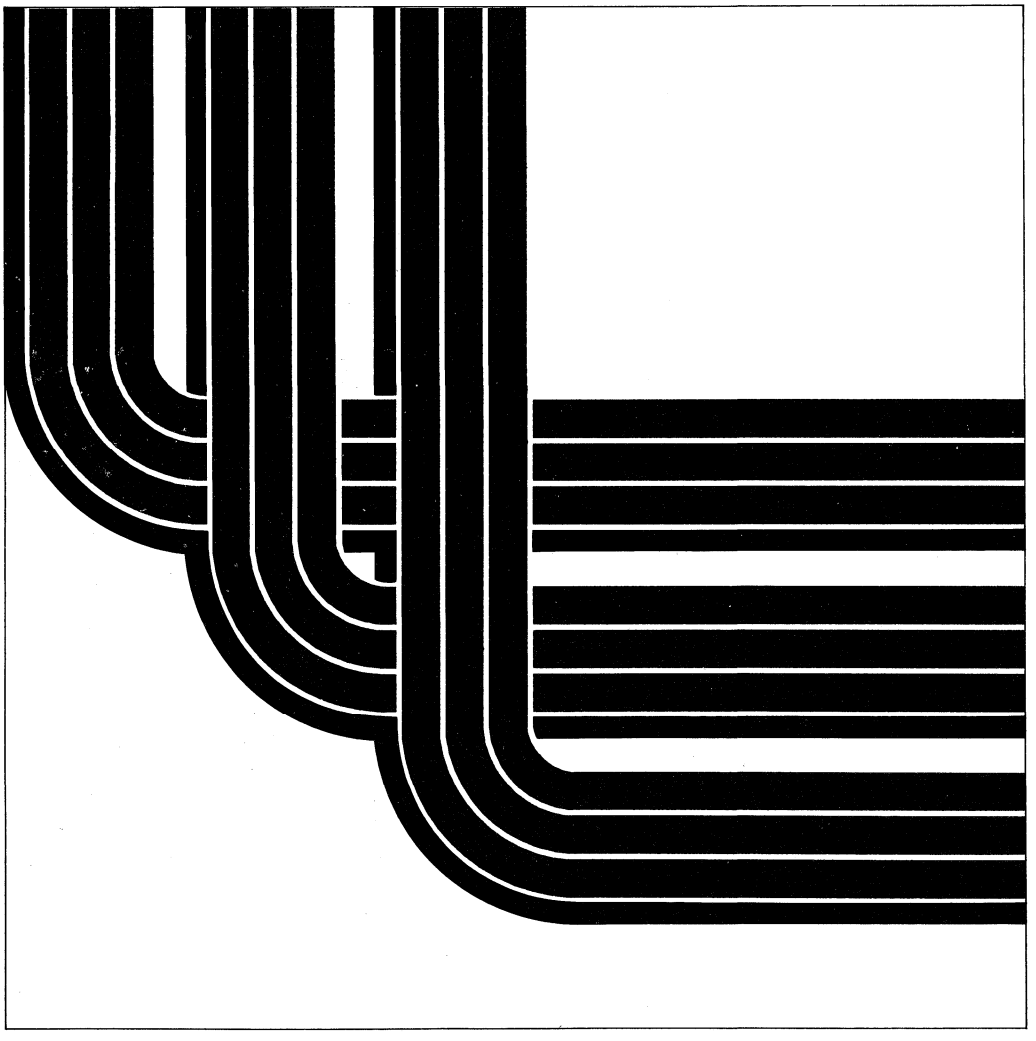
Application System/400™

SC41-9852-00

Application Development by Example

Version 2



Application Development

Take Note!

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

First Edition (May 1991)

This edition applies to the licensed program IBM Operating System/400, (Program 5738-SS1), Version 2 Release 1 Modification 0, and to all subsequent releases and modifications until otherwise indicated in new editions. Make sure you are using the proper edition for the level of the product.

Order publications through your IBM representative or the IBM branch serving your locality. Publications are not stocked at the address given below.

A form for readers' comments is provided at the back of this publication. If the form has been removed, you may address your comments to:

Attn Department 245
IBM Corporation
3605 Highway 52 N
Rochester, MN 55901-7899

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

Contents

Notices

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates.

Any reference to an IBM licensed program or other IBM product in this publication is not intended to state or imply that only IBM's program or other product may be used.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Commercial Relations, IBM Corporation, Purchase, NY 10577.

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

Application System/400	AS/400	IBM
Operating System/400	OS/400	RPG/400
SQL/400	400	

This publication could contain technical inaccuracies or typographical errors.

This manual may refer to products that are announced but are not yet available.

This manual contains small programs that 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.

About This Manual

This manual presents a hypothetical mailing list application that illustrates how you design and create application programs on the AS/400 system. This simple application was chosen because it is easy to understand. The application being performed is of minor importance. The major intent of this manual is to:

- Show how to approach an application design
- Understand the basic tools and requirements
- Understand some of the coding techniques which are common to all applications.

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

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 for the application programmer.

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 the OS/400 licensed program. You should also be familiar with the display stations and printers you are using.

You may need to refer to other IBM manuals for more specific information about a particular topic. The *Publications Guide*, GC41-9678, provides information on all the manuals in the AS/400 library.

For a list of related publications, see the "Bibliography."

Chapter 1. Introduction

This manual presents an application to help you understand designing and creating application programs on the AS/400* system. The mailing list application described in this manual is not *real* from several viewpoints but it is used because it is simple in concept and yet it contains many of the basic requirements of any application.

This chapter gives an overview of application programming tasks and how they relate to the contents of each of the following chapters in this manual.

The discussion in each subsequent chapter assumes a knowledge of the previous chapter. Therefore, the chapters should be read in sequence. The AS/400 system functions are introduced as they are needed for a particular application approach followed by the actions you should do to actually develop the sample application as you follow the instructions in this manual.

This manual demonstrates *an approach* to developing an application using various system functions, languages, and utilities. Though there may be several ways to perform a particular function, all ways will not be shown. For example, RPG is the high-level language that is used. However, you do not need to know RPG in order to use this manual. The discussions in this manual cover system-level functions and utilities that will be useful no matter what AS/400 high-level language you plan to use.

Within the program descriptions throughout the manual, RPG-specific and general programming techniques are highlighted.

The appendixes provide additional information on various topics.

The best way to use this manual is to do the tasks described in each chapter on your system.

Overview of Application Programming Tasks

As you develop the mailing list application in this manual, you will go through some of the basic tasks that are associated with application programming. Figure 1-1 on page 1-2 shows the basic tasks that a programmer normally goes through when developing an application from start to finish. A brief discussion of each task follows the figure.

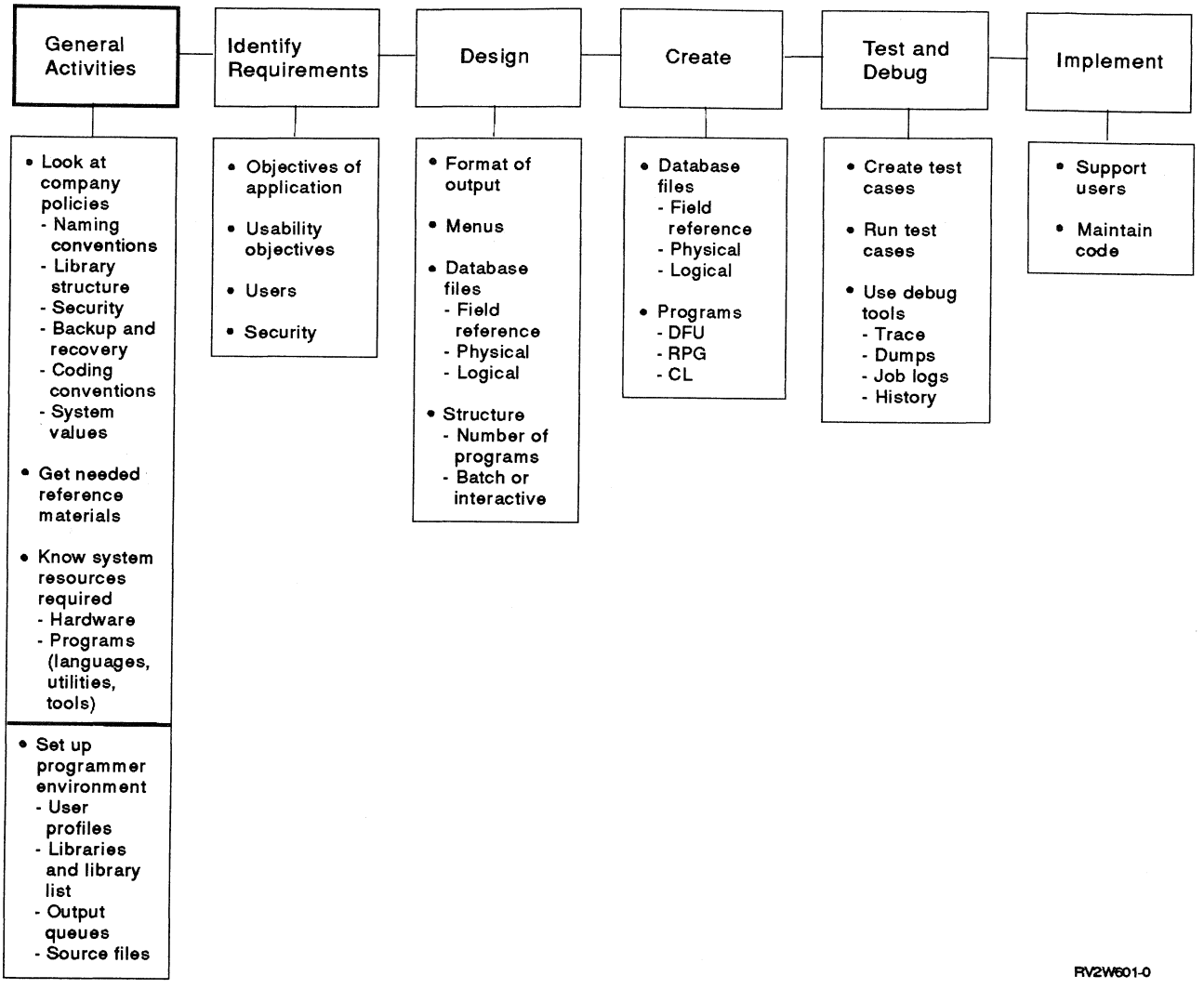


Figure 1-1. Overview of Application Programming Tasks

General Activities

There are many things that must be considered when creating an application program. General activities include determining if you have policies concerning naming conventions, library structures, coding conventions and so on.

Some of the other general activities are discussed in this section.

Security

You should think of security in terms of the users of the application. This is discussed under "User and Security Considerations" on page 4-2.

General Recovery Considerations

Backup and recovery must be considered when developing any application. There are many different recovery situations. Specific recovery alternatives will be discussed throughout the manual.

Reference Materials

Another item you should consider under general activities is any reference materials you will need or concepts you should know about when developing an application.

For the examples in this manual, programs are often simplified by using the system defaults for file and library names, queues, and system functions. You may want to refer to the *CL Reference* for an explanation of the defaults for each command.

System Requirements

You should know what system resources you have available. To actually do all of the steps involved with the mailing list application, you should have any model of the AS/400 system plus the following:

- Licensed Programs:
  - Operating System/400* (Program 5738-SS1)
  - SAA RPG/400* (Program 5738-RG1)
  - AS/400 Application Development Tools (Program 5738-PW1)
- Disk storage sufficient for:
  - The machine product
  - The licensed programs
  - Objects created for the application
  - Data required for the files

Programmer Environment

Setting up the programmer environment is another item that should be looked at under general activities. Setting up the programmer environment usually includes defining and creating:

- User profiles
- Libraries and the library list
- Source files
- Output queues

Chapter 2 goes through the tasks of setting up the programmer environment for the mailing list application.

Identify Requirements

Identifying requirements for the application helps clarify what programs, reports, data files, and other objects are required. This includes defining the following:

- Purpose of the application: What does the application need to do? What output or reports need to be produced?
- Users of the application: Who will be using the application? How many users will there be?
- Usability objectives: What types of menus or displays need to be produced?

- Security needed: Who should be able to update files? Who should be able to read the data? Who should be able to run the programs?
- Performance: What should be done interactively or in batch? What is the volume of updates expected?

When developing any application, you will need to get the objectives, understand the volumes, and get agreement on a proposed solution. This is generally an iterative process. As more information is provided, you can respond with more specifications as to how the application will be done.

In general, it is desirable to get as much agreement as possible before any programs are written. In practice, this can be difficult to achieve. Many end users cannot properly state requirements until they see things that are working or gain experience with a working product. Sometimes a prototype may be needed to assist the end users in determining what is needed.

Chapter 4 identifies the requirements used for the mailing list application.

Design

Once you have gone through the general activities and identified the requirements of the application, you can begin to design the application. Designing an application includes determining:

- If field reference files should be used
- Methods and format of input and output
- What menus or displays should be used
- What database files to use:
  - Physical
  - Logical
  - Source
- Performance considerations
- Security considerations
- Structure of programs:
  - Placement of data
  - Number of programs
  - Batch or interactive programs
- Implementation tools

Chapter 4 discusses the design considerations for the mailing list application.

Create

Once the application is designed, the files used in the application must be described and created, and the various programs must be coded and compiled. The creation phase includes the creation of the following:

- Database files:
  - Source
  - Field reference
  - Physical
  - Logical
- Programs:
  - CL
  - RPG
  - DFU
  - Other

Chapters 5, 6, 7, and 9 go through the task of creating the files and programs used in this manual.

Chapters 10, 11, 12, and 13 go through the tasks of creating an interactive inquiry program, an interactive maintenance program, a menu, and a name search program. These programs help to maintain the application and enhance the application database already defined.

Chapter 14 creates a display using SDA.

Test and Debug

Once the files and programs have been created and are running, test cases should be created and run. This step is normally ongoing throughout the create step.

Chapter 8 goes through the task of creating some of the normal errors that occur and debugging the programs. This chapter also discusses some of the debugging tools.

Implement

The task of implementation is putting the application into production. Once an application is put into production, you should consider the following:

- Maintenance of the application
- Updates to the application
- Enhancements to the application

This manual does not cover this step.

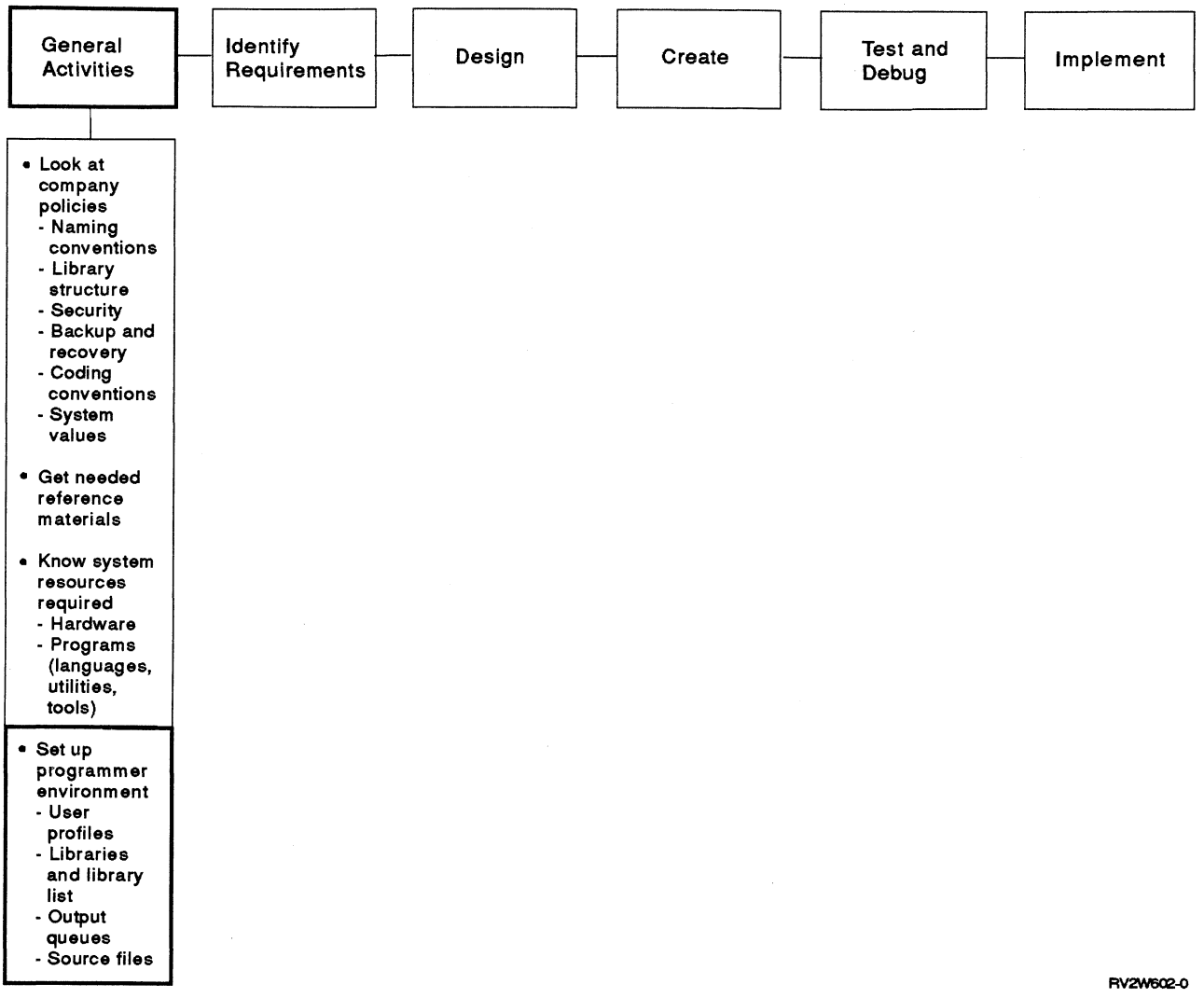
Source in QUSRTOOL Library

The source statements used for the application in this manual are contained in the QUSRTOOL library. The QUSRTOOL library is an optionally installed library that is shipped with the system. It must be installed on your system to use it.

The purpose of this library is to provide you access to examples of various tools and programming techniques that may help you with application development and management of your system. These are not integrated system functions, are not considered part of the OS/400* licensed program, and may change from release to release.

You can copy the source from the QUSRTOOL library instead of retyping it. For more information on the QUSRTOOL library and an example of how to copy the source into your own files to use, see Appendix B, "Overview of QUSRTOOL Library."

Chapter 2. Setting Up the Environment



When you are designing and creating applications, there are many tasks that need to be done that are basic tasks for any application. Included in these tasks are ones associated with setting up the environment in which you will operate.

System Preparation

The application approaches discussed in this manual assume that you or your system operator have prepared the system for daily operations. This means that the system has been powered on and the proper subsystems have been started. For more information on this, see the *Operator's Guide.*

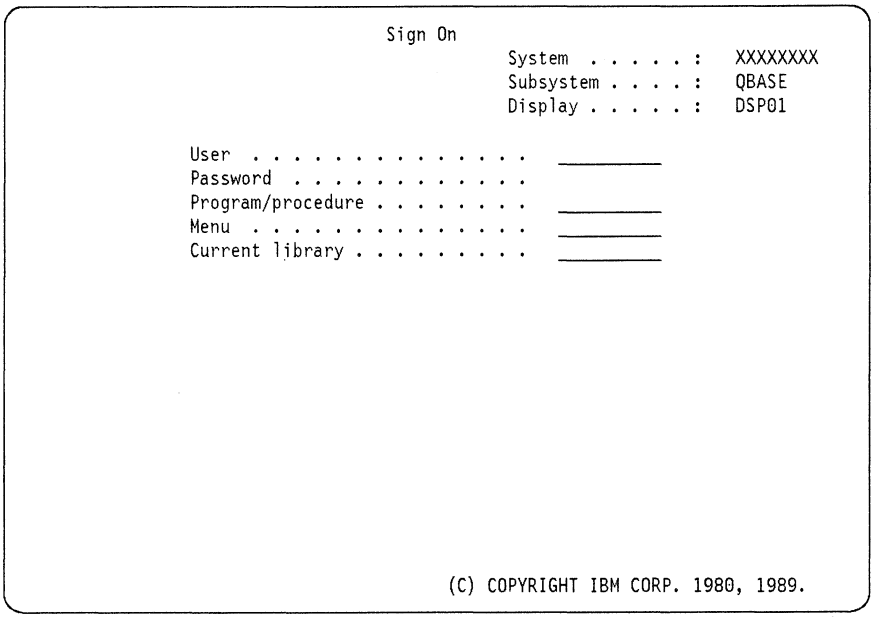
If your system requires passwords, it is assumed that the security officer has created a user profile and given you a password.

Step 1. Signing On

Follow these steps to sign on to the system.

1. Turn on the power to your work station.

   The sign-on display appears first on your screen. The cursor is automatically positioned for you to enter your user profile name.



2. Enter your user profile name. A password must also be entered if security is active on your system.

   Note: The *Password* prompt is shown if security is active on the system. Your password does not appear on the screen as you key it in (so that no one can see the password you are using).

Overview of Libraries

A *library* is an object that serves as a directory to other system objects. It is used to group related objects and to find objects by name when they are used.

On the AS/400 system, a library can contain objects of different types, including programs, data files, source files, and other kinds of objects. The name of an object must be unique for each object of a particular type within a library. For example, two files in the same library cannot have the same name, but a file and a program in the same library can have the same name as shown in Figure 2-1.

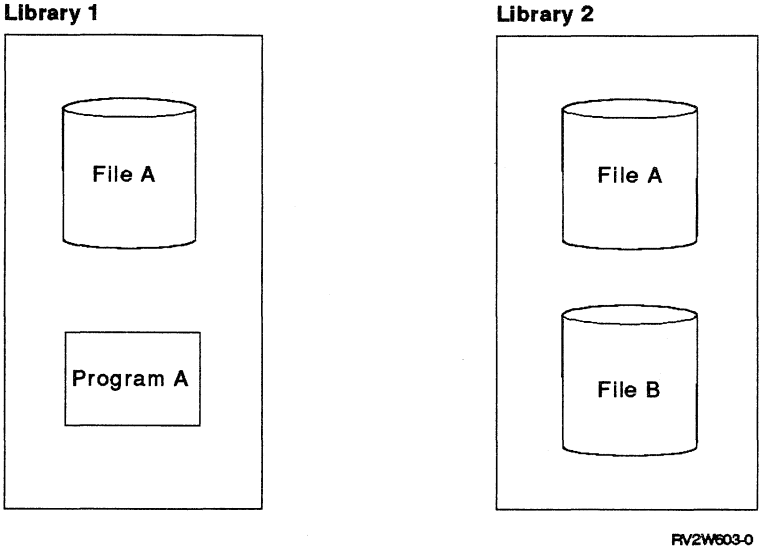


Figure 2-1. Overview of Objects in Libraries

The system supports the concept of a *current* library for each user. A simple use of the library structure is to place all of the required objects for a given application into the current library. Since the current library is searched before any other user-defined libraries, you can avoid any need to understand the library list concept. The mailing list application will assume this simple case.

For more information on libraries and library lists, refer to Appendix A, "Overview of Libraries and Library Lists."

Step 2. Creating a Library

After signing on, the next display you see will either be the Main Menu or the initial display specified in your user profile. The next step is to set up a library.

Note: You will need to be on a display that has a command line in order to follow the steps described.

A single library is used to group all of the objects that are used for the example application in this manual.

Your security officer may have already created a library for you to use. If you already have a library, it is generally desirable to use that library for the mailing list application. If you do not have a library or want to use a separate library for the mailing list application, go through the following steps.

This manual assumes that your initial display is the Main Menu supplied by IBM.

```
MAIN                          AS/400 Main Menu
                                                    System:   XXXXXXXX
Select one of the following:
     1. User tasks
     2. Office tasks
     3. General system tasks
     4. Files, libraries, and folders
     5. Programming
     6. Communications
     7. Define or change the system
     8. Problem handling
     9. Display a menu
    10. User support and education

    90. Sign off


Selection or command

===> CRTLIB
F3=Exit    F4=Prompt    F9=Retrieve    F12=Cancel    F13=User support
F23=Set initial menu
```

1. To create a library, use the Create Library (CRTLIB) command. Type CRTLIB on the command entry line and press F4 to be prompted for the required information. In the following example and throughout this manual, where USERXX is used, you should use the name of the library you plan to use.

   Note: System libraries start with the letter Q and it is not good practice for user-defined libraries to start with that letter. Therefore, you should not name your library with anything that starts with a Q.

```
                        Create Library (CRTLIB)

Type choices, press Enter.

Library . . . . . .  USERXX        Name
Library type . . . .  *PROD         *PROD, *TEST
Text 'description' .  USERXX personal library

                                                                  Bottom
F3=Exit   F4=Prompt   F5=Refresh   F10=Additional parameters   F12=Cancel
F13=How to use this display        F24=More keys
```

2. On the Create Library display, enter the library name and a text description of the library. The Library type defaults to *PROD for *production*. Press the Enter key.

   You should receive a message that your library was created.

Conventions Used for Entering Commands and Parameters

You can enter commands and parameters in a number of different ways on the system.

- If you are unfamiliar with the parameters of a command, you can type the command on any command entry line and press F4 to prompt you to fill in the parameters. The command prompter function helps you enter correct information and minimize keystrokes. The command prompter function also provides help for each parameter to help you fill in the values.
- If you are familiar with the command and know its parameters, you can simply type the command, its parameters and values on a command entry line. When you type a parameter followed by a value that is enclosed in parentheses, it is called a parameter in *keyword form*.

  The normal format that is used when typing commands like this includes a space between parameters and no space between the parameter value and the parentheses. The CRTLIB command would look like:

  ```
  CRTLIB LIB(xxx)
  ```

- If you know the *positional form* of the command, you can type a command followed by the parameter values excluding the parameter names. The parameters can only be entered in the sequence shown in the syntax diagram for the command (shown in the *CL Reference*). Values are separated by one or more blanks. The CRTLIB command would look like:

  ```
  CRTLIB USERXX *PROD TEXT('USERXX personal library')
  ```

  Notice the apostrophes surrounding the text entry. The TEXT parameter expects a single value to be entered. If there were no apostrophes, the system would consider each blank (space) to be the end of a value. The apostrophes tell the system to treat everything within the apostrophes as a single value.

  Note: The TEXT parameter appears on most change and create commands and is used to briefly describe the object being created or changed. The description can be up to 50 characters and must be enclosed in apostrophes if it contains any blanks or special characters, unless you are using the prompter. The prompter does not require you to enter the apostrophes when entering text descriptions.

In many of the discussions in this manual, the command, parameter, and value will be shown entered on the command line. You can choose this convention or simply enter the command and use the prompter function.

Throughout this manual, the values you should enter or view are highlighted and underlined on the displays. This is for usability only; they are not actually highlighted on the displays.

Step 3. Changing Your Current Library

The current library is the library that is specified to be the first user library searched for objects requested by a user. Use the Change Current Library (CHGCURLIB) command to change the current library to be the new library you just created (or the one you plan to use) as follows:

Type CHGCURLIB CURLIB(xxx) on the command entry line where xxx is the name of the library you just created, such as:

```
CHGCURLIB CURLIB(USERXX)
```

Overview of Output Queues

The processing you do normally results in spooled output records that may be produced on an output device. These output records are stored in spooled output files until they should 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.

You should have a special output queue so all of your spooled output is always sent to that queue. This allows you to use certain commands to display the spooled output at your work station and then selectively print or delete the spooled output. Normally, a programmer would not print all of the spooled output that is created.

Step 4. Creating an Output Queue

If you don't already have a personal output queue, create your own output queue using the Create Output Queue (CRTOUTQ) command.

Type the following on the command entry line (USERXX is used as the output queue name in the following example):

```
CRTOUTQ OUTQ(USERXX) TEXT('USERXX personal output queue')
```

Overview of Message Queues

When a user profile is created, the system automatically creates a message queue of the same name. The message queue is used for either messages being sent by the system or by other users.

When you submit batch work (such as compilation of source), the system will send a message to your user message queue describing whether the job completed normally (the object was created) or the job completed abnormally (there were errors in the source that prevented a successful creation).

When you sign on, your user message queue is in *notify mode* meaning you will not see your messages unless you issue the Display Message (DSPMSG) command. Enter the command now to see how a message queue appears.

```
DSPMSG
```

Normally, after you read your messages, you would clear them by using F13. Press F3 to return to your original display.

Rather than using the DSPMSG command each time you want to see your messages, many users prefer to be informed immediately when a message arrives. This is known as *break mode*. The next section describes how to change your user profile to specify *break mode*.

Step 5. Changing Your User Profile to Automatically Set Up Your Environment When You Sign On

Each time you sign on, you would normally want to work in the same environment. Rather than specifying commands every time you sign on, the system allows you to specify various environmental attributes in your user profile. Once these are changed in your user profile, they will be set automatically each time you sign on.

Some of these attributes that you could change in your user profile are:

- The current library to the one you will use for this application
- The output queue to the one you are using
- The delivery mode to *BREAK for the user's message queue

If you choose to make these changes to your user profile, follow these steps by using the Change Profile (CHGPRF) command to make the necessary changes.

1. Type CHGPRF on the command line and press F4 to get the prompt for the command.

```
                          Change Profile (CHGPRF)

Type choices, press Enter.

Current library . . . . . > USERXX        Name, *SAME, *CRTDFT
Initial program to call . . . .  *SAME    Name, *SAME, *NONE
  Library . . . . . .                     Name, *LIBL, *CURLIB
Initial menu . . . . . .  *SAME           Name, *SAME, *SIGNOFF
  Library . . . . . .                     Name, *LIBL, *CURLIB
Text 'description' . . . . . .  *SAME

                                                                    Bottom
F3=Exit   F4=Prompt   F5=Refresh   F10=Additional parameters   F12=Cancel
F13=How to use this display        F24=More keys
```

2. Change the *Current library* prompt to the new library you created, and press F10 (Additional parameters).

```
                        Change Profile (CHGPRF)

Type choices, press Enter.

Current library . . . . . > USERXX        Name, *SAME, *CRTDFT
Initial program to call . . . .  *SAME    Name, *SAME, *NONE
  Library . . . . . .                     Name, *LIBL, *CURLIB
Initial menu . . . . . .  *SAME           Name, *SAME, *SIGNOFF
  Library . . . . . .                     Name, *LIBL, *CURLIB
Text 'description' . . . . . .  *SAME


                      Additional Parameters

Job description . . . . . .  *SAME        Name, *SAME
  Library . . . . . .                     Name, *LIBL, *CURLIB
Document password . . . . . .  *SAME      Name, *SAME, *NONE
Message queue . . . . . .  *USRPRF        Name, *SAME, *USRPRF
  Library . . . . . .                     Name, *LIBL, *CURLIB
Delivery . . . . . .  *BREAK              *SAME, *NOTIFY, *BREAK...
                                                                More...

F3=Exit   F4=Prompt   F5=Refresh   F12=Cancel   F13=How to use this display
F24=More keys
```

3. Change the *Delivery* prompt to *BREAK, then page down to change the output queue.

```
                        Change Profile (CHGPRF)

Type choices, press Enter.

Severity code filter . . . . .  *SAME     0-99, *SAME
Print device . . . . . .  *SAME           Name, *SAME, *SYSVAL
Output queue . . . . . .  USERXX          Name, *SAME, *DEV
  Library . . . . . .                     Name, *LIBL, *CURLIB
Attention program . . . . . .  *SAME      Name, *SAME, *NONE
  Library . . . . . .                     Name, *LIBL, *CURLIB
User options . . . . . .  *SAME           *SAME, *NONE, *CLKWD...
              + for more values

                                                                Bottom
F3=Exit   F4=Prompt   F5=Refresh   F12=Cancel   F13=How to use this display
F24=More keys
```

4. Change the *Output queue* prompt to the new output queue you created and press the Enter key.

The command is run and your attributes should be changed.

Step 6. Changing Your Job Information

The changes you made in your user profile will not affect your current job. When a job starts, the attributes from the user profile are copied into your current job.

To get your job to recognize the user profile changes you made, sign off the system (option 90 on the Main Menu) and sign back on. You should do this now.

Note: In place of signing off and signing on again, you could use the Change Job (CHGJOB) or Change Message Queue (CHGMSGQ) command.

Displaying Job Attributes and Working with Spooled Files

The remaining tasks described in this chapter verify that the objects you created and the changes you made actually took place and show you how to display, print, and clear spooled files from your output queue.

Step 7. Displaying Your Library List

After you have signed back on, you can determine if your profile was really changed by displaying the attributes of your job. Follow these steps:

1. First, display your library list to check your current library by typing the Display Library List command as follows:

   ```
   DSPLIBL
   ```

   The Display Library List display is shown.

```
                          Display Library List
                                                      System:   RCH38342

Type options, press Enter.
  5=Display objects in library

Opt  Library     Type      Text
     QSYS        SYS       AS/400 System Library
     QHLPSYS     SYS       AS/400 System Library
     QUSRSYS     SYS       AS/400 System Library
     USERXX      CUR       USERXX personal library
     QTEMP       USR
     QGPL        USR       AS/400 General Purpose Library
     QRPG        USR       AS/400 System Library









                                                                      Bottom
F3=Exit    F12=Cancel    F17=Top    F18=Bottom
                                     (C) COPYRIGHT IBM CORP. 1980, 1989.
```

2. Look at the current (CUR) library to make sure it has been changed.
3. Press F3 (Exit).

Step 8. Displaying Your Message Delivery Mode

1. Display your message delivery mode for your user profile to check that it is set to *BREAK mode by typing the Display Message command as follows:

   ```
   DSPMSG USERXX
   ```

   The Display Messages display is shown.

```
                              Display Messages
                                                    System:   RCH38342
Queue . . . . . :   USERXX                  Program . . . . :   *DSPMSG
  Library . . . :     QUSRSYS                 Library . . . :
Severity  . . . :   00                      Delivery  . . . :   *BREAK

Press Enter to continue.

                                                                      Bottom
F3=Exit          F10=Display all       F11=Remove a message
F12=Cancel       F13=Remove all        F16=Remove all except unanswered
```

2. Look at the delivery mode to make sure it has been changed to *BREAK.
3. Press F3 (Exit).

Step 9. Checking Your Output Queue

To display the output queue attribute for your jobs, use the Work with Jobs command as follows:

1. Type WRKJOB on the command entry line and press the Enter key.

```
                              Work with Job
                                                          System:   XXXXXXXX
Job:   DSP62          User:   USERXX         Number:   003736

Select one of the following:

     1. Display job status attributes
     2. Work with job definition attributes
     3. Work with job run attributes, if active
     4. Work with spooled files

    10. Display job log, if active or on job queue
    11. Display program stack, if active
    12. Work with locks, if active
    13. Display library list, if active
    14. Display open files, if active
    15. Display file overrides, if active
    16. Display commitment control status, if active
                                                                        More...
Selection or command
===> 2

F3=Exit    F4=Prompt    F9=Retrieve    F12=Cancel
```

2. Select option 2 (Work with job definition attributes) on the Work with Job display.

```
                    Work with Job Definition Attributes
                                                          System:   RCH38342
Job:   DSP020604      User:   USERXX         Number:   000263

Job description . . . . . :   QDFTJOBD
  Library . . . . . :     QGPL
Job queue . . . . . :
  Library . . . . . :
Job priority (on job queue) . . . . . :
Output priority (on output queue) . . . . . :   5
End severity . . . . . :   30
Message logging:
  Level . . . . . :   4
  Severity . . . . . :   0
  Text . . . . . :   *NOLIST
Log: CL program commands . . . . . :   *NO
Printer device . . . . . :   IDPRT
Default output queue . . . . . :   USERXX
  Library . . . . . :     USERXX
                                                                        More...
Press Enter to continue.

F3=Exit    F5=Refresh    F9=Change job    F12=Cancel    F16=Job menu
```

3. Look at the *Default output queue* and *Library* prompts to make sure they have been changed.

   Note: Most of the attributes displayed were set by the security officer when your user profile was created. Because of this, some of the values shown here may be different.

4. Press F3 (Exit).

Step 10. Sending Jobs to an Output Queue

To provide an example of spooled output, direct the information for your job to the output queue for printing by typing the following Work with Jobs command:

```
WRKJOB OUTPUT(*PRINT)
```

This will cause the information you saw displayed to be sent to a spooled file. This spooled file can then be printed or just displayed.

Step 11. Displaying the Spooled File

Now, display the file you sent to the output queue by following these steps:

1. Type the following Work with Output Queue command on the command entry line using the name of your output queue:

   ```
   WRKOUTQ OUTQ(USERXX)
   ```

2. The Work with Output Queue display is shown.

```
                          Work with Output Queue

Queue:   USERXX         Library:   USERXX         Status:   RLS

Type options, press Enter.
  2=Change   3=Hold   4=Delete   5=Display   6=Release   8=Attributes

Opt  File        User        User Data   Sts   Pages  Copies  Form Type   Pty
 5   QPDSPJOB    USERXX                  RDY       5       1  *STD         5

                                                                      Bottom
Parameters for option 2 or command
===>
F3=Exit   F11=View 2   F12=Cancel   F22=Printers   F24=More keys
```

3. Type a 5 in the *Opt* column next to the QPDSPJOB file to display it. You should see the same information that was previously displayed.
4. Press F3 twice to exit.

Step 12. Printing Output

To print output, a printer writer program is used to send the spooled file from the output queue to a printer. You can begin printing by doing either of the following:

- Starting a writer to your output queue (requires an available printing device)
- Changing the spooled file so that it will be on an output queue that has a writer already started

In a programming environment, it is normal not to print all of the output. Therefore, it would be typical to change the spooled file to a different output queue when printing is desired.

IBM supplies output queues such as QPRINT. There is also an output queue created for each printer on the system. Ask your system operator for the name of an output queue where you can print from. You may need to know the library name also.

Assuming a printer writer is running, print your output following these steps:

1. Type the following command using the output queue you created:

   ```
   WRKOUTQ OUTQ(USERXX)
   ```

```
                          Work with Output Queue

Queue:   USERXX          Library:   USERXX          Status:   RLS

Type options, press Enter.
  2=Change   3=Hold   4=Delete   5=Display   6=Release   8=Attributes

Opt  File       User       User Data   Sts   Pages  Copies  Form Type   Pty
 2   QPDSPJOB   USERXX                 RDY       5       1  *STD         5

                                                                      Bottom
Parameters for option 2 or command
===>
F3=Exit   F11=View 2   F12=Cancel   F22=Printers   F24=More keys
```

2. Type a 2 in the *Opt* column next to the QPDSPJOB file to change the spooled file attributes and press the Enter key.

```
                  Change Spooled File Attributes (CHGSPLFA)

Type choices, press Enter.

Spooled file . . . . . > QPDSPJOB      Name, *SELECT
Job name . . . . . > DSP020604    Name, *
  User . . . . . >   USERXX      Name
  Number . . . . . >   000263      000000-999999
Spooled file number . . . . . > 1          1-9999, *ONLY, *LAST
Print device . . . . . *OUTQ       Name, *SAME, *OUTQ
Print sequence . . . . . *SAME       *SAME, *NEXT
Output queue . . . . . QPRINT      Name, *SAME, *DEV
  Library . . . . . QGPL       Name, *LIBL, *CURLIB
Form type . . . . . *STD       Form type, *SAME, *STD
Copies . . . . . 1          1-255, *SAME
Save file . . . . . *NO          *SAME, *NO, *YES

                         Additional Parameters

File separators . . . . . 0          0-9, *SAME
                                                                    More...
F3=Exit   F4=Prompt   F5=Refresh   F12=Cancel   F13=How to use this display
F24=More keys
```

3. Change the *Output queue* prompt to the name of the queue that is assigned to your printer writer and press the Enter key. For example, if the output queue is QPRINT in the QGPL library, you would type the fields as shown above.

   The parameter line at the bottom of the Work with Output Queue display could have also been used to change to a different output queue. For example, you could have specified the 2 option next to each spooled file to be changed and entered OUTQ (QGPL/QPRINT) as follows:

```
                          Work with Output Queue

Queue:   USERXX        Library:   USERXX        Status:   RLS

Type options, press Enter.
  2=Change   3=Hold   4=Delete   5=Display   6=Release   8=Attributes

Opt  File       User       User Data   Sts   Pages  Copies  Form Type   Pty
 2   QPDSPJOB   USERXX                 RDY       5       1  *STD         5

                                                                    Bottom
Parameters for option 2 or command
===> OUTQ (QGPL/QPRINT)
F3=Exit   F11=View 2   F12=Cancel   F22=Printers   F24=More keys
```

4. Press F3 (Exit).

Step 13. Clearing the Output Queue

Once you have displayed your spooled files, you should determine what spooled files you do not need to keep and delete them.

The number of spooled files in your output queue should be limited. When jobs have completed, spooled files and internal job control information are kept until the spooled files are printed or deleted. The number of jobs on the system and the number of spooled files known to the system increase the amount of time needed to do an IPL and internal searches, and increase the amount of temporary storage required.

You can delete all files from your output queue by using the Clear Output (CLROUTQ) command as follows:

```
CLROUTQ OUTQ(USERXX)
```

Or, you can delete individual files in the output queue by using option 4 (delete) on the Work with Output Queue display for only the files you want to delete.

Both of these functions are valuable for you. Normally programmers do not print all of their output. You will want to keep the number of spooled files in your output queue to a minimum. You should always delete the ones you do not want.

Overview of Source Files

A source file contains source specifications that the system uses to create an object. IBM-supplied source files, such as QCLSRC and QDDSSRC, are designed to contain source statements for items such as control language (CL) programs and database files (data description specifications (DDS)).

The IBM-supplied source files could be used to contain all source; however, user-created source files are generally preferred because the source can then be controlled by application area. There are generally two approaches to source files:

- Use the same names as the IBM-supplied versions and keep the source files in the appropriate library for a given application. For example, all RPG source for the mailing application would be in the QRPGSRC file in your library.
- Use a single source file per application.

A single source file is used in this manual.

Step 14. Creating the MLGSRC Single Source File

Using the Create Source Physical File (CRTSRCPF) command to create a single source file without specifying a library will automatically put the file in the current library set up for the job. Create a single source file named MLGSRC by typing the following command on the command entry line:

```
CRTSRCPF FILE(MLGSRC) TEXT('Source file for the Mailing List Application')
```

You should receive the following message:

File MLGSRC created in library USERXX

Chapter 3. Using PDM and SEU

Now that the environment is set up, you should be familiar with the tools used to manage the objects used in the application. The programming development manager (PDM) and source entry utility (SEU) are two AS/400 Application Development Tools (ADT) used in this manual. PDM is used to work with libraries, objects, and members. SEU is used to create and change source members. Other tools also exist, but will not be specifically used in this manual.

This chapter gives you an overview of the programming development manager (PDM) and the source entry utility (SEU).

Overview of PDM

Throughout this manual, you will be using the programming development manager (PDM). PDM provides an easy interface to work with lists of libraries, objects, and members. You can perform a number of different operations such as edit, compile, copy, and rename without having to know the commands. Productivity is increased by being able to see a list of items and to work with them at the same time.

To start PDM, type STRPDM on the command entry line and press the Enter key. The AS/400 Programming Development Manager (PDM) display is shown.

```
                AS/400 Programming Development Manager (PDM)

Select one of the following:

     1. Work with libraries
     2. Work with objects
     3. Work with members

     9. Work with user-defined options









Selection or command
===>

F3=Exit       F4=Prompt       F9=Retrieve       F10=Command entry
F12=Cancel    F18=Change defaults
                                        (C) COPYRIGHT IBM CORP. 1981, 1989.
```

This display gives you an easy way to work with libraries, objects, and members. From this display, you select which type of list you want to work with and are presented with a list of the objects.

For example, if you select option 3 (Work with members), you would see the following display.

```
UIM-PART
                        Specify Members to Work With

 Type choices, press Enter.

   File . . . . . .  SOURCE        Name

     Library . . . . . .    USERXX      *LIBL, *CURLIB, name

   Member:
     Name . . . . . .   *ALL         *ALL, name, *generic*
     Type . . . . . .   *ALL         *ALL, type, *generic*, *BLANK









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

On the Specify Members to Work With display, you would type the name of the physical file or source physical file and the library for the member you want to work with. You would be presented with the Work with Members Using PDM display that contains a list of the existing members (if there are any).

```
                          Work with Members Using PDM

 File . . . . . .   SOURCE
   Library . . . .  USERXX                Position to . . . . .

 Type options, press Enter.
   2=Edit         3=Copy       4=Delete       5=Display      6=Print
   7=Rename       8=Display description       9=Save         13=Change text . . .

 Opt  Member      Type         Text

      SAMP1       *FILE        Sample source physical file
      SAMP2       *FILE        Sample logical file
      SAMP3       *PGM         Sample program




 Parameters or command
 ===>
 F3=Exit          F4=Prompt             F5=Refresh           F6=Create
 F9=Retrieve      F10=Command entry     F23=More options     F24=More keys
```

From the Work with Members Using PDM display, you can see that you can edit, delete, copy, create, and so on. One major advantage in using PDM is that you do not need to know any CL commands when using the options available on the displays.

Overview of SEU

You can enter and edit source and create source members from a work station using the *source entry utility* (SEU). SEU operates on *members* of a source file.

A member is an identifiable group of records in a database file. Each member conforms to the characteristics of the file and has its own access path.

When you use SEU, source records are entered into a member of a source physical file. In this application, a file named MLGSRC in your library is used.

A source file can have multiple members, with each member containing a separate set of source statements, as shown in Figure 3-1. In this figure, each member of file MLGSRC contains the source statements for a different file or program.

You create programs and file objects from source members. The programs and objects are normally assigned the same name as the member (you can specify a different name when you create the objects).

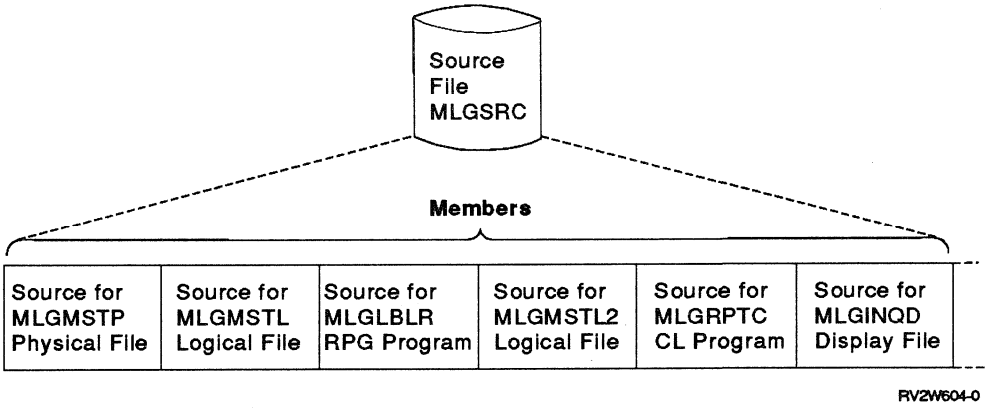


Figure 3-1. Example of a Source File with Multiple Members

There are several advantages to using SEU to create or update members of a source file:

- When a source statement is entered, its syntax can be automatically checked for validity. Coding and keying errors (such as ZADD entered for Z-ADD) are found; however, relational errors (such as a GOTO without a corresponding TAG) are not detected. Relational errors are found when the source is compiled.
- While a source statement is being entered, the formats supplied by IBM for source types, such as PF (physical file) and RPG, can be used to enter data only in fields that apply to a statement. This method of entering data can prevent both positional and field-content keying errors.
- Source entered through SEU is online and can be easily controlled by using online data storage and maintenance. For example, a source file is saved the same way as other types of files and the date of last change is automatically kept as part of each record.
- The following functions are available to help you create and update source:
  - Adding or deleting source statements
  - Changing existing source statements

- Moving or copying source statements within a member
- Copying source statements from one source file member to another
- Searching (scanning) for a specific character string
- Selecting a particular source member from a list of members in a file

SEU can be called by using the Start SEU (STRSEU) command through the PDM Work with Members Using PDM display or through the programmer menu.

Throughout this manual, SEU is usually called from the Work with Members Using PDM display. For example, if you press F6 (Create) on the following display:

```
                        Work with Members Using PDM

File . . . . . .   MLGSRC
  Library . . . .  USERXX                Position to  . . . . .

Type options, press Enter.
  2=Edit         3=Copy       4=Delete       5=Display      6=Print
  7=Rename       8=Display description       9=Save         13=Change text . . .

Opt  Member      Type        Text

  (No members in file)




Parameters or command
===>
F3=Exit          F4=Prompt             F5=Refresh            F6=Create
F9=Retrieve      F10=Command entry     F23=More options      F24=More keys
```

The Start Source Entry Utility (STRSEU) display is shown.

```
                    Start Source Entry Utility (STRSEU)

Type choices, press Enter.

Source file  . . . . . > MLGSRC        Name, *PRV
  Library  . . . . . . > USERXX        Name, *LIBL, *CURLIB, *PRV
Source member  . . . . .  TEST         Name, *PRV, *SELECT
Source type  . . . . . .  PF           Name, *SAME, BAS, BASP, C...
Text 'description' . . . .  SEU Test Member




                                                                  Bottom
F3=Exit   F4=Prompt   F5=Refresh   F12=Cancel   F13=How to use this display
F24=More keys
```

On this display, you would type the name of the source member (TEST) and its type (PF) and description (SEU Test Member) and press the Enter key to get to the Edit display.

```
Columns . . . .:   1  71              Edit                      USERXX/MLGSRC
Find  . . .                                                             TEST
FMT PF .....A.........T.Name++++++RLen++TDpB......Functions+++++++++++++++++
       *************** Beginning of data ************************************
'''''''
'''''''
'''''''
'''''''
'''''''
'''''''
'''''''
'''''''
'''''''
'''''''
'''''''
'''''''
'''''''
'''''''
'''''''
       ****************** End of data ***************************************
 F3=Exit                   F4=Prompt                 F5=Refresh
 F10=Top                   F11=Bottom                F24=More keys
Member TEST added to file USERXX/MLGSRC.                                  +
```

The Edit display is where you would enter your source coding. SEU has many functions available to help you. If you press F24 (More keys), the following keys are displayed.

```
 F13=Change defaults       F14=Find/Change Services  F15=Browse/Copy Services
 F16=Perform find          F17=Perform change        F24=More keys

Member TEST added to file USERXX/MLGSRC.                                  +
```

These keys provide many valuable functions to help you enter your source coding. For more information about these keys, you can place your cursor on either of the function key lines and press the Help key.

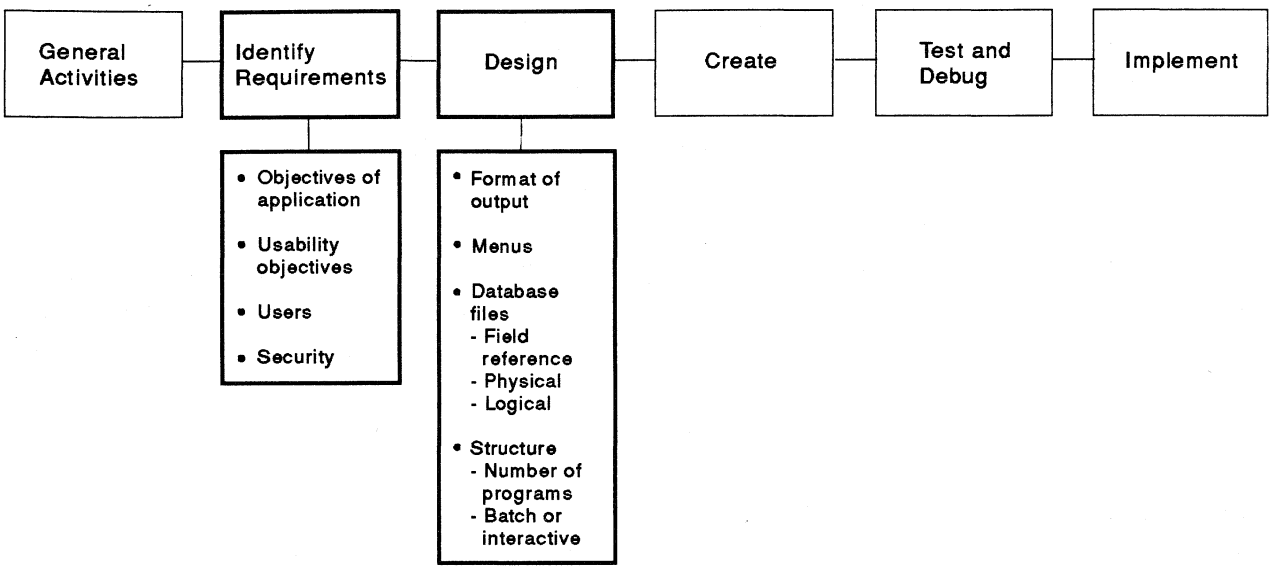
If you press F3 to exit from the SEU Edit display, the Exit display is shown.

```
                                  Exit
Type choices, press Enter.

  Change/create member . . . . . .  N           Y=Yes, N=No
    Member . . . . . . . . . . . .  TEST        Name
    File . . . . . . . . . . . . .  MLGSRC      Name
      Library . . . . . . . . . .  USERXX       Name
    Text . . . . . . . . . . . . .  SEU Test Member

    Resequence member . . . . . .  Y           Y=Yes, N=No
      Start . . . . . . . . . . .  0001.00     0000.01 - 9999.99
      Increment . . . . . . . . .  01.00       00.01 - 99.99

  Print member . . . . . . . . . .  N           Y=Yes, N=No

  Return to editing . . . . . . .  N           Y=Yes, N=No

  Go to member list . . . . . . .  N           Y=Yes, N=No




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

On the Exit display, you can choose to change or create the member, print the member, or return to editing. Since there were no changes or additions made to the member here, the default is not to create the member. If changes or additions had been made to the member, the default would be to create the member.

Knowing how to use SEU is important in going through the exercises contained throughout this manual. You should at least know how to create a new member, and enter, correct, and delete a statement. If you are unfamiliar with using SEU, refer to Appendix C, "Example of Using SEU Functions."

Chapter 4. Mailing List Application Requirements and Design



The mailing list application described in this manual is not *real* from several viewpoints. Rather, it is defined in such a way that most readers should be able to relate to what the application must do, that is, to create and maintain a mailing list file (called a master file), to print mailing labels from the file, and to provide analysis of the file. The mailing list application is intended to be simple and is used only to describe the typical application development steps. However, you will find that the application must address many of the same considerations that exist in *real* applications.

This chapter describes the overall requirements and design of the mailing list application. It then describes the characteristics of the application including the structure and the naming conventions.

Mailing List Application Requirements

Overall Objectives and Anticipated Transactions

The mailing list application will provide a file of names and addresses for the accounts handled by a company. The file must be maintained. Various mailing labels, reports, and queries will be produced.

The following volumes and approximate transaction load have been considered:

- There will be approximately 20,000 accounts. Future growth may double the number of accounts.
- Once the database is created, it is assumed there will be approximately 600 changes a day (additions, changes, or deletions), plus another 100 inquiries.
- Mailing labels will be produced for all accounts every 3 months. Selected mailings will occur on demand. Assume 2 per week.
- Analysis reports will occur on demand. Assume 2 per week.

Basic Input and Output Fields

The mailing label output will have a format such as the following:

```
Susan Smith
121 W 15th St
Minneapolis MN
55601
```

Two-character state abbreviations will be used. One line of address will be sufficient. Five-character zip codes will be required. Twenty characters will be sufficient for each of the name, address, and city fields. Mailing labels will always be printed in zip code order.

Each account will be assigned a unique account number and a 1-character code for *type of account* to assist in selection data (such as 2 = Government account).

Usability Objectives

To make the application more usable, a solution must be provided to allow the account number to be determined if only the name and address is known. It is acceptable to provide a *search value* in each record and allow the user to enter a *search field* and receive all of the matching values. For example, if the name of the account is *Joseph Jones*, the search value could be JONES.

User and Security Considerations

The following requirements exist for the users of the application and the security needed:

- Multiple users must be able to access and maintain the database at the same time.
- Not all users of the system should be allowed to change records in the mailing list database. To provide for this objective, your system must be at the proper security level. This is discussed later. Any user of the system should be allowed to inquire (read only) into the database.

Overview of AS/400 Database

The AS/400 system database is different from traditional system databases because of its innovative design which integrates the database with the operating system program and integrates support for several different interfaces into a single database manager.

As the amount of data that you keep on your system grows and the interrelationships between various objects become more complex, an organized approach to handling this data is essential.

The AS/400 *database* is an organized collection of all data files stored in the system. The AS/400 system allows you to store data in one physical structure, but allows multiple users to access the data in various logical formats, organizations, and sequences.

A *database file* is an organized collection of related records in the database. There are two types of database files on the AS/400 system as shown in Figure 4-1:

- A *physical file* is a database file that contains data records. All the records have the same format; that is, they are fixed-length records and they contain the same fields in the same order.
- A *logical file* is a database file through which data that is stored in one or more physical files can be accessed by means of record formats and/or access paths that may be different from the physical representation of the data in the database.

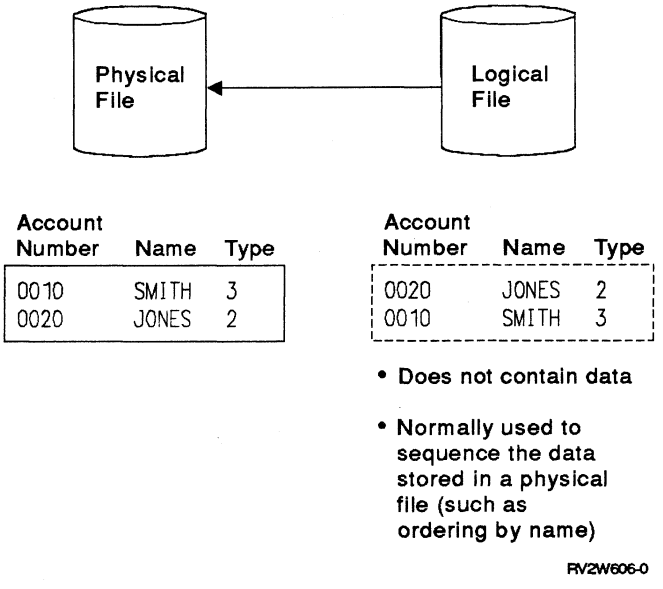


Figure 4-1. Overview of Physical and Logical Files

A file is normally created on the AS/400 system by a CL command before a program can use the file. An RPG/400 program cannot create a file, but can add records to a file that was created previously. When a physical file is created, no data records exist in it. A program, such as an RPG/400 program, a command such as the Copy File (CPYF) command, or DFU must be used to add records to the file.

Access Paths: Data in both types of database files can be processed by a program. The file definition includes an *access path*, which is the means by which the system provides a logical sequence to the data records in a database file so that they can be processed by a program. There are two types of access paths (keyed sequence and arrival sequence), but only the keyed sequence type is discussed in this manual. A *keyed sequence access path* is based on the content of key fields within a record. Using key fields, records can be logically sequenced in a file by specifying such things as:

- The fields in the record that are key fields
- Ascending or descending order for the key field(s)
- The order of the key fields when multiple fields compose the key

Selection criteria can also be used to form a subset of the records in the logical file. For example, you might only be interested in the customer records for an industry type (such as manufacturing).

For more information on keyed sequence access paths, refer to the *Database Guide*.

Externally Described Data Files: The physical database file and the logical database files used in this manual are *externally described*. The records of an externally described data file are described to the system through the use of data description specifications (DDS) when the file is created. The DDS defines the individual fields of the file and their characteristics.

A file can be defined without specifying individual fields. This is called a program-described file. This technique will not be used in this manual. External field descriptions are used in this application because of the following considerations:

- Any fields used as keys or in selection criteria must be defined.
- Field definitions are not coded in every program because the RPG/400 compiler can automatically copy the definition. This helps to keep field names and their attributes consistent and reduces the coding time for input and output fields used in a program.

Database Design

When the requirements of the application have been agreed to, you can proceed with the detail design. The major considerations are to define the database files needed and to define what major interfaces will be provided.

The basic design of the mailing list database is:

- There will be one record per account number with the mailing list information. The account number will be used as the access path to the file.
- A search field will be included in each record and there will be a logical file for the search field.
- There will be other logical files for the label printing programs and analysis reports.

Design of the Basic Database Record

You can now begin the design of the basic database record which will be in the master file.

The account number field must be designed to hold the current 20,000 records with the ability to double. A 5-digit account number is decided on. This will allow for up to 99,999 accounts.

A typical solution for assigning the account numbers (if they do not already exist) would be to arrange the file in *search* order and then number the accounts with gaps in between the numbers. This will allow new accounts to be added and still keep a reasonably consistent sequence.

Another solution is to number the accounts sequentially beginning with 00001. A new account would receive the next sequential number or receive a number made available by a deletion.

Characteristics of the Master File

Each mailing list record in the master file has the following format:

Mailing List Record				
Field Description	**Field Name**	**Length (Characters)**	**Decimal Positions**	**Field Type**
Account number	MLACCT	5	0	Numeric
Type of account	MLTYPE	1		Character
Name	MLNAME	20		Character
Search field for name search	MLSRCH	10		Character
Address	MLADDR	20		Character
City	MLCITY	20		Character
State	MLSTAT	2		Character
Zip Code	MLZIP	5	0	Numeric
Note: The field names used are 6 characters or less. This meets an RPG requirement.				

The account number (MLACCT) is unique for each record. The account type (MLTYPE) can be any of the following values:

MLTYPE Field	
Value	**Meaning**
1	Business
2	Government
3	Organization
4	School
5	Private
9	Other

A typical mailing list record might look like this:

MLACCT	MLTYPE	MLNAME	MLSRCH	MLADDR	MLCITY	MLSTAT	MLZIP
10522	5	JJ Johnson	JOHNSON	489 Whitney St	Chicago	IL	54857

You maintain the master file by adding, deleting, and changing the mailing list records. When printing a mailing list from the master file, you can use various selection criteria to select particular types of records and to arrange the records in various sequences. For example, a mailing list printout might include:

- All government accounts in New Jersey. The list would be printed in numeric sequence by account number within zip code areas.
- All school and private accounts in a specific zip code area. The list would be printed in alphabetic sequence by name.
- All records in account number sequence within zip code areas.

Although this application is simple, its requirements are typical of many data processing applications:

- It has a master file.
- Transactions are applied to the master file. These can change or delete an existing record or add a new record (maintenance of the file).
- An inquiry capability is needed to display a record.
- The data is printed in various sequences.
- There can be multiple work station users.
- Backup of files and programs must be considered.

Interface Design

The basic design of the interfaces and menus is:

- A separate interactive inquiry program will be provided.
- An interactive maintenance program will be provided.
- A name search function will be provided.
- Label printing and analysis will be submitted to batch.
- A menu will be provided with the interactive functions and an easier method to submit batch work.

Naming Rules and Conventions

General

The AS/400 system requires you to assign names to libraries, programs, and files. In addition to naming rules established for the system, you may want to establish your own naming conventions to help you organize your application. No single set of naming conventions is the best; your naming conventions should be easy for you to use and understand.

Some AS/400 naming rules to remember are:

- Objects of the same type within a specific library must have unique names.
- Members of a specific database file must have unique names; however, a member name can be the same as a program or file name within the same library.
- Fields within a specific record format must have unique names.
- RPG/400 requires that all file, record, and field names be unique within a program.
- RPG allows for 6-character field names, 8-character file names, and 8-character format names. The OS/400 licensed program allows for 10-character names. Because the application is coded in RPG, the RPG restrictions are used. RPG does allow a rename capability to handle longer names, but it is generally desirable to meet the restrictions of your language requirements.

The *CL Reference* and the RPG/400 manuals contain more information on naming rules.

For the Mailing List Example

Standards are normally set when your system is installed and not for a specific application. In this mailing list example, the following naming conventions are used:

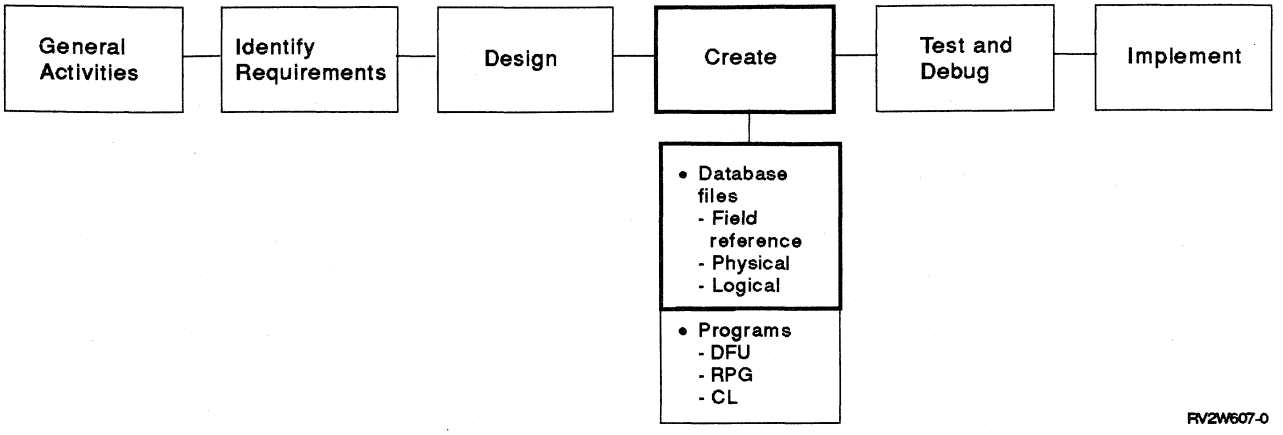
- Object names begin with MLG for *mailing* (such as MLGMSTP).
- Physical file names are 7 characters, the last of which is P (such as MLGMSTP). Record format names for the files are 7 characters, the last of which is R (such as MLGMSTR).
- Logical file names are 7 or 8 characters, the seventh of which is L (such as MLGMSTL); if there is an eighth character, it is a digit (such as MLGMSTL1). A logical file could also have a name such as MLGNAML to assist in understanding the name.
- Other object names are 7 characters:
  - The first 3 characters are MLG.
  - The second 3 characters represent a function, such as MTN for maintenance and NAM for name search.
  - The seventh is C for control language (CL) programs, R for RPG programs, D for display files, U for DFU programs, and Q for Query programs.

The field names for the mailing list master file all begin with ML. Making the first 2 characters unique per database file helps designate which file they come from. In this application, only one physical file exists. If multiple files existed, they could each have their own unique 2-character prefix.

In Conclusion

It is difficult to determine when the application *design* phase ends and the *create* phase begins. There is normally no clear cut boundary. In many cases the same person does both tasks. In the next chapter, you will start creating the files used in the mailing list application.

Chapter 5. Creating Files



Now that the environment has been set up and you have the single source file created, you can begin creating the basic application objects that your application will use. These objects are the field reference, the master physical, and the logical files.

This chapter describes the field reference, physical, and logical file specifications and goes through the following tasks:

- Creating the field reference file (MLGREFP)
- Displaying the output
- Deleting the spooled files
- Creating the master physical file (MLGMSTP)
- Creating the logical file that will be used for label printing in a sequence of zip, state, city, name (MLGMSTL)

Overview of Field Reference Files

After the database files and fields are designed, you normally begin by defining a *field reference file* to be used by the application. The field reference file will contain all of the basic fields and their attributes which will be used by the application. The other files in the application will reference the fields in the field reference file.

Note: There are other solutions to defining a database with the OS/400 licensed program, including Structured Query Language/400 and the interactive data definition utility (IDDU). Neither of these will be used in this manual.

A field reference file is created from data description specifications (DDS). DDS describe data attributes in file descriptions external to the application program that processes the data. To the system, a field reference file is a normal file. You can use either a physical or logical file as a reference file, but normally you create a physical file (it does not need to have a member or any data).

Some of the advantages of using a field reference file are that it:

- Allows you to define the fields only once to the system
- Is a single source for all field information (such as attributes and text description)
- Simplifies the coding needed to achieve consistency and documentation

It is good practice to spend extra time defining the fields in a field reference file. Having a good definition of a field can be very helpful in minimizing changes later on.

The field reference file is not dynamic. If you change the definition of a field, the change is not dynamically duplicated in all of the objects that refer to the field reference file. Every object that would be affected by the change still needs to be re-created. However, no coding changes would normally need to occur in the files that refer to the field reference file.

The field reference file and the system support for externally described data encourage the programmer to use consistent field names in all programs that use the field reference file. This not only assists in debugging, but in maintenance as well.

DDS are defined to be entered in specification sheet form. Figure 5-1 on page 5-3 shows a blank DDS form.

There are two typical approaches to entering DDS into the system:

- You can fill in the form and then transfer the coded entries into a source member using SEU.
- You can enter the specifications directly into a source member using SEU. You may have written down on scratch paper the fields that you want to enter and then fill in the remaining specifications as you go along.

Most programmers do not fill out the DDS form. They just start entering from SEU with some scratch paper to assist. The DDS form is only used logically. However, the positions described on the form must be accurately followed when you enter the source statements. If not, the DDS source for the file will not be created (or will be created incorrectly). SEU assists you in entering specification type statements and will check the syntax of your statements. See Appendix C for an example of how to use the DDS prompts in SEU.

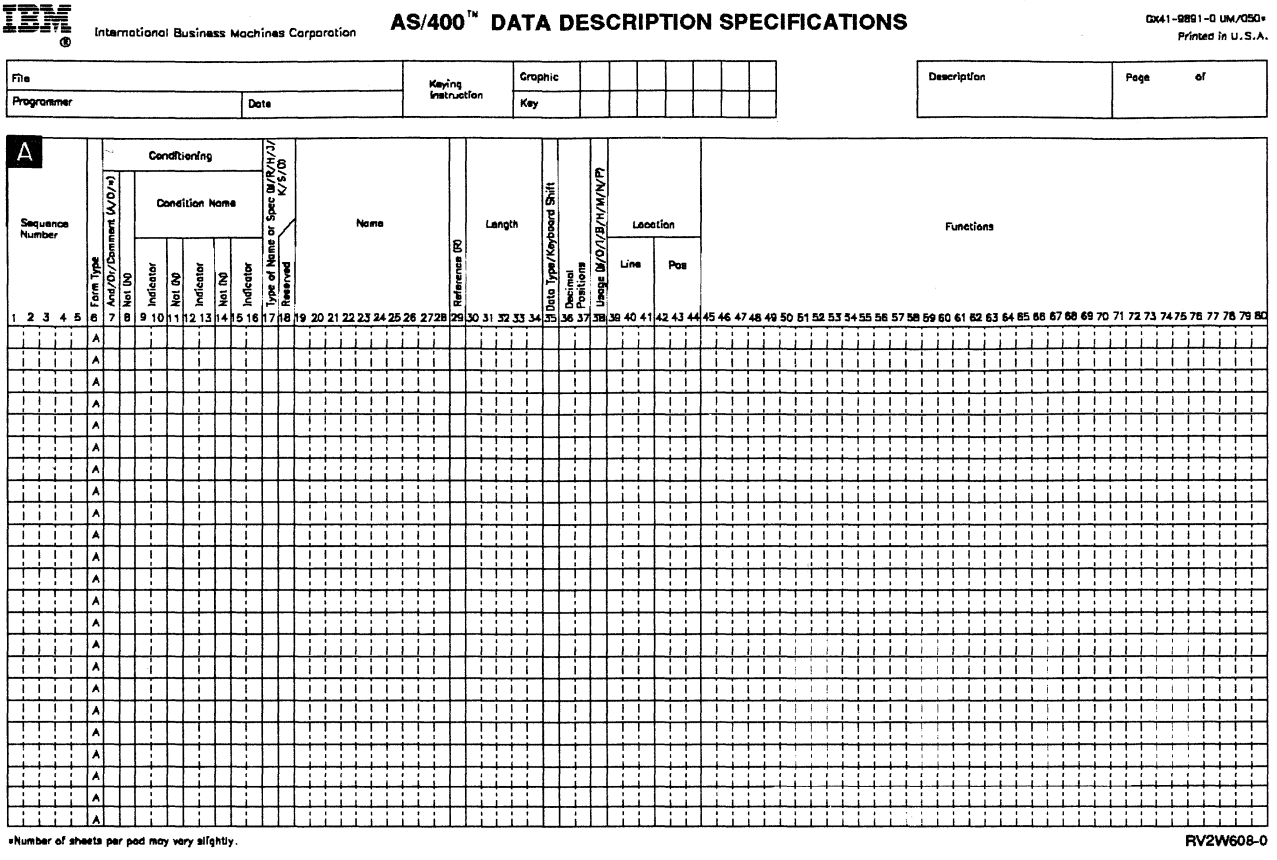


Figure 5-1. Example of a DDS Form

An example of some of the DDS for the MLGREFP field reference file entered on a DDS form is shown in Figure 5-2. The complete source is shown in Figure 5-3 on page 5-4.

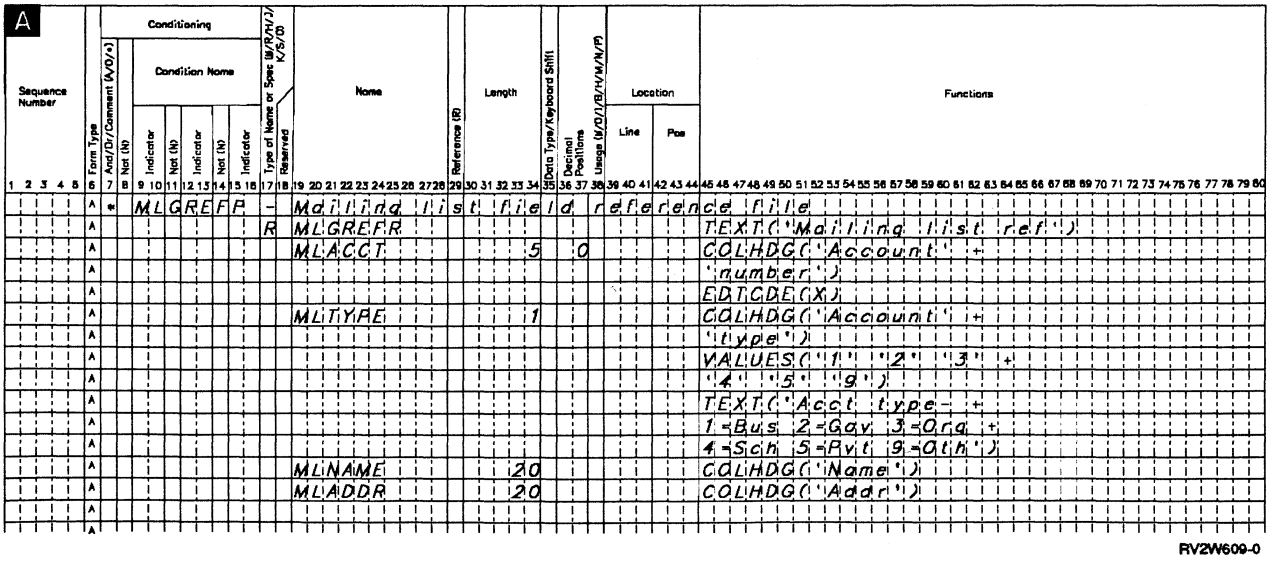


Figure 5-2. DDS for MLGREFP Field Reference File

Rather than showing the specifications in specification sheet form, this manual will show them in source form where the source information has been printed after it was entered.

Description of DDS for Field Reference File (MLGREFP)

Figure 5-3 shows the source for the MLGREFP field reference file. A description of each line follows the figure.

A ruler is shown above the source to assist you in determining the position being described.

```
        ...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7
0001.00     A* MLGREFP - Mailing list field reference file
0002.00     A          R MLGREFR                  TEXT('Mailing list ref')
0003.00     A            MLACCT         5  0      COLHDG('Account' +
0004.00     A                                     'number')
0005.00     A                                     EDTCDE(X)
0006.00     A            MLTYPE         1        COLHDG('Type')
0007.00     A                                     VALUES('1' '2' '3' +
0008.00     A                                     '4' '5' '9')
0009.00     A                                     TEXT('Acct type- +
0010.00     A                                     1=Bus 2=Gov 3=Org +
0011.00     A                                     4=Sch 5= Pvt 9=Oth')
0012.00     A            MLNAME        20        COLHDG('Name')
0013.00     A            MLSRCH        10        COLHDG('Search')
0014.00     A                                     TEXT('Search field for +
0015.00     A                                     name search')
0016.00     A            MLADDR        20        COLHDG('Addr')
0017.00     A            MLCITY        20        COLHDG('City')
0018.00     A            MLSTAT         2        COLHDG('State')
0019.00     A                                     VALUES('AL' 'AK' 'AZ' +
0020.00     A                                     'AR' 'CA' 'CO' 'CT' +
0021.00     A                                     'DE' 'DC' 'FL' 'GA' +
0022.00     A                                     'HI' 'ID' 'IL' 'IN' +
0023.00     A                                     'IA' 'KS' 'KY' 'LA' +
0024.00     A                                     'ME' 'MD' 'MA' 'MI' +
0025.00     A                                     'MN' 'MS' 'MO' 'MT' +
0026.00     A                                     'NE' 'NV' 'NH' 'NJ' +
0027.00     A                                     'NM' 'NY' 'NC' 'ND' +
0028.00     A                                     'OH' 'OK' 'OR' 'PA' +
0029.00     A                                     'RI' 'SC' 'SD' 'TN' +
0030.00     A                                     'TX' 'UT' 'VT' 'VA' +
0031.00     A                                     'WA' 'WV' 'WI' 'WY')
0032.00     A            MLZIP          5  0      COLHDG('ZIP' 'code')
0033.00     A                                     EDTCDE(X)
```

Figure 5-3. DDS Source for MLGREFP Field Reference File

Line 1.00: All DDS specifications have an A in position 6. A comment is indicated by an * in position 7. A comment is used to help describe the source and is not used when an object is created. The comment on line 1.00 describes the name of the member and its purpose. This convention will be followed for all of the source members created.

Line 2.00: The R in position 17 defines a record line. The name of the record MLGREFR is entered in positions 19 through 28. For a field reference file, the record name has no purpose other than to satisfy a DDS requirement.

The left side of the form (positions 7 through 44) has specific positions for specific functions. Position 45 to the end of the specification line is the keyword area. The keyword area is more free form and allows a wide variety of entries.

The keyword entered is TEXT which provides a text description of the record being created. TEXT is optional and provides documentation only. If you display the file description, you could see the record text. Notice that the text entry has single apostrophes surrounding the text. The apostrophes describe a string of data.

Lines 3.00−5.00: The MLACCT field is defined. The field name is given and the 5 in position 34 describes a length of 5. The 0 in position 37 indicates that the field has zero decimal positions. Because there is an entry in the decimal field, the field is defined as a *numeric* type. The default is that if no decimal positions are defined (position 37 is blank), the field will be defined as a *character* type.

Field Types: The two basic field types are character and numeric.

Any characters are valid in a character field.

A numeric field defaults to be in packed decimal form. It is more efficient from both a disk storage space and from a processing unit viewpoint to use odd length packed decimal fields. Because of this, you should define numeric fields with odd field lengths as shown in the example. The system is very restrictive about what is valid in a packed decimal field. Only the digits 0 through 9 are valid plus a valid sign (plus or minus).

The system also restricts various functions involving character versus numeric fields. You can only perform arithmetic calculations on numeric fields and you can only edit (insert a decimal point) in a numeric field. Operations that perform a comparison require the same field type (numeric versus numeric or character versus character). There are instructions that let you change from one field type to another.

COLHDG Keyword: One keyword for the MLACCT field is COLHDG meaning *column heading*. Two values are entered (Account and number). The plus symbol means that the specification of the keyword is continued on the next line. A column heading is used by various system functions to help describe the field when a value is presented. If no COLHDG keyword is entered, the column heading assigned is the field name. Since MLACCT is not meaningful to an end user, a column heading is assigned to this field as well as every field in the field reference file. You can have up to 3 lines for the column heading per field.

TEXT Keyword: A *text description* also exists for each field. If no TEXT keyword is specified, the text description of the field is the column heading values assigned. Multiple column heading values would appear with a blank between them for the text description. The text description assigned to the MLACCT field would be *Account number*. The text description appears in a variety of places in the system such as DFU, Query, SQL, and on an RPG compilation listing that uses the field.

EDTCDE Keyword: The EDTCDE (edit code) keyword is also assigned to MLACCT. EDTCDE describes an editing value for a field. For example, a dollar amount field is normally presented as *1,312.15* although it is normally stored in the database as 131215. The edit code can be used to describe where the

comma and decimal point appear and whether leading zeros should be suppressed.

The keywords for a given field end when another field name appears in positions 19 through 28.

Edit Code X: The X edit code says that the field should not be edited and that leading zeros should not be suppressed. In this application, assume the account number should appear with the leading zero such as 01234. The leading zero should not be suppressed. The edit code function is used by various system functions. RPG also supports the same definition for edit codes.

Lines 6.00−11.00: The MLTYPE field is described as a 1-byte character field (no decimal positions). A column heading is assigned.

VALUES Keyword: The VALUES keyword identifies the valid values that may be entered into this field. The apostrophes surrounding each value are needed because a digit (such as 1) entered by itself is considered to be a numeric value. A DDS error would occur unless the valid values are all character constants if the field defined is a character type.

The VALUES keyword applies when a value for the field is entered from a work station where the display file is described using DDS. DFU does not use the VALUES description.

A text description is provided for the field to help describe what the codes mean. Up to 50 characters of text can be entered for a text description. For this field, the coded text description is a better description than the default text described by use of the column heading.

Lines 12.00−17.00: The name, search value, address, and city fields are described. Each is given specific attributes. The TEXT keyword is used when the column heading default is insufficient.

Lines 18.00−31.00: The MLSTAT field is defined. The VALUES entry defines all of the 2-character state abbreviations.

Lines 32.00−33.00: The zip code is defined as a numeric field. Zip codes are normally displayed with leading zeros so the X edit code is used.

Step 1. Entering the Field Reference File Source (MLGREFP)

The field reference file named MLGREFP contains the record format for the mailing list example. To enter the source for MLGREFP, follow these steps:

1. Start PDM by typing the STRPDM command on the command entry line.
2. Request SEU by selecting option 3 (Work with members) from the PDM main menu.

The Specify Members to Work With display is shown.

```
                    Specify Members to Work With

Type choices, press Enter.

  File . . . . . .  MLGSRC       Name

    Library . . . . . .  USERXX       *LIBL, *CURLIB, name

  Member:
    Name . . . . . .  *ALL         *ALL, name, *generic*
    Type . . . . . .  *ALL         *ALL, type, *generic*, *BLANK

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

3. Fill in the file and library names on the Specify Members to Work With display as shown above and press the Enter key.

   The Work with Members Using PDM display is shown.

```
                        Work with Members Using PDM

File . . . . . .  MLGSRC
  Library . . . .  USERXX                Position to . . . . .

Type options, press Enter.
  2=Edit         3=Copy       4=Delete       5=Display      6=Print
  7=Rename       8=Display description       9=Save         13=Change text ...

Opt  Member      Type        Text

  (No members in file)

Parameters or command
===>
F3=Exit          F4=Prompt            F5=Refresh           F6=Create
F9=Retrieve      F10=Command entry    F23=More options     F24=More keys
```

4. No members are shown because there are no existing members in file MLGSRC (unless you already copied the source from the QUSRTOOL library). Press F6 (Create) to create a member.

The Start Source Entry Utility display is shown.

```
                    Start Source Entry Utility (STRSEU)

Type choices, press Enter.

Source file . . . . . > MLGSRC       Name, *PRV
  Library . . . . . >   USERXX      Name, *LIBL, *CURLIB, *PRV
Source member . . . . .  MLGREFP     Name, *PRV, *SELECT
Source type . . . . . .  PF          Name, *SAME, BAS, BASP, C...
Text 'description' . . .  Mailing list field reference file

                                                                  Bottom
F3=Exit   F4=Prompt   F5=Refresh   F12=Cancel   F13=How to use this display
F24=More keys
```

5. Fill in the fields as shown above and press the Enter key. The source type is PF for physical file.

   The SEU Edit display is shown.

```
Columns . . . .:   1  71              Edit                    USERXX/MLGSRC
Find  . . .                                                       MLGREFP
FMT PF .....A.........T.Name++++++RLen++TDpB......Functions++++++++++++++++++
       ********** Beginning of data ************************************
'''''''
'''''''
'''''''
'''''''
'''''''
'''''''
'''''''
'''''''
'''''''
'''''''
'''''''
'''''''
'''''''
'''''''
'''''''
       ****************** End of data ***************************************

F3=Exit                 F4=Prompt               F5=Refresh
F10=Top                 F11=Bottom              F24=More keys
Member MLGREFP added to file USERXX/MLGSRC.                               +
```

6. The next step is to enter the DDS.

Note: You can use the browse/copy services function to copy the source from the QUSRTOOL library or copy all the source needed in this manual. This will remove the need for you to have to type the specifications and remove keying errors. Refer to Appendix B, "Overview of QUSRTOOL Library" for the name of the file to use and an example of how to do this. If you choose to copy the source from the QUSRTOOL library, you still must create the file using the next step.

SEU provides a prompting method for you to enter physical file DDS. (The statements do not need to be entered knowing the required positions.) You can see what this prompt looks like by positioning the cursor to a statement and pressing F4. Press F12 (Cancel) to remove the prompt.

7. Enter the DDS for MLGREFP as shown in Figure 5-3 on page 5-4. When you are done, press the Enter key.
8. Press F3 (Exit).

```
                                  Exit

Type choices, press Enter.

  Change/create member . . . . . .  Y            Y=Yes, N=No
    Member . . . . . . . . . . . .  MLGREFP      Name
    File . . . . . . . . . . . . .  MLGSRC       Name
      Library . . . . . . . . . .  USERXX       Name
    Text . . . . . . . . . . . . .  Mailing list field reference file

    Resequence member . . . . . .  Y            Y=Yes, N=No
      Start . . . . . . . . . . .  0001.00      0000.01 - 9999.99
      Increment . . . . . . . . .  01.00        00.01 - 99.99

  Print member . . . . . . . . . .  N            Y=Yes, N=No

  Return to editing . . . . . . .  N            Y=Yes, N=No

  Go to member list . . . . . . .  N            Y=Yes, N=No


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

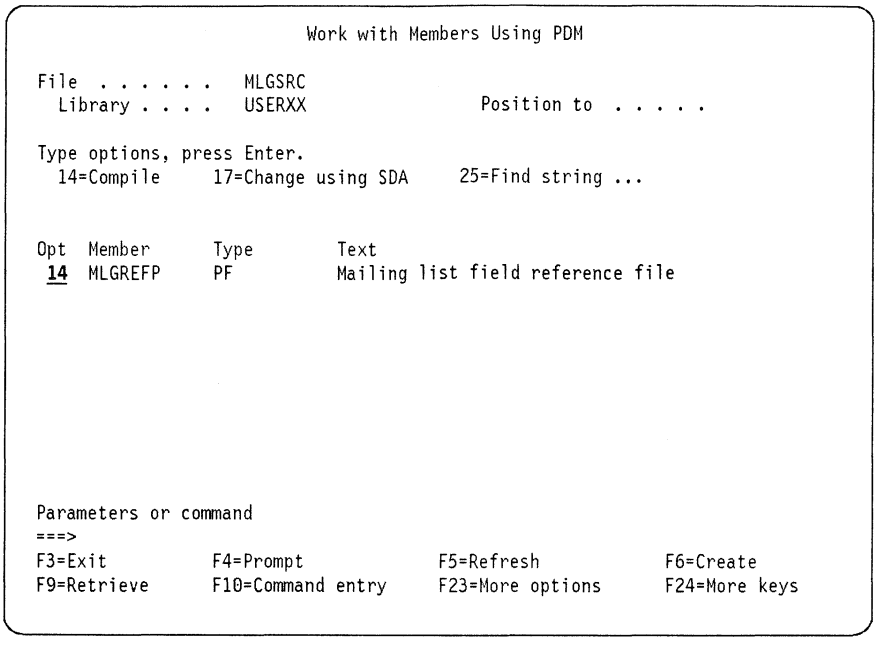
9. On the Exit display, press the Enter key to take the defaults to create the member.

You should now be back to the Work with Members Using PDM display.

Step 2. Creating the Field Reference File (MLGREFP)

The information you have entered so far is only known to the system as source. It is not known as a field reference file. You now need to use a create command which will use the source and create an object (the field reference file) on the system. Rather than directly entering a command, PDM makes it easy to do this by allowing a special option (14) for compile. The compile option will submit the proper create command to batch.

On the Work with Members Using PDM display, type a 14 (Compile) in the *Opt* column next to the file to create the MLGREFP member.



PDM issues the appropriate create command based on the member type. The actual command issued is:

```
CRTPF FILE(MLGREFP) SRCFILE(MLGSRC) MBR(*NONE)
```

The job will be submitted to batch processing and you will receive a completion message when the job is done. The amount of time it will take to run the batch job will vary depending on how much work exists in your system. When a batch job is submitted, it will be placed on a job queue. Various system commands control when the job is scheduled to run and what its priority is.

Displaying Your Output

After you receive a completion message for the job, you can check the file you just created. Use the Work with Output Queue (WRKOUTQ) command and select option 5 to display the spooled file MLGREFP. Spooled files associated with a create (CRT) command will have the same name as the specified file name.

If you want to print the file, go through "Step 12. Printing Output" on page 2-13. Otherwise, you can select option 4 to delete the MLGREFP spooled file.

If you have questions about the field reference file, you can do the following:

- Display the source using SEU. This is normally an adequate solution, but displaying the source does not guarantee that the source agrees with the existing field reference file. For example, you may have changed the source without re-creating the field reference file.
- Display the field reference file. This can be achieved with the Display File Field Description (DSPFFD) command. This will show the object version that is being used by the system. You can try this by entering:

  ```
  DSPFFD FILE(MLGREFP)
  ```

- Keep the printed output. This requires filing a new version every time you make a change.

Overview of Master Physical File

The master file used in the mailing list application is the physical file, MLGMSTP. The file will be keyed on the account number field. MLGMSTP contains the master data for the mailing list application. The logical file, MLGMSTL, provides a different access path to access the master data.

Description of the DDS for Physical File (MLGMSTP)

Figure 5-4 shows the DDS for the MLGMSTP physical file. Each line of the DDS is described following the figure.

```
      ....+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7
0001.00     A* MLGMSTP - Mailing list master physical file
0002.00     A                                     REF(MLGREFP)
0003.00     A                                     UNIQUE
0004.00     A          R MLGMSTR                  TEXT('Mailing list +
0005.00     A                                     master record')
0006.00     A            MLACCT    R
0007.00     A            MLTYPE    R
0008.00     A            MLNAME    R
0009.00     A            MLSRCH    R
0010.00     A            MLADDR    R
0011.00     A            MLCITY    R
0012.00     A            MLSTAT    R
0013.00     A            MLZIP     R
0014.00     A          K MLACCT
```

Figure 5-4. DDS for Master Physical File (MLGMSTP)

Line 1.00: A comment.

Line 2.00: The REF keyword identifies the field reference file (MLGREFP) that was created previously.

Note: The REF keyword is specified before the record description. In DDS, it is possible to define a file with multiple record formats (a physical file can have only one format). Certain keywords apply to the entire file in which case they must be specified before the first record description. This is called a keyword at the *file level*. The REF keyword must be specified at the *file level*.

Line 3.00: The UNIQUE keyword indicates that the values for the key fields defined for this file must be unique. This means that the system will prevent two account numbers from having the same value.

Lines 4.00−5.00: The record MLGMSTR is described. Notice that the name differs from the file name. This is to meet an RPG requirement that any names (files or records) must be unique in the program. A text description is given to the record by the TEXT keyword.

Lines 6.00−13.00: The fields for the mailing list master file are defined. Each field specifies an R in position 29 to indicate that the definition of the field should be extracted from the file referred to by the REF keyword on line 2.00.

Note that the entries for a physical file are very easy to specify because all of the work has been done in the field reference file.

Line 14.00: The key field of the file is described. The K in position 17 indicates that the field specified in positions 19 through 28 is a key field. Each key field (if there is more than one) must have a K entry. The key field is identified as MLACCT for the account number.

File Processing: A file does not have to have a key field, in which case you can process it either in arrival sequence order (the order in which the records were added to the file) or relative record order. You can process an arrival sequence file either sequentially or randomly. Random processing occurs by specifying a relative record number. For example, relative record 100 would be the one-hundredth record added to the file. In this application, keyed processing is used (not arrival sequence processing).

Step 1. Entering the Master Physical File Source (MLGMSTP)

To enter the source for MLGMSTP, do the following:

1. Request SEU from the PDM menu by selecting option 3 (Work with members).

```
                    Specify Members to Work With

Type choices, press Enter.

  File . . . . . .  MLGSRC      Name

    Library . . . . . .  USERXX      *LIBL, *CURLIB, name

  Member:
    Name . . . . . .  *ALL        *ALL, name, *generic*
    Type . . . . . .  *ALL        *ALL, type, *generic*, *BLANK

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

2. Fill in the fields on the display as shown above and press the Enter key.

The Work with Members Using PDM display is shown.

```
                        Work with Members Using PDM

File . . . . . MLGSRC
  Library . . . . USERXX                Position to . . . . .

Type options, press Enter.
  2=Edit         3=Copy       4=Delete       5=Display      6=Print
  7=Rename       8=Display description       9=Save         13=Change text ...

Opt  Member      Type         Text
     MLGREFP     PF           Mailing list field reference file









Parameters or command
===>
F3=Exit          F4=Prompt             F5=Refresh            F6=Create
F9=Retrieve      F10=Command entry     F23=More options      F24=More keys
```

3. Press F6 (Create) to create a new member.

   The Start Source Entry Utility display is shown.

```
STRSEU                    Start Source Entry Utility

Type choices, press Enter.

Source file . . . . . > MLGSRC        Name, *PRV
  Library . . . . . > USERXX          Name, *LIBL, *CURLIB, *PRV
Source member . . . . . . MLGMSTP     Name, *PRV, *SELECT
Source type . . . . . . . PF          Name, *SAME, BAS, BASP, C...
Text 'description' . . . . . . . Mailing list master physical file









                                                                  Bottom
F3=Exit   F4=Prompt   F5=Refresh   F12=Cancel   F13=How to use this display
F24=More keys
```

4. Fill in the fields as shown above and press the Enter key.

The following Edit display appears.

```
 Columns . . . .:   1  71              Edit                      USERXX/MLGSRC
 Find  . . .                                                        MLGMSTP
 FMT PF .....A.........T.Name++++++RLen++TDpB......Functions+++++++++++++++++
        *************** Beginning of data ************************************
'''''''
'''''''
'''''''
'''''''
'''''''
'''''''
'''''''
'''''''
'''''''
'''''''
'''''''
'''''''
'''''''
'''''''
'''''''
        ****************** End of data ***************************************


 F3=Exit                  F4=Prompt                F5=Refresh
 F10=Top                  F11=Bottom               F24=More keys
Member MLGMSTP added to file MLGSRC/USERXX.                                 +
```

Enter the DDS as shown in Figure 5-4 on page 5-11 and press the Enter key. Then, press F3 to exit.

Note: You can use the browse/copy services function to copy the source from the QUSRTOOL library. Refer to Appendix B, "Overview of QUSRTOOL Library" for the name of the file to use and an example of how to do this. If you choose to copy the source from the QUSRTOOL library, you still must create the file using the next step.

```
                                  Exit

Type choices, press Enter.

  Change/create member . . . . . .  Y            Y=Yes, N=No
    Member . . . . . . . . . . . .  MLGMSTP      Name
    File . . . . . . . . . . . . .  MLGSRC       Name
      Library . . . . . . . . . .   USERXX       Name
    Text . . . . . . . . . . . . .  Mailing list master physical file

    Resequence member . . . . . .  Y            Y=Yes, N=No
      Start . . . . . . . . . . .  0001.00      0000.01 - 9999.99
      Increment . . . . . . . . .  01.00        00.01 - 99.99

  Print member . . . . . . . . . .  N            Y=Yes, N=No

  Return to editing . . . . . . .  N            Y=Yes, N=No

  Go to member list . . . . . . .  N            Y=Yes, N=No


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

5. On the Exit display, press the Enter key to take the defaults to create the member.

   You should be back to the Work with Members Using PDM display.

Step 2. Creating the Master Physical File (MLGMSTP)

Type a 14 (Compile) in the *Opt* column to create the MLGMSTP physical file and press the Enter key.

```
                        Work with Members Using PDM

File . . . . . .  MLGSRC
  Library . . . .  USERXX              Position to  . . . . .

Type options, press Enter.
  14=Compile     17=Change using SDA     25=Find string ...


Opt  Member      Type        Text
 14  MLGMSTP     PF          Mailing list master physical file
     MLGREFP     PF          Mailing list field reference file









Parameters or command
===>
F3=Exit          F4=Prompt             F5=Refresh            F6=Create
F9=Retrieve      F10=Command entry     F23=More options      F24=More keys
```

A job will be submitted to batch to create the file. You should receive a completion message when the job is done. When you have received a completion message that the job ended normally, you can then create the master logical file as discussed in the following section.

Overview of Logical File

A master physical file has been created. Now, a logical file can be created over the master physical file. The logical file will create a view of the master data ordered by zip, state, city, then name. This is the sequence required to print the labels. There are other solutions to selecting and sequencing records that will be described later.

The relationship of the logical file to the physical file is shown in Figure 5-5 on page 5-16.

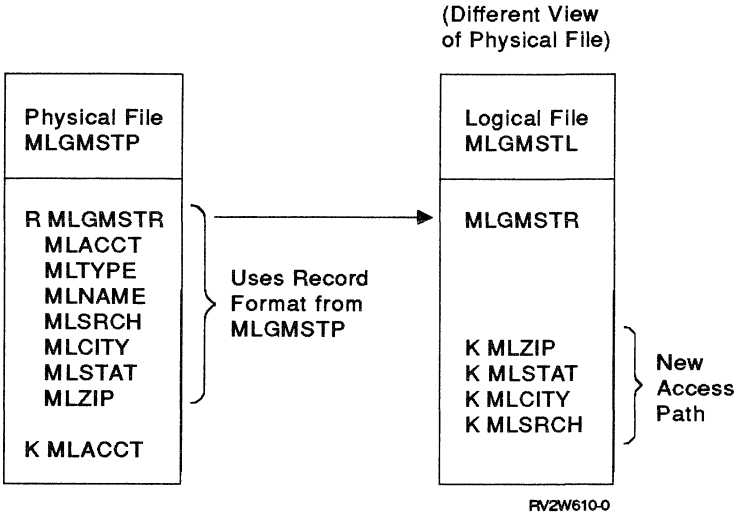


Figure 5-5. Relationship of Master Physical File to Master Logical File

In this example, the record format of the logical file is the same as that used in the physical file. A different format could have been used. For example, you might want a subset of the fields, a different ordering of the fields, or certain conversions to take place.

Description of the DDS for Master Logical File (MLGMSTL)

Figure 5-6 shows the DDS for the MLGMSTL logical file. A description of each line follows the figure.

```
        ....+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7
0001.00      A* MLGMSTL - Mailing list label printing logical file
0002.00      A          R MLGMSTR                  PFILE(MLGMSTP)
0003.00      A          K MLZIP
0004.00      A          K MLSTAT
0005.00      A          K MLCITY
0006.00      A          K MLSRCH
```

Figure 5-6. DDS for Master Physical File (MLGMSTP)

Line 1.00: A comment.

Line 2.00: The record is defined as MLGMSTR (R in position 17). This is the same name as that used in the MLGMSTP file. When there are only key fields described (K in position 17), this tells the system to use the format of the same name as is described in the file named by the PFILE keyword.

A new format will not be created. The same format used to process the physical file will be used in the logical file.

The PFILE (physical file) keyword describes which physical file should be used. A logical file does not have any data. It must process the data through a physical file and the PFILE keyword describes the file that was previously created.

Lines 3.00−6.00: The key fields for the file are described. Each has a K in position 17. The key field specified first is the high-order value. This means that the access path will be built by zip code. Within each zip code, the records are

ordered by state and within each state, by city and then finally, by the search field.

Step 1. Entering the DDS and Creating the Master Logical File (MLGMSTL)

Now, you can create the definition for the MLGMSTL logical file. Follow the same steps you used to create the MLGMSTP physical file earlier in this chapter entering the DDS as shown in Figure 5-6 on page 5-16 or copying the source from the QUSRTOOL library as described in Appendix B, "Overview of QUSRTOOL Library." The member source type must be LF (logical file) for MLGMSTL.

Once you have entered the DDS, create the file by selecting option 14 (Compile) on the Work with Members Using PDM display.

Note: You need to see the completion message that the logical file created successfully before doing the steps in Chapter 7, "Creating the Label Printing Program."

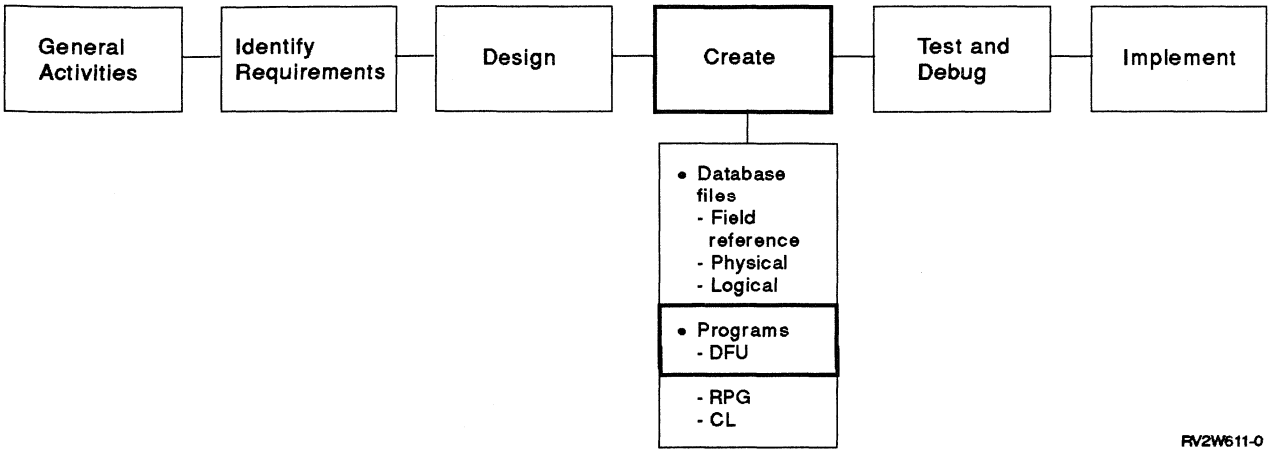
A Review of Files and Their Use

When you create files for an application, they must be created in a definite sequence:

- If a field reference file is to be used, it must be created before any physical files are created that refer to the field reference file.
- Physical files must be created before any logical files that use them are created.
- Externally described data files must be created before the program that uses them is created.

A field reference file will normally grow as an application is developed. If a field reference file is changed, the change does not automatically occur in the files that reference the field reference file. Changes to the field reference file do not require that physical and logical files be re-created unless the change affects the physical and logical files.

Chapter 6. Adding Records to the Database File Using DFU



Now that the basic application objects the mailing list application uses have been created, you can add records to the database file. You can add records to a file by using a program, such as an RPG program, a command such as the Copy File (CPYF) command, or the data file utility (DFU). In this chapter, a DFU program is described, created, and used to enter the mailing list data.

Overview of DFU Program

In the mailing list application, a DFU program named MLGMSTU is created to enter new records into a file. This program is created through DFU by responding to a series of prompts.

The MLGMSTU DFU program uses the externally described data in the master physical file MLGMSTP which you just created.

Step 1. Creating the DFU Program (MLGMSTU)

This section goes through creating the DFU program, then entering the mailing list data. Follow these steps to create a DFU program called MLGMSTU.

1. Start DFU by typing STRDFU on the command entry line and pressing the Enter key.

```
                        AS/400 Data File Utility (DFU)

Select one of the following:

     1. Run a DFU program
     2. Create a DFU program
     3. Change a DFU program
     4. Delete a DFU program
     5. Update data using temporary program









Selection or command
===> 2

F3=Exit    F4=Prompt    F9=Retrieve    F12=Cancel
                                        (C) COPYRIGHT IBM CORP. 1981, 1989.
```

2. Select option 2 (Create a DFU program) on the DFU menu and press the Enter key.

   The Create a DFU Program display appears.

```
                              Create a DFU Program

Type choices, press Enter.

  Program . . . . . .  MLGMSTU      Name, F4 for list
    Library . . . . .  *CURLIB      Name, *CURLIB

  Data file . . . . .  MLGMSTP      Name, F4 for list
    Library . . . . .  USERXX       Name, *LIBL, *CURLIB









F3=Exit      F4=Prompt      F12=Cancel
```

3. Fill in the *Program*, *Data file*, and *Library* prompts as shown above and press the Enter key. The data file specified, MLGMSTP, is the physical file you just created in your own library.

   Note: If you had forgotten the name of the data file you wanted to use, you could have moved your cursor to the *Data file* prompt and pressed F4 to go to the Select File display. A list of data files available to you in the specified library would have appeared. From this list, you could have selected the data file that you wanted to use for the DFU program.

The General Information/Indexed File display appears next because the data file, MLGMSTP, is an indexed file. This display allows you to define the format for your data entry display and to choose whether or not to print an audit report.

```
                    General Information/Indexed File

Type choices, press Enter.

  Job title . . . . . .  MLGMSTU
  Display format  . . . .  2          1=Single
                                      2=Multiple
                                      3=Maximum
  Audit report  . . . . .  Y          Y=Yes, N=No
  S/36 style  . . . . . .  N          Y=Yes, N=No
  Suppress errors . . . .  N          Y=Yes, N=No
  Edit numerics . . . . .  N          Y=Yes, N=No

  Generate keys . . . . .  N          Y=Yes, N=No

F3=Exit      F12=Cancel       F14=Display definition
```

4. Press the Enter key to take the default values.

   Because you took the defaults on the previous display which had a Y (yes) for Audit report, the Select Audit Control display appears.

```
                        Select Audit Control

Type choices, press Enter.

  Print additions . . . .  Y          Y=Yes, N=No
  Print changes . . . . .  Y          Y=Yes, N=No
  Print deletions . . . .  Y          Y=Yes, N=No
  Printer
    Line width  . . . . .  132        60-198
    Column spacing  . . .  1          0-9

F3=Exit      F12=Cancel       F14=Display definition
```

5. Press the Enter key to again take the defaults. This will cause a spooled file to be written describing the changes made when you run the DFU program.

   The Select Record Formats display appears next because the MLGMSTP data file is described through DDS. This display lists the various record formats in your file specification. You can select one or more formats for processing.

```
                          Select Record Formats

File . . . :   MLGMSTP                        Library . . . . :   USERXX

Type options, or press F21 to select all; press Enter.
  1=Select   4=Delete

Opt  Format      Defined  Description
 1   MLGMSTR         N    Mailing list master record

                                                                    Bottom
F3=Exit definition          F5=Refresh          F12=Cancel
F14=Display definition      F21=Select all
```

6. Type a 1 (Select) next to the MLGMSTR format and press the Enter key.

```
                       Select and Sequence Fields

File . . . . . :   MLGMSTP       Library . . . . :   USERXX
Record Format . . . . . :   MLGMSTR

Select fields and their sequence or press F21 to select all; press Enter.

Sequence  Field      Attr    Length  Type       Description
          MLACCT     KEY       5,0  PACK       Account number
          MLTYPE                 1  CHAR       Acct type- 1=Bus 2=Gov 3=Org
          MLNAME                20  CHAR       Name
          MLSRCH                10  CHAR       Search field for name search
          MLADDR                20  CHAR       Addr
          MLCITY                20  CHAR       City
          MLSTAT                 2  CHAR       State
          MLZIP                 5,0  PACK       ZIP code

                                                                    Bottom
F3=Exit          F5=Refresh       F12=Cancel        F14=Display definition
F20=Renumber     F21=Select all
```

7. The Select and Sequence Fields display allows you to select the fields and the field order that your DFU program uses for each selected record format. Your field selections appear on the data entry display when you run the program. The displayed information is from the MLGMSTR DDS file descriptions.

   Press F21 to select all of the displayed fields from the displayed record format.

8. Press the Enter key again to confirm.

```
                              Define Fields

File . . . . . :     MLGMSTP        Library . . . . :    USERXX
Record Format . . . . . :     MLGMSTR

Type options, or press F21 to select all; press Enter.
  1=Select for extended definition
  4=Delete extended definition


                  Extended
Opt  Field        Definition   Heading
     MLACCT          N        Account
     MLTYPE          N        Type
  1  MLNAME          N        Name
     MLSRCH          N        Search
  1  MLADDR          N        Addr
  1  MLCITY          N        City
     MLSTAT          N        State
     MLZIP           N        ZIP

                                                                  Bottom
F3=Exit definition          F5=Refresh          F12=Cancel
F14=Display definition      F21=Select all
```

9. The Define Fields display appears. From this display you can select the fields that need extended definitions, such as headings, auto-duplicate, allow lowercase, and so on.

   Type a 1 (Select for extended definition) next to the MLNAME, MLADDR, and MLCITY fields as shown and press the Enter key. This is needed to specify that lowercase can be entered. The default is that only uppercase can be entered.

   The Specify Extended Field Definitions display allows you to define additional features for the selected alphameric fields. For the mailing list application, you should allow lowercase for the name, address, and city fields.

```
                  Specify Extended Field Definition

Field name . . . . . :   MLNAME              Format . . . . :   MLGMSTR

Type choices, press Enter.

  Auto-duplicate . . . . . .  N          Y=Yes, N=No
  Allow lowercase . . . . . .  Y         Y=Yes, N=No
  Extended field
    heading . . . . . .  Name




F3=Exit      F12=Cancel        F14=Display definition
```

10. Specify a Y for the *Allow lowercase* field for the MLNAME field and press the Enter key.
11. Do the same for the next two displays for the MLADDR and MLCITY fields.

12. Press the Enter key to return to the Define Fields display. You are finished defining the DFU program, so press the Enter key to continue to the End DFU Program Definition display.

```
                        End DFU Program Definition

Type choices, press Enter.

  Save program . . . . . .  Y          Y=Yes, N=No
  Run program . . . . . .   Y          Y=Yes, N=No
    Type of run . . . . . .  1         1=Change
                                       2=Display

  Modify program . . . . . .  N        Y=Yes, N=No


Location for saved DFU program:
  Program . . . . . .   MLGMSTU
    Library . . . . . .  *CURLIB
  Authority . . . . . .  *CHANGE
  Text  . . . . . .     MLGMSTU



F3=Exit      F14=Display definition      F17=Fast path
```

13. The End DFU Program Definition display allows you to save, run, or save and then run your newly defined DFU program.

    Press the Enter key to save your program and continue to the Change a Data File display.

    A message will appear that the DFU program was saved successfully.

Step 2. Adding Records to the Physical File (MLGMSTP)

Once the DFU program is created, you can enter the mailing list information into the physical file through DFU. Follow these steps to add the records information into the MLGMSTP data file:

1. When you finished creating the DFU program, the last display shown was the Change a Data File display.

```
                              Change a Data File

Type choices, press Enter.

  Program . . . . . .  MLGMSTU      Name, F4 for list
    Library . . . . . .  *CURLIB     Name, *LIBL, *CURLIB

  Data file . . . . . .  MLGMSTP     Name, *SAME, F4 for list

    Library . . . . . .  USERXX      Name, *LIBL, *CURLIB
  Member  . . . . . .  *FIRST      Name, *FIRST, F4 for list




F3=Exit     F4=Prompt     F12=Cancel
The DFU program was saved successfully.
```

2. Press the Enter key.

   The first display that is shown appears in entry mode because the data file is empty.

```
MLGMSTU                                        Mode . . . . :  ENTRY
Format . . . . :  MLGMSTR                      File . . . . :  MLGMSTP

Account number: 10057
Type:           1
Name:           Samuel Jones
Search:         JONES
Addr:           220 4th Ave NW
City:           Minneapolis
State:          MN
ZIP code:       55454




F3=Exit                F5=Refresh              F6=Select format
F9=Insert              F10=Entry               F11=Change
```

3. Enter the records as shown in Table 6-1. An example of the first account is shown.

   Note: You should enter all of the records shown in Table 6-1 on page 6-8 so that examples of various reports and queries shown later in this manual will be meaningful. You can also add additional records of your own. The account numbers were randomly chosen.

Table 6-1. Mailing List Data for MLGMSTU DFU Program

Account Number	Type	Name	Search	Address	City	State	Zip Code
10057	1	Samuel Jones	JONES	220 4 Ave NW	Minneapolis	MN	55454
11458	1	James Grover	GROVER	442 Belmont St	Trenton	NJ	08690
14477	4	Charles Hanley	HANLEY	331 Cheery Lane NE	Rochester	MN	55920
15902	2	Joseph Jones	JONES	3008 Brook St	Little Rock	AR	44877
18890	5	Carol Larson	LARSON	758 N Broadway	Rochester	MN	55920
26640	1	Daniel Benson	BENSON	980 Peoria Lane	Syracuse	NY	13212
24882	1	Kathryn Donty	DONTY	488 55 ST SW	Dallas	TX	75248
28903	9	Philip Jones	JONES	365 Parkway	San Diego	CA	66903
38724	5	Maria Jonesa	JONESA	559 9th Ave S	Philadelphia	PA	22809

4. When you are done entering all of the records, press F3 to exit.
5. An End Data Entry display shows the number of records added.

```
                              End Data Entry

  Number of records processed

    Added  . . . . . :          9
    Changed  . . . . :          0
    Deleted  . . . . :          0



Type choice, press Enter.

  End data entry  . . . . . .    Y          Y=Yes,  N=No

F3=Exit       F12=Cancel
All records added, changed, or deleted will be printed.
```

6. Press the Enter key to end data entry, then press F3 to exit from the AS/400 Data File Utility (DFU) menu.
7. A spooled file (named QPDZDTALOG) will be created with the Audit Report showing a list of added records. Use the Work with Output Queue (WRKOUTQ OUTQ(USERXX)) command to display the spooled file, then delete the file.

Backup and Recovery Considerations

Backup and recovery must be considered in any application. There are many different recovery situations; however, these situations can generally be categorized as follows:

- *Incorrect data* is usually caused by human error, such as entering data incorrectly, not running a program when it should be, or using a program that contains an error. These error situations are common and normally corrected by obvious solutions, such as reentering correct data, running a program, or correcting a program. In an extreme case, you may want to restore a backup version and start processing again.
- *Damage* to an object means that the object can no longer be processed by the system (such as when an object has been physically damaged). Although an object is unlikely to become damaged, recovery must be considered. To recover a damaged IBM-supplied object, you must delete and restore the object, start the system again, or install the system again. To recover a damaged user-defined object, you normally restore a saved version or create the object again.
- *System failure* can occur for various reasons, such as electrical power interruptions, and requires restarting the system. In most system failures, data on auxiliary storage remains usable. Data that is in main storage at the time of the failure will be lost. In many applications, the program is not coded for recovery and the operator may have to re-enter some data.

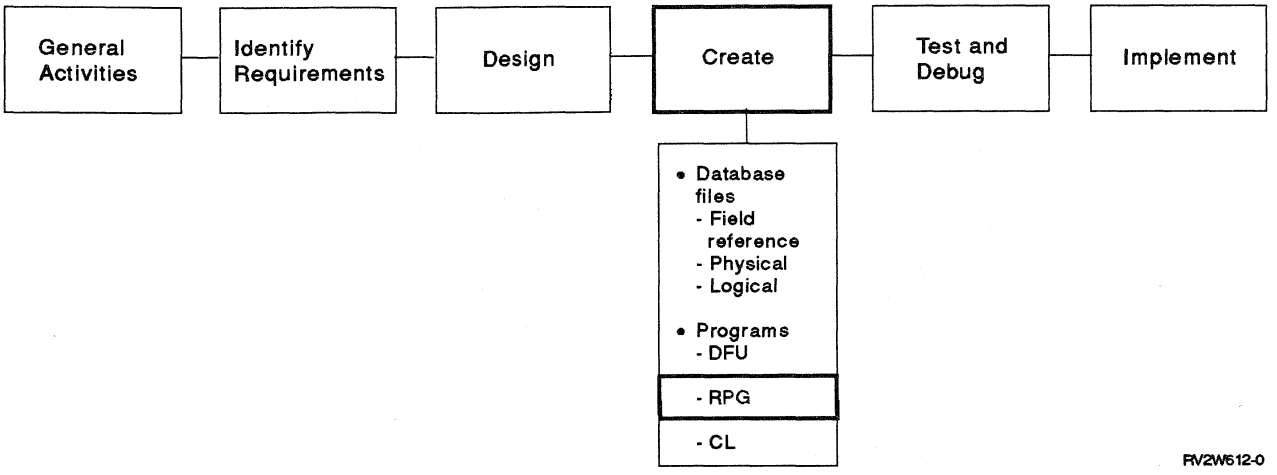
The minimum requirement is to save the objects to offline media. A typical backup solution would be to save the physical files that contain data every night. If a file needs to be restored, the users would have to rekey any changes that had occurred since the last backup.

If the system abnormally ends (such as when a power failure occurs), the data on disk is normally usable (a restore is not required). However, some of the records that were added or changed by the user may not have been written to disk when the failure occurred. The system default causes any changes to occur in main storage, but the changes are not immediately written to disk.

Recovery normally occurs by the user checking the last few changes and ensuring that they were made correctly.

System functions are available to provide for better backup and recovery. Some of these are discussed in Chapter 15, "Additional Topics."

Chapter 7. Creating the Label Printing Program



Up to this point, you have entered the mailing list data through a DFU program and created the following files and program:

- MLGSRC: Single source file
- MLGREFP: Field reference file
- MLGMSTP: Master physical file
- MLGMSTL: Logical file
- MLGMSTU: DFU program

You can now create a program to run the mailing list labels. This is done with an RPG program. This chapter describes the RPG program and goes through the following tasks:

- Entering the RPG specifications
- Compiling the RPG program
- Running the RPG program
- Displaying the results of the program

Overview of RPG Label Printing Program

An RPG label printing program named MLGLBLR is used in this application to print the labels. Figure 7-1 on page 7-2 shows an overview of the MLGLBLR RPG program.

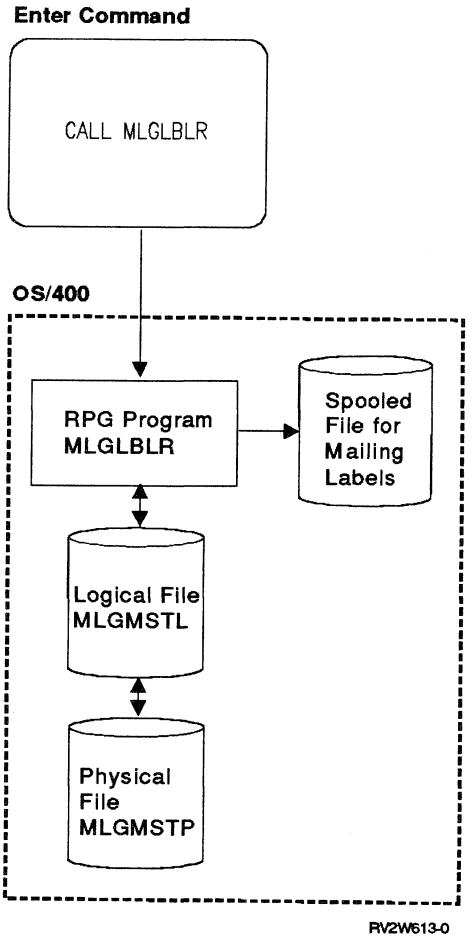


Figure 7-1. Overview of Mailing Label RPG Program (MLGLBLR)

Description of RPG/400 Specifications for Label Printing Program (MLGLBLR)

Figure 7-3 on page 7-3 shows the RPG specifications used for the MLGLBLR program. A description of each specification line follows the figure.

This program is coded with a primary file and uses the RPG program cycle. Many RPG programmers prefer to use a procedural coding solution and avoid the program cycle. The procedural method will be shown in other examples.

The format of the printed labels as defined on the printer layout form is as follows:

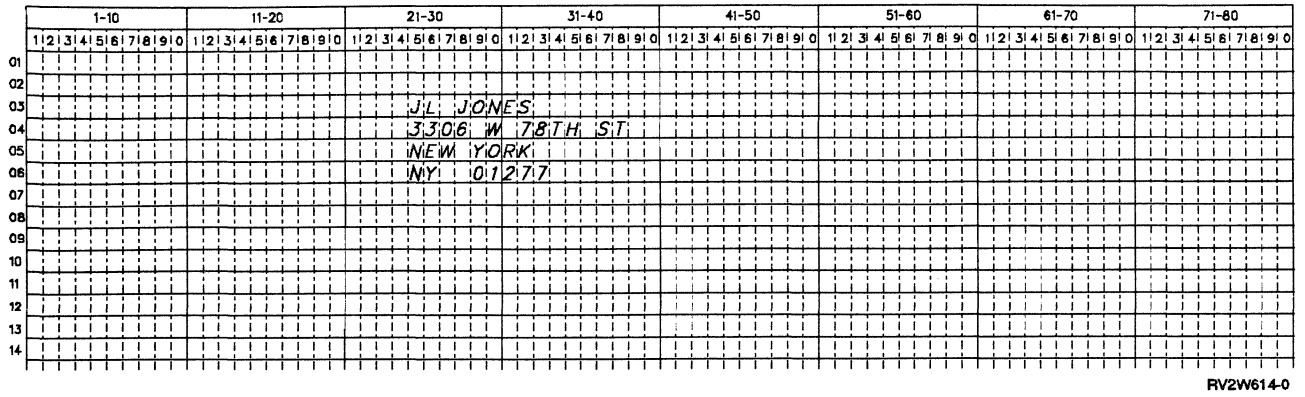


Figure 7-2. Printed Mailing Labels Format on Printer Layout Form

```
      ....+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7
0001.00     F* MLGLBLR - Mailing list label printing program
0002.00     FMLGMSTL IP  E           K        DISK
0003.00     FQPRINT  O   F     132     OF    LPRINTER
0004.00     LQPRINT    8FL  60L
0005.00     IMLGMSTR     01
0006.00     OQPRINT  D  103   01
0007.00     O                            MLNAME    44
0008.00     O        D  1     01
0009.00     O                            MLADDR    44
0010.00     O        D  1     01
0011.00     O                            MLCITY    44
0012.00     O        D  1     01
0013.00     O                            MLSTAT    26
0014.00     O                            MLZIP     33
```

Figure 7-3. RPG Specifications for MLGLBLR Label Printing Program

Line 1.00: A comment is defined by an * in position 7. A comment is used to help describe the source and is not part of the object program. The specification type (F in position 6) is ignored when a comment is specified.

Line 2.00: The MLGMSTL logical file is defined. The F specification (position 6) describes a *file* specification. The name of the file appears in positions 7 through 14. The I in position 15 specifies *input* meaning this file will be read only (it will not be updated). The P in position 16 specifies that this is the primary file of the program. A primary file in RPG means that it will be read when the file is opened, the file will be read sequentially, and the program will end when there are no more records in the file.

The E in position 19 specifies that the field definitions should come from the externally defined data described with the file. When the program is created, the RPG compiler will request the field definitions for the file from the system.

The K in position 31 specifies that the file will be processed by keys as opposed to arrival sequence. Because the key to the file specified has a high order of MLZIP, the first record read by the program will be for the lowest value in the MLZIP field (such as zip code 00001). The DISK entry in positions 40 through 46 specify that this file resides on disk.

Line 3.00: The printer file QPRINT is defined in the program. The file QPRINT is a standard printer file supplied with the operating system. The intent is that you will use it for printing when the formatting of the printed output is done inside the program (as opposed to using printer DDS and formatting outside of the program). Any file you name must exist on the system or an override must be used to describe a file that does exist.

The file QPRINT is specified as an O for output in position 15. The F entry (file format) in position 19 specifies that this is a program-described file. The definition of the format will be done inside of the RPG program (with output specifications) rather than by using an externally described file.

DDS does support externally described printer files, but none will be used for this application. For simple output, many programmers find it more convenient to place the printer specifications inside the program.

The record length entry of 132 in positions 24 through 27 describes the length of the printed output record. This entry is much wider than the size of a label. No error will occur as long as the print positions specified are less than 132. The actual width of the label will be determined by the print positions specified. A 132 record length is normally used for printed output as it is the maximum length of most print devices.

The overflow indicator OF is specified in positions 33 through 34. This indicator will be set on by RPG when printer overflow is sensed. Printer overflow means that the specified overflow line on a page has been reached. Normally the program will skip to a new page and print some headings. In this program, the overflow indicator is defined but not used because the program always skips to a new form when a label is complete.

The L in position 39 specifies that a line counter specification will be used for this printer file. The line counter specification is rarely used, but is useful for a program that does not want to use an Override with Printer File (OVRPRTF) CL command to specify form length and the overflow line. The PRINTER entry in positions 40 through 46 identifies the device to be used as a printer.

Line 4.00: The line counter specification is defined (L in position 6). This specification is seldom used with an RPG program because most programmers prefer to use the OVRPRTF CL command. In this example, the RPG program will be self-contained and will not need an override outside of the program.

The 8 in position 17 with the FL entry indicates that there will be 8 form lines in the label to be used.

The 6 in position 22 and the OL in positions 23 and 24 indicate the overflow line is line 6.

These statements are needed to allow the program to skip to line 3 of a new page (meaning the next label) and have the program and the system understand what that means.

Printer Overflow: A printer device holding continuous forms is not aware of where the form is positioned nor when the end of a form is reached. The printer is just aware of a continuous roll of paper passing through. The printer file that you specify must describe the length of the form in lines and where the overflow line is. When a print line causes spacing or skipping, the OS/400 program keeps

a count of the lines that you have printed and is thus able to determine where you are on the printed page. When you insert a form to print on, you must align the form properly in the printer to synchronize where the form physically is and where the system will begin the line counting.

Line 5.00: The I in position 6 describes an input specification. The name in positions 7 through 14 describes the record format (MLGMSTR) of the file specified on line 2.00. In case the file has multiple formats, the record name, not the file name, is used.

The only entry on the input specification is to specify the indicator 01 in positions 19 and 20. This means that RPG will set on indicator 01 whenever the record format MLGMSTR is read. Because there are no other record formats in the file, indicator 01 will be on for every record read in the program.

Programming Notes: In many programs using externally described data, no input specifications are needed. However, if you have a primary file, you will normally need to describe an indicator to the record format in the file.

Line 6.00: The first output line to the QPRINT file is described. The file name in positions 7 through 14 defines the file to be spooled to. The remaining output lines all have a blank file name which tells RPG that they are all spooled to the previously described file (QPRINT in this case).

The D in position 15 describes *detail time* to the RPG program cycle. Detail time means the output line will be printed after a record is read. The 1 in position 18 is the *space after* specification meaning that after this line is printed, the printer will logically space one line. Some printers only support spacing before printing, so the system may need to perform the requested function just prior to the next line being printed. From a programmer viewpoint, the printer can both space and skip before and after the printed line.

The 03 entry in positions 19 and 20 describes a *skip before* to line 03. This is the first line on each label to be printed.

The 01 in positions 24 and 25 specifies that this line should be printed each time indicator 01 is on when the RPG program cycle comes to detail calculation. This essentially means once per record.

Line 7.00: The field is specified to end printing in position 44.

Lines 8.00 through 14.00: The other print lines are specified. Each D entry in position 15 specifies a new line to be printed. Each line specifies space one after printing. The last line has two fields to be printed.

Note that there are no field definitions in the program. The fields used (such as MLACCT) will be defined by the compiler accessing the externally described definition for the file MLGMSTL. All fields used in the program must be defined or the compilation step will fail.

Step 1. Entering the RPG Specifications and Creating the RPG Label Printing Program (MLGLBLR)

You should now enter the RPG specifications as shown in Figure 7-3 on page 7-3. through PDM. Follow the same steps you used to create the MLGMSTP physical file in Chapter 5, "Creating Files" or copy the source from the QUSRTOOL library as described in Appendix B, "Overview of QUSRTOOL Library." The member source type should be RPG.

Once you have entered the RPG specifications, create the program by typing a 14 (Compile) in the *Opt* column next to the program on the Work with Members Using PDM display.

Step 2. Running the Label Printing Program (MLGLBLR)

After you get the completion message when the label printing program is created, you can run the program. Run the program now by typing the following on the command entry line:

```
CALL MLGLBLR
```

Step 3. Displaying the Output

When the program has run, follow these steps to look at the spooled file in your output queue.

1. Type the following on the command entry line where USERXX is the name of your output queue.

```
WRKOUTQ OUTQ(USERXX)
```

```
                          Work with Output Queue

Queue:   USERXX         Library:   USERXX         Status:   RLS

Type options, press Enter.
  2=Change   3=Hold   4=Delete   5=Display   6=Release   8=Attributes

Opt  File        User        User Data   Sts   Pages  Copies  Form Type   Pty
     MLGLBLR     USERXX                  RDY       6       1  *STD          5
 5   QPRINT      USERXX      MLGLBLR     RDY       9       1  *STD          5

                                                                        Bottom
Parameters for option 2 or command
===>
F3=Exit   F11=View 2   F12=Cancel   F22=Printers   F24=More keys
```

2. Type a 5 (Display) in the *Opt* column next to the QPRINT file.

The format of your output should look similar to the following display.

```
                              Display Spooled File
File . . . . . :   QPRINT                          Page/Line   1/3
Control . . . . .                                  Columns     1 - 78
Find . . . . . .
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+...
                      James Grover
                      442 Belmont St
                      Trenton
                      NJ  08690
                      Daniel Benson
                      980 Peoria Lane
                      Syracuse
                      NY  13212
                      Maria Jonesa
                      559 9th Ave S
                      Philadelphia
                      PA  22809
                      Joseph Jones
                      3008 Brook St
                      Little Rock
                      AR  44877

                                                                        More...
F3=Exit    F12=Cancel    F19=Left    F20=Right    F24=More keys
```

3. The Display Spooled File display does not show blank lines. If the labels were printed, blank lines would appear as shown in Figure 7-4 on page 7-8.

   Press F3 (Exit) to return to the Work with Output Queue display.

```
                              Work with Output Queue

Queue:   USERXX          Library:   USERXX          Status:   RLS

Type options, press Enter.
  2=Change   3=Hold   4=Delete   5=Display   6=Release   8=Attributes

Opt  File       User       User Data   Sts   Pages  Copies  Form Type   Pty
 4   MLGLBLR    USERXX                 RDY       6       1  *STD          5
 4   QPRINT     USERXX     MLGLBLR     RDY       9       1  *STD          5

                                                                        Bottom
Parameters for option 2 or command
===>
F3=Exit    F11=View 2    F12=Cancel    F22=Printers    F24=More keys
```

4. Type a 4 (Delete) in the *Opt* column next to the spooled files created from running the MLGLBLR program (MLGLBLR and QPRINT) and press the Enter key. Press the Enter key again to confirm the choices. Then, press F3 to exit.

An example of the printed mailing labels from the MLGLBLR program is shown in Figure 7-4.

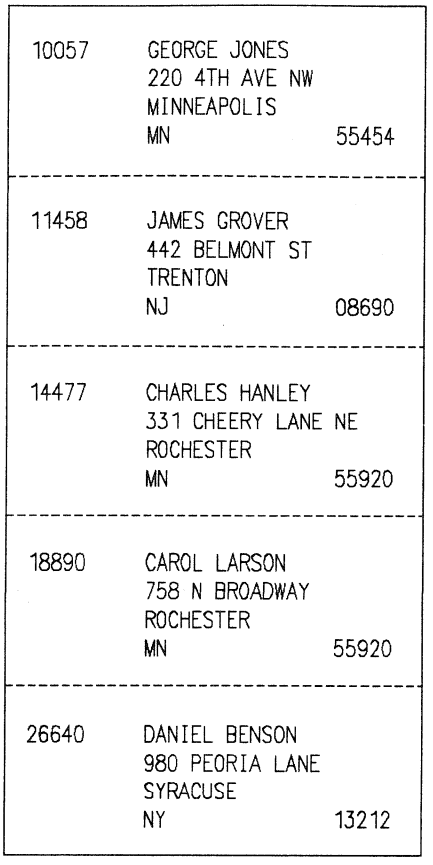
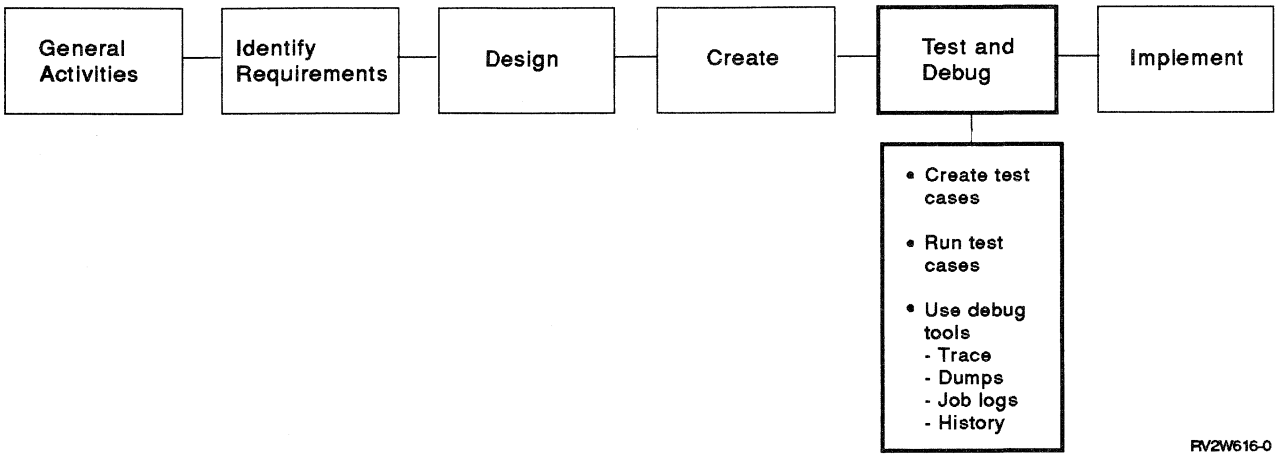


Figure 7-4. Example of Printed Mailing Labels

Chapter 8. Testing and Debugging



This chapter goes through some testing and debugging examples for compilation and object errors. Logic errors are also discussed. Some of the tasks include:

- Making an error in RPG source and locating it using compiler messages
- Deleting an object used in a program, calling the program, and debugging the program

Compilation Errors

Compilation errors occur when you try to create or compile a new file or program. A typical compilation error would be an error in the source coding, possibly a typographical error.

Step 1. Creating the Compilation Error

The following steps go through creating a compilation error in your RPG source, and then debugging the program.

Follow steps 1 through 3 if you are not already in PDM.

1. Start PDM by typing the following command on the command entry line:

   ```
   STRPDM
   ```

2. Select option 3 (Work with members) and press the Enter key.

```
                    Specify Members to Work With

Type choices, press Enter.

  File . . . . . .   MLGSRC       Name

    Library . . . . . .   USERXX       *LIBL, *CURLIB, name

  Member:
    Name . . . . . .   *ALL         *ALL, name, *generic*
    Type . . . . . .   *ALL         *ALL, type, *generic, *BLANK*









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

3. Specify the file and library as shown above and press the Enter key.

```
                        Work with Members Using PDM

File . . . . . .   MLGSRC
  Library . . . .  USERXX               Position to . . . . .

Type options, press Enter.
  2=Edit         3=Copy       4=Delete       5=Display      6=Print
  7=Rename       8=Display description       9=Save         13=Change text

Opt Member     Type       Text
2    MLGLBLR     RPG          Mailing list label printing program
     MLGMSTL     LF           Mailing list label printing logical
     MLGMSTP     PF           Mailing list master physical file
     MLGREFP     PF           Mailing list field reference file




                                                                      Bottom
Parameters or command
===>
F3=Exit          F4=Prompt            F5=Refresh           F6=Create
F9=Retrieve      F10=Command entry    F23=More options     F24=More keys
```

4. Type a 2 (Edit) in the *Opt* column next to the MLGLBLR member and press the Enter key.

```
Columns . . . .:   1  71              Edit                    USERXX/MLGSRC
Find  . . .                                                       MLGLBLR
FMT *  ..... *. 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7
       ********** Beginning of data ************************************
0001.00      F* MLGLBLR - Mailing list label printing program
0002.00      FMLGMSTL IP  E           K        DISK
0003.00      FQPRINT  O   F     132     OF    LPRINTER
0004.00      LQPRINT    8FL  6OL
0005.00      IMLGMSTR     01
0006.00      OQPRINT  D  103   01
0007.00      O                          MXNAME    44
0008.00      O        D  1     01
0009.00      O                          MLADDR    44
0010.00      O        D  1     02
0011.00      O                          MLCITY    44
0012.00      O        D  1     01
0013.00      O                          MLSTAT    26
0014.00      O                          MLZIP     33
       ****************** End of data ***************************************


F3=Exit                  F4=Prompt               F5=Refresh
F10=Top                  F11=Bottom              F24=More keys
                                  (C) COPYRIGHT IBM CORP. 1981, 1989.
```

5. To create typical source errors in the RPG source, go to line 0007.00 and change the L in MLNAME to an X. Then, change indicator 01 on line 0010.00 to 02 and press the Enter key. Then, press F3 (Exit).
6. On the Exit display, press the Enter key to take the defaults and change the member.

```
                          Work with Members Using PDM

File . . . . . .   MLGSRC
  Library . . . .  USERXX                Position to  . . . . .

Type options, press Enter.
  14=Compile     17=Change using SDA     25=Find string


Opt  Member      Type        Text
14   MLGLBLR     RPG         Mailing list label printing program
     MLGMSTL     LF          Mailing list label printing logical
     MLGMSTP     PF          Mailing master physical file
     MLGREFP     PF          Mailing list field reference file




Parameters or command
===>
F3=Exit          F4=Prompt            F5=Refresh           F6=Create
F9=Retrieve      F10=Command entry    F23=More options     F24=More keys
```

7. Try to compile the member by typing a 14 next to the MLGLBLR member and pressing the Enter key.

   The Confirm Compile of Member display is shown.

```
                        Confirm Compile of Member

The following object already exists for the compile operation:

  Object which exists . . . . . :   MLGLBLR
    Library . . . . . :   USERXX
  Object type . . . . . :   *PGM

  Member to compile . . . . . :   MLGLBLR
  File . . . . . :   MLGSRC
    Library . . . . . :   USERXX

Type choice, press Enter.

  Delete existing object . . . . . .  Y   Y=Yes, N=No

Press F12=Cancel to return and not perform the compile operation
  (existing object will not be deleted).



F12=Cancel
```

8. Type a Y (Yes) for *Delete existing object* and press the Enter key.

   The job should go to batch processing. When the job has ended, you should get a message that the job ended abnormally.

   When a program is being re-created, the CRTRPGPGM defaults to REPLACE(*YES). When this is specified, the existing version will not be deleted until the new version is successfully created. Since the new version will not create successfully, the old version will still exist. You will want to determine why the create request failed.

Step 2. Debugging the Program

1. Display your spooled files so that you can look at the listing by typing the following on the command entry line.

   ```
   WRKOUTQ OUTQ(USERXX)
   ```

```
                          Work with Output Queue

Queue:   USERXX         Library:   USERXX         Status:   RLS

Type options, press Enter.
  2=Change   3=Hold   4=Delete   5=Display   6=Release   8=Attributes

Opt  File        User        User Data   Sts   Pages  Copies  Form Type   Pty
     MLGLBLR     USERXX                  RDY       6       1  *STD         5
 5   QPJOBLOG    USERXX      MLGLBLR     RDY       1       1  *STD         5

                                                                        Bottom
Parameters for option 2 or command
===>
F3=Exit   F11=View 2   F12=Cancel   F22=Printers   F24=More keys
```

2. There should be an output queue file (MLGLBLR) and a job log (QPJOBLOG).

Note: The job log may be sent to another queue depending on your installation. If this is the case, you can see the job log by using the WRKSPLF command.

First, type a 5 next to the job log to display it.

```
                          Display Spooled File
File . . . . . :   QPJOBLOG                          Page/Line   1/1
Control . . . . .                                    Columns     1 - 78
Find . . . . . .
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+...
 5728SS1 R02 M00 891006                          Job Log
  Job name . . . . . . . . . . :   MLGLBLR        User . . . . :   USERXX
  Job description . . . . . . . :   QBATCH        Library . . . :   QGPL
TIME   MSGID  SEV TYPE  MESSAGE TEXT
161041 CPF1124 00 INFO  Job 000294/USERXX/MLGLBLR started on 05/08/89 at 16:10
                         in QSYS. Job entered system on 05/08/89 at 16:10:41.
161041 CPI1125 00 INFO  Job 000294/USERXX/MLGLBLR submitted.
                            Cause . . . . . :   Job 000294/USERXX/MLGLBLR subm
                        TCH in QGPL from job 000287/USERXX/DSP020604. Job 0002
                        started using the Submit Job (SBMJOB) command with the
                        tes: JOBPTY(5) OUTPTY(5) PRTTXT() RTGDTA(QCMDB) SYSLIB
                          QUSRSYS) CURLIB(USERXX) INLLIBL(QTEMP      QGDDM
                          RELIABL    QRPG       SYSUSE) LOG(4 00 *NOLIST) LOGC
                        QD) OUTQ(USERXX/USERXX) PRTDEV(IDPRT) HOLD(*NO) DATE(*
                        MSGQ(QUSRSYS/USERXX).
                  RQS  -CRTRPGPGM   PGM(USERXX/MLGLBLR) SRCFILE(USERXX/MLGSRC)
                                                                        More...
F3=Exit   F12=Cancel   F19=Left   F20=Right   F24=More keys
```

3. Scan through the job log to find any errors.

```
                          Display Spooled File
File . . . . . :   QPJOBLOG                          Page/Line   1/21
25
Find . . . . . .
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+...
                        LACE(*YES)
161042 CPF1015 40 ESC   Data area RPGHSPEC in *LIBL not found.
                            Recovery . . . :   Either correct the data area n
                        ary name (DTAARA parameter).  Then try the request aga
161042 CPF1015 40 ESC   Data area DFTHSPEC in QRPG not found.
                            Recovery . . . :   Either correct the data area n
                        ary name (DTAARA parameter).  Then try the request aga
161043 QRG0008 50 COMP  Compile terminated. Severity level 30 errors found in
                            Cause . . . . . :   The RPG III compiler found at
                         source member of severity level equal to or greater t
                        the GENLVL option on the CRTRPGPGM command. Recovery
                        errors in the source member. Recompile.
161043 CPC0904 00 COMP  Data area RETURNCODE created in library QTEMP.
161044 QRG9001 50 ESC   Compile failed. Program not created.
                            Cause . . . . . :   See compiler listing for error
                         normally occurs when severity of issued message excee
                                                                        More...
F3=Exit   F12=Cancel   F19=Left   F20=Right   F24=More keys
```

```
                              Display Spooled File
File . . . . . :   QPJOBLOG                          Page/Line   1/39
Control . . . . .                                    Columns     1 - 78
Find . . . . . .
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+...
                         GENLVL parameter on CRTRPGPGM command. Recovery  . .
                        r change the value specified for the GENLVL parameter.
161044 CPC2402 50 COMP  Job canceled. Cancel message received at command proce
                            Cause . . . . . :   A message with a severity equa
                        cancel severity was received at the command processor.
                        ee the messages previously listed to determine the mes
                        ob to be canceled. Correct the errors, and then try th
161044 CPC2191 00 COMP  Object RETURNCODE in QTEMP type *DTAARA deleted.
161044 CPF1164 00 COMP  Job 000294/USERXX/MLGLBLR completed 05/08/89 16:10:44.
                        unit time, ending code 20 .
                            Cause . . . . . :   The job ended after 1 routing
                        s and meanings are as follows: -- 0-Normal completion.
                        on during controlled ending or controlled subsystem en
                        ed end severity (ENDSEV). -- 30-Job ended abnormally.
                         becoming active. -- 50-Job ended while active. -- 60-
                        ally while job was active. -- 70-System ended abnormal
                                                                        More...
F3=Exit    F12=Cancel    F19=Left    F20=Right    F24=More keys
```

```
                              Display Spooled File
File . . . . . :   QPJOBLOG                          Page/Line   1/55
Control . . . . .                                    Columns     1 - 78
Find . . . . . .
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+...
                        e. -- 80-Job ended using ENDJOBABN. -- 90-Job forced t
                        imit expired for ENDJOBABN. Recovery  . . :   For mo
                         Programming: Work Management Guide, SC21-8078.









                                                                        Bottom
F3=Exit    F12=Cancel    F19=Left    F20=Right    F24=More keys
```

4. The job log provides message QRG9001 that states that the compile failed. Press F3 (Exit). Then on the Work with Output Queue display, type a 5 next to the MLGLBLR file to display it.

   Note: Normally, you would not look at a job log if your compile fails. Most likely your error is in your source and the job log only states that the compile failed.

5. The following display shows the first page of the compilation listing. Continue to page down to look for errors.

```
                          Display Spooled File
File . . . . . :   MLGLBLR                        Page/Line   1/1
Control . . . . .                                 Columns     1 - 78
Find . . . . . .
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+...
 5728RG1 R02M00  891006            IBM AS/400 RPG/400                   GREEN
 Compiler . . . . . . . . . . . . :   IBM AS/400 RPG/400
 Command Options:
   Program . . . . . . . . . . . . :   USERXX/MLGLBLR
   Source file . . . . . . . . . . :   USERXX/MLGSRC
   Source member . . . . . . . . . :   MLGLBLR
   Source listing options . . . . . :   *SOURCE     *XREF       *GEN        *N
   Generation options . . . . . . . :   *NOLIST     *NOXREF     *NOATR      *N
   SAA flagging . . . . . . . . . . :   *NOFLAG
   Generation severity level  . . . :   9
   Print file . . . . . . . . . . . :   *LIBL/QSYSPRT
   Replace program  . . . . . . . . :   *YES
   Target release . . . . . . . . . :   *CURRENT
   User profile . . . . . . . . . . :   *USER
   Authority  . . . . . . . . . . . :   *CHANGE
   Text . . . . . . . . . . . . . . :   *SRCMBRTXT
                                                                      More...
F3=Exit   F12=Cancel   F19=Left   F20=Right   F24=More keys
```

```
                          Display Spooled File
File . . . . . :   MLGLBLR                        Page/Line   1/19
Control . . . . .                                 Columns     1 - 78
Find . . . . . .
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+...
   Phase trace  . . . . . . . . . :   *NO
   Intermediate text dump . . . . . :   *NONE
   Snap dump  . . . . . . . . . . . :   *NONE
   Codelist . . . . . . . . . . . . :   *NONE
   Ignore decimal data error  . . . :   *NO
 Actual Program Source:
   Member . . . . . . . . . . . . . :   MLGLBLR
   File . . . . . . . . . . . . . . :   MLGSRC
   Library  . . . . . . . . . . . . :   USERXX
   Last Change  . . . . . . . . . . :   05/11/89  15:51:12
   Description  . . . . . . . . . . :   Mailing list label printing
 5728RG1 R02M00  891006            IBM AS/400 RPG/400                   GREEN
 SEQUENCE
 NUMBER    *...1....+....2....+....3....+....4....+....5....+....6....+....7..
                          S o u r c e   L i s t i n g
      100  F* MLGLBLR - Mailing list label printing program
                                                                      More...
F3=Exit   F12=Cancel   F19=Left   F20=Right   F24=More keys
```

```
                          Display Spooled File
File . . . . . :   MLGLBLR                          Page/Line   2/10
Control . . . . .                                   Columns     1 - 78
Find . . . . . .
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+...
          H
      200  FMLGMSTL IP  E           K        DISK
            RECORD FORMAT(S):  LIBRARY USERXX FILE MLGMSTL.
                     EXTERNAL FORMAT MLGMSTR RPG NAME MLGMSTR
      300  FQPRINT  O   F     132     OF    LPRINTER
      400  LQPRINT    8FL  60L
      500  IMLGMSTR     01
      500   INPUT  FIELDS FOR RECORD MLGMSTR FILE MLGMSTL FORMAT MLGMSTR.
      500         Mailing list master record
  A000001                                        P   1   30MLACCT
  A000002                                            4    4 MLTYPE
  A000003                                            5   24 MLNAME
  A000004                                           25   34 MLSRCH
  A000005                                           35   54 MLADDR
  A000006                                           55   74 MLCITY
  A000007                                           75   76 MLSTAT
                                                                    More...
F3=Exit   F12=Cancel   F19=Left   F20=Right   F24=More keys
```

On the following display, diagnostic message 7086 refers to statement 200. It states that RPG will provide block support. This means that RPG will have a buffer of several records that is filled by the system. This is a performance advantage and is usually desirable in most applications. Message 7086 is a warning message only and is not the cause of the compilation failure.

```
                          Display Spooled File
File . . . . . :   MLGLBLR                          Page/Line   2/33
Control . . . . .                                   Columns     1 - 78
Find . . . . . .
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+...
  A000008                                        P  77  79OMLZIP
      600  OQPRINT  D  103   01
      700  O                          MXNAME    44
      800  O        D  1     01
      900  O                          MLADDR    44
     1000  O        D  1     02
     1100  O                          MLCITY    44
     1200  O        D  1     01
     1300  O                          MLSTAT    26
     1400  O                          MLZIP     33
           * * * * *  E N D  O F  S O U R C E  * * * * *
         A d d i t i o n a l  D i a g n o s t i c  M e s s a g e s
* 7086       200   RPG PROVIDES BLOCK OR UNBLOCK SUPPORT FOR FILE MLGMSTL.
 5728RG1 R02M00  891006             IBM AS/400 RPG/400                  GREEN
                 K e y  F i e l d  I n f o r m a t i o n
                     PHYSICAL    LOGICAL
                                                                    More...
F3=Exit   F12=Cancel   F19=Left   F20=Right   F24=More keys
```

```
                              Display Spooled File
File . . . . . :   MLGLBLR                          Page/Line   3/9
Control . . . . .                                   Columns     1 - 78
Find . . . . . .
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+...
        FILE/RCD       FIELD       FIELD       ATTRIBUTES
    01  MLGMSTL
            MLGMSTR
                       MLZIP                   PACK  5,0  SIGNED
                       MLSTAT                  CHAR    2
                       MLCITY                  CHAR   20
                       MLSRCH                  CHAR   10
 5728RG1 R02M00  891006              IBM AS/400 RPG/400                  GREEN
                          C r o s s   R e f e r e n c e
 File and Record References:
        FILE/RCD    DEV/RCD     REFERENCES (D=DEFINED)
    01  MLGMSTL     DISK            200D
          MLGMSTR                   200D      500
    02  QPRINT      PRINTER         300D      400       600       800      1000
                                   1200
 Field References:
                                                                          More...
F3=Exit   F12=Cancel   F19=Left   F20=Right   F24=More keys
```

```
                              Display Spooled File
File . . . . . :   MLGLBLR                          Page/Line   4/22
Control . . . . .                                   Columns     1 - 78
Find . . . . . .
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+...
        FIELD       ATTR    REFERENCES (M=MODIFIED D=DEFINED)
* 7031  MLACCT      P(5,0)  A000001D
        MLADDR      A(20)   A000005D     900
        MLCITY      A(20)   A000006D    1100
* 7031  MLNAME      A(20)   A000003D
* 7031  MLSRCH      A(10)   A000004D
        MLSTAT      A(2)    A000007D    1300
* 7031  MLTYPE      A(1)    A000002D
        MLZIP       P(5,0)  A000008D    1400
* 7030  MXNAME      A(4)        700
 Indicator References:
        INDICATOR  REFERENCES (M=MODIFIED D=DEFINED)
        LR            200D
* 7031  OF            300D
        01            500M      600       800      1200
* 7030  02           1000
                                                                          More...
F3=Exit   F12=Cancel   F19=Left   F20=Right   F24=More keys
```

On the above display, diagnostic message 7031 describes fields which are defined in the program but not used. (The text of the message appears at the end of the listing.) This is a normal warning condition when using an externally described file because all of the fields are defined, but you normally don't use all of them. This is only a warning message and not the cause of the compilation failure.

Diagnostic message 7030 (described at the end of the listing) describes a condition where a name or indicator is used but has not been defined. Two cases exist, MXNAME and indicator 02, which are the error conditions that were entered. Because all fields must be defined, this type of error will cause a failure.

The listing describes the attributes of the field and where the field is used in the program as follows:

- Field MLACCT is defined as P for packed.
- The (5,0) entry describes the field as having 5 digits and 0 decimals.

- The A0000001 entry describes where the field is found in the program.
- The *A* means that it is an externally described field and the definition has been provided by copying in the description of all of the fields for the referenced file.
- The *D* means this is the definition of the field.

Most of the other fields are described as *A* meaning characters. The MXNAME field is given a definition of 4 bytes. RPG will assign a default value to let the compiler continue. The *700* entry describes the line number where the field is used. In SEU, line number 700 appears as 7.00. RPG drops the decimal point when the source is listed.

```
                          Display Spooled File
File . . . . . :   MLGLBLR                          Page/Line   4/47
Control . . . . .                                   Columns     1 - 78
Find . . . . . .
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+...
     * * * * *  E N D  O F  C R O S S  R E F E R E N C E  * * * * *
 5728RG1 R02M00  891006            IBM AS/400 RPG/400                  GREEN
                       M e s s a g e  S u m m a r y
* QRG7030 Severity:  30   Number:    2
          Message . . . . :   The Field or indicator is not defined.
* QRG7031 Severity:  00   Number:    5
          Message . . . . :   The Name or indicator is not referenced.
* QRG7086 Severity:  00   Number:    1
          Message . . . . :   The RPG handles blocking function for file.
            INFDS contents updated only when blocks of data transferred.
     * * * * *  E N D  O F  M E S S A G E  S U M M A R Y  * * * * *
 5728RG1 R02M00  891006            IBM AS/400 RPG/400                  GREEN
                         F i n a l  S u m m a r y
 Message Count:  (by Severity Number)
            TOTAL    00     10     20     30     40     50
              8       6      0      0      2      0      0
                                                                    More...
F3=Exit   F12=Cancel   F19=Left   F20=Right   F24=More keys
```

```
                          Display Spooled File
File . . . . . :   MLGLBLR                          Page/Line   6/11
Control . . . . .                                   Columns     1 - 78
Find . . . . . .
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+...
 Program Source Totals:
    Records . . . . . :   14
    Specifications  . . . . . :   13
    Table Records . . . . . :   0
    Comments  . . . . . :   1
 Compile terminated. Severity level 30 errors found in file.
         * * * * *  E N D  O F  C O M P I L A T I O N  * * * * *
                                                                    Bottom
F3=Exit   F12=Cancel   F19=Left   F20=Right   F24=More keys
```

The last page of the spooled output describes the error message text and a summary of the errors. The message count shows a total of what needs to be corrected to make a successful compilation. By default, any severity of 10 or greater will cause the compilation to fail. In this example, both serious errors occurred because the fields were undefined.

Viewing Spooled Output Using SEU

Many programmers prefer an alternate method of viewing the spooled output using SEU. SEU provides a split screen option that allows you to see the errors on the same display you are using to change the source. There is special support to display the message text with the statement in error.

To use the split screen option, you would follow these steps once you have created the errors and attempted to compile the MLGLBLR program.

1. On the Work with Members Using PDM display, select option 2 to edit the MLGLBLR member.

```
Columns . . . .:   1  71              Edit                    USERXX/MLGSRC
Find  . . .                                                        MLGLBLR
FMT *  ..... *. 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7
       ********** Beginning of data ***************************************
0001.00      F* MLGLBLR - Mailing list label printing program
0002.00      FMLGMSTL IP  E           K        DISK
0003.00      FQPRINT  O   F     132     OF    LPRINTER
0004.00      LQPRINT    8FL  6OL
0005.00      IMLGMSTR     01
0006.00      OQPRINT  D  103   01
0007.00      O                          MXNAME    44
0008.00      O        D  1     01
0009.00      O                          MLADDR    44
0010.00      O        D  1     02
0011.00      O                          MLCITY    44
0012.00      O        D  1     01
0013.00      O                          MLSTAT    26
0014.00      O                          MLZIP     33
       ****************** End of data *************************************


F3=Exit                  F4=Prompt               F5=Refresh
F10=Top                  F11=Bottom              F24=More keys
                                  (C) COPYRIGHT IBM CORP. 1981, 1989.
```

2. On the Edit display, move your cursor half-way down the display (just after statement 6.00) and press F15 for the browse/copy services function.

```
                        Browse/Copy Services

Type choices, press Enter.

  Selection . . . . . .  2              1=Member
                                        2=Spool file
                                        3=Output queue
  Copy all records . . . . . .  N       Y=Yes, N=No
  Browse/copy member . . . . .  MLGLBLR Name, F4 for list
    File . . . . . . . . . . .  MLGSRC  Name
      Library . . . . . . . .  USERXX   Name, *CURLIB, *LIBL

  Browse/copy spool file . . . .  MLGLBLR  Name
    Job . . . . . . . . . . . .  MLGLBLR  Name
      User . . . . . . . . . .  USERXX    Name
      Job number . . . . . . .  *LAST     Number, *LAST
    Spool number . . . . . . .  *LAST     Number, *LAST, *ONLY

  Display output queue . . . . .  QPRINT  Name, *ALL
    Library . . . . . . . . . .  *LIBL    Name, *CURLIB, *LIBL

F3=Exit          F4=Prompt         F5=Refresh       F12=Cancel
F13=Change defaults                F14=Find/Change Services
```

3. On the Browse/Copy Services display, type a 2 for the Spool file and press Enter. The spool file fields have already been filled in for you with the name of the last spooled file for member MLGLBLR.

   You will see a split screen with the Edit display on top and the Browse display for the spooled file on the bottom as follows:

```
 Columns . . . .:   1  71              Edit                    USERXX/MLGSRC
 Find  . . .                                                       MLGLBLR
 FMT *  ..... *. 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7
        ********** Beginning of data ************************************
0001.00      F* MLGLBLR - Mailing list label printing program
0002.00      FMLGMSTL IP  E           K        DISK
0003.00      FQPRINT  O   F     132     OF    LPRINTER
0004.00      LQPRINT    8FL  60L
0005.00      IMLGMSTR     01
0006.00      OQPRINT  D  103   01

 Columns . . . .:   1  71              Browse     Spool file  . . :    MLGLBLR
 Find  . . . *
        ********** Beginning of data ************************************
0000.01  5728RG1 R02M00  891006             IBM AS/400 RPG/400
0000.02  Compiler . . . . . . . . . . . . :   IBM AS/400 RPG/400
0000.03  Command Options:
0000.04    Program  . . . . . . . . . . . :   USERXX/MLGLBLR
0000.05    Source file  . . . . . . . . . :   USERXX/MLGSRC
0000.06    Source member  . . . . . . . . :   MLGLBLR

 F3=Exit                    F5=Refresh               F6=Move split line
 F12=Cancel                 F24=More Keys
```

4. SEU has special support when you enter an * in the *Find* field (of the bottom section) and use F16. This will cause a search for the first statement in error. The associated text will appear at the bottom of the display.

   As each error is displayed, you can page through to find the actual statement on the upper half of the display and correct the error. (You must move the cursor to the upper half before paging.)

   Then move the cursor to the bottom half of the display and press F16 again. It will find the next error and display the associated error text.

   As you can see, with this function, you can easily view the spooled file and at the same time, make the corrections without going in and out of files.

   When you have made the corrections to the MLGLBLR member, press F3 (Exit).

5. On the Exit display, press the Enter key to take the defaults and change the member.

Once you have found where the errors are, you can repeat the steps you used earlier to create the error, to correct the error, and to re-create the file.

Object Errors

Object errors occur from a variety of situations. A typical case is where a program tries to access a file and the file does not exist. For example, someone may have deleted the file or renamed it.

Step 1. Creating the Object Error

The following steps take you through creating and debugging an object error where a program is called and there is no logical file.

```
                AS/400 Programming Development Manager (PDM)

Select one of the following:

     1. Work with libraries
     2. Work with objects
     3. Work with members

     9. Work with user-defined options









Selection or command
===> 2

F3=Exit       F4=Prompt       F9=Retrieve       F10=Command entry
F12=Cancel    F18=Change defaults
```

1. On the AS/400 Programming Development Manager (PDM) display, select option 2 to work with objects.

```
                    Specify Objects to Work With

Type choices, press Enter.

  Library . . . . . .  USERXX       *CURLIB, name

  Object:
    Name . . . . . .   *ALL         *ALL, name, *generic*
    Type . . . . . .   *ALL         *ALL, *type
    Attribute . . . .  *ALL         *ALL, attribute, *generic*,
                                    *BLANK









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

2. On the Specify Objects to Work With display, type the name of your library and press the Enter key.

   All of the objects in your library will be displayed. If you do not have a unique library for the mailing list objects, type MLG in the *Position to* field and press the Enter key.

```
                        Work with Objects Using PDM

Library . . . . .   USERXX          Position to . . . . . .
                                    Position to type  . . . . .

Type options, press Enter.
  2=Change       3=Copy        4=Delete      5=Display      7=Rename
  8=Display description        9=Save        10=Restore     11=Move ...

Opt  Object      Type        Attribute   Text
     MLGLBLR     *PGM        RPG         Mailing list label printing
     MLGMSTU     *PGM        DFU         MLGMSTU
     USERXX      *OUTQ                   USERXX personal output queue
7    MLGMSTL     *FILE       LF          Mailing list label printing logical
     MLGMSTP     *FILE       PF-DTA      Mailing master file
     MLGMSTU     *FILE       DFU         MLGMSTU
     MLGREFP     *FILE       PF-DTA      Mailing list field reference file
     MLGSRC      *FILE       PF-SRC
                                                                      Bottom
Parameters or command
===>
F3=Exit          F4=Prompt           F5=Refresh          F6=Create
F9=Retrieve      F10=Command entry   F23=More options    F24=More keys
```

3. To provide an example where the logical file does not exist, the file will be renamed to a different name. Then the program will be called that will still attempt to access the previous name.

   Type a 7 in the *Opt* column to rename the MLGMSTL logical file and press the Enter key.

```
                              Rename Objects

Library . . . . . :   USERXX

To rename object, type New Name, press Enter.

Object      Type        New Name
MLGMSTL     *FILE       MLGMSTM




                                                                      Bottom
F3=Exit     F5=Refresh     F12=Cancel     F19=Submit to batch
```

4. On the Rename Objects display, type MLGMSTM for the new name and press the Enter key.

```
                        Work with Objects Using PDM

Library . . . . .   USERXX          Position to . . . . . .
                                    Position to type  . . . . .

Type options, press Enter.
  2=Change       3=Copy        4=Delete      5=Display      7=Rename
  8=Display description        9=Save        10=Restore     11=Move ...

Opt  Object      Type      Attribute   Text
     MLGLBLR     *PGM      RPG         Mailing list label printing
     MLGMSTU     *PGM      DFU         MLGMSTU
     USERXX      *OUTQ                 USERXX personal output queue
     MLGMSTM     *FILE     LF          Mailing list label printing logical
     MLGMSTP     *FILE     PF-DTA      Mailing master file
     MLGMSTU     *FILE     DFU         MLGMSTU
     MLGREFP     *FILE     PF-DTA      Mailing list field reference file
     MLGSRC      *FILE     PF-SRC
                                                                    Bottom
Parameters or command
===> CALL MLGLBLR
F3=Exit          F4=Prompt            F5=Refresh          F6=Create
F9=Retrieve      F10=Command entry    F23=More options    F24=More keys
Object MLGMSTL in USERXX type *FILE renamed MLGMSTM.
```

5. Notice that the file has been renamed. Run the MLGLBLR program by typing CALL MLGLBLR on the command line and pressing the Enter key.

   You should receive an error message such as the following:

```
                        Display Program Messages

Job 000287/USERXX/DSP020604 started on 05/08/89 at 15:46:21 in subsystem QBA
Error message CPF4101 appeared during OPEN (C S D).









Type reply, press Enter.
  Reply . . .


F3=Exit   F12=Cancel
```

Step 2. Debugging the Program

When a file cannot be found, you will normally get a general RPG message which states *Error message CPF4101 appeared during OPEN*.

1. To get more detailed information about this message you can move your cursor up to the line with *Error message CPF4101 . . .* on it and press F10 to look at the detailed messages.

   To reply to the message, type a C to cancel.

2. After looking at the detailed messages and finding the error, you can rename the logical file. Do this by selecting option 2 (Work with objects) on the AS/400 Programming Development Manager (PDM) display.

3. Next, specify the name of the library to work with (in this case the name is USERXX).

```
                        Work with Objects Using PDM

Library . . . . .   USERXX           Position to . . . . . .
                                     Position to type  . . . . .

Type options, press Enter.
  2=Change       3=Copy        4=Delete      5=Display      7=Rename
  8=Display description        9=Save        10=Restore     11=Move ...

Opt  Object      Type        Attribute   Text
     MLGLBLR     *PGM        RPG         Mailing list label printing
     MLGMSTU     *PGM        DFU         MLGMSTU
     USERXX      *OUTQ                   USERXX personal output queue
7    MLGMSTM     *FILE       LF          Mailing list label printing logical
     MLGMSTP     *FILE       PF-DTA      Mailing master file
     MLGMSTU     *FILE       DFU         MLGMSTU
     MLGREFP     *FILE       PF-DTA      Mailing list field reference file
     MLGSRC      *FILE       PF-SRC
                                                                      Bottom
Parameters or command
===>
F3=Exit          F4=Prompt           F5=Refresh          F6=Create
F9=Retrieve      F10=Command entry   F23=More options    F24=More keys
```

4. Type a 7 next to the MLGMSTM member name to rename the logical file.

```
                              Rename Objects

Library . . . . . :   USERXX

To rename object, type New Name, press Enter.

Object      Type        New Name
MLGMSTM     *FILE       MLGMSTL









                                                                      Bottom
F3=Exit     F5=Refresh     F12=Cancel     F19=Submit to batch
```

5. Rename the logical file to MLGMSTL and press the Enter key.

```
                        Work with Objects Using PDM

Library . . . . .   USERXX          Position to . . . . . .
                                    Position to type  . . . . .

Type options, press Enter.
  2=Change      3=Copy        4=Delete      5=Display      7=Rename
  8=Display description       9=Save        10=Restore     11=Move ...

Opt  Object      Type        Attribute   Text
     MLGLBLR     *PGM        RPG         Mailing list label printing
     MLGMSTU     *PGM        DFU         MLGMSTU
     USERXX      *OUTQ                   USERXX personal output queue
     MLGMSTL     *FILE       LF          Mailing list label printing logical
     MLGMSTP     *FILE       PF-DTA      Mailing master file
     MLGMSTU     *FILE       DFU         MLGMSTU
     MLGREFP     *FILE       PF-DTA      Mailing list field reference file
     MLGSRC      *FILE       PF-SRC
                                                                      Bottom
Parameters or command
===> CALL MLGLBLR
F3=Exit          F4=Prompt            F5=Refresh          F6=Create
F9=Retrieve      F10=Command entry    F23=More options    F24=More keys
                                                                        +
```

6. Notice that the file is renamed. After the logical file is renamed, you can check for the proper specifications by calling the mailing list label printing program as follows:

   ```
   CALL MLGLBLR
   ```

7. After the program has finished running, you can check the output queue by typing the following where USERXX is the name of your output queue.

   ```
   WRKOUTQ OUTQ(USERXX)
   ```

```
                          Work with Output Queue

Queue:   USERXX         Library:   USERXX         Status:   RLS

Type options, press Enter.
  2=Change   3=Hold   4=Delete   5=Display   6=Release   8=Attributes

Opt  File       User       User Data   Sts   Pages  Copies  Form Type   Pty
 5   QPRINT     USERXX     MLGLBLR     RDY       5       1   *STD         5









                                                                      Bottom
Parameters for option 2 or command
===>
F3=Exit   F11=View 2   F12=Previous   F21=Description   F24=More keys
```

8. Display the output queue by typing a 5 (Display) in the left column next to the QPRINT file.
9. After verifying that the output is correct, you can delete the QPRINT file by typing a 4 (Delete) in the option column next to the file on the Display Spooled File display.

Logic Errors

You can normally debug logic errors by looking at the output and finding where there are errors, then looking back at the source.

You can also use the system testing functions which are designed to help you write and maintain your application. You can run your programs in a special testing environment (called debug mode) while observing and controlling the processing of the programs in the testing environment.

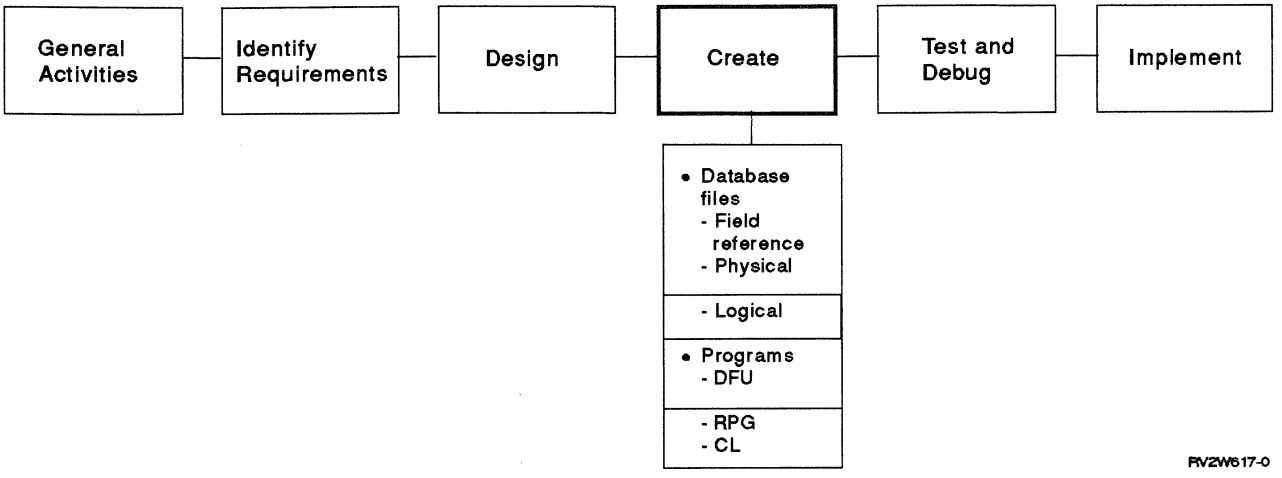
You do not have to create the program with any special options to allow the program to be debugged with the testing functions.

These testing functions are done using a set of commands that can be used interactively or in a batch job. The basic debugging commands are explained briefly here. The *CL Programmer's Guide* contains more information on testing. The *CL Reference* contains more information on the commands.

The basic functions allow you to stop at a specific point in a program, display or change variables, or trace a series of statements. The typical debug commands are:

- Start Debug (STRDBG): Places the job in debug mode.
- Add Breakpoint (ADDBKP): Allows you to have the system stop program processing and give control to you at a display station (if in interactive mode) or to a specified program (if in batch mode).
- Display Program Variable (DSPPGMVAR): Displays the current value of one or more variables.
- Change Program Variable (CHGPGMVAR): Changes the value of a variable.
- Add Trace (ADDTRC): Requests that the system record the sequence in which the statements in a program are processed.
- Display Trace Data (DSPTRCDTA): Displays the output of any traces performed.

Chapter 9. Analyzing the Database



Once you have your database set up and your name and address information entered into the system, you may need to produce different reports or to query an account. For example, you may need to produce a report that is sorted by state or by account number. You may also want to find out information about a certain account number. This chapter goes through some different ways of analyzing the database and producing various reports and also does a query of the mailing list database files. The different reports that are produced are:

- MLGRPTR: A general-purpose report that puts each account's information on one line produced by an RPG program.
- MLGRPTC: A report by state, city, and name produced by a CL program which uses a standard logical file and overrides.
- MLGRPTC2: A report in name order of a particular zip code produced by a CL program which uses a general-purpose logical file and overrides.

Overview of RPG General-Purpose Report

The first example goes through the tasks of producing a general-purpose report. One line will be printed per account. The format of the report would look like the following:

```
                          Customer Listing
Number     Name                City                State     Zip      Type
10057      George Jones        Minneapolis          MN       55454     1
11458      James Grover        Trenton              NJ       08690     1
```

This program could have been created using Query. An example of using Query is shown later.

An overview of the RPG MLGRPTR program is shown in Figure 9-1 on page 9-2.

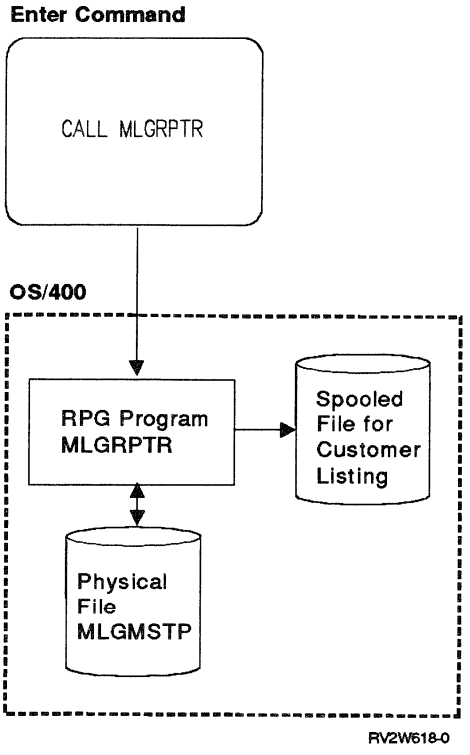


Figure 9-1. Overview of RPG Program MLGRPTR

Description of RPG General-Purpose Report Program (MLGRPTR)

The RPG specifications for the MLGRPTR program are shown in Figure 9-2 on page 9-3 and each line of the program is described following the figure.

```
      ....+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7
0001.00     F* MLGRPTR - Mailing list report format
0002.00     FMLGMSTP IF  E           K        DISK
0003.00     FQPRINT  O   F     132     OF     PRINTER
0004.00     C* Initialize and print heading
0005.00     C                    TIME           TIME    60       Time of day
0006.00     C                    EXCPTHEADNG                     Print headng
0007.00     C* Read an account record and print
0008.00     C           READ     TAG                             Read a rcd
0009.00     C                    READ MLGMSTR                  20 EOF
0010.00     C   20               GOTO ENDPGM                     If EOF
0011.00     C                    ADD  1         COUNT   70       Count rcds
0012.00     C                    EXCPTDETAIL                     Print detail
0013.00     C   OF               EXCPTHEADNG                     If overflow
0014.00     C                    GOTO READ                       Loop back
0015.00     C* End of program routine
0016.00     C           ENDPGM   TAG                             End of pgm
0017.00     C                    EXCPTTOTAL                      Print total
0018.00     C                    SETON                     LR    Set LR
0019.00     C                    RETRN                           Return
0020.00     OQPRINT  E  206          HEADNG
0021.00     O                        UDATE Y   10
0022.00     O                        TIME      20 '0 :  :  '
0023.00     O                                  60 'Customer listing'
0024.00     O                                 120 'Page'
0025.00     O                        PAGE  Z  125
0026.00     O        E  2            HEADNG
0027.00     O                                   7 'Number'
0028.00     O                                  16 'Name'
0029.00     O                                  38 'City'
0030.00     O                                  61 'State'
0031.00     O                                  69 'Zip'
0032.00     O                                  78 'Type'
0033.00     O                                  92 'Name search'
0034.00     O        E  1            DETAIL
0035.00     O                        MLACCT     6
0036.00     O                        MLNAME    32
0037.00     O                        MLCITY    54
0038.00     O                        MLSTAT    59
0039.00     O                        MLZIP     71
0040.00     O                        MLTYPE    76
0041.00     O                        MLSRCH    91
0042.00     O        E 11            TOTAL
0043.00     O                                  25 'Count of records-'
0044.00     O                        COUNT 1
```

Figure 9-2. RPG Specifications for MLGRPTR Program

Line 1.00: A comment.

Line 2.00: The MLGMSTP file is defined. The entries are similar to the label writing program. However, this file is specified as F in position 16 meaning full procedural. This is needed if operation codes will be used to process the file (such as READ).

Note: This file will not be used when running the program. An Override Database File (OVRDBF) command will be used to cause a different file (a logical file) to be opened. The program is compiled against the physical file, but an OVRDBF command is used to change the file to be opened when the program is run.

Both the physical and the logical files have the same format (they share the format described for the physical file). If this is not the case, the system will cause an exception if the file which is actually used when the program is run does not have the same definition as the one used to compile the program. See the comments in Chapter 10, "RPG Solution for Inquiry Program" about *level check*.

Line 3.00: The QPRINT standard IBM printer file is defined. This is similar to the definition used in the MLGLBLR program except that the line counter specification is not used. Instead, the form length and overflow line as specified in the QPRINT file will be used. IBM ships these values as form length = 66 and overflow line = 60. This is the normal specification for a standard form of 11 inches in depth. When your system was set up, these values may have been changed to reflect a different standard.

Lines 5.00 and 6.00: The program begins by accessing the current time which will be printed on every page to indicate the time the report was run.

Time of Day: The TIME operation can access both the system date and system time. When a 6-digit field is described as the result field, RPG retrieves the current time. Normally you will want the same value for the current time to appear on every page. Therefore, the current time is only retrieved at the beginning of the program. Rather than using the TIME operation to access the current date, the UDATE field is used in the output specifications.

The heading lines are then printed using exception output (EXCPT operation). This causes any output lines with HEADNG in positions 32 through 37 to be printed.

Lines 8.00−10.00: The READ tag identifies the label for the READ operation to the file. Indicator 20 is set when *end of file* is detected meaning there are no more records in the file. If indicator 20 is on, the program then branches to the ENDPGM label.

The program is written so that reading of the file is done by an explicit READ operation. This is contrasted with the label writing program which used a primary file and took advantage of the RPG program cycle. This program wants to print the current time on every page. This is awkward to do when using the RPG program cycle so this program is written procedurally.

Using the GOTO Statement: This program uses a GOTO statement that causes a branch. Some programmers prefer to write programs using DO and IF logic so the program is completely free of GOTO operations. Some programmers use a combination of DO and IF logic and GOTO operations.

An excessive use of GOTO operations can make program logic difficult to follow. However, an excessive use of DO and IF logic can also make program logic difficult to follow, particularly if the groups are large and no indentation exists. RPG does not support a method of indentation. You can print the RPG source in an indented manner. For example, see the PRTRPGDO tool in the QUSRTOOL library.

RPG requires the use of indicators for certain functions such as the *end of file* condition. An excessive or incorrect use of indicators can also make program logic difficult to follow.

In general, it is desirable to closely follow the operation that sets the indicator condition with the operation that tests the condition. In some cases, RPG rules or the use of indicators in DDS require that the indicator be used several statements later in the logic.

For program logic that must be tested later in a program, some programmers prefer to create a field name to better describe the condition such as:

```
MOVE 'YES' SAMTYP
```

so that later in the program they can specify:

```
SAMTYP IFEQ 'YES'
```

RPG allows up to 3 indicators to condition a statement (plus AND/OR lines). In general, it is best to avoid the use of multiple indicators to help make the program logic easier to follow.

This program uses indicator testing in positions 9 through 17.

Some programmers prefer to avoid these positions and always use an IF test to test the on/off value of an indicator such as:

```
*IN20 IFEQ '1'
```

which means *IF indicator 20 is on*.

There are no recommended solutions for the use of GOTO or indicator use other than to write for clarity and maintainability.

This manual uses a minimum amount of GOTO and indicator testing. Your installation may have standards or guidelines for how programs should be written.

Lines 11.00−14.00: The program adds 1 to the COUNT field to count the number of records which will be printed. Each record that is read is printed using exception output. If overflow has occurred (meaning the overflow line of the printer file was met or passed), the OF indicator will be set on. The EXCPT to HEADNG line is conditioned by this indicator. Skipping to a new page will set off the indicator so the program has only one statement to cause a new page heading to occur. The program then branches to read another record.

Lines 16.00−19.00: When there are no more records to be processed, the end-of-file indicator is set on by RPG and the program branches to the ENDPGM label. The final total line is printed using exception output. The program then sets on the LR indicator and does a return.

When LR is on and the program ends, it tells RPG that the program is complete. The files are then closed and the work areas associated with this use of the program (such as the COUNT field) are deleted. A subsequent call to this program will cause the work areas to be re-initialized and the files to be re-opened.

RPG defaults to open the files you have specified at the beginning of the program. You can control this with a special file entry and the OPEN and CLOSE operations. For most simple uses, you can let the RPG defaults occur.

Lines 20.00−33.00: The QPRINT entry in positions 7 through 14 specify the name of the file to be sent to the output queue. The E entry in position 15 specifies exception output. Exception output is not considered during the normal RPG program cycle. In this program, the RPG program cycle is not really used.

However, the RPG cycle still exists in the structure of any RPG program. In general, when a program is coded procedurally as is this program, the cycle is of no consequence.

Exception output lines are only considered when the EXCPT operation is used. The label in positions 32 through 37 can be used, or you can condition the line by indicators, or both. In this case, the two heading lines are identified as HEADNG and are only output when the program specifies an EXCPT to HEADNG.

The first HEADNG line prints the date, time, page number and a title. The second heading line prints column heading information. The first line is specified to skip to line 06 (the beginning of a new page) before printing (the 06 entry in positions 19 through 20 specifies *skip before*). After the line is printed, the 2 entry in position 18 tells the printer to space 2 after.

Current Date: The UDATE field name is a reserved word in RPG. It causes RPG to access the job date assigned when the job was started. This defaults to be the current system date, but can be changed.

The TIME field is edited so that it will appear with colons and the leading zero suppressed such as 9:50:00.

The PAGE field is edited with a Z edit code meaning to suppress the leading zeros. The field name PAGE is a reserved word in RPG. A 4-digit field is automatically defined and incremented by one prior to each printing.

Lines 34.00−41.00: The detail line is printed with the information from the database.

Lines 42.00−44.00: The final total line is printed when all records have been processed. A space 1 before and after are specified and the count field is printed. Notice that the COUNT field has no end position. RPG allows this kind of a specification to mean that the COUNT field should begin at the next position following the previous field.

The 1 entry in position 38 specifies the edit code to be used. This will cause the leading zeros to be suppressed, a comma to print (if the value is greater than 999), a zero balance to print (if the COUNT field is zero because no records exist in the file) and no plus or minus sign (the COUNT field must be a positive value).

The *RPG/400 User's Guide* contains more information on edit codes.

Step 1. Entering the RPG Specifications and Creating the Program (MLGRPTR)

You should now enter the RPG specifications as shown in Figure 9-2 on page 9-3 through PDM. Follow the same steps you used to create the MLGMSTP physical file in Chapter 5, "Creating Files" or copy the source from the QUSRTOOL library as described in Appendix B, "Overview of QUSRTOOL Library." The member source type should be RPG .

Once you have entered the RPG specifications, create the program by typing a 14 (Compile) in the *Opt* column next to the program on the Work with Members

Using PDM display. PDM will submit the correct CRTRPGPGM command into batch.

Note: You need to wait for the completion message before doing the next step.

Step 2. Running the RPG General-Purpose Report Program (MLGRPTR)

Now that you have created the general-purpose report, you can run the program. Run the program now by typing the following on the command entry line:

```
CALL MLGRPTR
```

Step 3. Displaying the Output

After the program has run, you can display the output. Follow these steps to look at the spooled file in your output queue.

1. Type the following on the command entry line where USERXX is the name of your output queue:

   ```
   WRKOUTQ OUTQ(USERXX)
   ```

2. Type a 5 next to the QPRINT file to display the report.

   The output should look similar to the following display. Notice that the sequence of the report is by account number. This occurs because the RPG program requested the MLGMSTP file to process in keyed sequence.

```
                              Display Spooled File
File . . . . . :   QPRINT                          Page/Line   1/6
Control . . . . .                                  Columns     1 - 78
Find . . . . . .
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+...
  5/09/89   8:47:59                    Customer listing
 Number     Name                 City                 State     Zip      Type
 10057      Samuel Jones         Minneapolis           MN       55454     1
 11458      James Grover         Trenton               NJ       08690     1
 14477      Charles Hanley       Rochester             MN       55920     4
 15902      Joseph Jones         Little Rock           AR       44877     2
 18890      Carol Larson         Rochester             MN       55920     5
 24882      Kathryn Donty        Dallas                TX       75248     1
 26640      Daniel Benson        Syracuse              NY       13212     1
 28903      Philip Jones         San Diego             CA       66903     9
 38724      Maria Jonesa         Philadelphia          PA       22809     5
        Count of records-        9


                                                                    Bottom
F3=Exit   F12=Cancel   F19=Left   F20=Right   F24=More keys
```

3. Press F3 (Exit).

4. Delete all of the spooled files from the output queue by typing the following command on the command entry line where USERXX is the name of your output queue:

   ```
   CLROUTQ OUTQ(USERXX)
   ```

   You could also select option 4 for the files you want to delete if you do not want to delete all files in the output queue.

Overview of a Report Using a Standard Logical File and Overrides

This example goes through using a standard logical file and overrides to produce a report by state, city, and name. A logical file specified as MLGMSTL2 is used. There are other solutions to selecting and sequencing records that will be described later.

The relationship of the MLGMSTL2 logical file to the master physical file (MLGMSTP), logical file (MLGMLSTL) and the RPG program (MLGRPTR) are shown in Figure 9-3.

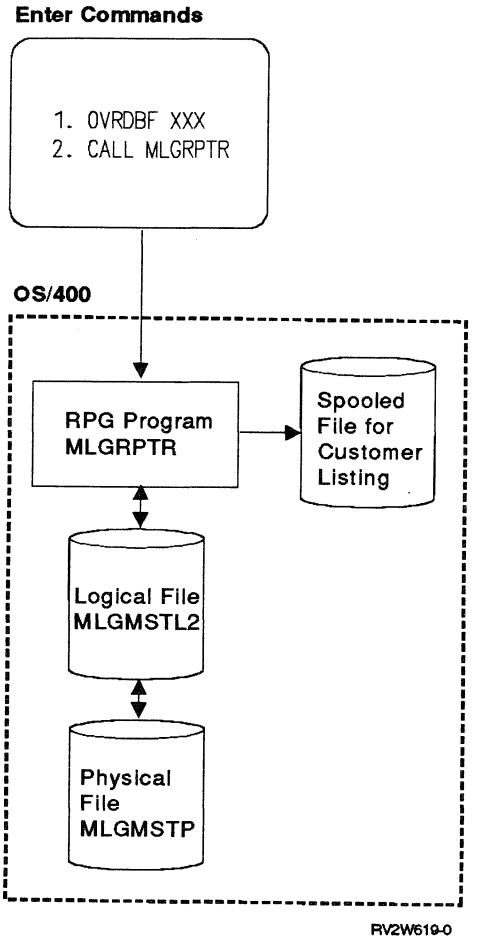


Figure 9-3. Overview of MLGMSTL2 Logical File

Description of DDS for Standard Logical File (MLGMSTL2)

Figure 9-4 shows the DDS for the MLGMSTL2 logical file and a description of each line follows the figure.

```
        ....+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7
0001.00      A* MLGMSTL2 - Mailing list by state, city, name
0002.00      A          R MLGMSTR                     PFILE(MLGMSTP)
0003.00      A          K MLSTAT
0004.00      A          K MLCITY
0005.00      A          K MLSRCH
```

Figure 9-4. DDS Specifications for MLGMSTL2 Logical File

Line 1.00: A comment.

Line 2.00: This is the same definition as MLGMSTL. The physical file MLGMSTP and the format of the physical file (MLGMSTR) will be used.

Logical File Formats: When a logical file is defined, you must define a format to be used. The following choices exist:

- The same format as used in the physical file (as in this case).
- A new format. A format name and the fields that make up the format name must be described. For example, you might want the format to contain only a subset of the fields from the physical file. The purpose of this type of approach is normally to provide more data independence. See the *Database Guide* for a more complete discussion.
- An existing format in a different file. For example, you may have already created a new format in a different logical file and want to re-use it. See the FORMAT keyword for this type of specification.

Lines 3.00−5.00: The key fields for the file are described. Each has a K in position 17. The high-order key is the state field, followed by city, and then the search field.

Step 1. Entering the DDS and Creating the Standard Logical File (MLGMSTL2)

You should now enter the DDS as shown in Figure 9-4 on page 9-8 through PDM. Follow the same steps you used to create the MLGMSTP physical file in Chapter 5, "Creating Files" or copy the source from the QUSRTOOL library as described in Appendix B, "Overview of QUSRTOOL Library." The member source type should be LF .

Once you have entered the DDS, create the file by typing a 14 (Compile) in the *Opt* column next to the file on the Work with Members Using PDM display.

Step 2. Running the RPG Program with an Override (MLGRPTR)

Once the logical file has been created and you have received the completion message, you can specify an override and run the MLGRPTR program again by following these steps:

1. Specify the override by typing the following:

   ```
   OVRDBF FILE(MLGMSTP) TOFILE(MLGMSTL2)
   ```

   The override command specifies that any program in this job that requests to open the file MLGMSTP will actually open the MLGMSTL2 file. An override command can be used for this type of function or to change the attributes of the file being opened.

2. Next, run the MLGRPTR program by typing the following:

   ```
   CALL MLGRPTR
   ```

   The program requests that MLGMSTP be opened. However, the override changes the request so that the MLGMSTL2 file will be used instead.

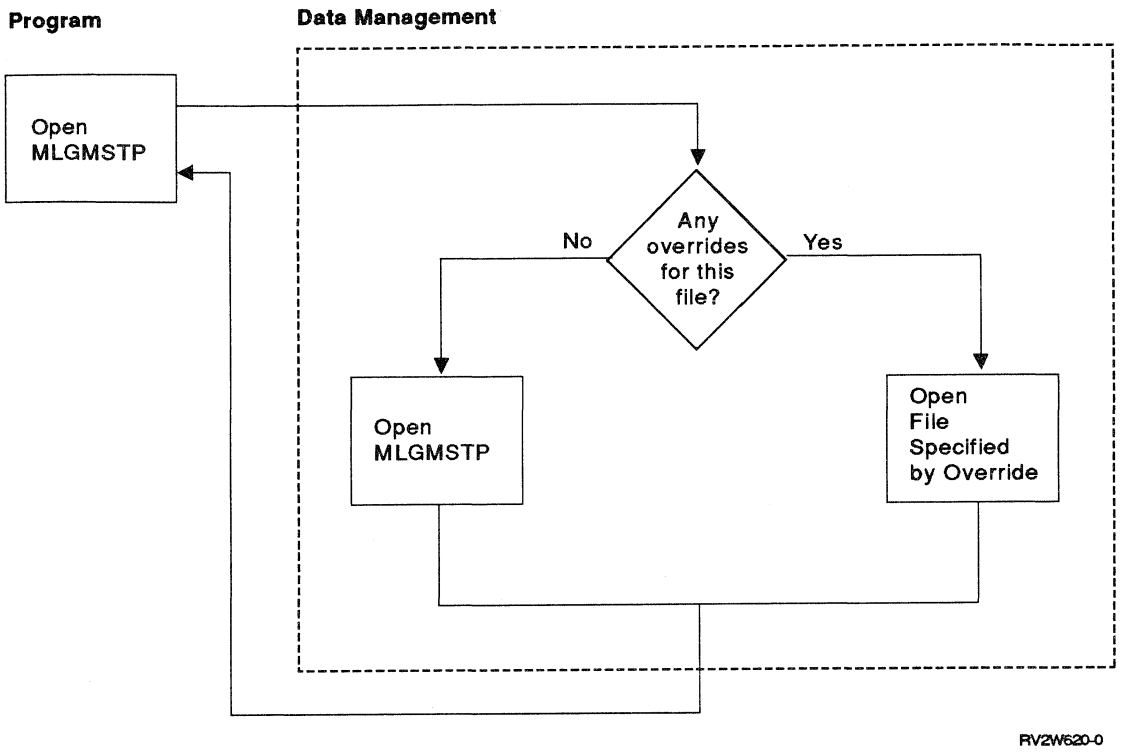


Figure 9-5. Overview of Using an Override

The system data management function performs the file open. The program is not aware that a different file is being used.

3. Look at the results of the program by displaying your QPRINT spooled file using the WRKOUTQ command. The sequence should appear by search field within city within state. The search field is not printed. This is the order of the access path of the MLGMSTL2 logical file used. The report should look similar to the following display.

```
                              Display Spooled File
File . . . . . :   QPRINT                          Page/Line   1/6
Control . . . . .                                  Columns     1 - 78
Find . . . . . .
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+...
  6/20/89  14:49:03                    Customer listing
 Number     Name                 City                 State     Zip      Type
 15902      Joseph Jones         Little Rock           AR       44877    2
 28903      Philip Jones         San Diego             CA       66903    9
 10057      Samuel Jones         Minneapolis           MN       55454    1
 14477      Charles Hanley       Rochester             MN       55920    4
 18890      Carol Larson         Rochester             MN       55920    5
 11458      James Grover         Trenton               NJ       08690    1
 26640      Daniel Benson        Syracuse              NY       13212    1
 38724      Maria Jonesa         Philadelphia          PA       22809    5
 24882      Kathryn Donty        Dallas                TX       75248    1
        Count of records-       9


                                                                    Bottom
F3=Exit    F12=Cancel    F19=Left    F20=Right    F24=More keys
```

4. You should either delete the spooled files you just created in your output queue (option 4 next to files) or clear the output queue (using CLROUTQ command).
5. The override you specified for the MLGMSTP file will remain in effect for your job until you delete it, change it, or until you press F3 on the display from which you entered the OVRxxx. You can see what overrides are in effect by specifying DSPJOB and using option 15 to display the overrides.

   To avoid using the override when you do not intend to, you should delete the override by using the Delete Override (DLTOVR) command as follows:

   ```
   DLTOVR FILE(MLGMSTP)
   ```

Access Path Maintenance Options

The default when a logical file is created causes the system to keep the access path up to date whenever a new record is added or changed in the file. Because this access path will only be used when a specific report is run, there is a performance tradeoff to be made. You can either cause a little overhead to maintain the access path each time a record is added or deleted or you can take away the overhead and cause the system to rebuild the entire access path each time the file is opened. Normally, you would make a decision based on the frequency of use of the file and when the file is opened. For example, if the open operation causes a rebuild at night when there is unused system time, no one may care about the rebuild time.

For this application example, we will assume that all logical files will use immediate maintenance. In a real application, this would normally not be done.

Some different alternatives for selecting and sequencing records are described in Chapter 15, "Additional Topics."

Overview of a Report Using a CL Program for Overrides

Instead of having to manually specify and delete overrides whenever you want to run this report, you could write a CL program to do it for you. CL programs allow you to specify a series of commands to be run by a single CALL command. CL programs are normally used to control the running of one or more high-level language programs. In most cases, the same command that you use interactively can also be placed in a CL program. There are also special commands that can only be used within a CL program. Another benefit of using a CL program is that an override is only active during the program and is automatically deleted at the end of the program.

The MLGRPTC program is a simple CL program that contains the override and calls the report printing program. Figure 9-6 shows an overview of the MLGRPTC program.

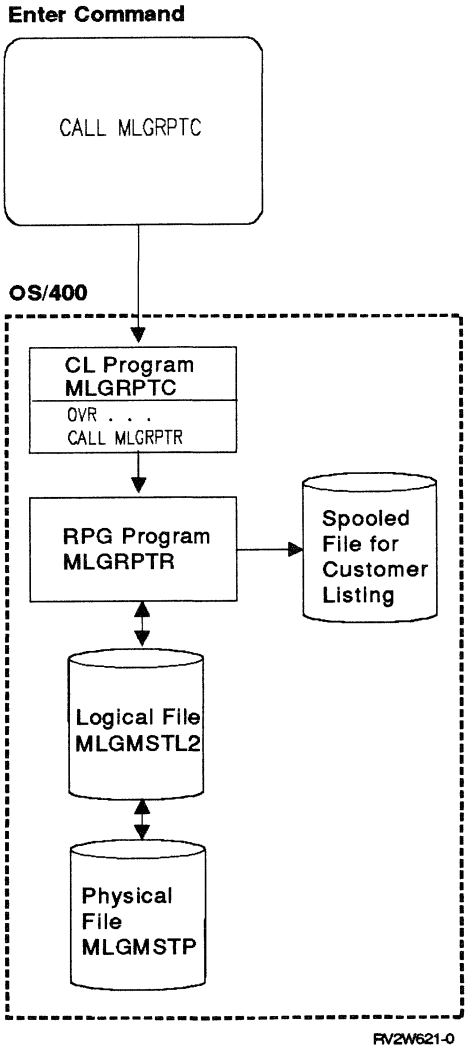


Figure 9-6. Overview of CL Program MLGRPTC with Overrides

Description of CL Program (MLGRPTC)

Figure 9-7 shows the MLGRPTC program and a description of each line follows the figure.

```
        ...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7
0001.00 /* MLGRPTC - Print one liner with MLGMSTL2 LF                    */
0002.00             PGM
0003.00             OVRDBF     MLGMSTP TOFILE(MLGMSTL2)
0004.00             CALL       MLGRPTR
0005.00             ENDPGM
```

Figure 9-7. MLGRPTC Program

Line 1.00: A comment is defined to exist between the '/*' and the '*/'.

Line 2.00: The PGM statement specifies the beginning of the program. The PGM statement has an optional PARM parameter to identify parameters being passed into the program. This program is called with no parameters passed in.

Line 3.00: The OVRDBF command overrides the MLGMSTP file which is specified in the program to use the MLGMSTL2 logical file. This is the same command that was entered in the previous discussion. The MLGMSTL2 file is keyed by state, city, and search field so the records will be presented to the program in that sequence.

Lines 4.00—5.00: The MLGRPTR program is called. When the program returns, the ENDPGM statement is run which ends the CL program normally.

Step 1. Entering and Creating the CL Override Program (MLGRPTC)

You should now enter the CL coding as shown in Figure 9-7 through PDM. Follow the same steps you used to create the MLGMSTP physical file in Chapter 5, "Creating Files" or copy the source from the QUSRTOOL library as described in Appendix B, "Overview of QUSRTOOL Library." The member source type should be CLP.

Once you have entered the CL coding, create the program by typing a 14 (Compile) in the *Opt* column next to the program on the Work with Members Using PDM display.

Step 2. Running the CL Override Program (MLGRPTC)

When you have received the message that the program was successfully created, run the program by typing the following command:

```
CALL MLGRPTC
```

Look at the results of the report in your output queue. You should see the same results as when you entered the command interactively. Delete the spooled files when done.

Overview of a General-Purpose Logical File Report

The RPG program MLGRPTR is a general-purpose report printing program. You can precede it with a general-purpose logical file.

For example, assume you get a special request to print a listing in name search order of a particular zip code. You could have a general-purpose logical file that you change for these types of requests.

The following discussion goes through the steps of creating a general-purpose logical file, and creating a CL program to cause the override in a manner similar to the MLGRPTC program.

Description of DDS for General-Purpose Logical File (MLGMSTL3)

Figure 9-8 shows the DDS for the MLGMSTL3 logical file and a description of each line follows.

```
          ...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7
0001.00       A* MLGMSTL3 - General purpose LF for querying MLGMSTP
0002.00       A          R MLGMSTR                   PFILE(MLGMSTP)
0003.00       A          K MLSRCH
0004.00       A          S MLZIP                     COMP(EQ 55920)
```

Figure 9-8. DDS for MLGMSTL3 Logical File

Line 1.00: A comment.

Line 2.00: This is the same definition as MLGMSTL.

Line 3.00: The key field is defined to be on the search field MLSRCH.

Line 4.00: The S entry in position 17 indicates that this is a select statement. The first S entry indicates the end of the key field section of the specifications. There can be multiple S entries and also O entries (omit). The S for select entry specifies that the record in the physical file will only be included in the access path if it meets the specified values.

You can specify a combination of multiple select and omit statements. See the *DDS Reference* for more information.

In this case, the MLZIP field is being compared for an equal value of 55920. Only the records that match 55920 will be included in the access path.

Step 1. Creating the General-Purpose Logical File (MLGMLSTL3)

You should now enter the DDS as shown in Figure 9-8 through PDM. Follow the same steps you used to create the MLGMSTP physical file in Chapter 5, "Creating Files" or copy the source from the QUSRTOOL library as described in Appendix B, "Overview of QUSRTOOL Library." The member source type should be LF .

Once you have entered the DDS, create the file by typing a 14 (Compile) in the *Opt* column next to the file on the Work with Members Using PDM display.

Description of CL Program (MLGRPTC2)

Once you get the completion message that the MLGMSTL3 logical file has successfully been created, you can create a CL program to run the override and call the MLGRPTR program. This program also specifies a different name for the printer file and sends a special message. Figure 9-9 shows the program and each line is described following the figure.

```
        ...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7
0001.00 /* MLGRPTC2 - General purpose query of MLGMSTP                    */
0002.00             PGM
0003.00             OVRDBF     MLGMSTP TOFILE(MLGMSTL3)
0004.00             OVRPRTF    QPRINT SPLFNAME(MLGRPT)
0005.00             CALL       MLGRPTR
0006.00             SNDPGMMSG  MSG('Special report printed to MLGRPT') +
0007.00                          MSGTYPE(*COMP)
0008.00             ENDPGM
```

Figure 9-9. MLGRPTC2 CL Program

Lines 1.00—2.00: Same as MLGRPTC.

Line 3.00: The OVRDBF command overrides the MLGMSTP file which is specified in the program to use the file MLGMSTL3. This file is keyed by the search field and does selection based on a zip code.

Line 4.00: The OVRPRTF command overrides the QPRINT file to specify that the name of the spooled file should not be QPRINT, but rather MLGRPT. The SPLFNAME does not cause a different file to be opened, but instead changes the name of the file being sent to the output queue. You can specify what name you want the spooled file to be as an assistance to the user who is printing spooled files. If you do not have an override, the QPRINT name will appear as the spooled file name.

Notice the difference between the SPLFNAME parameter on the OVRPRTF command and the TOFILE parameter on the OVRDBF command. For the database file, you are specifying that the file in the program should not be used. Instead, a different file (MLGMSTL3) should be used.

The OVRPRTF command does not specify to use a different file. It specifies that an attribute of the file should be overridden for this use. (The file is not permanently changed.) The attribute that is being overridden is the name of the spooled file that will be created. The program will open and use the QPRINT file.

Override Options: There are three things you can do with an override:

- Override an attribute of the existing file (for example, SPLFNAME)
- Override to a different file (for example, use the logical file instead of the physical file)
- Override to a different file and specify a special attribute of the new file (this type has not been used)

Line 5.00: The MLGRPTR program is called.

Lines 6.00 and 7.00: The Send Program Message (SNDPGMMSG) command sends a program message. If you are operating from a command entry display, the message that is sent will appear on the display. Messages are helpful in providing feedback as to what happened. In this case, the message is specified as MSGTYPE(*COMP) meaning it is a completion message.

Line 8.00: The End Program (ENDPGM) command ends the program normally.

Step 2. Entering and Creating the CL Program (MLGRPTC2)

You should now enter the CL coding as shown in Figure 9-9 on page 9-15 through PDM. Follow the same steps you used to create the MLGMSTP physical file in Chapter 5, "Creating Files" or copy the source from the QUSRTOOL library as described in Appendix B, "Overview of QUSRTOOL Library." The member source type should be CLP .

Once you have entered the DDS, create the program by typing a 14 (Compile) in the *Opt* column next to the program on the Work with Members Using PDM display.

Step 3. Running the CL Program (MLGRPTC2)

Follow these steps to run the MLGRPTC2 program once you have received the completion message that the CL program was successfully created.

1. Type the following command on the command entry line:

   ```
   CALL MLGRPTC2
   ```

   Notice that you receive a message stating the name of the spooled file that was sent to the output queue specified in the CL program as follows:

   ```
   Special report printed to MLGRPT
   ```

2. Use the Work with Output Queue (WRKOUTQ OUTQ(USERXX)) command to look at the report.

3. Type a 5 in the *Opt* column next to the file to display the report. Notice that the spooled file name is MLGRPT and not QPRINT because of the OVRPRTF command.

   The format of the report should look like the following display.

```
                              Display Spooled File
File . . . . . :   MLGRPT                          Page/Line   1/6
Control . . . . .                                  Columns     1 - 78
Find . . . . . .
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+...
   5/09/89   9:06:18                      Customer listing
 Number     Name                City                State     Zip      Type
 14477      Charles Hanley      Rochester             MN      55920     4
 18890      Carol Larson        Rochester             MN      55920     5
        Count of records-       2
                                                                      Bottom
F3=Exit    F12=Cancel    F19=Left    F20=Right    F24=More keys
```

4. Press F3 (Exit).
5. Delete the spooled files in the output queue.

Additional Special Requests

The MLGMSTL3 file can be changed and re-created whenever you have a special request. As long as the request can be answered by using the standard report printing program, you only need to:

- Change the DDS for MLGMSTL3
- Create the MLGMSTL3 file
- Call the MLGRPTC2 program

For example, you might try to simulate a special request for all those accounts that are type 1 (Business) in Minnesota. You could delete statement 4.00 in the MLGMSTL3 file and add the following:

```
S MLTYPE  COMP(EQ '1')
S MLSTAT  COMP(EQ 'MN')
```

Overview of AS/400 Query

You can use the AS/400 Query licensed program to select, arrange, and analyze the information from the files you have created to produce reports and other data files. You determine what data the query is to retrieve, the format of the report, and whether it should be displayed, printed, or sent to another database file.

The rest of this chapter goes through an example of using AS/400 Query to produce a report sorted by type, state, and search field.

Example of Querying the Database

Follow these steps to use AS/400 Query to produce a report.

1. First, start Query by using the Start Query (STRQRY) command as follows:

   STRQRY

```
QUERY                                Query
                                                    System:   RCH38342
Select one of the following:

     1. Work with queries
     2. Run an existing query
     3. Delete a query

    20. Files
    21. Office tasks









Selection or command
===> 1

F3=Exit   F4=Prompt   F9=Retrieve   F12=Cancel   F13=User support
F16=System main menu
                                        (C) COPYRIGHT IBM CORP. 1980, 1989.
```

2. Select option 1 (Work with queries).

```
                          Work with Queries

Type choices, press Enter.

  Option . . . . . .  1             1=Create, 2=Change, 3=Copy, 4=Delete
                                    5=Display, 6=Print definition
                                    9=Run
  Query . . . . . .   MLGMSTQ       Name, F4 for list
    Library . . . . .   USERXX      Name, *LIBL, F4 for list









F3=Exit       F4=Prompt       F5=Refresh       F12=Cancel
                                                (C) COPYRIGHT IBM CORP. 1988
```

3. Select option 1 (Create) and name the query MLGMSTQ as shown above. Press the Enter key.

   The Define the Query display is shown.

```
                            Define the Query

Query . . . . . :   MLGMSTQ          Option . . . . . :   CREATE
  Library . . . . :     USERXX

Type options, press Enter.  Press F21 to select all.
  1=Select

Opt    Query Definition Option
 1     Specify file selections
       Define result fields
 1     Select and sequence fields
       Select records
 1     Select sort fields
       Select collating sequence
       Specify report column formatting
       Select report summary functions
       Define report breaks
       Select output type and output form
       Specify processing options

F3=Exit            F5=Report          F12=Cancel
F13=Layout         F18=Files          F21=Select all
```

4. This display allows you to select any of the definition steps (options) needed to define your query. The Specify file selections is defaulted to option 1 (Select) because it is required.

   Type a 1 in the *Opt* column next to Select and sequence fields and Select sort fields. These definition options allow you to select and to sequence the fields that you want to appear on your report and also to select what fields you want sorted.

```
                        Specify File Selections

Type choices, press Enter.  Press F9 to specify an additional
  file selection.

  File . . . . . .   MLGMSTP       Name, F4 for list
    Library  . . . .   USERXX       Name, *LIBL, F4 for list
  Member . . . . . .   *FIRST       Name, *FIRST, F4 for list
  Format . . . . . .   *FIRST       Name, *FIRST, F4 for list

F3=Exit          F4=Prompt        F5=Report          F9=Add file
F12=Cancel       F13=Layout       F24=More keys
```

5. On the Specify File Selections display, fill in the name of the file you want to query (MLGMSTP) as shown above and press the Enter key.

```
                        Specify File Selections

Type choices, press Enter.  Press F9 to specify an additional
  file selection.

  File . . . . . .   MLGMSTP       Name, F4 for list
    Library  . . . .   USERXX       Name, *LIBL, F4 for list
  Member . . . . . .   *FIRST       Name, *FIRST, F4 for list
  Format . . . . . .   MLGMSTR      Name, *FIRST, F4 for list

F3=Exit          F4=Prompt        F5=Report          F9=Add file
F12=Cancel       F13=Layout       F24=More keys
Select file(s), or press Enter to confirm.
```

Press the Enter key to confirm your selection.

```
                        Select and Sequence Fields

Type sequence number (0-9999) for the names of up to 500 fields to
  appear in the report, press Enter.

Seq   Field
4     MLACCT
1     MLTYPE
5     MLNAME
3     MLSRCH
      MLADDR
      MLCITY
2     MLSTAT
6     MLZIP

                                                                  Bottom
F3=Exit          F5=Report          F11=Display text      F12=Cancel
F13=Layout       F20=Renumber       F21=Select all        F24=More keys
```

6. The Select and Sequence Fields display shows you the fields contained in file MLGMSTP and allows you to select and to sequence the fields that you want to appear in your report.

   Sort on MLTYPE, MLSTAT, and MLNAME in the order shown and press the Enter key.

```
                        Select and Sequence Fields

Type sequence number (0-9999) for the names of up to 500 fields to
  appear in the report, press Enter.

Seq   Field
   1  MLTYPE
   2  MLSTAT
   3  MLSRCH
   4  MLACCT
   5  MLNAME
   6  MLZIP
      MLADDR
      MLCITY

                                                                  Bottom
F3=Exit          F5=Report          F11=Display text      F12=Cancel
F13=Layout       F20=Renumber       F21=Select all        F24=More keys
Press Enter to confirm.
```

7. Verify your sequence and selections and press the Enter key to confirm.

```
                          Select Sort Fields

Type sort priority (0-999) and A (Ascending) or D (Descending) for
  the names of up to 32 fields, press Enter.

Sort
Prty A/D  Field
 1    A   MLTYPE
 2    A   MLSTAT
 3    A   MLSRCH
          MLACCT
          MLNAME
          MLZIP

                                                                  Bottom
F3=Exit          F5=Report        F11=Display text     F12=Cancel
F13=Layout       F18=Files        F20=Renumber         F24=More keys
```

8. The Select Sort Fields display shows you the fields you selected on the Select and Sequence Fields display. You can now sort these fields as shown above.

```
                          Select Sort Fields

Type sort priority (0-999) and A (Ascending) or D (Descending) for
  the names of up to 32 fields, press Enter.

Sort
Prty A/D  Field
   1  A   MLTYPE
   2  A   MLSTAT
   3  A   MLSRCH
          MLACCT
          MLNAME
          MLZIP

                                                                  Bottom
F3=Exit          F5=Report        F11=Display text     F12=Cancel
F13=Layout       F18=Files        F20=Renumber         F24=More keys
Press Enter to confirm.
```

9. Verify your sorting sequence and press the Enter key to confirm.

   You return to the Define the Query display where > symbols designate the definition values that are now different from the system-supplied (default) values.

```
                              Define the Query

Query . . . . . :    MLGMSTQ          Option  . . . . . :   CREATE
  Library . . . . :     USERXX

Type options, press Enter.  Press F21 to select all.
  1=Select

Opt    Query Definition Option
     > Specify file selections
       Define result fields
     > Select and sequence fields
       Select records
     > Select sort fields
       Select collating sequence
       Specify report column formatting
       Select report summary functions
       Define report breaks
       Select output type and output form
       Specify processing options

F3=Exit            F5=Report
F13=Layout         F18=Files          F21=Select all
Select options, or press F3 to save or run the query.
```

10. Press F3 to save the changes.

```
                              Exit this Query

Type choices, press Enter.

  Save definition  . . .   Y             Y=Yes, N=No

  Run this query . . . .   Y             Y=Yes, N=No


  For a saved definition:
    Query  . . . . . .     MLGMSTQ       Name
      Library  . . . .       USERXX      Name, F4 for list

    Text . . . . . .

    Authority  . . . . .   *CHANGE       *CHANGE, *ALL, *EXCLUDE, *USE
                                         authorization list name



F4=Prompt        F5=Report        F13=Layout        F14=Define the query
```

11. On the Exit This Query display, press the Enter key to save the definition and run the query.
12. The results of running the query are shown on the Display Report display as shown.

```
                              Display Report
Query . . . :   USERXX/MLGMSTQ             Report width . . . . . :      61
Position to line  . . . . .                Shift to column  . . . . . .
Line    ....+....1....+....2....+....3....+....4....+....5....+....6.
       Type  State  Search      Account  Name                      ZIP
                                number                            code
000001  1     MN    JONES        10057   Samuel Jones             55454
000002  1     NJ    GROVER       11458   James Grover             08690
000003  1     NY    BENSON       26640   Daniel Benson            13212
000004  1     TX    DONTY        24882   Kathryn Donty            75248
000005  2     AR    JONES        15902   Joseph Jones             44877
000006  4     MN    HANLEY       14477   Charles Hanley           55920
000007  5     MN    LARSON       18890   Carol Larson             55920
000008  5     PA    JONESA       38724   Maria Jonesa             22809
000009  9     CA    JONES        28903   Philip Jones             66903
****** ********  End of report  ********
                                                                  Bottom
F3=Exit      F12=Cancel      F19=Left      F20=Right      F21=Split
```

13. Look at the results and then Press F3 (Exit).

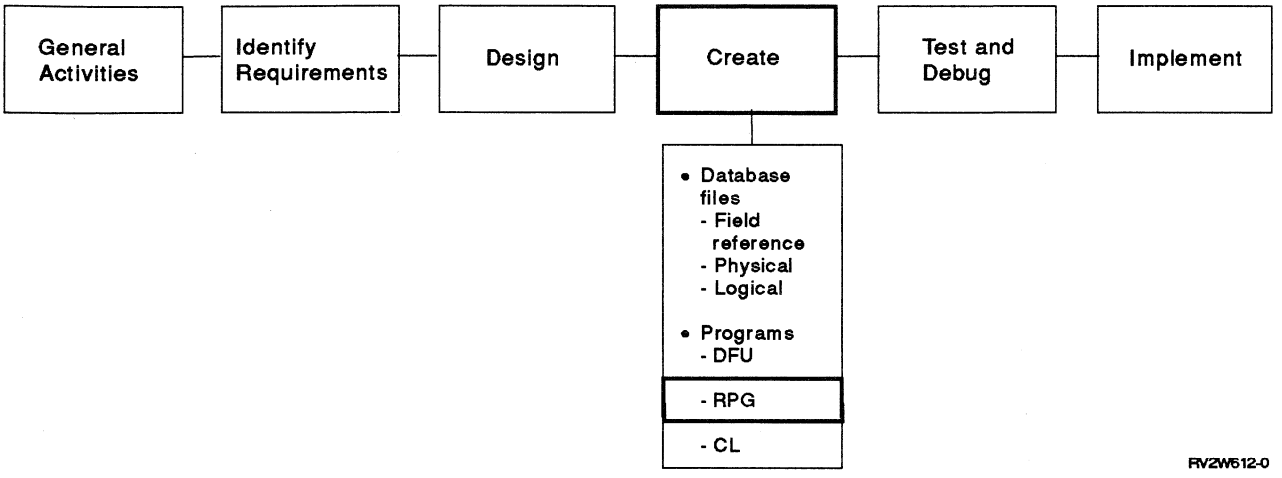
```
                            Work with Queries

Type choices, press Enter.

  Option  . . . . . .                1=Create, 2=Change, 3=Copy, 4=Delete
                                     5=Display, 6=Print definition
                                     9=Run
  Query . . . . . .   MLGMSTQ        Name, F4 for list
    Library . . . . .   USERXX       Name, *LIBL, F4 for list









F3=Exit       F4=Prompt       F5=Refresh       F12=Cancel
Query option processing completed successfully.
```

14. On the Work with Queries display, you should receive a message that the query completed successfully. Press F3 to exit.

This example only went through querying the database and producing one report. AS/400 Query can be used to produce all kinds of reports without writing programs, creating logical files and so on. You can save a query for later use (as was done in this example) or create a temporary query. The *Query/400 User's Guide* contains more information on using AS/400 Query.

A query is normally used to produce a report. While it was used interactively in this example, the normal use would be to do report work in batch. A query is normally too slow to be used for a specific request for a particular record. This type of request is normally called an *inquiry*. The next chapter describes an inquiry example.

Chapter 10. RPG Solution for Inquiry Program



This chapter describes an RPG solution for creating an inquiry program that allows you to display information on an account by entering an account number.

In many applications, it is sufficient to have only a maintenance program that performs both the inquiry and maintenance function. However, a separate inquiry program will be used for the following reasons:

- It provides a simple solution for allowing some users to inquire only (not update). One of the security objectives of the application is to allow only certain users to inquire without being able to change the data.
- Because maintenance programs are usually complex, a simple inquiry program is used to make it easier to understand the basic functions required in any interactive program.

  This inquiry function could have been created using DFU. DDS and RPG are used in this example to help introduce the interactive coding techniques.

Overview of Mailing List Inquiry Program

The mailing list inquiry program is a simple RPG program that displays a single record from the MLGMSTP file. The user calls the program and is presented with displays as shown in Figure 10-1.

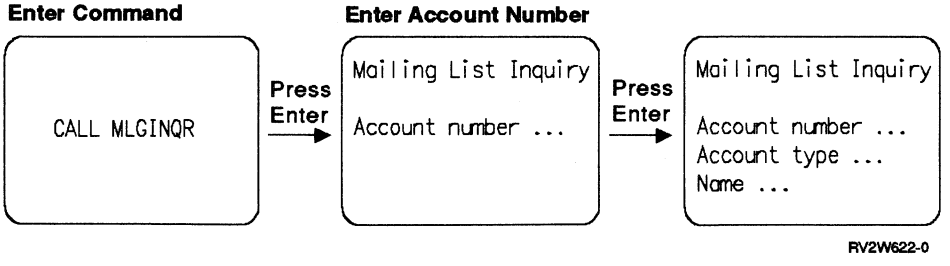


Figure 10-1. Overview of Inquiry Program Displays

The mailing list inquiry prompt is presented to the user as follows:

```
                    Mailing List Inquiry

Account number  . . . . . : 10057


F3=Exit
```

The user types in an account number, for example 10057, presses the Enter key, and is presented with the following display:

```
                    Mailing List Inquiry

Account number  . . . . . :  10057
Account type  . . . . . :  1
Name  . . . . . :  Samuel Jones
Search name . . . . . :  JONES
Address . . . . . :  220 4th Ave NW
City  . . . . . :  Minneapolis
State . . . . . :  MN
Zip code  . . . . . :  55454


F3=Exit
```

While the function performed is very simple, the coding demonstrates some of the basic specifications needed in order to write interactive programs on the AS/400 system.

The structure of the inquiry program consists of the following:

- A display file, MLGINQD that describes the display formats used by the MLGINQR program.
- An RPG program, MLGINQR, that performs the inquiry keyed on the account number.

Figure 10-2 on page 10-3 shows an overview of the RPG program MLGINQR.

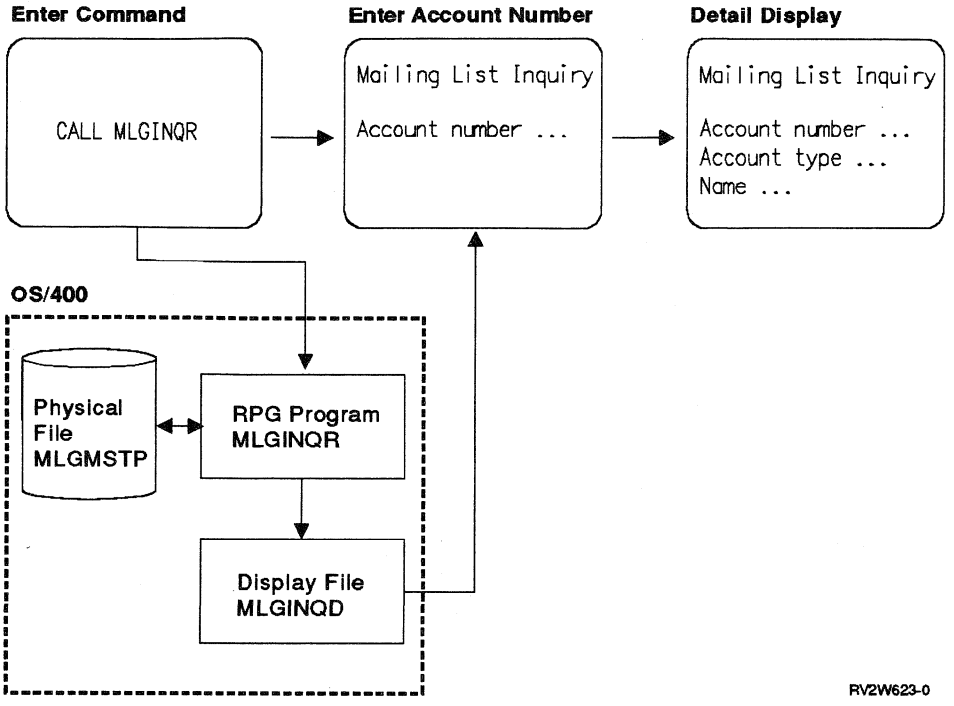


Figure 10-2. Overview of RPG Inquiry Program MLGINQR

Description of DDS for Inquiry Display File (MLGINQD)

To do a work station function on the system, you would normally do one of the following:

- Code display file DDS.
- Use screen design aid (SDA) which is part of the Application Development tools package. SDA offers an interactive solution to screen design. The output of SDA is DDS which can be used to automatically create a display file.

To best understand SDA, it is important to know the type of DDS statements used to create a display file. For this reason, the following examples will show the DDS coding. Chapter 14, "Creating a Display Using SDA" shows an example of how SDA can be used to design the display file that is used for this inquiry function.

The display file DDS has the same format as that used previously in the DDS for physical and logical files, but has different functions. Figure 10-3 on page 10-4 shows the DDS for the MLGINQD display file. A description of each line follows the figure.

```
        ...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7
0001.00     A* MLGINQD - Mailing List Inquiry - used by MLGINQR
0002.00     A                                      PRINT
0003.00     A                                      REF(MLGREFP)
0004.00     A          R DSPLY1                    TEXT('Prompt for account +
0005.00     A                                      number')
0006.00     A                                      CA03(93 'Exit')
0007.00     A                                    1 25'Mailing List Inquiry'
0008.00     A                                      DSPATR(HI)
0009.00     A                                    3  2'Account number  . . . . +
0010.00     A                                       . . . :'
0011.00     A            ACCT      R     D  I    +1REFFLD(MLACCT)
0012.00     A  41                                  ERRMSG('The account +
0013.00     A                                      number cannot be zeros' 41)
0014.00     A  42                                  ERRMSG('The account +
0015.00     A                                      number does not exist' 42)
0016.00     A                                   24  2'F3=Exit'
0017.00     A          R DSPLY2                    TEXT('MLGMSTP record +
0018.00     A                                      display')
0019.00     A                                      CA03(93 'Exit')
0020.00     A                                    1 25'Mailing List Inquiry'
0021.00     A                                      DSPATR(HI)
0022.00     A                                    3  2'Account number  . . . . +
0023.00     A                                       . . . :'
0024.00     A            MLACCT    R             +3
0025.00     A                                    4  2'Account type  . . . . . +
0026.00     A                                       . . . :'
0027.00     A            MLTYPE    R             +3
0028.00     A                                    5  2'Name  . . . . . . . . . +
0029.00     A                                       . . . :'
0030.00     A            MLNAME    R             +3
0031.00     A                                    6  2'Search name . . . . . . +
0032.00     A                                       . . . :'
0033.00     A            MLSRCH    R             +3
0034.00     A                                    7  2'Address . . . . . . . . +
0035.00     A                                       . . . :'
0036.00     A            MLADDR    R             +3
0037.00     A                                    8  2'City  . . . . . . . . . +
0038.00     A                                       . . . :'
0039.00     A            MLCITY    R             +3
0040.00     A                                    9  2'State . . . . . . . . . +
0041.00     A                                       . . . :'
0042.00     A            MLSTAT    R             +3
0043.00     A                                   10  2'Zip code  . . . . . . . +
0044.00     A                                       . . . :'
0045.00     A            MLZIP     R             +3
0046.00     A                                   24  2'F3=Exit'
```

Figure 10-3. Description of DDS for Display File MLGINQD

Line 1.00: A comment which is indicated by the * in position 7.

Line 2.00: The PRINT keyword defines that the end user is allowed to press the Print key on the keyboard when the screen is displayed. Without this keyword, pressing the Print key will cause an error message to be shown on this display. If the Print key is pressed, a spooled file will be printed by the system for what appears on the display.

Line 3.00: The REF keyword states that the MLGREFP file will be used as a reference for the later statements. This is the same statement that was used to create the physical file MLGMSTP.

Line 4.00: The first record format is described in the work station file. There are two record formats which are identified by the R in position 17. Each record format must have a unique name. The first is DSPLY1 and is used to describe the first screen (the prompt for the account number). The TEXT keyword provides a text description for the record format (it is not used by the system).

Line 6.00: The CA03 keyword describes that function key 3 will be valid for the user to press.

Function Keys: Any other function keys which are not described and are pressed will cause an error message to be shown to the user. The remainder of the CA03 keyword describes that indicator 93 will be set on if F3 is pressed. This indicator will be passed back to the RPG program. If F3 is not pressed, indicator 93 will be automatically set off. The text *Exit* is used to help describe indicator 93.

Lines 7.00 and 8.00: A constant will appear on line 1 of the display and begin at position 25. The keyword DSPATR and the value HI indicate that the constant described should appear in high intensity. Normally, you want to use high intensity on the display to bring attention to certain fields such as this one which is the title of the display.

Lines 9.00−15.00: The constant *Account number . . .* will appear on line 3 beginning in position 2. A DDS statement may overflow one line to the next by use of the + symbol. The first variable field is described and given the name of ACCT. The REFFLD keyword specifies that the field should have the same characteristics as the MLACCT field described in the field reference file (defined on line 3.00).

Digits-Only Field: The D in position 35 for *data type* specifies that this will be a *digits-only* field. This means that the input field length of the display can only be digits 0 through 9.

The default for data type for numeric fields is *signed numeric* which provides for an additional position in the field on the display for a plus or minus sign. The only way to exit a signed numeric field is with a Field Exit key. Because the ACCT number field can only be a positive value, the digits-only designation is used.

Display Input Fields: The I in position 38 specifies that this will be an input field. All input fields will have an underline appear on the display to describe the length of the field which can be keyed. The underline occurs implicitly. An input field is also implicitly blanked out each time the display is shown. The +1 in positions 43 and 44 indicates that the field should appear on the same line as the last line described (line 3 in this case) and at the location which is 1 position greater than the ending position of the previous field. The use of the + entry is helpful when coding because it minimizes the number of changes required if you decide to change the format of the display.

ERRMSG Keyword: An error condition is described on line 12.00. It is invalid to enter an account number of all zeros, so this error condition is controlled by indicator 41. The program need only set on indicator 41 when an error occurs

and resend the record format. Indicator 41 is an option indicator and the function will only be performed when the indicator is on.

When the ERRMSG keyword is active (the option indicator is on), the fields on the display do not change. For example, if there were other input fields on the display, they would retain their original values. The input field in error also keeps its value (as opposed to being blanked out) and is displayed in reverse image. The cursor is positioned to the field in error. The message text associated with the ERRMSG keyword will appear on the message line. The keyboard is locked to draw the user's attention to the error. The user must press the Reset key before correcting the value in the field.

Note that the ERRMSG keyword supports two parameters. The first is the text that will appear on the error line. The second value (41) is optional and is an indicator that will be reset. This is normally the same indicator that conditions the error and takes away the need to reset the indicator in the program. Whenever the program receives information from the display, indicator 41 will be off.

A second error condition is described on line 14.00. If the program senses that the account number entered does not exist (missing) in the database, indicator 42 is set on and the same format is redisplayed.

If the ERRMSG keyword is active, it causes the keyboard to be locked and requires the user to press the reset key. You may choose to not lock the keyboard, but this requires more coding effort in most cases and is not shown in this manual.

Line 16.00: A constant is used to describe the valid function keys to the user.

Lines 17.00 and 18.00: The second record format used in the file is described.

Line 19.00: F3 is defined as a valid function key. Function keys described at the record level (that is after the R line) are unique to the record format. In this case the keyword could have been defined at the file level (that is before the first R line) and both record level function key specifications could have been deleted. However, there are times when you would not want the same function key to be valid on every record format so it is reasonable to define them each time.

Lines 20.00–45.00: The constants for the second record format describe the prompts and fields that will be displayed. Each of the fields to be displayed uses the *R* entry in position 29 to specify that the characteristics should be taken from the field reference file defined on line 3.00. Since position 38 is blank, all of the fields default to be output fields (meaning they cannot be keyed into when the display is shown).

Notice that there are no field lengths described in this DDS example. All of the fields are defined by referring to the field reference file.

Line 46.00: A constant is used to describe the valid command keys to the user.

Step 1. Entering the DDS and Creating the Display File (MLGINQD)

You should now enter the DDS as shown in Figure 10-3 on page 10-4 through PDM. Follow the same steps you used to create the MLGMSTP physical file in Chapter 5, "Creating Files" or copy the source from the QUSRTOOL library as described in Appendix B, "Overview of QUSRTOOL Library." The member source type should be DSPF .

Once you have entered the DDS, create the file by typing a 14 (Compile) in the *Opt* column next to the file on the Work with Members Using PDM display. You need to receive the completion message before doing the next step.

Overview of RPG Inquiry Program

The MLGINQR RPG program allows the inquiries into the mailing list file.

Description of the RPG Specifications for Inquiry Program (MLGINQR)

Now that you have created the MLGINQD display file, you can create the MLGINQR RPG program. Figure 10-4 shows the RPG specifications for the program. A description of each line follows the figure.

```
        ...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7
0001.00     F* MLGINQR - Mailing List Inquiry
0002.00     FMLGMSTP IF  E           K        DISK
0003.00     FMLGINQD CF  E                    WORKSTN
0004.00     C* Prompt for account number
0005.00     C           PROMPT    TAG                              Prompt dsp
0006.00     C                     EXFMTDSPLY1                      Display 1
0007.00     C* Check for F3
0008.00     C   93                GOTO ENDPGM                      If F3
0009.00     C* If account number is zeros, show prompt again with error msg
0010.00     C           ACCT      CABEQ*ZEROS    PROMPT         41 If zero
0011.00     C* Chain to account number record in file
0012.00     C           ACCT      CHAINMLGMSTR              42     Chain
0013.00     C* If missing, show prompt again with error message
0014.00     C   42                GOTO PROMPT                      Loop back
0015.00     C* Display the record
0016.00     C                     EXFMTDSPLY2                      Display
0017.00     C                     GOTO PROMPT                      Loop back
0018.00     C* End of program routine
0019.00     C           ENDPGM    TAG                              End pgm
0020.00     C                     SETON                     LR     Set LR
0021.00     C                     RETRN                            Return
```

Figure 10-4. RPG Specifications for MLGINQR

Line 1.00: A comment is described by the * in position 7. The other comment lines will not be described. Comment lines are used to help document the program and are not part of the running of the program.

Line 2.00: The mailing list master file (MLGMSTP) is identified. The I in position 15 indicates that this will be an input file meaning that the records will be *read* in the program (they will not be updated). The F in position 16 indicates this file as a *full procedural* file. This is needed for the random processing which will occur.

The E in position 17 indicates that externally described data will be used. When the RPG compiler reads this entry, it will extract the field descriptions for the MLGMSTP file and include them in the program.

The K entry in position 31 indicates that the file will be processed by *keys*, meaning the keyed access path specified for the MLGMSTP file will be used. The key to the file is the account number field.

The DISK entry beginning in position 40 indicates that this file will be a database file.

Line 3.00: The work station file (MLGINQD) specified previously is specified to be used in this program. The C in position 15 indicates a *combined* function meaning both input and output operations will be performed on the file. The entry is always used for work station files. The F and E entries are the same as for the disk file. The WORKSTN entry indicates a file that will be used with a work station device. There can only be one WRKSTN file per RPG program.

Line 5.00: The TAG operation specifies a label for the program to branch to. The statement is not processed when run.

Line 6.00: The EXFMT operation does a *write* and a *read* to the DSPLY1 record format defined in the work station file.

Writing to a Display: The DSPLY1 record format specified by EXFMT is placed on the display and the program waits until the user presses either the Enter key or one of the valid command keys. When this occurs, the input fields from the display are passed back to the program and are now available to be processed.

The EXFMT operation is a simple way to cause a *write* followed by a *read* to the display. Individual operations of WRITE and READ are also available. EXFMT is more efficient than the separate operations and is simpler to specify.

Until the user responds, the program is waiting. In a single job only one program at a time may be in control. Because the program is waiting, the entire job is waiting. In a busy system, many jobs may be waiting for their user to respond. The system normally places the jobs that are waiting on a special queue so it can respond to other users. The waiting job is normally taken out of main storage and placed in auxiliary storage. This is a very normal situation in a typical environment where many users are active. The system normally does not have enough main storage to satisfy all users. The system tries to manage main storage in an optimal manner so the jobs that are actually running (not waiting) have sufficient main storage to perform their functions.

The user may respond quickly or may take several seconds or minutes before responding. The system keeps track of which user belongs to which program and where within the program the *wait* actually occurred. When the user responds, the system will schedule the job back into main storage and cause the program to continue at the next instruction.

In many applications, there may be multiple users using the same program. The system keeps track of this implicitly. You normally code your program as if there will only be a single user of the program at a time. The system keeps separate work areas for each user of the program.

While this is the normal coding approach, good coding practice says that you must be aware that other users will exist on the system at the same time. Therefore, you want to code in an efficient manner. Normally this means that you want to avoid abusing the system functions and avoid locking records for long periods of time.

In this program, the MLGMSTP file is opened for input only (the file will not be updated) so there is no concern about record locking. Record locking is discussed in the next example.

Line 8.00: After the user presses a key which causes the system to take action, the program checks for F3 which means to end the program. F3 was specified in DDS to set on indicator 93. If 93 is on, the program branches to the label ENDPGM. If 93 is not on, the program continues to the next instruction.

Line 10.00: For this application, an all-zeros key field is invalid (you do not want an account number of all zeros in the database). The program will have a specific error message for this case.

Note: It would be possible to let the next check suffice (record does not exist in the database), but assume a more specific error message is desired.

To make the check, the program does a CABEQ (Compare and Branch on Equal) operation. The special entry of *ZEROS is used for the comparison and the label of the branch is described in the result field. Indicator 41 is set if the comparison is equal (41 will be set off if it is not equal).

The label of the branch point is PROMPT which is at statement 5.00 of the program. If the error is found, the program branches back and runs the EXFMT statement again with indicator 41 on. Because indicator 41 is used to option an ERRMSG keyword in DDS, the screen would remain the same (the input fields are not cleared), the field in error would appear in reverse image, the keyboard would be locked and the error text associated with the error would be displayed.

CABXX Operations: The CABXX operations are very powerful in that they can perform multiple things at the same time:

- A comparison. This is similar to COMP or IFxx.
- A label to branch to if the comparison is met.
- A setting of indicators based on the comparison.

The DDS ERRMSG keyword makes it simple to describe the handling of error conditions. The only thing the program needs to do is to set on the indicator and redisplay the format.

If the CAB test is not successful (ACCT does not equal zero), indicator 41 is set off and the program continues to the next instruction.

Line 12.00: The ACCT field value is used to *chain* or randomly access a record from the MLGMSTR record format. This is the only record format used in the MLGMSTP file, but must still be specified. There is only a single field (account number) which makes up the key to the file. The system takes the value of the key field provided and searches through the keyed access path which is part of the file. If a match is found, the system retrieves the record from the file and passes it to your program. The field values from the record (such as the name and address) are now available to be processed.

Missing Records: If the record cannot be found (the key does not exist), an error is passed back and the CHAIN operation sets on indicator 42. A *missing* indicator must always be specified on a CHAIN operation and you must program for the possibility that the record will not exist in the file. If the record exists, indicator 42 will be set off. RPG describes this as the *no record found indicator.* This manual will describe it as the *missing* indicator.

Line 14.00: If the *missing* indicator is on, the program branches to redisplay the first format again. Indicator 42 is used in DDS to describe an error message.

Line 16.00: If the key field does exist, the record for the specified ACCT number is now in main storage. The EXFMT operation requests the DSPLY2 format to be written to the display and a read to occur. The program will wait until the user responds.

Externally Described Data: Because of the use of externally described data, the program knows the fields that need to be passed to the work station support so they may be displayed. Similarly, the program knows the fields and indicators which will be passed back to the program when the user responds.

Notice that the program does not even specify the fields in the database file nor the fields in the work station display format. The function of externally described data makes this happen automatically. See the additional comments at the end of this program on externally described data.

Line 17.00: When return occurs from the EXFMT statement, the user has signaled an end .and the program branches back to the PROMPT label. This will cause the first record format to be displayed again.

In the DDS for the work station file, F3 was described as a valid function key. However, the program does not distinguish whether F3 was pressed or just the Enter key was pressed. The program always redisplays the first format again.

Lines 19.00−21.00: The ENDPGM label is branched to when the user presses F3 to end the program from the first display. RPG supports the LR indicator (*last record*) which means the program is completely finished. When this indicator is set and the RETRN operation occurs, the files will be closed (implicitly) and the program's work areas will be deleted. This means that if the program is called again, a fresh copy of the work areas will be made and the files will be reopened.

While reopening files is the right thing to do in the normal case, the opening of files and reestablishing of work areas causes performance overhead. In a later program, you will see a return without setting LR on in order to improve performance. There are other techniques for leaving files open. See the *Database Guide* discussion of the SHARE capability.

Step 2. Entering the RPG Specifications and Creating the Program (MLGINQR)

You should now enter the RPG specifications as shown in Figure 10-4 on page 10-7 through PDM. Follow the same steps you used to create the MLGMSTP physical file in Chapter 5, "Creating Files" or copy the source from the QUSRTOOL library as described in Appendix B, "Overview of QUSRTOOL Library." The member source type should be RPG .

Once you have entered the RPG specifications, create the program by typing a 14 (Compile) in the *Opt* column next to the program on the Work with Members Using PDM display.

Step 3. Running the RPG Inquiry Program (MLGINQR)

When you get a completion message that the MLGINQR program has been compiled, you can run the program. Do this now by typing the command:

```
CALL MLGINQR
```

When the Mailing List Inquiry display is shown, you can enter an account number to display the account information. Do this now (such as account number 10057) and then press F3 twice to exit when you are done.

More on Externally Described Data

The RPG compiler uses the externally described formats and the specifications you have made to determine what needs to be done on each input/output operation.

For example, the CHAIN operation is strictly an input operation (no output occurs to the database). Therefore, the RPG compiler creates instructions to move the fields from the record format to work areas. These work areas are the field names specified in the format, but the work areas are now inside the RPG program as shown in Figure 10-5. These field names (work areas) are now normal fields and can be processed with RPG operations such as ADD, MOVE.

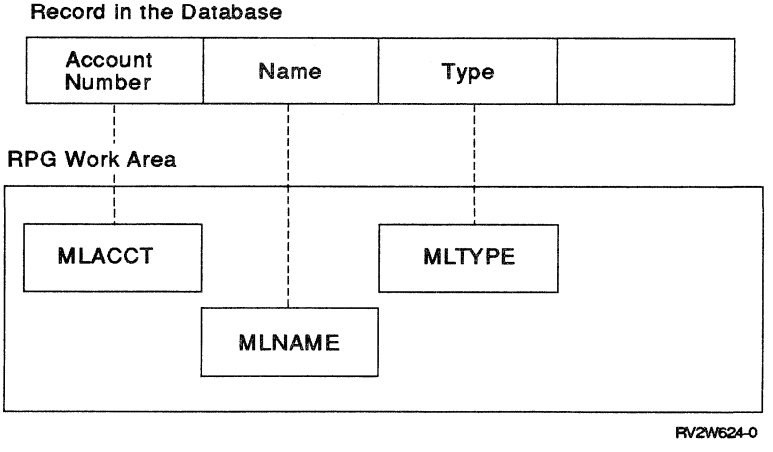


Figure 10-5. RPG Work Areas

The work station file is defined for *combined* operations. Each record format in a work station file has two format descriptions that are used with a program. The input format describes the field names that will be passed from the program to be placed on the display and the option indicators that will control the functions performed (such as when keywords are used). The output format describes what should be passed back to the program in terms of fields that were entered by the user and response indicators (such as the status of the function key indicators).

The externally described data function occurs when the program is created (not when the program is run). Therefore, if the database changes or you want to display other fields from the same database record, the program must be re-created. In most instances a simple change where you want to display a field

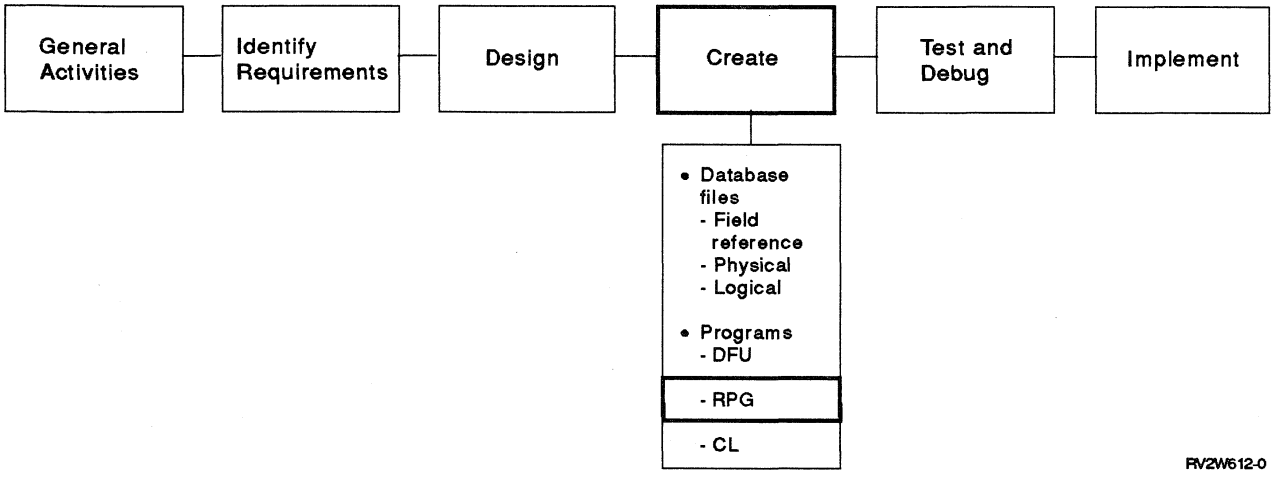
that was not previously being displayed does not require any changes to the RPG source. After the work station file is re-created, the RPG program must be re-created to bring in the new definitions.

Level Check: This now leads to the potential problem of what would happen if the database was changed (assume you increased the size of the name field) and you did not re-create the MLGINQR program. After the database file is re-created and the data copied to the new version, the program would try to read a format which differs from the format that was used at program create time. The level check function of the system is normally used to protect against this. When a file is created, the default is LVLCHK(*YES). If the program reads the same file, but the format has changed, the system will prevent the file from being opened. The program will abnormally end with an appropriate message.

If LVLCHK(*NO) is used with the inquiry program and the format had changed, the user might see the information from one field displayed under a prompt for a different field. A worse case is where the program updates a database. Without LVLCHK(*YES), it would be possible for the user to corrupt the database. You can specify LVLCHK(*NO), but for most processing it is better to take the default and protect the integrity of the data. To correct a level check mismatch, you normally just re-create the program.

Level check also works on work station files. The system uses a special algorithm which is compiled into the program to determine if the format has significantly changed. For example, in a work station file, the program doesn't care about changes to the constants nor where they appear on the display. The algorithm used to calculate the level check identification does not consider this type of information. You can change this type of information without causing a change to the level check value and, therefore, avoid the program being re-created. However, if you add a field or change the length of an existing field, this will affect the program and the algorithm will cause a level check to occur.

Chapter 11. RPG Solution for Interactive Maintenance



This chapter describes an RPG solution for creating a maintenance program that provides a display to maintain the account information.

Overview of Mailing List Maintenance

In Chapter 6, "Adding Records to the Database File Using DFU," a DFU program was used to maintain the data in the physical file MLGMSTP. DFU programs are easy to create, but they do not allow you to tailor the type of processing that you may want to do. For this reason, this section is devoted to providing a tailored approach. Programs that maintain files can be some of the more complex functions written in any application. Not only must the programs display, add, update, and delete records in the file, but they must also perform validation checking, provide good ease-of-use factors, provide recovery capability, and be conscious of performance considerations. The maintenance program used in this chapter is intended to do these things and is shown in an overview in Figure 11-1.

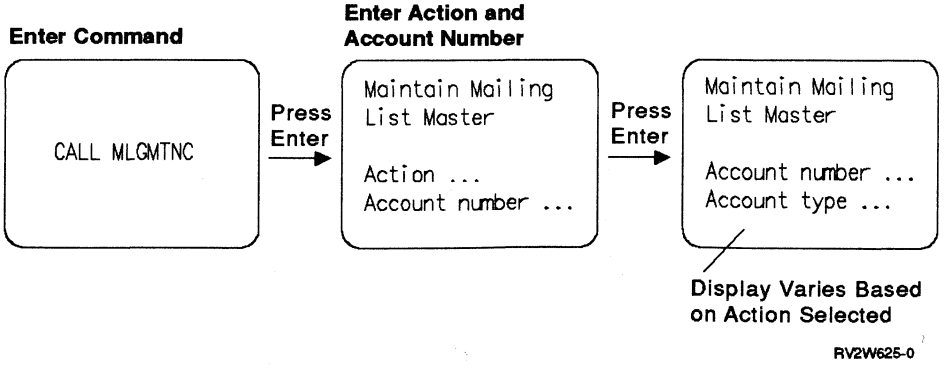


Figure 11-1. Overview of Maintenance Program Displays

To do maintenance to a file, the user would call a program and would be presented with the following display:

```
                    Maintain Mailing List Master

Action . . . . . :              1=Display  2=Change
                                3=Add      4=Delete
                                5=Display GE value
                                6=Name search

Account number . . . . . :      Numeric 5.0

Search field . . . . . :        Char




For Options 1-5, enter an Account number.
For Option 6, enter a Search field.


F3=Exit
```

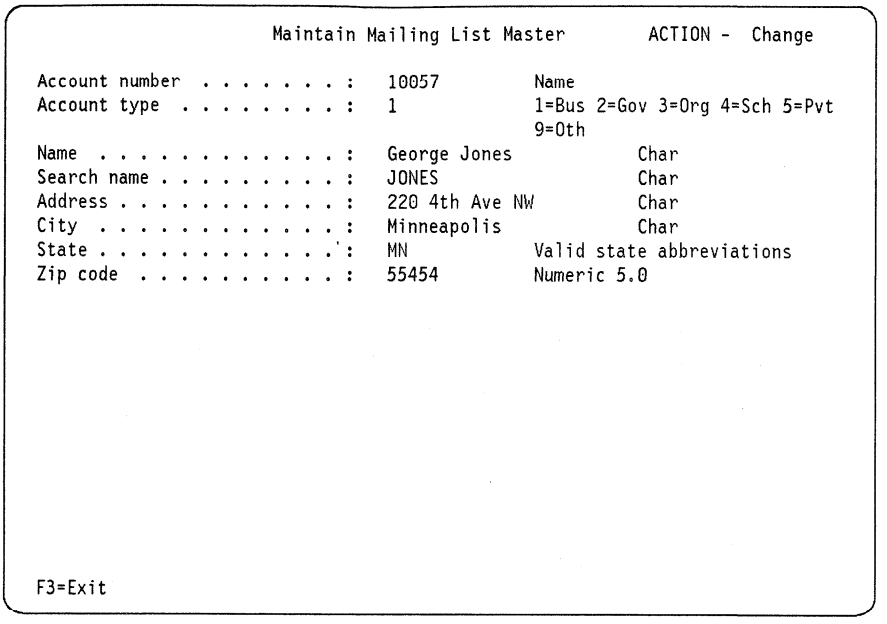
Options 1 through 4 require the user to enter a specific account number to be processed.

Option 5 allows the user to search through the file (one record at a time) either from the account number entered or from the beginning of the file if no account number is entered. If no account number is entered, the search begins by displaying the first record in the file and the user may page down to see the next record. If a value is entered such as account number 50000, the search starts with account number 50000. If the specified account number does not exist, the first account number after 50000 will be displayed. The user can page up or page down. If the user sees a record he wants to change or delete, function keys allow the display to be switched to change or delete mode.

The option 6 search function allows a method of determining the account number when only the account name is known. Option 6 allows a search by entering in one or more characters in the search field. A display appears of all those accounts matching the characters keyed in. For example, if SMITH is keyed in, all the accounts with a search field of SMITH, SMITHERMAN, SMITHE, and so on are shown. This is based on the search field entered into every record. The user could respond by requesting a return with a specific account number. When the return occurs, the record would be displayed in display mode. The user could then press a function key to go into change mode on the record. Option 6 will be described in more detail in a later chapter.

Note: Option 6 will **not** work until you have completed the tasks in Chapter 11 where this function is done.

Once an account number has been entered, the second display shows that record from the database. It allows the user to enter a new record, change an existing record, or delete an existing record.



The ACTION type is shown in the upper-right corner of the display. In this case, the user has requested to change a record. The fields for the existing record can be changed.

The structure of the maintenance function for MLGMSTP consists of the following:

- A CL program, MLGMTNC, that is called to run the maintenance function. It is a small program, but provides for some control functions and error handling.
- A display file, MLGMTND, containing the DDS that describes the display formats that are used.
- An RPG program, MLGMTNR, that does the maintenance work.

Figure 11-2 on page 11-4 gives an overview of the maintenance program.

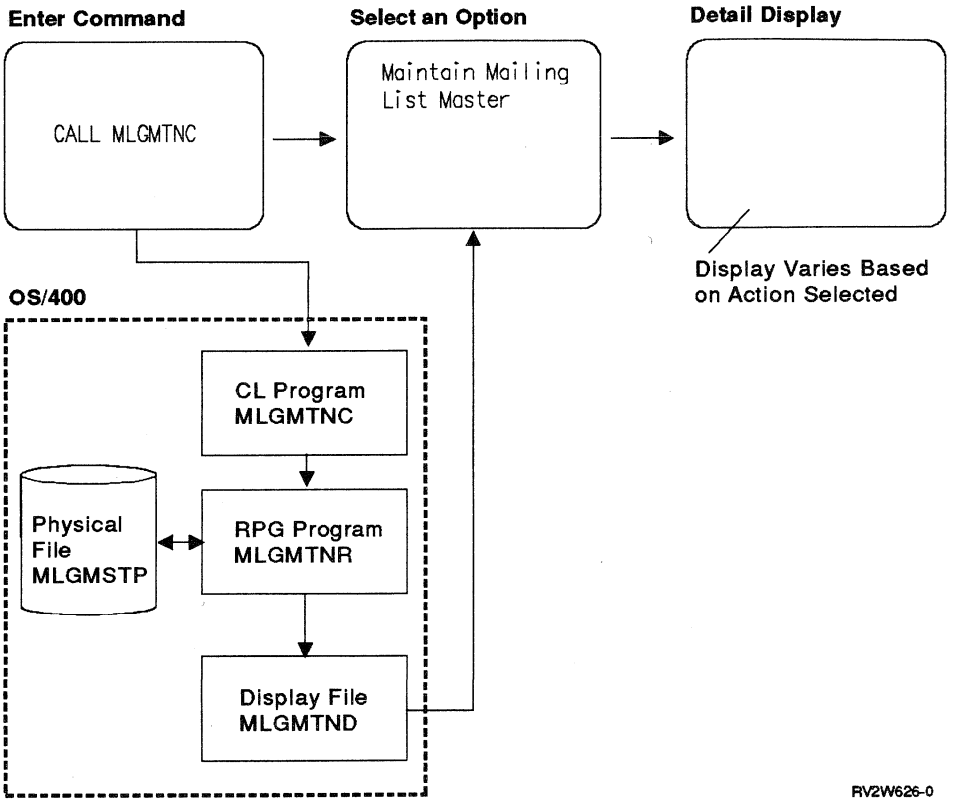


Figure 11-2. Overview of Maintenance Program MLGMTNR

Description of the CL Maintenance Program (MLGMTNC)

Figure 11-3 shows the code for the CL program MLGMTNC. A description of each line follows the figure.

```
        ...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7
0001.00 /* MLGMTNC - Maintain mailing list master                          */
0002.00             PGM
0003.00             DCL        &RTNCDE *CHAR LEN(8)
0004.00             CALL       PGM(MLGMTNR) PARM(&RTNCDE)
0005.00             IF         (&RTNCDE *EQ 'GOOD') RETURN
0006.00             SNDPGMMSG  MSGID(CPF9898) MSGF(QCPFMSG) MSGTYPE(*ESCAPE) +
0007.00                          MSGDTA('Bad return code of ' *CAT +
0008.00                          &RTNCDE *TCAT ' from MLGMTNR')
0009.00             ENDPGM
```

Figure 11-3. Description of CL Program MLGMTNC

Lines 1.00 and 2.00: The /* xxx */ entry specifies a comment. A comment must begin with a /* and end with an */.

The PGM statement defines the beginning of the program.

Line 3.00: The DCL statement declares a variable named &RTNCDE which is defined as a character type of length 8. All variables in CL must begin with the & symbol. This is helpful as it distinguishes between constants and variables.

Line 4.00: The program calls the RPG program MLGMTNR and receives a parameter named &RTNCDE. A parameter is a two-way street. It is both passed to the called program and then returned. The called program may update it. In this case, the called program (RPG) does not care about the value when the program begins. It must only set the value on a return.

Line 5.00: The IF statement checks the value of the &RTNCDE variable. If the value is equal to GOOD, the RETURN command is run. This causes an end to the CL program and is the normal way the program is ended.

Lines 6.00−8.00: The Send Program Message (SNDPGMMSG) command is used to send an escape message for a return code of other than GOOD. When an escape message is sent, the program immediately ends. If the program that called this program is not monitoring for the escape message, it would be abnormally ended.

The escape message sent by the SNDPGMMSG command is used for very unlikely conditions and also assists in debugging the program.

Sending an Escape Message: To send an escape message, a message description must exist in a message file. Each message description has a unique message identification (MSGID). The MSGID sent is CPF9898 which is a general purpose message supplied by the OS/400 program. It allows you to specify the MSGDTA parameter (message data) to say whatever is needed. This avoids having to add a message for every situation. The string of text is built up as the MSGDTA by the use of literals (surrounded by apostrophes) and concatenation symbols (*CAT and *TCAT). The *CAT entry says to do a concatenation function. The *TCAT entry says to trim off the trailing blanks from the left value and then perform concatenation. Normally, you use *TCAT following a variable that may contain less characters than the full length of the field. For example, if &RTNCDE contained a value of *ERRORbbb*, you would want to take off the trailing blanks to make a more readable message. If the &RTNCDE had a value of ERROR, the message would read as:

```
Bad return code of ERROR from MLGMTNR
```

Line 9.00: The ENDPGM command designates the end of the source and acts as a RETURN command if it is run. In this case, the ENDPGM command would never be run because the program will either end by the RETURN command or the SNDPGMMSG command that sends an escape message.

You will find many high-level language programs preceded with CL programs. CL programs offer a convenient place to specify various system functions. In this case the functions are simple.

Step 1. Entering and Creating the CL Maintenance Program (MLGMTNC)

You should now enter the CL coding as shown in Figure 11-3 on page 11-4 through PDM. Follow the same steps you used to create the MLGMSTP physical file in Chapter 5, "Creating Files" or copy the source from the QUSRTOOL library as described in Appendix B, "Overview of QUSRTOOL Library." The member source type should be CLP .

Once you have entered the CL coding, create the program by typing a 14 (Compile) in the *Opt* column next to the program on the Work with Members Using PDM display.

You cannot call the program until the completion message occurs. Since this program calls the RPG program MLGMTNR, the RPG program must also exist.

Overview of Display File

The MLGMTND display file contains the DDS that describes the display formats that are used to produce the display used by the maintenance program.

Description of DDS for Maintenance Display File (MLGMTND)

Figure 11-4 shows the DDS for display file MLGMTND. A description of each line follows the figure.

```
        ...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7
0001.00     A* MLGMTND - Maintain Mailing List Master - used by MLGMTNR
0002.00     A                                       PRINT
0003.00     A                                       REF(MLGREFP)
0004.00     A          R DSPLY1                     TEXT('Prompt for action +
0005.00     A                                       and account number')
0006.00     A                                       CA03(93 'Exit')
0007.00     A                                       SETOFF(40 'Addl F keys')
0008.00     A                                       SETOFF(43 'No longer exst')
0009.00     A                                       SETOFF(45 'Roll EOF')
0010.00     A                                       SETOFF(49 'Bad sub pgm')
0011.00     A                                       SETOFF(50 'Protect')
0012.00     A                                       SETOFF(53 'Protect')
0013.00     A                                     1 25'Maintain Mailing List +
0014.00     A                                       Master'
0015.00     A                                       DSPATR(HI)
0016.00     A                                     3  2'Action  . . . . . . . +
0017.00     A                                       . . . :'
0018.00     A            ACTION         1   I    +1
0019.00     A  44                                   ERRMSG('Invalid action' 44)
0020.00     A                                     3 50'1=Display  2=Change'
0021.00     A                                     4 50'3=Add      4=Delete'
0022.00     A                                     5 50'5=Display GE value'
0023.00     A                                     6 50'6=Name search'
0024.00     A                                     8  2'Account number  . . . . +
0025.00     A                                       . . . :'
0026.00     A            ACCT      R     D  I    +1REFFLD(MLACCT)
0027.00     A  41                                   ERRMSG('The account +
0028.00     A                                       number already exists' 41)
0029.00     A  42                                   ERRMSG('The account +
0030.00     A                                       number does not exist' 42)
0031.00     A  46                                   ERRMSG('The account +
0032.00     A                                       number cannot be zeros' 46)
0033.00     A  47                                   ERRMSG('End of file has +
0034.00     A                                       been reached' 47)
0035.00     A                                     8 50'Numeric 5.0'
0036.00     A                                    10  2'Search field  . . . . . +
0037.00     A                                       . . . :'
0038.00     A            SEARCH    R        I    +1REFFLD(MLSRCH)
0039.00     A  48                                   ERRMSG('The search field +
0040.00     A                                       cannot be blank' 48)
0041.00     A                                    10 50'Char'
0042.00     A*   39 is set after the 1st display
0043.00     A  39                                15  2'Last action -'
```

Figure 11-4 (Part 1 of 4). DDS for Display File MLGMTND

```
          ...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7
0044.00      A  39        ACTTXT        10     15 16
0045.00      A  45                              16  5'Rollup or rolldown has +
0046.00      A                                       reached end of file'
0047.00      A  39N93                           16  5'Account number'
0048.00      A  39N93     PRVACC    R           16 20REFFLD(MLACCT)
0049.00      A  39N93     PRVNAM    R           16 30REFFLD(MLNAME)
0050.00      A  43                              16  5'The account number no +
0051.00      A                                       longer exists when it +
0052.00      A                                       was re-accessed'
0053.00      A  93                              16  5'Function was cancelled'
0054.00      A  45                              16  5'Rollup or rolldown has +
0055.00      A                                       reached end of file'
0056.00      A  49                              16  5'The sub program for name +
0057.00      A                                       search ended in error'
0058.00      A                                  19  2'For Options 1-5, enter +
0059.00      A                                       an Account number.'
0060.00      A                                  20  2'For Option 6, enter +
0061.00      A                                       a Search field.'
0062.00      A                                  24  2'F3=Exit'
0063.00      A          R DSPLY2                     TEXT('MLGMSTP record +
0064.00      A                                       display')
0065.00      A                                       CA03(93 'Exit')
0066.00      A  40                                   CA11(91 'Delete')
0067.00      A  40                                   CA06(96 'Change')
0068.00      A*   51 allows the delete option
0069.00      A  51                                   CA08(92 'Confirm delete')
0070.00      A  40                                   ROLLUP(97 'Rollup')
0071.00      A  40                                   ROLLDOWN(98 'Rolldown')
0072.00      A                                       SETOFF(50 'Protect')
0073.00      A                                       SETOFF(51 'Spcl msg')
0074.00      A                                       SETOFF(52 'Addl F keys')
0075.00      A                                       SETOFF(59 'HI dlt text')
0076.00      A                                       SETOFF(64 'Chgd after rel')
0077.00      A                                       RTNDTA
0078.00      A                                   1 25'Maintain Mailing List +
0079.00      A                                       Master'
0080.00      A                                       DSPATR(HI)
0081.00      A                                     +8'ACTION - '
0082.00      A                                       DSPATR(HI)
0083.00      A            ACTTXT    R              +1REFFLD(ACTTXT *SRC)
0084.00      A                                       DSPATR(HI)
0085.00      A                                   3  2'Account number  . . . . +
0086.00      A                                       . . . :'
0087.00      A            MLACCT    R     D  B      +3
0088.00      A  53                                   DSPATR(PR)
0089.00      A  65                                   ERRMSG('The field is not +
0090.00      A                                       unique in the master +
0091.00      A                                       file')
0092.00      A  71                                   ERRMSG('The field cannot +
0093.00      A                                       be blank' 71)
0094.00      A                                   3 50'Name'
0095.00      A                                   4  2'Account type  . . . . . +
0096.00      A                                       . . . :'
```

Figure 11-4 (Part 2 of 4). DDS for Display File MLGMTND

```
0097.00      A            MLTYPE    R        B    +3
0098.00      A*   50 is set for inquiry or delete
0099.00      A  50                                  DSPATR(PR)
0100.00      A                                   4 50'1=Bus 2=Gov 3=Org 4=Sch +
0101.00      A                                     5=Pvt'
0102.00      A                                   5 50'9=Oth'
0103.00      A                                   6  2'Name  . . . . . . . . . +
0104.00      A                                     . . . :'
0105.00      A            MLNAME    R        B    +3CHECK(LC)
0106.00      A  50                                  DSPATR(PR)
0107.00      A                                   6 60'Char'
0108.00      A                                   7  2'Search name . . . . . . +
0109.00      A                                     . . . :'
0110.00      A            MLSRCH    R        B    +3
0111.00      A  50                                  DSPATR(PR)
0112.00      A  72                                  ERRMSG('The field cannot +
0113.00      A                                     be blank' 72)
0114.00      A                                   7 60'Char'
0115.00      A                                   8  2'Address . . . . . . . . +
0116.00      A                                     . . . :'
0117.00      A            MLADDR    R        B    +3CHECK(LC)
0118.00      A  50                                  DSPATR(PR)
0119.00      A                                   8 60'Char'
0120.00      A                                   9  2'City  . . . . . . . . . +
0121.00      A                                     . . . :'
0122.00      A            MLCITY    R        B    +3CHECK(LC)
0123.00      A  50                                  DSPATR(PR)
0124.00      A  73                                  ERRMSG('The field cannot +
0125.00      A                                     be blank' 73)
0126.00      A                                   9 60'Char'
0127.00      A                                  10  2'State . . . . . . . . . +
0128.00      A                                     . . . :'
0129.00      A            MLSTAT    R        B    +3
0130.00      A  50                                  DSPATR(PR)
0131.00      A  74                                  ERRMSG('The field cannot +
0132.00      A                                     be blank' 74)
0133.00      A                                  10 50'Valid state abbreviations'
0134.00      A                                  11  2'Zip code  . . . . . . . +
0135.00      A                                     . . . :'
0136.00      A            MLZIP     R        B    +3
0137.00      A  50                                  DSPATR(PR)
0138.00      A  75                                  ERRMSG('The field cannot +
0139.00      A                                     be zeros' 75)
0140.00      A                                  11 50'Numeric 5.0'
```

Figure 11-4 (Part 3 of 4). DDS for Display File MLGMTND

```
0141.00      A  51                                  21  2'Press F8 to delete. +
0142.00      A                                            Use F3 to return without +
0143.00      A                                            deletion.  You cannot +
0144.00      A                                            press Enter.'
0145.00      A  59                                        DSPATR(HI)
0146.00      A  64                                 21  2'The record has been +
0147.00      A                                            changed since first +
0148.00      A                                            accessed. Current values +
0149.00      A                                            displayed.'
0150.00      A                                            DSPATR(HI)
0151.00      A                                     24  2'F3=Exit'
0152.00      A  40                                    +5'Rollup'
0153.00      A  40                                    +5'Rolldown'
0154.00      A  40                                    +5'F6=Change'
0155.00      A  40                                    +5'F11=Delete'
0156.00      A  51                                 24 15'F8=Delete'
```

Figure 11-4 (Part 4 of 4). DDS for Display File MLGMTND

Lines 1.00−6.00: These statements are similar to the previous statements used in the inquiry program.

Lines 7.00−12.00: The SETOFF keyword allows you to set off indicators that are used in DDS. Any option indicators are considered in the formatting of the display. The set off function does not occur until the data is passed back to the program.

Setting Off Error Indicators: Most of the indicators described are used to control constants (informational or errors) that may appear on the display. The indicators which are used in the ERRMSG keyword are not needed because they can be set off by the ERRMSG keyword. Setting off the indicators on the return to the program avoids having the program reset them.

In some cases it is not necessary to reset the indicators because the program logic will either set the indicator on or off each time the display is shown. However, in a complex program this is often not the case. The program that processes this file will redisplay the first error encountered. If the user makes an error and then makes an invalid change on the redisplayed error, the indicators could be set to provide misleading information. In general, it is a good practice to keep setting the indicators off.

Lines 13.00−15.00: These statements are similar to the previous statements used in the inquiry program.

Lines 16.00−23.00: This is the action field that the user will key into. It is described as an *input* field (I in position 38) and constants describe the values that may be keyed. Indicator 44 is used for an error condition where the user has keyed an invalid value.

Using the Field Reference File: Notice that this field is defined as 1 byte in length and is one of the few fields on the display which has a hard-coded length. Most of the fields are defined using the field reference file concept. This allows good productivity because:

- There is more accuracy in the original specification (less debugging to bring the application into production).
- Handling changes is easier because many of the changes can be made by simply re-creating the display file. For example, if the size of the name field

changes, it can normally be handled just by re-creating it. When there are hard-coded specifications in the program that control field lengths, you are more subject to code changes and errors. This is particularly important when you are originally creating the application because it is then that the field lengths are most likely to change.

Lines 24.00−35.00: The ACCT field is described as an I for *input* field. This means that the field will be used for input only. When it is displayed, the field will be blank. The user keys into the field and the contents are returned to the program when the user presses the Enter key.

There are several error conditions associated with the account number field. Indicator 41 should be set on if the account number already exists when the user requests to add a new record. Indicator 42 should be set on when the user requests a function like display and the account number does not exist. Indicator 46 should be set on when the user attempts to add a new record with an all-zero account number. Indicator 47 should be set on for option 5 when the user is searching through the file (record after record) and end of file is reached.

Lines 36.00−41.00: The SEARCH field is used with option 5 to search through the file. How the user works with this field was described with the explanation of the first display.

Lines 43.00−57.00: Various constants and fields are optionally displayed to provide feedback to the user relative to the last function performed. For example, if the user did a change function, the display would show *Last action - Change* and the account number and name of the account changed.

Line 44.00 defines the ACTTXT field as 10 characters long. This will provide a text description of the action requested last by the user. The same value also appears on the second display.

Lines 48.00 and 49.00 contain the entries for the previous account number and name. Both of these fields use the reference function so that the fields are defined without hard-coded entries for lengths. It is up to the program to move data to these fields after processing a transaction.

Feedback for Users: Feedback for users is usually an important consideration. For new users, it is a comforting feeling to know that the system processed the request. For experienced users, the feedback is rarely needed, but is desirable when the user loses his train of thought.

Lines 47.00−49.00 are conditioned by indicators. The three fields (one constant and two output fields) should only be displayed when an action has occurred.

To prevent the constant from being displayed the first time the program writes to the display, indicator 39 is used. The fields will only appear if indicator 39 is set on.

Indicator 93 is the exit function key. If it is pressed on the second display, the return to the first display should not show that an action has been performed. Specifying N93, says that the exit key was not pressed from the second display.

Lines 58.00−62.00: Constants are used to describe the normal user actions and the valid function keys.

Lines 63.00−77.00: The second record format describes the function keys that are valid. Indicator 51 is set on in the program when delete is requested. If indicator 51 is on, F3 is not valid, but F8 and F12 are. Indicator 40 controls whether some of the keys are valid. Indicator 40 will be set by the program when option 1 or 5 is being used. The *set off* function is also used for this record format.

Line 78.00: The RTNDTA keyword is used to allow the work station display to be read again without doing any physical input/output on the second read. This is a valuable keyword because of the way RPG operates in that it has only one work area per field name in the program. The function of RTNDTA is explained further in the program description.

Lines 79.00−85.00: Constants are used to describe the first line of the display. Note that the action to be performed is described in text by the field ACTTXT in the upper-right corner of the display. On the first record format, the user entered a code (the option number). The program will take the code (such as *1*) and translate it into a word (such as *Display*) to assist the user in determining what function is to be performed.

The ACTTXT field also uses the *reference* function, but describes the field name ACTTXT as being already defined in the source (*SRC entry). The ACTTXT field was used on the first format to describe the last action the user performed.

Lines 86.00−95.00: The account number field (MLACCT) is displayed as a *both* field meaning it is both *input* by the program to the display and *output* from the display back to the program.

The user should be allowed to change the account number while in *change* mode. All of the other functions (display, add, delete) should prevent the user from making a change.

Changing Key Fields: Some application designs will prevent the user from changing the account number (key field) once the record is added to the file. From an auditing viewpoint, it may be required that the old record be deleted and a new record added instead of just a change to the account number. It is actually easier to code for this type of design approach, but assume that a change to the account number is considered valid.

Indicator 53 is used to control whether the user is allowed to change the value displayed. The indicator conditions the DSPATR (display attribute) keyword which specifies PR meaning protect. When the indicator is on, the value in the field is protected on the display meaning the user cannot key into the field. Because the field is a *both* field, the value is returned to the program unchanged. The field will appear with an underline on the display, but will be protected when the indicator is on.

If the user is allowed to change the field, then the program must check for error conditions and send error messages if the errors occur. This is the purpose of the ERRMSG keywords defined with the MLACCT field.

The D entry in position 35 describes a *digits-only* entry which was described in the MLGINQD file.

Lines 96.00−103.00: The account type field (MLTYPE) is also a *both* type field. It is protected if indicator 50 is on. The field reference file used the VALUES keyword to describe the valid values for this field. Therefore, there is no reason for an error message to be associated with this field nor for the program to do any checking. The system will ensure the value is correct for a new or changed record. If an error occurs, the system will send an error message and the user must correct the field to a valid value. The program will never receive a value that is not specified in the VALUES list.

Constants are used to assist the user in properly coding the entry for this field. The error message produced by the system will not include the valid values. For this reason, some programmers prefer not to use the DDS validity checking keywords.

Lines 104.00−141.00: The remaining fields are specified in a similar manner. The fields are defined as type both and are protected if indicator 50 is on. Error messages are defined for some of the fields.

Lines 142.00−157.00: The remainder of the display provides for constants that are mostly controlled based on indicator settings. These describe error conditions, additional instructions, or additional function keys.

Deleting Records: To delete a record, the user enters the account number and is shown the record. Text appears which describes that the user can either delete the record or cancel the delete request.

To delete the record, the user must press F8 or F12. If the Enter key is pressed, the user will see the line of text describing what to do in highlighted mode. Normally, it is good practice to make a different and noticeable action (something other than the Enter key) to delete a record.

Step 2. Entering the DDS and Creating the Maintenance Display File (MLGMTND)

You should now enter the DDS as shown in Figure 11-4 on page 11-6 through PDM. Follow the same steps you used to create the MLGMSTP physical file in Chapter 5, "Creating Files" or copy the source from the QUSRTOOL library as described in Appendix B, "Overview of QUSRTOOL Library." The member source type should be DSPF.

Once you have entered the DDS, create the file by typing a 14 (Compile) in the *Opt* column next to the file on the Work with Members Using PDM display.

You need to receive the completion message that the display was created successfully before creating the next RPG program.

Overview of RPG Maintenance Program

The MLGMTNR RPG program maintains the mailing list file.

Description of the RPG Specifications for Maintenance Program (MLGMTNR)

Now that you have created the MLGMTNC CL program and the MLGMTND display file, you can create the MLGMTNR RPG program. Figure 11-5 shows the RPG specifications for the program. A description of each line follows the figure.

```
        ...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7
0001.00     F* MLGMTNR - Maintenance of Mailing list master - call by MLGMTNC
0002.00     FMLGMSTP UF  E           K        DISK         KINFDS FILEDSA
0003.00     FMLGMTND CF  E                    WORKSTN
0004.00     E                    ACT     6   6 10               Action text
0005.00     I* Data structure to get file error status
0006.00     IFILEDS      DS
0007.00     I                                      B 125 1260RCDLEN
0008.00     I                                       *STATUS  STS
0009.00     I* DSPRCD1 DS is used to contain the fields from MLGMSTP
0010.00     IDSRCD1    E DSMLGMSTP
0011.00     I* DSPRCD2 DS is used to test 1st access against 2nd access
0012.00     IDSRCD2      DS                             300
0013.00     C           *ENTRY    PLIST                           Parm list
0014.00     C                     PARM           RTNCDE  8        Return code
0015.00     C* Check for DSRCD2 being too small
0016.00     C           RCDLEN    IFGT 300                        GT DS len
0017.00     C                     MOVE 'LENERR  'RTNCDE           Set error
0018.00     C                     GOTO BADEND                     Bad pgm end
0019.00     C                     END                             LT rcd len
0020.00     C* Prompt for action and account number
0021.00     C           PROMPT    TAG                             Prompt dsp
0022.00     C                     MOVE ACCT      PRVACC           Prev acct
0023.00     C                     EXFMTDSPLY1                     Display 1
0024.00     C                     SETON                      39   Dsp switch
0025.00     C* Check for F3 and invalid action
0026.00     C   93                GOTO ENDPGM                     If F3
0027.00     C           ACTION    IFLT '1'                        LT 1
0028.00     C           ACTION    ORGT '6'                          GT 6
0029.00     C                     SETON                      44   Error msg
0030.00     C                     GOTO PROMPT                     Loop back
0031.00     C                     END                             LT 1 etc
0032.00     C* Set action text for detail display
0033.00     C           SETTXT    TAG                             Set text
0034.00     C                     MOVE ACTION    AX       30      Make numeric
0035.00     C                     MOVE ACT,AX    ACTTXT          Move text
0036.00     C* If name search, call sub program
0037.00     C           '6'       IFEQ ACTION                     If search
0038.00     C           SEARCH    CABEQ*BLANK    PROMPT        48 If blank
0039.00     C                     EXSR SUBPGM                      Sub pgm
0040.00     C                     MOVE 'X'       CALLED  1        Set switch
0041.00     C* If no account is returned, re-display first prompt
0042.00     C           ACCT      CABEQ*ZEROS    PROMPT           Loop back
```

Figure 11-5 (Part 1 of 5). RPG Specifications for MLGMTNR

```
          ...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7
0043.00      C* If account is returned, use display mode to display
0044.00      C                    MOVE '1'       ACTION          Assume dsp
0045.00      C                    MOVE ACTION    AX      30      Make numeric
0046.00      C                    MOVE ACT,AX    ACTTXT          Move text
0047.00      C                    END                            If search
0048.00      C* If Display GE opt, set lower limit, read rcd and branch to dsp
0049.00      C           '5'      IFEQ ACTION                    Option 5
0050.00      C           ACCT     SETLLMLGMSTR                   Set low lmt
0051.00      C                    READ MLGMSTR                  47 Read next
0052.00      C   47               GOTO PROMPT                    EOF loop bak
0053.00      C                    GOTO DSPACT                    Dsp action
0054.00      C                    END                            Option 5
0055.00      C* Check for zero account number and chain to record
0056.00      C           ACCT     CABEQ*ZEROS    PROMPT         46 If blank
0057.00      C           ACCT     CHAINMLGMSTR             30    Chain
0058.00      C* If Add, check for unique key, write out record with key field
0059.00      C           '3'      IFEQ ACTION                    Add action
0060.00      C* If not missing, send back error
0061.00      C  N30               DO                             Not missing
0062.00      C                    SETON                    41    Set error
0063.00      C                    GOTO PROMPT                    Loop back
0064.00      C                    END                            Not missing
0065.00      C* Write out record with just key field
0066.00      C                    MOVE ACCT      MLACCT          Account nbr
0067.00      C                    EXCPTADDNEW                    Add new rcd
0068.00      C* Re-access to bring in defaults for all fields
0069.00      C           ACCT     CHAINMLGMSTR             20    Re-access
0070.00      C   20               DO                             Missing
0071.00      C                    MOVE 'REACCESS'RTNCDE          Set return
0072.00      C                    GOTO BADEND                    Bad pgm end
0073.00      C                    END                            Missing
0074.00      C* Release the record lock and move record to save area
0075.00      C                    EXCPTRELESE                    Release lock
0076.00      C                    MOVELDSRCD1    DSRCD2          Save record
0077.00      C                    SETON                    53    Protect key
0078.00      C                    GOTO DSPLY                     Display
0079.00      C                    END                            Add action
0080.00      C* If missing, return error for other actions (CHG, DLT, DSP)
0081.00      C   30               DO                             Missing
0082.00      C                    SETON                    42    Set error
0083.00      C                    GOTO PROMPT                    Loop back
0084.00      C                    END                            Missing
0085.00      C* Release the record lock and move record to save area
0086.00      C                    EXCPTRELESE                    Release lock
0087.00      C                    MOVELDSRCD1    DSRCD2          Save record
0088.00      C* If Dlt, set for protect and display record
0089.00      C           '4'      IFEQ ACTION                    Dlt action
0090.00      C                    SETON                    5053  DSPATR(PR)
0091.00      C                    SETON                    51    Dlt spcl txt
0092.00      C                    GOTO DSPLY                     Display
0093.00      C                    END                            Dlt action
```

Figure 11-5 (Part 2 of 5). RPG Specifications for MLGMTNR

```
            ...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7
0094.00      C* If either form of display, set for protect and display record
0095.00      C           DSPACT    TAG                                 Dsp tag
0096.00      C           '1'       IFEQ ACTION                         Dsp action
0097.00      C           '5'       OREQ ACTION                         Dsp action
0098.00      C                     SETON                         5053  DSPATR(PR)
0099.00      C                     SETON                         40    Addl F keys
0100.00      C                     GOTO DSPLY                          Display
0101.00      C                     END                                 Dsp action
0102.00      C* If Chg, display record and allow change
0103.00      C           '2'       CABEQACTION    DSPLY                Chg action
0104.00      C* If code gets to here, it is a bad action code
0105.00      C                     MOVEL'BADACTN 'RTNCDE               Set return
0106.00      C                     GOTO BADEND                         Bad pgm end
0107.00      C* Display the record
0108.00      C           DSPLY     TAG                                 Display
0109.00      C                     EXFMTDSPLY2                         Detail dsply
0110.00      C                     MOVE MLNAME    PRVNAM               Prev name
0111.00      C* If F3 and not add, loop back to prompt again
0112.00      C   93      '3'       CABNEACTION    PROMPT               If not Add
0113.00      C* Handle roll keys (Only valid on dsp action)
0114.00      C   97                                                    If rollup
0115.00      COR 98                DO                                  If rolldown
0116.00      C   97                READ MLGMSTR                   45 EOF
0117.00      C   98                READPMLGMSTR                   45 EOF
0118.00      C   45                GOTO PROMPT                         EOF loop bak
0119.00      C                     SETON                         5053  DSPATR(PR)
0120.00      C                     GOTO DSPLY                          Goto dsply
0121.00      C                     END                                 If rollup/dw
0122.00      C* Handle F keys for change, add and delete
0123.00      C                     MOVE MLACCT    ACCT                 Acct nbr
0124.00      C                     SETOF                         40    Addl Fs
0125.00      C   96                DO                                  If F6-Chg
0126.00      C                     SETOF                         5053  DSPATR(PR)
0127.00      C                     MOVE '2'       ACTION               Set action
0128.00      C                     GOTO SETTXT                         Loop back
0129.00      C                     END                                 If F6-Chg
0130.00      C   91                DO                                  If F11-Dlt
0131.00      C                     MOVE '4'       ACTION               Set action
0132.00      C                     GOTO SETTXT                         Loop back
0133.00      C                     END                                 If F11-Dlt
0134.00      C* If DSP type (Record has already been displayed)
0135.00      C           '1'       CABEQACTION    PROMPT               Dsp action
0136.00      C           '5'       CABEQACTION    PROMPT               GE action
0137.00      C* Re-access record for CHG, ADD, or DLT: send error if missing
0138.00      C           ACCT      CHAINMLGMSTR                  43    Chain
0139.00      C   43                GOTO PROMPT                         If missing
0140.00      C* Compare against previous record value to see if it changed
0141.00      C           DSRCD1    IFNE DSRCD2                         If NE
0142.00      C                     EXCPTRELESE                         Release lock
0143.00      C                     MOVELDSRCD1    DSRCD2               Save record
0144.00      C                     SETON                         64    Set error
0145.00      C                     GOTO DSPLY                          Display
0146.00      C                     END                                 If NE
0147.00      C* The re-access of the data base record has overlayed the display
0148.00      C*   file field values. Use RTNDTA keyword and read again
0149.00      C                     READ DSPLY2                    20 Read RTNDTA
```

Figure 11-5 (Part 3 of 5). RPG Specifications for MLGMTNR

```
          ...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7
0150.00      C* If Add type,  check for F3 and if so delete rcd already added
0151.00      C           '3'       IFEQ ACTION                        Add action
0152.00      C   93                DO                                 If F3
0153.00      C                     DELETMLGMSTR                       Delete
0154.00      C                     GOTO PROMPT                        Loop back
0155.00      C                     END                                If F3
0156.00      C*    Validate routine goes to DSPLY if error occurs
0157.00      C                     EXSR VALIDT                        Validate
0158.00      C* Record is already added, perform update and check for error
0159.00      C                     UPDATMLGMSTR                 20    Add new rcd
0160.00      C*    Check error routine will not return if error occurs
0161.00      C   20                EXSR CHKERR                        Check error
0162.00      C                     GOTO PROMPT                        Loop back
0163.00      C                     END                                Add action
0164.00      C* If Chg type, validate fields
0165.00      C           '2'       IFEQ ACTION                        Chg action
0166.00      C*    Validate routine goes to DSPLY if error or re-prompt occurs
0167.00      C                     EXSR VALIDT                        Validate
0168.00      C                     UPDATMLGMSTR                 20    Update
0169.00      C*    Check error routine will not return if error occurs
0170.00      C   20                EXSR CHKERR                        Check erorr
0171.00      C                     GOTO PROMPT                        Loop back
0172.00      C                     END                                Chg action
0173.00      C* If Dlt
0174.00      C           '4'       IFEQ ACTION                        Dlt action
0175.00      C* F3 has already been processed. If not F8, return with error
0176.00      C  N92                DO                                 Not F8
0177.00      C                     SETON                      5159    Dlt text HI
0178.00      C                     EXCPTRELESE                        Release lock
0179.00      C                     GOTO DSPLY                         Loop back
0180.00      C                     END                                Not F8
0181.00      C                     DELETMLGMSTR                       Delete
0182.00      C                     GOTO PROMPT                        Loop back
0183.00      C                     END                                Dlt action
0184.00      C* End of program routine: end the sub pgm if it was called
0185.00      C           ENDPGM    TAG                                End pgm
0186.00      C           CALLED    IFEQ 'X'                           If called
0187.00      C                     MOVE 'X'       PGMEND  1           Pgm end sw
0188.00      C                     EXSR SUBPGM                        Sub pgm
0189.00      C                     END                                If called
0190.00      C                     MOVE 'GOOD     'RTNCDE             Return code
0191.00      C           BADEND    TAG                                Bad pgm end
0192.00      C                     SETON                      LR      Set LR
0193.00      C                     RETRN                              Return
0194.00      C* Validation subroutine
0195.00      C           VALIDT    BEGSR                              Validate
0196.00      C*    Valid state abbrev and account types are checked via DDS
0197.00      C           MLACCT    CABEQ*ZEROS    DSPLY          71 Error
0198.00      C           MLSRCH    CABEQ*BLANKS   DSPLY          72 Error
0199.00      C           MLCITY    CABEQ*BLANKS   DSPLY          73 Error
0200.00      C           MLSTAT    CABEQ*BLANKS   DSPLY          74 Error
0201.00      C           MLZIP     CABEQ*ZEROS    DSPLY          75 Error
0202.00      C                     ENDSR                            Validate
```

Figure 11-5 (Part 4 of 5). RPG Specifications for MLGMTNR

```
          ...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7
0203.00      C* Check for disk write or update error
0204.00      C           CHKERR    BEGSR                           Check error
0205.00      C* Status 01021 is a duplicate key error
0206.00      C           STS       CABEQ01021     DSPLY          65 Dup key
0207.00      C                     MOVE 'DSKERR  'RTNCDE            Set error
0208.00      C                     GOTO BADEND                      Bad pgm end
0209.00      C                     ENDSR                            Check error
0210.00      C* Call sub pgm for name search
0211.00      C           SUBPGM    BEGSR                           Sub pgm
0212.00      C                     CALL 'MLGNAMR'                49 Call subpgm
0213.00      C                     PARM           SEARCH            Search field
0214.00      C                     PARM           ACCT              Account nbr
0215.00      C                     PARM           PGMEND            Pgm end sw
0216.00      C                     ENDSR                            Sub pgm
0217.00      O* Output specs for releasing a locked record and adding new key
0218.00      OMLGMSTR E               RELESE
0219.00      O        EADD            ADDNEW
0220.00      O                        MLACCT
0221.00 ** ACT   Action text
0222.00 Display   Change    Add       Delete    GE value  Search
```

Figure 11-5 (Part 5 of 5). RPG Specifications for MLGMTNR

Line 1.00: A comment.

Line 2.00: The mailing master file is defined as an *update* file (U in position 15). This means that the program can read, update, and delete records in the file (*add* is not included and must be specified in position 66). The F in position 16 means it is a full procedural file and can be processed with operation codes. The K in position 31 means the file will be processed by key meaning the file can be processed either (or both) randomly using a key field (the account number) or sequentially in keyed sequence. The device is defined as DISK for a database file.

The K entry in position 53 indicates a continuation entry. This is defined in positions 54 through 58 as INFDS meaning the file information feedback area is needed for this file. The entry in positions 60 through 65 identifies the name of the data structure to contain the feedback information. It is FILEDS and is described on line 6.00.

The A entry in position 66 specifies that the file will be added to. This entry is needed if new records will be added to the file.

Line 3.00: The display file MLGMTND is defined. The entries are similar to that used for the MLGINQR program.

Line 4.00: An E specification (extension) is needed to describe an array that will be used in the program. The array will contain the text description for the *action* field entered by the user. The user enters a code on the first display and the code is translated into a text description which is displayed on the second display.

Arrays: The array name is ACT. An array can be filled by entries from source statements or you can fill the array with calculation specifications within the program. In this case, the entries are constants and are filled by source statements in the program. The 6 entry in position 35 indicates that the array data is

included in the source of the program (see line 221.00) and that each record will contain 6 entries.

The 6 entry in position 39 specifies that there will be a maximum of 6 entries in the array. The 10 entry in positions 41 and 42 specifies that each entry has a length of 10. Therefore, the RPG compiler would expect to find one source statement (6 entries of 10 bytes each) to contain the data to be loaded into the array.

Later on in the program are the calculation specifications that will use the data that is stored in the array.

Lines 6.00−8.00: On line 6.00, the file information feedback area data structure is described as was referred to from the file specification on line 2.00.

File Feedback Area: The feedback area can contain many entries about the file such as the file and member name being processed. For example, if an override is used, the file specified in the program source may not be the file being processed and you may want to know the actual name.

Normally, you would not define all of the fields in the feedback area. For this program, only two fields in the feedback area are of interest.

The definition of the feedback area is described in the *RPG/400* User's Guide*. You essentially find the fields you are interested in, define them as described in the manual and specify field names as required by your program.

The first field is the RCDLEN field which is defined to exist in positions 125 to 126 and has 0 decimals (0 in position 52). The field is defined as B for binary in position 43. This means that the field is stored as a binary value in the data structure. RPG always processes numeric values as packed decimal values. The binary entry will cause RPG to convert the value before processing.

The second field is the special status information that is supplied on error conditions. This is defined by the special entry of *STATUS in positions 44 through 50. The field name STS is defined to contain this information.

The *STATUS field defined is implicitly defined as a 5-digit 0 decimal field. The error status codes are defined in the *RPG/400* User's Guide*. This program is only interested in one particular code and it is described later.

Lines 9.00−12.00: On line 10.00 a data structure is defined to hold the fields from the MLGMSTP file. Normally, this would not be done, but two data structures will be used to provide for solutions that would require many additional specifications to solve. These will be described later in the program.

Data Structures: A data structure allows the definition of an area of storage. The storage may contain a subdefinition of fields which make up the area. This has the advantage that the individual fields within the data structure can be processed as normal RPG fields and that a name can be given to the entire data structure. In this case, the program is interested in having a field name represent the entire record (all fields).

The first data structure name is DSRCD1. The E entry in position 17 specifies that an externally described definition will be used. The DS entry in positions 19 and 20 defines this as a data structure. The MLGMSTP entry in positions 21 through 27 identifies that the external definition should come from the file

MLGMSTP. This is the same file that is being used in the program on line 2.00. This means that any RPG operations like READ or CHAIN will cause data from the file to be read into the field names within the data structure. This is no different than normal RPG processing of externally described files except that the DSRCD1 name can now be used to move all of the fields of the record.

The second data structure (line 12.00) describes DSRCD2. It does not have any fields defined. It will be used to hold a copy of the entire record. The 300 entry in positions 48 through 51 describes a 300-byte area. This essentially creates a field name DSRCD2 which is 300 bytes long. RPG has a restriction that allows only fields of 256 bytes or less to be defined. The data structure specification can be larger.

The 300 entry is just a large value. It must be large enough to hold a record from the MLGMSTP file. The entry can be larger than the actual record length, but cannot be smaller. The program will check to ensure that the value is big enough so there is no danger of an error going undetected. The only requirement is to make the entry large enough so that the program source will not need to be changed every time the master file description is changed. From a performance viewpoint, an excessively large size should not be used because the field will be used in a compare operation.

You can determine the length of the record by using the Display File Description (DSPFD) command such as:

```
DSPFD FILE(MLGMSTP)
```

and look for the description of the record length.

Lines 13.00 and 14.00 (Parameter Lists): The PLIST operation code along with the *ENTRY value defines that this program will accept a parameter list when the program is called. The program is to be called from the CL program MLGMTNC (described earlier). In the CL program the CALL statement was used and a parameter list described. A program may have multiple parameter lists and it is necessary to describe which list will be used when the program is called. The *ENTRY value describes that this parameter list is used when the program is called.

The PARM statement identifies the parameter that will be passed at entry time. The field RTNCDE is passed. It must use the exact same definition as was defined in the CL program which in this case is a character field of length 8.

There is no checking by the system to ensure that the parameter list and attributes are identical so it is very important to specify the correct list and values.

In this case, there is only one field on the parameter list. The RPG program does not look at the value when it is passed. The RPG program is only interested in setting a value for the return.

Lines 16.00–19.00: The record length of the record from MLGMSTP is checked to ensure that it is not greater than 300 bytes. The field referred to is RCDLEN which is defined on line 7.00. It is from the file information feedback area and will contain the length of the record read. The program must ensure that this is not greater than 300 or the logic of the program will not operate correctly.

If the record size grows beyond 300, this entry must be changed as well as the value described for the data structure on line 12.00. The size of the data struc-

ture on line 10.00 will not have to be changed because it will be determined by the external definitions copied in when the program is created.

If the record length is greater than 300, an error must be signaled so the programmer can correct the program. Line 16.00 sets a value of LENERR into the return code field and the next line branches to BADEND where the program will end (described later).

This type of coding is for a very unlikely condition. The intent is to inform the programmer who can then correct the problem.

Line 19.00 describes the END operation. This is not run by the program, but marks the end of the preceding DO group. A DO group is started by a DO or IF operation and ended by an END operation. There can be nested levels of DO groups (DOs within DOs within DOs and so on). This can become confusing to read.

DO Groups: The RPG compiler notes the DO group nesting on the compilation list to assist you. However, many times you will just be in SEU reviewing your source. For this reason, the code shown uses the technique of making an entry in the comments section of each calculation statement (positions 60 through 74) to describe the beginning and end of a DO group.

Lines 20.00–24.00: The program begins by prompting the user with the DSPLY1 record format in the work station file. The EXFMT operation performs a write and a read to the display using the DDS definitions for the DSPLY1 record format found in the MLGMTND file.

Notice that the EXFMT operation specifies the record name and not the file name. RPG has a rule that all record format names used in the program must be unique. Therefore, the compiler is able to determine what to do on the EXFMT operation. RPG also has a rule that says the record format names cannot be the same as a file name. If you use RPG, you must be aware of these rules and strive for unique names to avoid these conflicts. If you use a naming convention, you will normally avoid these potential conflicts.

Prior to displaying the prompt, the program moves the ACCT field (account number) to PRVACC (the previous account number field). The previous account number is used on the first display to describe to the user what was the last account number that was worked on.

The PRVACC field is not defined in the program because it is defined in DDS and the attributes of the field will be copied into the program when the program is created. The DDS causes the previous account number to be displayed only when indicator 39 is on (plus other indicator settings). For the first use of the prompt, indicator 39 should not be on because there is no previous value. The program then sets on indicator 39, so that any subsequent re-display of the DSPLY1 record format may show the value. Indicator 39 is never set off in the program.

Lines 25.00–26.00: The program is written to end if function key 3 is pressed. The DDS entries have turned on indicator 93 if F3 was pressed. If 93 is on, a branch occurs to the ENDPGM label (described later).

Lines 27.00−31.00: If the action entered by the user is other than 1 through 6, the program will cause an error message. The IFLT followed by the ORGT operation define that line 29.00 and 30.00 will be run if the action entered is less than 1 or greater than 6.

The error message for an invalid action is conditioned by indicator 44. The program just sets this indicator on and branches back to PROMPT which causes the DSPLY1 record format to be re-displayed with the special ERRMSG DDS support.

Lines 32.00−35.00: The program must translate the action code entered by the user (such as 1) into a text description (such as Display). The text description will be displayed on the second display to provide feedback to the user as to what mode (display, update, add, delete) the display is in. The same text is also used on the first display as feedback on the last action performed.

The SETTXT tag allows the program to branch back to this point at a later step in the program (described later).

The first MOVE operation takes the character field entered by the user and converts it to a numeric value in the field AX which is defined as a decimal field of 3 digits and 0 decimals. The AX field will be used as an index to an array. RPG requires that a field used as an index must be numeric.

Note that the program has already ensured that the entry in ACTION contains a value of 1 through 6. If this check had not been made and the user had mistakenly entered * as the action, the MOVE operation will fail because an * is not a valid value to move to a decimal field. It is important to know the data values you are dealing with in a program or the program may fail.

The second MOVE operation extracts information from the array of text descriptions and moves it to the ACTTXT field. The AX field is used as an index to the array. Assume AX has a value of 3 for the add action code. The MOVE operation would cause the third element in the array (the text description Add) to be moved to ACTTXT.

Lines 36.00−47.00: If the user requests option 6 to search for a name, the maintenance program must call a subprogram (described in a later chapter) to perform the function.

Line 38.00 ensures that the user has entered a value in the field SEARCH. If not, the program branches back to display the prompt with indicator 48 on which will cause an appropriate error message.

Calling a Subprogram: The subprogram is called using an RPG internal subroutine SUBPGM (described later). The subprogram will leave its files in an open state to allow a fast re-run in case the user requests another search. Therefore, the maintenance program sets a switch (the CALLED field) which is tested when the maintenance program ends. At the end of the program, if the switch is on (meaning the subprogram was used at least once), the maintenance program will tell the subprogram that it is no longer needed and it can properly close its files.

The opening of files is a performance overhead that should be minimized to what is actually needed. By using the technique described, the subprogram's files are

not opened unless needed. If needed, they are opened only the first time the maintenance program is called.

The subprogram will pass back the value of the ACCT field.

The ACCT field may contain zeros meaning the user did not select a specific account number. If this occurs, the program branches back to display the first prompt.

If the ACCT field contains a value, the user has requested to see the detail display of a specific record. The program changes the ACTION field to 1 for display mode and uses the same two instructions from lines 34.00 and 35.00 to set the text description for the action code to appear later.

The intent of lines 44.00−46.00 is to simulate that the user entered an account number on the first display and requested a *display* action code. The subprogram is used in case the user does not know the account number and needs to search for it. The program then proceeds as if the user had specified a display action.

Lines 49.00−54.00: The program checks for the action code of 5 meaning that the user has requested to search through the file one record at a time. Whatever value was in the ACCT field is used to set a lower limit into the file. If the ACCT field was left blank on the display, the work station support will return zeros because the field is defined as a decimal field.

Set Lower Limit: The set lower limit function causes an access to the database files' index and positions the database cursor to be at a record which is equal to the account number entered or just prior to it. Thus, if 10000 was entered, the database would be positioned so that account number 10000 would be read next if it exists. If account number 10000 does not exist, then the first account number greater than 10000 would be read.

Line 51.00 causes a read to the record format MLGMSTR and indicator 47 is set if the database support finds the end of file indication. End of file means that there are no records or no more records in the file. If the user had entered account number 99999 and the account number does not exist, the first read after the set lower limit will find end of file. If end of file is sensed, the program branches back to the first display and indicator 47 is used to condition a constant which explains the end of file condition.

If a record was read (end of file was not found) the work areas for the name, address, and so on are now filled with the values for the record. The program branches to the label DSPACT to display the record.

Line 56.00: The program must display a specific error message if the user did not key a value into the ACCT field or entered all zeros. The CABEQ operation checks for this. It sets indicator 46 and branches back to the first prompt. Indicator 46 controls a specific error message.

Line 57.00: Since action codes 5 and 6 have already been handled, the remaining action codes all request an action on a specific record. The program takes the ACCT number and performs a CHAIN operation to see if the record exists. Indicator 30 is set on if the record does not exist. Sometimes this is referred to as the *not found* condition, but this can be confusing in using the Nxx statement trying to describe the *Not not found* condition. For this reason, this

manual will describe it as the *missing condition* meaning the requested key is missing in the file.

The add action code must cause an error if a record with the same account number exists (not missing). The other actions (display, update, delete) must cause an error if the account number is missing.

Program Space Considerations: There is a space overhead produced in the program for each operation specified to the database or work station. Therefore, it is generally desirable to keep the program smaller by using a single statement such as the CHAIN operation which is used by all of the action codes rather than coding unique CHAIN operations for each of the actions to be performed. The program will not run any faster with a single CHAIN statement, but the size of the program will be smaller.

Lines 59.00–79.00: The add action is checked for. If indicator 30 (missing) is not on (meaning a record already exists with the same account number), the program sets on indicator 41 and branches back to the first display. Indicator 41 is used to condition an appropriate error message.

Single Work Area Considerations: RPG uses a single work area for each field name. Therefore, when a program is both adding new records and re-accessing old records, you must be careful to ensure that the new records will be added with default values for the fields that do not come from the user. Otherwise, what can happen is that the user will retrieve a record which will fill the values for a field and if the next request is to add a new record, the new record will contain the values from the previously accessed record. A solution is to use move statements or user input fields for every field in the record.

Another solution is described in the code. The program will add a new record with just the account number as soon as a valid add request exists. The new record will be written with specific output specifications which describe which fields will be written. The EXCPT statement on line 67.00 causes output to occur. The output specifications (described later) cause a new record to be written with just the account number field. The other fields will be output with default values assigned by the database support.

When a physical file is defined, you can describe default values for fields in any new records. These values will be used if the program which uses the record does not use all of the fields. If no defaults are assigned, the system will assign blanks for character fields and zeros for decimal fields. Thus, after line 67.00 a record now exists in the database with just the new account number, and all other fields are blanks or zeros.

Line 69.00 then reads this newly written record back into the program using the CHAIN operation. The effect is to fill the work areas for the fields which describe the record with blanks or zeros with the exception of MLACCT which will have the real account number. This technique avoids the problem of the field values containing information from some previously accessed record.

This is not the most efficient technique for adding a new record. If you have a large number of additions, you should code so that only the add of the entire new record occurs.

It is very unlikely that the record just written will be missing when the CHAIN operation occurs. However, it might occur. For example, the missing condition

would occur if one user added the record and before the CHAIN operation occurred, another user deleted the record. If the record is missing, the program is coded to set an error return code and end. It would be possible to inform the user of what happened and tell the user to start over. However, this is more code and the programmer may draw a line on what he considers to be a very unlikely situation.

The program is sensitive to the fact that a record could be missing. Poor coding practice would assume that the record is always found and ignore the missing condition. This would result in either the program ending without good programmer feedback or corrupting the database. A small amount of code is normally desirable in a very unlikely situation.

Locking Records: When a record is accessed in a file which is opened for update, the system will implicitly lock the record to prevent another user from accessing it at the same time for update. If the second user opened the file for input, he can read the record even if it is locked. A record is unlocked by:

- Updating the record.
- Reading another record in the same file. Only one record can be locked from a file per job. (There is an exception to this if commitment control is being used.)
- Using the release lock function.

The record locking rules are a good integrity feature. Without the locking protocol, it would be possible for two users to access the same record simultaneously. If both users changed the record and then updated it, the first update would be totally lost.

While locking of records is a good integrity feature, it can also cause problems where a second user is attempting to access a locked record. If this occurs, the system waits for a period of time (default is 60 seconds). If the lock has not been released during the time, the program will receive an exception. If the exception is not coded for, the program will abnormally end. While this is good from an integrity viewpoint, it is not good from a human factors viewpoint. While the system is waiting for the lock to be released, the user would receive no feedback. The job would appear to be hanging.

To avoid this problem, it is normally desirable to release record locks while the program is waiting for user input. The user may take a long time to decide what to do because of interruptions or the user may go out to lunch for an hour.

Releasing Record Locks: Releasing a record lock is done by causing an output line to the file which has no fields specified. This internally tells RPG to signal the release lock function to the database. No write actually occurs to the disk file (only the internal lock is released). On line 75.00, the EXCPT operation to the output record RELESE does this function.

Releasing the record lock avoids the problem of other users being locked out, but creates the potential problem of multiple users updating the same record with different information.

While it is unlikely that a second user would access the same record for update before the first user had updated it, this condition can occur frequently enough that it normally should be programmed for. What is needed is a solution that will

tell the user that the contents of the record have changed since the user first started the action. When a new record is added and the lock released, there is nothing to prevent another user from finding the record and updating it. To guard against this, two data structures are defined in the program and are used to compare the first version with the second version (described later).

On line 76.00 the contents of the first data structure (the newly written record added to the database) are moved into the second data structure. Note that this is a MOVEL operation because the MOVE operation will move from the right which will not provide a proper comparison on a subsequent operation. A MOVE operation could be used if the lengths of the two data structures are the same. However, in this case, they are not. The first data structure has the exact length of the record because it is the sum of the fields compiled into the program using externally described data. The second data structure is 300 bytes long which is an arbitrary value.

Later on in the program, a comparison will be made between the two data structures.

Line 77.00 sets an indicator to describe that the account number field should be protected and the program branches to display the record. Indicator 53 is used with the DSPATR(PR) keyword in DDS to prevent the user from changing the key on a newly added record. The user is allowed to end the add request and not add the record. In this case, the program must delete the record since it already exists in the database (described later).

When you are adding a new record, you may want to display defaults for the user for some of the fields. In this program, all of the fields displayed did not have a default and, therefore, should appear as blanks or zeros when the user is ready to add a new record. In the work station DDS, the fields to be entered are all described as both fields meaning the contents from the program will be shown on the display.

Since the record was actually added and then re-accessed, what the user will see is the default record written to the database. To provide defaults for some of the fields, you could either assign defaults in the database DDS (described earlier) or use the MOVE operation prior to displaying the record.

Lines 81.00–84.00: At this point, the only actions left to be processed are display, update, and delete. The user has entered an account number and line 57.00 did the CHAIN operation to access the existing record. If indicator 30 is on, a missing condition exists (account number is not in the file) and an error message is needed. Indicator 42 is set on and a branch occurs to re-display the first prompt.

Lines 86.00 and 87.00: To avoid locking the record while the user is looking at the details, the record lock is released and the contents of the record that the user is viewing are saved in the second data structure. This is the same code as was described on lines 75.00 and 76.00.

Lines 89.00–93.00: If the delete action was specified, the program sets on indicators 50 and 53 to protect the fields on the display. The user will be able to see all of the fields in the record, but not change any of them.

Indicator 51 is then set to cause the special text to be displayed that describes how to delete a record. A branch occurs to display the record.

Lines 95.00−101.00: If either action 1 or action 5 was specified, the program will now display the contents of the record that is in main storage. Indicators 50 and 53 are set on to protect the fields from being changed. Indicator 40 is set on to allow additional function keys and the program branches to display the record.

Line 103.00: If action 2 was specified, the user wants to change the record that is in main storage and the program branches to display the record. Indicators 50 and 53 will be off so the fields on the display will be input capable (meaning the user can change anything on the display).

The DDS for the various fields (such as MLADDR) are specified as *both* fields. This causes the contents of the fields in the program to be displayed to the user and allows a change to the values.

Lines 105.00 and 106.00: If the program reaches this point, an internal error has occurred. The programmer thinks all of the action codes have been processed, but something has gone wrong. Rather than let the next instruction occur, the program is set to end with a specific error. This is good coding practice to protect against falling through when it is not expected. This is valuable when initially debugging the program and also when maintaining the program.

Lines 108.00−110.00: The TAG statement identifies the branch point for the DSPLY label. The second format DSPLY2 in the work station file is displayed and the program waits for a response. After the user presses the Enter key or a valid Function key, the program saves the contents of the MLNAME field to the PRVNAM field. This allows it to be displayed on the first display to describe the account that was operated on. There is no definition of PRVNAM in the program. It is defined in DDS using the REFFLD keyword to extract the same attributes as the MLNAME field. If the definition of the MLNAME field changes, the program can probably be re-created without any coding changes.

The LC values on the CHECK keyword allows entering in lowercase.

Line 112.00: The program checks for indicator 93 which is set for either F3 (Exit) or F12 (Cancel). If the action is other than add, the program branches back to the first display. If the action is add, the user is requesting to end the add request. However because of the program logic, a record has already been added to the database and must be deleted (described later).

Lines 114.00−121.00: The roll keys are not valid on every action type, but the code is written so they are checked here regardless of the action type. Indicator 97 is set if the user requested rollup and indicator 98 is set for rolldown. If either indicator is on, the DO group is run.

If rollup (97) was requested, the user wants the next record in the database so the READ operation is performed. For rolldown (98), the READP operation (read previous) is performed. Both operations set indicator 45 as end of file. If end of file occurred, the program branches back to the first display and indicator 45 conditions an appropriate message.

If a record was found, indicators 50 and 53 are set for protecting the fields and the program branches back to display the record.

Lines 123.00 and 124.00: If the roll keys were not pressed, the program saves the account number field which was read and sets off indicator 40 which controls the additional function keys.

Lines 125.00−129.00: F6 is not valid on every action type, but the program tests regardless. F6 is set from the display of a single record to mean the user wants to go into change mode on the record that is currently displayed. F6 in DDS sets on indicator 96. If 96 is on, the program changes the action to 2 (meaning the change type) and branches back to the SETTXT label. The program will now proceed as if the user had requested the change action on the first display.

Lines 130.00−133.00: The program always checks for the F11 key (indicator 91) which is used to request delete mode for the record being displayed, even though F11 is not valid on every display.

If 91 is on, the action type is set to 4 for delete and the program branches to SETTXT. The program will proceed as if the user had requested delete on the first display.

Lines 135.00 and 136.00: The program checks for the two display action codes (1 and 5). Since the record has already been displayed and the special function keys already processed, the program branches back to prompt for the first display.

Lines 138.00 and 139.00: Only the add, update, and delete requests remain to be processed. The program must re-access the record to allow the update to occur because the record lock was released previously. An update or normal delete cannot be performed unless the record is accessed and the lock is held. The CHAIN operation re-accesses the record.

It is possible that some other user accessed the same record and deleted it before the current user responded. If so, the record will be missing and indicator 43 will be set on. If 43 is on, the program branches to the PROMPT display and 43 is used to condition an appropriate message.

Lines 141.00−146.00: The discussion for line 75.00 described the problems of releasing the record lock and then protecting against a change from some other user. The CHAIN operation on line 138.00 has filled the data structure DSRCD1 with the current contents of the record. Now the two data structures are compared to determine if any change occurred between the first access and the CHAIN just run.

If a change has occurred (an unequal comparison), the record lock is released, the record from DSRCD1 is moved by MOVEL to the second data structure, indicator 64 is set on to describe the error, and the program branches back to display the new contents. The releasing of the lock and moving of the record is essentially a repeat of the code that was done when the record was initially read.

If you are interested in testing this condition, you could use the system request function described at the end of this chapter.

Line 149.00: When the EXFMT operation occurred on line 109.00, the RPG work areas which represent the field names would be filled with the values as they appeared on the display. They are now ready to update the database, but because the record was released, the CHAIN operation on line 138.00 had to be issued to re-access the same record. This has the unfortunate result of overlaying all of the fields that the user keyed and the original database record is now back in main storage.

The use of RPG work areas works well for many operations, but this is a case where it does not. However, the system supports a unique function to be able to re-access the information from the display. This is the function of the RTNDTA keyword which was described in DDS for this record format.

RTNDTA Keyword: When RTNDTA is specified, it tells the system that if successive read operations occur, the data from the work station should be read from the main storage contents for the second and subsequent reads. When the EXFMT occurred on line 109.00, the system knows to read from the work station device because a write operation occurred prior to the read. When the READ occurs on line 149.00, the RTNDTA keyword tells the system to return the same data. The system does not go back to the work station device. Instead it accesses the data that it has been holding in main storage since the first read and returns it to the program.

Consequently, the steps appear as follows for a change action.

- The user requests a record from the database and the contents are read into main storage. The record lock is released. The record contents are displayed on the work station.
- The user changes the fields on the display and presses Enter. The fields in main storage are overlaid and now contain the results of what the user wants the database record to be.
- The record is re-accessed from the database. If no other job changed the record, the field values now represent what the user saw before making the change.
- The READ operation in conjunction with the RTNDTA keyword re-accesses what the user keyed and the field values now represent what the user wants the database record to be.
- The record is then updated as desired by the user.

Remember that the whole reason for this type of coding was the fact that the program did not want to keep the lock on the record while the user was deciding what to do. The program would be a lot simpler if we assume that the lock can be held while the user is making a decision. In a database environment with many users, locking records during user decision time can cause significant problems.

The user decision time or think time can be a long period of time. When a read operation occurs to the work station, the default is to lock the keyboard after the user responds. This means the user cannot use the keyboard until the next write occurs. The default for a write operation is to unlock the keyboard after the information is sent to the display. In general, it is a good idea to try and minimize or delete the database record locks that are held when the keyboard is unlocked.

Notice indicator 20 on the READ operation on line 149.00. RPG requires an indicator for end of file on any READ. Because end of file will never occur to the work station device, the indicator can be ignored in the program.

Lines 151.00–155.00: If an add request occurred (action 3), the user is allowed to exit through F3 and not add a record. However, the program has already added a record to the file and now must delete this record. The record has

already been re-accessed by line 138.00. The DELET operation is used and the program branches back to the first display.

Line 157.00: Before adding or updating a record in the database, it is normal to validate the information entered by the user.

The system has prevented the user from typing a bad value into the state field and account type fields. This was checked using the DDS VALUES keyword. Other fields may need to be checked. You may also have relational things to check such as you may not allow one field to contain certain values if a second field contains a specific value. Only your application can decide how to make these validations.

Validity Checking Subroutine: In this program design, the validity checking is all placed in an internal RPG subroutine. This is not necessary, but is done for two reasons:

- The same validity checking routine will be used from multiple points in the program so coding it only once is a better choice.
- Subroutines allow the code to be logically separated which can help documentation.

Normally you would code a subroutine to return to the next instruction following the EXSR statement. However, in this case if any errors are found, the subroutine will branch to the DSPLY tag.

Lines 159.00−163.00: If the validity checking subroutine does not find any errors, the program updates the record. The UPDAT operation collects all of the fields defined for the record and changes the record in the database. No output specifications are needed.

The UPDATE operation is performed for the add type action. The add is handled this way because the outline record (just the account number) was already written to the database record.

Indicator 20 is set if the database is not able to add the record. There are many potential errors, but it is normal to code for only a few expected exceptions. If indicator 20 is on, the program runs the internal subroutine CHKERR. The program is coded so that it will not return to the next instruction following the EXSR to CHKERR. This is because the same error routine is also called from another point in the same program.

If the record is successfully updated, the program branches to PROMPT for the next action.

Lines 165.00−172.00: The handling of the change request is similar to the add. The information is validated, the record is updated, and the potential error indicator from the UPDAT operation is checked.

Lines 174.00−183.00: A delete action has already handled the case of F12 (Cancel) to end the delete request. The prompt for the delete requires the user to press F8 to delete the record. If the user presses the Enter key, the program sets on 51 and 59 to cause the same prompt text explaining how to delete a record to appear again and this time in a highlighted manner. The program releases the lock and branches to display the record.

Delete Confirmation: Some program designs may allow the user to just press the Enter key to confirm the delete. Another choice would be to not ask for a confirmation, but to delete the record based on the information from the first display. While this is a question of program design, many users find it too easy to delete information from the system. Providing a confirmation option that is unique (other than the Enter key) can be helpful in preventing careless errors.

If the user pressed F8 to cause a delete, the program uses the DELET operation on line 181.00 to delete the record. The program returns to the first prompt.

When the DELET occurs, the database will prevent any further access to the record. There is no way to retrieve the original contents. (There is an exception to this with journaling. See the later discussion.) When a record is deleted, the space still exists on the system for the record. It is possible to re-use the space, but the normal method of reclaiming the space is to periodically reorganize the file. This is done with the RGZPFM command.

Lines 185.00—190.00: The ENDPGM tag is the label to be branched to when the user requested F3 on the first display. See line 26.00.

The program checks to see if the subprogram for name search was called by testing the CALLED switch. This was described on line 40.00. If the subprogram was called, an X is moved to the field named PGMEND and the internal subroutine SUBPGM is called. This will cause the subprogram to be called, which will sense the parameter passed for PGMEND and properly close.

The RTNCDE field is set to GOOD. This will be passed back to the CL program which called this RPG program. The program then proceeds to the next instruction.

Lines 191.00—193.00: The BADEND tag is the label for the very unlikely errors to branch to. For either a good or bad end of the program, the program sets on LR and returns with the RETRN operation. The LR indicator tells RPG to close the files and properly end the program. If the program is called again, the files will have to be re-opened and the work areas will have to be re-initialized.

Lines 195.00—202.00: All internal RPG subroutines begin with a BEGSR operation that describes the name of the subroutine and end with the ENDSR operation.

Input Validation: The VALIDT subroutine validates the fields that were keyed by the user. It is used for both adding new records and changing existing records. The valid entries for account type and state are handled implicitly by the DDS use of the VALUES keyword.

DDS also allows checking to test for non-blank fields which can be used in some cases to avoid some of the tests that were made in these statements. However, the DDS checking cannot be used in all cases and the program is coded to ensure non-blank entries.

It is normal to have more elaborate checking. For example, you might want to check to see if the zip code is valid for the state that was entered. This could be done by entering array data for the valid states and a range of valid zip codes.

Lines 204.00—209.00: The CHKERR subroutine is run if the error indicator was on following either update operation.

Database Errors: Most of the database errors are passed back as a series of codes that can be checked. In this case the STS field defined in the file information feedback data structure is checked to see if it contains 01021. This is the value that is set if the system signals to RPG that a duplicate key has been attempted to be added.

When the MLGMSTP file was created, the DDS UNIQUE keyword was specified meaning that an error should be signaled if a key field is attempted to be added which already exists. It is normal for most files that use a code like account number to want to ensure that the key is unique in the master file (two records should never exist for the same account number).

It is unlikely that this error will occur on an add. The program has already checked to see if the requested account number was unique and then added a record with the unique key.

It is possible that if the user decided to change the account number, a change could occur that would cause a non-unique key. If this happens, the program sets on indicator 65 and branches back to the display of the record. Indicator 65 is used to condition appropriate error text.

If some other error occurs, the program turns it into a very unlikely error by setting a return code and branching to BADEND.

A typical error that might occur would be if the user was not authorized to change a record in the database. This condition will be checked for in the CL program described in Chapter 15, "Additional Topics."

Lines 211.00−216.00: The SUBPGM subroutine is used to call the name search program (described in another chapter). If the name search program fails (or does not exist), indicator 49 is set on. The program branches back to the first display. If indicator 49 is on, an appropriate error text appears.

Line 218.00: The first output specification identifies an exception output line. This will only be run when the EXCPT operation code is used. The label in positions 32 through 37 (RELESE) describes that this will be used when the EXCPT operation names the label.

The output line with no fields described, tells the system to use the record release function. It was described earlier on line 75.00.

Lines 219.00−220.00: The second output line specifies another exception line which will be output by the EXCPT to ADDNEW. This was described on line 67.00. This adds a new record with just the account number field which is described on the next line.

Note that there are no output specifications for the update or delete operations. These are performed by calculation operations which work on the entire record. For example, when the UPDAT operation is requested, all of the fields defined for the record are gathered up and written to the database.

Lines 221.00 and 222.00: The ** entry in positions 1 and 2 indicate that this is the beginning of array data that was specified on the extension specification on line 4.00. The following records (in this case only 1 record) provide the array data. The data for the array describes the text descriptions that will be used to describe the actions requested. The program must translate the action code (for

example, 1) into a text description using this array data. Each text description is defined to be 10 bytes in length. It is important to enter the exact number of characters for each text description.

Step 3. Entering the RPG Specifications and Creating the RPG Program (MLGMTNR)

You should now enter the RPG specifications as shown in Figure 11-5 on page 11-13 through PDM. Follow the same steps you used to create the MLGMSTP physical file in Chapter 5, "Creating Files" or copy the source from the QUSRTOOL library as described in Appendix B, "Overview of QUSRTOOL Library." The member source type should be RPG.

Once you have entered the RPG specifications, create the program by typing a 14 (Compile) in the *Opt* column next to the program on the Work with Members Using PDM display.

Step 4. Running the Maintenance Program (MLGMTNC)

When you get a completion message that the MLGMTNR program has been compiled, you can run the program. The MLGMTNR program should be run by using the MLGMTNC CL program. Do not directly call the MLGMTNR program. Call the maintenance program by typing the command:

```
CALL MLGMTNC
```

When the Maintain Mailing List Master display is shown, you can select the different options to see what functions are available from the display. Press F3 twice to exit when you are done.

Error Handling

Programmers spend a good deal of their time coding for what-if situations. The main line instructions that actually do the work are normally a small part of any maintenance program (and particularly with externally described data).

You have a choice when considering error handling to do the following and all are illustrated in the maintenance function:

- Perform some checking in CL such as the check that is made from the MLGMNUC program (shown in Chapter 12, "Creating Menus") to ensure that the user is authorized to update the file before calling the maintenance program. This solution provides checking outside of the main processing program.
- Let the system handle certain keying errors such as keying alphabetic characters into a numeric field or using the system validity checking functions such as the DDS VALUES keyword to check the contents of the account type field. There are no program statements needed for this type of checking.
- Program to handle the error and provide error messages and recovery for the user. This is the case for such situations as a missing record, or an invalid selection on the first display (such as option 8 was entered).
- Handling very unlikely cases by providing minimal feedback. This is the case with the non-GOOD value for the return code. The program recognizes a problem and chooses not to recover or cannot recover. However, it does

not let the user continue. The solution used in this program is to pass back a return code that causes the MLGMTNC program to abnormally end.

This can be a reasonable solution when you are writing a new program. As you are testing, you can try some typical types of errors and improve the error handling. If the program goes into production and starts experiencing problems for what you thought to be a very unlikely condition, you can change the program to be able to recover.

- Not coding for catastrophic failures such as when the MLGMSTP file does not exist or the display file does not exist. In these cases the system will signal an exception and since it is not monitored for in the code shown, the job will probably abnormally end or provide messages as to what is missing.

  It is normal to not code for most of these situations. However, you may have a few special cases that need to be considered.

In general, if an error occurs that you did not monitor for, the program will stop the function and send an exception message. This may cause your job to abnormally end.

The AS/400 system provides exception handling functions to allow you to handle errors and has many simple functions to allow you to do a good job of checking the user input and providing error messages.

If you choose to monitor for an error condition, then it is your responsibility to handle the error. The system will go on to your next instruction. If you monitor for an exception and then choose to ignore it, your situation can be a lot worse than if you had not monitored for the exception.

System Request Function

The system request function has some good uses that you should be familiar with. You can use the system request function to try several of the functions to determine what happens when two users access the same record at the same time. This is particularly important for the maintenance program. The system also supports a group job function which is similar to system request, but offers additional functions.

The System Request menu has several options. You must be authorized to the menu and to perform the option requested. One of the options is to transfer to a secondary job. A secondary job means that you are signed on the system in two interactive jobs using the same work station.

Assume the job you are on now is Job 1 and you want to transfer to Job 2.

1. Press the System Request key. A line will appear at the bottom. You can type on this line, but to see the System Request menu, press the Enter key. The system request menu will appear.

```
                                System Request
                                                          System:   RCH38360

Select one of the following:
     1. Display sign on for alternative job
     2. End previous request
     3. Display current job
     4. Display messages
     5. Send a message
     6. Display system operator messages

    80. Disconnect job

    90. Sign off




                                                                        Bottom
Selection
     —

F3=Exit    F12=Cancel
                                          (C) COPYRIGHT IBM CORP. 1980, 1989.
```

2. Select option 1 to transfer to a secondary job.

   Note: The security officer may have revoked your authority to the Transfer Secondary Job (TFRSECJOB) command. In this case, you will need another user to assist you in doing the rest of this section.

3. The signon prompt will appear. Type your name and password and press the Enter key. Your initial menu or initial display will appear. You can then request a CALL to MLGMNUC and select the maintenance function. Try accessing a specific account number for change. After the detail display appears, press the System Request key.

4. A line on the bottom of the display will appear. Press the Enter key and the System Request Menu will appear. Select option 1 to transfer to a secondary job. You can only be signed on to two jobs using this technique. Because you are already signed on to Job 1, the system will transfer you back to the display that you were on when you pressed the System Request key initially.

5. Access the maintenance program in the same manner. Then access the same account number for change, change one of the values on the display, and press the Enter key.

6. You should return to the first prompt for the maintenance program. Now press System Request again. Use option 1 to transfer back Job 2. Now try to change one of the fields and press the Enter key. You should receive an error message describing what happened and how to recover.

7. You should be on Job 2. If you want to return to Job 1, you can either sign off or use the system request/transfer to secondary job function again.

Chapter 12. Creating Menus

Now that you have created a number of different programs to perform various functions, you can create a menu that will bring all of those functions together. By creating a menu, you can provide users with a simple way to do various functions without having to know how to call programs and remember the names of the programs.

Overview of the Mailing List Menu

The Mailing List Menu allows you to perform several different functions by selecting an option off of the menu. You can call several of the different programs you have already created or even start Query. Figure 12-1 shows an overview of how the program is called and what displays are shown.

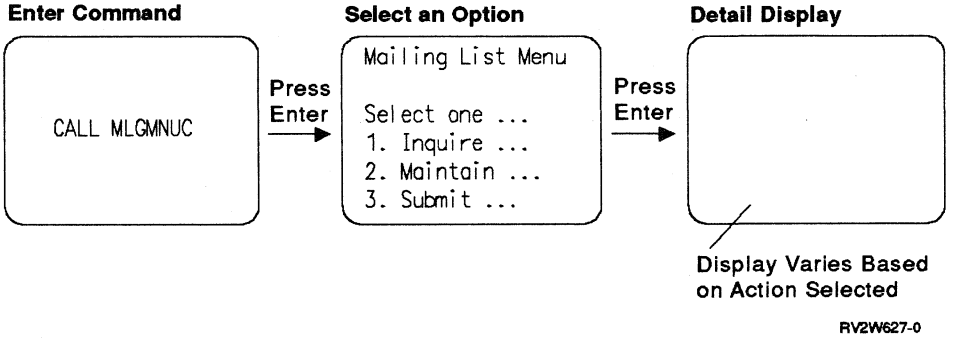


Figure 12-1. Overview of Mailing List Menu Displays

The Mailing List Menu contains the following options for the user:

```
                    Mailing List Menu
                                                    System:  RCH38342
Select one of the following

     1. Inquire into Mailing List Master
     2. Maintain Mailing List Master
     3. Submit listing by account number
     4. Submit special analysis report
     5. Query Mailing List file









Selection
===> 1


F3=Exit
```

Methods of Creating Menus

There are two typical methods of creating a menu:

- SDA. SDA allows an interactive approach to designing menus. The output of SDA is a display file and a menu object. SDA generates a source member for each function. This menu can be run using the GO command.

  As described earlier, SDA allows an interactive screen design solution and can best be understood by knowing the type of DDS statements that must be used.

- CL program and a display file. This approach requires a program to be written in CL and a display file to be created. The menu is run by using the CALL command. The display file can be created by either directly entering DDS or by using SDA.

This chapter describes the CL program and DDS entry method.

The structure of the Mailing List Menu consists of the following:

- A display file, MLGMNUD, that contains the DDS describing the display format.
- A CL program, MLGMNUC, that is called to bring up the menu.

An overview of the Mailing List Menu structure is shown in Figure 12-2.

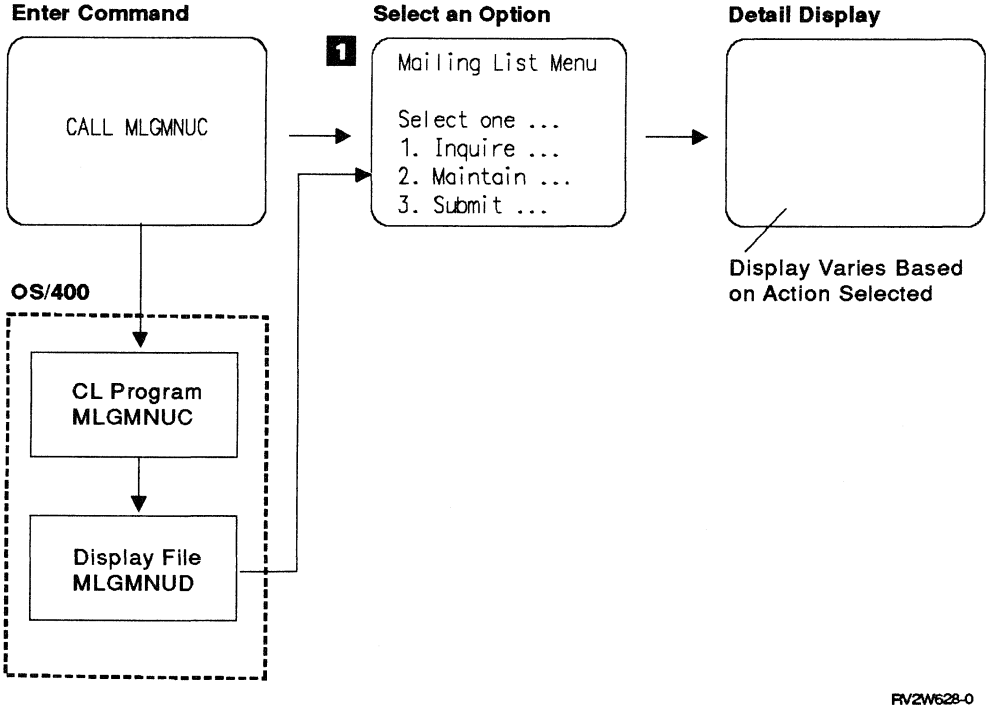


Figure 12-2. Overview of Mailing List Menu Program MLGMNUC

1 The options on the Mailing List Menu are as follows:

Option	Action
1	Calls MLGINQR
2	Calls MLGMTNC
3	Submits job MLGRPTC to call MLGRPTC
4	Submits job MLGRPT to call MLGRPTC2

Description of DDS for Menu Display File (MLGMNUD)

The Mailing List Menu must be created before the CL program that uses the display is created. The CL program declares the use of the file and it must exist to allow a successful creation of the CL program. Figure 12-3 shows the DDS for the MLGMNUD display file. A description of each line follows the figure.

```
        ...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7
0001.00     A* MLGMNUD - Mailing list menu - used by MLGMNUC
0002.00     A                                      PRINT
0003.00     A          R MENU                      TEXT('Mailing menu')
0004.00     A                                      CA03(93 'Exit')
0005.00     A                                      SETOFF(71 'Bad select')
0006.00     A                                      SETOFF(51 'Not auth')
0007.00     A                                    1 25'Mailing List Menu'
0008.00     A                                      DSPATR(HI)
0009.00     A                                    2 64'System:'
0010.00     A            SYSNAM         8          +2
0011.00     A                                    3  2'Select one of the +
0012.00     A                                       following'
0013.00     A                                    5  7'1. Inquire into Mailing +
0014.00     A                                       List Master'
0015.00     A                                    6  7'2. Maintain Mailing +
0016.00     A                                       List Master'
0017.00     A                                    7  7'3. Submit listing by +
0018.00     A                                       account number'
0019.00     A                                    8  7'4. Submit special +
0020.00     A                                       analysis report'
0021.00     A                                    9  7'5. Query Mailing List +
0022.00     A                                       file'
0023.00     A  51                               16  2'You are not authorized +
0024.00     A                                       to change the master +
0025.00     A                                       file.'
0026.00     A                                       DSPATR(HI)
0027.00     A                                   19  2'Selection'
0028.00     A                                   20  2'===>'
0029.00     A            SELECT         2   B    +1
0030.00     A                                   23  2'F3=Exit'
0031.00     A  71                               24  2'Specified menu option +
0032.00     A                                       not correct'
0033.00     A                                       DSPATR(HI)
```

Figure 12-3. DDS for Display File MLGMNUD

Line 1.00: A comment.

Line 2.00: The PRINT keyword is used. This allows printing the menu if the Print key is specified.

Line 3.00: The MENU format is described.

Lines 4.00−7.00: The F3 function key is defined to allow the operator to end the menu. The SETOFF keywords identify some indicators to be set off when a read occurs to the format. These are error indicators which will be set on in the program to cause error messages.

Lines 8.00–10.00: The menu title is displayed on line 1. The system name (the name of the system you are operating on) is also displayed. This is consistent with the way IBM menus are displayed.

Lines 11.00–22.00: Constants are described to specify what the operator should do and what the choices are.

Lines 23.00–26.00: An error message is described conditioned by indicator 51. The program will set this indicator if the user selects the maintenance program but is not authorized to update the MLGMSTP file.

Lines 27.00–29.00: The SELECT field is defined. It would be possible to use the values keyword to edit the valid responses from the user. Instead, another constant is used as an error message and the program will cause the check. This approach is used to emulate what occurs on a similar error on an IBM-supplied menu.

Line 30.00: The F3 key option is described by a constant.

Lines 31.00–33.00: An error condition is described for the case of the user entering a bad selection.

Step 1. Entering the DDS and Creating the Menu Display File (MLGMNUD)

You should now enter the DDS as shown in Figure 12-3 on page 12-3 through PDM. Follow the same steps you used to create the MLGMSTP physical file in Chapter 5, "Creating Files" or copy the source from the QUSRTOOL library as described in Appendix B, "Overview of QUSRTOOL Library." The member source type should be DSPF.

Once you have entered the DDS, create the file by typing a 14 (Compile) in the *Opt* column next to the file on the Work with Members Using PDM display.

You need to receive the completion message that the file was successfully created before the CL program MLGMNUC can be created.

Overview of CL Menu Program

The CL program MLGMNUC is called to bring up the menu.

Description of CL Menu Program (MLGMNUC)

Now that you have created the MLGMNUD display file, you can create the MLGMNUC CL program. Figure 12-4 shows the coding for the program. A description of each line follows the figure.

```
        ...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7
0001.00 /* MLGMNUC - Mailing list menu                                        */
0002.00             PGM
0003.00             DCLF        MLGMNUD
0004.00             DCL         &SYSNAM *CHAR LEN(8)
0005.00             RTVNETA     SYSNAME(&SYSNAM)
0006.00 LOOP:       CHGVAR      &SELECT ' '
0007.00 DISPLAY:    SNDRCVF     /* Prompt with menu and wait for response */
0008.00             IF          (&IN93 *EQ '1') DO /* CF3 */
0009.00             RETURN      /* Normal end of program */
0010.00             ENDDO       /* CF3 */
0011.00 SELECT1:    IF          ((&SELECT *EQ '1 ') *OR +
0012.00                           (&SELECT *EQ ' 1')) DO /* Select 1 */
0013.00             CALL        MLGINQR
0014.00             GOTO        LOOP
0015.00             ENDDO       /* Select 1 */
0016.00 SELECT2:    IF          ((&SELECT *EQ '2 ') *OR +
0017.00                           (&SELECT *EQ ' 2')) DO /* Select 2 */
0018.00             CHKOBJ      OBJ(MLGMSTP) OBJTYPE(*FILE) AUT(*CHANGE)
0019.00             MONMSG      MSGID(CPF9802) EXEC(DO) /* Not authorized */
0020.00             CHGVAR      &IN51 '1' /* Set for not auth message */
0021.00             GOTO        DISPLAY
0022.00             ENDDO       /* Not authorized */
0023.00             CALL        MLGMTNC
0024.00             GOTO        LOOP
0025.00             ENDDO       /* Select 2 */
0026.00 SELECT3:    IF          ((&SELECT *EQ '3 ') *OR +
0027.00                           (&SELECT *EQ ' 3')) DO /* Select 3 */
0028.00             SBMJOB      JOB(MLGRPT) RQSDTA('CALL MLGRPTC')
0029.00             GOTO        LOOP
0030.00             ENDDO       /* Select 3 */
0031.00 SELECT4:    IF          ((&SELECT *EQ '4 ') *OR +
0032.00                           (&SELECT *EQ ' 4')) DO /* Select 4 */
0033.00             SBMJOB      JOB(MLGRPT) RQSDTA('CALL MLGRPTC2')
0034.00             GOTO        LOOP
0035.00             ENDDO       /* Select 4 */
0036.00 SELECT5:    IF          ((&SELECT *EQ '5 ') *OR +
0037.00                           (&SELECT *EQ ' 5')) DO /* Select 5 */
0038.00             STRQRY
0039.00             GOTO        LOOP
0040.00             ENDDO       /* Select 5 */
0041.00                         /* Bad selection, set error message */
0042.00             CHGVAR      &IN71 '1'
0043.00             GOTO        DISPLAY
0044.00             ENDPGM
```

Figure 12-4. Description of CL Program MLGMNUC

Lines 1.00 and 2.00: A comment is used and the PGM statement describes the beginning of the program.

Line 3.00: The DCLF statement describes that the file MLGMNUD will be used in the program. This will cause the CL compiler to extract the definitions of the MLGMNUD file and define them to the program. This entry is similar to the RPG E entry on the file specification to indicate externally described data.

Lines 4.00 and 5.00: The DCL statement describes a variable named &SYSNAM. It is a character variable of length 8.

System Name: The Retrieve Network Attributes (RTVNETA) command will retrieve the system name which is stored as a network attribute. The system name network attribute is normally set when your system is installed. The intent of this entry is to emulate what the IBM-supplied menus do. They display the name of the system that the user is currently operating on.

In a single system environment, this entry is not very helpful because it always says the same name. If, however, the operator is dealing with multiple systems in a pass-through environment, having the system name on displays can be very helpful.

Lines 6.00 and 7.00: This line begins the basic display of the menu. The label LOOP is used later on in the program each time the menu should be displayed. The &SELECT variable which is defined in DDS is blanked out. Therefore, it will normally be blank when the operator sees the display.

The SNDRCVF command is similar to the RPG EXFMT operation. It does a write and then a read to the display. The program waits for the user's response. No format name is needed because the MLGMNUD file has only a single format.

Lines 8.00–10.00: The F3 function key (indicator 93) is checked for. The two-digit indicators used in DDS are implicitly named &INxx in CL and defined as 1-character fields. The value will be *0* if the indicator is off and *1* if the indicator is on. Therefore, to work with DDS indicators in CL, you will normally see statements that move a *1* to &INxx or test &INxx for a *1* or *0*.

If the indicator is on, the program performs a DO. The DO group only contains the RETURN statement which will cause the program to end normally. It would have been possible to code the IF as:

```
IF (&IN93 *EQ '1') RETURN
```

and avoid the DO group.

Variables for On and Off Switches: Some programmers prefer to specify two DCL statements such as:

```
DCL &OFF *CHAR LEN(1) VALUE('0')
DCL &ON *CHAR LEN(1) VALUE('1')
```

in every program using a display file. The VALUE parameter starts the value of the variable. This allows the programmer to write a more easily readable statement such as:

```
IF (&IN93 *EQ &ON) RETURN
```

Lines 11.00−15.00: The selection of a 1 entry is tested for. Note that the code allows the user to type the selection with either a blank before the entry or following it. The IF statement compares for both types of entries. The MLGINQR program is called for an inquiry. When the program ends, the GOTO statement is run which branches back to display the menu again.

The entry SELECT1: beginning in position 2 is a label. A label may be used as a branch point (such as on line 6.00) or just to help describe the program. This label is just used for documentation.

Lines 16.00−25.00: Selection 2 is tested to run the maintenance program.

Authority Checking: Before calling the program, the CL statement CHKOBJ is used to determine if the user is authorized to change the file. The CHKOBJ command identifies the object to be checked, the fact that it is a *FILE object type and requests to know if the user is authorized for *CHANGE. The *CHANGE entry corresponds to what can be specified on the Grant Object Authority (GRTOBJAUT) command to specify who can change the object. See "Granting Authorizations" on page 15-6 for more information on authorizing users to the file.

MONMSG Command: The MONMSG (Monitor Message) command follows the CHKOBJ command and is used to handle the exception if it occurs. The system will signal an exception if the user is not authorized. It will not signal anything if the user is authorized.

The MONMSG command identifies that it is interested in the CPF9802 message identification (known as MSGID). This is the exception message that will be sent if the user is not authorized as specified on the CHKOBJ command. Most commands have one or more exceptions that they may send if errors occur. If you do not monitor for them, the CL program will abnormally end.

Many messages exist in the system to describe various conditions. Messages exist in message files. Each message file must have unique MSGIDs. The OS/400 messages are stored in the QCPFMSG message file.

When you monitor for an exception, you are saying to the system *let me take care of it*. It is up to your processing logic to handle the condition. If your program did not prevent the user from calling the maintenance program if he is not authorized, the system will prevent any changes by the user. However, because the RPG program is not coded to monitor for this type of exception, the program would abnormally end. Unless the CL program is coded to handle an exception from RPG, it would also abnormally end.

Any abnormal ending should be avoided for normal user activity. Therefore, good coding practice would check for authority, monitor for the exception, and present the user with an understandable error message.

The CL program only monitors for this specific exception. It does not check for nor monitor for programs not being in existence or abnormally ending. There is not much the CL program can do for these cases other than prevent an abnormal ending of itself. Most coding techniques operate on this basis.

You must decide what your strategy will be. It is normal to initially code the program with a few MONMSG commands and then add to the program as experience is gained with the use of the program.

The MONMSG command causes a DO group to be run if an exception occurred for CPF9802. The CHGVAR command sets on indicator 51 and the program branches to DISPLAY which displays the menu again. Indicator 51 will cause the error message to be displayed. Note that the GOTO is to the label DISPLAY which occurs after the LOOP label. This will cause the &SELECT variable to remain as it was keyed so the user can see the entry he keyed as well as the error message.

If an exception did not occur on the CHKOBJ command, the CL program will implicitly branch to line 23.00. This calls the maintenance program. When the user ends the program, the GOTO command would be run to branch to LOOP and display the menu again.

If the user is authorized, the MLGMTNC program is called. When the program ends, the menu is displayed again.

Lines 26.00−35.00: The 3 and 4 selections are handled by the Submit Job (SBMJOB) command. This causes a batch job to be submitted as if the user had entered the command himself.

Note that in a previous chapter when the SBMJOB command was used, the CMD parameter was used to specify the command to be run in batch. When the SBMJOB command is used in a CL program, the CMD parameter cannot be used. Instead, the RQSDTA (Request Data) parameter performs the same function.

Lines 36.00−40.00: The 5 selection is handled by running the Start Query (STRQRY) command. This is the same command as was used in a previous chapter and lets the user create or run a previously created query for a unique request.

Lines 41.00−43.00: If the program reaches this point, a bad selection value exists. Indicator 71 is set on and the program branches back to display the menu. Indicator 71 will cause the appropriate error message.

Line 44.00: The ENDPGM statement is not run in the program, but is used to describe the end of source. The user ends the program by pressing F3 which causes the RETURN statement to be run (see lines 8.00−10.00).

Step 2. Entering and Creating the CL Menu Program (MLGMNUC)

You should now enter the CL coding as shown in Figure 12-4 on page 12-5 through PDM. Follow the same steps you used to create the MLGMSTP physical file in Chapter 5, "Creating Files" or copy the source from the QUSRTOOL library as described in Appendix B, "Overview of QUSRTOOL Library." The member source type should be CLP .

Once you have entered the DDS, create the program by typing a 14 (Compile) in the *Opt* column next to the program on the Work with Members Using PDM display.

Step 3. Running the CL Menu Program (MLGMNUC)

When you get a completion message that the MLGMNUC program has been compiled, you can run the program. Do this now by typing the command:

```
CALL MLGMNUC
```

When the Mailing List Menu is displayed, you can try each of the options. Press F3 to exit from each option.

Chapter 13. Creating an RPG Name Search

When the maintenance program is called, the user can access records if the account number is known. However, identifying the correct account number can be difficult if the account number is not known.

With a large number of accounts, a good deal of time can be spent looking through various lists to determine the account number. This chapter solves this problem by using a search technique. There are a variety of search techniques which could be used. The one shown in this chapter requires an extra field in the record and the field must be entered and maintained by the user. More sophisticated search techniques exist, but are not discussed in this manual.

Overview of Mailing List Name Search

When a record is entered into the file, the MLSRCH field contains the most likely entry that would be used to find the account. The name field should be entered as people would like to read their name. For example:

Name Field	Search Field
The Johnson Co	JOHNSON
Ms. Mary Jonathan	JONATHAN
The Jones Co.	JONES
Dr. Thomas R. Jasper	JASPER
Joseph Jones	JONES
Maria Jonesa	JONESA
Phillip Jones	JONES

If Joseph Jones has changed his address, his record needs to be changed. If he does not know his account number, the user would request the search function from the maintenance display. The user can enter any of the leading characters from the search field.

If a J is entered, then all of the names shown above would appear on the display. If JO is entered, only 6 of the 7 would appear. If JONES is entered, only 4 of the 7 would appear.

The user should try to enter as many characters as possible to minimize the number of records that will appear on the display.

The following shows the first display of the maintenance function with the entry requesting the name search on JONES.

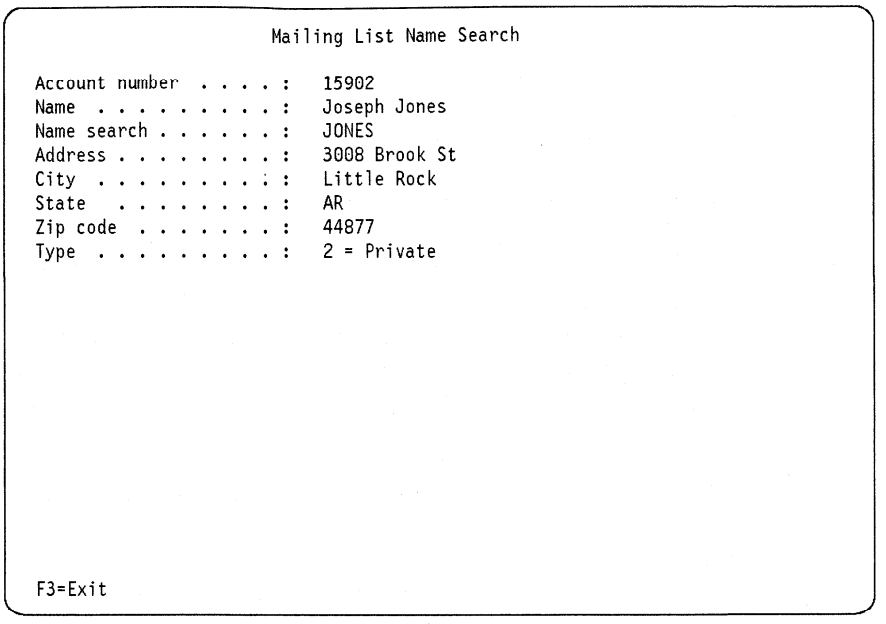
```
                    Maintain Mailing List Master

Action  . . . . . . . . . . . : 6          1=Display  2=Change
                                           3=Add      4=Delete
                                           5=Display GE value
                                           6=Name search

Account number  . . . . . . . :            Numeric 5.0

Search field  . . . . . . . . : JONES      Char

For Options 1-5, enter an Account number.
For Option 6, enter a Search field.

F3=Exit
```

The following display would appear if the user selected option 6 (Name search) and typed JONES for the *Search field* prompt.

```
                Mailing List Name Search              Search - JONES

Type options, press Enter.
   1=Dsp details   2=Return with Acct Nbr

Opt Search      Name          St  City          Address        Typ  Account
 1  JONES       Joseph Jones  AR  Little Rock   3008 Brook St  2    15902
    JONES       Philip Jones  CA  San Diego     365 Parkway    9    28903
    JONES       Samuel Jones  MN  Minneapolis   220 4th Ave N  1    10057
    JONESA      Maria Jonesa  PA  Philadelphia  559 9th Ave S  5    38724

F3=Exit
```

The user can now specify a record to work with.

If the user typed a 1 (Dsp details) for the first JONES record, the following display would appear.



The name search function is written so it can be called from multiple programs. Therefore, it allows two entries in the selection field. The first supports a simple display of the record in case the user wants to just display the information or needs to further identify the record.

The second function is the one that will normally be used in conjunction with the maintenance program. When the user enters a 2 next to the record desired, the name search program returns to the maintenance program which displays the record selected in display mode. The user can then use a function key to access the same record in change or delete mode.

When the list of records is presented in the name search function, there may be more records than will fit on one display. Each displays worth of records is called a page. If this is the case, a plus sign (+) will appear after the last record and the key to page down will be made active. If the user presses the key to page down, the next page will be shown. If the second page is displayed, the user can then use the key to page up to return to the first page. The keys will be active in either direction as long as there are records to meet the criteria.

In this solution, only a single key field makes up the search field. In a larger database, you might have to add state or zip code or some other criteria to assist the user in determining the right record.

When the user enters a 2 to select a record, the user does not have to know or enter the account number. Accessing the correct record is very critical to any good database solution and entering account numbers can be a very error-prone solution.

Subfile Support

To provide the display of records, the program will use the system supplied subfile support. This simplifies programming and provides a good solution for users.

The concept of a subfile should be viewed as a set of records that can be seen one page at a time by the user. The program will select the records it wants and will write them to the subfile.

It is very typical to have many records to fill a subfile. For example, if the user requests a search on JO, there may be many pages worth of records that will satisfy the request. The best performance solution for the program is to fill the first page and display it. If the user presses the key to page down, the program regains control and fills the next page. This is a good performing solution because there are many instances where the user will have seen enough after seeing the first page. The user may realize that JO is not a meaningful search criterion or he may actually find the desired record on the first page. Minimizing the number of database read operations is a key ingredient in achieving good performance.

The system subfile support can be used to handle the key to page up. If the second through Nth pages of the subfile are displayed, the user can press the key to page up and the system will cause the previous page to be displayed. The system will also handle the key to page down as long as there is another page. For example, if the key to page down has been pressed twice and the subfile contains 3 pages, the system will allow the rolldown function to occur and then the rollup function without ever returning to the program. As long as the system has records to display, it will not return to the program. This system subfile support makes it easy to produce this type of application which is very typical of interactive work.

The system performs work for each user interaction even though the program is not returned to for some of the functions. It costs system overhead to roll up or down. The response time is normally good, but designing for performance is always important.

Keyboards do not have roll up or roll down engraved keys. The functions are normally requested by holding the Shift key and pressing the *cursor up* or *cursor down* keys. Some keyboards have *Page Up* and *Page Down* engraved keys.

When either type of request is made, the system receives an indication that is treated identically. Many system displays are coded to allow rollup or rolldown. You can specify DDS keywords to allow a return to your program when the keys are pressed.

The important thing to remember is that the term *page up* is the equivalent of *rolldown*. The following describes this:

Table 13-1. DDS Keywords and Related Function Keys

	User Presses:	
DDS Keywords	**Keyboard Shift Key and Cursor Key To:**	**Engraved Keys**
ROLLDOWN (or PAGEUP)	Roll down	Page Up
ROLLUP (or PAGEDOWN)	Roll up	Page Down

Some users prefer to follow the Systems Application Architecture (SAA) guidelines where F7 is equal to *Page down* and F8 is equal to *Page up*. DDS supports this type of approach also. See the ALTPAGUP and ALTPAGDWN keywords in the *DDS Reference*.

The subfile shown in this application is mostly a display function. Only a single input field exists to allow options. The data on the display is not changed. There are other subfile approaches where the user could update some or all of the data shown. Subfiles offer a significant programmer productivity gain. If you are going to be coding interactive applications, you will want to become familiar with the function.

A DDS subfile requires some special keywords that all begin with SFL. Two formats are required:

- The subfile format is identified by the subfile (SFL) keyword and describes the fields in the subfile. When you write to the subfile record format, changes occur within main storage work areas. No change occurs to the display.
- The subfile control format which is identified by the subfile control (SFLCTL) keyword. This format controls subfile operations (such as when to display the subfile) and may also contain fields. In many displays (including this one) the column headings are placed in the subfile control format. A single display file can have up to 12 subfiles.

Structure of Name Search Program

The structure of the name search program consists of the following:

- A display file, MLGNAMD, that contains the DDS describing the format of the display.
- A logical file, MLGNAML, that provides a new access path on the name search field.
- An RPG program, MLGNAMR, that performs the search.

Figure 13-1 shows an overview of the name search program.

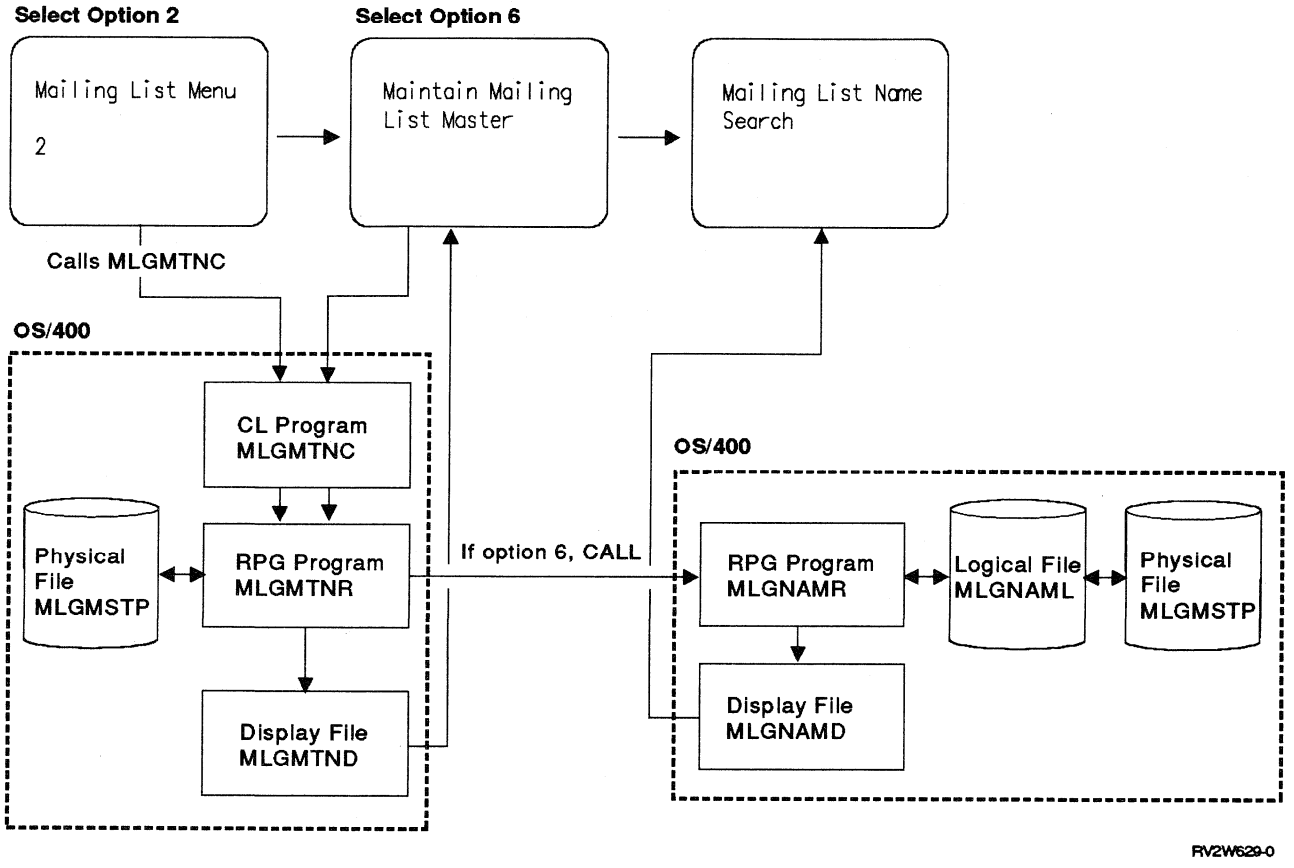


Figure 13-1. Overview of Mailing List Name Search Program MLGNAMR

Overview of Name Search Logical File

To provide a fast name search, a logical file with an access path must exist on the name search field. This access path will control the sequence in which the records are displayed. Therefore, instead of using just a single key on the search field, multiple keys will be specified in the order of:

Search field
State
City
Name

Description of DDS for Name Search Logical File (MLGNAML)

Figure 13-2 shows the DDS for the MLGNAML logical file. A description of each line follows the figure.

```
          ...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7
0001.00       A* MLGNAML - Mailing list name search logical file
0002.00       A          R MLGMSTR                   PFILE(MLGMSTP)
0003.00       A          K MLSRCH
0004.00       A          K MLSTAT
0005.00       A          K MLCITY
0006.00       A          K MLNAME
```

Figure 13-2. DDS Specifications for MLGNAML

Line 1.00: A comment which describes the file.

Line 2.00: The R in position 17 describes the record line. The record name used (MLGMSTR) is the same as the record format of the physical file. The PFILE keyword identifies the physical file to be used.

Lines 3.00−6.00: The K in position 17 identifies the fields as key fields. The keys are named in the sequence desired with the high order first.

Step 1. Entering the DDS and Creating the Name Search Logical File (MLGNAML)

You should now enter the DDS as shown in Figure 13-2 through PDM. Follow the same steps you used to create the MLGMSTP physical file in Chapter 5, "Creating Files" or copy the source from the QUSRTOOL library as described in Appendix B, "Overview of QUSRTOOL Library." The member source type should be LF.

Once you have entered the DDS, create the file by typing a 14 (Compile) in the *Opt* column next to the file on the Work with Members Using PDM display.

You need to receive the completion message that the file was successfully created before you can create the next RPG program.

Description of DDS for Name Search Display File (MLGNAMD)

The MLGNAMD contains the display formats used to produce the Mailing List Name Search display. This display file could also have been created using SDA. See Chapter 14, "Creating a Display Using SDA." Figure 13-3 shows the DDS for MLGNAMD. A description of each line follows the figure.

```
        ...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7
0001.00     A* MLGNAMD  - Mailing list name search - used by MLGNAMR
0002.00     A                                      PRINT
0003.00     A                                      REF(MLGREFP)
0004.00     A          R BOTTOM                    TEXT('Text and messages +
0005.00     A                                      at bottom of display')
0006.00     A N67                              21  2'No records exist to +
0007.00     A                                      be displayed'
0008.00     A                                      DSPATR(HI)
0009.00     A                                  24  2'F3=Exit'
0010.00     A          R SFLRCD                    TEXT('Subfile record')
0011.00     A                                      SFL
0012.00     A            SELECT         1   B  7  3VALUES('1' '2' ' ')
0013.00     A            MLSRCH    R            7  6
0014.00     A            ABBNAM         13      7 18
0015.00     A            MLSTAT    R            7 33
0016.00     A            ABBCTY         13      7 37
0017.00     A            ABBADD         13      7 52
0018.00     A            MLTYPE    R            7 67
0019.00     A            MLACCT    R            7 72
0020.00     A            MLNAME    R        H
0021.00     A            MLCITY    R        H
0022.00     A            MLADDR    R        H
0023.00     A            MLZIP     R        H
0024.00     A          R SFLCTL                      TEXT('Sub file control +
0025.00     A                                        record')
0026.00     A                                        SFLCTL(SFLRCD)
0027.00     A                                        SFLSIZ(100)
0028.00     A                                        SFLPAG(13)
0029.00     A* 67 is on if there is at least one record to display
0030.00     A* 68 is used to clear the subfile
0031.00     A N68 67                                 SFLDSP
0032.00     A N68                                    SFLDSPCTL
0033.00     A  68                                    SFLCLR
0034.00     A                                        OVERLAY
0035.00     A                                        CA03(93 'Exit')
0036.00     A  71                                    ROLLUP(98 'Rollup')
0037.00     A  83                                    SFLEND
0038.00     A            SFLNBR         4  0H       SFLRCDNBR(CURSOR)
0039.00     A                                    1 20'Mailing List Name Search'
0040.00     A                                       DSPATR(HI)
0041.00     A                                    1 60'Search -'
0042.00     A            SEARCH    R              +1REFFLD(MLSRCH)
0043.00     A                                    3  2'Type options, press +
```

Figure 13-3 (Part 1 of 2). DDS for Display File MLGNAMD

```
0044.00      A                                      Enter.'
0045.00      A                                  4  5'1=Display details'
0046.00      A                                    +3'2=Return with Acct Number'
0047.00      A                                  6  2'Opt'
0048.00      A                                     DSPATR(HI)
0049.00      A                                  6  6'Search'
0050.00      A                                     DSPATR(HI)
0051.00      A                                  6 18'Name'
0052.00      A                                     DSPATR(HI)
0053.00      A                                  6 33'St'
0054.00      A                                     DSPATR(HI)
0055.00      A                                  6 37'City'
0056.00      A                                     DSPATR(HI)
0057.00      A                                  6 52'Address'
0058.00      A                                     DSPATR(HI)
0059.00      A                                  6 66'Typ'
0060.00      A                                     DSPATR(HI)
0061.00      A                                  6 71'Account'
0062.00      A                                     DSPATR(HI)
0063.00      A          R DETAIL                   TEXT('MLGMSTP record +
0064.00      A                                     display')
0065.00      A                                     CA03(93 'Exit')
0066.00      A                                  1 25'Mailing List Name Search'
0067.00      A                                     DSPATR(HI)
0068.00      A                                  3  2'Account number  . . . . :'
0069.00      A            MLACCT    R             +3
0070.00      A                                  4  2'Name  . . . . . . . . . :'
0071.00      A            MLNAME    R             +3
0072.00      A                                  5  2'Name search . . . . . . :'
0073.00      A            MLSRCH    R             +3
0074.00      A                                  6  2'Address . . . . . . . . :'
0075.00      A            MLADDR    R             +3
0076.00      A                                  7  2'City  . . . . . . . . . :'
0077.00      A            MLCITY    R             +3
0078.00      A                                  8  2'State   . . . . . . . . :'
0079.00      A            MLSTAT    R             +3
0080.00      A                                  9  2'Zip code  . . . . . . . :'
0081.00      A            MLZIP     R             +3
0082.00      A                                 10  2'Type  . . . . . . . . . :'
0083.00      A            MLTYPE    R             +3
0084.00      A                                    +1'='
0085.00      A            TYPTXT         15        +1
0086.00      A                                 24  2'F3=Exit'
```

Figure 13-3 (Part 2 of 2). DDS for Display File MLGNAMD

Lines 1.00—3.00: The statements are similar to those described in display files in previous chapters.

Lines 4.00 and 5.00: The BOTTOM record format describes the format which will appear on the bottom of the display. It will appear on lines 21 through 24 of the display.

Lines 6.00—8.00: A constant is conditioned by indicator 67 not being on. This is the error condition when no records exist to be displayed. The program will set on indicator 67 as soon as one record exists.

Line 9.00: Describes a valid function key. Note that this is the constant that the user sees. The valid function key from a DDS viewpoint is defined in the subfile control format.

Lines 10.00 and 11.00: The subfile record is defined. The keyword SFL identifies this. The subfile record will contain the fields that will exist in the subfile.

Line 12.00: The SELECT field is defined as a 1-character *both* field and accepts the values 1, 2, or a blank. A blank is allowed in case the user enters a value and then changes his mind before pressing Enter. The program must be sensitive to a blank being returned.

Lines 13.00–23.00: The fields for the subfile are defined. The line number (7) is the line of the first subfile record. Most of the fields are defined using the reference capability (R in position 29). Because the screen is only 80 characters wide, displaying the full length of the fields would not allow as much data to be seen. In general, for this type of display, you will want to present as much information as possible for the user to select from. Therefore, the DDS statements define some abbreviated fields which will contain the first 13 bytes of larger fields.

Hidden Fields: Some of the fields are hidden (H in position 38). This means they are known to the subfile, but not to the user. If the user selects one of the records for a detail display, the program retrieves all of the fields specified in the subfile record and can display the entire record without re-retrieving from the database. This would provide a performance gain over re-accessing the record from the database, but is only practical when the record contains a small number of fields.

Lines 24.00–28.00: The subfile control record is defined. The SFLCTL keyword identifies this record as the subfile control record and it identifies the record format (SFLRCD) which is the subfile record.

Subfile Size: The SFLSIZ keyword identifies the size (100) of the subfile. However, the exact entry is not important. What is important is that it is greater than the SFLPAG entry (13) which is the number of subfile records which will appear on a single display (the number on a page). As long as the SFLSIZ value is greater than SFLPAG, the system will allow the entry of many records (more than 100) and will handle much of the roll requests. The subfile record was defined to start on line 7. Since SFLPAG is specified as 13, lines 7 through 19 of the display will contain the subfile records. You should avoid entering a very large value for SFLSIZE (such as 9999) because of internal performance considerations.

Lines 31.00–33.00: The SFLDSP keyword describes when the subfile should be displayed. It should occur when indicator 68 is off and indicator 67 is on. The program will set indicator 67 on if it writes at least one record to a page of the subfile. It will set indicator 68 on when the subfile should be cleared.

The subfile control record is displayed when indicator 68 is not on (SFLDSPCTL keyword). The subfile is cleared when indicator 68 is on (SFLCLR keyword).

Writing to the Subfile Control Record: The program must control what the subfile should do when it writes to the subfile control record. There are three conditions to be considered:

- The normal condition where the subfile control record and the subfile are displayed.
- The subfile should be cleared. This means that a new request exists and all records in the subfile should be deleted before starting the new request. When this occurs, there should be no output to the display and the SFLCLR option keyword should be conditioned on.
- If no records exist, the system will consider it an error if you attempt to display the subfile. To protect against this, indicator 68 will be set when at least one record is written. If the not 68 condition occurs, neither the subfile nor the subfile control record will be displayed.

Lines 34.00 and 35.00: The OVERLAY keyword is used because the bottom record format is written first onto the display and it will be overlaid by the other formats. If OVERLAY is not specified, the entire display would be deleted and then the new format written.

The CA03 keyword identifies the standard exit function.

Lines 36.00−37.00: The ROLLUP keyword is conditioned by indicator 71. Only in the case where the program senses that it has more than one page worth of records to be displayed will it allow the key to page down to be used. Indicator 98 is returned to the program if ROLLUP is valid and the key to page down is pressed.

As described earlier, the program will be written to fill one page at a time. If the user presses the key to page down, the program will be given control and will fill an additional page.

The SFLEND function provides the plus sign (+) at the bottom of the subfile if there are more records to be displayed. Indicator 83 controls this. The plus symbol will be displayed unless indicator 83 is set on.

A SETOFF keyword is used.

Line 38.00: The SFLRCDNBR keyword provides a variety of functions. It must be defined as 4 digits with 0 decimals and is normally a hidden field (H in position 38). The field name SFLNBR is used to communicate to the work station support. It will describe which page of the subfile should be displayed by containing the record number of a record on a specific page. Normally you want to display the page that contains the record number of the last record written.

The CURSOR specification describes that the cursor should be placed on the first input-capable field of the record described by SFLNBR.

If SFLNBR contains a 7, the first page of the subfile would be displayed and the cursor would be positioned to the first input field on the seventh record. If SFLNBR contains 15, the second page of the subfile would be displayed (13 per page according to the SFLPAG specification) and the cursor would be positioned to the 15th record which would be the second record on the display.

In this application, you would normally want the cursor positioned to the first record on the display. This would not require the CURSOR entry, but specifying CURSOR is useful for other functions where you want to control the exact location.

Lines 39.00−42.00: This describes the heading of the display. Note that the user specified search value from the MLGMTNR program is placed on the first line to inform the user of what he requested.

Lines 43.00−46.00: The constants describe the user actions allowed.

Lines 47.00−62.00: The column headings for the subfile are defined.

Lines 63.00−67.00: The DETAIL format is defined with its heading and valid function key.

Lines 68.00−85.00: The fields for the detail display and the constants are described. Note on line 86.00, the field TYPTXT is described as 15 bytes. This is the text description of the account type codes. The program will translate the one character code into a meaningful text description.

Line 86.00: The valid function key is described.

Step 2. Entering the DDS and Creating the Name Search Display File (MLGNAMD)

You should now enter the DDS as shown in Figure 13-3 on page 13-8 through PDM. Follow the same steps you used to create the MLGMSTP physical file in Chapter 5, "Creating Files" or copy the source from the QUSRTOOL library as described in Appendix B, "Overview of QUSRTOOL Library." The member source type should be DSPF .

Once you have entered the DDS, create the file by typing a 14 (Compile) in the *Opt* column next to the file on the Work with Members Using PDM display.

You need to receive the completion message that the file was successfully created before you can create the next RPG program.

Overview of RPG Name Search Program

The MLGNAMR RPG program performs the search.

Description of RPG Specifications for Name Search Program (MLGNAMR)

Now that you have created the MLGNAMD display file and the MLGNAML logical file, you can create the MLGNAMR RPG program. Figure 13-4 shows the RPG specifications for the program. A description of each line follows the figure.

```
        ...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7
0001.00     F* MLGNAMR - Mailing list name search
0002.00     FMLGNAML IF  E           K        DISK
0003.00     FMLGNAMD CF  E                    WORKSTN
0004.00     F                                        SFLNBRKSFILE SFLRCD
0005.00     E                    TYP     1  10  1   TXT    15   Type of acct
0006.00     E                    NAM        11  1                Name search
0007.00     C           *ENTRY   PLIST                             Parm list
0008.00     C                    PARM           SEARCH            Search field
0009.00     C                    PARM           MLACCT            Account nbr
0010.00     C                    PARM           PGMEND  1         Pgm end sw
0011.00     C* End program if Pgm end switch is set
0012.00     C           PGMEND   CABNE*BLANK    ENDPGM     LRLR   Not blank
0013.00     C* Determine len of the search field (end of non-blank or 0 len)
0014.00     C                    MOVEASEARCH    NAM,1             Move to arr
0015.00     C                    Z-ADD11        NX      30        Inlz to 11
0016.00     C           LOOP     TAG                              Begin loop
0017.00     C                    SUB  1         NX         20     20 if plus
0018.00     C   20      NAM,NX   CABEQ*BLANK    LOOP               If blank
0019.00     C                    ADD  1         NX                 Add 1
0020.00     C* Clear out the subfile
0021.00     C                    Z-ADD0         BOTNBR             Inlz lst nbr
0022.00     C                    SETON                     68       SFLCLR indic
0023.00     C                    WRITESFLCTL                        Clear SFL
0024.00     C                    SETOF                     68       SFLCLR indic
0025.00     C                    SETOF                     71       Prevnt rolup
0026.00     C* Use SEARCH field to set lower limit
0027.00     C           SEARCH   SETLLMLGMSTR                       Set low lmt
0028.00     C* Start rollup or first display
0029.00     C           ROLLUP   TAG                                ROLLUP tag
0030.00     C                    Z-ADDBOTNBR    SFLNBR             Set rcd nbr
0031.00     C                    Z-ADD0         LINCNT  30         Inlz line ct
0032.00     C                    SETOF                     67       Record wrtn
0033.00     C* 71 is rollup. If rollup was allowed, a DB rcd was already read
0034.00     C   71               GOTO WRTSFL                        Branch ahead
0035.00     C* Read a record
0036.00     C           RDRCD    TAG                                RDRCD tag
0037.00     C                    READ MLGMSTR                    83 Set low lmt
0038.00     C   83               GOTO ENDDB                         If EOF
0039.00     C* Determine if record matches the search field
0040.00     C                    MOVEAMLSRCH    NAM,1              Move to arr
0041.00     C                    MOVEA*BLANKS   NAM,NX             Blank excess
0042.00     C           *LIKE    DEFN MLSRCH    SAVNAM
0043.00     C                    MOVEANAM,1     SAVNAM             Move to fld
0044.00     C           SEARCH   CABNESAVNAM    ENDDB      8383    If not equal
0045.00     C           13       CABEQLINCNT    ENDDB          71  If full pag
```

Figure 13-4 (Part 1 of 3). RPG Specifications for MLGNAMR

```
0046.00     C* Write to the subfile record
0047.00     C           WRTSFL    TAG                              Write to SFL
0048.00     C                     SETOF                      71    Prevnt rolup
0049.00     C                     ADD  1         SFLNBR            Add to nbr
0050.00     C                     MOVE *BLANK    SELECT            Blank select
0051.00     C* Move left to shorten field lengths for display
0052.00     C                     MOVELMLNAME    ABBNAM            Abbrev name
0053.00     C                     MOVELMLCITY    ABBCTY            Abbrev city
0054.00     C                     MOVELMLADDR    ABBADD            Abbrev addr
0055.00     C* Blank the text for detail display and lookup text descrp
0056.00     C                     MOVE *BLANKS   TYPTXT            Type text
0057.00     C                     Z-ADD1         TX      30        Inlz
0058.00     C           MLTYPE    LOKUPTYP,TX                   20 Lookup eq
0059.00     C   20                MOVE TXT,TX    TYPTXT            If equal
0060.00     C                     WRITESFLRCD                      Write SFL
0061.00     C  N67                Z-ADDSFLNBR    TOPNBR  40        Top of pg #
0062.00     C                     SETON                      67    Rcd wrtn sws
0063.00     C                     ADD  1         LINCNT            Add line cnt
0064.00     C                     GOTO RDRCD                       Loop back
0065.00     C* End of data base processing
0066.00     C           ENDDB     TAG                              End of file
0067.00     C* N67 indicates that no records were written to the sub file
0068.00     C  N67                GOTO DISPLY                      Branch ahead
0069.00     C                     Z-ADDSFLNBR    BOTNBR  40        Save bottom
0070.00     C                     Z-ADDTOPNBR    SFLNBR            Set to top
0071.00     C* Display the subfile
0072.00     C           DISPLY    TAG                              Display SFL
0073.00     C* Reset BOTTOM indicators and write SFLCTL
0074.00     C                     MOVE *BLANK    SLTRCD  1         Selected rcd
0075.00     C                     WRITEBOTTOM                      Write BOTTOM
0076.00     C                     SETOF                    505152Error swtch
0077.00     C                     EXFMTSFLCTL                      Dsp subfile
0078.00     C  N67                                                 No rcds wrtn
0079.00     COR 93                DO                               CF 3
0080.00     C                     Z-ADD*ZEROS    MLACCT            Set return
0081.00     C                     GOTO ENDPGM                      Goto end
0082.00     C                     END                              No rcds etc
0083.00     C* If rollup key was pressed, loop back
0084.00     C   98                GOTO ROLLUP                      Loop back
0085.00     C* Do READC for detail display
0086.00     C           READC     TAG                              READC tag
0087.00     C                     READCSFLRCD                   21 Read chgd
0088.00     C* A record was selected in the subfile, check and display
0089.00     C  N21                DO                               SFL select
0090.00     C           SELECT    CABEQ' '       READC             Not selected
0091.00     C                     MOVE 'X'       SLTRCD            Selected rcd
0092.00     C* Select=1 Detail display of the record
0093.00     C           SELECT    IFEQ '1'                         Select 1
0094.00     C                     EXFMTDETAIL                      Detail dsp
0095.00     C                     END                              Select 1
```

Figure 13-4 (Part 2 of 3). RPG Specifications for MLGNAMR

```
0096.00      C* Select=2 Return with selected account number
0097.00      C           SELECT    IFEQ '2'                        Select 2
0098.00      C                     GOTO ENDPGM                     Branch ahead
0099.00      C                     END                             Select 2
0100.00      C* End of SFL rcd which has an action, blank selection entry
0101.00      C                     MOVE *BLANK    SELECT           Blank select
0102.00      C                     UPDATSFLRCD                     Update SFL
0103.00      C                     GOTO READC                      Loop back
0104.00      C                     END                             SFL select
0105.00      C* End of SFL reached (READC has found no more)
0106.00      C*    If one or more detail records displayed, redisplay subfile
0107.00      C           SLTRCD    CABEQ'X'       DISPLY           If selected
0108.00      C* No SFL records displayed, no F3 or rollup, return as good end
0109.00      C                     Z-ADD*ZEROS    MLACCT           Zero acct
0110.00      C           ENDPGM    TAG                             ENDPGM tag
0111.00      C                     RETRN                           Return
0112.00 ** TYP and TXT
0113.00 1Business
0114.00 2Government
0115.00 3Organization
0116.00 4School
0117.00 5Private
0118.00 9Other
```

Figure 13-4 (Part 3 of 3). RPG Specifications for MLGNAMR

Lines 1.00 and 2.00: A comment is used and the MLGNAML logical file is defined. The logical file is keyed by the MLSRCH field.

Lines 3.00 and 4.00: The work station file is defined. The additional K specification (K in position 53) allows for additional options to be stated about a file. When a subfile is to be used, the SFILE keyword must be defined. The field name SFLNBR defined in positions 47 through 52 specifies the field that will be passed back to RPG from the work station support to describe the record number read by a READC operation.

The SFLRCD record name in positions 60 through 65 identifies the record format in the subfile that will be used for the subfile record.

Lines 5.00 and 6.00: Two arrays are used on line 5.00. This is called an alternating array. The first is used to translate the type of account (a code) into a text description.

The second E specification identifies the NAM array which will be used as a work area to assist in determining if the search field entered matches the database record.

Lines 7.00–10.00: An entry parameter list is defined. Three parameters are identified that will be passed into the program. The program will use two of the values and return one.

The SEARCH field will be passed in. The MLACCT field will be passed back. The PGMEND field is passed in and is used as a switch to determine when to end the name search program.

Line 12.00: If the PGMEND parameter does not contain a *BLANK, the program branches to the ENDPGM label with the LR indicator on. The maintenance program will set this parameter to X when it is time to end the name search program.

Note: The LT/GT indicator locations are specified with LR. The CABxx operation allows any of the 3 indicators to be set. In this case, LR should be on if PGMEND is not blank.

Lines 14.00–19.00: The search field is examined to determine the number of characters keyed by the user. The field is up to 10 bytes long, but the program needs to know how many characters were actually entered.

The program uses MOVEA to move the SEARCH field beginning in position 1 of the NAM array. The NAM array is 11 positions long with the last position always blank. The index NX is then set to 11 and a loop begins. The first time through, NX is decremented by 1 and indicator 20 is set on if NX is still plus. If NX is plus, a comparison is made against the NXth character in the NAM array and if blank, the loop is repeated. The loop continues until a non-blank is sensed. The NX index is then bumped by one.

If the user entered JONES, the NX value would be 6 at the completion of the loop.

Note: The calling program must prevent an all blank search field from being entered.

Lines 21.00–25.00: Since the program will remain open, the work areas of the program and the subfile will contain the values from the last request. Because of this, the subfile must be initialized on each call to the program.

The BOTNBR field is zeroed out. It will contain the subfile record number of the last subfile record written for a page.

The BOTNBR field is used to help determine which page of the subfile should be displayed and where the rollup operation should start for a new page. In this particular program, the function is not needed because the database record will not be accessed again to perform a selection function on a subfile record. However, the code is left in so the same approach can be used regardless of how the subfile records are processed.

Indicator 68 controls the SFLCLR keyword. It is set on, the WRITE to the subfile control record causes the clear function to occur and the indicator is set off.

Indicator 71 is set off to prevent the use of the key to page down.

Line 27.00: The SETLL command uses the SEARCH field to set the position in the database. If JO has been specified as the search value, the database cursor is now set to the first record equal to JO or greater than JO. Note that SETLL does not read a data record. It just sets the cursor into the index to allow the READ statement to read the next database record.

Lines 29.00–32.00: This is the label to begin the rollup for a new page. If a page has been displayed and the user presses the key to page down, this is the beginning of the code to fill another page.

The program initializes the SFLNBR field with the BOTNBR field. This is the record number of the last record written to the subfile. The line count (how many lines have been written to the page) is set to zero. Indicator 67 is set off to describe that no records have been written to the subfile. It is important to know whether any records have been written because the system considers it an error if you attempt to display an empty subfile.

Line 34.00: In order to properly set the plus sign at the bottom of the subfile display to indicate that rollup is allowed, the program will read a record from the database before testing to see if the page is full and should be displayed. For example, if the last record written to the subfile is really the last record that will satisfy the search criteria, the key to page down should not be valid.

Allowing Rollup: The program will only set on indicator 71 to allow rollup if it determines that it has at least one more record after the page is full. If the allow rollup indicator (71) is on at this point, the program should bypass the first read for a new page because the record has already been read and can now be written to the subfile. Consequently, the branch to WRTSFL occurs if 71 is on.

Lines 36.00−38.00: The RDRCD label establishes the label which begins the processing to read a record from the database. The READ is run and indicator 83 is set on if end of file is found. The program branches to the label ENDDB on end of file.

Lines 40.00−44.00: The program now determines if the record read matches the search criteria. The MLSRCH field from the database is moved to the NAM array beginning in position 1. The excess positions of the search field are blanked out by the next instruction. If the database record read is JONESA and NX is 6, the value in the array is now equal to JONES.

The array is then moved to a field beginning with the first position of the array. Note that the SAVNAM field is not defined in the program. The DEFN operation defines the SAVNAM field to be the same length as the MLSRCH field.

A comparison is then made against the search field and the SAVNAM field and a branch occurs on unequal. An unequal comparison means either that 1) No records satisfy the search criteria or 2) no more records satisfy the search criteria.

Line 45.00: If the comparison is equal, the program now tests to see if the page is full. There can be 13 records on a page according to the SFLPAG keyword in DDS. If the limit has been reached, it is time to display the subfile. Note that the CABEQ operation sets on indicator 71 which is the allow rollup indicator. The program has read one more record than will fit into the subfile so it can allow for rollup to occur (there will be at least one record on the next page if the user requests rollup).

Lines 47.00−54.00: The WRTSFL label begins the write of the database record into the subfile. The SFLNBR which determines the record number to write to is incremented. The SELECT field is ensured to be blank and the abbreviated fields described in DDS are filled by MOVEL statements. This moves from the left to fill the abbreviated fields.

Lines 56.00−59.00: The type of account needs to be translated into a text description. The program initializes the description to blank and then uses an array lookup to find the text description. The type code is used for the lookup

and if a match is found, the corresponding description is moved from the TXT array.

Lines 60.00–64.00: The WRITE causes the subfile record to be written. 67 is off for the first record of each page. The N67 condition, therefore, writes the current subfile record number into the TOPNBR field. This is the record number of the first entry on a page. Indicator 67 is then set on. The line count (number per page) is then incremented and the program loops back to read another record.

Lines 66.00–70.00: The ENDDB describes the label to be branched to if either end of file or end of page is found. If 67 is not on, no records were found to satisfy the search criteria and the program branches to the DSPLY tag.

If 67 is on, at least one record was written to the page and the program sets the current subfile record number into the BOTNBR field (the bottom number). It then places the TOPNBR into the SLFNBR field. These fields will be used to help describe which page should be displayed within the subfile.

Line 72.00: The DISPLY tag describes the label for the display to the work station. The SLTRCD field is blanked and the BOTTOM record is written. The BOTTOM record describes the valid function keys. Error indicators are set off and the EXFMT causes a write/read to the display file. The program now waits for the user to respond.

The display file is written with the subfile so the user can now see one page full of records. The user may now make several decisions:

- To select one of the records. The choice is to display the details of the record or return to the maintenance program with a specific account number.
- Use F3 to tell the program to return to the maintenance program. The user can also press the Enter key to achieve this result. This will normally be used if the user entered a search argument that did not find any records or the user did not want to select any of the records that were displayed.
- If there is a plus sign under the subfile, the user can press the key to page down. If the user is looking at page 2 or greater, the key to page up may be pressed. The program is only informed of these decisions if the user is trying to rollup and the last page is being displayed. All of the other page down and page up keys are handled by the system and it will display the appropriate subfile page. When rolling down and there are no more records to roll to, the system will provide an error message.

Lines 78.00–82.00: The program uses the N67 condition to mean that no records were written to the subfile. The user has seen a message describing that no records satisfy the search. If this is the case or F3 is pressed, the name search program should end and an all zero account number is passed back to the maintenance program. This is the signal to the maintenance program, that the user does not want any further action.

Line 84.00: If the key to page down was pressed (indicator 98), the program branches back to the rollup tag.

Lines 86.00 and 87.00: This begins the READC (Read Next Changed) processing. The subfile has been displayed to the user. The user could have entered several selections on one or more subfile records. The program must now determine which subfile records were selected.

Read Next Changed: Rather than read every subfile record, RPG supports the READC operation which does a read from the beginning of the subfile looking for the next changed record. When the READC operation is run, the subfile record (if any are changed) is returned to the program and the SFLNBR field (as specified on line 4.00) is set to the value of the record number returned. This makes it easy to allow update of the same record.

Indicator 21 is set if the READC operation finds no changed records or no more changed records. You should consider 21 as being end of file was found on the subfile.

Lines 89.00−91.00: If N21 (not subfile end of file), the user entered a value and the program must now consider the selection entry.

If the user entered a value and then changed his mind and overkeyed a blank, the subfile record would be considered changed even though the user does not want any action performed. The CABEQ operation branches back to READC for this condition.

If the entry is not a blank, the DDS VALUES keyword has prevented any other value from being entered except a 1 or 2. The program knows that it will process one of the entries and, therefore, sets a switch SLTRCD so it knows that at least one record was processed.

Lines 93.00−95.00: If the selection is a 1, the DETAIL display is written and the program waits for a response. The user has requested to look at the details of a record.

Multiple records may be specified with 1s. Each entry requested would be displayed and then the user would be returned to the subfile display.

Lines 97.00−99.00: If the user entered a 2, he is requesting to return to the maintenance program with the account number of the record he selected.

Note: If the user specified multiple 2s or a combination of 1s and 2s, the program will end when the first 2 is encountered.

Lines 101.00−104.00: This set of code resets the select field to blank the user entry. If the subfile is redisplayed, the SELECT field needs to be blank. The UPDAT operation changes the record in the subfile. The program branches back to perform another READC.

Only a selection of 1 would run through this code. However, the approach allows a standard method of handling subfile processing so it is used here in case other options existed in the select field which do not cause the program to end.

Lines 105.00−111.00: At this point, there are no more changed records in the subfile.

The program checks the SLTRCD switch. If it is set to X, the user has displayed at least one detail entry. The subfile should be re-displayed to see what the user wants to do now. The program branches to DSPLY.

If the user pressed the Enter key without making any selections, the program assumes the user wants to end the name search program. Therefore, the program sets the account number field to zeros and returns. Note that LR is off,

so the program will still have all of its work areas filled and files opened if it is called again.

Step 3. Entering the RPG Specifications and Creating the RPG Name Search Program (MLGNAMR)

You should now enter the RPG specifications as shown in Figure 13-4 on page 13-13 through PDM. Follow the same steps you used to create the MLGMSTP physical file in Chapter 5, "Creating Files" or copy the source from the QUSRTOOL library as described in Appendix B, "Overview of QUSRTOOL Library." The member source type should be RPG .

Once you have entered the RPG specifications, create the program by typing a 14 (Compile) in the *Opt* column next to the program on the Work with Members Using PDM display.

Step 4. Running the RPG Name Search Program (MLGNAMR)

When you get a completion message that the MLGNAMR program has been compiled, you can run the program. The MLGNAMR program is implicitly called from the maintenance program when the name search function is used. Do this now by typing the following command:

```
CALL MLGMNUC
```

On the Maintain Mailing List Master display you can select option 6 and type JONES for the search field to test the function.

Chapter 14. Creating a Display Using SDA

The screen design aid (SDA) supports the following functions:

- Designing screens
- Designing menus
- Testing display files

SDA could have been used for any of the display files used in this manual. To best understand SDA, you should have a good understanding of the type of DDS used to create a display file. Without this knowledge, some of the prompts used in SDA would be difficult to follow.

As a programmer, you have several choices regarding SDA:

- Use SDA exclusively. You would never directly enter or change the DDS statements using SEU.
- Use SDA partially. For example, you may want to create much of your display work with SDA and then tailor the created DDS using SEU.
- Use SDA for some display files and SEU for some other files.
- Use only SEU.

The major advantage of SDA is that you can see the display as it is being developed. You can move fields and constants around, actually try different display attributes (such as highlighting or blinking), and test out the functions that are controlled by option indicators (such as error messages).

This chapter describes an alternative solution for the Inquiry display described in Chapter 10, "RPG Solution for Inquiry Program" as shown below (a different display file will be created):

```
                    Mailing List Inquiry

Account number  . . . . . :
```

The user types in an account number, for example 10057, presses the Enter key and is presented with the following display:

```
                    Mailing List Inquiry

Account number  . . . . . :   10057
Account type  . . . . . :   1
Name  . . . . . :   Samuel Jones
Search name . . . . . :   JONES
Address . . . . . :   220 4th Ave NW
City  . . . . . :   Minneapolis
State . . . . . :   MN
Zip code  . . . . . :   55454
```

Creating the Display

Follow these steps to create the display using SDA:

1. Start SDA by typing the following command on the command entry line:

   STRSDA

   The AS/400 Screen Design Aid (SDA) display is shown.

```
                        AS/400 Screen Design Aid (SDA)

Select one of the following:

     1. Design screens
     2. Design menus
     3. Test display files









Selection or command
===> 1

F3=Exit    F4=Prompt    F9=Retrieve    F12=Cancel
                                    (C) COPYRIGHT IBM CORP. 1981, 1989.
```

2. Select option 1 (Design screens) and press the Enter key.

   The Design Screens display is shown.

```
                              Design Screens

Type choices, press Enter.

  Design option . . . . . .  1           1=Select file keywords
                                         2=Select record keywords
                                         3=Select subfile keywords
                                         4=Select subfile control keywords
                                         5=Define screen image (fields)
                                         6=Save DDS source, create object

  Record  . . . . . .                   Name, F4 for Record list
  Additional records  . . .              Name
    (to be displayed
     on Work Screen)

  Source file . . . . . .  MLGSRC        Name
    Library . . . . . .    USERXX        Name, *LIBL, *CURLIB
  Member  . . . . . .      MLGINQD2      Name, F4 for Member list
  Source type . . . . . .  DSPF          DSPF,
                                         Blank for default

F3=Exit      F4=Prompt      F12=Cancel
```

3. Select option 1 (Select file keywords) and fill in the source file (MLGSRC), your library name, member name (MLGINQD2), and source type (DSPF).

   A unique member name is used to allow the DDS-coded member (MLGINQD) to also exist. Later on in this chapter, unique program source will be used to refer to the MLGINQD2 file.

Press the Enter key. The Select File Keywords display is shown.

```
                          Select File Keywords

Member . . . :   MLGINQD2

Type choices, press Enter.

                                      Y=Yes
  General keywords . . . . . .  . .   Y
  Indicator keywords . . . . . . . .
  Print keywords . . . . . . . . . .
  Help keywords . . . . . . . . . . .
  Display sizes . . . . . . . . . . .









F3=Exit   F12=Cancel
```

4. On the Select File Keywords display, type a Y to select general keywords and press the Enter key.

   The Select General Keywords display is shown.

```
                        Select General Keywords

Member . . . :   MLGINQD2

Type choices, press Enter.

                                          Keyword    Y=Yes   Indicators/+
  Invite devices for later read . . . . . .  INVITE
  Allow graphics . . . . . . . . . . . . .  ALWGPH
  Separate indicators area . . . . . . . .  INDARA
  Manage display in S/36 mode . . . . . . .  USRDSPMGT
  Allow blanks . . . . . . . . . . . . . .  CHECK(AB)
  Move cursor right-left, top-bottom . . .  CHECK(RLTB)
  Move cursor right to left . . . . . . . .  CHECK(RL)
  Change input defaults . . . . . . . . . .  CHGINPDFT
    Select parameters . . . . . . . . . . .

  Reference data base file . . . . . . . .  REF          MLGREFP     Name
    Library . . . . . . . . . . . . . . . .                         Name
    Record . . . . . . . . . . . . . . . .                         Name
  Record to pass unformatted data . . . . .  PASSRCD                Name

F3=Exit   F12=Cancel
```

5. Type MLGREFP for the database file to refer to and press the Enter key.

   The Select File Keywords display is shown again to allow you to make additional selections.

```
                          Select File Keywords

Member . . . :   MLGINQD2

Type choices, press Enter.

                                      Y=Yes
  General keywords . . . . . .
  Indicator keywords . . . . . .
  Print keywords . . . . . .  . .   Y
  Help keywords . . . . . .
  Display sizes . . . . . .


F3=Exit   F12=Cancel
```

6. Type a Y to select print keywords.

   The Define Print Keywords display is shown.

```
                        Define Print Keywords

Member . . . :   MLGINQD2

Type choices, press Enter.
                                        Keyword
  Enable keyword . . . . . .  . .       PRINT     Y          Y=Yes
    Indicators . . . . . .

  To have your program handle print:
    Response indicator . . . . . .                         01-99
    Text . . . . . .

  To have the system handle print:
    Print file . . . . . .                                 Name, *PGM
      Library  . . . . . .                                 Name,
                                                           *LIBL
    Leave print file open until
      display file is closed . . . . .  OPENPRT            Y=Yes

F3=Exit   F12=Cancel
```

7. Type a Y to enable the keyword PRINT and press the Enter key.

```
                          Select File Keywords

Member . . . :   MLGINQD2

Type choices, press Enter.

                                        Y=Yes
  General keywords . . . . . .
  Indicator keywords . . . . . .
  Print keywords . . . . . .
  Help keywords . . . . . .
  Display sizes . . . . . .









F3=Exit   F12=Cancel
```

8. Because there are no more file-level keywords to define, press the Enter key.

   The Design Screens display is shown again.

```
                              Design Screens

Type choices, press Enter.

  Design option . . . . . .   2            1=Select file keywords
                                           2=Select record keywords
                                           3=Select subfile keywords
                                           4=Select subfile control keywords
                                           5=Define screen image (fields)
                                           6=Save DDS source, create object

  Record . . . . . .         DSPLY1        Name, F4 for Record list
  Additional records . . .                 Name
    (to be displayed
     on Work Screen)

  Source file . . . . . .    MLGSRC        Name
    Library . . . . . .      USERXX        Name, *LIBL, *CURLIB
  Member . . . . . .         MLGINQD2      Name, F4 for Member list
  Source type . . . . . .    DSPF          DSPF,
                                           Blank for default

F3=Exit     F4=Prompt     F12=Cancel
```

9. Select option 2 for record keywords and type DSPLY1 for the record. DSPLY1 will be the record format you are defining. Press the Enter key.

   The Select Record Keywords display is shown.

```
                        Select Record Keywords

Record . . . :   DSPLY1

Type choices, press Enter.

                                        Y=Yes
  General keywords . . . . . .
  Indicator keywords . . . . . .  Y
  Help sequence . . . . . .
  Application help . . . . . .
  Output keywords . . . . . .
  Input keywords . . . . . .
  Overlay keywords . . . . . .


  TEXT keyword . . . . . .




F3=Exit   F12=Cancel
```

10. Type a Y to select indicator keywords and press the Enter key.

    The Define Indicator Keywords display is shown.

```
                       Define Indicator Keywords

Record . . . :   DSPLY1

Type keywords and parameters, press Enter.
  Conditioned keywords:       CFnn CAnn CLEAR ROLLUP ROLLDOWN HOME HELP HLPRTN
  Unconditioned keywords:     INDTXT VLDCMDKEY SETOF CHANGE

Keyword   Indicators/+ Resp Text
CA03                   93  Exit









                                                                      Bottom
F3=Exit   F12=Cancel
```

11. Type the keyword, response, and text as shown above. This describes the record-level keywords and the associated response indicators. In this case, F3 will be used to set on indicator 93. Press the Enter key.

    The Select Record Keywords display is shown again.

```
                          Select Record Keywords

Record . . . :   DSPLY1

Type choices, press Enter.

                                        Y=Yes
  General keywords  . . . . . .
  Indicator keywords  . . . . . .
  Help sequence . . . . . .
  Application help  . . . . . .
  Output keywords . . . . . .
  Input keywords  . . . . . .
  Overlay keywords  . . . . . .


  TEXT keyword  . . . . . .   Prompt for account number




F3=Exit    F12=Cancel
```

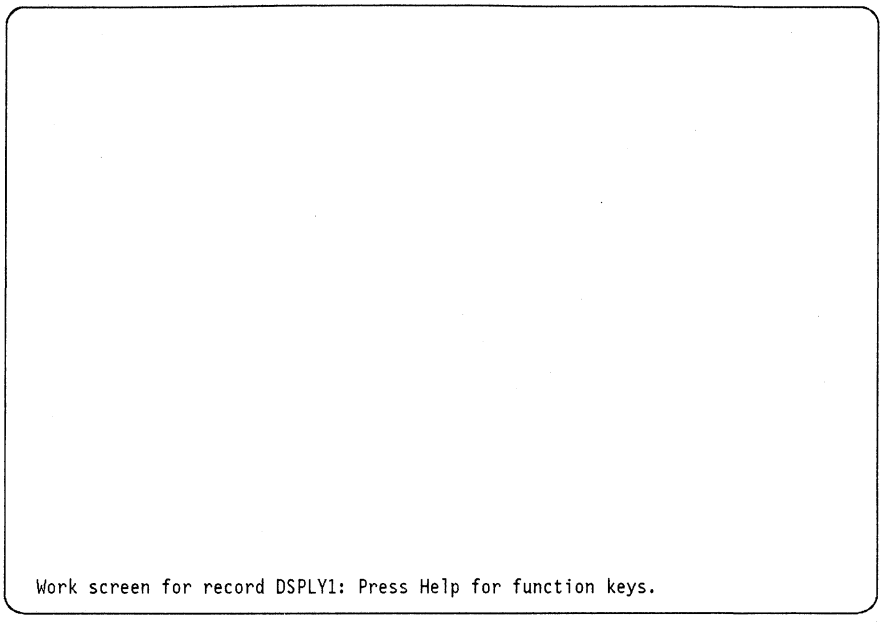
12. Type the information shown above for the TEXT keyword and press the Enter key. The TEXT keyword is only used to help comment the source. The cursor will go to the top of the display. Press the Enter key again.

    The Design Screens display is shown.

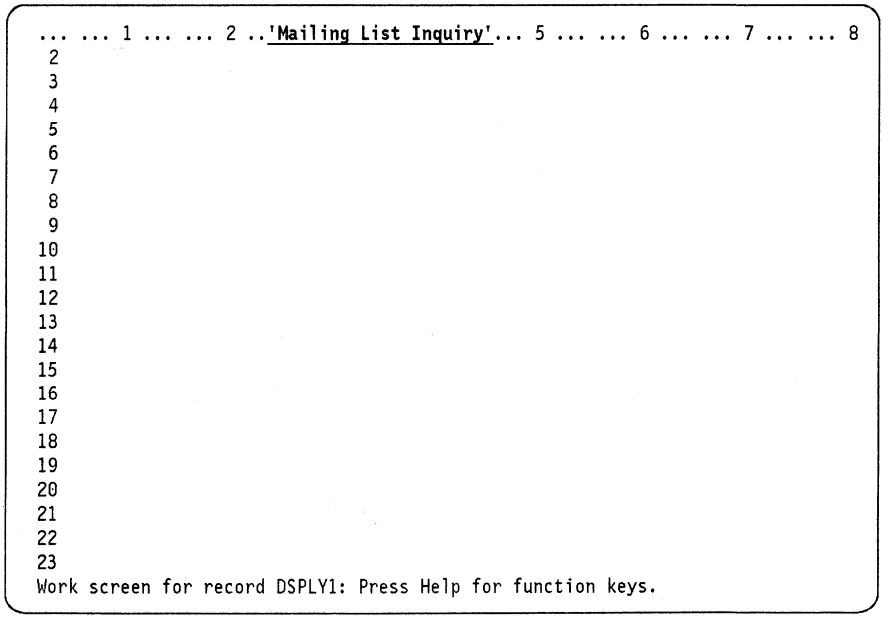
```
                              Design Screens

Type choices, press Enter.

  Design option . . . . . .   5            1=Select file keywords
                                           2=Select record keywords
                                           3=Select subfile keywords
                                           4=Select subfile control keywords
                                           5=Define screen image (fields)
                                           6=Save DDS source, create object

  Record  . . . . . .  . .   DSPLY1        Name, F4 for Record list
  Additional records  . . .                Name
    (to be displayed
     on Work Screen)

  Source file . . . . . .    MLGSRC        Name
    Library . . . . . .      USERXX        Name, *LIBL, *CURLIB
  Member  . . . . . .  . .   MLGINQD2      Name, F4 for Member list
  Source type . . . . . .    DSPF          DSPF,
                                           Blank for default

F3=Exit      F4=Prompt      F12=Cancel
```

13. Select option 5 to define the screen image (fields). The screen image is where you provide the layout of the display. Press the Enter key.

    The Work Screen display is shown.



14. The Work Screen is a blank display on which you design your display. Press F14 to get a ruler across the top of the display and down the left edge. You can see what command keys are valid by pressing the Help key.



15. Type the information as shown above in row 1 column 24 enclosed in apostrophes. Press the Enter key.

    SDA displays the heading you typed and removes the apostrophes.

    Note: If you were entering DDS, you would specify that you want the field to begin in position 25. The leading apostrophe is always entered one position to the left of where the field should begin.

```
... ... 1 ... ... 2 ..HMailing List Inquiry ... 5 ... ... 6 ... ... 7 ... ... 8
 2
 3
 4
 5
 6
 7
 8
 9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
```

16. Type the display attribute H (either upper or lowercase) immediately preceding the *Mailing List Inquiry* field to highlight the title. Press the Enter key. The title will be highlighted.

```
... ... 1 ... ... 2 .. Mailing List Inquiry ... 5 ... ... 6 ... ... 7 ... ... 8
 2
'Account number  . . . . . :'
 4
 5
 6
 7
 8
 9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
```

17. Type the information shown above on row 3 column 1 and press the Enter key. The apostrophes will be removed.

```
... ... 1 ... ... 2 .. Mailing List Inquiry ... 5 ... ... 6 ... ... 7 ... ... 8
 2
Account number  . . . . . :
 4
 5
 6
 7
 8
 9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
```

18. Press F10 to select database files.

```
                          Select Data Base Files

Type options and names, press Enter.
  1=Display data base field list
  2=Select all fields for input (I)
  3=Select all fields for output (O)
  4=Select all fields for both (B) input and output

Option   Data Base File   Library      Record
  1        MLGMSTP                     MLGMSTR









F3=Exit   F4=Record list   F12=Cancel
```

19. Type a 1 (Display database field list) in the *Option* column, type MLGMSTP for the database file (MLGREFP should have been the default shown), and MLGMSTR for the record and then press the Enter key. You are requesting SDA to provide the field definitions for the file and format named.

```
                        Select Data Base Fields

Record . . . :   MLGMSTR

Type information, press Enter.
  Number of fields to roll . . . . . . . . . . . . . . . . . .    8
  Name of field to search for . . . . . . . . . . . . . . . . .

Type options, press Enter.
  1=Display extended field description
  2=Select for input (I), 3=Select for output (O), 4=Select for both (B)

Option  Field       Length   Type    Column Heading
  2     MLACCT         5,0    P      Account number
        MLTYPE           1    A      Type
        MLNAME          20    A      Name
        MLSRCH          10    A      Search
        MLADDR          20    A      Addr
        MLCITY          20    A      City
        MLSTAT           2    A      State
        MLZIP          5,0    P      ZIP code
                                                                  Bottom
F3=Exit   F12=Cancel
```

20. Type a 2 in the *Option* column for the MLACCT field. This requests that the field will be *input* only. The user will see only an underlined area to key into. Press the Enter key.

```
                        Select Data Base Files

Type options and names, press Enter.
  1=Display data base field list
  2=Select all fields for input (I)
  3=Select all fields for output (O)
  4=Select all fields for both (B) input and output

Option   Data Base File   Library      Record
           MLGMSTP        *LIBL        MLGMSTR









F3=Exit   F4=Record list   F12=Cancel
```

21. You have no more database files to describe so press the Enter key to Exit.

```
... ... 1 ... ... 2 .. Mailing List Inquiry ... 5 ... ... 6 ... ... 7 ... ... 8
 2
Account number  . . . . . . . : &1
 4
 5
 6
 7
 8
 9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
1:MLACCT
```

22. The field you selected (MLACCT) will appear at the bottom of the display. The *1* that appears in front of the field describes a label to simplify the entry of where you want the field to appear on the display when you describe where the field is to be positioned, use &1. This tells SDA that the field it knows by the label *1* is being positioned.

    Type &1 following the *Account number* field as shown above and press the Enter key.

    The &1 will be replaced by 33333- which means that this is a numeric input field. The length of the field (the number of 3's) is shown. The minus sign indicates that the user could enter a plus or minus with the field. The account number field can only be positive so an entry will be made on a later display to correct this.

```
... ... 1 ... ... 2 .. Mailing List Inquiry ... 5 ... ... 6 ... ... 7 ... ... 8
 2
Account number  . . . . . . . : 33333-
 4
 5
 6
 7
 8
 9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
```

23. Press F4 to go to the Work with Fields display. This allows you to describe the actual field names to be used for input and output fields and other field attributes.

```
                              Work with Fields

Record . . . :   DSPLY1

Type information, press Enter.
  Number of fields to roll . . . . . .    7

Type options, change values, press Enter.
  1=Select keywords   4=Delete field

Option   Order   Field       Type Use  Length  Row/Col  Ref Condition  Overlap
           10    Mailing Li    C          20   01 025
           20    Account nu    C          31   03 002
  1        30    ACCT              I     5,0   03 034   Y

                                                                        Bottom
 Add                                  H           Hidden
 Add                                  M           Message

F3=Exit   F6=Sort by row/column   F12=Cancel
```

24. The Work with Fields display describes each of the fields defined on the work screen, its attributes, and location. The program uses a different field name (ACCT) than the key field in the database (MLACCT). You will want to specify the actual field name to be used. An abbreviated version of the constants you entered is also shown.

    Type a 1 (Select keywords) for order number 30 and type ACCT over the field value of MLACCT. Press the Enter key.

```
                          Select Field Keywords

Field . . . . . :   ACCT              Usage . . :  I
Length  . . . . :   5,0               Row . . . :  3    Column . . . :  34

Type choices, press Enter.
                                        Y=Yes   For Field Type
  Display attributes  . . . . . .               All except Hidden
  Colors  . . . . . . . . . . . .               All except Hidden
  Keying options  . . . . . . . .   Y           Input or Both
  Validity check  . . . . . . . .               Input or Both, not float
  Input keywords  . . . . . . . .               Input or Both
  General keywords  . . . . . . .               All types

  Data base reference . . . . . .               Hidden, Input, Output, Both
  Error messages  . . . . . . . .               Input, Output, Both


  TEXT keyword  . . . . . . . . .   Account number


F3=Exit   F12=Cancel
```

25. Type a Y for keying options and press the Enter key.

    The Select Keying Options display is shown.

```
                        Select Keying Options

Field . . . . . :   ACCT              Usage . . :  I
Length  . . . . :   5,0               Row . . . :  3    Column . . . :  34

Type choices, press Enter.

                                       Keyword  Y=Yes  Indicators/+
  Keying options:                      CHECK
    Mandatory entry . . . . . . . . .    ME
    Automatic record advance  . . . .    ER
    Mandatory fill  . . . . . . . . .    MF
    Field exit key required . . . . .    FE
    Right adjust blank fill . . . . .    RB
    Right adjust zero fill  . . . . .    RZ


  Keyboard shift attribute  . . . . . .          D    S N Y I D


F3=Exit   F12=Cancel
```

26. Type a D for the keyword shift attribute. You are specifying to SDA that this field is a digits-only field. This will cause 33333 to be displayed instead of 33333-. The user cannot enter a plus or minus value.

    Press the Enter key.

```
                        Select Field Keywords

Field . . . . . :   ACCT              Usage . . :  I
Length  . . . . :   5,0               Row . . . :  3    Column . . . :  34

Type choices, press Enter.
                                     Y=Yes   For Field Type
  Display attributes  . . . . . .             All except Hidden
  Colors  . . . . . . . . . . . .             All except Hidden
  Keying options  . . . . . . . .             Input or Both
  Validity check  . . . . . . . .             Input or Both, not float
  Input keywords  . . . . . . . .             Input or Both
  General keywords  . . . . . . .             All types

  Data base reference . . . . . .             Hidden, Input, Output, Both
  Error messages  . . . . . . . .  Y          Input, Output, Both


  TEXT keyword  . . . . . . . . .  Account number


F3=Exit   F12=Cancel
```

27. Type a Y for error messages and press the Enter key.

    The Define Error Messages display is shown.

```
                          Define Error Messages

Field . . . . . :   ACCT              Usage . . :  I
Length  . . . . :   5,0               Row . . . :  3    Column . . . :  34

Type parameters, press Enter.

  Indicators/+    ERRMSG - Message Text                                 Ind
   41             The account number cannot be zeros                    41
   42             The account number does not exist                     42




                                                                      Bottom

  Indicators/+    ERRMSGID  File        Library




                                                                      Bottom
F3=Exit   F12=Cancel
```

28. Type the information shown above for the *Indicators/+*, *ERRMSG - Message Text*, and *Ind* fields to describe the error messages. The indicators on the left are the option indicators. The first ERRMSG keyword will only be active when indicator 41 is on. The indicators on the right are defined to be set off when control returns to the program. Press the Enter key.

```
                          Select Field Keywords

Field . . . . . :   ACCT              Usage . . :  I
Length  . . . . :   5,0               Row . . . :  3    Column . . . :  34

Type choices, press Enter.
                                    Y=Yes   For Field Type
  Display attributes  . . . . . .           All except Hidden
  Colors  . . . . . .                       All except Hidden
  Keying options  . . . . . .               Input or Both
  Validity check  . . . . . .               Input or Both, not float
  Input keywords  . . . . . .               Input or Both
  General keywords  . . . . . .             All types

  Data base reference . . . . . .           Hidden, Input, Output, Both
  Error messages  . . . . . .               Input, Output, Both




  TEXT keyword  . . . . . .   Account number

F3=Exit   F12=Cancel
```

29. The ACCT field is now fully defined. Press the Enter key to go to the Work with Fields display again.

```
                              Work with Fields

Record . . . :   DSPLY1

Type information, press Enter.
  Number of fields to roll . . . . . .     7

Type options, change values, press Enter.
  1=Select keywords   4=Delete field

Option   Order   Field       Type Use  Length  Row/Col  Ref Condition  Overlap
           10    Mailing Li    C          20   01 025
           20    Account nu    C          31   03 002
           30    ACCT          D   I     5,0   03 034   Y

                                                                        Bottom
 Add                               H           Hidden
 Add                               M           Message

F3=Exit   F6=Sort by row/column   F12=Cancel
```

30. Press the Enter key to go to the Work Screen.

```
... ... 1 ... ... 2 .. Mailing List Inquiry ... 5 ... ... 6 ... ... 7 ... ... 8
 2
Account number  . . . . . : 33333
 4
 5
 6
 7
 8
 9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
'F3=Exit'
```

31. Notice that the value 33333 has changed from 33333- by use of the *digits-only* function. Type the information shown on row 24 column 1 to define the exit function key and then press the Enter key.

    The apostrophes will disappear.

```
... ... 1 ... ... 2 .. Mailing List Inquiry ... 5 ... ... 6 ... ... 7 ... ... 8
 2
Account number  . . . . . : 33333
 4
 5
 6
 7
 8
 9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
F3=Exit
```

32. The first format (DSPLY1) is now fully defined. Press F3 to Exit.

```
                              Design Screens

Type choices, press Enter.

  Design option . . . . . .   2            1=Select file keywords
                                           2=Select record keywords
                                           3=Select subfile keywords
                                           4=Select subfile control keywords
                                           5=Define screen image (fields)
                                           6=Save DDS source, create object

  Record  . . . . . .  . .   DSPLY2       Name, F4 for Record list
  Additional records  . . .                Name
    (to be displayed
     on Work Screen)

  Source file . . . . . .    MLGSRC        Name
    Library . . . . . .      USERXX        Name, *LIBL, *CURLIB
  Member  . . . . . .  . .   MLGINQD2      Name, F4 for Member list
  Source type . . . . . .    DSPF          DSPF,
                                           Blank for default

F3=Exit     F4=Prompt     F12=Cancel
```

33. Type a 2 (Select record keywords) for the design option and DSPLY2 for the record and press the Enter key.

    The Select Record Keywords display is shown.

```
                        Select Record Keywords

Record . . . :   DSPLY2

Type choices, press Enter.

                                      Y=Yes
  General keywords . . . . . .
  Indicator keywords . . . . . .   Y
  Help sequence . . . . . .
  Application help . . . . . .
  Output keywords . . . . . .
  Input keywords . . . . . .
  Overlay keywords . . . . . .


  TEXT keyword . . . . . .




F3=Exit   F12=Cancel
```

34. Type a Y for indicator keywords and press the Enter key.

```
                        Define Indicator Keywords

Record . . . :   DSPLY2

Type keywords and parameters, press Enter.
  Conditioned keywords:       CFnn CAnn CLEAR ROLLUP ROLLDOWN HOME HELP HLPRTN
  Unconditioned keywords:     INDTXT VLDCMDKEY SETOF CHANGE

Keyword   Indicators/+ Resp Text
CA03                    93  Exit
                                                                      Bottom
F3=Exit   F12=Cancel
```

35. Type the information shown above for the *Keyword, Resp,* and *Text* fields. This defines the F3 key to set on indicator 93. Press the Enter key.

```
                        Select Record Keywords

Record . . . :   DSPLY2

Type choices, press Enter.

                                      Y=Yes
  General keywords  . . . . . .
  Indicator keywords  . . . . . .
  Help sequence . . . . . .
  Application help  . . . . . .
  Output keywords . . . . . .
  Input keywords  . . . . . .
  Overlay keywords  . . . . . .


  TEXT keyword  . . . . . .   MLGMSTP record display




F3=Exit   F12=Cancel
```

36. Type the information shown above for the *TEXT keyword* field. The text is used to help document the source. Press the Enter key.

```
                        Select Record Keywords

Record . . . :   DSPLY2

Type choices, press Enter.

                                      Y=Yes
  General keywords  . . . . . .
  Indicator keywords  . . . . . .
  Help sequence . . . . . .
  Application help  . . . . . .
  Output keywords . . . . . .
  Input keywords  . . . . . .
  Overlay keywords  . . . . . .


  TEXT keyword  . . . . . .   MLGMSTP record display




F3=Exit   F12=Cancel
```

37. Press the Enter key again.

```
                              Design Screens

Type choices, press Enter.

  Design option . . . . . .  5           1=Select file keywords
                                         2=Select record keywords
                                         3=Select subfile keywords
                                         4=Select subfile control keywords
                                         5=Define screen image (fields)
                                         6=Save DDS source, create object

  Record  . . . . . .  . .  DSPLY2       Name, F4 for Record list
  Additional records  . . .              Name
    (to be displayed
     on Work Screen)

  Source file . . . . . .  .  MLGSRC     Name
    Library . . . . . .  . .  USERXX     Name, *LIBL, *CURLIB
  Member  . . . . . .  . . .  MLGINQD2   Name, F4 for Member list
  Source type . . . . . .  .  DSPF       DSPF,
                                         Blank for default

F3=Exit     F4=Prompt     F12=Cancel
```

38. Type a 5 (Define screen image (fields)) for the *Design option* field and press the Enter key.

```
... ... 1 ... ... 2 ..'Mailing List Inquiry'... 5 ... ... 6 ... ... 7 ... ... 8
 2
 3
 4
 5
 6
 7
 8
 9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
Work screen for record DSPLY2: Press Help for function keys.
```

39. Type the information as shown above in row 1 column 24 enclosed in apostrophes and then press the Enter key. The apostrophes will disappear.

```
... ... 1 ... ... 2 ..HMailing List Inquiry ... 5 ... ... 6 ... ... 7 ... ... 8
 2
 3
 4
 5
 6
 7
 8
 9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
```

40. Type the display attribute H preceding the *Mailing List Inquiry* field. Press the Enter key. The title will be highlighted.

```
... ... 1 ... ... 2 .. Mailing List Inquiry ... 5 ... ... 6 ... ... 7 ... ... 8
 2
'Account number  . . . . . :'
'Account type  . . . . . :'
'Name  . . . . . :'
'Search name . . . . . :'
'Address . . . . . :'
'City  . . . . . :'
'State . . . . . :'
'Zip code  . . . . . :'
11
12
13
14
15
16
17
18
19
20
21
22
23
'F3=Exit'
```

41. Type the information shown above in rows 3 through 10 and 24. Be sure to use the surrounding quotes. Press the Enter key.

```
... ... 1 ... ... 2 .. Mailing List Inquiry ... 5 ... ... 6 ... ... 7 ... ... 8
 2
Account number  . . . . . :
Account type  . . . . . :
Name  . . . . . :
Search name . . . . . :
Address . . . . . :
City  . . . . . :
State . . . . . :
Zip code  . . . . . :
11
12
13
14
15
16
17
18
19
20
21
22
23
F3=Exit
```

42. Press F10 to go to the Select Database Files display.

```
                          Select Data Base Files

Type options and names, press Enter.
  1=Display data base field list
  2=Select all fields for input (I)
  3=Select all fields for output (O)
  4=Select all fields for both (B) input and output

Option   Data Base File   Library      Record
  3        MLGMSTP                     MLGMSTR









F3=Exit   F4=Record list   F12=Cancel
```

43. Type a 3 (Select all fields for output (O)) for the *Option* field (all fields on this display will be output only), MLGMSTP for the Database file, and MLGMSTR for the Record as shown above. Press the Enter key.

```
                          Select Data Base Files

Type options and names, press Enter.
  1=Display data base field list
  2=Select all fields for input (I)
  3=Select all fields for output (O)
  4=Select all fields for both (B) input and output

Option   Data Base File   Library      Record
           MLGMSTP        *LIBL        MLGMSTR









F3=Exit   F4=Record list   F12=Cancel
```

44. Press the Enter key again.

```
... ... 1 ... ... 2 .. Mailing List Inquiry ... 5 ... ... 6 ... ... 7 ... ... 8
 2
Account number  . . . . . :  &1
Account type  . . . . . :  &2
Name  . . . . . :  &3
Search name . . . . . :  &4
Address . . . . . :  &5
City  . . . . . :  &6
State . . . . . :  &7
Zip code  . . . . . :  &8
11
12
13
14
15
16
17
18
19
20
21
22
23
1:MLACCT 2:MLTYPE 3:MLNAME 4:MLSRCH 5:MLADDR 6:MLCITY 7:MLSTAT 8:MLZIP
```

45. The database fields you can specify (and the number by which they are known to SDA) appear at the bottom of the display. Type the information as shown above to position the database fields and their associated attributes to the correct positions. Press the Enter key.

    The database field attributes are shown where an O indicates an alphabetic output field and a 6 indicates a numeric output field.

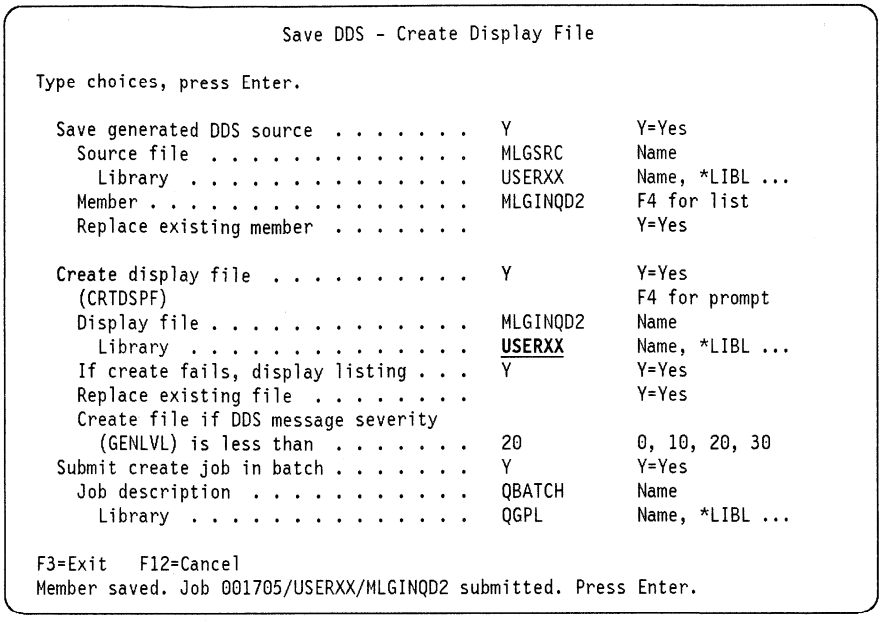
```
... ... 1 ... ... 2 .. Mailing List Inquiry ... 5 ... ... 6 ... ... 7 ... ... 8
 2
Account number  . . . . . :  66666
Account type  . . . . . :  0
Name  . . . . . :  0000000000000000000
Search name . . . . . :  0000000000
Address . . . . . :  0000000000000000000
City  . . . . . :  0000000000000000000
State . . . . . :  00
Zip code  . . . . . :  66666
11
12
13
14
15
16
17
18
19
20
21
22
23
F3=Exit
```

46. The second format (DSPLY2) is now fully defined. Press F3 (Exit).

```
                              Design Screens

Type choices, press Enter.

  Design option . . . . . .   6           1=Select file keywords
                                          2=Select record keywords
                                          3=Select subfile keywords
                                          4=Select subfile control keywords
                                          5=Define screen image (fields)
                                          6=Save DDS source, create object

  Record  . . . . . .  .  .   DSPLY2      Name, F4 for Record list
  Additional records  . . .               Name
    (to be displayed
     on Work Screen)

  Source file . . . . . .  .  MLGSRC      Name
    Library . . . . . .  .  .  USERXX      Name, *LIBL, *CURLIB
  Member  . . . . . .  .  .  .  MLGINQD2   Name, F4 for Member list
  Source type . . . . . .  .  DSPF        DSPF,
                                          Blank for default
F3=Exit     F4=Prompt     F12=Cancel
```

47. The display file is now fully defined. Select option 6 to save the DDS source and create the display file. Press the Enter key.

    The Save DDS display is shown.



48. Type your library name as shown above and press the Enter key to:

    - Save the DDS source created by SDA
    - Create a display file (MLGINQD2) from the DDS source
    - Display a spooled listing if the create fails
    - Submit MLGINQD2 as a batch job

    You will receive a completion message when the submitted batch job has created the display file.

49. Press F3 to return to the Screen Design Aid menu.

Testing the Display File

SDA provides a test function to assist you in determining that your display file is properly coded.

Follow these steps to test the display file:

1. If you are not already on the Screen Design Aid (SDA) display, type the STRSDA command on the command entry line.

```
                        AS/400 Screen Design Aid (SDA)

Select one of the following:

     1. Design screens
     2. Design menus
     3. Test display files









Selection or command
===> 3

F3=Exit    F4=Prompt    F9=Retrieve    F12=Cancel
```

2. Select option 3 to test display files and press the Enter key.

```
                              Test Display File

Type choices, press Enter.

  Display file . . . . . . . . . . . . .   MLGINQD2      Name
    Library  . . . . . . . . . . . . . .   USERXX        Name,
                                                        *LIBL ...
  Record to be tested  . . . . . . . . .   DSPLY1       Name,
                                                        F4 for list
  Additional records to display  . . . .                Name









F3=Exit      F4=Prompt      F12=Cancel
```

3. The prompts on the Test Display File display will be filled in with the values of the last file created or used. If this is not the same file you just created, change the entries as shown above.

   Press the Enter key.

```
                          Set Test Output Data

Record . . . :   DSPLY1

Type indicators and output field values, press Enter.

Field         Value
*IN41         0:
*IN42         0:

                                                                  Bottom
F3=Exit   F12=Cancel
```

4. The Set Test Output Data display allows you to modify the field values and option indicators. First, press the Enter key without entering any values.

   The Mailing List Inquiry display is shown.

```
                        Mailing List Inquiry

Account number  . . . . . . . :




F3=Exit
```

5. Press F3 (Exit).

```
                        Display Test Input Data

Record . . . :   DSPLY1

View indicators and input field values.

Field         Value
*IN93         1:
*IN41         0:
*IN42         0:
ACCT          00000:









                                                                  Bottom
Press Enter to continue.

F3=Exit   F12=Cancel     F14=Display input buffer
```

6. The Display Test Input Data display shows the status of the field values that would be sent to the program. Notice that *IN93 (indicator 93) has a value of 1 on the return. This indicator is set by the CA03 keyword.

   Press the Enter key.

```
                          Set Test Output Data

Record . . . :   DSPLY1

Type indicators and output field values, press Enter.

Field         Value
*IN41         1:
*IN42         0:









                                                                  Bottom

F3=Exit   F12=Cancel
```

7. Specify a value of 1 for *IN41. This will simulate indicator 41 being on when the display is written. Indicator 41 controls an ERRMSG keyword. Press the Enter key.

```
                    Mailing List Inquiry

Account number  . . . . . . . :



The account number cannot be zeros
```

8. You receive a highlighted error message stating that the account number cannot be zeros. Press Reset, then press F3 (Exit).

```
                        Display Test Input Data

Record . . . :   DSPLY1

View indicators and input field values.

Field         Value
*IN93         1:
*IN41         0:
*IN42         0:
ACCT          00000:




                                                                    Bottom
Press Enter to continue.

F3=Exit   F12=Cancel     F14=Display input buffer
```

9. The Display Test Input Data display shows the values being passed back to the program.

   Press the Enter key.

```
                          Set Test Output Data

Record . . . :   DSPLY1

Type indicators and output field values, press Enter.

Field         Value
*IN41         0:
*IN42         0:

                                                                  Bottom

F3=Exit   F12=Cancel
```

10. On the Set Test Output Data display you can try setting indicator 42 on (1) to see the results. When done, press Reset, then press F3.
11. Press F12 (Cancel) to return to the Test Display File display.

```
                          Test Display File

Type choices, press Enter.

  Display file . . . . . . . . . . . .  MLGINQD2     Name
    Library  . . . . . . . . . . . . .  USERXX       Name,
                                                     *LIBL ...
  Record to be tested  . . . . . . . .  DSPLY2       Name,
                                                     F4 for list
  Additional records to display  . . . .             Name

F3=Exit     F4=Prompt     F12=Cancel
```

12. On the Test Display File display, type DSPLY2 for the next record to be tested and press the Enter key.

```
                          Set Test Output Data

Record . . . :   DSPLY2

Type indicators and output field values, press Enter.

Field         Value
MLACCT        66666:
MLTYPE        0:
MLNAME        Joe Jones
MLSRCH        0000000000:
MLADDR        00000000000000000000:
MLCITY        00000000000000000000:
MLSTAT        00:
MLZIP         66666:

                                                                  Bottom

F3=Exit   F12=Cancel
```

13. Type Joe Jones for the value for field MLNAME. This will simulate a value being passed from the program when the display is written. Press the Enter key.

```
                        Mailing List Inquiry

Account number . . . . . :  66666
Account type . . . . . :  0
Name . . . . . :  Joe Jones
Search name . . . . . :  0000000000
Address . . . . . :  00000000000000000000
City . . . . . :  00000000000000000000
State . . . . . :  00
Zip code . . . . . :  66666

F3=Exit
```

14. Press the Enter key.

    The value for indicator 93 is shown.

```
                        Display Test Input Data

Record . . . :   DSPLY2

View indicators and input field values.

Field         Value
*IN93         0:









                                                                    Bottom
Press Enter to continue.

F3=Exit    F12=Cancel      F14=Display input buffer
```

15. Notice that *IN93 is set to 0. Press the Enter key.

```
                          Set Test Output Data

Record . . . :   DSPLY2

Type indicators and output field values, press Enter.

Field         Value
MLACCT        66666:
MLTYPE        0:
MLNAME        Joe Jones           :
MLSRCH        0000000000:
MLADDR        00000000000000000000:
MLCITY        00000000000000000000:
MLSTAT        00:
MLZIP         66666:









                                                                    Bottom

F3=Exit    F12=Cancel
```

16. Assuming you are satisfied with the testing of the file, press F3 (Exit).

```
                        AS/400 Screen Design Aid (SDA)

Select one of the following:

     1. Design screens
     2. Design menus
     3. Test display files









Selection or command
===>

F3=Exit    F4=Prompt    F9=Retrieve    F12=Cancel
```

17. Press F3 (Exit).

The MLGINQD2 display file has now been created and tested.

Using the SDA-Created File

To use the SDA-created file with a program, do the following steps:

1. Copy the RPG program MLGINQR and name the new program MLGINQR2 by typing the following on the command entry line.

   ```
   CPYSRCF FROMFILE(MLGSRC) TOFILE(MLGSRC) FROMMBR(MLGINQR) TOMBR(MLGINQR2)
   ```

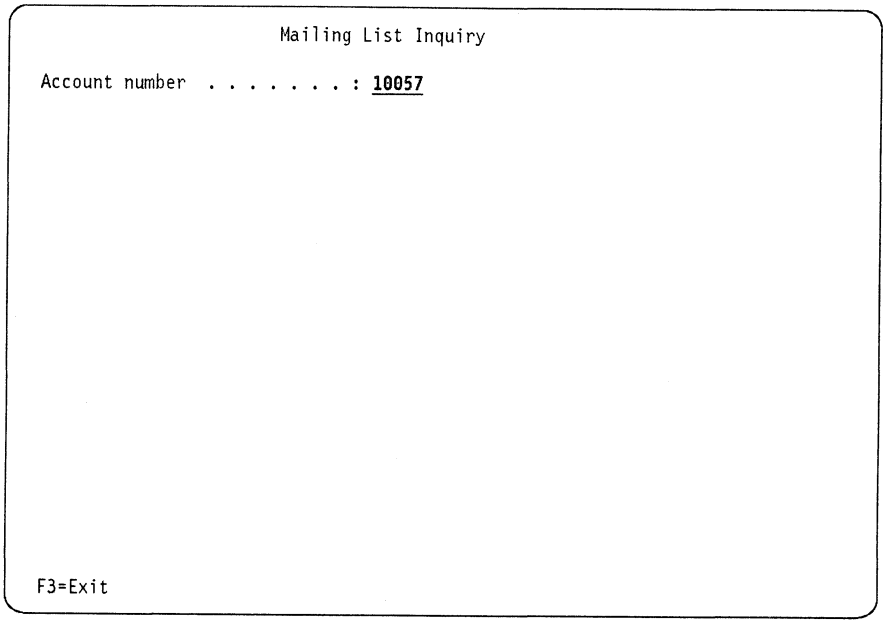
2. Go to the PDM Work with Members display by typing STRPDM, selecting option 3, and typing MLGSRC for the filename and typing your library name.
3. Next, type a 2 (Edit) in the *Opt* column next to the MLGINQR2 program to change the program. The SEU Edit display is shown as follows:

```
 Columns . . . .:   1  71              Edit                    USERXX/MLGSRC
 Find  . . .                                                       MLGINQR2
 FMT *  ..... *. 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7
       ********** Beginning of data **************************************
0001.00      F* MLGINQR - Mailing List Inquiry
0002.00      FMLGMSTP IF  E           K        DISK
0003.00      FMLGINQD2CF  E                    WORKSTN
0004.00      C* Prompt for account number
0005.00      C           PROMPT    TAG                              Prompt dsp
0006.00      C                     EXFMTDSPLY1                      Display 1
0007.00      C* Check for F3
0008.00      C   93                GOTO ENDPGM                      If F3
0009.00      C* If account number is zeros, show prompt again with error msg
0010.00      C           ACCT      CABEQ*ZEROS    PROMPT         41 If zero
0011.00      C* Chain to account number record in file
0012.00      C           ACCT      CHAINMLGMSTR              42    Chain
0013.00      C* If missing, show prompt again with error message
0014.00      C   42                GOTO PROMPT                      Loop back
0015.00      C* Display the record
0016.00      C                     EXFMTDSPLY2                      Display

 F3=Exit                  F4=Prompt                F5=Refresh
 F10=Top                  F11=Bottom               F24=More keys
                                  (C) COPYRIGHT IBM CORP. 1981, 1989.
```

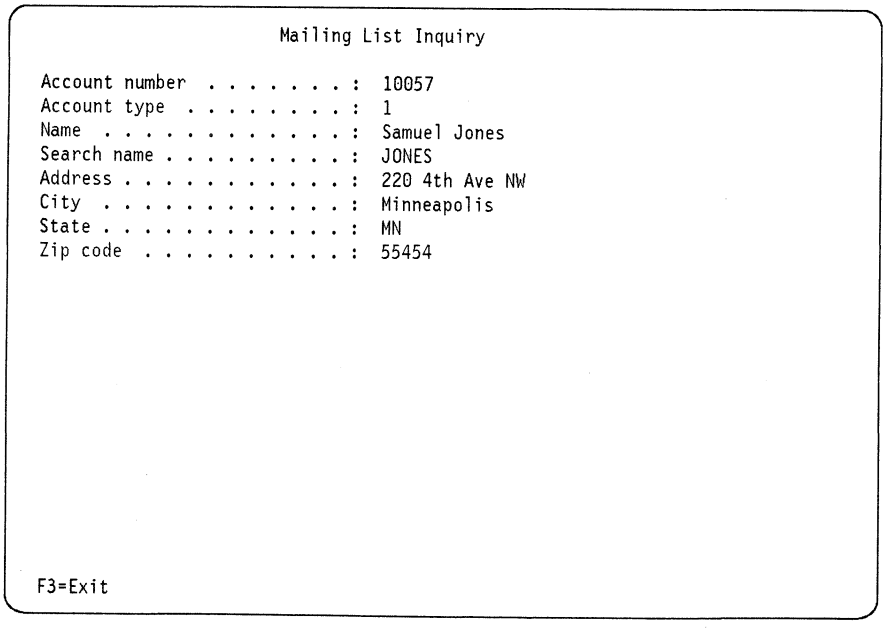
4. On line 3.00, change the file name of MLGINQD to MLGINQD2 and press the Enter key.
5. Press F3 (Exit).
6. Press the Enter key to make the change to the member.
7. Type a 14 (Compile) in the *Opt* column next to the MLGINQR2 program to compile the program in batch.
8. When you have received the completion message that the program was successfully created, call the program by typing the following:

   ```
   CALL MLGINQR2
   ```

   The Mailing List Inquiry display is shown.



9. Enter an account number to test the function.



10. Press F3 (Exit).

Using SDA for Menus

SDA offers a design menu option. When this option is requested, you are prompted for a standard menu layout and the commands that will be run for each menu option. The output of this solution is two source members, a display file, and a menu object. The menu object is accessed by the GO command.

SDA-created menus are effective for simple solutions, but have a restriction that only one command can be entered per menu option. This application requires authorization checking for one of the options. If this had not been the case, the SDA functions to create a menu could have been used. An alternative solution would have been to put the check for authorization into the MLGMTNC program and not in the menu program.

It is possible to use the GO command to cause a menu that will be displayed by a CL program to occur. The CRTMNU command supports the capability to name a program to be run. The program must accept a specific parameter list. See the discussion of CRTMNU in the *CL Programmer's Guide*.

Chapter 15. Additional Topics

This chapter discusses some additional ways in which you could have used the AS/400 system throughout this manual. Included in this chapter are the following topics:

- Using the Command Entry display
- Granting authorizations
- Describing and displaying the program stack
- Using additional PDM functions
- Describing some selecting and sequencing options
- Backup and recovery

Command Entry Display

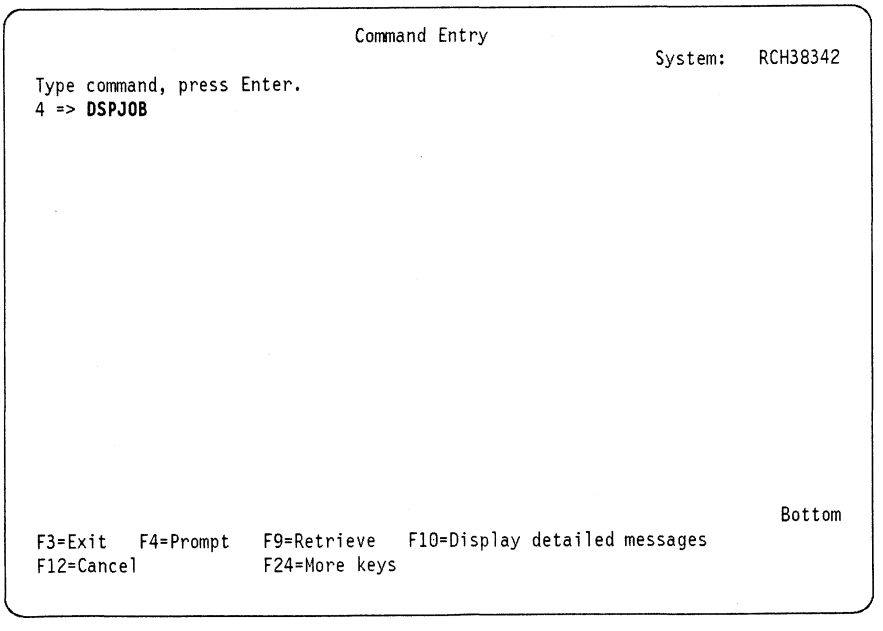
In the prior examples, commands have been entered on the command line of a menu such as the main menu or PDM. Another method is available called the Command Entry display that allows a different approach to entering commands. It is generally more useful when you want to enter several commands. Follow these steps.

Note: You can access the command entry display in PDM by pressing F10 (Command entry).

1. You can access the command entry function by typing:

   ```
   CALL QCMD
   ```

   The Command Entry display is shown.



2. Commands can be entered as you did previously. For example, enter the DSPJOB command as shown above and press the Enter key.

   The Display Job display is shown.

```
                                Display Job
                                                          System:   RCH38342
Job:   DSP060604      User:   USERXX         Number:   001507

Select one of the following:

     1. Display status attributes
     2. Display definition attributes
     3. Display run attributes, if active
     4. Display spooled files

    10. Display job log, if active or on job queue
    11. Display program stack, if active
    12. Display locks, if active
    13. Display library list, if active
    14. Display open files, if active
    15. Display file overrides, if active
    16. Display commitment control status, if active
                                                                        More...
Selection


F3=Exit   F12=Cancel
```

3. Press F3 (Exit).

   The Command Entry display is shown again.

```
                              Command Entry
                                                          System:   RCH38342
Type command, press Enter.
   > DSPJOB
4 => CRTSRCPF XXXX1

                                                                        Bottom
F3=Exit   F4=Prompt   F9=Retrieve   F10=Display detailed messages
F12=Cancel            F24=More keys
```

4. Notice that the DSPJOB command is still shown. The Command Entry display will show both commands and any messages produced by the commands. For example, enter the following:

   CRTSRCPF XXXX1

   You receive the following display with a message.

```
                              Command Entry
                                                          System:   RCH38342
Type command, press Enter.
   > DSPJOB
   > CRTSRCPF XXXX1
     File XXXX1 created in library USERXX.
4 => DLTF XXXX1



                                                                        Bottom
F3=Exit   F4=Prompt   F9=Retrieve   F10=Display detailed messages
F12=Cancel            F24=More keys
```

5. Now, enter the following command:

   DLTF XXXX1

   You get the following display with a message added that the file was deleted.

```
                              Command Entry
                                                          System:   RCH38342
Type command, press Enter.
   > DSPJOB
   > CRTSRCPF XXXX1
     File XXXX1 created in library USERXX.
   > DLTF XXXX1
     Object XXXX1 in USERXX type *FILE deleted.
4 =>



                                                                        Bottom
F3=Exit   F4=Prompt   F9=Retrieve   F10=Display detailed messages
F12=Cancel            F24=More keys
```

6. You can retrieve a previously entered command using F9. If you press F9 while the cursor is on the entry line, it will duplicate the previous command. Press F9 now, then press the Enter key. You will see the following:

```
                              Command Entry
                                                          System:   RCH38342
Type command, press Enter.
   > DSPJOB
   > CRTSRCPF XXXX1
     File XXXX1 created in library USERXX.
   > DLTF XXXX1
     Object XXXX1 in USERXX type *FILE deleted.
   > DLTF XXXX1
     Object XXXX1 in *LIBL type *FILE not found.
4 =>









                                                                      Bottom
F3=Exit    F4=Prompt    F9=Retrieve    F10=Display detailed messages
F12=Cancel              F24=More keys
```

7. Notice that the message states that the file is not found. The DLTF command has failed because there is no longer such a file to delete.

   The Command Entry display allows access to lower-level messages if they exist. Press F10 to display all messages.

```
                           Display All Messages
                                                          System:   RCH38342
    Object XXXX1 in *LIBL type *FILE not found.









                                                                      Bottom
Press Enter to continue.

F3=Exit     F12=Cancel
```

8. Press the Enter key to exit.

   If you move the cursor to a previous command and press F9, the command is duplicated on the entry line. Move the cursor up to the DSPJOB command and press F9.

   You will see the command duplicated on the entry line. Press Enter to run the DSPJOB command and then press F3 to exit from the Display Job display.

   Note: F9 can also be used on any display with a command entry line to retrieve any previous commands that have been run. Using F9 on the

Command Entry display is different only in that you can physically move your cursor to the command you want to retrieve.

```
                                Command Entry
                                                          System:   RCH38342
Type command, press Enter.
   > DSPJOB
   > CRTSRCPF XXXX1
     File XXXX1 created in library USERXX.
   > DLTF XXXX1
     Object XXXX1 in USERXX type *FILE deleted.
   > DLTF XXXX1
     Object XXXX1 in *LIBL type *FILE not found.
4 => DSPJOB









                                                                        Bottom
F3=Exit    F4=Prompt    F9=Retrieve    F10=Display detailed messages
F12=Cancel              F24=More keys
```

9. You can prompt for commands using F4 in the same manner you do on a single command entry line.

   When a command fails (such as the second DLTF command), an escape message is sent. Move the cursor to the DLTF message which says *Object XXXX1 in *LIBL type *FILE not found* and press Help. You should see the following display:

```
                        Additional Message Information

Message ID . . . . . :   CPF2105          Severity . . . . . :   40
Message type . . . . :   ESCAPE
Date sent . . . . . :   03/22/89          Time sent . . . . . :   10:44:54
From program . . . . :   QLIDLOBJ         Instruction . . . . :   021E
To program . . . . . :   QCMD             Instruction . . . . :   0141

Message . . . . :   Object XXXX1 in *LIBL type *FILE not found.
Cause . . . . . :   No object was found for the name or type specified.
Recovery . . . :   Specify the correct name or type of the object. Then try
  the request again.









                                                                        Bottom
Press Enter to continue.

F3=Exit          F12=Cancel
```

10. Press F3 (Exit).

The Command Entry display lets you page up to all of the commands you previously entered on the display. Many programmers find this more practical to use than the single command entry line. You may choose to use the system request function for establishing a secondary job so you could have one job for PDM and one for command entry.

Group jobs may also be used to establish another job where the Attention key can be used. For an example of a simple use of group jobs, refer to the ATNPGM tool in the QUSRTOOL library.

Granting Authorizations

If your system is operated with a security level of 30, you can authorize users to individual objects if you have the authority. To determine your system's level of security, specify:

```
DSPSYSVAL SYSVAL(QSECURITY)
```

When an object is created, you can specify how the public should be authorized. The default for a database file is that the public (meaning anyone who can access the system) can use and change the data within the file. You could also specify that the public is not authorized. If you only want certain users to be able to use (read) or change a file, you must grant them authority.

The objectives of the mailing list application require that only certain users should be able to change the data in the file. In general, this is good practice to limit the number of individuals who can change certain data in your system. For this file, the objectives allow the public to be authorized to read the file. You may have some files where only certain users (not the public) should be able to read the file.

Normally the owner of the file is the user who will control the security to the file. He must issue the authorization changes.

The system supports two general categories of security rights called *USE and *CHANGE. In most cases, these can be used for typical authorizations.

The entry of *USE means that the user has the authority for such things as reading a database file, calling a program, or operating with a display file. It means different things depending on the object type, but generally speaking it means that a user can do whatever is normal to the object without changing anything.

The *CHANGE entry is mostly used with objects that contain data such as database files. It means that the user can change the data in the file. This includes the right to add new records and change or delete existing records. If you want to be more specific (some users can change records, but not delete them), the system supports a solution.

There are other rights to the file which are normally left to the owner such as whether the object description can be changed or whether the object itself can be deleted.

Example of Granting Authorization

The following discussion assumes that your system uses a level 30 for security. (See the previous section for determining your system level.)

You should also check to see what authorities you have. If you have *ALLOBJ authority, you cannot perform this example. Display your authorities by typing the following command (using your user profile name in place of USERXX):

```
DSPUSRPRF USERXX
```

Look at the Special authority field to see if *ALLOBJ is listed, then press F3 to Exit.

Assuming you are on a level 30 system, do not have *ALLOBJ authority, and are the owner of the file (normally the user who created the file), you can simulate the authority check made by the CL program by revoking your own authority. Because you are the owner, you will be able to re-authorize yourself after the test. Follow these steps:

1. Go to the Command Entry display by typing the following command:

   ```
   CALL QCMD
   ```

2. Type the EDTOBJAUT command (Edit Object Authority) as:

   ```
   EDTOBJAUT OBJ(MLGMSTP) OBJTYPE(*FILE)
   ```

   The Edit Object Authority display is shown.

```
                          Edit Object Authority

Object . . . . . :   MLGMSTP         Object type  . . . . :   *FILE
  Library  . . . . :   USERXX         Owner  . . . . . . . :   USERXX

Type changes to current authorities, press Enter.

  Object secured by authorization list  . . . . . . . . . . .   *NONE

             Object
User         Authority
USERXX       *ALL
*PUBLIC      *USE




                                                                      Bottom
F3=Exit   F5=Refresh   F6=Add new users   F10=Grant with reference object
F11=Display detail     F12=Cancel         F17=Top
```

3. The first step is to make the public authorized to *USE (only be able to read the file, but not update it). Enter *USE instead of *CHANGE on the *PUBLIC line.

4. The next step would be to add a user to be able to change the file. There are blank lines on the display for you to add a user name and his authorization. If you don't know another user's name you can enter QSYSOPR (the IBM-supplied name for the system operator) as a sample.

   Your user profile name also exists as a separate authorization. You have *ALL authority meaning you can do anything you want to the object.

5. To test the CL program function that checks authority, blank out the *ALL entry for your name. Because you are the owner, you will be able to add yourself back. Press the Enter key and confirm your changes. End the EDTOBJAUT command with F3.

6. Call the MLGMNUC program and select option 2 for the maintenance program. You should receive the error message stating that you are not authorized. This is what a user would see who was not specifically authorized to the file.

7. Try selection 1 to use the inquiry function. Because the public is authorized to *USE, you should be able to perform this function.
8. Press F3 to return to the command entry display.
9. To get your authorizations back to the way they were originally, re-issue the EDTOBJAUT command as:

   ```
   EDTOBJAUT OBJ(MLGMSTP) OBJTYPE(*FILE)
   ```

10. Enter your name and specify *ALL authority.
11. If you want the file to be authorized as it was to begin with, remove the entry for the user you authorized (such as QSYSOPR) and set the *PUBLIC line back to *CHANGE as follows:

```
                          Edit Object Authority

Object . . . . . :   MLGMSTP        Object type  . . . . :   *FILE
  Library  . . . . :     USERXX      Owner  . . . . . . . :   USERXX

Type changes to current authorities, press Enter.

  Object secured by authorization list  . . . . . . . . . .   *NONE

             Object
User         Authority
USERXX       *ALL
*PUBLIC      *CHANGE

                                                                      Bottom
F3=Exit   F5=Refresh   F6=Add new users   F11=Display detail   F12=Cancel
F17=Top   F18=Bottom   F24=More keys
Object authorities changed.
```

You can read more about security in *Security Concepts and Planning*.

Program Stack

The OS/400 program operates on the basis of a program stack with calls and returns. When you sign on, you normally have an initial display or menu.

If you call the MLGMNUC program, a second program is now in the stack of programs. When you select an option from the menu such as the maintenance program, another program would be in the stack. In fact, the maintenance option is a CL program which calls an RPG program. Figure 15-1 on page 15-9 shows how the program stack would look when running the maintenance function.

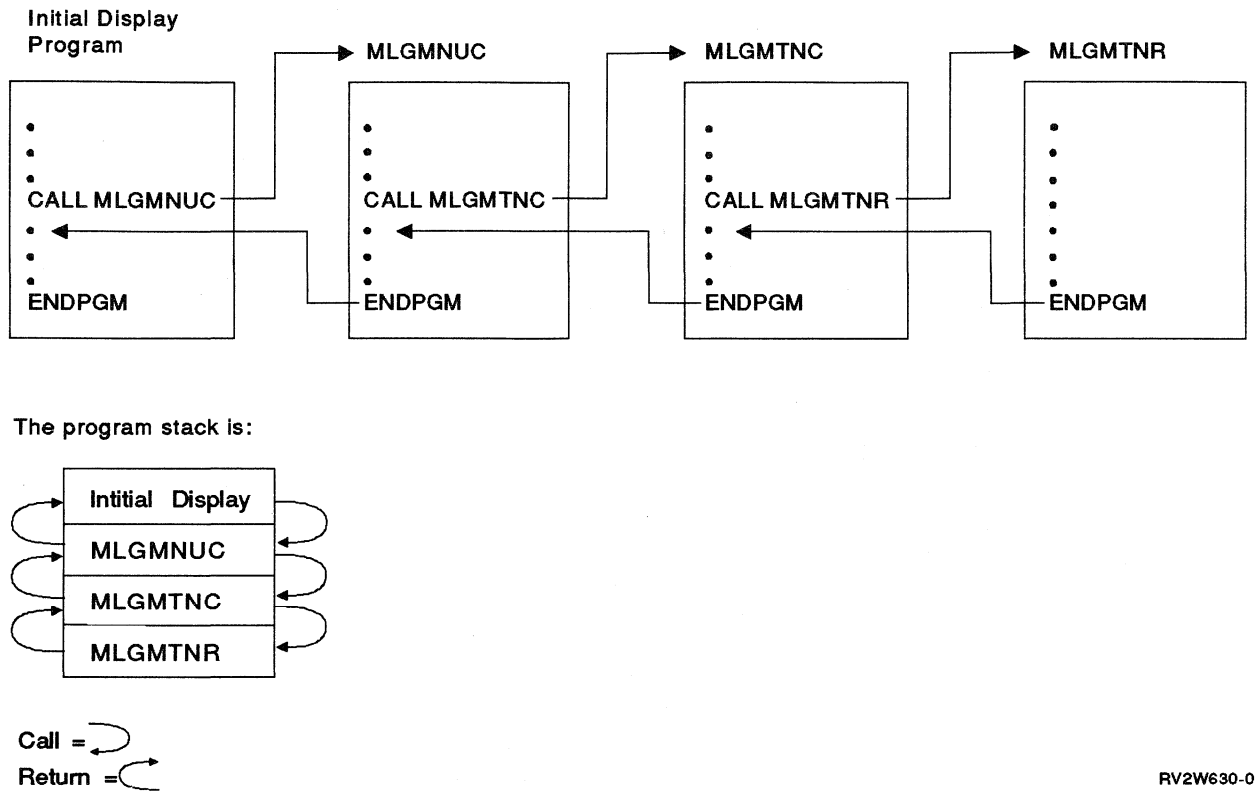


Figure 15-1. Program Stack for Maintenance Function

Only one program in the stack is active at a time. The system keeps track of where you are in each program. When the program that is active ends, a return occurs to the program that did the call. The calling program continues at the instruction following the call.

When a program is called, it always starts at the top of the program. When a program does a call, the return is always to the next instruction.

You can have several programs in the program stack without causing excess overhead on the system.

Displaying the Program Stack

You can display the program stack by using the DSPJOB command. Do the following to see the program stack while the maintenance program is active:

1. Type the following command:

   ```
   CALL MLGMNUC
   ```

2. Select option 2 (Maintain mailing list master) for the maintenance program.
3. Press the System Request key. When you see the line at the very bottom of the display, press the Enter key. You will see the System Request Menu.
4. Select option 3 (Display Current Job) on the System Request Menu to display your job.
5. Press the Enter key.
6. Select option 11 on the Display Job display to display the program stack. In the program stack, you will see some IBM-supplied programs along with your programs (MLGMNUC, MLGMTNC, MLGMTNR). The QWSGET program

is there because a program in the stack has performed a read to a work station and is waiting for input.

7. Return to the original display by pressing F3 until you get to the original display.

Additional PDM Functions

PDM supports several advanced functions that can help make you more productive. The following are a few of the items. For more information, you can begin with the Help key to read the online help information or see the *PDM User's Guide and Reference*.

The Work with Members Display

The PDM Work with Members display has many different options at the top of the display that allow you to do typical programmer functions. In this manual, we used only options 2 (Edit) and 14 (Compile). There are other functions supported such as copy and rename. The following steps go through some of these functions that have not been used previously in this manual.

1. Go to the Work with Members Display by typing STRPDM on the command entry line; then select option 3 (Work with members) and press the Enter key.
2. On the Specify Members to Work With display, type MLGSRC for the file name and your library name and press the Enter key.

   The Work with Members display is shown.

```
                        Work with Members Using PDM

File . . . . . .   MLGSRC
  Library . . . .  USERXX                Position to  . . . . .

Type options, press Enter.
  2=Edit         3=Copy       4=Delete       5=Display      6=Print
  7=Rename       8=Display description       9=Save         13=Change text ...

Opt  Member      Type        Text
     MLGINQD     DSPF        Mailing list inquiry display
     MLGINQD2    DSPF
     MLGINQR     RPG         Mailing list inquiry
     MLGLBLR     RPG         Mailing list label printing
     MLGMNUC     CLP         Mailing list menu program
     MLGMNUC2    CLP         Mailing menu to work with SDA created DSP file
8    MLGMNUD     DSPF        Mailing list menu
     MLGMSTL     LF          Mailing list label printing logical
                                                                        More...
Parameters or command
===>
F3=Exit          F4=Prompt             F5=Refresh            F6=Create
F9=Retrieve      F10=Command entry     F23=More options      F24=More keys
```

   Type an 8 (Display Description) in the *Opt* column next to the MLGMNUD member.

   This describes heading information about the member such as the last date and time that the member was changed.

```
                        Display Member Description

Member  . . . . . :   MLGMNUD
File  . . . . . :   MLGSRC
  Library . . . . . :     USERXX
Member type . . . . . :   DSPF

Creation date . . . . . :   01/06/90
Creation time . . . . . :   17:50:13
Change date . . . . . :   01/04/90
Change time . . . . . :   07:49:19

Save date . . . . . :   05/03/89
Save time . . . . . :   12:11:32
Restore date. . . . . :   05/08/89
Restore time. . . . . :   13:49:50

Number of records . . . :   33
Deleted records . . . . :    0

Text  . . . . . :   Mailing list menu

F3=Exit         F12=Cancel
```

3. Press the Enter key.

```
                          Work with Members Using PDM

File  . . . . . .   MLGSRC
  Library . . . .   USERXX                Position to  . . . . .

Type options, press Enter.
  2=Edit         3=Copy       4=Delete       5=Display      6=Print
  7=Rename       8=Display description       9=Save         13=Change text ...

Opt  Member      Type        Text
     MLGINQD     DSPF        Mailing list inquiry display
     MLGINQD2    DSPF
     MLGINQR     RPG         Mailing list inquiry
     MLGLBLR     RPG         Mailing list label printing
     MLGMNUC     CLP         Mailing list menu program
     MLGMNUC2    CLP         Mailing menu to work with SDA created DSP file
     MLGMNUD     DSPF        Mailing list menu
     MLGMSTL     LF          Mailing list label printing logical
                                                                        More...
Parameters or command
===>
F3=Exit          F4=Prompt             F5=Refresh            F6=Create
F9=Retrieve      F10=Command entry     F23=More options      F24=More keys
```

4. Press F23 to see the additional options.

```
                          Work with Members Using PDM

File  . . . . . .   MLGSRC
  Library . . . .   USERXX                  Position to  . . . . .

Type options, press Enter.
  14=Compile     17=Change using SDA     25=Find string ...
```

The command keys at the bottom of the display let you perform several other functions. Press F24 to see more keys and again to see the final set of keys.

```
                                                                More...
Parameters or command
===>
F11=Display names and types              F12=Cancel         F13=Repeat
F14=Display date        F15=Sort date    F23=More options   F24=More keys
```

```
                                                                More...
Parameters or command
===>
F16=User options    F17=Subset     F18=Change defaults     F21=Print list
                                   F23=More options        F24=More keys
```

5. F11 can be used to display a larger list of members on a single display.

   Press F11.

```
                        Work with Members Using PDM

File  . . . . . .   MLGSRC
  Library . . . .   USERXX                Position to  . . . . .

Type options, press Enter.
  14=Compile     17=Change using SDA     25=Find string ...


Opt Member     Type       Opt Member     Type       Opt Member     Type
    MLGINQD    DSPF           MLGMSTL2   LF             MLGNAMR    RPG
    MLGINQD2   DSPF           MLGMSTL3   LF             MLGREFP    PF
    MLGINQR    RPG            MLGMSTP    PF             MLGRPTC    CLP
    MLGLBLR    RPG            MLGMTNC    CLP            MLGRPTC2   CLP
    MLGMNUC    CLP            MLGMTND    DSPF           MLGRPTR    RPG
    MLGMNUC2   CLP            MLGMTNR    RPG
    MLGMNUD    DSPF           MLGNAMD    DSPF
    MLGMSTL    LF             MLGNAML    LF
                                                                  Bottom
Parameters or command
===>
F11=Display text                          F12=Cancel         F13=Repeat
F14=Display date        F15=Sort date     F23=More options   F24=More keys
```

6. Notice that the text description is dropped, but you can work with more members without using the key to page down.

   Press F11 to redisplay the original Work with Members display.

7. Press F3 (Exit).

User-Defined Options

One of the powerful functions provided by PDM lets you set your own options that can be entered next to any library, object, or member. The option you enter can perform any desired function. For example, you could create a general option for displaying your own output queue (assume you use option Q). By entering a Q next to any member, you could run the WRKOUTQ xxx command.

PDM has already provided several options for you that were not used in this manual. To see the options available, press F16 from any PDM list display, or select option 9 from the PDM menu. Table 15-1 describes some of the user-defined options.

Note: If you do not see this display, you must use F18 to request the Change Defaults display. The following should be entered:

```
Option file:   QAUOOPT
Library:       QGPL
Member:        QAUOOPT
```

Table 15-1. Sample User-Defined Options

Option Name	Command Called	Function
C	CALL &O/&N	Allows you to run a program on the Work with Members Using PDM display.
CS	STRSDA OPTION(1)SRCFILE(&L/&F) ??SRCMBR()	Allows you to create a member (display) using SDA.
CM	STRSDA OPTION(2)SRCFILE(&L/&F) ??SRCMBR()	Allows you to create a member (menu) using SDA.
CD	STRDFU OPTION(2)	Allows you to create a DFU program.
DM	DSPMSG	Allows you to display messages.
EA	EDTOBJAUT OBJ(&L/&N) OBJTYPE(&T)	Allows you to edit the authority to an object on the Work with Objects Using PDM display.
GO	GO &L/&N	Allows you to display the menu for a menu object.
JL	DSPJOBLOG	Allows you to display the job log.
SP	WRKSPLF	Allows you to work with spooled files.
WS	WRKSBMJOB	Allows you to work with jobs submitted to batch.

The option DM will provide you with an abbreviated method of specifying DSPMSG. Press Enter and return to the Work with Members display. Now enter DM on any member of the Option column.

You should see the Display Messages display.

Note: The DM entry (Display Message) can be entered next to any member name. The function requested has no meaning relative to a specific member.

PDM supplies the SP option for the WRKSPLF command. The WRKSPLF command was not used in this manual to display spooled files. Instead, the WRKOUTQ command was used. Many programmers prefer WRKSPLF that has the capability to display all of your spooled output regardless of which output queue it is on. You can try the WRKSPLF function by entering SP on any member of the Option column.

You should see the WRKSPLF display.

Selecting and Sequencing Solutions

The OS/400 licensed program provides a variety of techniques to allow selection and sequencing of database records. The coding examples shown in the manual all used DDS for logical files.

When a keyed database file is created, the system defaults to immediately maintain the access path. This means that any time a record is added or deleted or a key field is changed, the system will automatically change the access path to reflect the current status.

If you are using the access path for an interactive function such as a name search, you would normally want the system to perform immediate maintenance. If, however, you need an access path for a seldom used function such as report writing, it is normally a better performance tradeoff to not immediately maintain the access path.

There are several choices for selecting and sequencing and each has its advantages in certain situations.

Rebuild Maintenance

The CRTLF and CRTPF commands support the maintenance (MAINT) parameter that defaults to *IMMED to cause immediate maintenance. You could specify MAINT(*REBLD) to cause the access path to be built each time a program performs an open to the file. When the program closes a rebuild type file, the system deletes the access path (the file description still remains).

Sort (Format Data Command)

A sort function exists that allows selecting and sequencing criteria. It is typically used for batch applications. The full record sort will physically rearrange the records. While this is generally slower than building an access path, the total time of sorting and processing the file with a high-level language program can be a better overall performance solution. The reason for this is that the system performs very efficiently when accessing data in arrival sequence as opposed to a keyed sequence.

System efficiency with the sort and arrival sequence processing should only be considered when the number of records to be sorted is in the thousands. It is not how many records exist in the file, but how many your selection criteria selects for sorting. The sort function is run by using a command (FMTDTA) and providing source specifications (normally entered by SEU).

Open Query File (OPNQRYF) Command

The OPNQRYF command provides a powerful selecting and sequencing function that is specified by a single command. The function is normally used to precede a high-level language program. The OPNQRYF command supports such functions as:

- Selecting
- Sequencing (an access path is built or an existing access path is shared)
- Joining records
- Group processing
- Selecting after grouping
- Standard deviation

The OPNQRYF command provides a significant level of function that can be used in a wide variety of applications.

Structured Query Language/400 (SQL)

SQL provides both an interactive and program solution for database handling. In addition, files can be defined and maintained with SQL statements. SQL supports powerful selecting and sequencing functions that can be used to display the data. The same type of statements are supported inside several high-level languages to provide access to database records.

Backup and Recovery

A section in Chapter 6, "Adding Records to the Database File Using DFU" discussed the basic approaches used for backup and recovery. The system supports several additional techniques to improve the backup and recovery capability.

For example, journaling allows the capturing of any record changes and ensures that no changed records would be lost in main storage (not written to disk) if an abnormal system ending occurred. Each journal entry contains the complete new record that is called the *after image*. A *before image* option is also available. A journal can also be written to a user ASP (auxiliary storage pool) to avoid the loss of any database changes in case a single device failure occurs.

Journaling can also be used to help minimize what needs to be backed up offline to ensure an adequate daily backup. By saving only the journal changes instead of the entire file, a faster backup is normally achieved. For an application such as maintaining a mailing list, journaling can be very helpful because the percentage of records changed is usually very low on a daily basis.

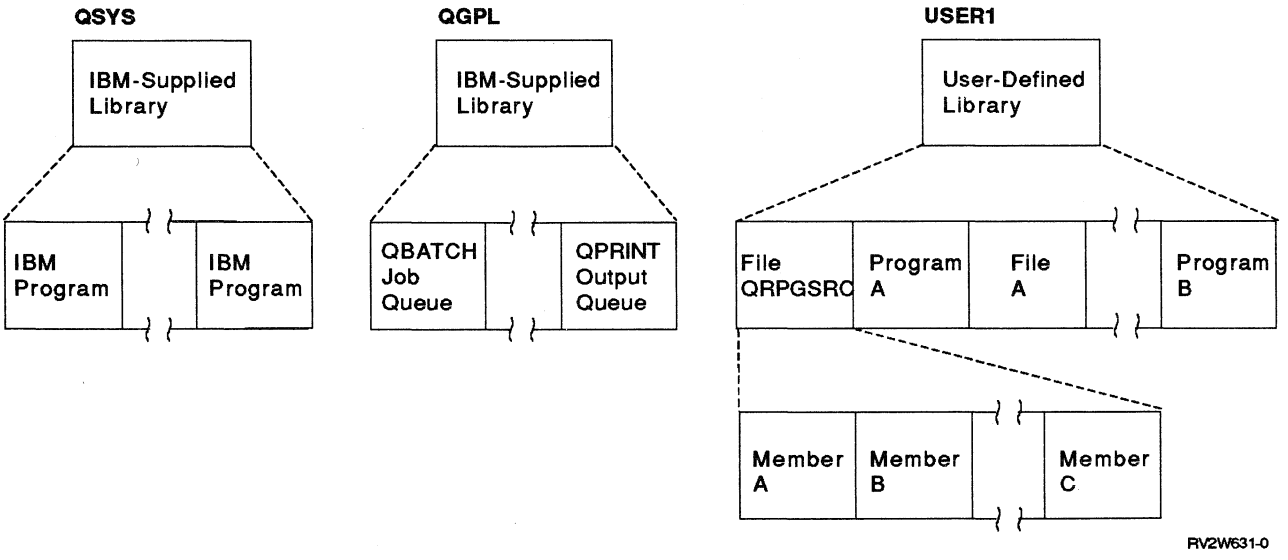
The system also supports an uninterruptible power supply approach to minimize the exposure of a power loss.

Commitment control is also supported to ensure that a transaction boundary is achieved when a failure occurs. Commitment control is useful when the application performs multiple additions or updates per transaction.

For a further discussion of backup and recovery, see the *Backup and Recovery Guide*.

Appendix A. Overview of Libraries and Library Lists

IBM supplies several libraries and you can use these along with libraries that you create. An example is shown below.



The QSYS library contains IBM-supplied objects needed for the operation of the OS/400 program. The QGPL library contains IBM-supplied objects, such as the QBATCH job queue, to help you use the system, as well as objects created by the system users.

In the illustration above, there are two objects in the library USER1 named A. These names are valid because one object is a program and the other is a file. The source file QRPGSRC contains many members. A member is not an object but does represent an important item in a system. Member names must be unique within a file.

The following sections define library processing.

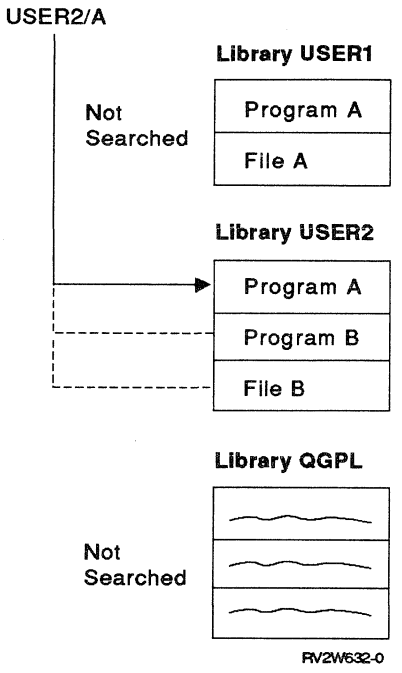
Qualified Names and the Library List

You can request an object on the system by stating a qualified name or by allowing the job's library list to be used.

A name is said to be *qualified* when both the library name and the name of the object named are used, such as:

```
USER2/A
```

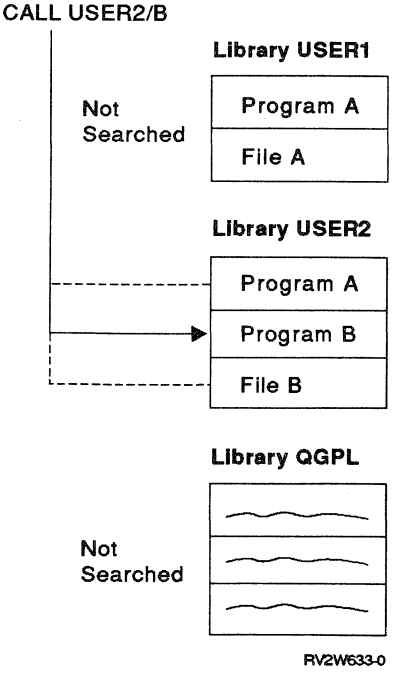
A slash must separate the two names. The library (USER2) is specified first followed by the object (A). If you use a qualified name when you request an object, only the library you specified is searched for the object. No other library is searched.



If a library contains two different object types having the same name, you must specify the object type or the object type must be implied in your request. For example, if you specify:

```
CALL USER2/B
```

as in the following illustration, object types other than programs are ignored in the search of the library USER2, because the CALL command assumes that only a program object type is to be found. The CALL command is always followed by a program name, which is B in this example. The *file* named B is ignored.



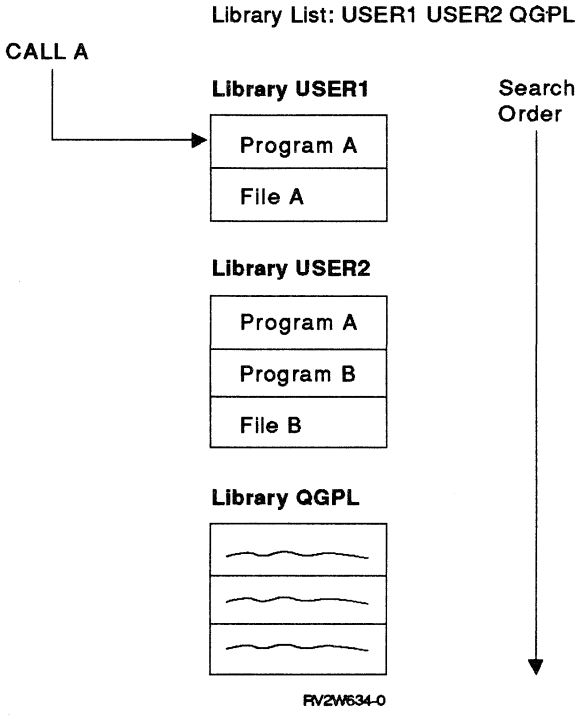
If a qualified name is not given when an object is requested, the system searches for the object by using the *library list* that is associated with the job in

which the request was made. A library list is an ordered list of library names indicating which libraries are to be searched, and the order in which they are to be searched, to find an object. In commands and on displays, the library list is indicated by *LIBL.

To use the library list to call program A in USER1, specify:

```
CALL A
```

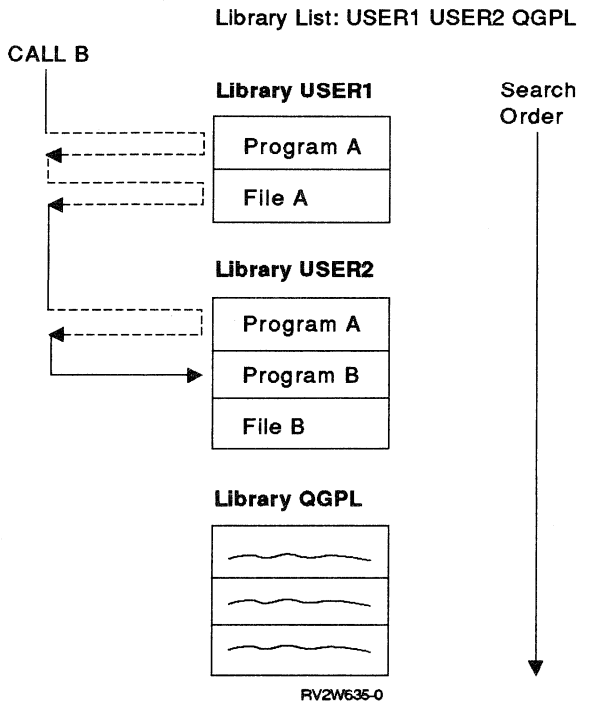
The first library, USER1, is searched for a program named A. Because program A is found in the first library, program A in USER2 is ignored.



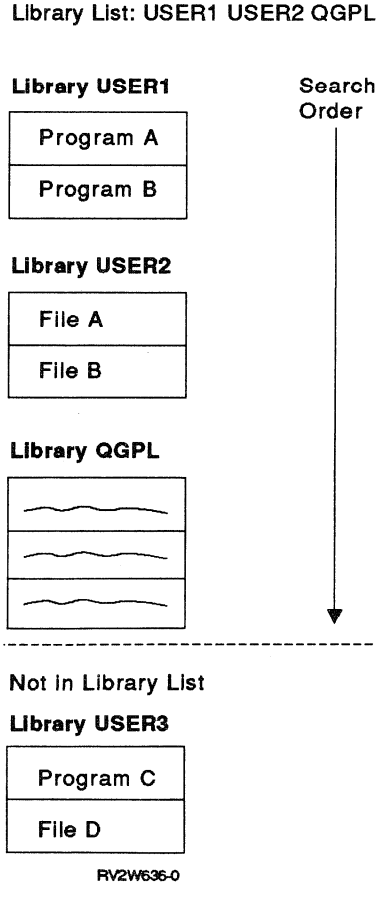
If you specify:

```
CALL B
```

the search starts in the first library, USER1. Because no program named B is found, the second library, USER2, is searched and the program is found.



The following illustration shows four sample libraries.



Assume the libraries USER1, USER2, and QGPL are in the library list and the library USER3 is not. If you specify:

```
CALL USER3/C
```

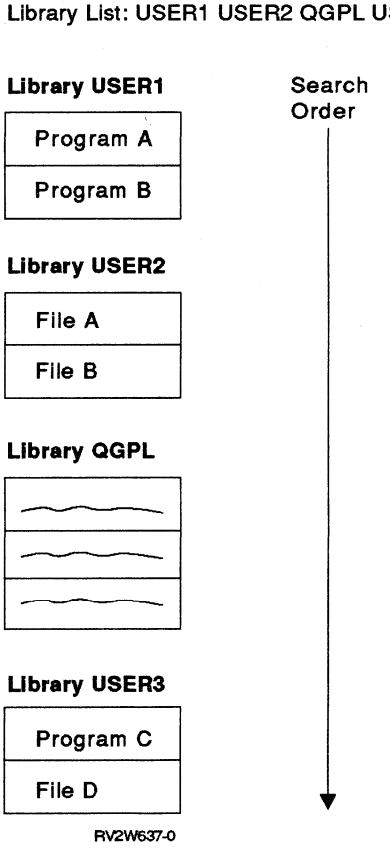
the program C is found in USER3 because USER3/C is a qualified name.

If program C needs to use file D and file D is not specified with a qualified name, the library list is used to find file D. The search is not successful because file D is in USER3, which is not in the library list.

The RPG/400 licensed program does not allow qualified names to be specified in a program. An override command can be used to specify a qualified name for some objects, but an easier approach is to specify all libraries that you will be using in the library list. If the library USER3 is added to the library list, as shown below, you can specify:

```
CALL C
```

and both program C and file D will be found.



The Library List in Applications

Each job in the system has its own library list. This library list consists of a system part and a user part, as in Figure A-1.

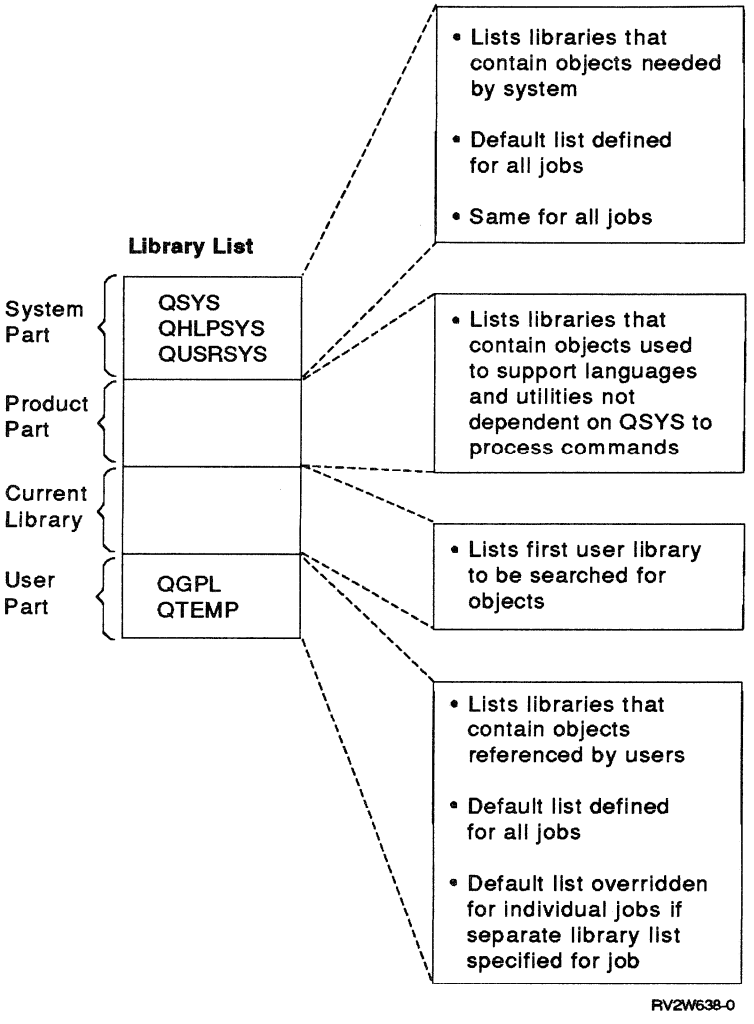


Figure A-1. Overview of the Library List

Figure A-1 shows the library list as shipped by IBM. The system part contains the QSYS, QHLPSYS, and QUSRSYS libraries; the user part contains only the QGPL and QTEMP libraries.

The product part can be set by a command when it is run. For example, when the CRTRPGPGM command is run, the QRPG library is placed in the product part. This allows the command to find the RPG compiler objects that are located in the QRPG library.

The current library was described earlier in the manual. It also acts as a default for created objects. If you do not have a current library, objects are created in QGPL by default.

Generally, only the user part is apparent to the system users because it affects objects they request in individual jobs.

You can determine which libraries are contained in the user part of the library list for a job by issuing the Display Library List (DSPLIBL) command in the job. The display that is shown after entering the DSPLIBL command would look similar to the following display:

```
                          Display Library List
                                                        System:   RCH38342

Type option, press Enter.
  5=Display objects in library

Opt  Library     Type      Text
     QSYS        SYS       System Library
     QHLPSYS     SYS
     QUSRSYS     SYS       *IN USE
     USERXX      CUR       General Purpose Library
     QTEMP       USR

                                                                      Bottom
F3=Exit   F12=Cancel   F17=Top   F18=Bottom
                                        (C) COPYRIGHT IBM CORP. 1980, 1989.
```

This sample display shows that the library list being used for the job is the default list. For many applications, this default library list may not be sufficient. If a reference is made to an object without specifying its library and the library is not in the job library list, the system will not be able to find the object. This will result in an error in the job.

Appendix B. Overview of QUSRTOOL Library

The purpose of this library is to provide you access to examples of various tools and programming techniques that may help you with application development and management of your system. These are not integrated system functions, are not considered part of the OS/400 licensed program, and may change from release to release.

The example tools are contained within the following source files in library QUSRTOOL:

- QATTINFO: contains documentation of the various tools
- QATTCL: contains source for CL programs
- QATTCMD: contains source for command definitions
- QATTDDS: contains source for DDS file definitions
- QATTPL1: contains source for PL1 programs
- QATTRPG: contains source for RPG programs
- QATTPAS: contains source for PASCAL programs

These files are owned by QPGMR. The library QUSRTOOL is owned by QSYS.

The QUSRTOOL library is optionally installable as *Option 7* of the base operating system. The source files will be completely overlaid each release. Therefore, new example tools may be added and existing ones may be changed or deleted. QUSRTOOL should contain no user objects other than these source files.

Each tool has a two-character identifier and a primary documentation member within file QATTINFO. The documentation may be viewed using the DSPPFM command, CPYF to a printer, or using the SEU display function. The documentation will tell you what to do to install the tool in a user library. Only source is shipped so the objects must be created. Some tools have install commands to assist in the creation.

Besides the documentation member in QATTINFO and the source for the install member in QATTCL, each tool consists of one or more members within the source files in QUSRTOOL. These members contain the source for programs, files, commands, and so on which show you how a particular function may be done. You can copy this example source to a user library and change it to suit your particular needs.

How to Use

You should not create any user objects in library QUSRTOOL because it will be overlaid each release. Similarly, you should make any changes to this source in a separate user library.

No command help is provided. Refer to the appropriate documentation member of QATTINFO for a description of the command and command parameters.

These example programs are not to be construed as normal operating system type function. Therefore, aspects of these examples (such as error handling, recovery, and performance) may not be equivalent to what you would expect if these were operating system functions.

When run, the tools install program will create and load all objects needed by the tool. Any exceptions to this (for example, other things that you must manually do) will be noted in the documentation member for that tool. Security considerations are also described in the tool's documentation member.

Example of Copying Source from QUSRTOOL Library

The source used in the files and programs throughout this manual is contained in the QUSRTOOL library in three different source files based on source type. The following table shows where the source is located for the different files and programs.

SOURCE LOCATED IN QUSRTOOL LIBRARY

QUSRTOOL Source File	Member	Type	Chapter Used In	Text
QATTDDS	MLGREFP	PF	4	Mailing list field reference file
QATTDDS	MLGMSTP	PF	5	Mailing master physical file
QATTDDS	MLGMSTL	LF	5	Mailing list label printing logical
QATTDDS	MLGMSTL2	LF	8	Mailing list by state, city, name
QATTDDS	MLGMSTL3	LF	8	General purpose LF for querying MLGMSTP
QATTDDS	MLGINQD	DSPF	9	Mailing list inquiry display
QATTDDS	MLGMTND	DSPF	9	Mailing maintenance display file
QATTDDS	MLGMNUD	DSPF	10	Mailing list menu display file
QATTDDS	MLGNAMD	DSPF	11	Mailing list name search display file
QATTDDS	MLGNAML	LF	11	Mailing list logical file by name, state, city
QATTCL	MLGRPTC	CLP	8	Print one liner with MLGMSTL2 LF
QATTCL	MLGRPTC2	CLP	8	General purpose query of MLGMSTP
QATTCL	MLGMTNC	CLP	9	Mailing list maintenance
QATTCL	MLGMNUC	CLP	10	Mailing list menu program
QATTRPG	MLGLBLR	RPG	6	Mailing list label printing
QATTRPG	MLGRPTR	RPG	8	Mailing list one line report per name
QATTRPG	MLGINQR	RPG	9	Mailing list inquiry
QATTRPG	MLGMTNR	RPG	9	Mailing list master maintenance
QATTRPG	MLGNAMR	RPG	11	Mailing list name search

There are basically two different ways that you can copy the source from the QUSRTOOL source file:

- You can copy one member at a time using the browse/copy services function of SEU.
- You can copy all members from a QUSRTOOL source file (for example, all CL source members contained in QATTCL) at one time using the Copy Source File (CPYSRCF) command.

Examples of using both of these approaches are shown. You should determine one approach to use.

Note: If you choose to copy the source from the QUSRTOOL library, you still must create the files or compile the programs using the steps described in each chapter.

Example of Using SEU to Copy One Member

To copy the source for the mailing list field reference file (MLGREFP) from the QUSRTOOL library, do the following:

1. Type STRPDM on the command entry line.
2. Select option 3 (Work with Members)

```
                    Specify Members to Work With

Type choices, press Enter.

  File . . . . . .  MLGSRC       Name

    Library . . . . . .  USERXX       *LIBL, *CURLIB, name

  Member:
    Name . . . . . .  *ALL         *ALL, name, *generic*
    Type . . . . . .  *ALL         *ALL, *BLANK, type, *generic*









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

3. Fill in the file and library names on the Specify Members to Work With display. In this example, the MLGSRC source file and library USERXX are used. Press the Enter key.

   The Work with Members Using PDM display is shown.

```
                        Work with Members Using PDM

File . . . . . .  MLGSRC
  Library . . . .  USERXX                Position to . . . . .

Type options, press Enter.
  2=Edit         3=Copy       4=Delete       5=Display      6=Print
  7=Rename       8=Display description       9=Save         13=Change text . . .

Opt  Member     Type        Text

  (No members in file)









Parameters or command
===>
F3=Exit          F4=Prompt            F5=Refresh           F6=Create
F9=Retrieve      F10=Command entry    F23=More options     F24=More keys
```

4. To create a new member, press F6.

   The Start Source Entry Utility display is shown.

```
                    Start Source Entry Utility (STRSEU)

Type choices, press Enter.

Source file  . . . . . > MLGSRC        Name, *PRV
  Library  . . . . . > USERXX        Name, *LIBL, *CURLIB, *PRV
Source member  . . . . .   MLGREFP      Name, *PRV, *SELECT
Source type  . . . . . .   PF           Name, *SAME, BAS, BASP...
Text 'description' . . . . .   Master list field reference file

                                                                  Bottom
F3=Exit   F4=List   F5=Refresh   F11=Keywords   F12=Previous
F13=How to use this display
```

5. Type the name of the source member you want to create, PF (physical file) for source type, and type the description of the member. Press the Enter key.

   The SEU Edit display is shown. You will get a message that your member was added to the library you specified.

```
Columns . . . .:   1  71              Edit                    USERXX/MLGSRC
Find  . . .                                                       MLGREFP
FMT PF .....A.........T.Name++++++RLen++TDpB......Functions++++++++++++++++++
       *************** Beginning of data ************************************
'''''''
'''''''
'''''''
'''''''
'''''''
'''''''
'''''''
'''''''
'''''''
'''''''
'''''''
'''''''
'''''''
'''''''
'''''''
       ****************** End of data ***************************************

F3=Exit                  F4=Prompt                F5=Refresh
F10=Top                  F11=Bottom               F24=More keys
Member MLGREFP added to file USERXX/MLGSRC.                               +
```

6. Press F24 to display more keys. Then press F15 to use the browse/copy services function.

```
                        Browse/Copy Services

Type choices, press Enter.

  Selection . . . . . .  1             1=Member
                                       2=Spool file
                                       3=Output queue
  Copy all records  . . . . . .  Y     Y=Yes, N=No
  Browse/copy member  . . . . . .  MLGREFP     Name, F4 for list
    File  . . . . . .  . . . . .  QATTDDS      Name
      Library . . . . . .  . . .  QUSRTOOL     Name, *CURLIB, *LIBL

  Browse/copy spool file  . . . .  MLGREFP     Name
    Job . . . . . .  . . . . . .  MLGREFP     Name
      User  . . . . . .  . . . .  USERXX       Name
      Job number  . . . . . .  .  *LAST        Number, *LAST
    Spool number  . . . . . .  .  *LAST        Number, *LAST, *ONLY

  Display output queue  . . . . .  QPRINT     Name, *ALL
    Library . . . . . .  . . . .  *LIBL       Name, *CURLIB, *LIBL

F3=Exit          F4=Prompt         F5=Refresh       F12=Cancel
F13=Change defaults                F14=Find/Change Services
```

7. Type the name of the file you want to copy source from. Select Y for the *Copy all records* field. In this example we are copying the DDS source from member MLGREFP in file QATTDDS in library QUSRTOOL. Press the Enter key.

   The next display shows the actual source contained in MLGREFP.

```
 Columns . . . .:   1  71              Edit                    USERXX/MLGSRC
 Find  . . .                                                        MLGREFP
 FMT PF .....A.........T.Name++++++RLen++TDpB......Functions++++++++++++++++++
A       *************** Beginning of data ***************************************

 Columns . . . .:   1  71             Browse       Pending . . . . . .:  CC
 Find  . . .
       *************** Beginning of data ***************************************
0001.00      A* MLGREFP - Mailing list field reference file
0002.00      A          R MLGREFR                    TEXT('Mailing list ref')
0003.00      A            MLACCT         5  0       COLHDG('Account' +
0004.00      A                                       'number')
0005.00      A                                       EDTCDE(X)
0006.00      A            MLTYPE         1          COLHDG('Type')
0007.00      A                                       VALUES('1' '2' '3' +
0008.00      A                                       '4' '5' '9')
0009.00      A                                       TEXT('Acct type- +
0010.00      A                                       1=Bus 2=Gov 3=Org +
0011.00      A                                       4=Sch 5= Pvt 9=Oth')
0012.00      A            MLNAME        20          COLHDG('Name')

 F3=Exit                  F5=Refresh               F6=Move split line
 F12=Cancel               F24=More Keys
Specify a target for the include.
```

8. To copy all of the records into your new member, type an A (after) for your target at the cursor position and press the Enter key.

9. The records will be copied into the new member MLGREFP. Press F3 (Exit).

   The SEU Exit display is shown.

```
                                    Exit

Type choices, press Enter.

  Change/create member . . . . . .  Y           Y=Yes, N=No
    Member . . . . . . . . . . . .  MLGREFP     Name
    File . . . . . . . . . . . . .  MLGSRC      Name
      Library . . . . . . . . . .   USERXX      Name
    Text . . . . . . . . . . . . .  Master list field reference

    Resequence member . . . . . .  Y           Y=Yes, N=No
      Start . . . . . . . . . . .  0001.00     0000.01 - 9999.99
      Increment . . . . . . . . .  01.00       00.01 - 99.99

  Print member . . . . . . . . .   N           Y=Yes, N=No

  Return to editing . . . . . .    N           Y=Yes, N=No

  Go to member list . . . . . .    N           Y=Yes, N=No


F3=Exit       F5=Refresh       F12=Previous
```

10. You can press the Enter key to take the defaults and create the new member MLGREFP.

    You should now be back to the Work with Members Using PDM display. Your newly created member should be shown.

11. Press F23 to display more options.

```
                          Work with Members Using PDM

File . . . . . .   MLGSRC
  Library . . . .  USERXX                Position to  . . . . .

Type options, press Enter.
  14=Compile     17=Change using SDA     25=Find string


Opt  Member      Type        Text
 14  MLGREFP     PF          Master list field reference









                                                                      Bottom
Parameters or command
===>
F3=Exit          F4=Prompt            F5=Refresh           F6=Create
F9=Retrieve      F10=Command entry    F23=More options     F24=More keys
Member MLGREFP added to file USERXX/MLGSRC.                            +
```

12. You can now compile the member by typing a 14 in the *Option* column next to member MLGREFP.

13. Press F3 (Exit).

Example of Using CPYSRCF Command to Copy Multiple Members

If you choose to use this approach, you need to run the CPYSRCF command three times to copy the source from QATTDDS, QATTCL, and QATTRPG.

1. This example uses the CPYSRCF command to copy all of the source members from QATTDDS into the MLGSRC source file in library USERXX. To do this, type the following command and press F4 to prompt:

   CPYSRCF

```
                          Copy Source File (CPYSRCF)

Type choices, press Enter.

Data base source file . . . . .   QATTDDS       Name
  Library . . . . . . . . . . .     QUSRTOOL    Name, *LIBL, *CURLIB
To file . . . . . . . . . . . .   MLGSRC        Name, *PRINT
  Library . . . . . . . . . . .     USERXX      Name, *LIBL, *CURLIB
From member . . . . . . . . . .   MLG*          Name, generic*, *FIRST, *ALL
To member or label . . . . . .    *FROMMBR      Name, *FROMMBR, *FIRST
Replace or add records . . . . .  *REPLACE     *REPLACE, *ADD
Source update options . . . . .   *SAME         *SAME, *SEQNBR, *DATE

                                                                      Bottom
F3=Exit   F4=Prompt   F5=Refresh   F12=Cancel   F13=How to use this display
```

2. The Copy Source File display is shown. Here is where you specify that you want to copy all members that start with MLG from QATTDDS in QUSRTOOL library to the MLGSRC file in USERXX library. Do this by filling in the prompt as shown and pressing the Enter key. (The entry of MLG* means to copy all members that begin with MLG.)

3. Press the Enter key again to run the command.

```
                AS/400 Programming Development Manager (PDM)

Select one of the following:

     1. Work with libraries
     2. Work with objects
     3. Work with members

     9. Work with user-defined options




Selection or command
===> CPYSRCF FROMFILE(QUSRTOOL/QATTDDS) TOFILE(USERXX/MLGSRC) FROMMBR(MLG*)

F3=Exit       F4=Prompt       F9=Retrieve       F10=Command entry
F12=Cancel    F18=Change defaults
```

The source should be copied. You can use option 3 in PDM to work with members and display the members copied.

4. To copy the remaining members, you should run the CPYSRCF command again using QATTCL and QATTRPG in place of QATTDDS.

Appendix C. Example of Using SEU Functions

The following example uses SEU through PDM. It describes some of the basic functions.

Go through the following steps to enter a new RPG source member named TEST in file MLGSRC and in your library.

1. Enter the STRPDM command and select option 3 (Work with members) on the AS/400 Programming Development Manager (PDM) display.

```
                    Specify Members to Work With

Type choices, press Enter.

  File . . . . . .   MLGSRC      Name

    Library . . . . . .   USERXX       *LIBL, *CURLIB, name

  Member:
    Name . . . . . .   *ALL         *ALL, name, *generic*
    Type . . . . . .   *ALL         *ALL, *BLANK, type, *generic*









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

2. Fill in the file and library information as shown above.

```
                        Work with Members Using PDM

File . . . . . .   MLGSRC
  Library . . . .  USERXX                Position to . . . . .

Type options, press Enter.
  2=Edit         3=Copy       4=Delete       5=Display       6=Print
  7=Rename       8=Display description       9=Save          13=Change text

Opt  Member      Type        Text

  (No members match the subsetting criteria)









Parameters or command
===>
F3=Exit          F4=Prompt             F5=Refresh            F6=Create
F9=Retrieve      F10=Command entry     F23=More options      F24=More keys
```

3. Press F6 to create a new member.

```
                    Start Source Entry Utility (STRSEU)

Type choices, press Enter.

Source file . . . . . > MLGSRC        Name, *PRV
  Library . . . . . > USERXX        Name, *LIBL, *CURLIB, *PRV
Source member . . . . . . TEST          Name, *PRV, *SELECT
Source type . . . . . . . RPG           Name, *SAME, BAS, BASP, C...
Text 'description' . . . . . . Test for SEU

                                                                  Bottom
F3=Exit   F4=Prompt   F5=Refresh   F12=Cancel   F13=How to use this display
F24=More keys
```

4. On the Start Source Entry Utility display, fill in the source member, source type, and a description of the member as shown above. Press the Enter key.

   Because you are creating a new member, the Edit display appears with a screen of blank lines. The last line on the display, *Member TEST added to file MLGSRC* indicates that SEU added the new member to the file you specified.

```
Columns . . . .:   1  71              Edit                    USERXX/MLGSRC
Find  . . .                                                          TEST
FMT H  .....H........1..CDYI....S.............1.F.........................
       ********** Beginning of data ***************************************
'''''''
'''''''
'''''''
'''''''
'''''''
'''''''
'''''''
'''''''
'''''''
'''''''
'''''''
'''''''
'''''''
'''''''
'''''''
       ****************** End of data ***************************************

 F3=Exit                  F4=Prompt                F5=Refresh
 F10=Top                  F11=Bottom               F24=More keys
Member TEST added to file USERXX/MLGSRC.                                +
```

5. On this SEU edit display, you can enter the source statements for the new member.

   You specify how you want to enter the source statements by entering one of the SEU line commands in the sequence number area (columns 1 through 7).

   As you enter each source statement, it is given a sequence number. If you need to use an SEU line command again, you type it in over the sequence number.

   The next series of displays will use some of these line commands.

```
Columns . . . .:   1  71              Edit                    USERXX/MLGSRC
Find  . . .                                                          TEST
FMT H  .....H........1..CDYI....S.............1.F.........................
IPC    *************** Beginning of data ***************************************
'''''''
'''''''
'''''''
'''''''
'''''''
'''''''
'''''''
'''''''
'''''''
'''''''
'''''''
'''''''
'''''''
'''''''
       ****************** End of data *****************************************

F3=Exit                  F4=Prompt               F5=Refresh
F10=Top                  F11=Bottom              F24=More keys
```

6. To insert an RPG calculation source statement with a prompt, type IPC (insert, prompt, calculations) as shown above and press the Enter key:

   SEU inserts the C prompt at the bottom of the display. The fields of the calculation specification are separated and labeled as shown.

```
Columns . . . .:   1  71              Edit                    USERXX/MLGSRC
Find  . . .                                                          TEST
FMT H  .....H........1..CDYI....S.............1.F.........................
       *************** Beginning of data ***************************************
'''''''
       ****************** End of data *****************************************









Prompt type . . .    C      Sequence number . . .  '''''''

Level    N01N02N03  Factor 1     Operation      Factor 2     Result
                                    ADD          A            B
           Decimal
Length     Positions     H/A    HI   LO   EQ    Comment


F3=Exit                  F4=Prompt               F5=Refresh
F10=Top                  F11=Bottom              F24=More keys
```

7. Type ADD under the *Operation* column, A under the *Factor 2* column, and B under the *Result* column and press the Enter key.

   SEU creates a source record with the calculation specification and assigns sequence number 0001.00 to the record. SEU assumes that you want to enter another calculation statement.

```
Columns . . . .:   1  71              Edit                    USERXX/MLGSRC
Find  . . .                                                          TEST
FMT C  .....CL0N01N02N03Factor1+++OpcdeFactor2+++ResultLenDHHiLoEqComments++++
       ********** Beginning of data ************************************
0001.00      C                   ADD  A         B
'''''''
       ****************** End of data ***********************************




Prompt type . . .    C      Sequence number . . .  '''''''

Level    N01N02N03  Factor 1     Operation      Factor 2     Result
                                   MOVE          C            D
           Decimal
Length     Positions     H/A    HI   LO   EQ    Comment


F3=Exit                    F4=Prompt                F5=Refresh
F10=Top                    F11=Bottom               F24=More keys
```

8. Next, type MOVE under the *Operation* column, C under the *Factor 2* column, and D under the *Result* column and press the Enter key.

   SEU creates a source record with the calculation specification and assigns sequence number 0002.00 to the record.

```
Columns . . . .:   1  71              Edit                    USERXX/MLGSRC
Find  . . .                                                          TEST
FMT C  .....CL0N01N02N03Factor1+++OpcdeFactor2+++ResultLenDHHiLoEqComments++++
       ********** Beginning of data ************************************
0001.00      C                   ADD  A         B
0002.00      C                   MOVE C         D
'''''''
       ****************** End of data ***********************************




Prompt type . . .    C      Sequence number . . .  '''''''

Level    N01N02N03  Factor 1     Operation      Factor 2     Result
                                   ADD           B            C
           Decimal
Length     Positions     H/A    HI   LO   EQ    Comment


F3=Exit                    F4=Prompt                F5=Refresh
F10=Top                    F11=Bottom               F24=More keys
```

9. Next, type ADD under the *Operation* column, B under the *Factor 2* column, and C under the *Result* column and press the Enter key.

   SEU creates a source record with the calculation specification and assigns sequence number 0003.00 to the record.

```
Columns . . . .:   1  71              Edit                    USERXX/MLGSRC
Find  . . .                                                          TEST
FMT C  .....CL0N01N02N03Factor1+++OpcdeFactor2+++ResultLenDHHiLoEqComments++++
       ********** Beginning of data ************************************
0001.00      C                    ADD  A         B
0002.00      C                    MOVE C         D
0003.00      C                    ADD  B         C
       ****************** End of data ***************************************



Prompt type . . .    C      Sequence number . . .  '''''''

Level    N01N02N03  Factor 1     Operation      Factor 2     Result

           Decimal
Length     Positions     H/A    HI   LO   EQ    Comment


F3=Exit             F4=Prompt          F5=Refresh          F10=Top
F11=Bottom          F12=Cancel         F24=More keys
```

10. Press F12 (previous) to remove the prompt and blank line.

```
Columns . . . .:   1  71              Edit                    USERXX/MLGSRC
Find  . . .                                                          TEST
FMT C  .....CL0N01N02N03Factor1+++OpcdeFactor2+++ResultLenDHHiLoEqComments++++
       ********** Beginning of data ************************************
0001.00      C                    ADD  A         B
B002.00      C                    MOVE C         D
M003.00      C                    ADD  B         C
       ****************** End of data ***************************************









F3=Exit                  F4=Prompt                F5=Refresh
F10=Top                  F11=Bottom               F24=More keys
```

11. To move line 3 before line 2, type an M (move) in sequence number 0003.00 and a B (before) in sequence number 0002.00 and press the Enter key.

```
Columns . . . .:   1  71              Edit                    USERXX/MLGSRC
Find  . . .                                                        TEST
FMT C  .....CL0N01N02N03Factor1+++OpcdeFactor2+++ResultLenDHHiLoEqComments++++
       *************** Beginning of data ************************************
0001.00      C                   ADD  A         B
M001.01      C                   ADD  B         C
A002.00      C                   MOVE C         D
'''''''
       ****************** End of data ***************************************









F3=Exit                    F4=Prompt                  F5=Refresh
F10=Top                    F11=Bottom                 F24=More keys
```

12. To move line 0001.01 after line 0002.00, type M (move) on line 0001.01 and an A (after) on line 0002.00 and press the Enter key.

```
Columns . . . .:   1  71              Edit                    USERXX/MLGSRC
Find  . . .                                                        TEST
FMT C  .....CL0N01N02N03Factor1+++OpcdeFactor2+++ResultLenDHHiLoEqComments++++
       *************** Beginning of data ************************************
0001.00      C                   ADD  A         B
0002.00      C                   MOVE C         D
0003.00      C                   ADD  B         C
       ****************** End of data ***************************************









F3=Exit                    F4=Prompt                  F5=Refresh
F10=Top                    F11=Bottom                 F24=More keys
```

13. You can prompt for a calculation by putting your cursor on one of the calculation lines and pressing F4 (Prompt). Do this now by putting your cursor somewhere on line 002.00 and pressing F4.

    The prompt will be shown on the lower portion of the display and the calculation specifications you have specified will be filled in. You can now add or change your calculations.

```
Columns . . . .:   1  71              Edit                    USERXX/MLGSRC
Find  . . .                                                          TEST
FMT C  .....CL0N01N02N03Factor1+++OpcdeFactor2+++ResultLenDHHiLoEqComments++++
       ********** Beginning of data ************************************
0001.00      C                    ADD  A         B
0002.00      C                    MOVE C         D
0003.00      C                    ADD  B         C
       ****************** End of data ***************************************



Prompt type . . .    C      Sequence number . . .  0002.00

Level    N01N02N03  Factor 1     Operation      Factor 2     Result
                                    MOVE         C            D
           Decimal
Length     Positions     H/A    HI   LO   EQ    Comment

F3=Exit            F4=Prompt          F5=Refresh         F10=Top
F11=Bottom         F12=Cancel         F24=More keys
```

14. Prompt for line 0003.00 by moving your cursor to that line and pressing F4.

```
Columns . . . .:   1  71              Edit                    USERXX/MLGSRC
Find  . . .                                                          TEST
FMT C  .....CL0N01N02N03Factor1+++OpcdeFactor2+++ResultLenDHHiLoEqComments++++
       ********** Beginning of data ************************************
0001.00      C                    ADD  A         B
0002.00      C                    MOVE C         D
0003.00      C                    ADD  B         C
       ****************** End of data ***************************************



Prompt type . . .    C      Sequence number . . .  0003.00

Level    N01N02N03  Factor 1     Operation      Factor 2     Result
                                    ADD          B            C
           Decimal
Length     Positions     H/A    HI   LO   EQ    Comment

F3=Exit            F4=Prompt          F5=Refresh         F10=Top
F11=Bottom         F12=Cancel         F24=More keys
```

15. The prompt will change to the specifications in line 0003.00. Press the Enter key. Because 0003.00 is the last record, prompting is stopped.

```
Columns . . . .:   1  71              Edit                    USERXX/MLGSRC
Find  . . .                                                          TEST
FMT C  .....CL0N01N02N03Factor1+++OpcdeFactor2+++ResultLenDHHiLoEqComments++++
       ********** Beginning of data ************************************
P001.00      C                   ADD  A         B
0002.00      C                   MOVE C         D
0003.00      C                   ADD  B         C
       ****************** End of data ***********************************




F3=Exit                  F4=Prompt                F5=Refresh
F10=Top                  F11=Bottom               F24=More keys
```

16. Another way of prompting is to use the line command P. Type a P on line 0001.00 and press the Enter key.

```
Columns . . . .:   1  71              Edit                    USERXX/MLGSRC
Find  . . .                                                          TEST
FMT C  .....CL0N01N02N03Factor1+++OpcdeFactor2+++ResultLenDHHiLoEqComments++++
       ********** Beginning of data ************************************
0001.00      C                   ADD  A         B
0002.00      C                   MOVE C         D
0003.00      C                   ADD  B         C
       ****************** End of data ***********************************



Prompt type . . .    C      Sequence number . . .  0001.00

Level    N01N02N03  Factor 1     Operation      Factor 2     Result
                                    ADD          A            B
           Decimal
Length     Positions     H/A    HI   LO   EQ    Comment
   5           0

F3=Exit            F4=Prompt          F5=Refresh         F10=Top
F11=Bottom         F12=Cancel         F24=More keys
```

17. Type a 5 under the *Length* column and a 0 under the *Decimal Positions* column and press the Enter key.

```
Columns . . . .:   1  71              Edit                      USERXX/MLGSRC
Find  . . .                                                          TEST
FMT C  .....CL0N01N02N03Factor1+++OpcdeFactor2+++ResultLenDHHiLoEqComments++++
       *************** Beginning of data ************************************
0001.00      C                    ADD  A         B       50
0002.00      C                    MOVE C         D
D003.00      C                    ADD  B         C
       ****************** End of data ***************************************




Prompt type . . .    C      Sequence number . . .  0002.00

Level    N01N02N03  Factor 1     Operation      Factor 2     Result
                                    MOVE         C            D
           Decimal
Length     Positions     H/A    HI   LO   EQ    Comment


F3=Exit            F4=Prompt          F5=Refresh         F10=Top
F11=Bottom         F12=Cancel         F24=More keys
```

18. Next, delete line 0003.00 by typing a D on that line and pressing the Enter key. Because 0002.00 is the last record, prompting is stopped.

```
Columns . . . .:   1  71              Edit                      USERXX/MLGSRC
Find  . . .                                                          TEST
FMT C  .....CL0N01N02N03Factor1+++OpcdeFactor2+++ResultLenDHHiLoEqComments++++
       *************** Beginning of data ************************************
0001.00      C                    ADD  A         B       50
0002.00      C                    MOVE C         D
       ****************** End of data ***************************************









F3=Exit                  F4=Prompt                F5=Refresh
F10=Top                  F11=Bottom               F24=More keys
```

```
Columns . . . .:   1  71              Edit                    USERXX/MLGSRC
Find  . . .                                                          TEST
FMT C  .....CL0N01N02N03Factor1+++OpcdeFactor2+++ResultLenDHHiLoEqComments++++
       *************** Beginning of data ************************************
0001.00      C                   ADD  A         B       50
C002.00      C                   MOVE C         D
B       ****************** End of data ***************************************

F3=Exit                  F4=Prompt               F5=Refresh
F10=Top                  F11=Bottom              F24=More keys
```

19. Copy line 0002.00 before the End of data line by typing a C (copy) on line 0002.00 and a B (before) on the End of data line and pressing the Enter key. Then, change the C and D to X and Y on line 0003.00.

    You will see the following display.

```
Columns . . . .:   1  71              Edit                    USERXX/MLGSRC
Find  . . .                                                          TEST
FMT C  .....CL0N01N02N03Factor1+++OpcdeFactor2+++ResultLenDHHiLoEqComments++++
       *************** Beginning of data ************************************
0001.00      C                   ADD  A         B       50
0002.00      C                   MOVE C         D
0003.00      C                   MOVE X         Y
       ****************** End of data ***************************************

F3=Exit                  F4=Prompt               F5=Refresh
F10=Top                  F11=Bottom              F24=More keys
```

    SEU has a great deal more function than this example has shown. You can learn more about how to work with SEU by pressing the Help key and reading the text.

20. Press F3 (Exit). An Exit display appears.

```
                                  Exit

Type choices, press Enter.

  Change/create member . . . . . .  Y            Y=Yes, N=No
    Member . . . . . . . . . . . .  TEST         Name
    File . . . . . . . . . . . . .  MLGSRC       Name
      Library . . . . . . . . . . .  USERXX       Name
    Text . . . . . . . . . . . . .  Test for SEU

    Resequence member . . . . . .  Y            Y=Yes, N=No
      Start . . . . . . . . . . . .  0001.00      0000.01 - 9999.99
      Increment . . . . . . . . . .  01.00        00.01 - 99.99

  Print member . . . . . . . . . .  N            Y=Yes, N=No

  Return to editing . . . . . . .  N            Y=Yes, N=No

  Go to member list . . . . . . .  N            Y=Yes, N=No


F3=Exit       F5=Refresh       F12=Previous
```

21. Up to this point, the source statements you have entered have been placed in an SEU work space. What you specify on the Exit display determines whether the work space is actually placed in the member. If you wanted the source statements to be placed in the member TEST in this example, you would take the default of Y for Change/create member. If this is not the case, select N for Change/create member, so the source file will remain as it was when you entered SEU. Specifying N cancels whatever changes you have made.

    Press the Enter key.

Glossary

access. To read; the ability to use or read.

access method. A method used to read a record from, or to write a record into a file. Access can be sequential (records are referred to one after another in the order in which they appear in the file), it can be random (the individual records can be referred to in any order), or it can be dynamic (records can be accessed sequentially or randomly, depending on the specific request).

access path. The order in which records in a database file are organized for processing by a program. See *arrival sequence access path* and *keyed sequence access path*.

accounting code. A 15-character field, assigned to a job by the system when it is processed by the system, that is used to collect statistics for the system resources used for that job when job accounting is active.

address. The location in the storage of a computer where particular data is stored. Also, the numbers that identify such a location.

algorithm. A finite set of well-defined rules for the solution of a problem in a finite number of steps.

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.

area-specific help. In an application program, help information supplied by the programmer for the area of the screen where the cursor is located when the person using the program presses the Help key.

array. A series of elements with like characteristics. An array can be searched for a uniquely identified element, or elements in an array can be accessed by their position relative to other elements. Contrast with *table*.

arrival sequence access path. An access path to a database file that is arranged according to the order in which records are stored in the physical file. See also *keyed sequence access path* and *access path*.

AS/400 Query. The IBM licensed program used to select, format, and analyze information from data files to produce reports and other files.

ascending key sequence. The arrangement of data in order from the lowest value of the key field to the highest value of the key field. Contrast with *descending key sequence*.

ascending sequence. The arrangement of data in order from the lowest value to the highest value, according to the rules for comparing data. Contrast with *descending sequence*.

assumed value. A value supplied by the system when no value is specified by the user.

attention identifier. A character in a data stream indicating that the user pressed a key (such as the Enter key) that requests an action by the system.

Attention-key-handling program. A user-defined program that is called when the work station user presses the Attention (Attn) key.

attribute. (1) A characteristic or property of one or more objects. (2) In SQL, in database design, a characteristic of an entity; for example, the telephone number of an employee is one of that employee's attributes.

attribute character. A character associated with a field in a display file record format that defines how the field is displayed.

authorization list. A list of two or more user IDs and their authorities for system resources.

authorize. Permit or give authority to.

automatic report. A function of the RPG/400 licensed program that uses simplified specifications and standard RPG/400 specifications to create a complete RPG/400 source program.

automatic report program. A set of instructions (program) that use the RPG/400 automatic report function. See also *automatic report*.

automatic report specifications. An RPG/400 coding form that the programmer uses to specify options for an automatic report program.

auxiliary storage. All addressable disk storage other than main storage.

backup. Pertaining to an alternative copy used as a substitute if the original is lost or destroyed.

base. The numbering system in which an arithmetic value is represented.

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*.

batch processing. A method of running a program or a series of programs in which one or more records (a batch) are processed with little or no action from the user or operator. Contrast with *interactive processing*.

batch subsystem. A part of main storage where batch jobs are processed.

binary. A numbering system with a base of two (0 and 1).

bit. Either of the binary digits, 0 or 1. Compare with *byte*.

blank after. An output specification option that changes the contents of a field so that it contains either zeros (if it is a numeric field) or blanks (if it is a character field) after that field is written to the output record.

block. (1) A group of records that are recorded or processed as a unit. (2) A sequential group of statements (defined using line commands) that are processed as a unit. (3) In SEU, a group of records (defined using line commands) that are processed as a unit.

block copy. To copy two or more adjoining source records from one part of a source member to another part, or from one source member to another.

block delete. To delete two or more adjoining source records from a source member.

block exclude. To exclude two or more adjoining records from the Edit or Browse display.

block move. To move two or more adjoining source records from one part of a source member to another part, or from one of a source member to another.

block overlay. To overlay two or more adjoining records with other records defined by the Copy or Move line command.

both field. A field that can be used for either input data or output data.

break delivery. The method of delivering messages to a message queue in which the job associated with that message queue is interrupted as soon as the message arrives.

breakpoint. A place in a program (specified by a command or a condition) where the system stops processing of that program and gives control to the display station user or to a specified program.

breakpoint program. For a batch job, a user program that can be called when a breakpoint is specified.

buffer. (1) A routine or an area of storage that corrects for the different speeds of data flow or timings of events, when transferring data from one device to another. (2) 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.

calculation specifications. A coding form on which the programmer describes the processing to be done by the program.

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.

chain. (1) A group of logically linked records. (2) In DFU, a way to change from one display format to another after the user signals that the first display format was completed. (3) In RPG/400, an operation code that reads input records identified by specified relative record numbers or keys.

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 key. A keyboard key that allows the user to type into the system the character shown on the key. See also *function key*.

character string. A sequence of consecutive characters that are used as a value.

character variable. Character data whose value is assigned and/or changed while the program is running.

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*.

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 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 key indicator. An indicator defined to correspond with the function keys to tell the program when one of the function keys is pressed.

command line. The blank line on a display where commands, option numbers, or selections can be entered.

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.

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.

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.

compiler listing. A printout that is produced by compiling a program or creating a file and that optionally includes, for example, a line-by-line list of the high-level language source, a cross-reference list, diagnostic information; and for programs, the description of the externally described files. See also *source listing*.

completion message. A message that tells the operator when work is successfully ended.

concept. An abstract idea.

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.

conditioning indicator. An indicator used to specify when to do calculations or which characteristics apply to a record format or field.

constant. (1) Data that has an unchanging, predefined value to be used in processing. (2) In RPG/400, data that has an unchanging, predefined value to be used in processing. A constant does not change during the running of a program, but the contents of a field or variable can.

continuation line. (1) A line of a source statement where characters are entered when the source statement cannot be contained on the previous line or lines. (2) An additional line (or lines) required to continue the coding of a CL command or a DDS keyword and its value. (3) In RPG/400, additional lines specified on the file description specifications to provide more information about the file being defined.

control field. One or more fields that are compared from record to record to determine when the information in the fields changes. When the information changes, the control level indicator (L1 through L9) assigned to a control field is set on.

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.

control specification. A coding form on which the programmer provides information that affects the creating and running of programs.

control statement. (1) In programming languages, a statement that is used to interrupt the continuous sequential processing of programming statements; for example, a conditional statement such as IF, PAUSE, or STOP. (2) Entries on a control specification.

control-level indicator. An indicator (L1 through L9) used to specify certain fields as control fields and to control the operations that are performed at total and detail time in the RPG/400 program cycle.

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 *job date* and *system date*.

current library. The library that is specified to be the first user library searched for objects requested by a user. The name for the current library can be specified on the Sign-On display or in a user profile. When you specify an object name (such as the name of a file or program) on a command, but do not specify a library name, the system searches the libraries in the system part of the library list, then searches the current library before searching the user part of the library list. The current library is also the library that the system uses when you create a new object, if you do not specify a library name.

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.

data area. A storage area used to communicate data such as CL variable values between the programs within a job and between jobs. The system-recognized identifier for the data area is *DTAARA.

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 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.

data file utility (DFU). The part of the AS/400 Application Development Tools licensed program that is used to enter, maintain, and display records in a database file.

data structure. An area in storage that defines the layout of the fields, called subfields within the area. A data structure can be either program-described or externally described.

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*.

DDS. See *data description specifications (DDS)*.

debug mode. An environment in which programs can be tested.

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 record. A record that consists entirely of default values (numeric fields are filled with zeros, character fields are filled with blanks, or either data type, numeric or character, can be filled with a value specified by the user with the DFT keyword in DDS).

default value. A value supplied by the system that is used when no value is specified by the user. See also *assumed value*.

descending key sequence. The arrangement of data in order from the highest value of the key field to the lowest value of the key field. Contrast with *ascending key sequence*.

descending sequence. The arrangement of data in order from the highest value to the lowest value, according to the rules for comparing data. Contrast with *ascending sequence*.

detail calculation. Specified calculation operations that are performed for every record read.

detail line. A detail record in an output file.

detail time. That part of the RPG/400 program cycle in which calculation and output operations are performed for each record read. Contrast with *total time*.

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.

DFU. See *data file utility (DFU)*.

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.

display file. A device file created by the user to support a display 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.

do group. (1) A set of commands in a control language program defined by a DO command and an ENDDO command that is conditionally processed as a group. (2) In RPG/400, a group of calculations done one or more times based on the results of comparing factor 1 and factor 2 of certain calculation operations (for example, DOUXX). A DO operation and an END operation are the delimiters for a do group.

dump. To copy data from main or auxiliary storage onto an external medium, such as tape, diskette, or printer.

duplicate key value. The occurrence of the same value in a key field or in a composite key in more than one record in a file.

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 make changes to a document by adding, changing, or removing text. (3) 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 word. A user-defined word with a specific format that indicates how editing should be done. Contrast with *edit code*.

element. (1) In a list of parameter values, one value. (2) The smallest addressable unit of an array or table.

escape message. A message that reports a condition that caused the program to end before the requested function was complete.

external indicators. Indicators that can be set by another program before a program is run, or changed by another program while the program is running. Valid external indicators are U1 through U8.

external message queue. The part of the job message queue that sends messages between an interactive job and the work station user. For batch jobs, messages sent to the external message queue appear only in the job log.

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*.

factor. An entry (for example, a field name, file name, literal, or data structure) that identifies the data to be used in an operation.

field. A group of related characters (such as name or amount) that are treated as a unit in a record.

field definition. Information that describes the characteristics of data in a field.

field indicator. An indicator shows whether a given field in an input record is plus, minus, zero, or blank.

field record relation indicator. An indicator that associates fields in an input record with a particular record type. The field record relation indicator is normally used when the record type is one of several in an OR relationship.

field reference file. A physical file that contains no data, only descriptions of fields.

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 definition. (1) In RPG/400, file description and input specifications that describe the records and fields in a file. (2) Information that describes the contents and characteristics of a file.

file description specifications. Coding form on which the programmer identifies and describes all files used in a program.

file name. The name used by a program to identify a file.

file overrides. Attributes specified at run time that change the attributes specified in the file description or in the program.

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.

full procedural file. A file that uses input operations controlled by programmer-specified operation codes instead of by the program cycle.

function key. A keyboard key that allows the user to select keyboard functions or programmer functions. Contrast with *character key*.

function key indicator. An indicator that is set on when a valid corresponding function key is pressed. Valid function key indicators are KA. through KN and KP through KY.

general-purpose library. The library shipped with the system that contains IBM-provided objects required for many system functions and user-created objects that are not explicitly placed in a different library when they are created. Named QGPL.

group job. One of up to sixteen interactive jobs that are associated in a group with the same work station device and user.

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 OfficeVision/400.

HLL. See *high-level language (HLL)*.

IDDU. See *interactive data definition utility (IDDU)*.

indicator. 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.

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.

initialize. To set the addresses, switches, or the contents of storage to zero, or to the starting value set by the manufacturer.

input field. A field specified in a display file or database file that is used for data the user supplies. Contrast with *output field*.

input file. A database or device file that has been opened to allow records to be read. Contrast with *output file*.

input specifications. The means by which the programmer describes the input records and their fields, adds RPG functions to an externally described file, or defines a data structure and its subfields.

input/output. Data provided to the computer or data resulting from computer processing.

inquiry message. A message that gives information and requests a reply.

inquiry program. (1) A program that allows an operator to get information from a disk file. (2) A program that runs while the system is in inquiry mode.

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*.

interactive processing. A processing method in which each operator action causes a response from the program or the system. Contrast with *batch processing*.

interface. A shared boundary. An interface might be the hardware to connect two devices or it might be a part of main storage, or registers used by two or more computer programs.

job. A unit of work to be done by a computer.

job date. The date associated with a job. The job date usually assumes the system date, but it can be changed by the user. See also *creation date* and *system date*.

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 message queue. A message queue that is created for each job. A job message queue receives requests to be processed (such as commands) and sends messages that result from processing the requests. A job message queue consists of an external message queue and a set of program message queues. See also *external message queue* and *program message queue*.

job name. The name of the job as identified to the system. For an interactive job, the job is assigned the name of the work station at which the job was started; for a batch job, the name is specified in the command used to submit the job. Contrast with *qualified job name*.

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 logical file. A logical file that combines (in one record format) fields from two or more physical files. See also *logical file*.

journal. A system object used to record entries in a journal receiver when a change is made to the database files associated with the journal. The object type is *JRN. See also *journal receiver*.

journal receiver. A system object that contains journal entries recorded when changes are made to the data in database files or the access paths associated with the database files. The object type is *JRNRCV. See also *journal*.

key. The value used to identify a record in a keyed sequence file.

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. See also *arrival sequence access path* and *access path*.

keyword. (1) A name that identifies a parameter in a command. (2) A name that identifies a function. (3) A word that is essential to the meaning and structure of a statement in a programming language.

keyword functions. The result of processing DDS keywords in a record format specified on an operation. See also *operation*.

label. (1) The name of a file on a diskette or tape. (2) In RPG/400, a symbolic name that represents a specific location in a program. A label can serve as the destination point for one or more branching operations.

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.

level indicator. Two characters (L0 through L9 and LR) that control calculation and output processing during total time.

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 command. An abbreviation used to request a function for a specific line or lines from the line number area of the line or lines affected. For example, C for Copy or M for Move.

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. See also *sequence number*.

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.

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.

locked keyboard. A keyboard condition where the display station accepts no input.

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. See also *join logical file*. Contrast with *physical file*.

main storage. The part of the processing unit where programs are run. Contrast with *auxiliary storage*.

master file. A collection of permanent information, such as a file of customer addresses.

member. Different sets of data within one file. See also *source member*.

message description. Information describing a particular message.

message file. An object that contains message descriptions. The system-recognized identifier for the object type is *MSGF.

message queue. A list on which messages are placed when they are sent to a person or program.

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.

next record. The record that logically follows the current record of a file.

notify delivery. The method of delivering messages to a message queue in which the work station user is notified that a message arrived. The signal is a light or an audible alarm.

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.

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.

object name. The name of an object. Contrast with *qualified name*.

object owner. A user who creates an object or to whom the ownership of an object was reassigned. The object owner has complete control over the object.

offline. Pertaining to the operation of a functional unit that is not under the continual control of the system. Contrast with *online*.

online. Pertaining to the operation of a functional unit that is under the continual control of the system. Contrast with *offline*.

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. Contrast with *close*.

operating system. A collection of system programs that control the overall operation of a computer system.

Operating System/400 (OS/400). The operating system used by the AS/400 system.

operation. The result of processing statements in a high-level language. See also *keyword functions*.

operation code. (1) A code used to represent the operations of a computer. (2) In RPG/400, a word or abbreviation, specified in the calculation specifications, that identifies an operation.

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.

OS/400. See *Operating System/400 (OS/400)*.

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. Contrast with *input field*.

output file. A database or device file that has been opened to allow records to be written. Contrast with *input file*.

output queue. An object that contains a list of spooled files to be written to an output device, such as a printer.

output specifications. The means by which the programmer describes the output records and their fields or adds RPG functions to an externally described output file.

overflow. The condition that occurs when the last line specified as the overflow line to be printed on a page has been passed.

overflow indicator. An indicator that signals when the overflow line on a page has been printed or passed. The indicator (OV and OA through OF) can be used to specify which lines are to be printed on the next page.

overlay. (1) To write over (and therefore destroy) an existing file. (2) A program segment that is loaded into main storage and replaces all or part of a previously loaded program segment.

owner. The user who creates an object (or is named the owner of an object).

packed decimal format. Representation of a decimal value in which each byte within a field represents two numeric digits except the far right byte, which contains one digit in bits 0 through 3 and the sign in bits 4 through 7. For all other bytes, bits 0 through 3 represent one digit; bits 4 through 7 represent one digit. For example, the decimal value +123 is represented as 0001 0010 0011 1111. Contrast with *zoned decimal format*.

page. (1) A 512-byte block of information that can be moved between auxiliary storage and main storage. (2) Each group of records in a subfile that are displayed at the same time. (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. Contrast with *page up*.

page up. To move down the data shown on the display, which allows the user to move toward the beginning of the data. Contrast with *page down*.

parameter. A value supplied to a command or program that is used either as input or controls the actions of the command or program.

parameter list. A list of values that provide a means of associating addressability of data defined in a called program with data in the calling program. It contains parameter names and the order in which they are to be associated in the calling and called program.

PDM. See *programming development manager (PDM)*.

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*.

primary file. (1) In DDS, in a join logical file, the first physical file specified on the JFILE keyword. Contrast with *secondary file*. (2) In RPG/400, if specified, the main file from which an RPG/400 program first reads a record in the program cycle. In multifile processing, the primary file is used to determine whether the MR indicator is set on. Contrast with *full procedural file*.

print file. A file created by the host system that is printed on your system.

print queue. A list of output waiting to be printed by the system.

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.

processing. The action of performing operations and calculations on data.

processing unit. The part of the system that performs instructions and contains main storage.

production library. A library containing objects needed for normal processing. Contrast with *test library*.

profile. Data that describes the characteristics of a user, program, device, or remote location.

program cycle. The series of operations performed by the computer for each record read.

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 variable. A named changeable value that can exist only within programs. Its value cannot be obtained or used when the program that contains it is no longer running.

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*.

programming development manager (PDM). A part of the AS/400 Application Development Tools licensed program that allows users to perform several operations (such as copy, delete, and rename) from lists of libraries, objects, and members. PDM also allows users to create user-defined options to perform operations.

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.

QGPL. See *general-purpose library*.

qualified job name. A job name and its associated user name and a system-assigned job number. Contrast with *job name*.

qualified name. The name of the library containing the object and the name of the object. Contrast with *object name*.

Query. The shortened name for the IBM AS/400 Query licensed program.

query. A request to select and copy from a file or files one or more records based on defined conditions. For example, a request for a list of all customers in a customer master file, whose balance is greater than $1,000.

query definition. Information about a query that is stored in the system.

queue. A list of messages, jobs, or files waiting to be read, processed, printed, or distributed in the order they appear in the list.

random processing. A method of processing in which records can be read from, written to, or deleted from a file order requested by the program that is using them.

read operation. An input operation that obtains a record from a file and passes it to a program.

record. A collection of related data or words, treated as a unit; such as one name, address, and telephone number.

record format. See *record*.

resource. Any part of the system required by a job or task, including main storage, devices, the processing unit, programs, files, libraries, and folders.

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.

return indicator. An indicator to an RPG/400 program that control should be returned to the calling program.

routine. A set of statements in a program that causes the system to perform an operation or a series of related operations.

RPG. Report Program Generator. A programming language designed for writing application programs for business data processing requirements. The application programs range from report writing and inquiry programs to applications such as payroll, order entry, and production planning.

RPG/400. An IBM licensed program that is the SAA RPG programming language available on the AS/400 system, including system-specific functions.

SAA. See *Systems Application Architecture (SAA)*.

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. Contrast with *restore*.

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.

SDA. See *screen design aid (SDA)*.

search value. User-defined information that is used to either make a list of filed documents with similar document details or find a directory entry.

secondary file. (1) Any input file other than the primary file. (2) In a join logical file, any physical file, other than the first physical file, that is specified on the JFILE keyword. Contrast with *primary file*.

security officer. A person assigned to control all of the security authorizations provided with the system. A security officer can, for example, remove password or resource security; or add, change, or remove security information about any system user.

sequence checking. A function that checks the sequence of records in input, update, or combined files used as primary and secondary files.

sequence number. (1) The number of a record that identifies the record within the source member. (2) A field in a journal entry that contains a number assigned by the system. This number is initially 1 and is increased by 1 until the journal is changed or the sequence number is reset by the user. See also *line number*.

sequential processing. A method of processing in which records are read, written to, or deleted in the order determined by the value of the key field.

session. The length of time that starts when a user signs on and ends when the user signs off at a display station.

SEU. See *source entry utility (SEU)*.

severity code. A number that indicates how important a message is. The higher the number, the more serious the condition.

shift. A keyboard action to allow uppercase or other characters to be entered.

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 listing. A portion of a compiler listing that contains source statements and, optionally, test results. See also *compiler listing*.

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 statement. A statement written in symbols of a programming language. For example, RPG/400, COBOL, BASIC, PL/I, and DDS statements are source statements.

spool. The system function of putting jobs into a storage area to wait to be printed or processed.

spooled file. A file that holds output data waiting to be printed, or input data waiting to be processed by the program.

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.

statement. An instruction in a program.

store. To put or keep data in a storage device.

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.

subprogram. A called program. A subprogram is combined with the calling program at run time to produce a run unit and is below the calling program in the program stack.

subroutine. (1) A group of instructions within another group of instructions that can be called by another program or another subroutine. (2) In RPG/400, a group of calculation specification statements in a program that can be run several times in that program.

subsystem. An operating environment, defined by a subsystem description, where the system coordinates processing and resources.

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. See also *creation date*, and *job date*.

system library. The library shipped with the system that contains objects, such as authorization lists and device descriptions created by a user; and the licensed programs, system commands, and any other system objects shipped with the system. The system identifier is QSYS.

system name. An IBM-supplied name that uniquely identifies the system. It is used as a network value for certain communications applications such as APPC.

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 Application Architecture (SAA). An architecture defining a set of rules for designing a common user interface, programming interface, application programs, and communications support for strategic operating systems such as OS/2, OS/400, VM/370, and MVS/370.

table. In RPG/400, a series of elements with like characteristics. A table can be searched for a uniquely identified element, but elements in a table cannot be accessed by their position relative to other elements. Contrast with *array*.

table file. An input file that contains a table.

test library. A user-defined library used for debugging operations that does not contain objects needed for normal processing. Contrast with *production library*.

total time. The part of the RPG/400 program cycle in which calculation and output operations specified for a group of records are done. Contrast with *detail time*.

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.

update file. A file from which a program reads a record, changes data fields in the record, and writes the record back to the location from which it came.

user identification (user ID). The name used to associate the user profile with a user when a user signs on the system. See also *user profile name*.

user message queue. A user-created object used to receive messages sent from the system, other users, and application programs.

user password. A unique string of characters that a system user must enter to identify himself to the system, if the system resources are secured.

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 profile name. The name or code that the system associates with a user when he or she signs on the system. Also known as user ID. See also *user identification (user ID)*.

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.

work station. A device used to transmit information to or receive information from a computer; for example, a display station or printer.

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.

zoned decimal format. A format for representing numbers in which the digit is contained in bits 4 through 7 and the sign is contained in bits 0 through 3 of the far right byte; bits 0 through 3 of all other bytes contain 1's (hex F). For example, in zoned decimal format, the decimal value of +123 is represented as 1111 0001 1111 0010 1111 0011. Same as unpacked decimal format. Contrast with *packed decimal format*.

Bibliography

You may want to refer to other AS/400 manuals for more specific information about a particular topic. The following is a list of the additional manuals and the information you may want to find in them sorted by topic.

Programming Manuals

- *Backup and Recovery Guide*, SC41-8079

  The *Backup and Recovery Guide* provides the programmer with information about the different media available to save and protect system data, as well as a description of how to record changes made to database files and how that information can be used for system recovery and activity report information. This manual describes user auxiliary storage pools (ASPS) and checksums along with other availability recovery topics. It also provides the system operator with information about how to install the system again from backup.

- *Programming: Control Language Programmer's Guide*, SC41-8077

  The *CL Programmer's Guide* provides the application programmer with a wide-ranging discussion of AS/400 programming topics, including the following:

  - A general discussion of objects and libraries
  - Control language (CL) programming, controlling flow and communicating between programs, working with objects in CL programs, and creating CL programs
  - Predefined and impromptu messages and message handling
  - How to define and create user-defined commands and menus
  - Application testing, including debug mode, breakpoints, traces, and display functions

- *Programming: Control Language Reference*, SC41-0030

  The *CL Reference* provides the application programmer with a description of the AS/400 control language (CL) and its commands. Each command description includes a syntax diagram, parameters, default values, keywords, and an example. The information should be used to refer to the control language commands to request functions of the Operating System/400 (5728-SS1) licensed program and of the various languages and utilities.

- *Data Management Guide*, SC41-9658

  The *Data Management Guide* provides the application programmer with information about using files in application programs. A file is the OS/400 object type that provides storage of and access to data in the database, or devices such as display stations and printers, or on another system. This manual includes information on the following topics.

  - Fundamental structure and concepts of data management support on the system
  - Overrides and file redirection (temporarily making changes to files when an application program is run)
  - Copying files by using system commands to copy data from one place to another
  - Tailoring a system using double-byte data

- *Guide to Programming Application and Help Displays*, SC41-0011

  The *Guide to Programming Displays* provides the application programmer with information about using DDS to create and maintain displays for any application, creating online help information for displays and commands, and working with display files on the system.

- *Guide to Programming for Printing*, SC41-8194

  The *Guide to Programming for Printing* provides information on how to control and understand printing, including printer files, spooling, printer connectivity, PC attached printers, and advanced function printing (AFP) printers.

- *Guide to Programming for Tape and Diskette*, SC41-0012

  The *Guide to Programming for Tape and Diskette* provides information about data management support for tapes and diskettes.

- *Programming: Performance Tools/400 Guide*, SC41-8084

  The *Performance Tools/400 Guide* provides the programmer with information about what AS/400 Performance Tools are, gives an overview of the tools, and tells how the tools can be used to help manage system performance. The manual gives instructions on how to approach the analysis of system performance and how to do system performance measurement, reporting, capacity planning, and application analysis.

  This manual also includes information about additional functions and related commands for further analysis.

- *Security Concepts and Planning*, SC41-8083

  The *Security Concepts and Planning* manual provides the programmer (or someone who is assigned the responsibilities of a security officer) with information about system security concepts, planning for security, and setting up security on the system. This manual does not describe secu-

rity for specific licensed programs, languages, and utilities.

This manual tells how system security support can be used to:

- Protect the system and the data from being used by people who do not have the proper authorization
- Protect the data from intentional or unintentional damage or destruction
- Keep security information up-to-date
- Set up security on the system

- *Database Guide*, SC41-9659

  The *Database Guide* provides the application programmer with a detailed discussion of the AS/400 database organization, including information on how to create, describe, and manipulate database files on the system.

  This manual also describes how to define files to the system using OS/400 DDS (data description specifications) keywords.

- *Data Description Specifications Reference*, SC41-9620

  The *DDS Reference* provides the application programmer with detailed descriptions of the entries and keywords needed to describe database files (both logical and physical) and certain device files (for displays, printers, and intersystem communications function (ICF)) external to the user's programs.

- *Programming: Work Management Guide*, SC41-8078

  The *Work Management Guide* provides the programmer with information about how to create a work management environment and how to change it. Other topics include the following.

  - A description of tuning the system
  - Collecting performance data including information on record formats and contents of the data being collected
  - Working with system values to control or change the overall operation of the system
  - A description of how to gather data to determine who is using the system and what resources they are using

Application Development Tools Manuals

- *Application Development Tools: Data File Utility User's Guide and Reference*, SC09-1381

  The *DFU User's Guide and Reference* provides the application programmer with information about using the Application Development Tools data file utility (DFU) to create programs to enter data into files, update files, inquire into files, and run DFU programs.

  This manual also provides the work station operator with activities and material to learn DFU.

- *Application Development Tools: Screen Design Aid User's Guide and Reference*, SC09-1340

  The *SDA User's Guide and Reference* provides the application programmer or system operator with information about using the Application Development Tools screen design aid (SDA) to design, create, and maintain displays, menus, and online help information. The manual contains examples and information to help the user learn how to use SDA on the AS/400 system and in the System/38 environment of the AS/400 system.

- *Application Development Tools: Programming Development Manager User's Guide and Reference*, SC09-1339

  The *PDM User's Guide and Reference* provides the application programmer with information about using the Application Development Tools programming development manager (PDM) to work with lists of libraries, objects, members, and user-defined options to easily do such operations as copy, delete, and rename.

  This manual contains activities and some material to help the user learn PDM. The most commonly used operations and function keys are explained in detail using examples.

- *Application Development Tools: Source Entry Utility User's Guide and Reference*, SC09-1338

  The *SEU User's Guide and Reference* provides the application programmer with information about using the Application Development Tools source entry utility (SEU) to create and edit source members. The manual explains how to start and end an SEU session and how to use the many features of this full-screen text editor. The manual contains examples to help both new and experienced users accomplish various editing tasks, from the simplest line commands to using predefined prompts for high-level languages and data formats.

Language Manuals

- *Languages: Systems Application Architecture AD/Cycle* RPG/400* Reference*, SC09-1349

  The *RPG/400* Reference* provides the application programmer with the information needed to write programs for the AS/400 system using the RPG/400 programming language. This manual describes, position by position, the valid entries for all RPG specification forms, and provides a detailed description of all the operation codes. This manual also contains information on the RPG logic cycle, arrays and tables, editing functions, and indicators.

- *Languages: Systems Application Architecture AD/Cycle* RPG/400* User's Guide*, SC09-1348

  The *RPG/400* User's Guide* provides the application programmer with the information needed to write, test, and maintain RPG/400 programs on

the AS/400 system. The manual provides information on data organizations, data formats, file processing, multiple file processing, automatic report function, RPG command statements, testing and debugging functions, application design techniques, problem analysis, and compiler service information. The differences between the System/38 RPG III, System/38 compatible RPG, and RPG/400 are identified.

Other Manuals

- *System Concepts*, GC41-9802

  The *System Concepts* manual provides the application programmer, system administrator, or system operator with a general understanding of the concepts related to the overall design and use of the AS/400 system and its operating system. This manual includes general information about AS/400 features such as user interface, object, work, and system management, data management, database, communications, environments, Office and PC Support, and architecture.

- *Query/400 User's Guide*, SC41-9614

  The *Query/400 User's Guide* provides the administrative secretary, business professional, or programmer with detailed information about how to use AS/400 Query to get data from any database file. It describes how to sign on to Query, and how to define and run queries to create reports containing the selected data. This manual describes all the tasks for defining the query of one or more files, the way the report is to look when the query is run. and whether the report is to be displayed, printed, or put in a database file.

- *System Operator's Guide*, SC41-8082

  The *Operator's Guide* provides the system operator or system administrator with information about how to use the system unit operator display, send and receive messages, respond to error messages, start and stop the system, use control devices, work with program temporary fixes (PTFS) and process and manage jobs on the system.

Index

D

E

F

G

H

I

J

K

L

M

N

O

R

S

T

U

V

W

X

Reader's Comments

Application System/400™
Application Development by Example
Version 2

Publication No. SC41-9852-00

Use this form to tell us what you think about this manual. If you have found errors in it, or if you want to express your opinion about it (such as organization, subject matter, appearance) or make suggestions for improvement, this is the form to use.

To request additional publications, or to ask questions or make comments about the functions of IBM products or systems, you should talk to your IBM representative or to your IBM authorized remarketer. This form is provided for comments about the information in this manual and the way it is presented.

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you.

Be sure to print your name and address below if you would like a reply.

Name

Address

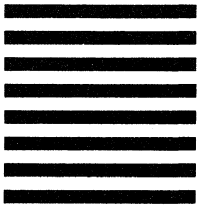
Company or Organization

Phone No.

Cut c
Alon

Fold and Tape **Please do not staple** Fold and Tape

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES



BUSINESS REPLY MAIL

FIRST CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

ATTN DEPT 245
IBM CORPORATION
3605 HWY 52 N
ROCHESTER MN 55901-7899

Fold and Tape **Please do not staple** Fold and Tape

Cut or
Along

Reader's Comments

Application System/400™
Application Development by Example
Version 2

Publication No. SC41-9852-00

Use this form to tell us what you think about this manual. If you have found errors in it, or if you want to express your opinion about it (such as organization, subject matter, appearance) or make suggestions for improvement, this is the form to use.

To request additional publications, or to ask questions or make comments about the functions of IBM products or systems, you should talk to your IBM representative or to your IBM authorized remarketer. This form is provided for comments about the information in this manual and the way it is presented.

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you.

Be sure to print your name and address below if you would like a reply.

Name

Address

Company or Organization

Phone No.

Reader's Comments
SC41-9852-00

IBM®

Cut o
Along

Fold and Tape | **Please do not staple** | Fold and Tape

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

ATTN DEPT 245
IBM CORPORATION
3605 HWY 52 N
ROCHESTER MN 55901-7899

Fold and Tape | **Please do not staple** | Fold and Tape

Cut or
Along

Program Number: 5738-SS1

Printed in Denmark by Bonde's

SC41-9852-00