is represented by a hexadecimal number. For example, in code page 256 (EBCDIC), the letter "e" is assigned a code point of hex 85.

code-page ID. A 5-digit registered identifier used to specify a particular assignment of code points to graphic characters. The code-page ID is the second part of the QCHRID system value or the CHRID parameter value.

collating sequence. The order in which characters are arranged within the computer for sorting, combining, or comparing.

command. A statement used to request a function of the system. A command consists of the command name, which identifies the requested function and parameters.

command attention (CA) key. A keyboard key that can be specified with the CA keyword to request the function specified by the keyword. Data is not returned to the system. Contrast with *command function (CF) key*.

command definition. An object that contains the definition of a command (including the command name, parameter descriptions, and validity checking information) and identifies the program that performs the function requested by the command. The system-recognized identifier for the object type is *CMD.

command function (CF) key. A keyboard key that can be specified with the CF keyword to request the function specified by the keyword. Data is returned to the system. Contrast with *command attention (CA) key*.

command processing program (CPP). A program that processes a command. This program performs some validity checking and processes the command so that the requested function is performed.

commit. To make all changes permanent that were made to one or more database files since the last commit or rollback operation, and make the changed records available to other users.

commitment control. A means of grouping file operations that allows the processing of a group of database changes as a single unit through the Commit command or the removal of a group of database changes as a single unit through the Rollback command.

compatibility. Ability to work in the system or ability to work with other devices or programs.

compatible. Pertaining to the characteristics that make devices, programs, products, or systems work together.

compilation. Translation of a source program (such as RPG/400 or COBOL specifications) into a program in machine language.

compile. To translate a program written in a high-level programming language into a machine-language program.

compile time. The time during which a source program is translated by a compiler into a machine-language program.

compiled program. The set of machine language instructions that is the output from the compilation of a source program. The actual processing of data is done by the machine-language program.

compiler. A program that translates programming language into machine language for use by the computer.

completion message. A message that tells the operator when work is successfully ended.

composite key. A key for a file or record format that is composed of more than one key field.

compression. A function that removes duplicate characters from the data being processed and replaces the duplicate characters with control characters. Compression reduces the amount of storage space required for the data.

concatenate. (1) To link together. (2) To join two character strings.

concept. An abstract idea.

condition name. For display files, a name used to control the selection of DDS keywords and display locations based on the display size specified for the display file.

conditioning. The use of indicators in a program to control when calculations or output operations are done, or in a file the use of indicators or condition names to control when certain functions or operations are done.

configuration. The physical and logical arrangement of devices and programs that make up a data processing system. See also *device configuration*.

conform. To change to a prevailing standard.

constant. Data that has an unchanging, predefined value to be used in processing.

constant field. In an externally described display or printer file, an unnamed field that contains actual data that is passed to the display or printer but is unknown to the program passing it.

control language (CL). The set of all commands with which a user requests system functions.

control language (CL) program. A program that is created from source statements consisting entirely of control language commands.

control language (CL) variable. A program variable that is declared in a control language program and is available only to the CL program.

controller. A device that coordinates and controls the operation of one or more input/output devices (such as work stations) and synchronizes the operation of such devices with the operation of the system as a whole.

controller description. An object that contains a description of the characteristics of a controller that is either directly attached to the system or attached to a communications line.

conversation. In interactive communications, the communication between the application program and a specific item (usually another application program) at the remote system.

creation date. The system date when an object is created. See also *system date*.

currency symbol. A character such as the dollar sign ($) used to identify monetary values.

cursor. A movable symbol, often a blinking or solid block of light, that tells the user where to type, or identifies a choice to select.

cylinder. The tracks on a diskette that can be read without changing the position of the read/write head.

data authority. A specific authority to read, add, update, or delete data.

data definition. A definition that describes a data object, reserves storage for a data object, and can provide an initial value for a data object. A data definition appears outside a function or at the beginning of a block statement.

data description specifications (DDS). A description of the user's database or device files that is entered into the system in a fixed form. The description is then used to create files.

data dictionary. An object for storing field, record format, and file definitions.

data file. (1) A collection of related data records organized in a specific order. (2) A file created by the specification of FILETYPE(*DATA) on the create commands. Contrast with *source file*.

data item. A unit of information to be processed.

data management. The part of the operating system that controls the storing and accessing of data to or from an application program. The data can be on internal storage (for example, database), on external media (diskette, tape, or printer), or on another system.

data stream. All information (data and control commands) sent over a data link usually in a single read or write operation.

data type. A characteristic used for defining data as numeric or character.

database. The collection of all data files stored in the system.

database file. An object that contains descriptions of how input data is to be presented to a program from internal storage and how output data is to be presented to internal storage from a program. See also *physical file* and *logical file*.

DBCS. See *double-byte character set (DBCS)*.

DBCS code. The hexadecimal code, 2 bytes in length, that identifies a double-byte character.

DBCS conversion. A function of the operating system that allows a DBCS display station user to enter alphameric data and request that the alphameric data be converted to double-byte data.

DBCS conversion dictionary. A collection of alphameric entries with the double-byte entries that correspond to the alphameric entries. It is used by the DBCS conversion function.

DBCS font table. A system-supplied table that holds either 24x24 or 32x32 character images of a double-byte character set. Japanese 24x24 DBCS font table holds Japanese extended Kanji and user-defined characters. Korean 24x24 DBCS font table holds a subset of Hanja and user-defined characters. Traditional Chinese 24x24 DBCS font table holds a subset of primary Traditional Chinese, all secondary Chinese, and user-defined characters. Simplified Chinese 24x24 DBCS font table holds user-defined characters. A 32x32 DBCS font table holds character images of a complete double-byte character set, including its user-defined characters.

DBCS sort table. A system-supplied object that contains sequencing information to sort double-byte characters. See also *master sort table* and *active sort table*.

DDM. See *distributed data management (DDM)*.

DDM file. A file description, created by a user on the local (source) system, for a database file that is stored on a remote (target) system. The DDM file provides the information needed for a local system to locate a remote system and to access the data in the remote database file.

DDS. See *data description specifications (DDS)*.

deallocate. To release a resource that is assigned to a specific task. Contrast with *allocate*.

debug mode. An environment in which programs can be tested.

decimal position. The location of the decimal point in a series of numbers.

default. A value automatically supplied or assumed by the system or program.

default printer. A printer that is assigned to a system or user and accepts all the printed output from that system or user, if no other printer is specified.

default reply. A system-assigned reply to an inquiry or notify message, which is used when the message queue at which the message arrives is in default delivery mode.

default value. (1) A value supplied by the system that is used when no value is specified by the user. (2) The value specified by the user with the DFT keyword in DDS.

delayed maintenance. A method of logging changes to an access path for database files and applying the changes the next time the file is opened instead of rebuilding the access path completely or maintaining it immediately.

delimiter. A character or sequence of characters that marks the beginning or end of a unit of data.

device configuration. The physical placement of display stations, printers, and so forth; and the configuration descriptions that describe the physical configuration to the system and describe how the configuration will be used by the system.

device description. An object that contains information describing a particular device or logical unit that is attached to the system.

device file. A file that contains a description of how data is to be presented to a program from a device or how data is to be presented to the device from the program. Devices can be display stations, printers, a diskette unit, tape units, or a remote system.

device name. The symbolic name of an individual device.

diagnostic. Pertaining to the detection and isolation of an error.

diagnostic message. A message that contains information about errors or possible errors. This message is generally followed by an escape message.

digit. Any of the numerals from 0 through 9.

disk. A direct-access storage medium with magnetically recorded data.

diskette. A thin, removable magnetic disk in a protective jacket.

diskette file. A device file created by the user for a diskette unit.

diskette writer. A function of the operating system that writes the spooled output from a program to a diskette unit. See also *printer writer* and *spooling writer*.

diskette 1. A diskette that contains information on only one side.

diskette 2D. A diskette that contains information on both sides, and with twice the amount of information stored in the same space as a diskette 1. Therefore, a diskette 2D holds approximately four times the amount of information as a diskette 1.

display area. For double-byte character set support, the area that is used to display the character currently being defined or changed.

display file. A device file created by the user to support a display station.

display screen. The part of the display device, which is similar to a television (TV) picture tube, used to display information entered or received at a work station.

display station. A device that includes a keyboard from which an operator can send information to the system and a display screen on which an operator can see the information sent to or the information received from the system.

distributed data management (DDM). A function of the operating system that allows an application program or user on one system to use database files stored on remote systems. The systems must be connected by a communications network, and the remote systems must also be using DDM.

document name. The 1- through 12-character name for documents in folders, assigned by the user when creating the document.

dot matrix. In computer graphics, a two-dimensional pattern of dots used for designing an image on the display.

double-byte character set (DBCS). A set of characters in which each character is represented by 2 bytes. Languages such as Japanese, Chinese, and Korean, which contain more symbols than can be represented by 256 code points, require double-byte character sets. Because each character requires 2 bytes, the typing, displaying, and printing of DBCS characters requires hardware and programs that support DBCS. Four double-byte character sets are supported by the system: Japanese, Korean, Simplified Chinese, and Traditional Chinese. Contrast with *single-byte character set*.

draft. A printed copy of a document that is not yet completed.

dump. To copy data from main or auxiliary storage onto an external medium, such as tape, diskette, or printer.

duplex. Pertaining to printing on both sides of a sheet of paper.

edit. (1) To interactively add, change, delete, or rearrange the data; for example, to insert or remove characters, sentences, or paragraphs, or to insert or remove characters in dates or decimal numbers.
(2) To modify a numeric field for output by suppressing zeros and inserting commas, periods, currency symbols, the sign status, or other constant information.

edit code. A letter or number indicating that editing should be done according to a defined pattern before a field is displayed or printed. Contrast with *edit word*.

edit description. A description of a user-defined edit code. The system-recognized identifier is *EDTD.

edit word. A user-defined word with a specific format that indicates how editing should be done. Contrast with *edit code*.

element. In a list of parameter values, one value.

end-of-text (ETX) character. The BSC transmission control character used to end a logical set of records that began with the start-of-text character.

ending attribute character. For a display file, the character following the last position in a field.

error log. A record of machine checks, device errors, and media statistics.

escape character. The symbol used to enclose a delimited identifier. The symbol is the quotation mark ("), except in COBOL programs where the symbol can be assigned by the user as either a quotation mark or an apostrophe.

escape message. A message that reports a condition that caused the program to end before the requested function was complete.

expiration date. (1) The date after which a diskette or tape file is no longer protected from being automatically deleted by the system. (2) The date after which a database file member should not be used.

exponent. (1) A number, indicating to which power another number (the base) is to be raised. (2) In floating-point format, an integer constant specifying the power of ten by which the base of the decimal floating-point number is to be multiplied.

extended character processing. A function of the operating system that is required to make characters stored in a DBCS font file available to a DBCS device. Basic characters, which are stored in the device, do not require extended character processing. Extended characters, which are stored in a DBCS font table, require extended character processing before they can be displayed or printed. See also *basic characters* and *extended characters*.

extended characters. Double-byte characters that are stored in a DBCS font file, not in the hardware of a DBCS device. When displaying or printing extended characters, the device receives them from the DBCS font table under control of the extended character processing function of the operating system. Contrast with *basic characters*. See also *extended character processing*.

externally described data. Data contained in a file for which the fields and the records are described outside of the program (such as with DDS, IDDU, SQL/400), that processes the file. Contrast with *program-described data*.

externally described file. A file in which the records and fields are described to the system when the file is created, and used by the program when the file is processed. Contrast with *program-described file.*

field. A group of related characters (such as name or amount) that are treated as a unit in a record.

field reference file. A physical file that contains no data, only descriptions of fields.

field selection. A function that uses the state of the option indicators to display or print data when a record format is written.

file. A generic term for the object type that refers to a database file, a device file, or a set of related records treated as a unit. The system-recognized identifier for the object type is *FILE.

file name. The name used by a program to identify a file. See also *label*.

file overrides. Attributes specified at run time that change the attributes specified in the file description or in the program.

file separator. The pages produced at the beginning of each output file and used to separate the file from the other files being sent to an output device.

finance device. A device, such as the 4700 Finance Communications System devices and the 3694 Document Processor, that performs functions specifically related to the finance industry. The 3180, 3270, and 5250 work stations are not finance devices.

fixed-length. Pertaining to a characteristic of a file in which all of the records are the same length.

floating currency symbol. A currency symbol that appears immediately to the left of the far left position in an edited field.

floating-point. A mathematical notation in which a quantity of a number is shown as one number multiplied by a power of the number of the base.

fold. To continue data onto the next line. Contrast with *truncate.*

folder. A directory for documents. A folder is used to group related documents and to find documents by name. The system-recognized identifier for the object type is *FLR. Compare with *library*.

font. (1) An assortment of characters of a given size and type style. (2) (SAA) A particular style of type (for example, Bodini or Times Roman) that contains definitions of character sets, marker sets, and pattern sets.

font ID. A number that identifies the character style and size for certain printers.

form type. A 10-character identifier, assigned by the user, that identifies each type of form used for printed output.

format. (1) A defined arrangement of such things as characters, fields, and lines, usually used for displays, printouts, files, or documents. (2) A group of related fields, such as a record, in a file. (3) The arrangement or layout of data on a storage medium, such as disk, tape, or diskette.

function key. A keyboard key that allows the user to select keyboard functions or programmer functions.

generic. Relating to, or characteristic of, a whole group or class.

generic name. The characters common to object names that can be used to identify a group of objects. A generic name ends with an asterisk (*). For example, ORD* identifies all objects whose names begin with the characters ORD.

graphic character. A character that can be displayed or printed. Contrast with *control character*.

graphic character set. A set of graphic characters in a code page.

graphics. (1) Pictures and illustrations. (2) Pertaining to charts, tables, and their creation.

group authority. Authority to use objects, resources, or functions from a group profile.

Hangeul. A written language of Korea. Each Hangeul character is composed of two to six Jamo characters.

Hanja. Chinese characters used in Korean written language.

hardware. Physical equipment, rather than programs, procedures, rules, and associated information.

header. The 8-byte portion of the 520-byte disk sector used by the operating system for control and access information.

header label. A special set of information on a diskette or tape that describes the contents of the diskette or tape.

hex. See *hexadecimal*.

hexadecimal. Pertaining to a numbering system with a base of 16.

high-level language (HLL). A programming language, such as RPG, BASIC, PL/I, Pascal, COBOL, and C used to write computer programs.

high-level language (HLL) pointer. A source pointer that the programmer declares in the user program.

highlight. To define text to be shown in contrast with other text by underlining, italics, bold-face; or on a display, high-intensity (brightness of characters), blinking, or reverse image. You can highlight words, parts of words, or information typed into a form using the text definition function of AS/400 Office.

Hiragana. A graphic character set that consists of symbols used in one of the two common Japanese phonetic alphabets. Each character is represented by 1 byte. Contrast with *Katakana*.

HLL. See *high-level language (HLL)*.

horizontally displayed records. Subfile records that are grouped so that each line on the display shows more than one record of the same record format.

I/O. See *input/output*.

ICF. See *intersystem communications function (ICF)*.

ICF file. A device file that allows a program on one system to communicate with a program on another system. There can be one or more sessions with the same or different communications devices at the same time.

identifier. A sequence of bits or characters that identifies a user, program, device, or system to another user, program, device, or system.

image. An electronic representation of an original document recorded by a scanning device.

implicit. Capable of being understood from something else, though unexpressed.

indicator. (1) A 2-character code that is used by a program to test a field or record or to tell when certain operations are to be performed. (2) An internal switch used by a program to remember when a certain event occurs and what to do when that event occurs.

information (I) frame. An SDLC transmission frame that is sequentially numbered and used to transmit data.

informational message. A message that provides information to the user about the system as compared with a completion message, which indicates success, and escape or diagnostic messages, which indicate failure.

initial program. A user-profile program that runs when the user signs on and after the command processor program QCMD is started. QCMD calls the first program.

initial program load (IPL). The process that loads the system programs from the system auxiliary storage, checks the system hardware, and prepares the system for user operations.

initialize. To set the addresses, switches, or the contents of storage to zero, or to the starting value set by the manufacturer.

inline. Spooled input data that is read into a job by a reader.

inline data file. A file described by a Data command that is included as part of a job when the job is read from an input device. The file is deleted when the job ends.

input field. A field specified in a display file or database file that is used for data the user supplies.

input stream. 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.

input/output. Data provided to the computer or data resulting from computer processing.

inquiry message. A message that gives information and requests a reply.

integer. A positive or negative whole number.

intelligent printer data stream (IPDS). An all-points-addressable data stream that allows users to position text, images, and graphics at any defined point on a printed page.

interactive. Pertaining to the exchange of information between people and a computer. Contrast with *batch*.

interactive data definition utility (IDDU). A function of the operating system that can be used to externally define the characteristics of data and the contents of files.

interactive job. A job started for a person who signs on to a work station. Contrast with *batch job*.

internal storage. All main and auxiliary storage in the system.

intersystem communications function (ICF). A function of the operating system that allows a program to communicate interactively with another program or system.

intrasystem communications. A function that allows two programs that are running in two different jobs on the same system to communicate with each other through an ICF file.

inverse. Opposite in order, nature, or effect.

IPDS. See *intelligent printer data stream (IPDS)*.

job. A unit of work to be done by a computer.

job control authority. A special authority that allows a user to: change, delete, display, hold, and release all files on output queues; hold, release, and clear job queues and output queues; start writers to output queues; hold, release, change, and end other users' jobs; change the class attributes of a job; end subsystems; and start (IPL) the system.

job description. A system object that defines how a job is to be processed. The object name is *JOBD.

job log. A record of requests submitted to the system by a job, the messages related to the requests, and the actions performed by the system on the job. The job log is maintained by the system program.

job queue. A list of batch jobs waiting to be started or processed by the system. The system-recognized identifier for the object type is *JOBQ.

join. An operation that combines data from two or more files using specified fields.

Kanji. Chinese characters used in Japanese written language.

Katakana. A graphic Japanese character set that is used to write foreign words phonetically in Japanese.

key field. A field used to arrange the records of a particular type within a file member.

keyed sequence. The order in which indexed records are read by the program.

keyed sequence access path. An access path to a database file that is arranged according to the contents of key fields contained in the individual records.

keyword. (1) A name that identifies a parameter in a command. (2) (DDS) A name that identifies a function. See also *parameter*.

label. (1) The name of a file on a diskette or tape. (2) An identifier of a command or program statement generally used for branching.

leading zeros. Zeros that are place holders to the left of numbers that are aligned to the right and have fewer positions than the specified field length.

level checking. A function that compares the record format-level identifiers of a file to be opened with the file description that is part of a compiled program to determine if the record format for the file changed since the program was compiled.

library. An object on disk that serves as a directory to other objects. A library groups related objects, and allows the user to find objects by name.

library list. A list that indicates which libraries are to be searched and the order in which they are to be searched. The system-recognized identifier is *LIBL.

library name. A user-defined word that names a library.

licensed program. An IBM-written program that performs functions related to processing user data.

line. The physical path in data transmission.

line number. The number that precedes a line of information in a printout or on a display. This number can be up to 5 digits long, from 00001 through 99999.

lines per inch. The number of characters that can be printed vertically within an inch.

link protocol. The rules for sending and receiving data at the link level.

link protocol converter (LPC). A device that changes one type of link-level protocol information to another type of link-level protocol information for processing.

load. (1) To move data or programs into storage. (2) To place a diskette into a diskette unit. (3) To insert paper into a printer. (4) To put a tape reel or a tape cartridge into a tape unit.

local. Pertaining to a device or system that is connected directly to or a file that is read directly from your system, without the use of a communications line. Contrast with *remote*.

lock. The process by which integrity of data is ensured by preventing more than one user from accessing the same data or object at the same time.

lock state. A condition defined for an object that determines how it is locked, how it is used (read or write), and whether the object can be shared (used by more than one job).

logic. The systematized interconnection of digital switching functions, circuits, or devices.

logical file. A description of how data is to be presented to or received from a program. This type of database file contains no data, but it defines record formats for one or more physical files. Contrast with *physical file*.

logical file member. A named logical grouping of data records from one or more physical file members. See also *member*.

main storage. The part of the processing unit where programs are run. Synonymous with *memory*.

mask. A pattern of characters that is used to control the keeping, deleting, or testing of portions of another pattern of characters.

master file. A collection of permanent information, such as a file of customer addresses.

master sort table. A system-supplied table that contains sort information required for sorting double-byte characters. This table is maintained by the character generator utility function of the AS/400 Application Development Tools licensed program.

matrix. An arrangement in rows and columns.

member. Different sets of data within one file.

message file. An object that contains message descriptions. The system-recognized identifier for the object type is *MSGF.

message identifier. A seven-character code that identifies a predefined message, and is used to get the message description from a message file.

message line. An area on the display where messages are displayed.

message queue. A list on which messages are placed when they are sent to a person or program.

message subfile. A subfile where the records are messages from a program message queue.

microfiche. A photographic negative containing reduced images of pages of a document, arranged in a grid pattern.

migration. The process of moving data and source from one computer system to another without converting the data.

mode. The session limits and common characteristics of the sessions associated with advanced-program-to-program communications (APPC) devices managed as a unit with a remote location.

modified data tag (MDT). An indicator, associated with each input or output field in a displayed record, that is automatically set on when data is keyed into the field. The modified data tag is maintained by the display device and can be used by the program using the file.

monitor. (1) A functional unit that observes and records selected activities for analysis within a data processing system. (2) Devices or programs that observe, supervise, control, or verify system operations.

multiple-device file. A device file in which the maximum number of program devices is greater than one.

multivolume file. A file that occupies more than one diskette or tape.

negative response. In data communications, a reply indicating that data was not received correctly or that a command was incorrect or unacceptable.

network. A collection of data processing products connected by communications lines for exchanging information between stations.

next record. The record that logically follows the current record of a file.

notation. A system of characters, symbols, or abbreviated expressions used to express technical facts or qualities.

notify message. A message that describes a condition for which a program requires a reply from the calling program, or for which a reply is automatically sent to the program.

null. The name for an EBCDIC character that represents hex 00. See *null character*.

null character. The character hex 00 used to represent the absence of a displayed or printed character.

numeric field. An area that is reserved for a particular unit of information and that can contain only the digits 0 through 9. Contrast with *character field*.

object. A named storage space that consists of a set of characteristics that describe itself and, in some cases, data. An object is anything that exists in and occupies space in storage and on which operations can be performed. Some examples of objects are programs, files, libraries, and folders.

object authority. A specific authority that controls what a system user can do with an entire object. For example, object authority includes deleting, moving, or renaming an object. There are three types of object authorities: object operational, object management, and object existence.

object description. The characteristics (such as name, type, and owner name) that describe an object.

offline. Pertaining to the operation of a functional unit that is not under the continual control of the system.

offset. The distance from the beginning of an object to the beginning of a particular field, or for substring operations, the number of character positions from the beginning of a field.

online. Pertaining to the operation of a functional unit that is under the continual control of the system.

online information. Information, read on the display screen, that explains displays, messages, and programs.

open. The function that connects a file to a program for processing.

open data path (ODP). The path that handles all input/output operations for the file.

operating system. A collection of system programs that control the overall operation of a computer system.

operation. The result of processing statements in a high-level language.

option indicator. A 1-character field that is passed with an output data record from a program to the system that is used to control the output function, such as controlling which fields in the record are displayed.

output. Information or data received from a computer that is shown on a display, printed on the printer, or stored on disk, diskette, or tape.

output field. A field specified in a display file or database file that is reserved for the information processed by a program.

output queue. An object that contains a list of spooled files to be written to an output device, such as a printer.

output/input field. A field specified in a database, display, or ICF file that can be used for both the information supplied to the program and the information received from the program during processing. See also *input field* and *output field*.

overflow. The condition that occurs when the last line specified as the overflow line to be printed on a page has been passed.

overlapping fields. Fields in the same display or printer record that occupy the same positions on the display or page. Option indicators can be used to select which of the overlapping fields is to be displayed or printed.

owner. The user who creates an object (or is named the owner of an object).

page. (1) Each group of records in a subfile that are displayed at the same time. (2) One printer form. (3) To move information up or down on the display.

page down. To move up the data shown on the display, which allows the user to move toward the end of the data.

page up. To move down the data shown on the display, which allows the user to move toward the beginning of the data.

parameter. A value supplied to a command or program that is used either as input or controls the actions of the command or program.

pending. Waiting or undecided, as in an operation is pending. A request that was submitted and that is awaiting processing.

physical file. A description of how data is to be presented to or received from a program and how data is actually stored in the database. A physical file contains one record format and one or more members. Contrast with *logical file*.

physical file member. A named subset of the data records in a physical file. See also *member*.

pitch. The number of characters printed per inch.

point. The second byte of a DBCS code, which uniquely identifies double-byte characters in the same ward. Contrast with *ward*.

power down. An AS/400 command to turn the power off and bring an orderly end to system operation.

print band. An interchangeable metal band that contains the print characters used by some printers.

print text. An option that allows the user to specify a line of text at the bottom of a list.

printer file. A device file created by the user to describe a printer device.

printer writer. A function of the operating system that writes the spooled output from a program to a printer. See also *diskette writer* and *spooling writer*.

processing. The action of performing operations and calculations on data.

processor. (1) A device for processing data from programmed instructions. It may be part of another unit. (2) One or more integrated circuits that process coded instructions and perform a task.

profile. Data that describes the characteristics of a user, program, device, or remote location.

program device. A symbolic device that a program uses instead of a real device (identified by the device name). When the program uses a program device, the system redirects the operation to the appropriate real device.

program device override. The attributes specified at run time that change the attributes of the program device.

program message queue. An object used to hold messages that are sent between program calls of a routing step. The program message queue is part of the job message queue.

program stack. A list of programs linked together as a result of programs calling other programs with the CALL instruction, or implicitly from some other event, within the same job.

program-described data. Data contained in a file for which the fields in the records are described in the program that processes the file. Contrast with *externally described data*.

program-described file. A file for which the fields in the records are described only in the program that processes the file. To the operating system, the record appears as a character string. Contrast with *externally described file*.

prompt. (1) A reminder or a displayed request for information or user action. The user must respond to allow the program to proceed. (2) A list of values or a request for information provided by the system as a reminder of the type of information or action required.

protocol. A set of rules controlling the communication and transfer of data between two or more devices in a communications system.

public authority. The authority given to users who do not have any specific (private) authority to an object, who are not on the authorization list (if one is specified for the object), and whose group profile has no specific authority to the object.

qualified name. The name of the library containing the object and the name of the object.

queue. A list of messages, jobs, or files waiting to be read, processed, printed, or distributed in the order they appear in the list.

read authority. A data authority that allows the user to look at the contents of an entry in an object or to run a program.

read operation. An input operation that obtains a record from a file and passes it to a program.

read-from-invited-program-devices operation. An input operation that waits for input from any one of the invited program devices for a user-specified time.

reader. An internal program that reads jobs from an input device or a database file and places them on a job queue.

record. A collection of related data or words, treated as a unit; such as one name, address, and telephone number.

relative record number. A number that specifies the location of a record in relation to the beginning of a database file, member, or subfile. For example, the first record in a database file, member, or subfile has a relative record number of 1.

remote. Pertaining to a device, system, or file that is connected to another device, system, or file through a communications line. Contrast with *local*.

remote location name. Any other system with which your system can communicate in an SNA network. This corresponds to the remote location name specified in the communications configuration.

remote system. Any other system in the network with which your system can communicate.

remote work station. A work station that is connected to the system by data communications.

resolution. In computer graphics, a measure of the sharpness of an image, expressed as the number of lines per unit of length or the number of points per unit of area discernible in that image.

resolve. In programming, to change a predefined, symbolic value to the actual value of the item being processed. For example, a symbolic value of *LAST defined for the name of a file member is resolved to the name of the last member when the member is processed.

resource. Any part of the system required by a job or task, including main storage, devices, the processing unit, programs, files, libraries, and folders.

response indicator. A 1-character field passed with an input record from the system to a program to provide information about the data record or actions taken by the work station user.

restore. To copy data from tape, diskette, or a save file to auxiliary storage. Contrast with *save*.

return code. In data communications, a value sent by the system to a program to indicate the results of an operation by that program.

reverse image. Text that appears on the display in the opposite color (for example, black on green instead of green on black).

routine. A set of statements in a program that causes the system to perform an operation or a series of related operations.

routing step. The processing that results from running a program specified in a routing entry. Most jobs have only one routing step.

RPG/400. An IBM licensed program that is the SAA RPG programming language available on the AS/400 system, including system-specific functions.

save. To copy specific objects, libraries, or data by transferring them from main or auxiliary storage to magnetic media such as tape, diskettes, or a save file.

save file. A file allocated in auxiliary storage that can be used to store saved data on disk (without requiring diskettes or tapes), that can be used in I/O operations from a high-level language program, or can be used to receive objects sent through the network.

screen design aid (SDA). A function of the AS/400 Application Development Tools licensed program that helps the user design, create, and maintain displays and menus.

SCS. See *SNA character string*.

SDA. See *screen design aid (SDA)*.

sector. (1) An area on a disk track or a diskette track to record information. (2) The smallest amount of information that can be written to or read from a disk or diskette during a single read or write operation.

sequence. To arrange in order.

session. (1) The length of time that starts when a user signs on and ends when the user signs off at a display station. (2) In communications, the logical connection by which a program or device can communicate with a program or device at a remote location.

shared file. A file whose open data path can be shared between two or more programs processing in the same routing step. See *open data path*.

shift. A keyboard action to allow uppercase or other characters to be entered.

shift control character. See *shift-in character* and *shift-out character*.

shift-in character. A control character (hex 0F) that indicates the end of a string of double-byte characters. Contrast with *shift-out character*.

shift-out character. A control character (hex 0E) that indicates the start of a string of double-byte characters. Contrast with *shift-in character*.

significant digit. Any number of a series of numbers that follows the far left number, that is not a zero, and that is within the accuracy allowed.

Simplified Chinese. The Chinese character set that has been simplified by reducing the number of strokes in common characters and deleting complicated variants. Simplified Chinese characters are used primarily in the People's Republic of China.

single-byte character set (SBCS). A character set in which each character is represented by a one-byte code. Contrast with *double-byte character set*.

SNA character string (SCS). A data stream composed of EBCDIC controls, optionally intermixed with end-user data, which is carried within a request/response unit.

sort utility. A function of the operating system used to arrange records in a sequence determined by data contained in one or more fields in the record.

source entry utility (SEU). A function of the AS/400 Application Development Tools licensed program that is used to create and change source members.

source file. (1) A file of programming code that is not compiled into machine language. Contrast with *data file*. (2) A file created by the specification of FILETYPE(*SRC) on the Create command. A source file can contain source statements for such items as high-level language programs and data description specifications.

source member. A member of a database source file that contains source statements such as RPG/400, COBOL, BASIC, PL/I, or DDS statements. See also *member*.

source program. (1) A set of instructions that are written in a programming language and must be translated to machine language before the program can be run. (2) In communications, the program that starts a session with a remote system.

source system. The system that issues a request to establish communications with another system. (DDM) The system on which an application program issues a request to use a remote file.

spacebar. A control key for the spacing function.

special authority. The types of authority a user can have to perform system functions, including all object authority, save system authority, job control authority, security administrator authority, spool control authority, and service authority. Contrast with *specific authority*.

specific authority. The types of authority a user can be given to use the system resources, including object authorities and data authorities. See also *object authority* and *data authority*. Contrast with *special authority*.

spool. The system function of putting jobs into a storage area to wait to be printed or processed.

spool control authority. A special authority that allows the user to perform spooling functions, such as display, delete, hold, and release spooled files on the output queue for himself and other users. This authority also allows the user to change the spooled file attributes, such as the printer used to print the file.

spooled file. A file that holds output data waiting to be printed, or input data waiting to be processed by the program.

spooled input file. See *inline data file*.

spooled output file. A file that causes output data to be held for later printing.

spooling. The system function that saves data in a disk file for later processing or printing.

spooling subsystem. A part of the system that provides the operating environment for the programs that read jobs onto job queues to wait for processing and write files from an output queue to an output device. IBM supplies one spooling subsystem: QSPL.

spooling writer. The general name to refer to the function of the diskette writer and printer writer.

statement. An instruction in a program.

subfile. A group of records of the same record format that can be displayed at the same time at a display station. The system sends the entire group of records to the display in a single operation and receives the group from the display in another operation.

subfile control record format. One of two record formats required to define a subfile in DDS. The subfile control record format describes the size of the subfile and the size of the subfile page, and is used by the program to write the subfile to and read the subfile from the display.

subfile record format. One of two record formats required to define a subfile in DDS. The subfile record format defines the fields in a subfile record and is used by the program to perform input, output, and update operations to the subfile.

subsystem. An operating environment, defined by a subsystem description, where the system coordinates processing and resources.

subsystem description. A system object that contains information defining the characteristics of an operating environment controlled by the system.

superscript. A symbol, number, or letter written immediately above and to the right or left of another character. For example, a footnote can be identified in text with a superscript number.

syntax. The rules for constructing a command or statement.

syntax checking. A function of the system, a compiler, the BASIC interpreter, or SEU that checks individual statements for errors in the structure of the statement.

system date. The date assigned in the system values when the system is started.

system object. One of two machine object classifications. Any of the machine objects shipped with the system or any of the operating system objects created by the system.

system profile. The text profile named system that contains formatting and editing options to be used for creating documents.

system time. The elapsed time from the point where the system was started to the current time. If the system time is changed to the local time when the system is started, the current system time is the local time of day.

system unit. A part of a computer that contains the processing unit, and may contain devices such as disk units and tape units.

system value. Control information for the operation of certain parts of the system. A user can change the system value to define his working environment. System date and library list are examples of system values.

Systems Network Architecture (SNA). The description of the logical structure, formats, protocols, and operational sequences that are used for transmitting information units through networks, as well as controlling the configuration and operation of networks.

tape drive. A device used to move the tape and read and write information on magnetic tapes.

tape file. A device file created by the user to support a tape device.

tape mark. A unique mark written on the tape to distinguish file boundaries.

tape reel. A round device on which magnetic tape is wound.

tape volume. A single reel of magnetic tape.

target. In DDM, the remote system where the request for a file is sent.

target system. In a distributed data management (DDM) network, the system that receives a request from an application program on another system to use one or more files located on the target system. Contrast with *source system*.

template. A pattern to help the user identify the location of keys on a keyboard, functions assigned to keys on a keyboard, or switches and lights on a control panel.

throughput. The measure of the amount of work performed by a computer over a period of time, for example, number of jobs per day.

track. A circular path on the surface of a disk, diskette, or tape on which information is magnetically recorded and from which recorded information is read.

Traditional Chinese. The Chinese character set expressed in traditional form. Traditional Chinese characters are used in Taiwan, Hong Kong and some other parts of the world.

transaction. In communications, an exchange between a program on a local system and a program on a remote system that accomplishes a particular action or result. See also *conversation* and *session*.

translation table. (1) A system table that provides replacement characters for characters that cannot be printed. (2) An object that contains a set of hexadecimal characters used to translate one or more characters of data. For example, unprintable characters can be translated to blanks, and lowercase alphabetic characters can be translated to uppercase characters. The system-recognized identifier for the object type is *TBL.

truncate. (1) To cut off data that cannot be printed or displayed in the line width specified or available. Contrast with *fold*. (2) To cut off data that does not fit in the specified field length in a field definition.

unlock. To release an object or system resource that was previously locked, and return it to general availability.

update authority. A data authority that allows the user to change the data in an object, such as a journal, a message queue, or a data area. See also *read authority*.

update operation. An I/O process that changes the data in a record.

user profile. An object with a unique name that contains the user's password, the list of special authorities assigned to a user, and the objects the user owns.

user-defined edit code. A number (5 through 9) indicating that editing should be done on a numeric output field according to a pattern predefined to the system program. User-defined edit codes can take the place of edit words, so that repetitive coding of the same edit word is not necessary.

validity checking. To verify the contents of a field.

variable. A name used to represent data whose value can be changed while the program is running by referring to the name of the variable.

vary off. To make a device, controller, or line unavailable for its normal, intended use.

vary on. To make a device, controller, or line available for its normal, intended use.

volume. A storage medium that is put on or taken off the system as a unit, for example, magnetic tape or diskette.

volume label. The first 80 bytes on a standard tape used to identify the tape volume and its owner. This area contains VOL1 in the first four positions.

ward. A section of a double-byte character set (DBCS) where the first byte of the DBCS codes belonging to that section are the same value. According to IBM standards for DBCS codes, there are 190 wards, and each ward has up to 190 points on which DBCS characters can be assigned. Contrast with *point*.

window. An area of the display screen through which a display or portion of a display is shown.

work station. A device used to transmit information to or receive information from a computer; for example, a display station or printer.

work station controller. An I/O controller card in the card enclosure that provides the direct connection of local work stations to the system.

write operation. An output operation that sends a processed record to an output device or output file.

writer. The part of the operating system spooling support that writes spooled output to an output device independently of the program that produced the output.

writing. The action of making a permanent or temporary recording of data in a storage device or on a data medium.

zero suppression. The substitution of blanks for leading zeros in a number. For example, 00057 becomes 57 with zero suppression.

5250 display station. Any display station from the IBM 5250 Information Display System or the 5290 Display System; or the 3180 display station. A 3270 display station is not a 5250 display station.

Index

Application System/400™ Programming:
Data Management Guide

SC21-9658-1

READER'S COMMENT FORM

Please use this form only to identify publication errors or to request changes in publications. Direct any requests for additional publications, technical questions about IBM systems, changes in IBM programming support, and so on, to your IBM representative or to your IBM-approved remarketer. You may use this form to communicate your comments about this publication, its organization, or subject matter, with the understanding that IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

□ If your comment does not need a reply (for example, pointing out a typing error), check this box and do not include your name and address below. If your comment is applicable, we will include it in the next revision of the manual.

□ If you would like a reply, check this box. Be sure to print your name and address below.

Page number(s): Comment(s):

Please contact your IBM representative or your IBM-approved remarketer to request additional publications.

Name ______________________

Company or
Organization ______________________

Address ______________________

____
City State Zip Code

Phone No. ______________________
Area Code

No postage necessary if mailed in the U.S.A.

Cut or Fo
Along Li

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

International Business Machines Corporation
Information Development
Department 245
3605 North Hwy 52
ROCHESTER MN 55901-9986

Fold and Tape

Please do not staple

Fold and Tape

Cut or Fo
Along Li

Application System/400™ Programming:
Data Management Guide

SC21-9658-1

READER'S COMMENT FORM

Please use this form only to identify publication errors or to request changes in publications. Direct any requests for additional publications, technical questions about IBM systems, changes in IBM programming support, and so on, to your IBM representative or to your IBM-approved remarketer. You may use this form to communicate your comments about this publication, its organization, or subject matter, with the understanding that IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

□ If your comment does not need a reply (for example, pointing out a typing error), check this box and do not include your name and address below. If your comment is applicable, we will include it in the next revision of the manual.

□ If you would like a reply, check this box. Be sure to print your name and address below.

Page number(s): Comment(s):

Please contact your IBM representative or your IBM-approved remarketer to request additional publications.

Name ______________________________

Company or
Organization ______________________________

Address ______________________________

____
City State Zip Code

Phone No. ______________________________
Area Code

No postage necessary if mailed in the U.S.A.

Cut or F
Along Li

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

International Business Machines Corporation
Information Development
Department 245
3605 North Hwy 52
ROCHESTER MN 55901-9986

Fold and Tape

Please do not staple

Fold and Tape

IBM®

Cut or F
Along L

IBM®

Program Number
5728-SS1

21F2722

SC21-9658-1

Printed in U.S.A.