SC21-7881-1

File No. S38-20

IBM System/38

IBM System/38
Application Example I

SC21-7881-1

File No. S38-20

IBM System/38

IBM System/38 Application Example I

Second Edition (December 1981)

This is a major revision of, and obsoletes SC21-7881-0. This publication should be reviewed in its entirety.

This edition applies to release 3, modification 0, of the IBM System/38 Control Program Facility (Program 5714-SS1), RPG III (Program 5714-RG1), the Interactive Data Base Utilities (Program 5714-UT1); and to all subsequent versions and modifications until otherwise indicated in new editions or technical newsletters. Changes are periodically made to the information herein; these changes will be reported in technical newsletters or in new editions of this publication.

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

It is possible that this material may contain reference to, or information about, IBM products (machines and programs), programming, or services that are not announced in your country. Such references or information must not be construed to mean that IBM intends to announce such IBM products, programming, or services in your country.

Publications are not stocked at the address below. Request for copies of IBM publications and for technical information about the system should be made to your IBM representative or to the branch office serving your locality.

This publication could contain technical inaccuracies or typographical errors. Use the Reader's Comment Form at the back of this publication to make comments about this publication. If the form has been removed, address your comments to IBM Corporation, Publications, Department 245, Rochester, Minnesota 55901. IBM may use and distribute any of the information you supply in any way it believes appropriate without incurring any obligation whatever. You may, of course, continue to use the information you supply.

Contents

About This Manual

PURPOSE OF THIS MANUAL

This publication presents a hypothetical mailing list application that illustrates how you design and implement application programs on System/38. Various approaches to the applications are shown, starting with batch type processing and progressing to interactive type processing. Because these approaches are designed to teach you how to use various System/38 functions, they may not be practical for your use and may vary from the approaches presented in System/38 customer classes.

Note: The chapters should be read in sequence because the information presented assumes that you have an understanding of the previous chapters.

ORGANIZATION OF THIS MANUAL

This publication is divided into eleven chapters and three appendixes:

Chapter 1 gives an overview of this publication, introduces the mailing list application, and summarizes the various application approaches described in this publication.

Chapter 2 introduces basic System/38 functions and operations that are used or assumed to be available in subsequent chapters.

Chapter 3 describes an approach that uses diskettes tu enter data for printing mailing labels.

Chapter 4 describes an approach that uses diskettes to enter data for updating a master file and for printing a maintenance report as well as mailing labels.

Chapter 5 describes various system functions that are performed from the work station.

Chapter 6 describes how to use and create libraries and files on the System/38 and in the application.

Chapter 7 describes an approach to interactively enter data for the application and then apply those changes in a batch manner to the master file.

Chapter 8 describes several ways to interactively enter and maintain data for the application, use multiple access paths, and selectively display records.

Chapter 9 describes an approach in which the master file is updated by multiple batches of transactions from diskettes.

Chapter 10 describes an approach that allows work station users to enter new batches of transactions while completed batches are being applied to the master file.

Chapter 11 describes an approach that allows multiple work station users to apply immediate updates to the master file without interferring with one another.

Appendix A discusses multiple libraries and shows how to use them in the application.

Appendix B discusses a series of user-written menus that could be used in the application.

Appendix C presents an alphabetic list of the field reference file field names.

CONVENTIONS

Blank lines are occasionally used in the data description specifications and RPG specifications to improve clarity.

Many of the figures that depict programming components are presented from a conceptual viewpoint and are not intended to represent actual implementation.

Some words are printed entirely in uppercase. These include names of files, libraries, objects, queues, subsystems, and commands.

He means *he* or *she*.

SUMMARY OF CHANGES

The following major changes have been made:

- The publication has been reorganized to place the discussion of each application approach in a separate chapter.

- An illustrated summary of the various application approaches described in the publication has been added (Chapter 1).

- All examples of the source entry utility, data file utility, and query utility have been updated (Chapters 5, 7, and 8).

- An example of using the screen design aid has been added (Chapter 8).

Because of these extensive changes, the publication should be reviewed in its entirety.

WHAT YOU SHOULD KNOW

Before reading this publication, you should:

- Be able to write application programs using RPG.

- Read the *IBM System/38 Introduction*, GC21-7728, and *IBM System/38 Control Program Facility Concepts Manual*, GC21-7729, or have an equivalent level of knowledge (including System/38 terminology).

IF YOU NEED MORE INFORMATION

The following manuals contain additional information on the functions and procedures described in this publication, and help you determine where to look for that information.

CPF (Control Program Facility) Commands and Functions

- *IBM System/38 Control Program Facility Programmer's Guide*, SC21-7730
  - Creating control language programs
  - Using program variables
  - Invoking programs
  - Using data base files in programs
  - Access paths
  - Using DDS to describe files
  - Creating a physical file
  - Creating a logical file
  - Creating a field reference file
  - Creating a source file
  - Creating a display file
  - Overriding files
  - Copying files
  - Using data base logging in recovery
  - Creating user profiles
  - Saving/restoring objects and libraries

- *IBM System/38 Control Language Reference Manual*, SC21-7731
  - Control language syntax and syntax diagrams
  - Details on all parameters of control language commands

- *IBM System/38 Control Program Facility Reference Manual—Data Description Specifications*, SC21-7806
  - Referencing previously defined fields
  - Specifying record formats
  - Specifying key fields
  - Specifying select/omit fields
  - Specifying DDS keywords

- *IBM System/38 Programmer's/User's Work Station Guide*, SC21-7744
  - Signing on and off a work station
  - Interacting with displays
  - Using command entry and prompt facilities
  - Using the programmer menu
  - Using the program call menu
  - Handling messages from a work station

- *IBM System/38 Operator's Guide*, SC21-7735
  - Starting the system
  - Using the system operator menu
  - Submitting batch jobs through spooling
  - Submitting batch jobs from work stations
  - Handling readers and writers
  - Handling job queues and output queues
  - Handling spooled output files
  - Saving/restoring objects, libraries, and the system (procedures)

- *IBM System/38 Problem Determination Guide*, SC21-7876
  - Procedures for resolving problems with jobs or the system
  - Recovering from an abnormal termination

IBM System/38 Guide to Program Product Installation and Device Configuration, GC21-7775
  - Installing CPF
  - Installing languages and utilities

Languages and Utilities

- *IBM System/38 RPG III Reference Manual and Programmer's Guide*, SC21-7725
  - Coding RPG specifications
  - Considerations in specifying file descriptions
  - Using RPG indicators
  - Using operation codes
  - Using the CALL/RETRN function
  - Application design
  - Creating and executing RPG programs

- *IBM System/38 Data File Utility Reference Manual and User's Guide*, SC21-7714
  - Using DFU displays to create or change an application
  - Executing an application using the execution menu
  - Executing an application by entering a command
  - Managing existing applications

- *IBM System/38 Query Utility Reference Manual and User's Guide*, SC21-7724
  - Using query displays to create or change a query application
  - Executing a query
  - Considerations in query design
  - Managing existing applications

- *IBM System/38 Screen Design Aid Reference Manual and User's Guide*, SC21-7755
  - Using SDA displays to create a menu
  - Using SDA displays to create a display record format
  - Using SDA displays to test an existing display record format

- *IBM System/38 Source Entry Utility Reference Manual and User's Guide*, SC21-7722
  - Using SEU displays to enter and change source
  - Using line commands on the edit display

Content and Use of System/38 Publications

- *IBM System/38 Guide to Publications*, GC21-7726
  - Contents of System/38 publications
  - Reading sequences for System/38 publications

- *IBM System/38 Glossary and Master Index*, GC21-7727
  - Index entries from frequently used System/38 publications
  - Glossary of terms used in System/38 publications

OTHER MATERIALS USED

The following materials are used in this publication:

- *IBM System/38 Keyboard Template*, GX21-7756

- *IBM Data Description Specifications*, GX21-7754

- *IBM RPG Control and File Description Specifications*, GX21-9092

- *IBM RPG Extension and Line Counter Specifications*, GX21-9091

- *IBM RPG Calculation Specifications*, GX21-9093

- *IBM RPG Input Specifications*, GX21-9094

- *IBM RPG Output Specifications*, GX21-9090

- *IBM Data Description Specifications Debugging Template*, GX21-7717

- *IBM Printer/Display Layout*, GX21-9174

Chapter 1. Introduction

OVERVIEW OF PUBLICATION

This publication presents various design approaches to an application to help you understand designing and implementing application programs on System/38. A mailing list application is used because it is simple in concept and yet it contains many of the basic requirements of any application.

This chapter introduces the mailing list application and summarizes various design approches to implementing the application. These various approaches and the functions used in them are described in subsequent chapters of this publication.

The discussion in each subsequent chapter assumes a knowledge of the previous chapter. Therefore, the chapters should be read in sequence. System/38 functions are introduced as they are needed for a particular application approach. Some chapters are devoted entirely to a discussion of functions to be used in subsequent chapters.

The best way to use this publication is to try the approaches on your system with your data (see the additional details under *Summary of Application Approaches* later in this chapter).

MAILING LIST APPLICATION

This mailing list application allows you to create and maintain a mailing list file (called a master file), and to print mailing labels from the file.

Characteristics

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

Mailing List Record					
Field Description	**Field Name**	**Length (characters)**	**Position Range**	**Decimal Positions**	**Field Type**
Account number	ACTNUM	5	1–5	0	Numeric
Type of account	ACTTYP	1	6	0	Numeric
Name	NAME	18	7–24		Character
Address	ADDR	18	25–42		Character
City	CITY	18	43–60		Character
State	STATE	2	61–62		Character
Zip Code	ZIP	5	63–67	0	Numeric

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

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

A typical mailing list record might look like this:

ACTNUM	ACTTYP	NAME	ADDR	CITY	STATE	ZIP
10522	3	JJ 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 (maintenance of the file).

- The data is printed in various sequences.

- There can be multiple work station users.

- Backup of files and programs must be considered.

Naming Rules and Conventions

General

System/38 requires you to assign names to libraries, programs, and files. In addition to naming rules established for System/38, 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 System/38 naming rules to remember are:

- Objects of the same type within a specific library must have unique names.

- Members of a specific data base 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 III requires that all file, record, and field names be unique within a program.

For more naming rules, refer to the *Control Language Reference Manual* and the *RPG III Reference Manual and Programmer's Guide*.

For the Mailing List Example

In this mailing list example, the following naming conventions are used:

- Object names begin with MLG for *mailing* (such as MLG310).

- 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 MLGTRNL1).

- Program names are 6 or 7 characters; the first 3 are MLG, the second 3 are digits, and the seventh, if present, is C for control language (CL) programs, U for DFU programs, and Q for Query programs. The digits designate the program type as follows:

000-029	Initial menu
030-099	Logical function menu
100-199	Data entry
200-299	Inquiry
300-399	Maintenance
400-499	Update
500-699	Basic analysis
700-899	Monthly/yearly analysis
900-999	One-time program

  RPG III program names are 6 characters (such as MLG105). CL, DFU, and Query program names are each 7 characters (such as MLG035C, MLG315U, and MLG910Q respectively).

- Display file names are the name of the program with which they are associated plus D (such as MLG105D or MLG035CD).

System Requirements

The application requires that you have any model of the System/38 plus the following:

- IBM 5211, 3262, or 3203 Printer
- IBM 5251 Display Station Model 11 or 12, or equivalent
- Offline device (such as an IBM 5280 Distributed Data System) to create files on diskettes
- Program products:
  - Control Program Facility (Program 5714-SS1)
  - RPG III (Program 5714-RG1)
  - Interactive Data Base Utilities (Program 5714-UT1)
- Disk storage sufficient for:
  - The machine product
  - The program product
  - Objects created for the application
  - Data required for the files

For information on describing the devices to the system and installing the program products, refer to the *Guide to Program Product Installation and Device Configuration.*

Additional Considerations

For the examples in this publication, programs are often simplified by using the system defaults for file and library names, queues, and system functions. For an explanation of the defaults, refer to the appropriate command in the *Control Language Reference Manual*.

SUMMARY OF APPLICATION APPROACHES

Each application *approach* in this publication provides a different example of how to use a combination of System/38 functions to implement the mailing list application. The examples begin with a simple batch approach that uses a basic set of System/38 functions and progress to sophisticated batch and interactive approaches that use most of the System/38 functions. Each new approach builds on the previous approaches.

You can use these approaches as an effective learning tool by actually following through the steps of creating the example files and programs on your system. You can also observe the results of the application approaches by executing them on your system, using actual or sample data (names, addresses, account numbers, and so on) of your own.

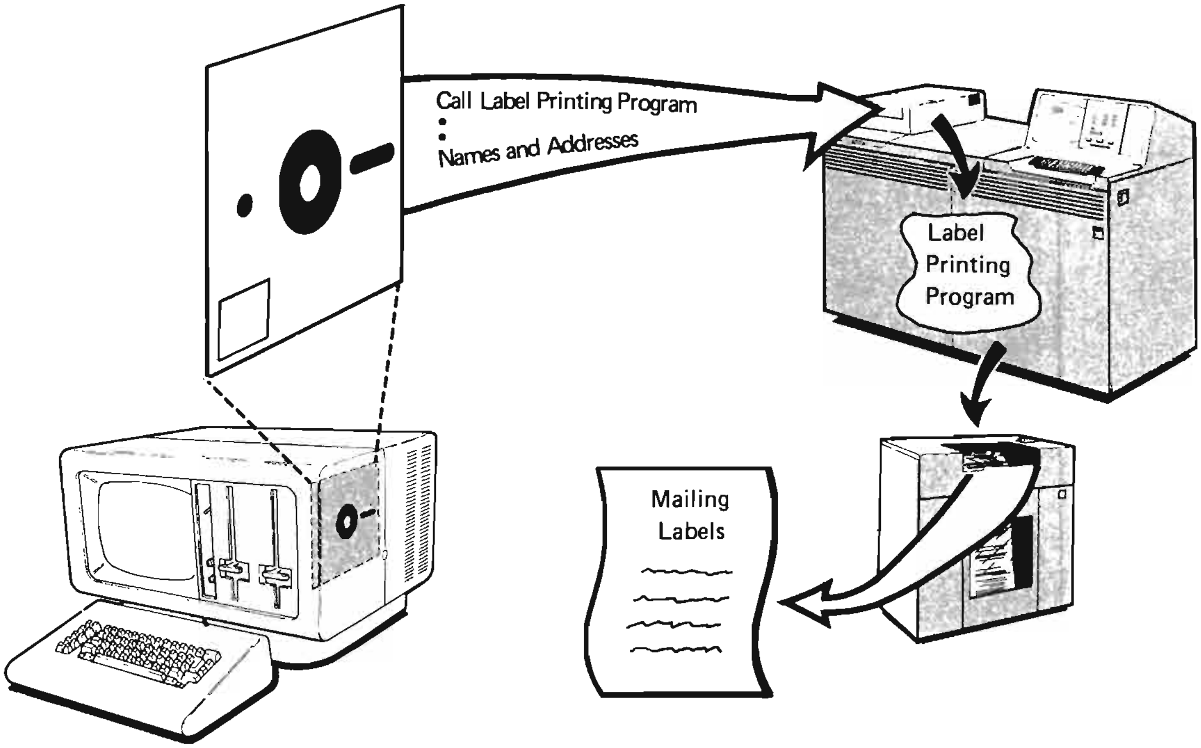
The approaches in Chapters 3 and 4 require an offline device, such as the 5280 Distributed Data System, to enter the instructions and data on diskette. If such an offline device is not readily available, you may want to start at Chapter 5; however, Chapters 2 through 4 should be read first because they introduce concepts used in later chapters. The procedures and approaches described in Chapters 5 through 8 can all be done at a work station.

Chapters 9 through 11 describe approaches that use advanced techniques. You can also execute these approaches on your system if you choose. To execute the approach in Chapter 9, you will need an offline device to enter the data on diskettes. The approaches in Chapters 10 and 11 are executed at a work station.

The following summarizes the approaches and how they relate.

Simple Batch Input (Chapter 3)

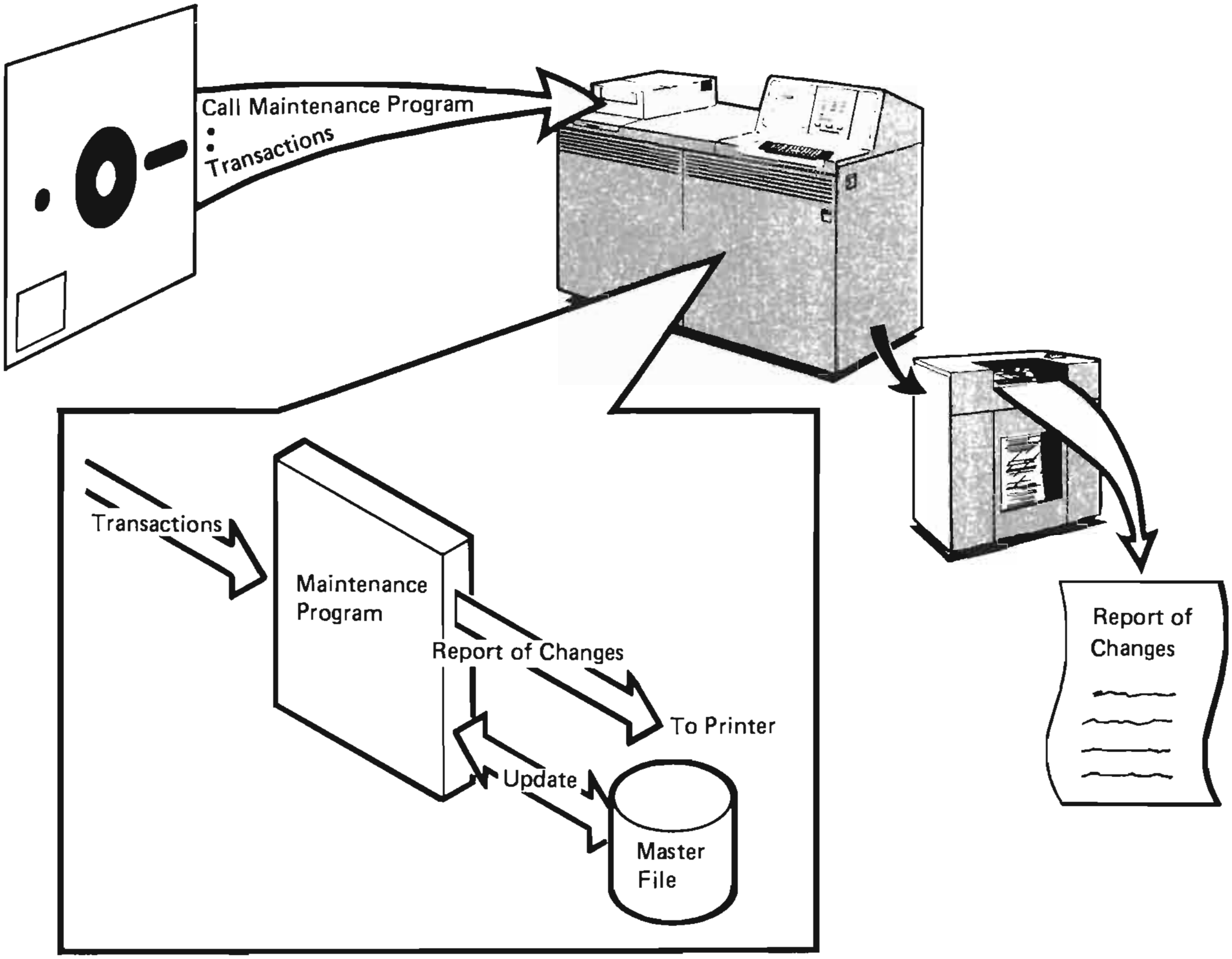
In this first approach, master records (names and addresses) are keyed onto diskette at an offline device. These master records are then read from diskette into the system, and an RPG III label printing program prints the records as mailing labels.



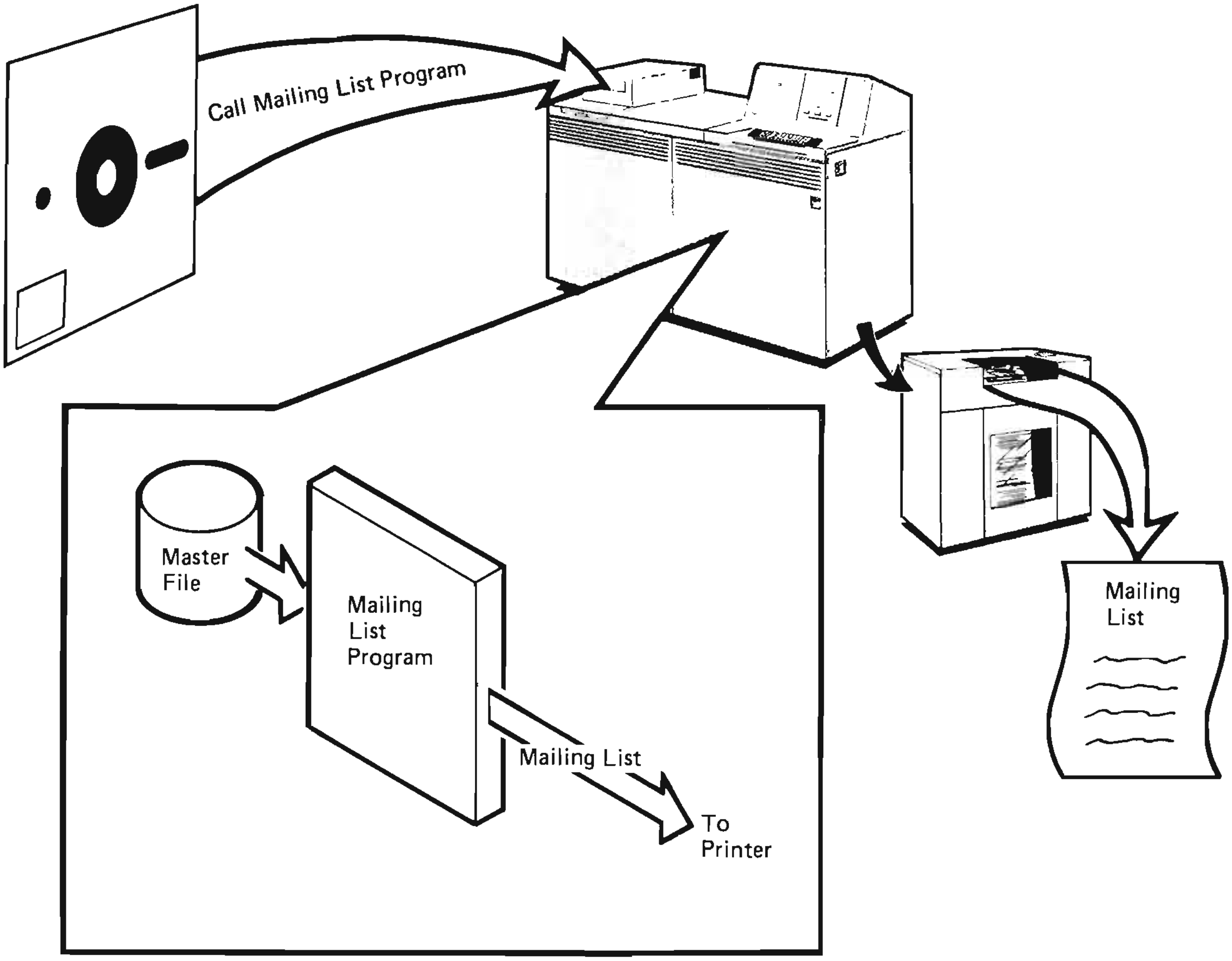
Chapter 2 describes the basic system functions that are used in this approach.

Batch Input and Batch Maintenance (Chapter 4)

This approach is an extension of the previous approach (Chapter 3). Instead of being read from diskette, the master records are stored in the system in a data base master file. Transaction records for updating the master file are keyed onto diskettes at an offline device. These transaction records are then read from diskette, and an RPG III maintenance program updates the master file with the transaction records. The maintenance program also produces a report of changes to the master file.



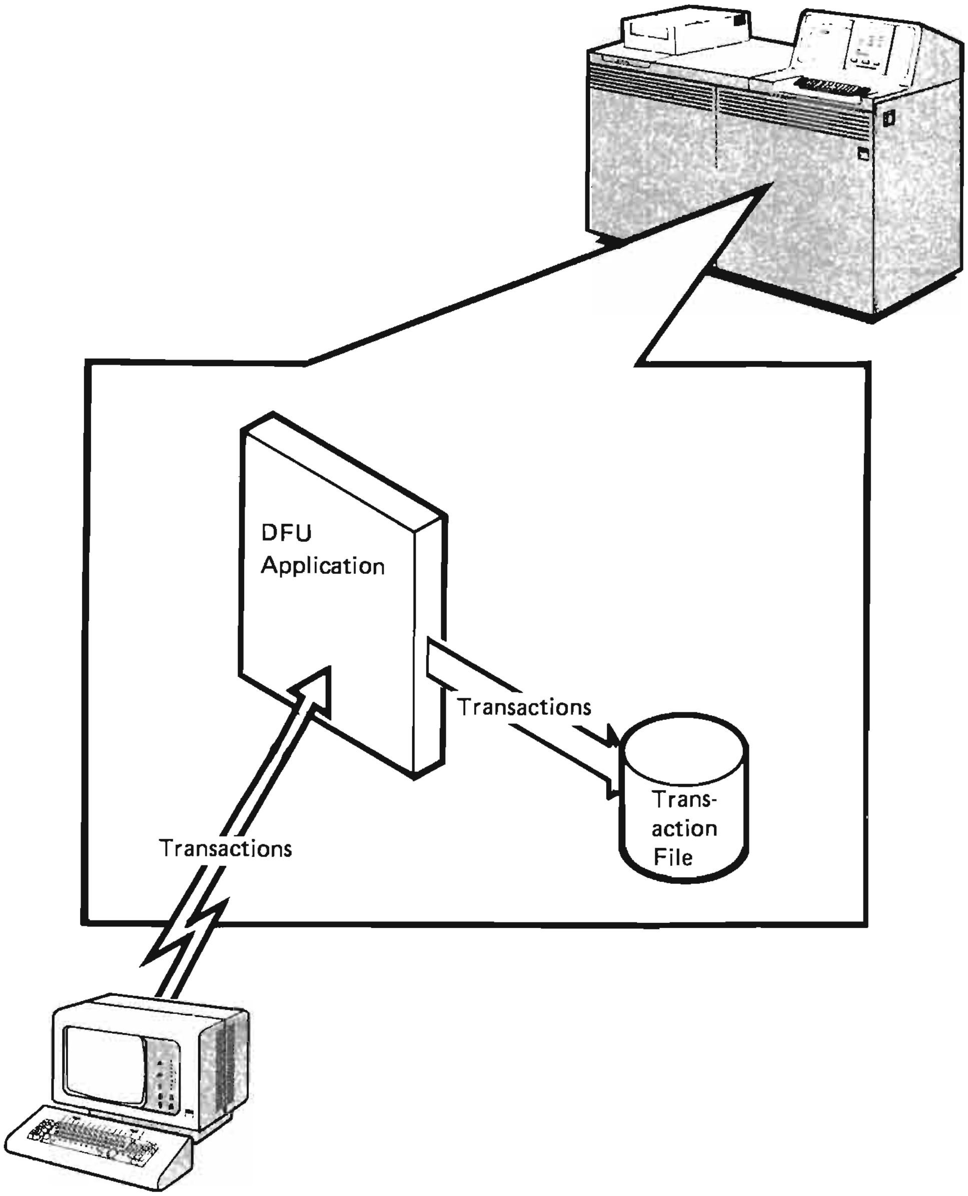
After the master file has been updated, an RPG III mailing list program produces a mailing list from the updated master records.



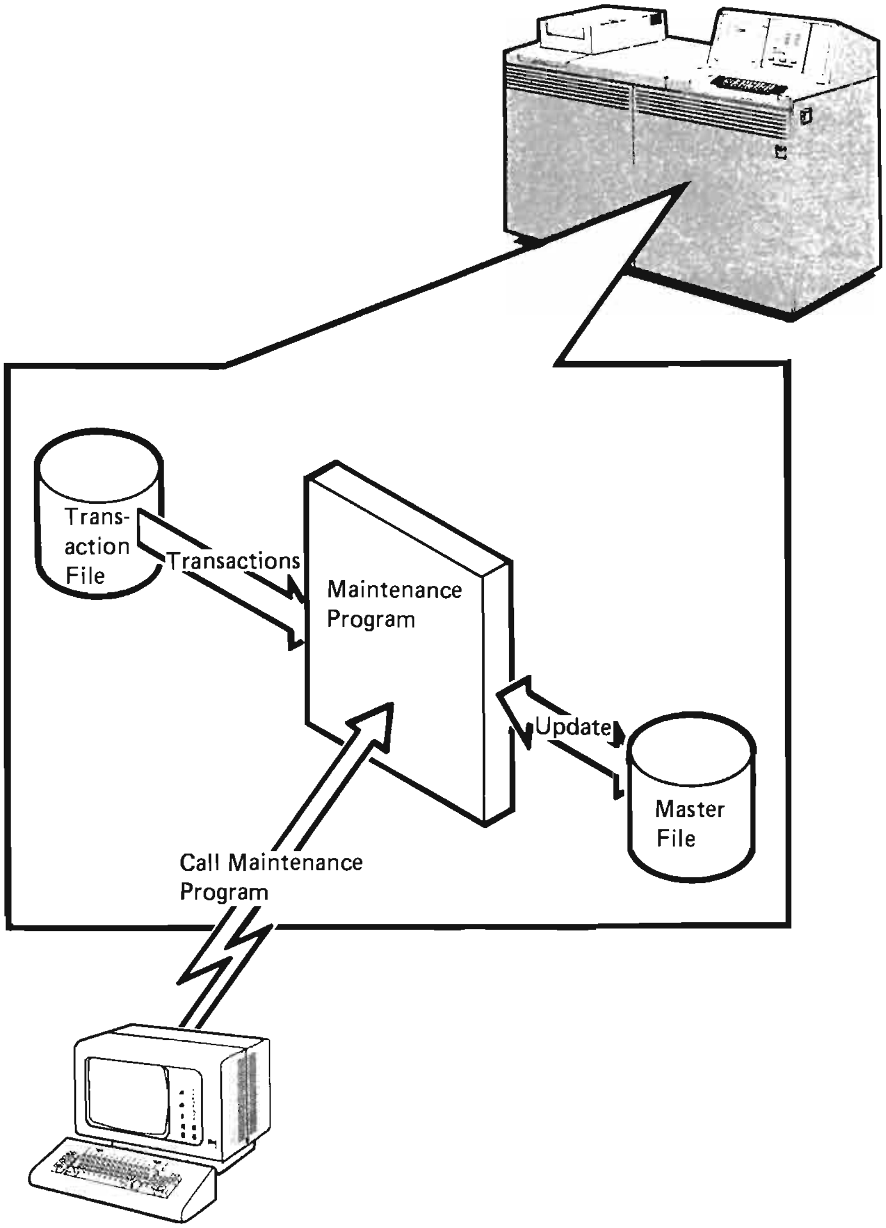
The RPG III programs and the master file are created from source that is read from diskette.

Interactive Input and Batch Maintenance (Chapter 7)

In the previous approach (Chapter 4), transactions to update a master file were read from diskette. In this approach, a work station user enters transactions interactively through a data file utility (DFU) application. The entered transactions are stored temporarily in a data base transaction file.



An RPG III maintenance program then applies the transaction records from the transaction file to the master records in the data base master file.

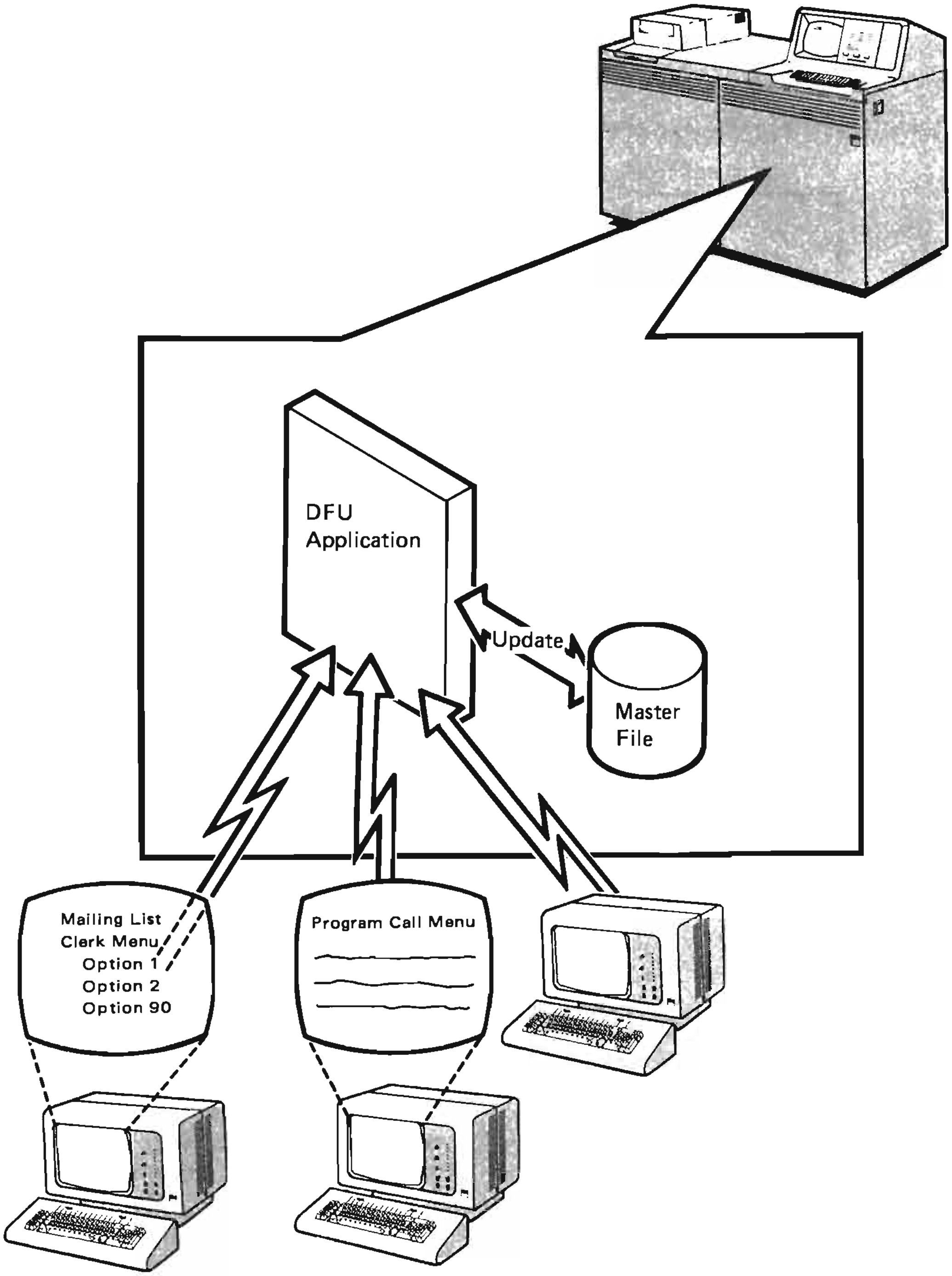


The maintenance program also produces a report of changes.

The DFU application, the RPG program, and the data base files are all created from a work station in this approach. The discussion of this approach assumes that you have previously read Chapters 5 and 6, which describe how to use the programmer menu and the source entry utility (SEU) to create files, libraries, and programs.

Interactive Input and Interactive Maintenance (Chapter 8)

In the previous approach (Chapter 7), a work station user interactively entered transactions into a transaction file, and the batch of transactions was then applied to the master file by a maintenance program. This approach allows multiple work station users to interactively update the master file itself through a DFU application.



Several ways of calling the DFU application for this approach are described:

- A work station user enters the control language command that directly calls the application.

- A work station user calls a control language program by entering the program name on a general user menu (the program call menu). The program then calls the DFU application.

- At sign-on, a work station user receives a specialized menu that is provided by a control language program and an associated display file. When the user selects a specific option on the menu, the control language program calls the DFU application.

Also included in this approach are two methods of making an inquiry into the master file:

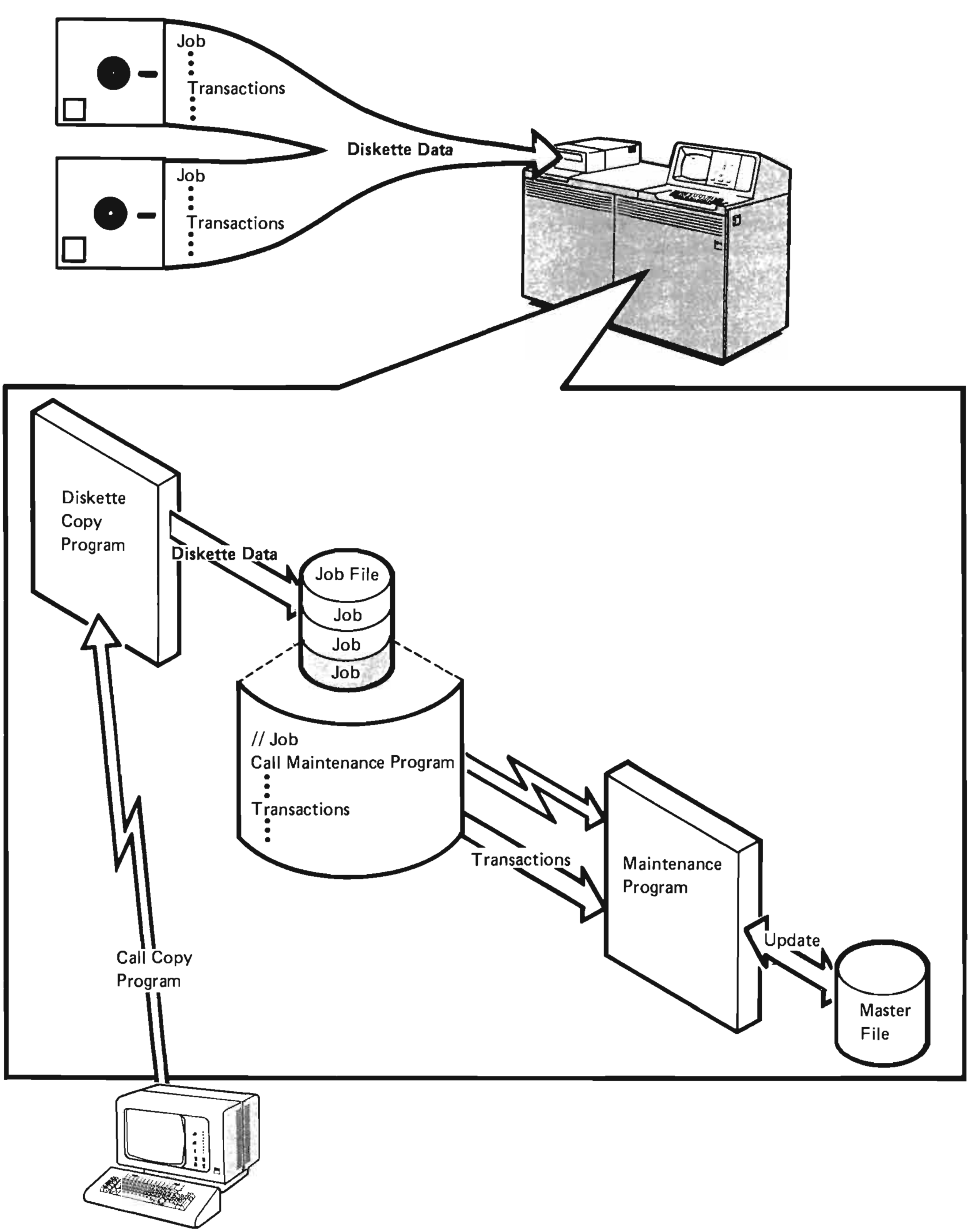
- Using a DFU application that examines specific fields of the master file through a separate access path.

- Using a query application that produces a report on specific fields of the master file.

Batch Maintenance of Multiple Diskette Batches (Chapter 9)

This approach extends the approach of Chapter 4 so that transactions on multiple diskettes can be handled in a single operation.

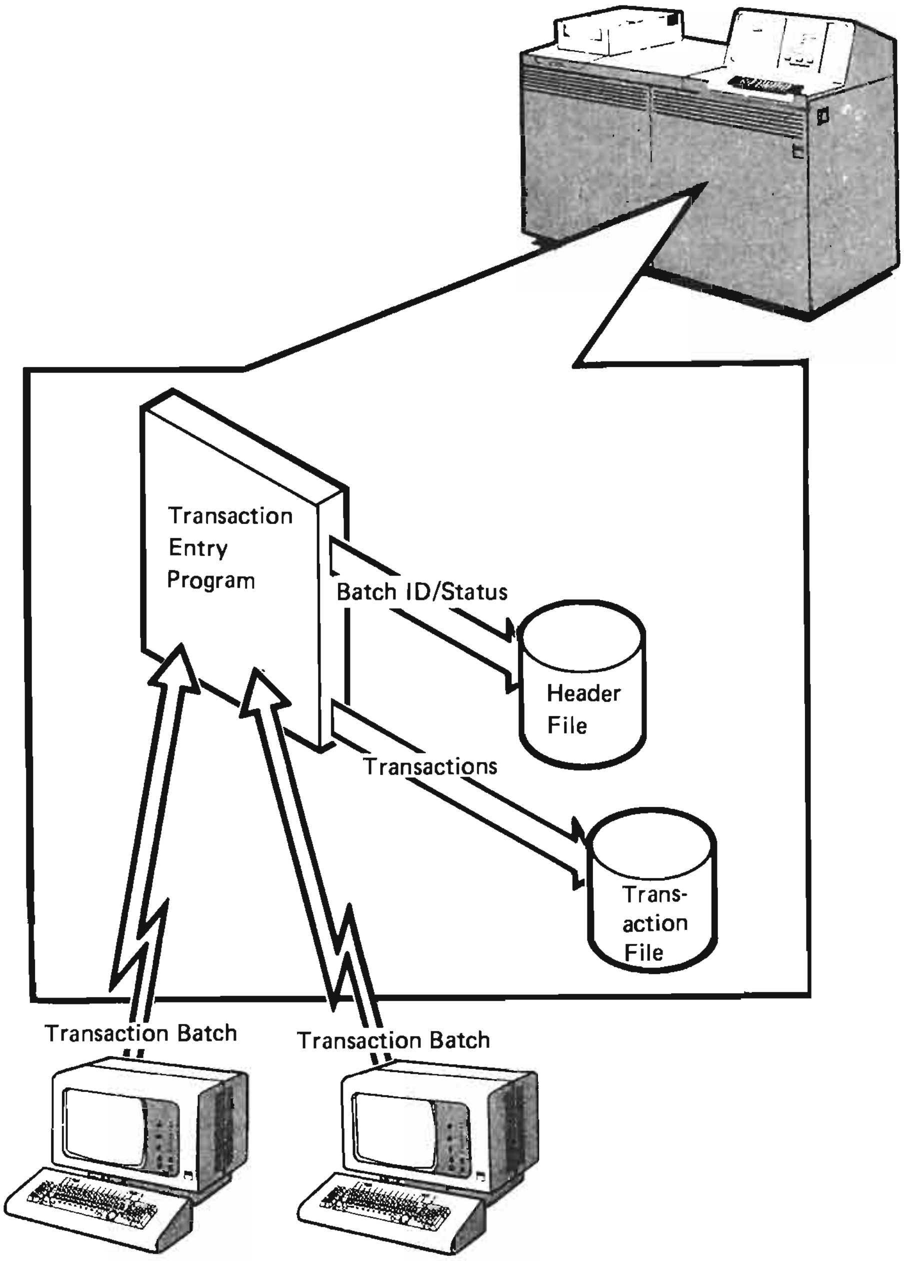
Batches of transactions are keyed onto diskettes at one or more offline devices. A control language program sequentially copies the data from each diskette into a data base file, and the resulting sequential arrangement of data in the file creates an input stream of batch jobs. The program then starts a spooling reader to read the input stream from the data base file. When each batch job in the input stream is executed, an RPG III maintenance program is called to apply the transaction records to a data base master file.


Job
Transactions
Diskette Data
Job
Transactions
Diskette Copy Program
Diskette Data
Job File
Job
Job
Job
// Job
Call Maintenance Program
Transactions
Transactions
Maintenance Program
Update
Master File
Call Copy Program

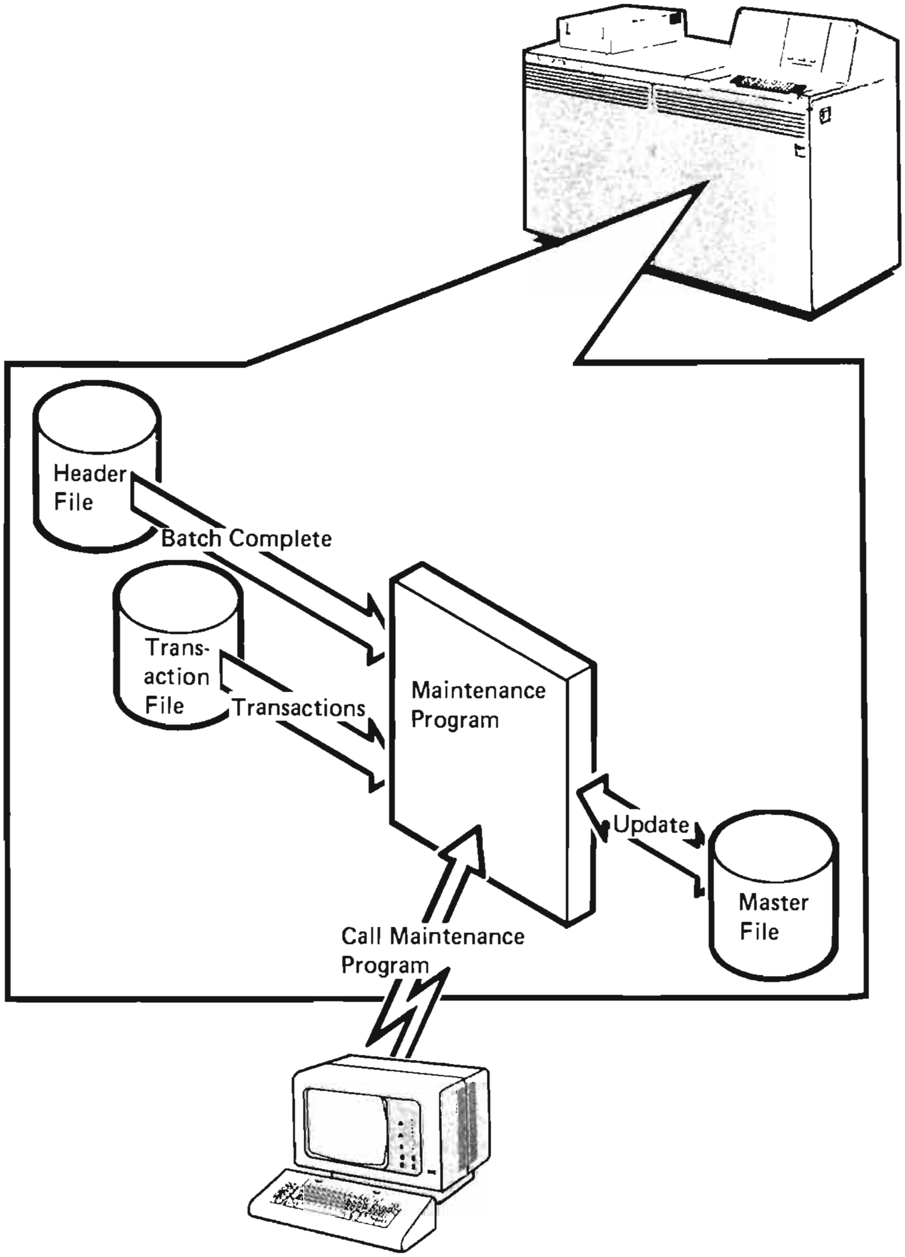
Batch Maintenance of Multiple Work Station Entries (Chapter 10)

In Chapter 7, a work station user entered transactions into a transaction file, and a maintenance program then applied the transactions to a master file. Only one work station user could enter transactions at a time. The approach in this chapter allows one or more work station users to enter new batches of transactions at the same time that completed batches are being processed.

An RPG III program and an associated display file provide the displays on which the work station users enter individual batches of transactions. The displays are formatted so that each work station user entering transactions provides a batch number (which identifies the batch) and indicates when the batch is complete. Header records that identify the batches of transactions and their status are stored in a batch header file. The transaction records being entered are stored in a transaction file. A logical file provides access to both the header information and the transaction records.



This approach assumes that a maintenance program similar to that in Chapter 7 will be used to apply the transaction records to the master file. Only batches whose header record indicates a status of complete are applied to the master file.

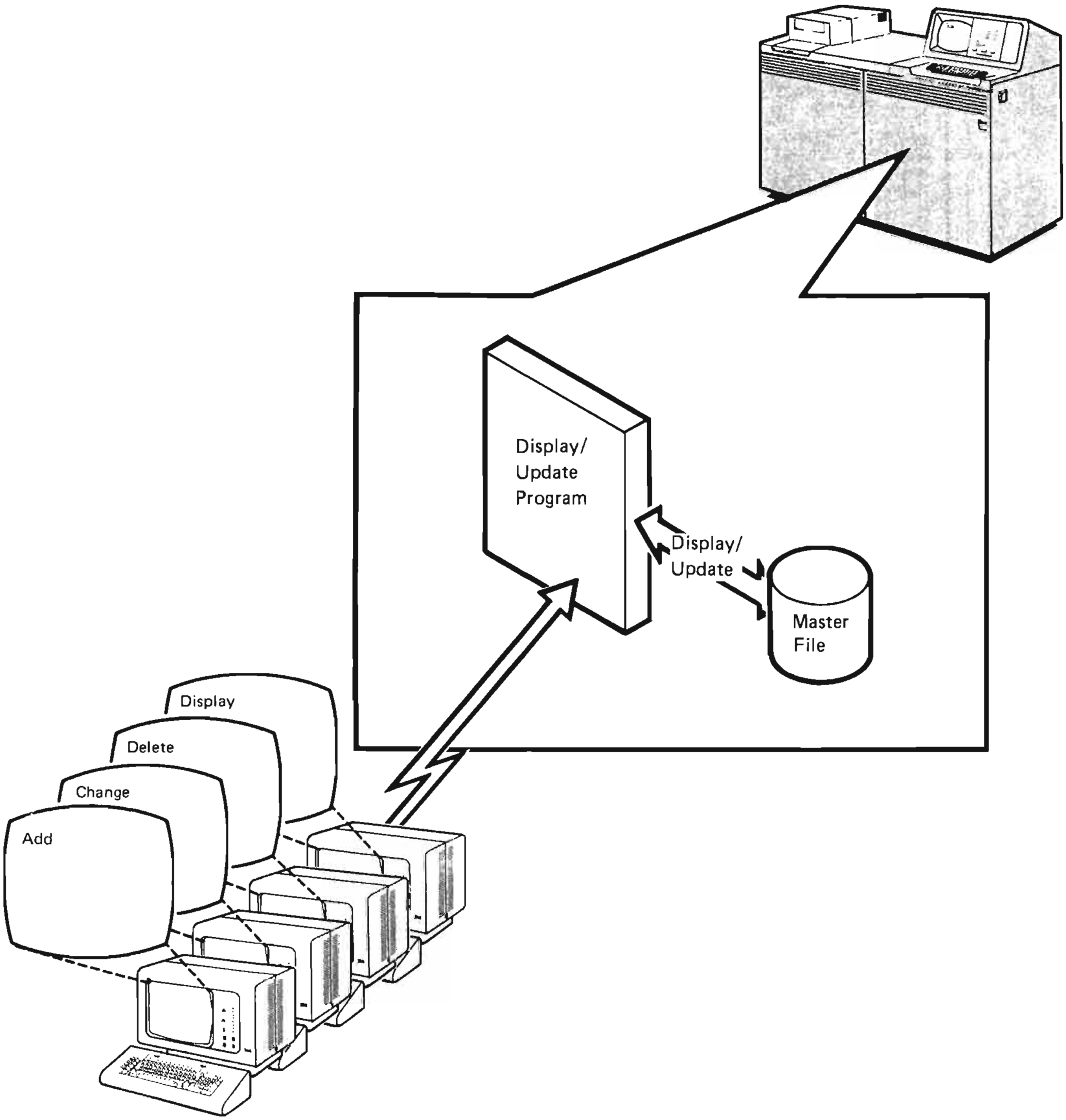


Interactive Maintenance from Multiple Work Stations (Chapter 11)

This approach is a more sophisticated version of the interactive maintenance approach in Chapter 8. Multiple work station users can:

- Add records to the master file.
- Change records in the master file.
- Delete records from the master file.
- Display records in the master file.

These functions are provided through an RPG III program and an associated display file.



The RPG III display/update program also produces a report of changes.

Chapter 2. Basic Functions and Operations

OVERVIEW

This chapter introduces some basic System/38 functions and operations that are used or assumed in the application approaches. Topics discussed include:

- Basic functions
  - Interactive and batch jobs
  - Input/output spooling
  - Subsystems
  - User profiles and passwords

- System installation

- Preparing for daily operations

- General recovery considerations

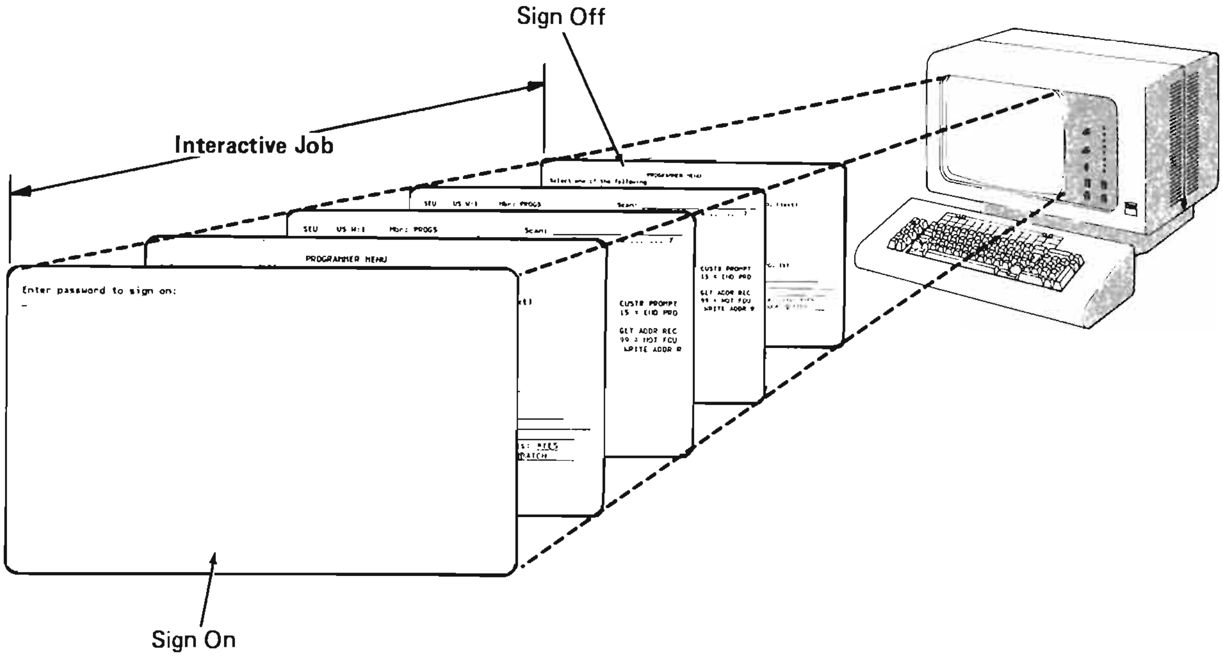
BASIC FUNCTIONS

System/38 can support a range of environments from one consisting almost entirely of batch processing to one that makes extensive use of interactive processing through work station applications. System/38 allows you to select and combine techniques from various approaches. The basic functions of System/38 to be used in the following chapters are introduced here. More detailed discussions of these concepts are in the *CPF Concepts Manual*.

Interactive Jobs

An *interactive job* is a job in which the processing actions are performed in response to input provided by a work station user. During the job, a dialog exists between the user and the system.

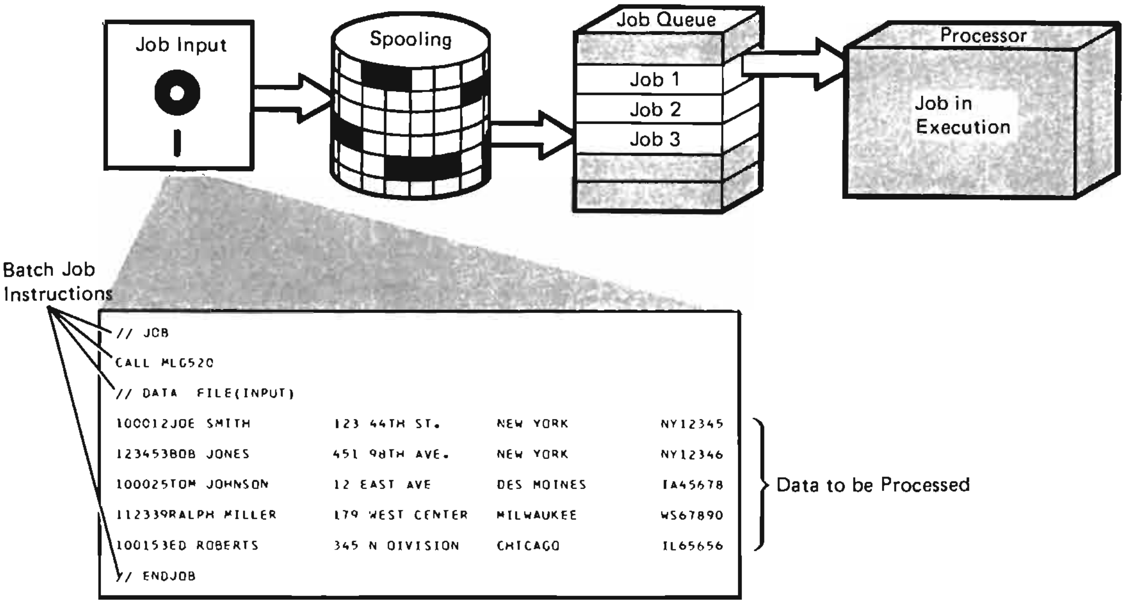
An interactive job starts when you enter your password at the work station or system console and ends when you sign off. Commands or application functions can be entered during an interactive job. Menus, prompts, and messages can assist you in interactively responding to the System/38.



Batch Jobs

A *batch job* is a group of processing actions submitted as a predefined series of actions, such as commands, to be performed without a dialog between the user and the system.

A batch job is first placed on a job queue and then selected for execution by the CPF. One method of placing jobs on a job queue is spooling.



```
// JOB
CALL MLG520
// DATA  FILE(INPUT)
100012JOE SMITH        123 44TH ST.      NEW YORK       NY12345
123453BOB JONES        451 98TH AVE.     NEW YORK       NY12346
100025TOM JOHNSON      12 EAST AVE       DES MOINES     IA45678
112339RALPH MILLER     179 WEST CENTER   MILWAUKEE      WS67890
100153ED ROBERTS       345 N DIVISION    CHICAGO        IL65656
// ENDJOB
```

Spooling

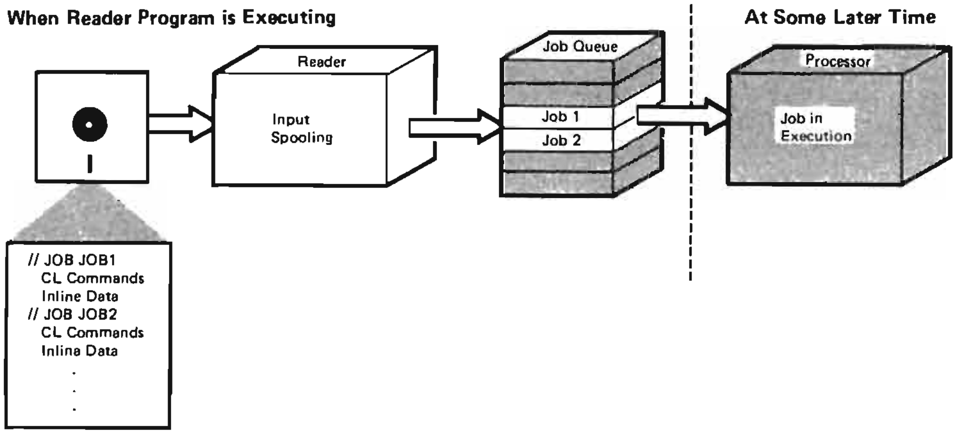
System/38 provides both *input spooling* and *output spooling.* Input spooling takes the information from an input device and places jobs on an input job queue. Output spooling allows programs to write to a spooled output file rather than directly to an output device. This spooled output file is placed on an output queue and written to an output device at a later time.

For input spooling, a CPF program called a *reader* transfers jobs from an input device to a job queue. The jobs read by the reader are an *input stream.* An input stream is a group of records submitted as batch input that contains control language (CL) commands for one or more jobs and, optionally, the data records of one or more *inline data files.* An inline data file is a data file included with a job when the job is read from an input device by a reader program.

All printed output is normally spooled. Spooling can allow multiple print jobs to execute simultaneously even if all of the jobs use the same printer. A CPF program, called a *writer,* transfers spooled output files from an output queue to an output device, such as a printer.

Job Queues

Batch jobs submitted to the system for processing are placed on a *job queue*, which is an ordered list of jobs in storage waiting to be executed. A typical way to place jobs on the queue is to have the system operator start a reader, such as a diskette reader. The default job queue for a start reader command is QBATCH. As shown in the following illustration, each job on a diskette file becomes a separate entity on a job queue. Inline data, if present, is stored in spooled *inline data files*, which are created automatically by the spooling function. Jobs are selected from the job queue for execution and, after execution, the associated spooled input files are automatically deleted.



Printed Output

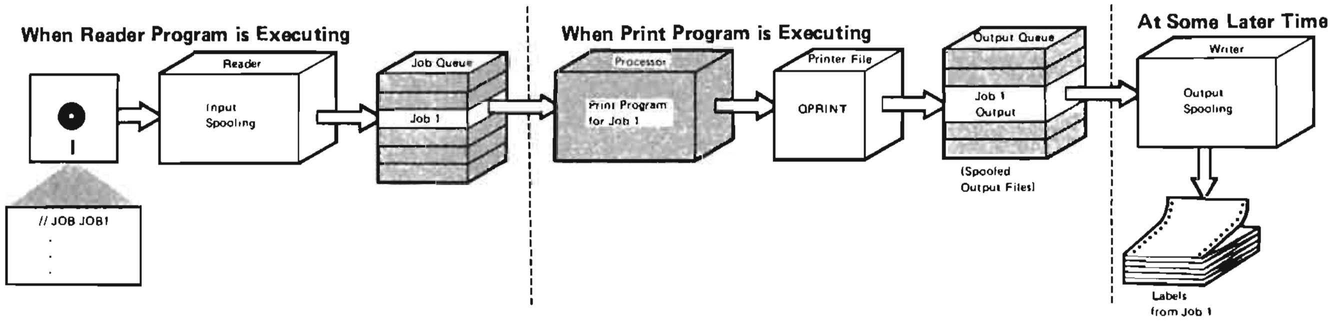
The several types of System/38 printed output include the following:

- Output from a printer file in an RPG III application program
- Output from a system function, such as the source listing of the RPG III compiler
- Output from CPF, such as the *job log*, which lists commands executed as part of the job and the associated system responses

Printed output is directed to a printer device file, which connects the program to a *spooled output file*. The spooled output file derives its name and attributes from the device file. A spooled output file contains output records that are waiting to be printed. For each spooled output file, an entry is placed on an *output queue*.

For this application example, the default printer device file and the default output queue are IBM-supplied and both are named QPRINT.

The spooled output files are stored until printed. After being printed, the files are automatically deleted.



Subsystems

Subsystems provide an operating environment for work performed within them. A job must be assigned to an active subsystem before any processing for the job can take place. A group of jobs that have common characteristics can be controlled independently of other jobs if the jobs are assigned to the same subsystem. All subsystems have the same basic capabilities, but they may be defined to operate on a specific type of work. Four of the IBM-supplied subsystems, shown on the following page, are used in the application approaches in this publication. These subsystems are as follows:

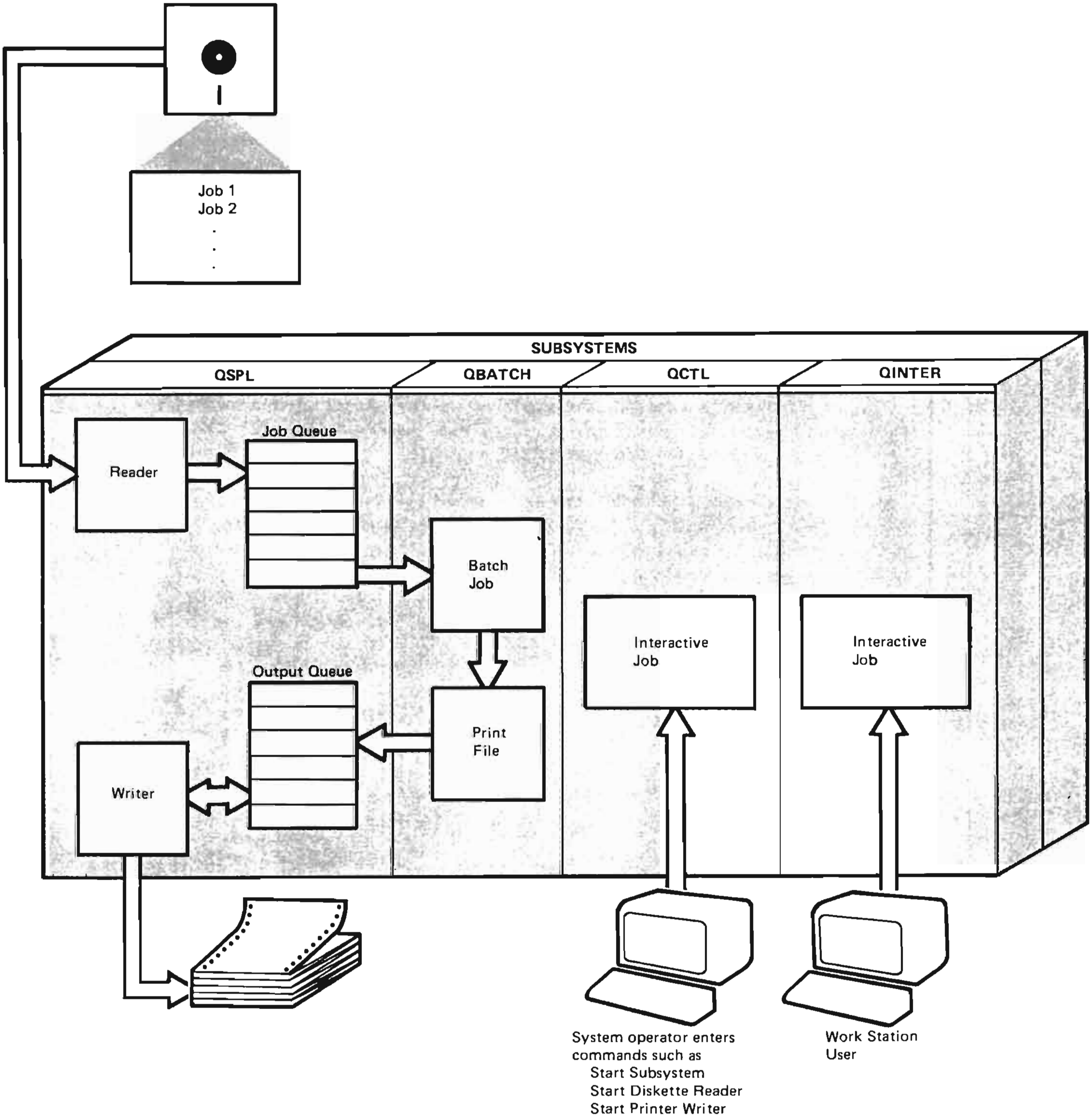
- QSPL is for both input and output spooling operations and is usually started by the system operator. The system operator must also start readers and writers that operate in this subsystem.

- QBATCH is for batch operations and is usually started by the system operator. The job queue on which jobs are to be placed for processing in this subsystem is also named QBATCH.

- QCTL is for interactive work and is the controlling subsystem. It is started automatically when the system is powered on. When only QCTL has been started, the system console is the only device that you can use.

- QINTER is for interactive work and is usually started by the system operator. Normally, work stations such as the 5250 series of display stations are used with this subsystem.

The illustration shown here is a simplification of subsystem usage. Although not shown in this illustration, interactive jobs can also place jobs on a job queue and place output files on an output queue.



User Profiles and Passwords

System/38 supports various security functions to help protect the system against improper use. These functions are based on *user profiles* and *passwords*. A user profile is an object that represents a particular user or group of users to the CPF and identifies which objects and commands the user is authorized to use. CPF is shipped with several IBM-supplied user profiles, each of which has a password. A password is a name that a system user enters at the console or a work station to identify himself to the system.

The basic user profiles that are shipped with CPF and used in this publication are listed below:

System User	**User Profile Name**	**Password**
Programmer	QPGMR	PGMR
System Operator	QSYSOPR	SYSOPR
Work Station User	QUSER	USER
Security Officer	QSECOFR	SECOFR

Many users can operate under the same user profile. In addition to the user profiles listed above, the security officer can create user profiles for individuals or departments to allow better control.

The IBM-supplied profiles (including the passwords) can be modified and new profiles can be added by using the System/38 security functions. An example of adding a new profile is given in Chapter 8 under *Using a User Profile and an Application-Oriented Menu.*

INSTALLING YOUR SYSTEM

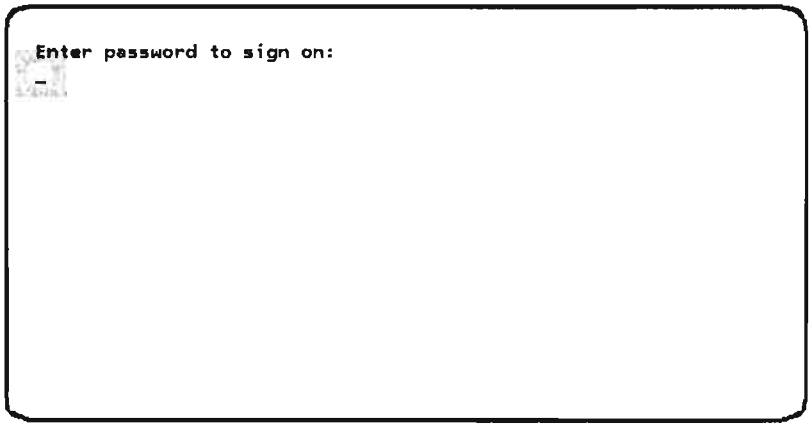
To use System/38, you must install the program products and describe the hardware configuration to the system. The procedure to do this is given in the *Guide to Program Product Installation and Device Configuration.*

System/38 offers a variety of options for specializing your system. For the mailing list application in this publication, none of these options are selected; only the basic requirements as described in the *Guide to Program Product Installation and Device Configuration* are selected.

PREPARING FOR DAILY OPERATIONS

The application approaches discussed in this publication assume that you or your system operator have prepared the system for daily operations. The procedures involved are briefly discussed here. For a complete discussion of these procedures, refer to the *System/38 Operator's Guide.*

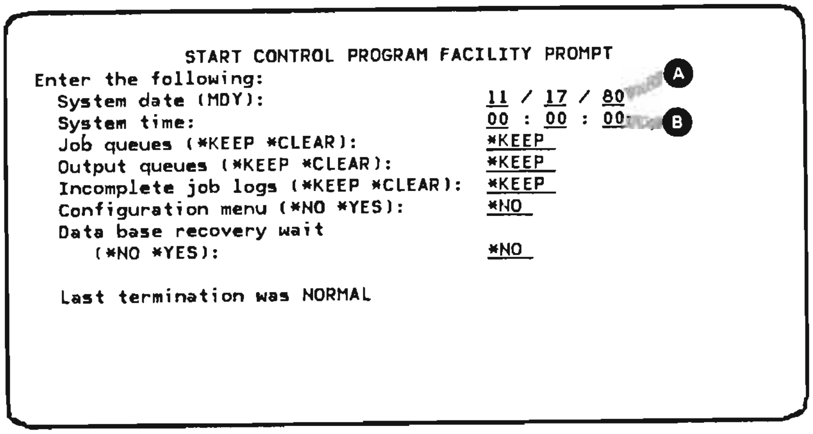
The system is powered on from the operator/service panel. Powering on causes the CPF start-up process to begin. The sign-on display appears first on the system console screen. The cursor is automatically positioned for you to enter a password:



```
Enter password to sign on:
_
```

Note that your password does not appear on the screen as you key it in (so that no one can see the password you are using).

Normally, the system operator's password, SYSOPR, is entered. Next, the start control program facility prompt appears:



```
                START CONTROL PROGRAM FACILITY PROMPT
Enter the following:
  System date (MDY):                    11 / 17 / 80
  System time:                          00 : 00 : 00
  Job queues (*KEEP *CLEAR):            *KEEP
  Output queues (*KEEP *CLEAR):         *KEEP
  Incomplete job logs (*KEEP *CLEAR):   *KEEP
  Configuration menu (*NO *YES):        *NO
  Data base recovery wait
     (*NO *YES):                        *NO

  Last termination was NORMAL
```

A Verify that this is the current date.

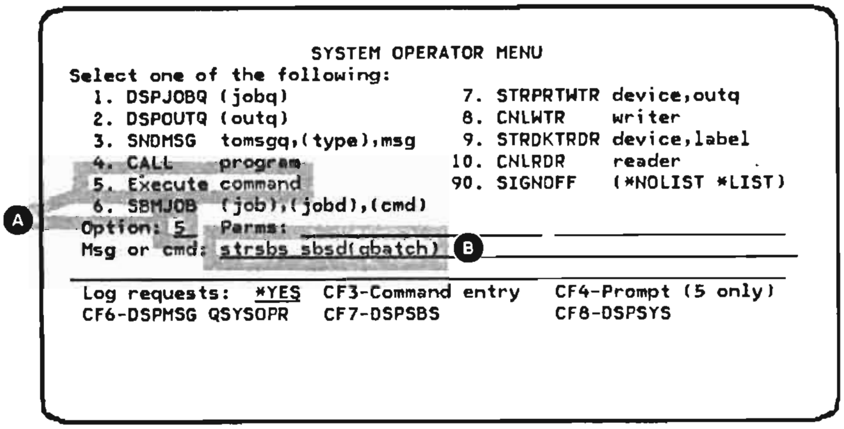
B Enter the current time.

The system uses the defaults unless you change them. Normally, you just verify the date and enter the current time.

During the CPF start-up, the QSYSOPR (system operator message queue) display may appear one or more times on the system console screen to show messages regarding the status of the system and system equipment.

If the system operator password was entered on the sign-on prompt, as in this application example, the system operator menu is displayed when the CPF start-up process is completed. The menu allows you to perform various operations by selecting one of the options listed on it. In this application example, you select the option that allows you to execute a command.

Because you will need to use the QBATCH subsystem, you enter the Start Subsystem (STRSBS) command:



```
                         SYSTEM OPERATOR MENU
Select one of the following:
  1. DSPJOBQ (jobq)                  7. STRPRTWTR device,outq
  2. DSPOUTQ (outq)                  8. CNLWTR    writer
  3. SNDMSG  tomsgq,(type),msg       9. STRDKTRDR device,label
  4. CALL    program                10. CNLRDR    reader
  5. Execute command                90. SIGNOFF   (*NOLIST *LIST)
  6. SBMJOB  (job),(jobd),(cmd)
 Option: 5   Parms: ____________________ ________________
 Msg or cmd: strsbs sbsd(qbatch) ____________________________

 Log requests:  *YES  CF3-Command entry   CF4-Prompt (5 only)
 CF6-DSPMSG QSYSOPR   CF7-DSPSBS          CF8-DSPSYS
```

A Specify option 5 to execute a command.

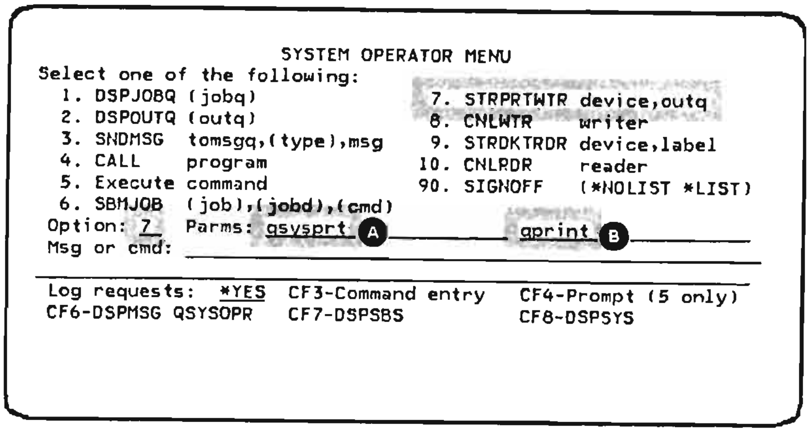
B Enter the command here.

Note that the alphabetic characters you enter appear as lowercase because the keyboard is normally in lower shift. The system will accept either uppercase or lowercase for most command functions.

Other subsystems you will need to use are QSPL and QINTER. When the system operator menu is displayed again with the option field blank, you use option 5 again to separately enter the following commands:

```
strsbs sbsd(qspl)
strsbs sbsd(qinter)
```

A printer writer also needs to be started to the device QSYSPRT from the output queue QPRINT. You enter the Start Printer Writer (STRPRTWTR) command by selecting option 7:



```
                      SYSTEM OPERATOR MENU
Select one of the following:
  1. DSPJOBQ (jobq)                    7. STRPRTWTR device,outq
  2. DSPOUTQ (outq)                    8. CNLWTR    writer
  3. SNDMSG  tomsgq,(type),msg         9. STRDKTRDR device,label
  4. CALL    program                  10. CNLRDR    reader
  5. Execute command                  90. SIGNOFF   (*NOLIST *LIST)
  6. SBMJOB  (job),(jobd),(cmd)
 Option: 7  Parms: qsysprt (A)___________qprint (B)___________
 Msg or cmd: ______________________________________________________
____________________________________________________________________
 Log requests:  *YES  CF3-Command entry    CF4-Prompt (5 only)
 CF6-DSPMSG QSYSOPR   CF7-DSPSBS           CF8-DSPSYS
```

A Specify here the name of the device on which the output is to be printed. This device name is also used as the name of the writer.

B Specify here the name of the output queue from which the output is to be printed.

Now your system is ready for the approaches described in this publication.

During the day, you may want to check on the status of a particular job or a spooled output file from a job. The commands and displays you can use to determine the status of jobs and spooled output files, or to change their status, are described in detail in the *System/38 Operator's Guide.* If you have a problem with the operation of the system, you may want to refer to the *System/38 Problem Determination Guide.*

When your system is ready to be powered down at the end of the day, select option 5 on the system operator menu and enter the Power Down System (PWRDWNSYS) command. This will properly terminate the system and turn off the power.

GENERAL RECOVERY CONSIDERATIONS

Recovery must be considered in any application design. 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 executing 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, executing a program, or correcting a program.

- *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 which is in main storage at the time of the failure will be lost. To avoid losing updates to data base files, the FRCRATIO parameter may be specified on the Create Physical File, Create Source Physical File, Create Logical File, and Override with Data Base File commands; however, there is some performance degradation when a value is specified for FRCRATIO. System/38 has some built-in recovery functions which, for example, properly close files and ensure that the data for the file is in auxiliary storage. To recover from a system failure, you must either restart or reprocess what was happening when your system failed. If a system failure occurs when you are not executing an application (such as when no programs are executing), no recovery is normally needed.

- *Damage* is the condition of any object that can no longer be processed by the system (such as when a disk 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, or restart the system, or install the system again. To recover a damaged user-defined object, you normally restore a saved version or create the object again.

The basic System/38 recovery support is the Save/Restore functions, which allow you to save backup versions of current objects on diskette or tape and to restore these objects as needed. Backup is essential to any application approach. How often you save your objects and which objects you choose to save varies with the application. There is no general recovery approach that is best for all applications. You should consider various trade-offs (such as the cost to frequently save objects versus reprocessing time) and risks (such as the probability of power interruption and fire) when developing your recovery plan.

There are comments on recovery throughout the remainder of this publication. These comments are not intended to be complete approaches to a recovery situation, nor are they the only approach to a particular recovery situation.

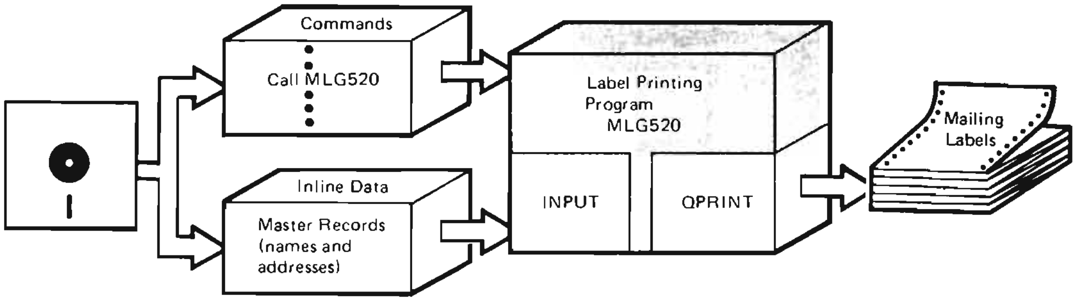
Chapter 3. Simple Batch Input

OVERVIEW

In this first approach to the mailing list application:

- An input stream consisting of job instructions (CL commands) and data (master records) is entered on diskette at an offline device.
- The input stream is read from diskette and placed in system storage. An inline data file is created to store the master records.
- A CALL command within the job instructions invokes an RPG III label printing program.
- The program reads the inline data file containing the master records and prints the master records as mailing labels.

The label printing program used in this approach is also created from an input stream that is read into the system from diskette.


Commands
Call MLG520
Inline Data
Master Records (names and addresses)
Label Printing Program MLG520
INPUT
QPRINT
Mailing Labels

The records are printed in the same sequence as they are read. The master records are not stored or updated on the System/38 in this approach. This approach is used to introduce and illustrate some System/38 functions, even though it may not meet the needs of many installations.

LABEL PRINTING PROGRAM

An RPG III label printing program named MLG520 is used to read the records from the inline data file, INPUT, and print a mailing label for each record using the printer device file, QPRINT. The program can execute with either spooled or nonspooled input or output data. Generally, the program is not aware of where the input data comes from or where the output data is placed. The program specifies the files INPUT and QPRINT so that:

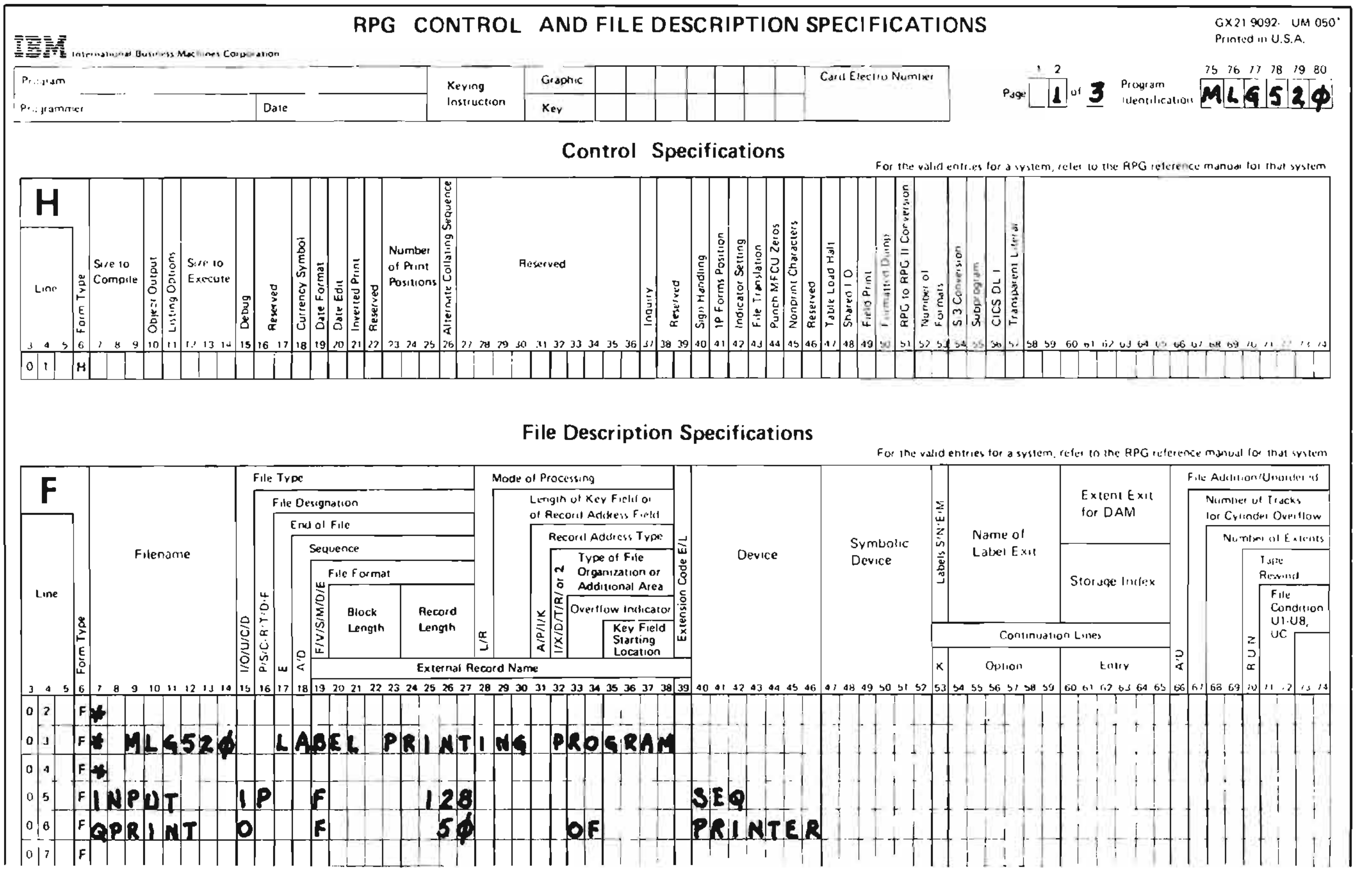
- An input file (INPUT) will normally come from an inline spooled data file (identified by the // DATA statement in the input stream).

- Output is normally written to a spooled output file (QPRINT), which is placed on the QPRINT output queue for later writing to a device.

Specifications for the Label Printing Program

The MLG520 program uses the RPG program cycle to read and print mailing list records.

The control and file description specifications for MLG520 are shown in Figure 3-1.



RPG CONTROL AND FILE DESCRIPTION SPECIFICATIONS

GX21-9092- UM 050*
Printed in U.S.A.

Page 1 of 3   Program Identification MLG520

Control Specifications

For the valid entries for a system, refer to the RPG reference manual for that system

```
  01 H
```

File Description Specifications

For the valid entries for a system, refer to the RPG reference manual for that system

```
  02 F*
  03 F* MLG520  LABEL PRINTING PROGRAM
  04 F*
  05 FINPUT   IP  F     128            SEQ
  06 FQPRINT  O   F      50     OF     PRINTER
  07 F
```

Figure 3-1. Control and File Description Specifications for MLG520 Label Printing Program

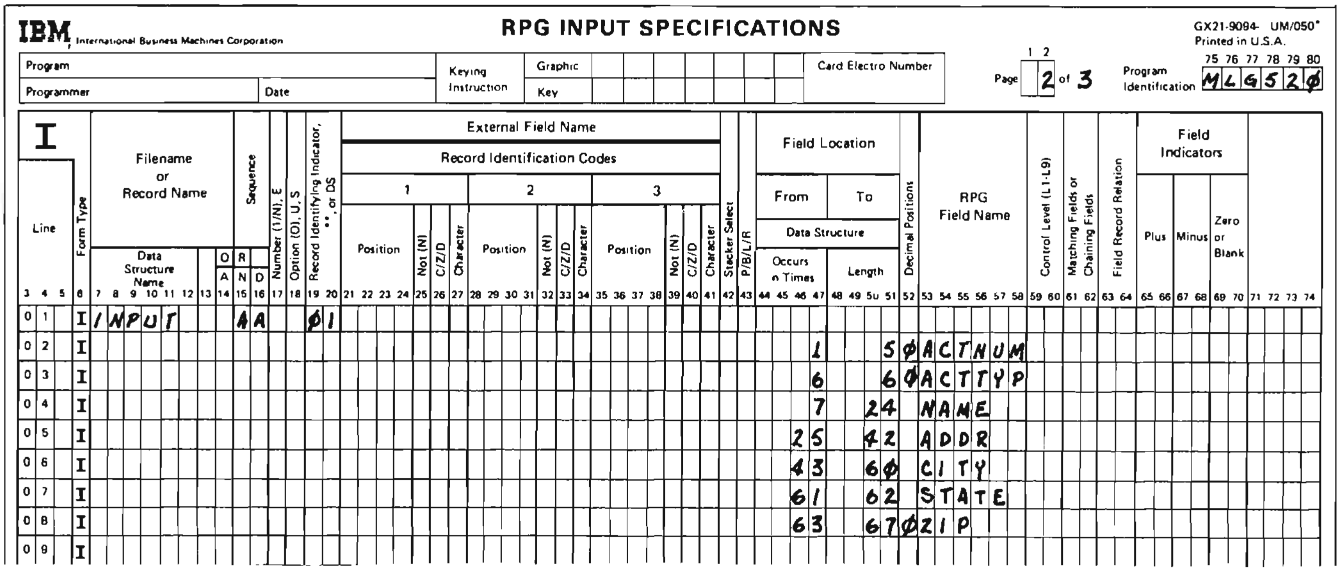
No entries are required in the control specifications. RPG III will default to standard values.

In the file description specifications, the first three lines are comments (* in position 7 designates a comment). The text in the second comment line identifies the source to the programmer.

The file names (positions 7 through 14) are the names of device files, data base files, or inline data files that are defined in the system. INPUT is the name of an inline data file, which is the master file. The diskette record length is specified as 128 bytes, which is the length of the basic data exchange format on diskette. QPRINT is the name of an IBM-supplied printer device file. Each spooled output file produced by the MLG520 program derives its name and attributes from this QPRINT device file.

The device names (positions 40 through 46) are used by the RPG III compiler to determine the valid language functions that can be specified for the file. The device names do not determine which device will be used. For example, the device name SEQ specifies a sequential device. This means the file is read or written without special device characteristics, such as space, skip, CHAIN, or SETLL. The device name PRINTER specifies a printing device, and entries for space or skip are valid.

The input specifications, shown in Figure 3-2, define the input fields for INPUT and identify the input record with indicator 01. The characters AA in positions 15 and 16 specify that the program is not to check the sequence of input records.



IBM International Business Machines Corporation

RPG INPUT SPECIFICATIONS

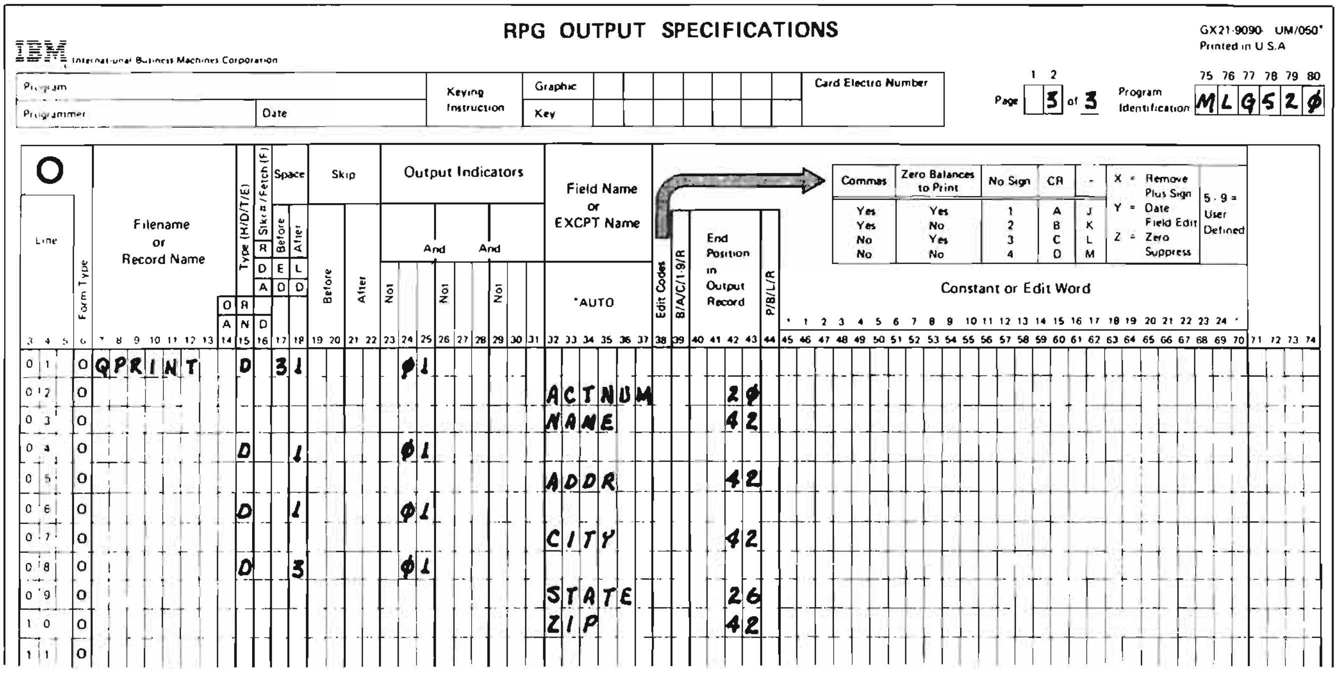
GX21-9094- UM/050*
Printed in U.S.A.

Program
Programmer
Date
Keying Instruction: Graphic / Key
Card Electro Number
Page 2 of 3
Program Identification MLG52Ø

Line	Form Type	Filename or Record Name	Sequence	Number (1/N), E	Option (O), U, S	Record Identifying Indicator, **, or DS	Field Location From	Field Location To	Decimal Positions	RPG Field Name
01	I	INPUT	AA			Ø1				
02	I						1	5	Ø	ACTNUM
03	I						6	6	Ø	ACTTYP
04	I						7	24		NAME
05	I						25	42		ADDR
06	I						43	6Ø		CITY
07	I						61	62		STATE
08	I						63	67	Ø	ZIP
09	I									

Figure 3-2. Input Specifications for MLG520 Label Printing Program

The output specifications, shown in Figure 3-3, define the four lines to be printed on each label. The format of the printed labels is shown in Figure 3-4.



IBM International Business Machines Corporation

RPG OUTPUT SPECIFICATIONS

GX21-9090 UM/050*
Printed in U.S.A.

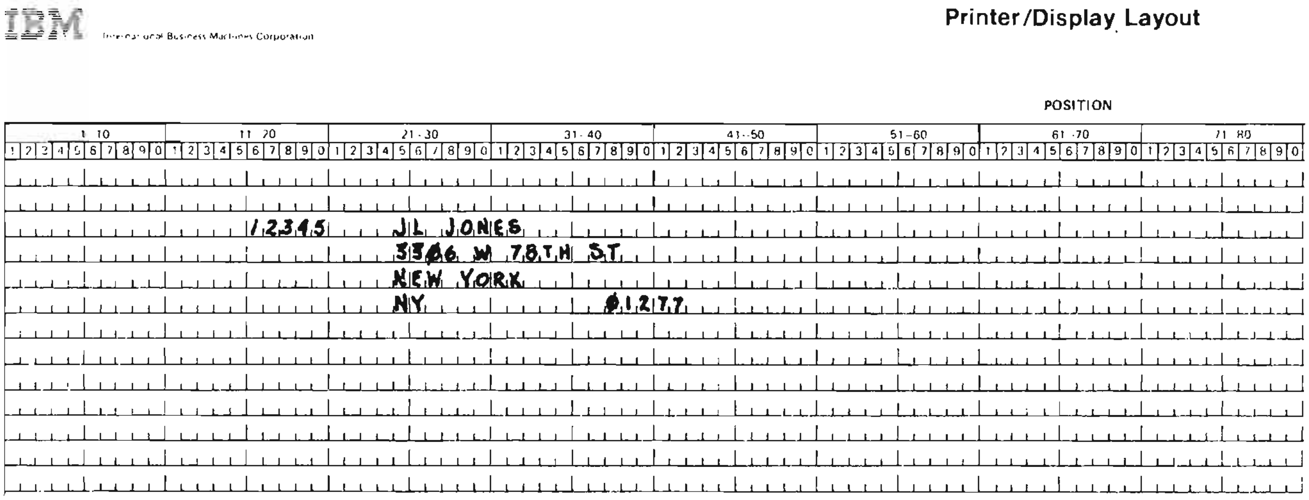
Program
Programmer Date
Keying Instruction: Graphic / Key
Card Electro Number

Page 3 of 3 Program Identification MLG52ø

Commas	Zero Balances to Print	No Sign	CR	-	X = Remove Plus Sign Y = Date Field Edit Z = Zero Suppress	5-9 = User Defined
Yes	Yes	1	A	J		
Yes	No	2	B	K		
No	Yes	3	C	L		
No	No	4	D	M		

Line	Form Type	Filename or Record Name	Type (H/D/T/E)	Space Before	Space After	Skip Before	Skip After	Output Indicators	Field Name or EXCPT Name	End Position in Output Record
01	O	QPRINT	D	3	1			ø1		
02	O								ACTNUM	2ø
03	O								NAME	42
04	O		D		1			ø1		
05	O								ADDR	42
06	O		D		1			ø1		
07	O								CITY	42
08	O		D		3			ø1		
09	O								STATE	26
10	O								ZIP	42
11	O									

Figure 3-3. Output Specifications for MLG520 Label Printing Program



IBM International Business Machines Corporation

Printer/Display Layout

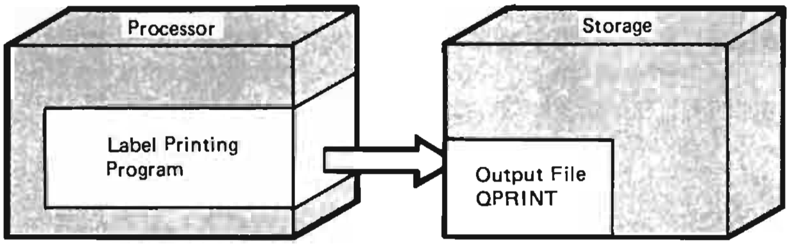
POSITION

```
         1-10      11-20     21-30     31-40     41-50
    1234567890123456789012345678901234567890123456789
01
02
03                 12345    JL JONES
04                          33ø6 W 78TH ST
05                          NEW YORK
06                          NY           ø1277
07
08
09
10
11
12
13
```

Figure 3-4. Format of a Mailing Label

Spacing entries cause the printer to move from one label to the next. The OF indicator is specified on the file description specifications (see Figure 3-1) even though no overflow line is specified on the output specifications. The OF indicator must be specified; otherwise, RPG causes a default overflow. The 01 indicator in positions 24 and 25 identifies the output record.

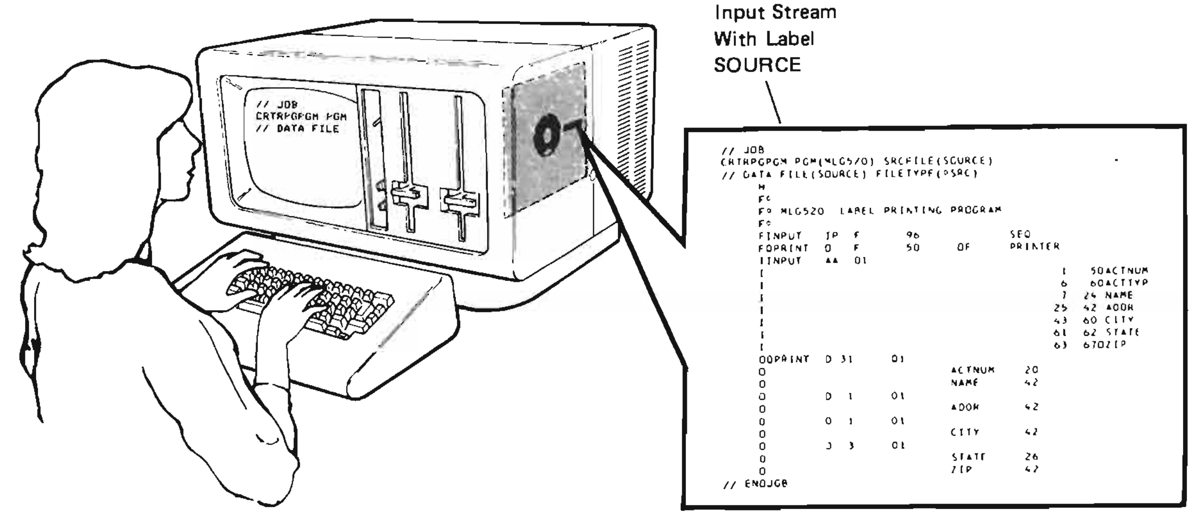
The output from the program will be written to the QPRINT spooled output file.



Creating the Label Printing Program

The label printing program MLG520 to be used in this approach must be created first. One way to create the program is from an input stream. An input stream is a group of records submitted to the system as batch input that contains CL commands and, optionally, inline data files for one or more jobs. An input stream will also be used to execute the program.

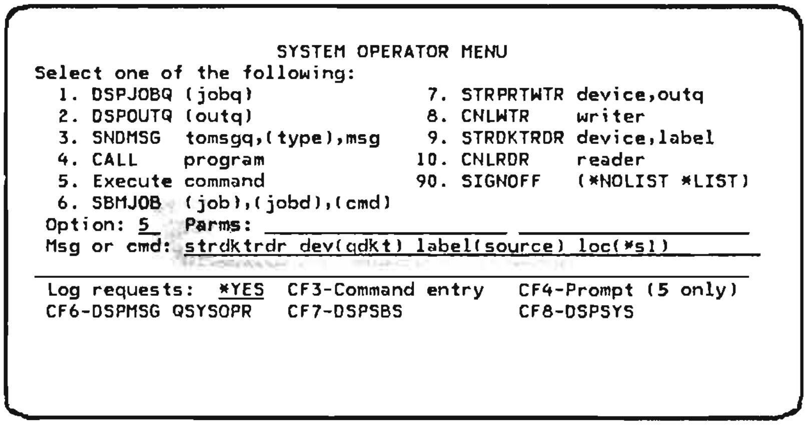
The input stream containing CL commands and the RPG source (Figures 3-1 through 3-3) for the MLG520 program is keyed onto a diskette using an offline device, such as the 5280 Distributed Data System (Figure 3-5). When placed on diskette, the input stream becomes a diskette data file, which is identified by a specific data file label. In this example, the label is SOURCE.



```
// JOB
CRTRPGPGM PGM(MLG520) SRCFILE(SOURCE)
// DATA FILE(SOURCE) FILETYPE(*SRC)
     H
     F*
     F* MLG520  LABEL PRINTING PROGRAM
     F*
     FINPUT   IP  F        96              SEQ
     FOPRINT  O   F        50       OF     PRINTER
     IINPUT   AA  01
     I                                           1   50ACTNUM
     I                                           6   60ACTTYP
     I                                           7  24 NAME
     I                                          25  42 ADDR
     I                                          43  60 CITY
     I                                          61  62 STATE
     I                                          63  670ZIP
     OOPRINT  D 31      01
     O                              ACTNUM    20
     O                              NAME      42
     O        D  1      01
     O                              ADDR      42
     O        D  1      01
     O                              CITY      42
     O        D  3      01
     O                              STATE     26
     O                              ZIP       42
// ENDJOB
```

Figure 3-5. Using an Offline Device to Create an Input Stream

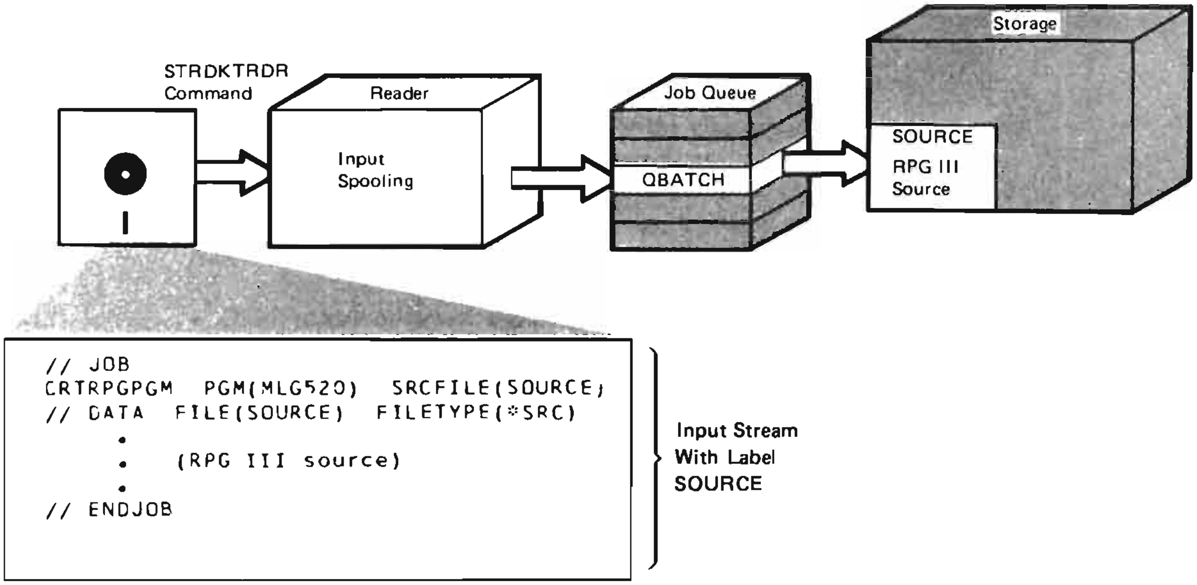
To get the input stream from diskette into the system, a system user such as the system operator places the diskette in the diskette magazine drive and enters the Start Diskette Reader (STRDKTRDR) command. This command might be entered, for example, from the system operator menu at the system console or another work station.



```
                        SYSTEM OPERATOR MENU
Select one of the following:
  1. DSPJOBQ (jobq)                 7. STRPRTWTR device,outq
  2. DSPOUTQ (outq)                 8. CNLWTR    writer
  3. SNDMSG  tomsgq,(type),msg      9. STRDKTRDR device,label
  4. CALL    program               10. CNLRDR    reader
  5. Execute command               90. SIGNOFF   (*NOLIST *LIST)
  6. SBMJOB  (job),(jobd),(cmd)
 Option: 5   Parms: ____________________ ____________________
 Msg or cmd: strdktrdr dev(qdkt) label(source) loc(*s1)_______
_______________________________________________________________
 Log requests:  *YES  CF3-Command entry    CF4-Prompt (5 only)
 CF6-DSPMSG QSYSOPR   CF7-DSPSBS           CF8-DSPSYS
```

The diskette file to be read is indicated by specifying the label SOURCE.

When the STRDKTRDR command is executed, a spooling reader reads the input stream into system storage. The reader puts an entry on a job queue that identifies the commands and source in the input stream as a batch job and creates an inline data file to hold the RPG III source (Figure 3-6).



```
// JOB
CRTRPGPGM   PGM(MLG520)   SRCFILE(SOURCE;
// DATA   FILE(SOURCE)   FILETYPE(*SRC)
       •
       •    (RPG III source)
       •
// ENDJOB
```

Figure 3-6. Reading an Input Stream from Diskette to Compile MLG520

The CL commands in the input stream provide the necessary instructions for the system to process the batch job that compiles the RPG III program MLG520.

The // *JOB statement* identifies the beginning of a batch job. This statement and all statements that begin with // are recognized by the spooling reader. The slashes must be in positions 1 and 2. The command can begin in position 3 or can be preceded by blanks. The default IBM-supplied job description, QBATCH, is used in this approach because a job description (the JOBD keyword of the JOB command) is not specified. A job description contains a set of job-related attributes (such as priority, job queue, and message logging level). One or more of the attributes may be overridden by specifying the attributes on the // JOB statement; but, in this case, they are not overridden. A job name may be specified on the // JOB statement to help the user identify the job in the system. In this case, a job name is not specified and the job name defaults to the job description name, QBATCH. The // JOB statement in this example does not specify a job queue name; therefore, the job queue on which this job is placed is determined by the command used to start the spooling reader (the STRDKTRDR command in this example). If a job queue name is not specified in the STRDKTRDR command, the job is placed on the default job queue, QBATCH, for execution in the QBATCH subsystem.

The *CRTRPGPGM command* instructs the system to compile the RPG III program from the source statements in the job's inline data file, SOURCE, and to store the executable (compiled) program under the name MLG520 so that it can be called later.

The // *DATA statement* identifies the data that follows as an inline data file, which, in this case, is called SOURCE. The keyword FILETYPE identifies the data as source (*SRC) so that the system can correctly process it. The data in this file named SOURCE consists of the RPG III source statements (Figures 3-1 through 3-3), which follow the DATA command. Because the CRTRPGPGM command specified the file SOURCE, the RPG program will be compiled from these source statements.

The // *ENDJOB statement* is at the end of the input stream and designates both the end of the inline data file and the end of the job.

When the diskette reader has finished reading the input stream from the diskette and has created an entry for the job on the QBATCH job queue, as shown in Figure 3-7, a message appears on the system operator message queue saying that the job has been successfully spooled into the system. When the job is the highest priority job on the job queue, CPF initiates the job and the CRTRPGPGM command is executed. The RPG III program is compiled and stored in the default system general purpose library, QGPL. At the same time, a compiler listing of the RPG III program is generated and placed on the default output queue QPRINT.

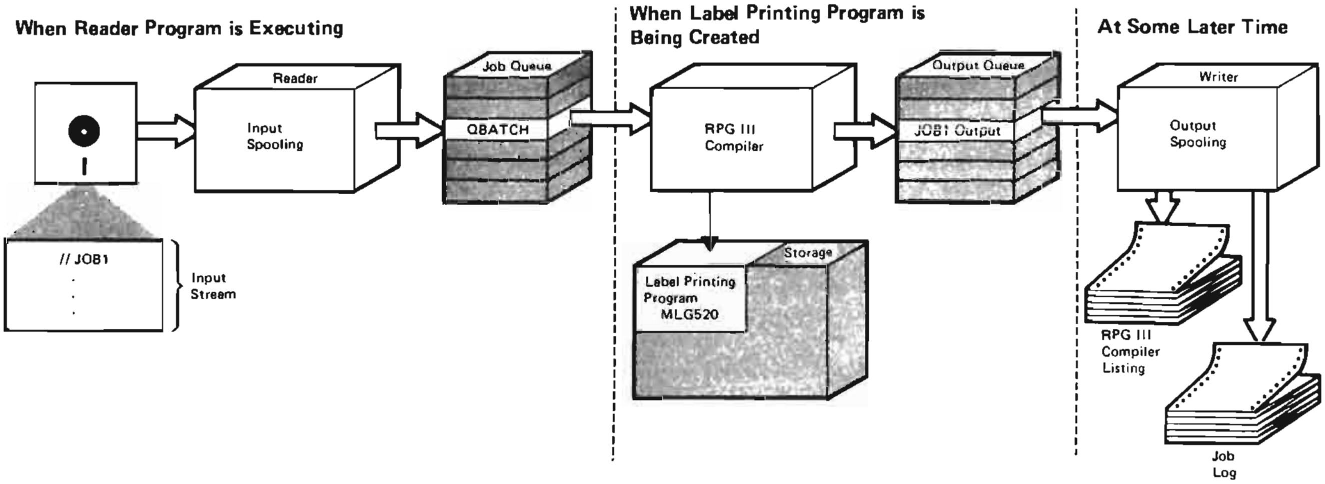


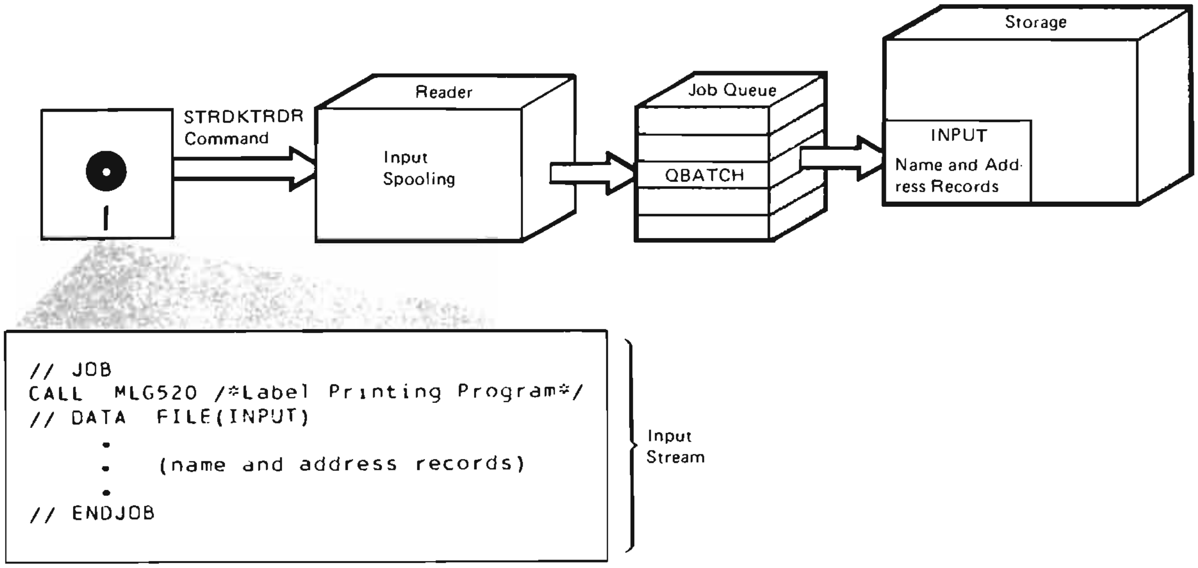
Figure 3-7. Creating an RPG III Program from Diskette

Because no other CL commands follow the CRTRPGPGM command (see Figure 3-6), CPF terminates the job, generates a job log, and places the job log on the QPRINT output queue along with the RPG III compiler listing. The compiler listing and the job log are printed when the output priority of the job is the highest on the output queue. This printing is done because the Start Printer Writer (STRPRTWTR) command was executed when you prepared your system for this example (see *Preparing for Daily Operations* in Chapter 2).

Executing the Label Printing Program

Once the MLG520 label printing program is compiled, system users can print mailing labels by calling the program and providing master records for the program to process. In this approach, the MLG520 program is called by a CL command (the CALL command) in an input stream that is read from diskette. The input stream also contains name and address records (master records) that the MLG520 program is to print. The format of the name and address records is described under *Characteristics* in Chapter 1.

An offline device is used to key the input stream onto diskette, as was done when the MLG520 program was compiled (see Figure 3-5). The input stream containing the CALL command and the name and address records is read from diskette and placed on the QBATCH job queue for processing, as shown in Figure 3-8.



```
// JOB
CALL   MLG520 /*Label Printing Program*/
// DATA   FILE(INPUT)
      .
      .   (name and address records)
      .
// ENDJOB
```

Figure 3-8. Reading an Input Stream from Diskette to Execute MLG520

Because this input stream is similar to the input stream for compiling the MLG520 program (see Figure 3-6), only the differences are discussed below.

The *CALL command* instructs the system to execute the previously compiled RPG III program MLG520. The /* Label Printing Program */ comment helps to document the CL command. A CL comment always begins with /* and ends with */.

The *// DATA statement* identifies the beginning of the inline data file, which contains the name and address records. The name INPUT is specified for this inline data file. The same file name is declared as the input file in MLG520 label printing program (see Figures 3-1 and 3-2).

When the job is selected from the job queue, MLG520 is called into execution (Figure 3-9). The program reads each input record (name and address) from the inline data file INPUT and writes each mailing label. The mailing label output is spooled and placed on the QPRINT output queue. A job log, which lists the input stream commands and any system messages, is also produced. The job log is spooled and placed on the QPRINT output queue along with the mailing labels. The mailing labels and job log can be printed when the job terminates.

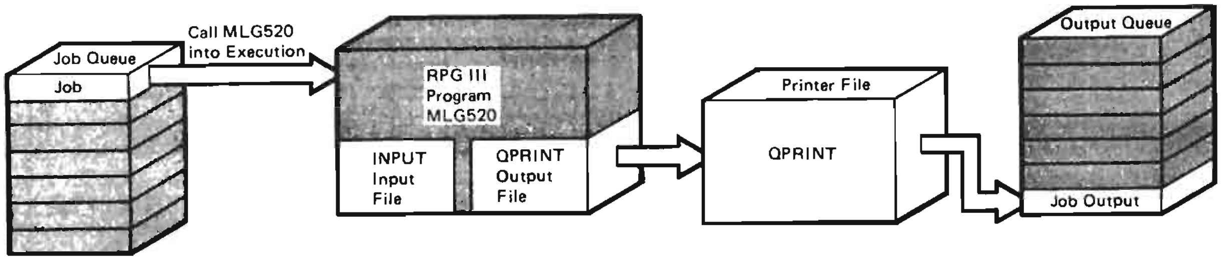
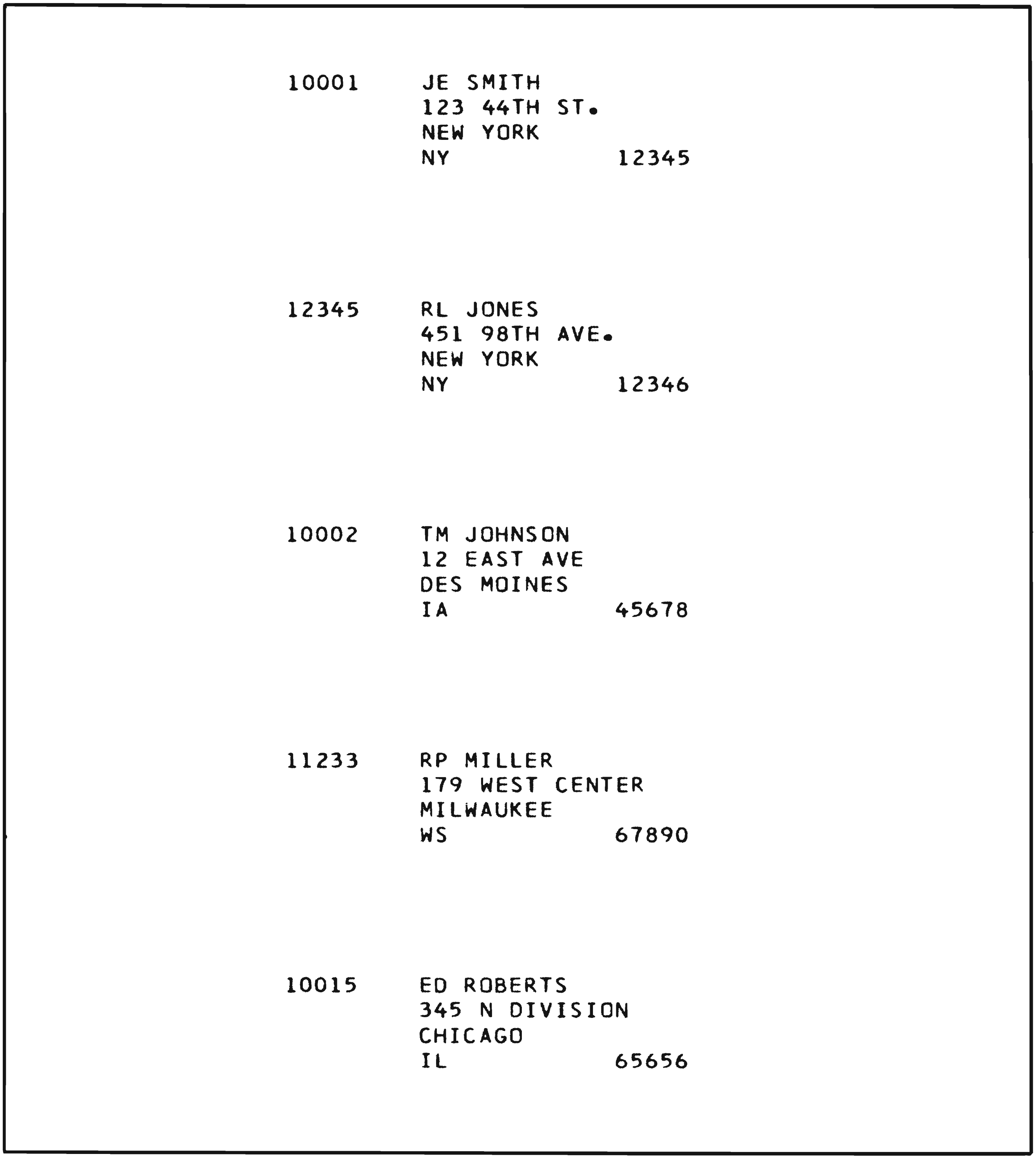


Figure 3-9. Calling a Job into Execution and Placing Labels on an Output Queue

An example of mailing labels printed by the MLG520 program is shown on the following page.



```
10001    JE SMITH
         123 44TH ST.
         NEW YORK
         NY             12345



12345    RL JONES
         451 98TH AVE.
         NEW YORK
         NY             12346



10002    TM JOHNSON
         12 EAST AVE
         DES MOINES
         IA             45678



11233    RP MILLER
         179 WEST CENTER
         MILWAUKEE
         WS             67890



10015    ED ROBERTS
         345 N DIVISION
         CHICAGO
         IL             65656
```

RECOVERY CONSIDERATIONS

If the mailing labels were not printed or were printed incorrectly, the cause is most likely an error in:

- The name and address records or associated commands on the diskette used to execute the label printing program.

- The RPG III source statements or associated commands on the diskette used to compile the label printing program.

In either situation the input stream must be correctly keyed on diskette and read into storage again.

Similarly, if a system failure occurred while the label printing program was executing, the diskette containing the input stream to execute the program must be read into storage again.

Chapter 4. Batch Input and Batch Maintenance

OVERVIEW

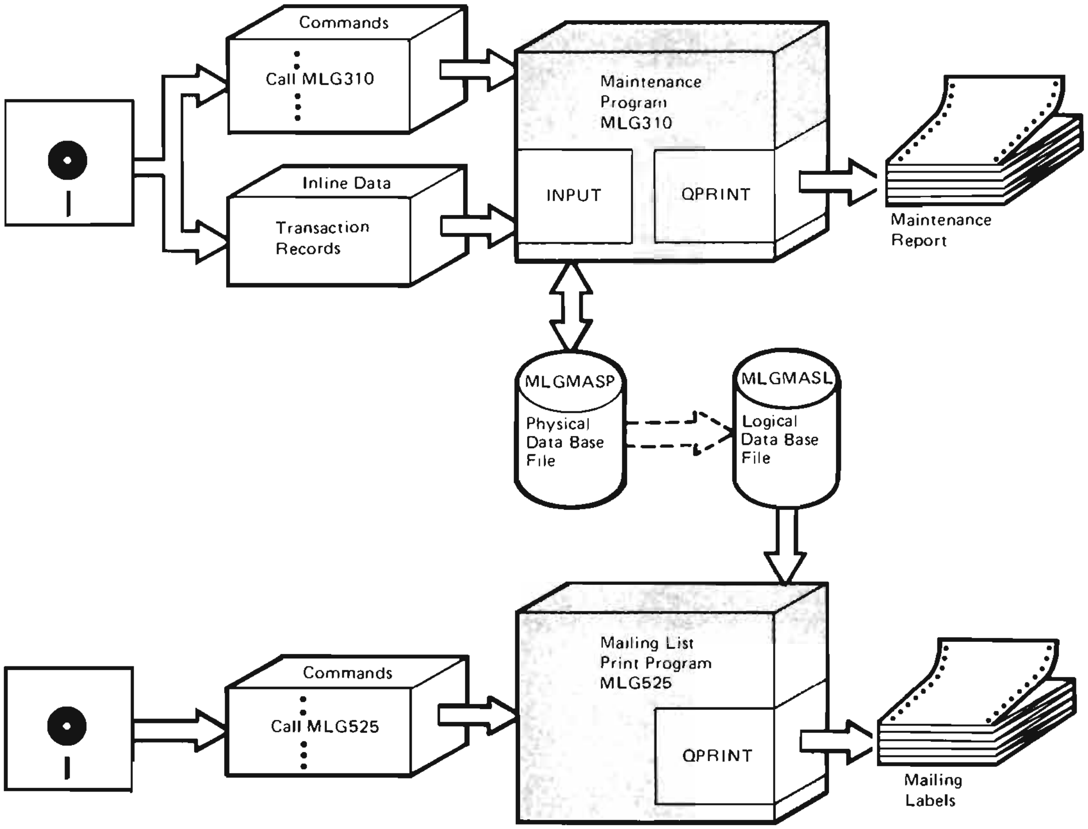
The approach described in this chapter expands on the previous approach in Chapter 3. In addition to printing mailing labels, this approach provides batch maintenance of a master file in the data base. Externally described data files with different access paths are used. This approach to the application has the following characteristics:

- Transaction records are read from diskette as part of an input stream and placed in an inline data file, INPUT.

- A CALL command in the input stream calls a maintenance program, MLG310. The program reads the transaction records from the inline data file and updates a physical data base file, MLGMASP, which contains the master records.

- The mailing list print program, MLG525, prints mailing labels from the MLGMASP master file using a different access path and selection criteria provided by a logical file, MLGMASL. An input stream read from diskette creates the logical file, calls the mailing list print program, and then deletes the logical file after the program has been executed.

- All files are created as externally described data files.



DATA BASE FILES

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 System/38 *data base* is an organized collection of all data files stored in the system. System/38 allows you to store data in one physical structure, but allows multiple users to access the data in various logical formats, organizations, and sequences. Consequently, data redundancy is reduced and you can now easily meet requirements that were difficult or impossible to meet on many previous systems.

A *data base file* is an organized collection of related records in the data base. There are two types of data base files on the System/38:

- A *physical file* is a data base 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 data base 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 are different from the physical representation of the data in the data base.

A file must be created on System/38 before a program can use the file. An RPG III program cannot create a file. RPG III can add records to a file that was created previously by a CL command. When a physical file is created, no data records exist in it. A program, such as an RPG III program, or the Copy File (CPYF) command must be used to add records to the file. In this application approach, an RPG III maintenance program, MLG310, is used to add records to the file.

Access Paths

Data in both types of data base files can be processed by a program. The file definition includes an *access path*, which is the means by which CPF provides a logical sequence to the data records in a data base 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 publication. 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:

- The fields in the record which are key fields

- Ascending or descending order for the key field(s)

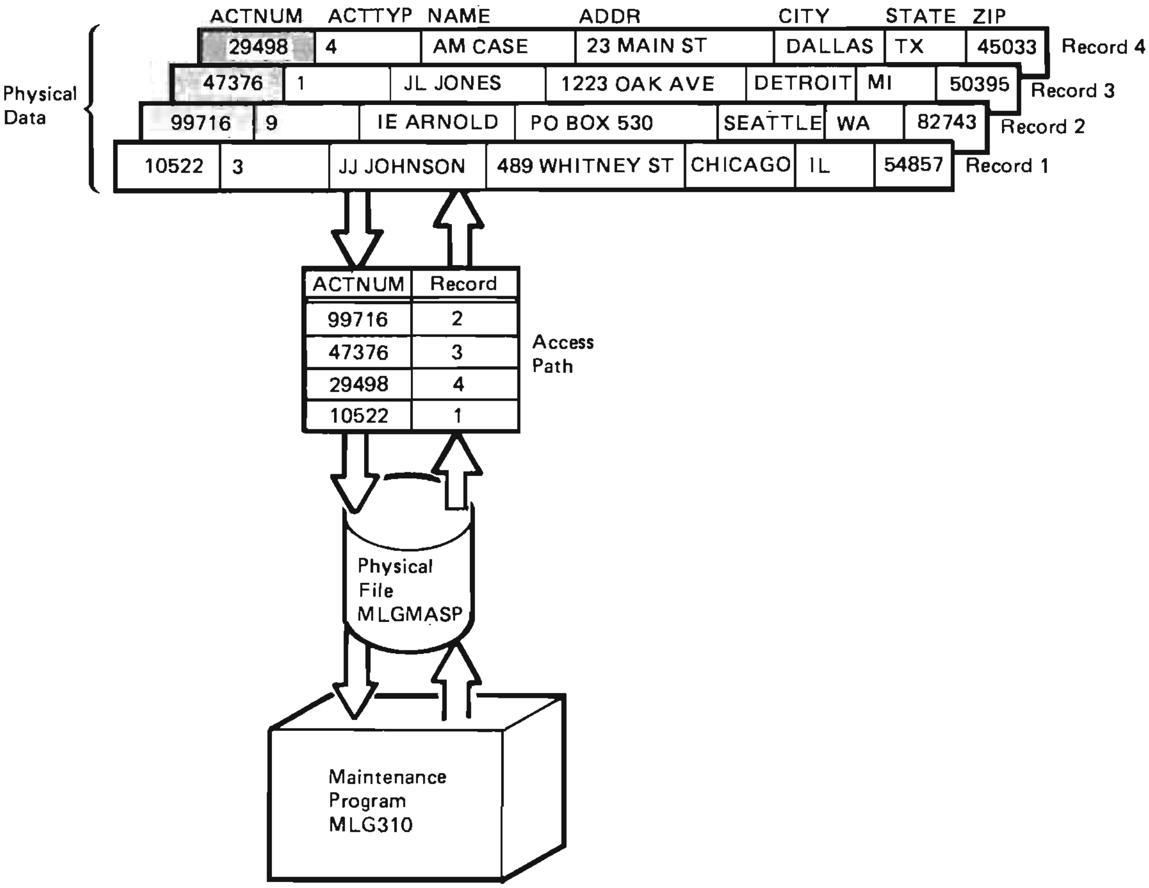
- The order of the key fields when multiple fields compose the key

Selection criteria can also be used to form a subset of records or to limit the number of records in a logical file.

For more information on keyed sequence access paths, refer to the *CPF Programmer's Guide*. Specifying an access path on data description specifications will be discussed later.

In this approach, a physical master file, MLGMASP, is created from data read from the diskette.

The following illustration shows physical data records in storage. The access path orders these records by the key field which, in this case, is account number (ACTNUM). When the program requests a specific key, the access path is used to access the specific records and to present the records to the program.

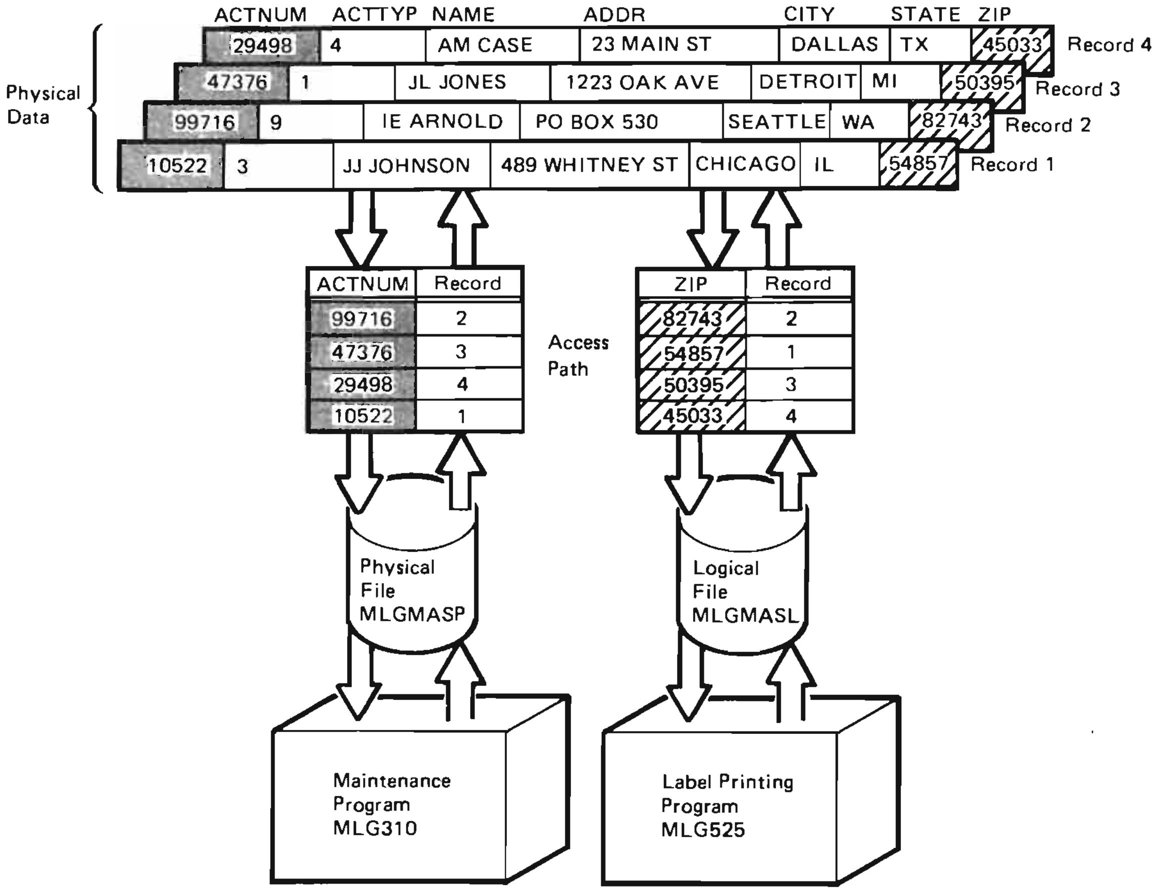


ACTNUM	ACTTYP	NAME	ADDR	CITY	STATE	ZIP	
29498	4	AM CASE	23 MAIN ST	DALLAS	TX	45033	Record 4
47376	1	JL JONES	1223 OAK AVE	DETROIT	MI	50395	Record 3
99716	9	IE ARNOLD	PO BOX 530	SEATTLE	WA	82743	Record 2
10522	3	JJ JOHNSON	489 WHITNEY ST	CHICAGO	IL	54857	Record 1

ACTNUM	Record
99716	2
47376	3
29498	4
10522	1

Note that the order of the physical data records is not important. The access path contains both the key to match against the program's request and a corresponding entry (record number) to determine where the physical data record is located.

Multiple access paths should be created when physical data needs to be accessed in different sequences. For example, the records in the following illustration must be accessed by both a maintenance program and a label printing program.



	ACTNUM	ACTTYP	NAME	ADDR	CITY	STATE	ZIP
Record 4	29498	4	AM CASE	23 MAIN ST	DALLAS	TX	45033
Record 3	47376	1	JL JONES	1223 OAK AVE	DETROIT	MI	50395
Record 2	99716	9	IE ARNOLD	PO BOX 530	SEATTLE	WA	82743
Record 1	10522	3	JJ JOHNSON	489 WHITNEY ST	CHICAGO	IL	54857

ACTNUM	Record
99716	2
47376	3
29498	4
10522	1

ZIP	Record
82743	2
54857	1
50395	3
45033	4

The maintenance program uses the account number (ACTNUM) access path. The label printing program uses the zip code (ZIP) access path.

A file must be created for each access path that is required. In this example, the account number access path is defined in the physical file MLGMASP (where the data is actually stored) and the zip code access path is defined in the logical file MLGMASL. If additional access paths were needed for the data in MLGMASP, one additional logical file would have to be created for each additional access path needed. Although only one key field is used for each access path in this example, more than one key field can be specified in each file that defines a separate access path.

Externally Described Data Files

The physical data base file and the logical data base file used in this approach are *externally described*. The records of an externally described data file are described to CPF 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, because the field descriptions can be automatically included in the program when it is compiled. However, 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.

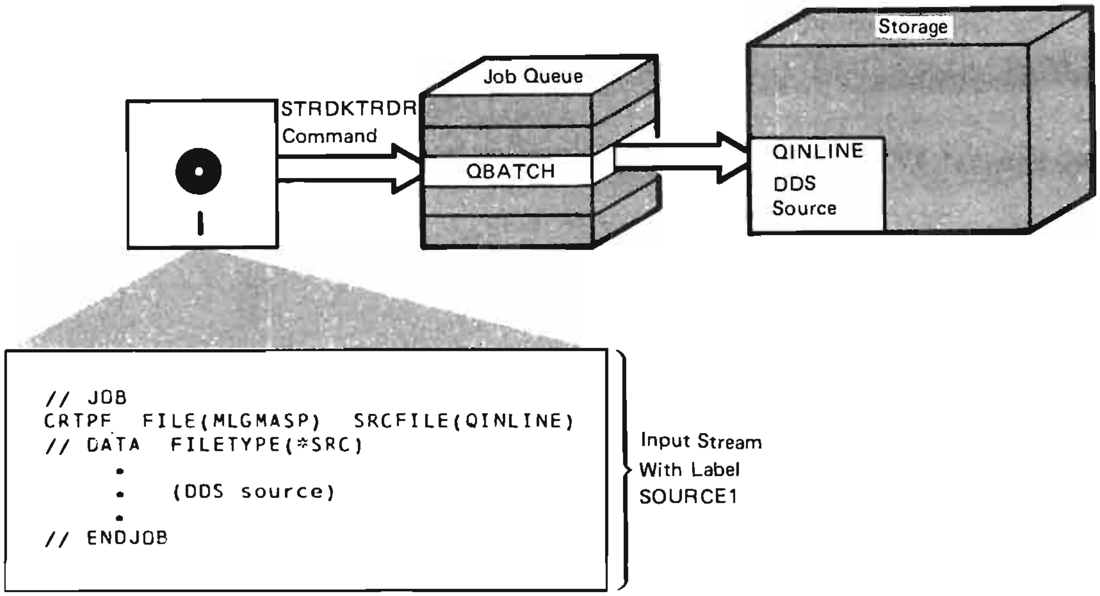
- Field definitions are not coded in every program because the RPG III compiler can automatically copy the definition. This helps to keep field names and their attributes consistent.

- Two of the interactive data base utilities (data file utility and query utility) require that fields be externally defined.

MASTER FILE

In this example, a definition for the file will be created in a batch job. Batch jobs will also be used to create and execute the programs used in this approach.

The physical file definition for MLGMASP is created by an input stream. This input stream is a file on diskette with a label of SOURCE1, as shown in Figure 4-1. The input stream is read into storage when you issue a STRDKTRDR command with SOURCE1 specified as the label.



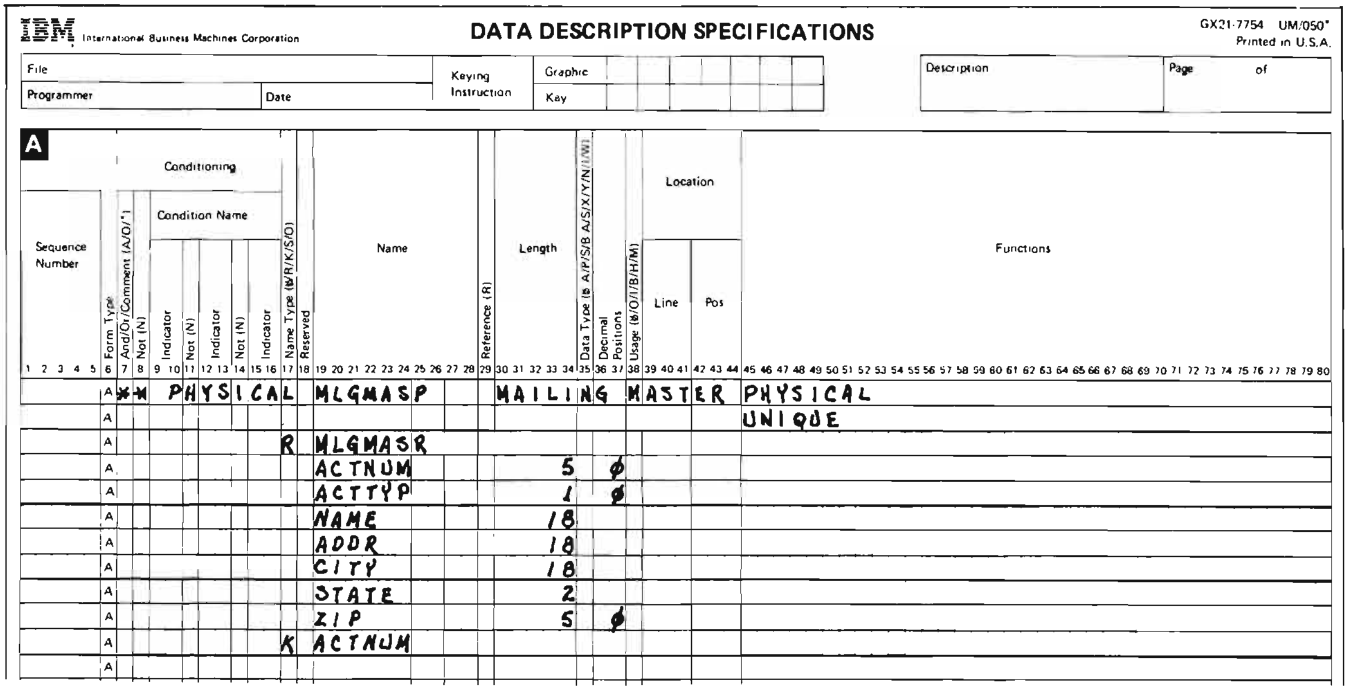
```
// JOB
CRTPF  FILE(MLGMASP)  SRCFILE(QINLINE)
// DATA  FILETYPE(*SRC)
      .
      .    (DDS source)
      .
// ENDJOB
```

Figure 4-1. Reading an Input Stream from Diskette to Create MLGMASP

The *CRTPF command* in the input stream creates a physical file. The file is to be named MLGMASP (physical master file).

The *// DATA* statement preceding the DDS source statements defines the DDS source as an inline data file. Because the // DATA statement does not specify a file name, the inline data file containing the DDS source is assigned the default name of QINLINE when the input stream is read from diskette into storage.

The *DDS source* in the inline data file is shown in Figure 4-2.



IBM International Business Machines Corporation

DATA DESCRIPTION SPECIFICATIONS

GX21-7754 UM/050*
Printed in U.S.A.

File
Programmer
Date
Keying Instruction
Graphic
Key
Description
Page of

A

Sequence Number (1-5)	Form Type (6)	And/Or/Comment (A/O/*) (7)	Not (N) (8)	Condition Name (9-16)	Name Type (b/R/K/S/O) (17)	Reserved (18)	Name (19-28)	Reference (R) (29)	Length (30-34)	Data Type (35)	Decimal Positions (36-37)	Usage (38)	Location Line (39-41)	Location Pos (42-44)	Functions (45-80)
	A	*	*	PHYSICA	L		MLGMASP		MAILING		MASTER				PHYSICAL
	A														UNIQUE
	A				R		MLGMASR								
	A						ACTNUM		5		0				
	A						ACTTYP		1		0				
	A						NAME		18						
	A						ADDR		18						
	A						CITY		18						
	A						STATE		2						
	A						ZIP		5		0				
	A				K		ACTNUM								
	A														

Figure 4-2. DDS for MLGMASP File

The first line of the DDS is a comment (* in position 7 designates a comment) that identifies the source to the programmer. The * in position 8 is used as a convention to format the first comment line in a standard manner.

The UNIQUE entry specifies that the access path is to contain unique keys. A keyed sequence access path is specified using the ACTNUM field. Consequently, each account number must be unique; if you attempt to add a new record with an account number that already exists in the file, that record will be rejected.

The R in position 17 defines a *record format* named MLGMASR. A record format defines the names and order of the fields in the records contained in a file. This definition includes the record name and field descriptions for the fields contained in the record. RPG III requires that the names of files and the names of formats defined by externally described data be unique within the program (a format may not have the same name as the file). Each field is defined with a name and a length. Decimal positions are defined for those fields that are to contain only numeric data.

In the last source line, the ACTNUM field is specified as the key field for this file by placing a K in position 17.

USING TRANSACTION RECORDS

In this approach, the data base master file, MLGMASP, is updated from transaction records that are keyed on diskette and then read into the system. A transaction record is defined as follows:

Field Name	Field Description	Length	Decimal Positions	Position Range	Field Type
BATNUM	Batch number	6	0	1-6	Numeric
TRNTYP	Type of transaction	1		7	Character
ACTNUM	Account number	5	0	8-12	Numeric
ACTTYP	Type of account	1	0	13	Numeric
NAME	Name	18		14-31	Character
ADDR	Address	18		32-49	Character
CITY	City	18		50-67	Character
STATE	State	2		68-69	Character
ZIP	Zip code	5	0	70-74	Numeric

The record definition is the same as that discussed under *Mailing List Application* in Chapter 1 except for the addition of fields to contain the batch number and type of transaction. The batch number is assigned externally and allows a unique control number over a group of transactions.

The type of transaction is designated by:

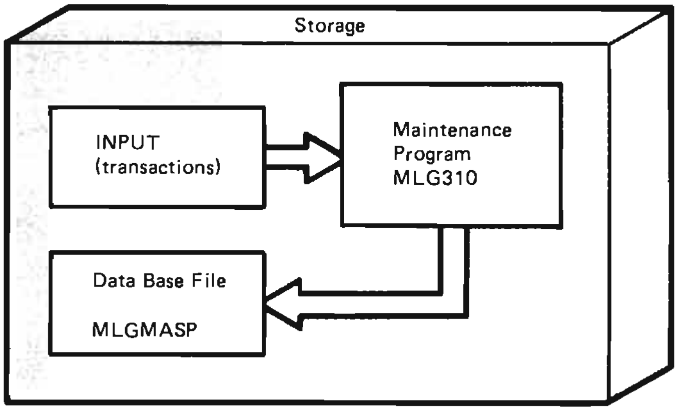
A for addition
C for change
D for deletion

If a transaction is coded for addition, all fields must be entered. If a transaction record is coded for change, all fields of that record must be reentered in the transaction record, even if some of the data is the same as data in the master record. To change the account number of a master record, the existing master record must be deleted and a transaction record must be entered with the new account number.

If a record is coded for deletion, only the batch number, type of transaction, and account number are entered.

MAINTENANCE PROGRAM

The maintenance program, MLG310, reads an inline data file of transaction records and updates the existing master data base file as shown below.



The program is written in RPG III. It is coded to read a transaction file (INPUT), update the mailing list file (MLGMASP), and produce a spooled output file (QPRINT), to be printed later. The INPUT transaction file is created as an inline data file when the transaction records are read into the system from diskette. The transaction records are described within the program by RPG input specifications.

The source for the maintenance program is shown in Figures 4-3 through 4-8 (see the following pages). No H Specification statement is required.

Control and File Description Specifications, shown in Figure 4-3, identify the three files to be used. The MLGMASP file is coded as follows:

- MLGMASP in positions 7 through 14 designates a file name that must be defined to the system.

- U in position 15 designates an update; the records are changed during the program.

- F in position 16 designates full procedural; the programmer, rather than the RPG III program cycle, specifies when the file is to be read.

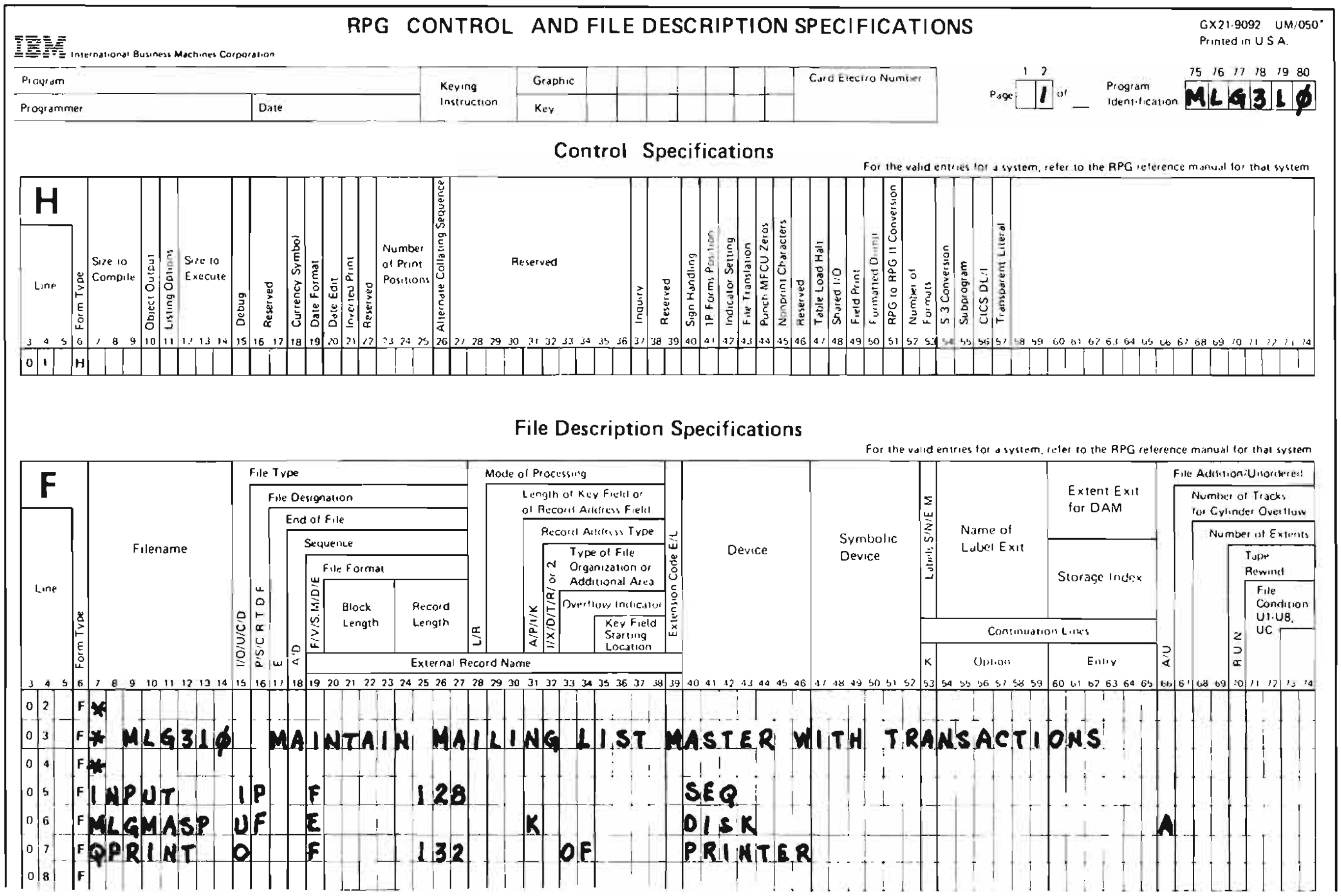
- E in position 19 designates an externally described file; the RPG III compiler copies the field descriptions from the externally described file at compilation time.

- K in position 31 designates a keyed sequence file.

- DISK in positions 40 through 46 designates device type.

- A in position 66 designates additions; the program can add new records to the file.

When an externally described file is used, the record length entry (positions 24 through 27) must be blank. When a program-described file is used, the record length must be entered. RPG III on System/38 does not use the block length entry (positions 20 through 23). The INPUT and QPRINT files are similar to the files with these names in the previous approach.



IBM International Business Machines Corporation

RPG CONTROL AND FILE DESCRIPTION SPECIFICATIONS

GX21-9092 UM/050*
Printed in U.S.A.

Page 1 of __ Program Identification MLG31Ø

Control Specifications

For the valid entries for a system, refer to the RPG reference manual for that system

```
01 H
```

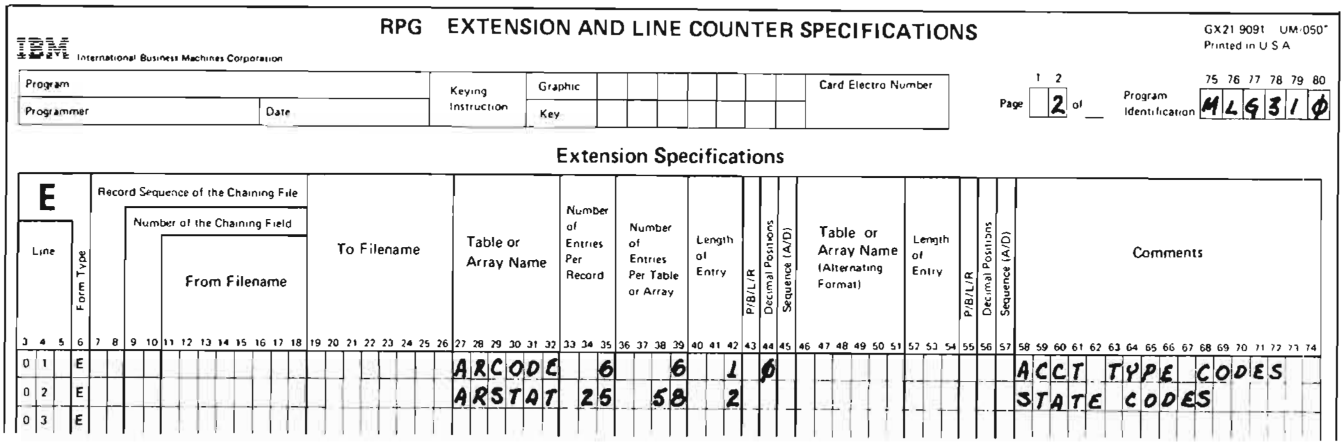
File Description Specifications

For the valid entries for a system, refer to the RPG reference manual for that system

```
02 F*
03 F* MLG31Ø  MAINTAIN MAILING LIST MASTER WITH TRANSACTIONS
04 F*
05 FINPUT   IP  F     128            SEQ
06 FMLGMASP UF  E           K        DISK                      A
07 FQPRINT  O   F     132     OF     PRINTER
08 F
```

Figure 4-3. Control and File Description Specifications for MLG310 Maintenance Program

Extension and Line Counter Specifications, shown in Figure 4-4, describe two compile-time arrays that are used to validate the account type and state being entered. The data for these arrays is shown later in Figure 4-8.



IBM International Business Machines Corporation

RPG EXTENSION AND LINE COUNTER SPECIFICATIONS

GX21 9091 UM/050*
Printed in U S A

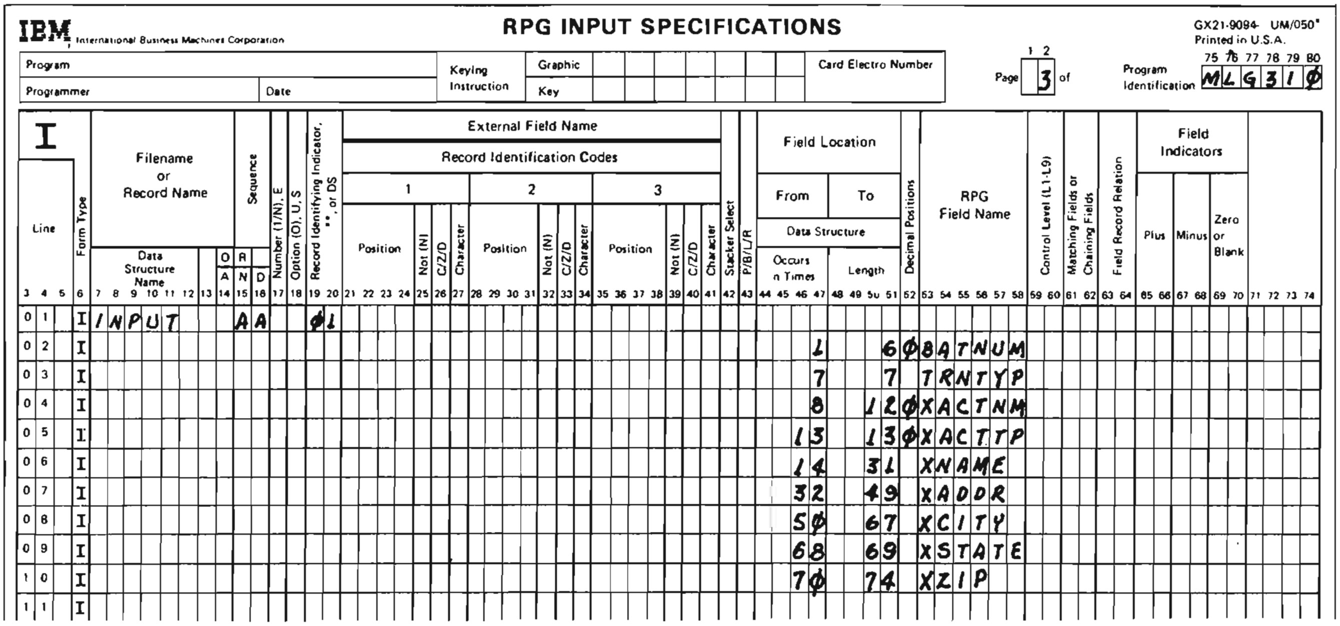
Program
Programmer Date
Keying Instruction: Graphic / Key
Card Electro Number
Page 2 of
Program Identification (75–80): MLG31Ø

Extension Specifications

Line (3-5)	Form Type (6)	Record Sequence of the Chaining File (7-8)	Number of the Chaining Field (9-10)	From Filename (11-18)	To Filename (19-26)	Table or Array Name (27-32)	Number of Entries Per Record (33-35)	Number of Entries Per Table or Array (36-39)	Length of Entry (40-42)	P/B/L/R (43)	Decimal Positions (44)	Sequence (A/D) (45)	Table or Array Name (Alternating Format) (46-51)	Length of Entry (52-54)	P/B/L/R (55)	Decimal Positions (56)	Sequence (A/D) (57)	Comments (58-74)
01	E					ARCODE	6	6	1		Ø							ACCT TYPE CODES
02	E					ARSTAT	25	58	2									STATE CODES
03	E																	

Figure 4-4. Extension and Line Counter Specifications for MLG310 Maintenance Program

Input Specifications, shown in Figure 4-5, define the field names for the transaction record because the file is program-described. These names are different from those defined in the master record because RPG has a single storage area for each defined field name. If the same field names are used in both files, the master record data would overlay the transaction record data when the master record is read. No input specifications are needed for the externally described file.



IBM International Business Machines Corporation

RPG INPUT SPECIFICATIONS

GX21-9094 UM/050*
Printed in U.S.A.

Program
Programmer Date
Keying Instruction: Graphic / Key
Card Electro Number
Page 3 of
Program Identification (75–80): MLG31Ø

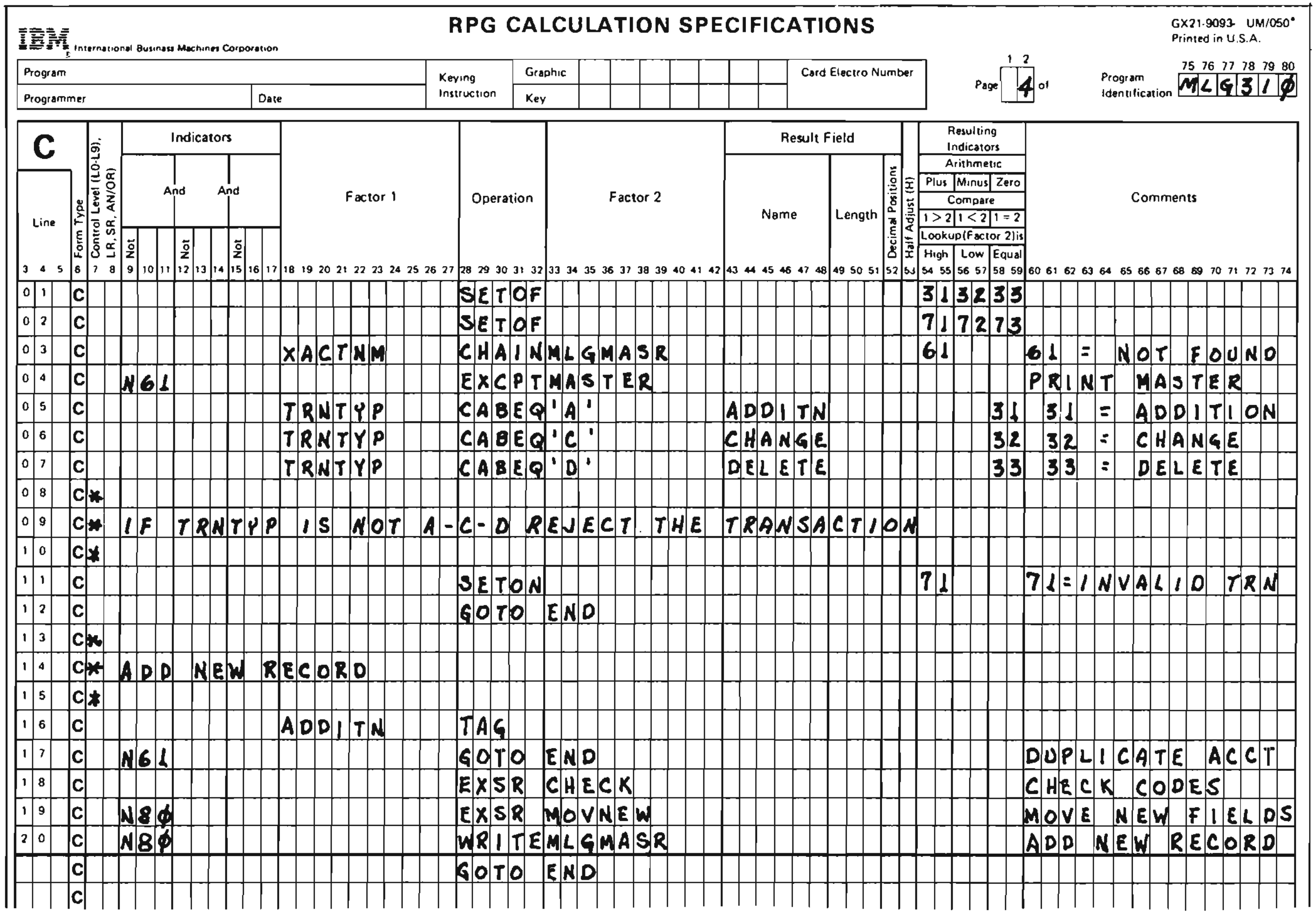
Line (3-5)	Form Type (6)	Filename or Record Name / Data Structure Name (7-14)	Sequence (15-16)	Number (1/N), E (17)	Option (O), U, S (18)	Record Identifying Indicator, **, or DS (19-20)	Record Identification Codes 1: Position (21-24)	Not (N) (25)	C/Z/D (26)	Character (27)	2: Position (28-31)	Not (N) (32)	C/Z/D (33)	Character (34)	3: Position (35-38)	Not (N) (39)	C/Z/D (40)	Character (41)	Stacker Select (42)	P/B/L/R (43)	Field Location From / Data Structure Occurs n Times (44-47)	To / Length (48-51)	Decimal Positions (52)	RPG Field Name (53-58)	Control Level (L1-L9) (59-60)	Matching Fields or Chaining Fields (61-62)	Field Record Relation (63-64)	Field Indicators Plus (65-66)	Minus (67-68)	Zero or Blank (69-70)	(71-74)
01	I	INPUT	AA			Ø1																									
02	I																				1	6	Ø	BATNUM							
03	I																				7	7		TRNTYP							
04	I																				8	12	Ø	XACTNM							
05	I																				13	13	Ø	XACTTP							
06	I																				14	31		XNAME							
07	I																				32	49		XADDR							
08	I																				5Ø	67		XCITY							
09	I																				68	69		XSTATE							
10	I																				7Ø	74		XZIP							
11	I																														

Figure 4-5. Input Specifications for MLG310 Maintenance Program

Calculation Specifications, shown in Figure 4-6, specify the operations to be performed on the data. The program sets off certain indicators to allow proper output for each transaction. The account number in the transaction record (XACTNM) is used to retrieve a master record of the same number. The name of the record (MLGMASR) is used instead of the name of the file because MLGMASP is an externally described file. If no master exists, indicator 61 is on. If the master is found, it is printed by the EXCPT operation, which specifies a label (MASTER) that is used on the output specifications (shown later in Figure 4-7).

The program tests the transaction type and branches to the required processing routine. The compare and branch instructions (CABxx) are used to:

- Make a comparison
- Set on an indicator if the compared items are equal
- Branch to a tag if the compare function is satisfied



IBM International Business Machines Corporation

RPG CALCULATION SPECIFICATIONS

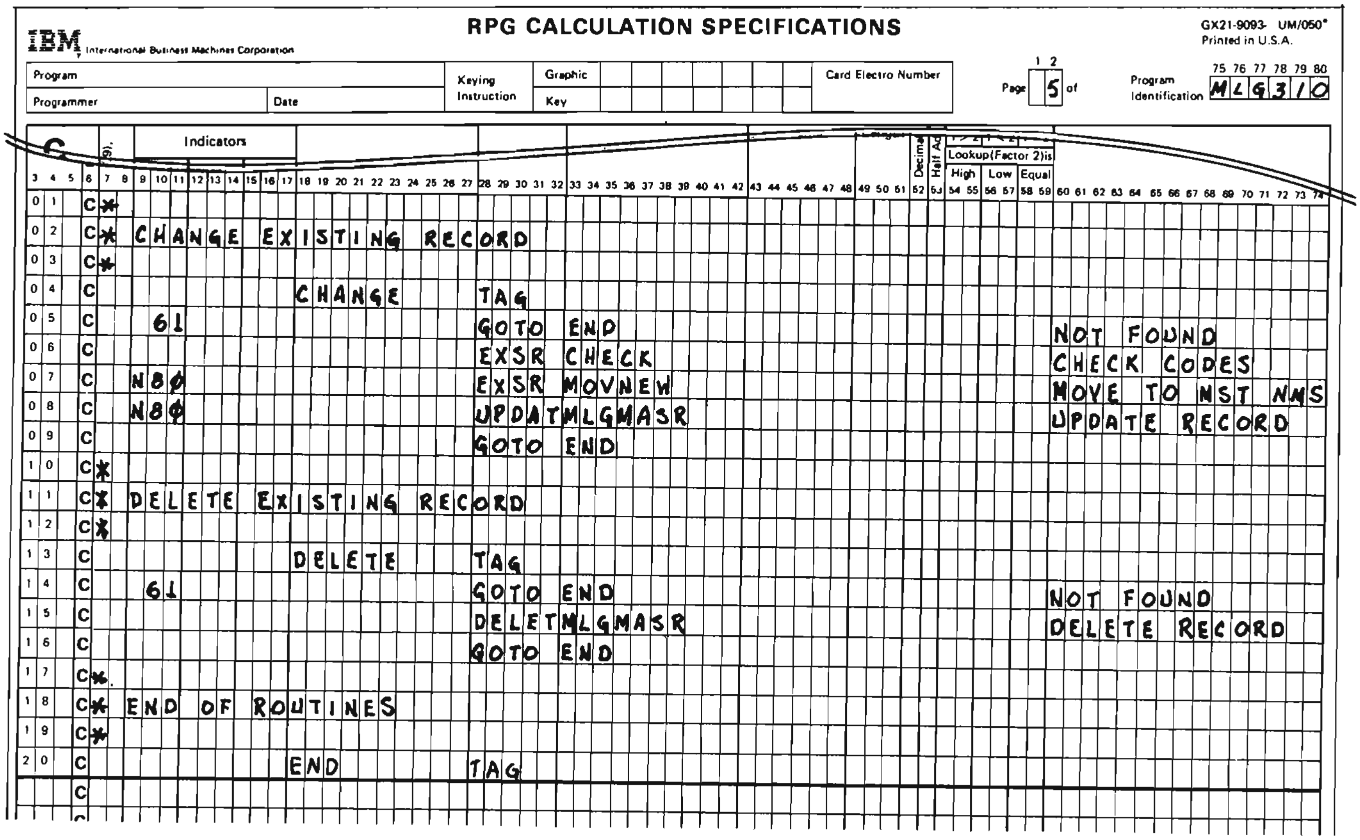
GX21-9093- UM/050*
Printed in U.S.A.

Program
Programmer
Date
Keying Instruction: Graphic / Key
Card Electro Number

Page 4 of
Program Identification MLG31Ø

Line	Form Type	Control Level (L0-L9, LR, SR, AN/OR)	Indicators	Factor 1	Operation	Factor 2	Result Field Name	Length	Decimal Positions	Half Adjust (H)	Resulting Indicators High (1>2, Plus)	Low (1<2, Minus)	Equal (1=2, Zero)	Comments
01	C				SETOF						31	32	33	
02	C				SETOF						71	72	73	
03	C			XACTNM	CHAIN	MLGMASR					61			61 = NOT FOUND
04	C		N61		EXCPT	MASTER								PRINT MASTER
05	C			TRNTYP	CABEQ	'A'	ADDITN						31	31 = ADDITION
06	C			TRNTYP	CABEQ	'C'	CHANGE						32	32 = CHANGE
07	C			TRNTYP	CABEQ	'D'	DELETE						33	33 = DELETE
08	C	*												
09	C	*	IF TRNTYP IS NOT A-C-D REJECT THE TRANSACTION											
10	C	*												
11	C				SETON						71			71=INVALID TRN
12	C				GOTO	END								
13	C	*												
14	C	*	ADD NEW RECORD											
15	C	*												
16	C			ADDITN	TAG									
17	C		N61		GOTO	END								DUPLICATE ACCT
18	C				EXSR	CHECK								CHECK CODES
19	C		N8Ø		EXSR	MOVNEW								MOVE NEW FIELDS
20	C		N8Ø		WRITE	MLGMASR								ADD NEW RECORD
	C				GOTO	END								
	C													

Figure 4-6 (Part 1 of 2). Calculation Specifications for MLG310 Maintenance Program



IBM International Business Machines Corporation

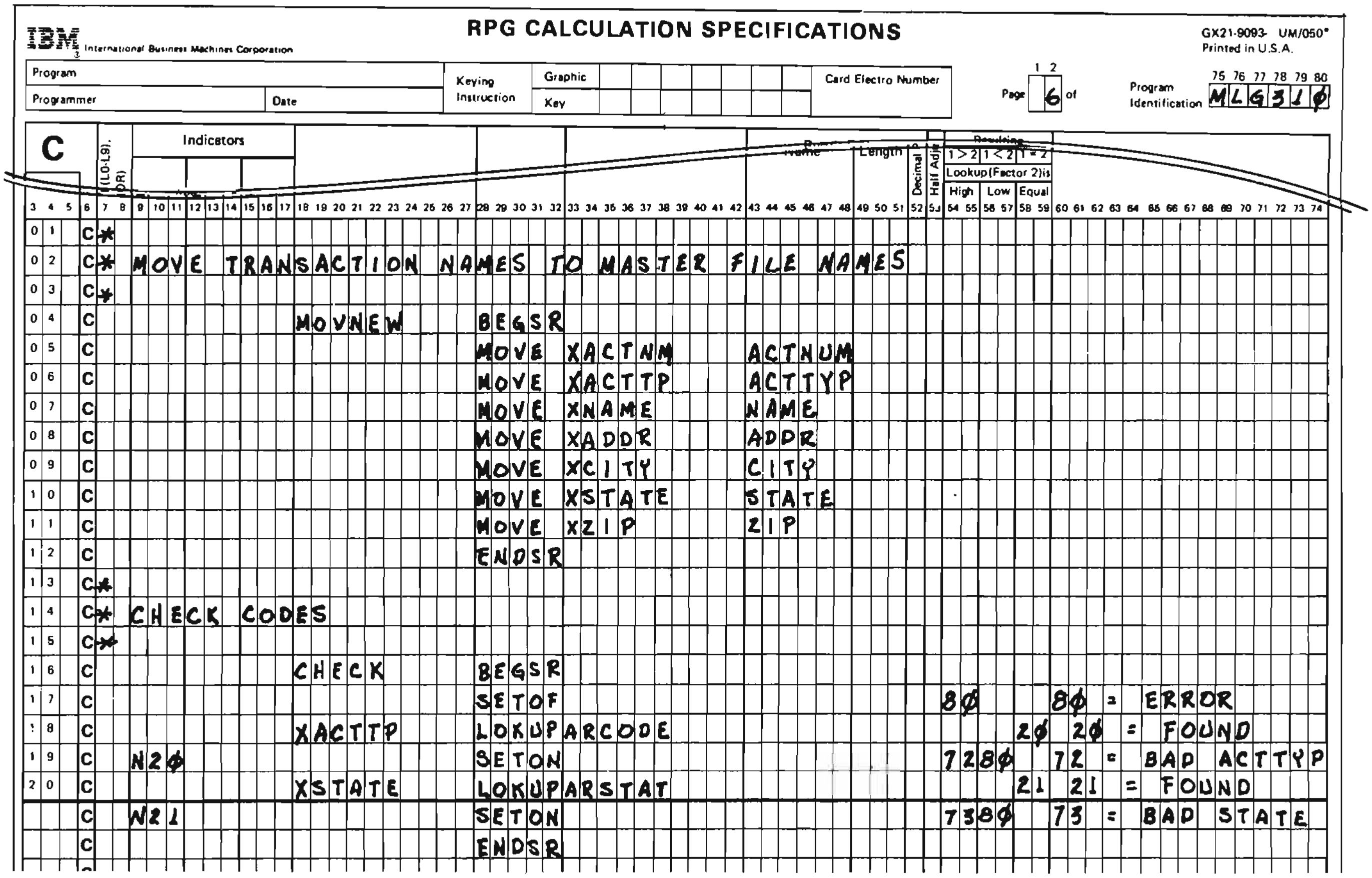
RPG CALCULATION SPECIFICATIONS

GX21-9093- UM/050*
Printed in U.S.A.

Program | Programmer | Date | Keying Instruction | Graphic | Key | Card Electro Number

Page 5 of — Program Identification MLG310

```
01 C*
02 C* CHANGE EXISTING RECORD
03 C*
04 C           CHANGE    TAG
05 C   61                GOTO END                        NOT FOUND
06 C                     EXSR CHECK                      CHECK CODES
07 C  N8Ø                EXSR MOVNEW                     MOVE TO MST NMS
08 C  N8Ø                UPDATMLGMASR                    UPDATE RECORD
09 C                     GOTO END
10 C*
11 C* DELETE EXISTING RECORD
12 C*
13 C           DELETE    TAG
14 C   61                GOTO END                        NOT FOUND
15 C                     DELETMLGMASR                    DELETE RECORD
16 C                     GOTO END
17 C*
18 C* END OF ROUTINES
19 C*
20 C           END       TAG
   C
```



IBM International Business Machines Corporation

RPG CALCULATION SPECIFICATIONS

GX21-9093- UM/050*
Printed in U.S.A.

Program | Programmer | Date | Keying Instruction | Graphic | Key | Card Electro Number

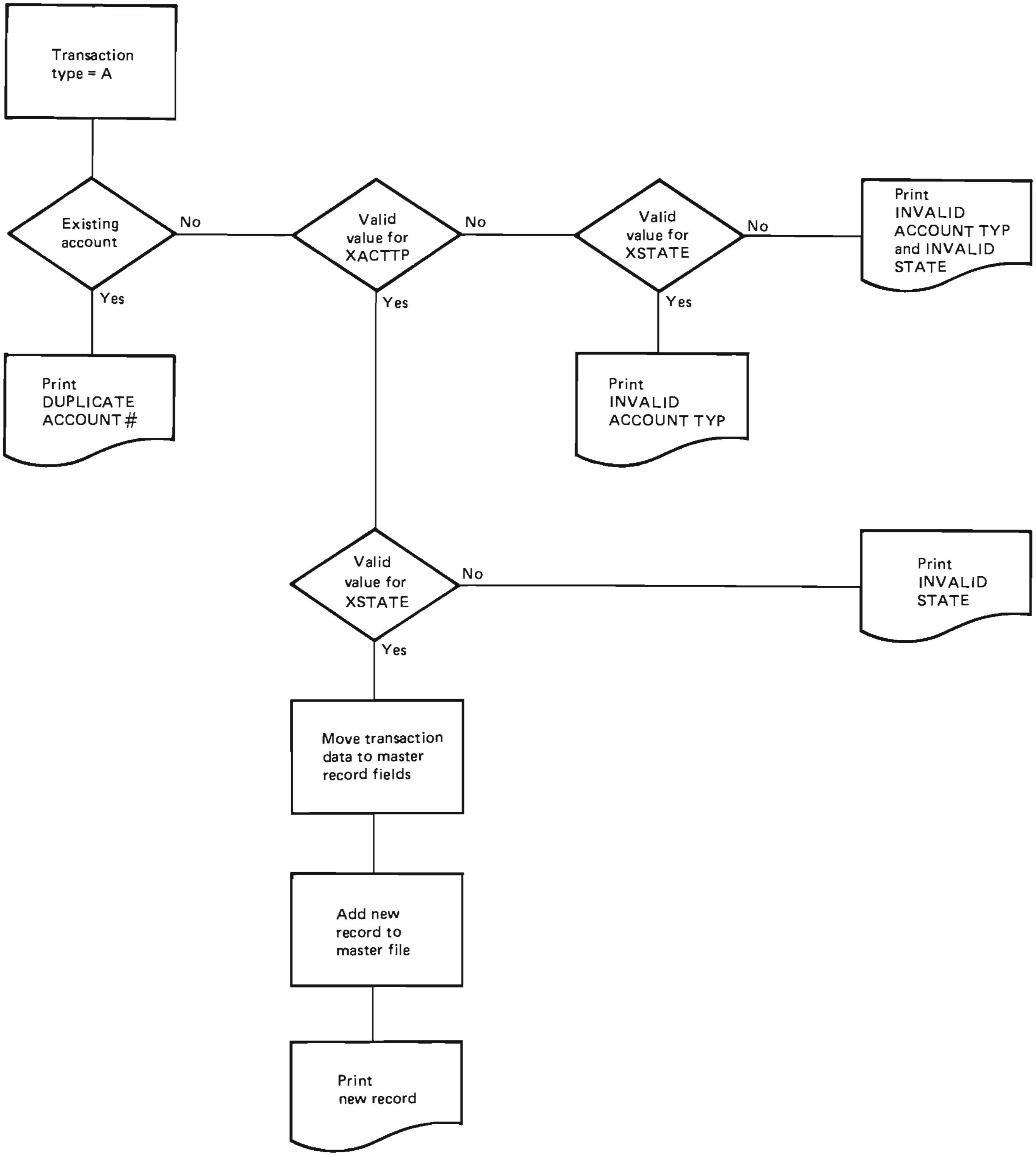
Page 6 of — Program Identification MLG31Ø

```
01 C*
02 C* MOVE TRANSACTION NAMES TO MASTER FILE NAMES
03 C*
04 C           MOVNEW    BEGSR
05 C                     MOVE XACTNM    ACTNUM
06 C                     MOVE XACTTP    ACTTYP
07 C                     MOVE XNAME     NAME
08 C                     MOVE XADDR     ADDR
09 C                     MOVE XCITY     CITY
10 C                     MOVE XSTATE    STATE
11 C                     MOVE XZIP      ZIP
12 C                     ENDSR
13 C*
14 C* CHECK CODES
15 C*
16 C           CHECK     BEGSR
17 C                     SETOF                     8Ø    8Ø = ERROR
18 C           XACTTP    LOKUPARCODE                   2Ø 2Ø = FOUND
19 C  N2Ø                SETON                     728Ø  72 = BAD ACTTYP
20 C           XSTATE    LOKUPARSTAT                   21 21 = FOUND
   C  N21                SETON                     738Ø  73 = BAD STATE
   C                     ENDSR
```

Figure 4-6 (Part 2 of 2). Calculation Specifications for MLG310 Maintenance Program

To add records, the transaction type code is A. The account number of the new record is compared with existing account numbers in the master file as shown in the following illustration.

Flowchart for Adding Records (Transaction Type A)


Transaction type = A
Existing account
No
Yes
Print DUPLICATE ACCOUNT #
Valid value for XACTTP
No
Yes
Valid value for XSTATE
No
Yes
Print INVALID ACCOUNT TYP and INVALID STATE
Print INVALID ACCOUNT TYP
Valid value for XSTATE
No
Yes
Print INVALID STATE
Move transaction data to master record fields
Add new record to master file
Print new record

If the new record's account number exists in the master file, indicator 61 is set on, an error message is printed, and the transaction is ignored.

The system will reject a request to add a nonunique key because the file is defined to contain unique keys (ACTNUM in this case). If there is a duplicate account number, the program terminates.

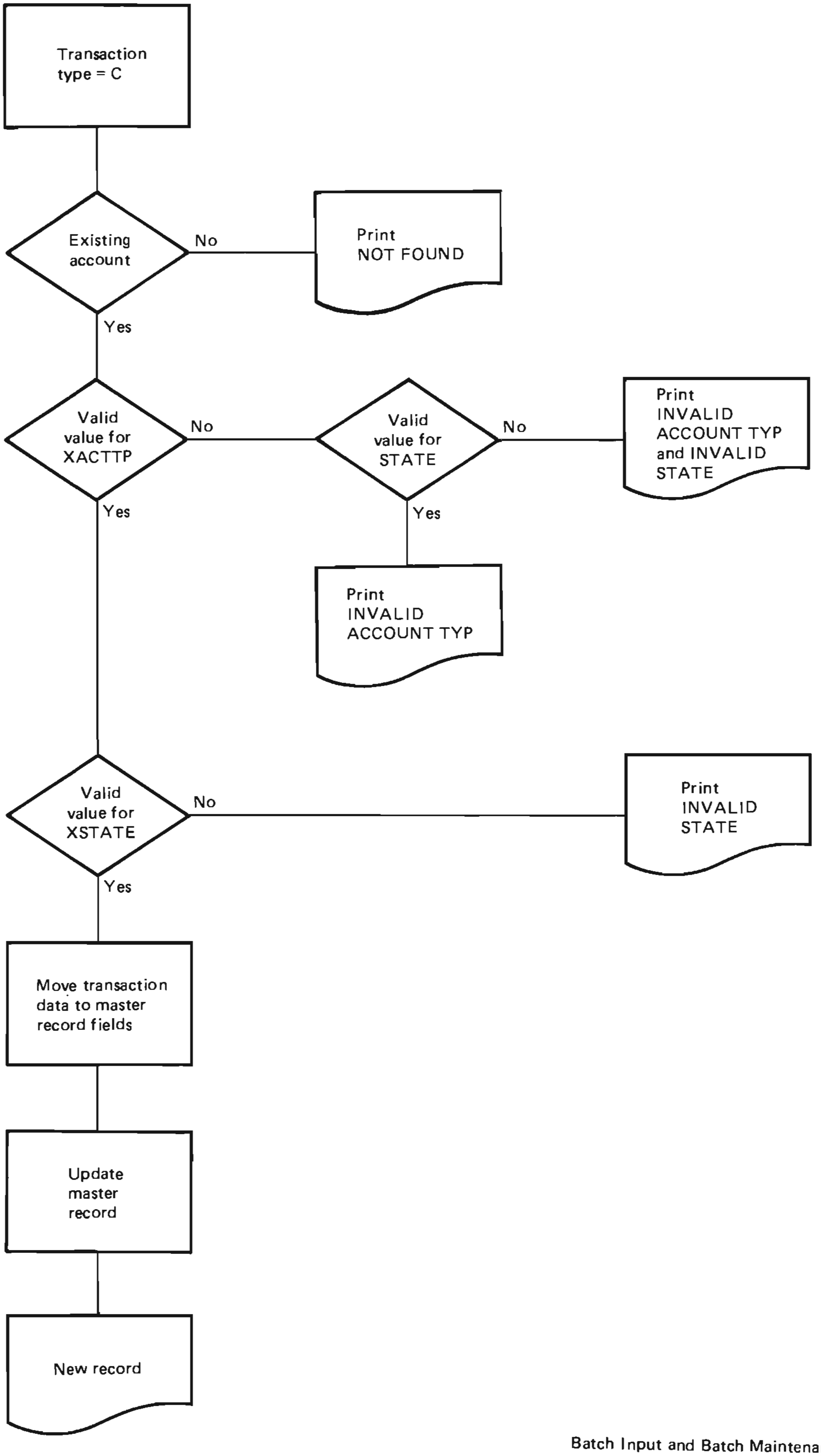
In this example, the program attempts to retrieve the record for the specified account number before adding the specified account number. In this way, the duplicate key error is avoided.

If the account number for the new record is unique, the EXSR CHECK statement invokes a subroutine to check the validity of the XACTTP and XSTATE fields by using LOKUP operations against arrays. Indicator 80 is set on, an error message is printed, and the transaction is ignored when one or both of these fields contains data that is not valid. If the data is valid, all of the transaction data fields are moved into the new master record fields (all fields in the master record must contain data) by the EXSR MOVNEW subroutine. The new master record is then written into the master file by the WRITE statement. No output specifications are required because the WRITE operation code supplies as output all fields for the externally described record.

To change records, the transaction type code is C. The account numbers in the master file are searched for the account number of the record to be changed as shown in the following illustration. If it is not found, indicator 61 is set on, an error message is printed, and the transaction is ignored.

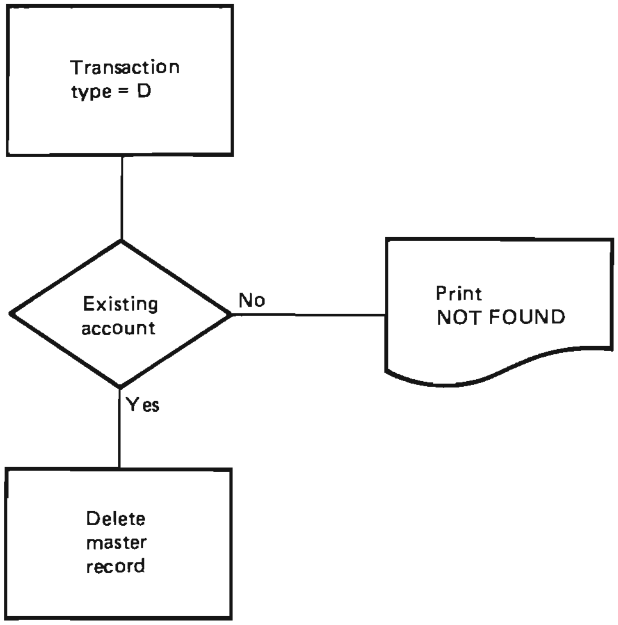
If the master record is found, the EXSR CHECK statement invokes a subroutine to check the validity of the XACTTP and XSTATE fields by using LOKUP operations against arrays. Indicator 80 is set on, an error message is printed, and the transaction is ignored when one or both of these fields contains data that is not valid. If the data is valid, the transaction data is moved into the master record field by the EXSR MOVNEW subroutine. The updated master record is then written into the master file by the UPDAT statement. The UPDAT operation code assembles all of the fields for the externally described record and replaces the existing fields.

Flowchart for Changing Records (Transaction Type C)



To delete records, the transaction type code is D. The account numbers in the master file are searched for the account number of the record to be deleted as shown in the following illustration.

Flowchart for Deleting Records (Transaction Type D)

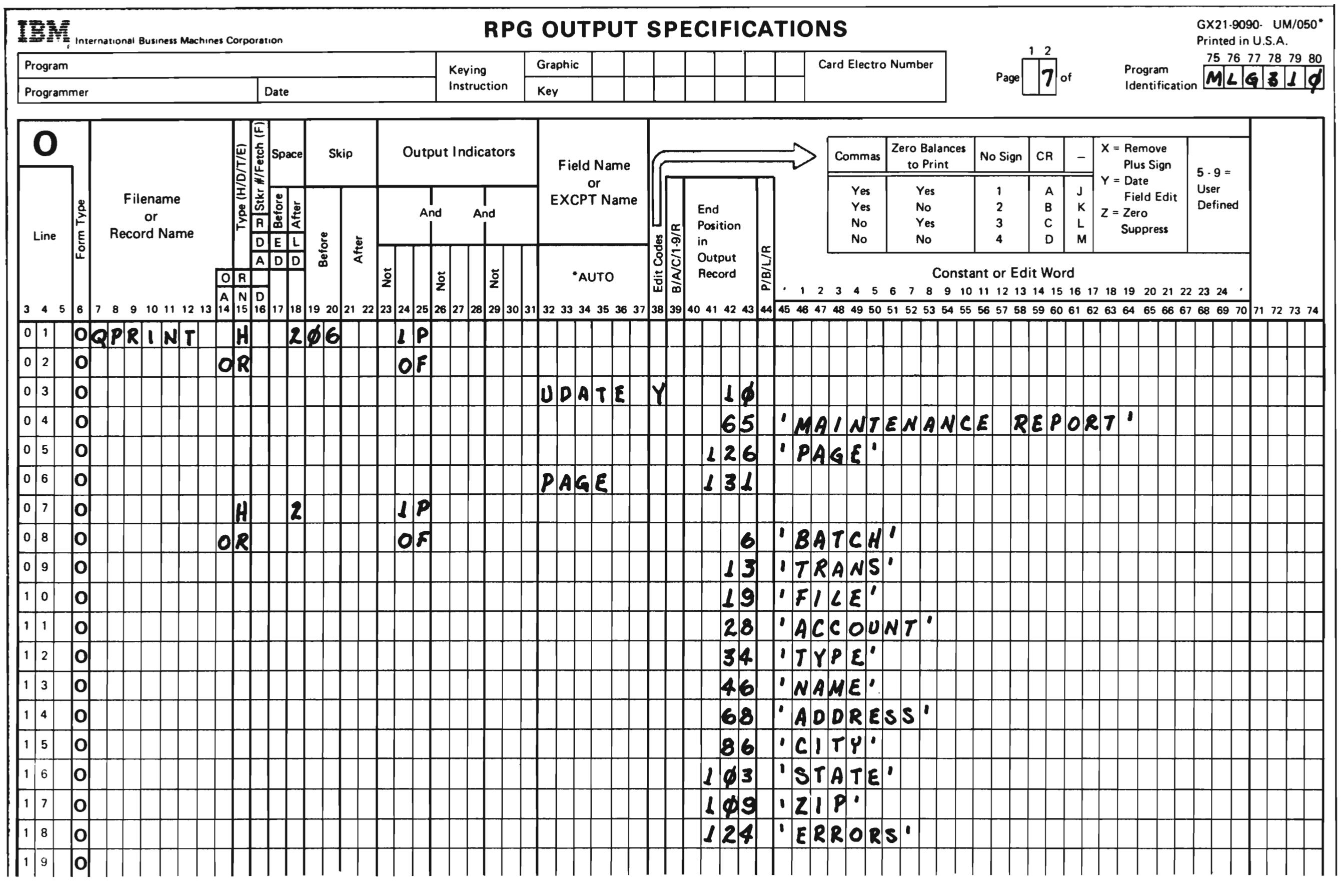


If the account number is not found, indicator 61 is set on, an error message is printed, and the transaction is ignored.

If the master record is found, it is deleted by the DELET statement. No program can retrieve this record after it is deleted.

If the specified transaction type is not one of the three valid codes, indicator 71 is set on and an error message is printed.

The *output specifications*, shown in Figure 4-7, print heading lines and format the output. If indicator 01 is on, the transaction record is printed. An error message, which lists any rejected transactions, is printed if an error indicator is on.



IBM International Business Machines Corporation

RPG OUTPUT SPECIFICATIONS

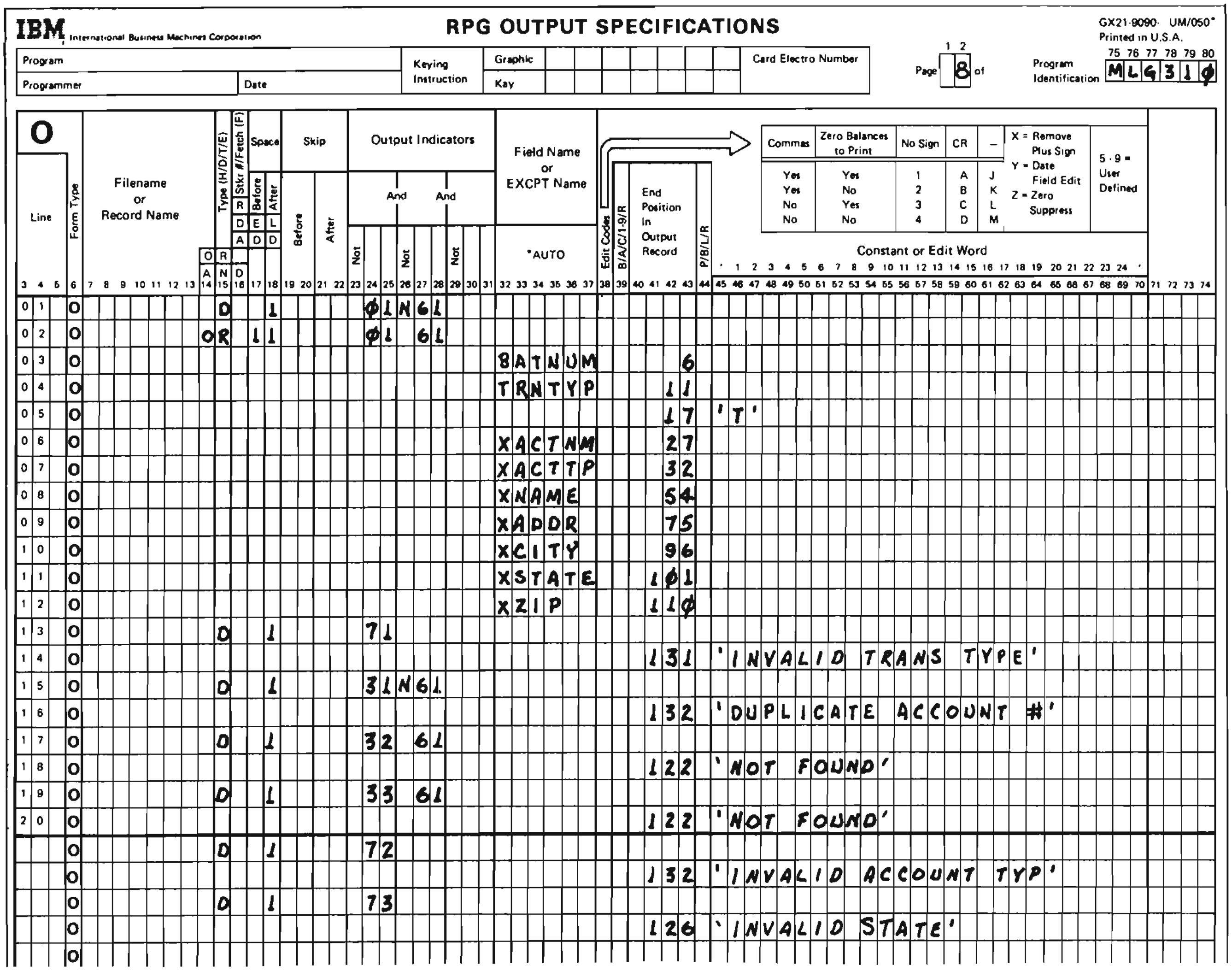
GX21-9090- UM/050*
Printed in U.S.A.

Program	
Programmer	
Date	
Keying Instruction	Graphic
	Key
Card Electro Number	
Page (1 2)	7 of
Program Identification (75 76 77 78 79 80)	MLG31Ø

Commas	Zero Balances to Print	No Sign	CR	–	X = Remove Plus Sign Y = Date Field Edit Z = Zero Suppress	5 - 9 = User Defined
Yes	Yes	1	A	J		
Yes	No	2	B	K		
No	Yes	3	C	L		
No	No	4	D	M		

Line	Form Type	Filename or Record Name	And/Or (14-16)	Type (H/D/T/E) (15)	Space Before (17)	Space After (18)	Skip Before (19-20)	Skip After (21-22)	Output Indicators (23-31)	Field Name or EXCPT Name (32-37)	Edit Codes (38)	B/A/C/1-9/R (39)	End Position in Output Record (40-43)	P/B/L/R (44)	Constant or Edit Word (45-70)
01	O	QPRINT		H		2	Ø6		1P						
02	O		OR						OF						
03	O									UDATE	Y		1Ø		
04	O												65		'MAINTENANCE REPORT'
05	O												126		'PAGE'
06	O									PAGE			131		
07	O			H		2			1P						
08	O		OR						OF				6		'BATCH'
09	O												13		'TRANS'
10	O												19		'FILE'
11	O												28		'ACCOUNT'
12	O												34		'TYPE'
13	O												46		'NAME'
14	O												68		'ADDRESS'
15	O												86		'CITY'
16	O												1Ø3		'STATE'
17	O												1Ø9		'ZIP'
18	O												124		'ERRORS'
19	O														

Figure 4-7 (Part 1 of 2). Output Specifications for MLG310 Maintenance Program



IBM International Business Machines Corporation

RPG OUTPUT SPECIFICATIONS

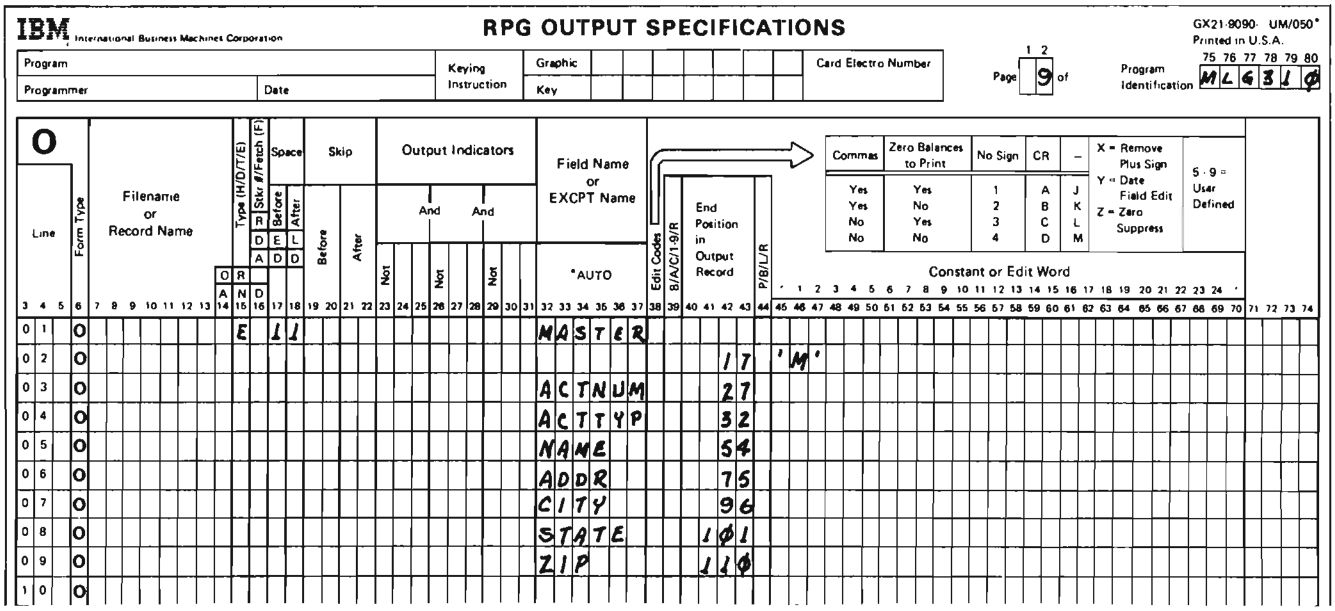
GX21-9090- UM/050* Printed in U.S.A.

Program | Programmer | Date | Keying Instruction | Graphic | Key | Card Electro Number

Page 8 of

Program Identification MLG31Ø

Line	Form Type	Filename or Record Name	Type (H/D/T/E)	Space Before	Space After	Output Indicators	Field Name or EXCPT Name	End Position in Output Record	Constant or Edit Word
01	O		D		1	Ø1 N61			
02	O		OR	1	1	Ø1 61			
03	O						BATNUM	6	
04	O						TRNTYP	11	
05	O							17	'T'
06	O						XACTNM	27	
07	O						XACTTP	32	
08	O						XNAME	54	
09	O						XADDR	75	
10	O						XCITY	96	
11	O						XSTATE	1Ø1	
12	O						XZIP	11Ø	
13	O		D		1	71			
14	O							131	'INVALID TRANS TYPE'
15	O		D		1	31 N61			
16	O							132	'DUPLICATE ACCOUNT #'
17	O		D		1	32 61			
18	O							122	'NOT FOUND'
19	O		D		1	33 61			
20	O							122	'NOT FOUND'
	O		D		1	72			
	O							132	'INVALID ACCOUNT TYP'
	O		D		1	73			
	O							126	'INVALID STATE'
	O								



IBM International Business Machines Corporation

RPG OUTPUT SPECIFICATIONS

GX21-9090- UM/050* Printed in U.S.A.

Program | Programmer | Date | Keying Instruction | Graphic | Key | Card Electro Number

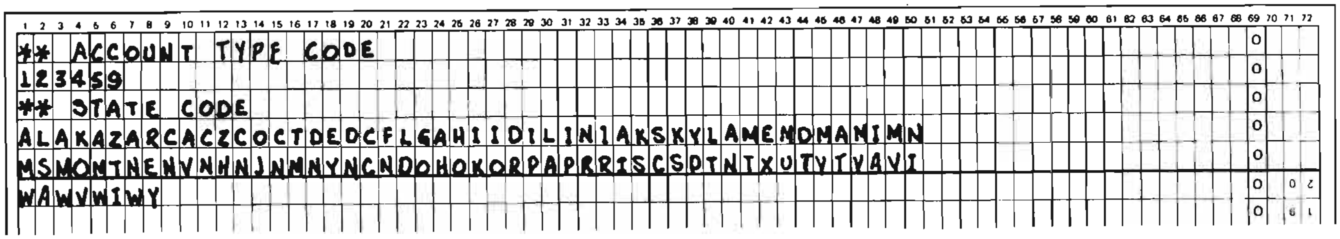
Page 9 of

Program Identification MLG31Ø

Line	Form Type	Filename or Record Name	Type (H/D/T/E)	Space Before	Space After	Output Indicators	Field Name or EXCPT Name	End Position in Output Record	Constant or Edit Word
01	O		E	1	1		MASTER		
02	O							17	'M'
03	O						ACTNUM	27	
04	O						ACTTYP	32	
05	O						NAME	54	
06	O						ADDR	75	
07	O						CITY	96	
08	O						STATE	1Ø1	
09	O						ZIP	11Ø	
10	O								

Figure 4-7 (Part 2 of 2). Output Specifications for MLG310 Maintenance Program

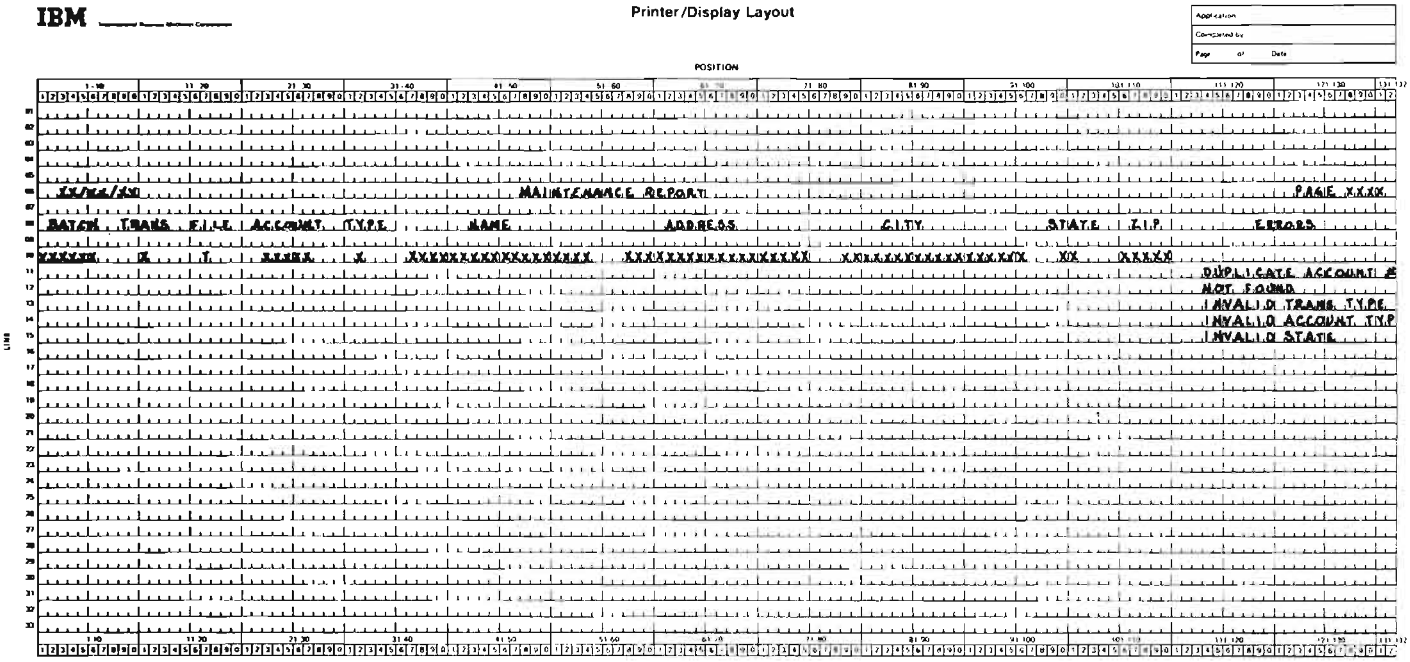
MASTER is the EXCPT name that is specified by the EXCPT operation code (see the previous discussion of the calculation specifications). The compile time arrays (Figure 4-8) are part of the program.



```
** ACCOUNT TYPE CODE
123459
** STATE CODE
ALAKAZARCACZCOCTDEDCFLGAHIIDILINIAKSKYLAMEMDMAMIMN
MSMOMTNENVNHNJNMNYNCNDOHOKORPAPRRISCSDTNTXUTVTVAVI
WAWVWIWY
```

Figure 4-8. Compile-Time Arrays for MLG310 Maintenance Program

The master record is printed as it existed before changes were made to it, and then the transaction record is printed. The printer layout for the maintenance program is shown in Figure 4-9.



IBM

Printer/Display Layout

```
06  XX/XX/XX                                  MAINTENANCE REPORT                                                                          PAGE XXXX
08 BATCH  TRANS  FILE  ACCOUNT  TYPE      NAME                ADDRESS              CITY                STATE  ZIP              ERRORS
10 XXXXXX   X      X    XXXXX    X     XXXXXXXXXXXXXXXXXXXX  XXXXXXXXXXXXXXXXXXXX  XXXXXXXXXXXXXXXXXXXX  XX    XXXXX
11                                                                                                                      DUPLICATE ACCOUNT #
12                                                                                                                      NOT FOUND
13                                                                                                                      INVALID TRANS TYPE
14                                                                                                                      INVALID ACCOUNT TYP
15                                                                                                                      INVALID STATE
```

Figure 4-9. Printer Layout for MLG310 Maintenance Program

Creating the Maintenance Program

The maintenance program is created by an input stream that is similar to the input stream previously used to create the label printing program, MLG520 (see Figure 3-6 in Chapter 3). In this case, MLG310 is specified as the program:

```
// JOB
CRTRPGPGM  PGM(MLG310)  SRCFILE(SRC310)
// DATA  FILE(SRC310)  FILETYPE(*SRC)
    .
    .  (RPG III source)
    .
// ENDJOB
```

This input stream is a file on diskette with a label name of SOURCE3. To create MLG310, issue the STRDKTRDR command with SOURCE3 specified as the label.

The RPG III source for MLG310 is shown in Figures 4-3 through 4-8.

Executing the Maintenance Program

The maintenance program is executed by an input stream that is similar to the input stream previously used to execute the label printing program, MLG520 (see Figure 3-8 in Chapter 3). In this case, MLG310 is the program called and the data is transaction records (the transaction records are described earlier in this chapter under *Using Transaction Records*):

```
// JOB
CALL MLG310  /* Mailing List Maintenance Program */
// DATA FILE(INPUT)
    .
    .  (transaction records)
    .
// ENDJOB
```

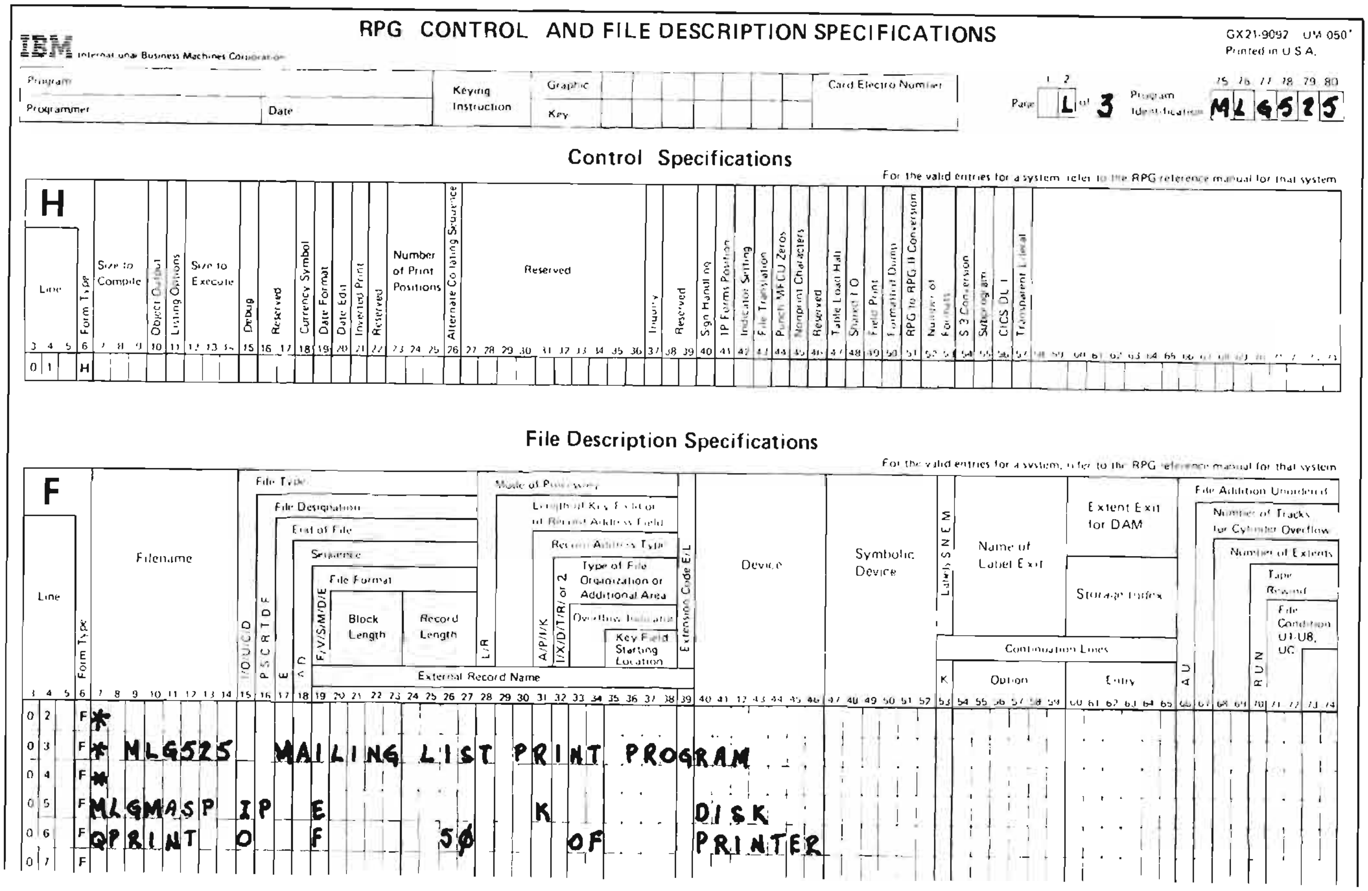
The input stream is a file on diskette with a label name of MLGMAINT. To execute MLG310, issue the STRDKTRDR command with MLGMAINT specified as the label.

When the program executes, the master file is updated. New records are written at the end of the file, but the access path on account number is modified immediately so that the records can be accessed in the correct order (they are not moved within the file). Records that have been changed are updated in place. They are not moved within the file. Records that have been deleted can never be accessed again. A deleted record occupies space on the system; however, this space can be reclaimed by reorganizing the file. For information on how to do this, refer to the *CPF Programmer's Guide.*

MAILING LIST PROGRAM

The mailing list program, MLG525, prints the labels for a specified mailing list. The program is written in RPG III and uses the RPG program cycle to read and print mailing list records.

The source for this program is shown in Figures 4-10 through 4-12. The input file for this program is MLGMASP. This approach assumes that the sequence and selection criteria for each mailing list will vary. When the program is executed, the Override with Data Base File (OVRDBF) command is specified so that the program uses a logical file to access the data in MLGMASP. The override command specifies the name used in the program and the file to be used on the system. The same program is used regardless of the mailing list sequence. The override is needed because the program is compiled to execute against the physical data, but the data needs to be in a different logical sequence. Overrides can also be used to redirect data and temporarily modify the attributes of the file.



IBM International Business Machines Corporation

RPG CONTROL AND FILE DESCRIPTION SPECIFICATIONS

GX21-9092 UM 050* Printed in U.S.A.

Page 1 of 3 Program Identification MLG525

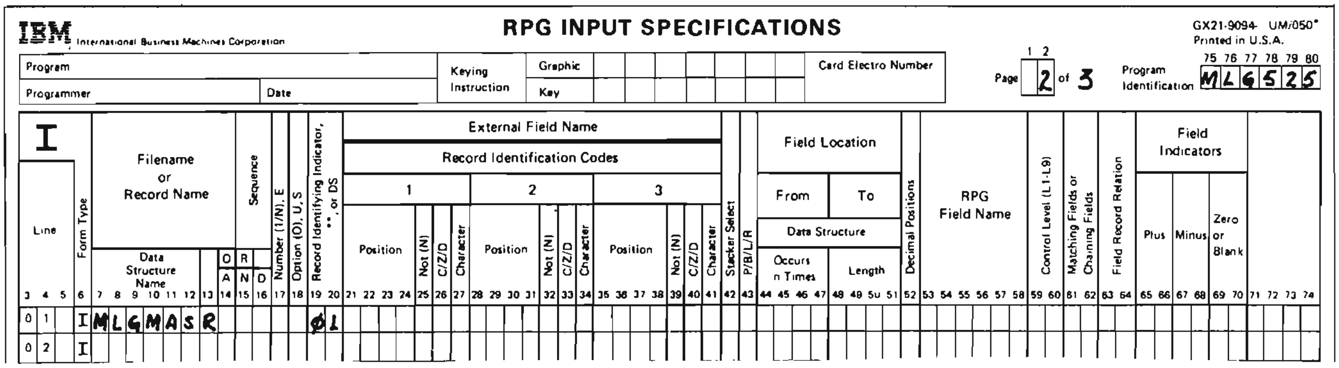
Control Specifications

```
01 H
```

File Description Specifications

```
02 F*
03 F* MLG525  MAILING LIST PRINT PROGRAM
04 F*
05 FMLGMASP IP  E           K        DISK
06 FQPRINT  O   F      5Ø     OF     PRINTER
07 F
```

Figure 4-10. Control and File Description Specifications for MLG525 Print Program



IBM International Business Machines Corporation

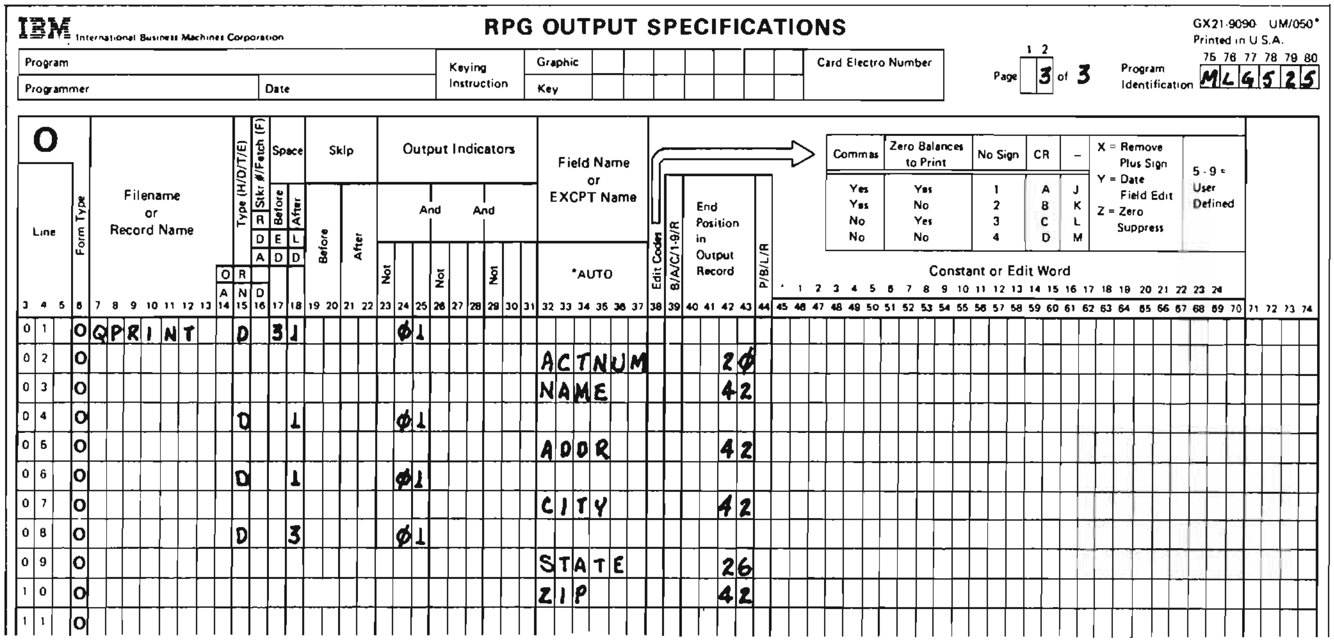
RPG INPUT SPECIFICATIONS

GX21-9094- UM/050* Printed in U.S.A.

Page 2 of 3 Program Identification ML6525

Line	Form Type	Filename or Record Name	Sequence	Number (1/N), E	Option (O), U, S	Record Identifying Indicator, **, or DS
01	I	MLGMASR				ø1
02	I					

Figure 4-11. Input Specifications for MLG525 Print Program



IBM International Business Machines Corporation

RPG OUTPUT SPECIFICATIONS

GX21-9090 UM/050* Printed in U.S.A.

Page 3 of 3 Program Identification ML6525

Line	Form Type	Filename or Record Name	Type (H/D/T/E)	Space Before	Space After	Output Indicators	Field Name or EXCPT Name	End Position in Output Record
01	O	QPRINT	D	3	1	ø1		
02	O						ACTNUM	2ø
03	O						NAME	42
04	O		D		1	ø1		
05	O						ADDR	42
06	O		D		1	ø1		
07	O						CITY	42
08	O		D		3	ø1		
09	O						STATE	26
10	O						ZIP	42
11	O							

Figure 4-12. Output Specifications for MLG525 Print Program

The record format (MLGMASR) in the physical file (MLGMASP) is the same as that in the logical file (MLGMASL). Consequently, the System/38 *level checking* function is passed successfully. Level checking ensures that the format that was copied into the program at compilation time is the same format that the program uses at execution time. If the formats are not the same, the System/38 terminates the program when it attempts to open the file.

The E in position 19 of the file description specification, shown in Figure 4-10, indicates that an externally described file, MLGMASP, is used. The RPG III compiler copies the field descriptions from MLGMASP at compilation time, so the fields do not have to be described in the RPG program. However, the input specification, shown in Figure 4-11, is used to add information to the record format description that was copied at compilation time. This input specification assigns the record identifying indicator 01 to the record format, MLGMASR (the record format name, MLGMASR, is used on the input specification instead of the file name). At execution time, the system retrieves a record from the file and provides the format name of the record it read to RPG III. If the format name is MLGMASR, RPG III sets on the record identifying indicator 01. This indicator is then used to condition the output operations.

Creating the Mailing List Program

The mailing list print program is created by an input stream that is similar to the input stream previously used to create the label printing program, MLG520 (see Figure 3-6 in Chapter 3). In this case, MLG525 is specified as the program. The input stream is a file on diskette with a label name of SOURCE4. To create MLG525, issue the STRDKTRDR command with SOURCE4 specified as the label.

The RPG III source statements shown in Figures 4-10 through 4-12 follow the // DATA statement in the input stream.

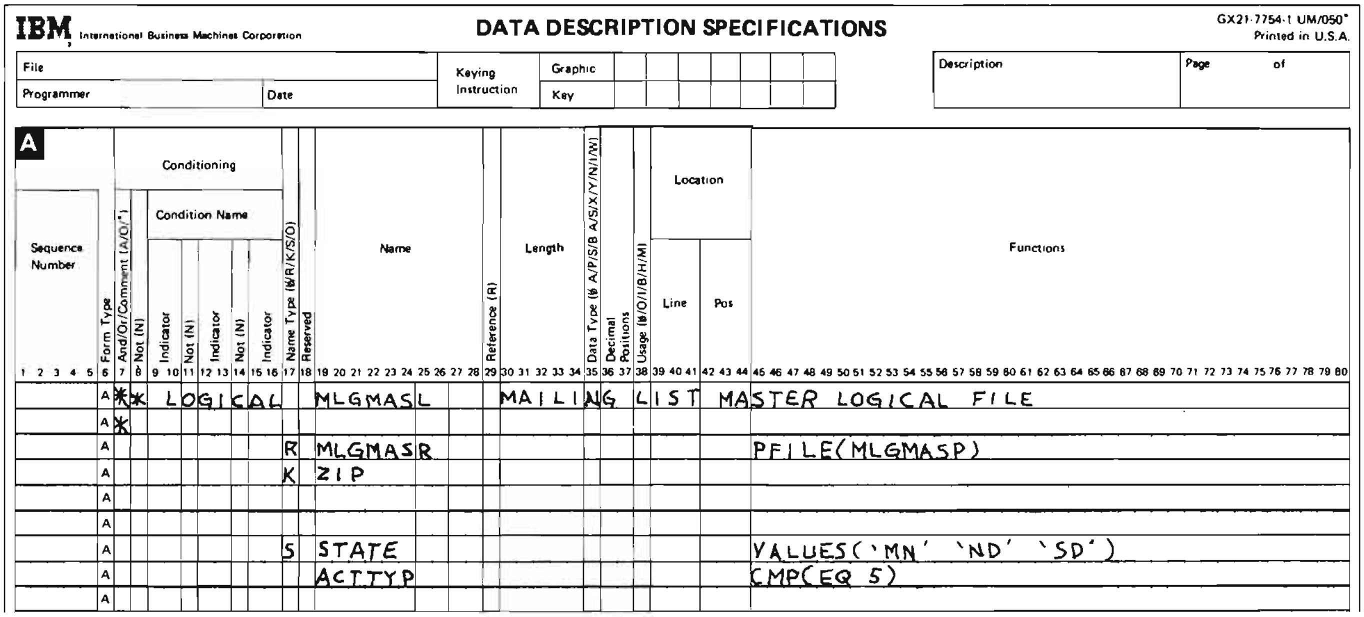
Executing the Mailing List Program through a Logical File

If you have requests for several different mailing lists, you can create a logical file to access the data for each mailing list and then delete the logical file after it is used. An override (OVRDBF command) can be used to temporarily substitute this logical file for the file specified in the program.

The logical file used in this approach is named MLGMASL and is designed to do the following:

- Select only records from the states MN, ND, and SD
- Select only private accounts (ACTTYP=5)
- Sequence the access path by zip code

Figure 4-13 shows the DDS for this file. The record format (MLGMASR) used in the physical file is also used in this file. The physical file is identified by the PFILE keyword. Because the same record format name is used and no fields are defined, all fields defined in the physical file are automatically defined for this logical file.



IBM International Business Machines Corporation

DATA DESCRIPTION SPECIFICATIONS

GX21-7754-1 UM/050*
Printed in U.S.A.

File
Programmer
Date
Keying Instruction
Graphic
Key
Description
Page of

A
Sequence Number
Form Type
And/Or/Comment (A/O/*)
Conditioning
Condition Name
Not (N)
Indicator
Not (N)
Indicator
Not (N)
Indicator
Name Type (b/R/K/S/O)
Reserved
Name
Reference (R)
Length
Data Type (b A/P/S/B A/S/X/Y/N/I/W)
Decimal Positions
Usage (b/O/I/B/H/M)
Location
Line
Pos
Functions
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80

```
     A** LOGICAL  MLGMASL     MAILING LIST MASTER LOGICAL FILE
     A*
     A          R MLGMASR                   PFILE(MLGMASP)
     A          K ZIP
     A
     A
     A          S STATE                     VALUES('MN' 'ND' 'SD')
     A            ACTTYP                    CMP(EQ 5)
     A
```

Figure 4-13. DDS for MLGMASL File

The key field (K in position 17) specifies that the file is to be sequenced by zip code.

The select logic (S in position 17) specifies the values for the STATE field (which can be MN, ND, or SD) and that the ACTTYP field must equal 5. Only records meeting both of these selection criteria will be on the access path; consequently, only those records will be read by the program.

The following input stream creates the logical file definition for MLGMASL and then executes the mailing list print program through the logical file:

```
// JOB
CRTLF FILE(MLGMASL) SRCFILE(QINLINE)
// DATA FILETYPE(*SRC)
  •
  •  (DDS source)
  •
OVRDBF FILE(MLGMASP) TOFILE(MLGMASL)
CALL MLG525 /* PRINT MAILING LABELS */
DLTF MLGMASL
// ENDJOB
```

This input stream is a file on diskette with a label name of SOURCE2. To create MLGMASL and execute MLG525, issue the STRDKTRDR command with SOURCE2 specified as the label. The input stream is similar to those previously shown (see, for example, Figure 3-8 in Chapter 3 and Figure 4-1). The CRTLF command creates a logical file named MLGMASL as described by the DDS source, which is shown in Figure 4-13. The create command builds an access path over the physical data according to the selection and sequencing criteria. It does not duplicate the data.

The OVRDBF command specifies that the file name MLGMASP is overridden to use the file MLGMASL. The program uses the OVRDBF command to access the logical file and print the labels for each record on the access path. The DLTF command deletes the MLGMASL logical file, including its access path, after the MLG525 program has printed master records from MLGMASP as defined in MLGMASL.

RECOVERY CONSIDERATIONS

Recovery from error situations likely to be encountered in applications was introduced in the section, *General Recovery Considerations*, in Chapter 2. The following specific recovery considerations apply to the application approach in this chapter.

If incorrect data has been entered for a transaction, the entire transaction should be reentered with correct data by using the maintenance program MLG310.

If there is a system failure while the maintenance program is executing, the current batch of transactions should be reprocessed in most situations. The maintenance program rejects invalid transactions, such as a request to add a record with an account number that already exists in the master file. If a system failure occurs when you are not executing the maintenance program, no recovery is normally needed.

To allow you to recover damaged objects (assuming you are maintaining the master file with one or more batches per day), the master file could be saved every 2 to 3 days and the transactions should be saved from the last backup. If the damaged master file is not usable, the backup can be restored and the saved transactions can be reprocessed. You should keep both the current backup and at least one earlier level of backup. Because you recover by reprocessing the transaction, a specific entry for the FRCRATIO parameter is not needed for the data base files used in this approach.

Chapter 5. Programmer Use of a Work Station

OVERVIEW

In Chapter 4, input streams including the transaction records were read from diskette to execute batch jobs. Normally, you can be more productive if you perform these functions directly from a work station. The remainder of this publication assumes that you are working from a display station with a 24-line screen (such as a 5251 Model 11 or 12).

Many of the functions available on the work station are used in the remaining application approaches. This chapter introduces the following work station functions and shows how they are used for application development:

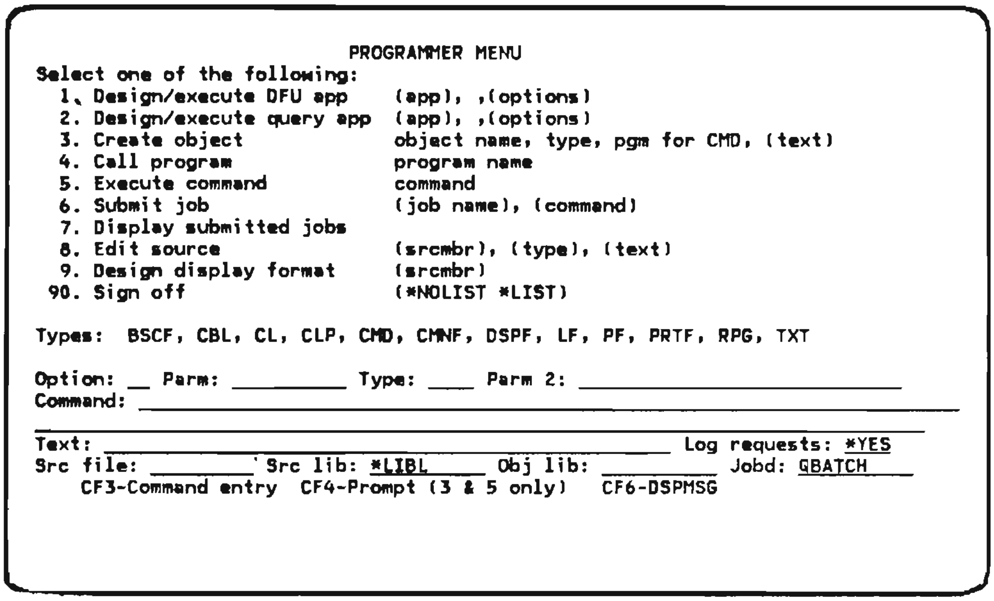
- Programmer menu
- Command prompting
- Source entry utility (SEU)

You can use the System/38 work station to transmit information to or receive information from the computer as you perform your job. For example, you can use the work station to do the following:

- Enter or change source
- Request compilations
- Review the compilation results
- Enter input streams
- Enter single batch jobs
- Request various types of system status (such as a displayed list of active jobs)
- Execute programs
- Perform debug functions

PROGRAMMER MENU

If you sign on the system with the password for the QPGMR user profile, you receive the programmer menu (shown below) as your basic working display.

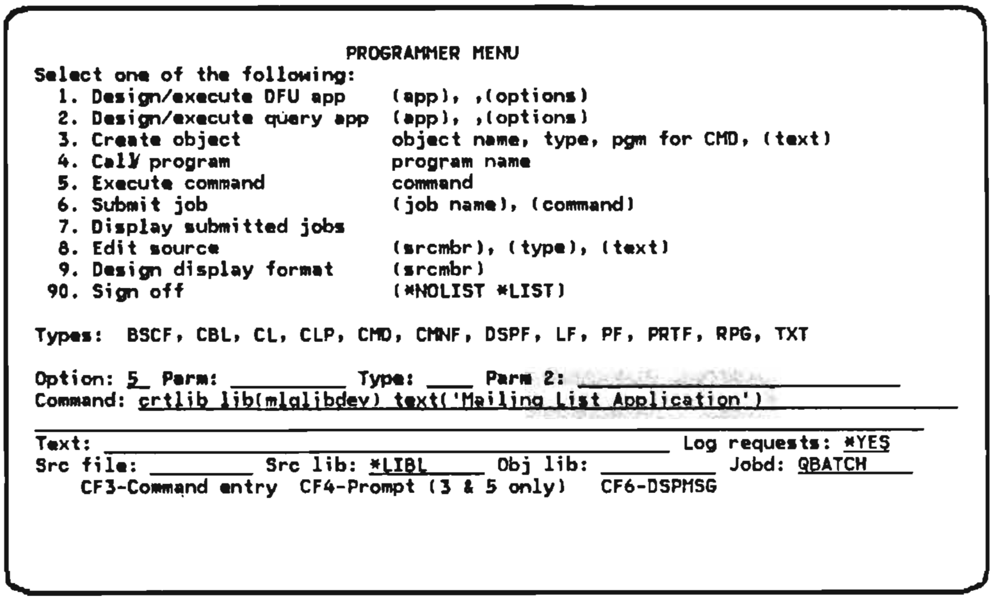


```
                              PROGRAMMER MENU
Select one of the following:
  1. Design/execute DFU app    (app), ,(options)
  2. Design/execute query app  (app), ,(options)
  3. Create object             object name, type, pgm for CMD, (text)
  4. Call program              program name
  5. Execute command           command
  6. Submit job                (job name), (command)
  7. Display submitted jobs
  8. Edit source               (srcmbr), (type), (text)
  9. Design display format     (srcmbr)
 90. Sign off                  (*NOLIST *LIST)

Types:  BSCF, CBL, CL, CLP, CMD, CMNF, DSPF, LF, PF, PRTF, RPG, TXT

Option: __ Parm: __________ Type: ____ Parm 2: ____________________________
Command: ____________________________________________________________________
_____________________________________________________________________________
Text: ___________________________________________________ Log requests: *YES
Src file: _________ Src lib: *LIBL_____ Obj lib: __________ Jobd: QBATCH____
    CF3-Command entry  CF4-Prompt (3 & 5 only)   CF6-DSPMSG
```

The programmer menu can be a shortcut to requesting functions that are used frequently in performing programmer tasks. By generating the commands needed to perform these programmer tasks, the programmer menu helps to eliminate repetitive keying and some errors. For the programmer menu to be effective, your system needs to have the IDU program product or an equivalent installed. Also, a 5251 Display Station Model 11 or 12, or equivalent with a 24-line screen, must be used because the programmer menu requires a 24-line screen.

When you request an option other than option 5, you do not need to key in the command; you simply select the desired option and key in only the essential information.

Option 5 allows you to enter a command name and parameters to request a function that is not listed as an option on the menu. The following example shows how the Create Library (CRTLIB) command can be executed:



```
                          PROGRAMMER MENU
Select one of the following:
  1. Design/execute DFU app    (app), ,(options)
  2. Design/execute query app  (app), ,(options)
  3. Create object             object name, type, pgm for CMD, (text)
  4. Call program              program name
  5. Execute command           command
  6. Submit job                (job name), (command)
  7. Display submitted jobs
  8. Edit source               (srcmbr), (type), (text)
  9. Design display format     (srcmbr)
 90. Sign off                  (*NOLIST *LIST)

Types:  BSCF, CBL, CL, CLP, CMD, CMNF, DSPF, LF, PF, PRTF, RPG, TXT

Option: 5_ Parm: __________ Type: ____ Parm 2: ____________________________
Command: crtlib lib(mlglibdev) text('Mailing List Application')____________
___________________________________________________________________________
Text: _________________________________________________ Log requests: *YES
Src file: _________ Src lib: *LIBL_____ Obj lib: __________ Jobd: QBATCH____
    CF3-Command entry  CF4-Prompt (3 & 5 only)   CF6-DSPMSG
```

For detailed information on how to use the programmer menu, refer to the *Programmer's/User's Work Station Guide.*

To take full advantage of the programmer menu, you may need to establish the proper environment for application development and maintenance. For example, you may need to create specific types of files for an application, and you may want to group those files in a special library for convenient access. The procedures and considerations for creating and using libraries and files are discussed in Chapter 6.

COMMAND PROMPTING

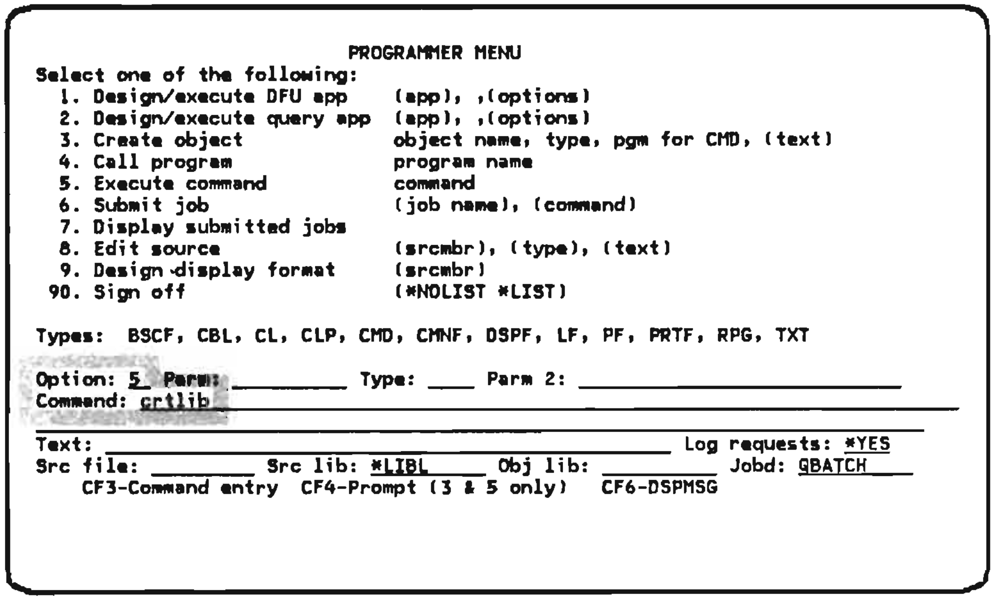
Each command on System/38 has a prompt display that helps you enter the command. You can request command prompting for options 3 and 5 on the programmer menu by using the CF4 key as follows:

- Select option 3 and specify an object name and type; then press CF4. You receive a prompt display for the appropriate command to create the object you specified. The prompt lists all required and optional parameters of the command and shows the object name and type you provided. If you key in other parameter values on the programmer menu before pressing CF4, those values are also shown on the prompt.

- Select option 5 and specify at least a command name; then press CF4. You receive a prompt display that lists the required and optional parameters for the specified command. If you key in parameter values on the programmer menu before pressing CF4, those values are also shown on the prompt.

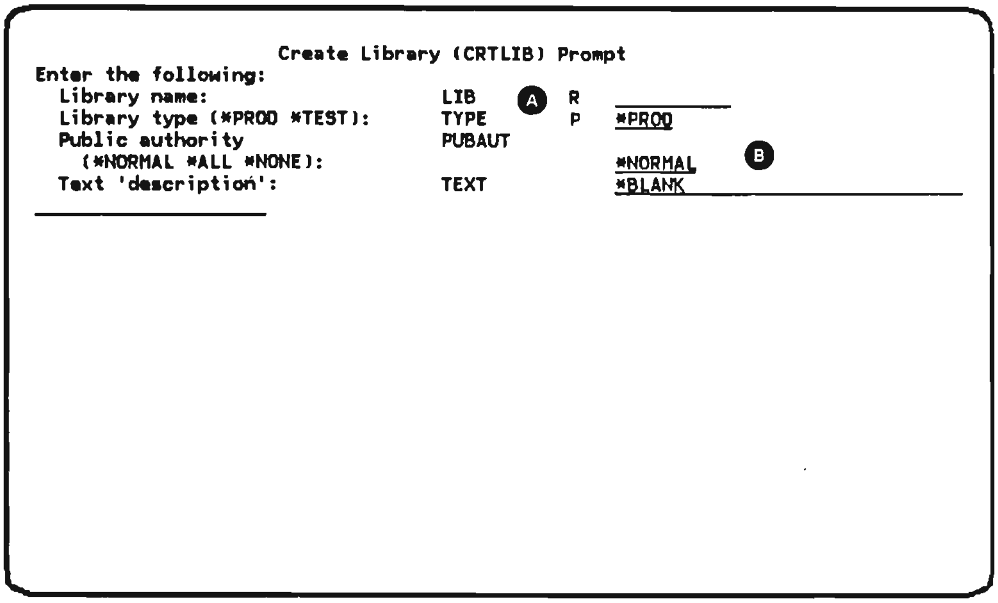
- Select option 5 and press CF4 immediately *without* specifying a command name. You receive a set of menus that help you select the appropriate command to perform a desired function. When you select a specific command on the menus, you receive the prompt display for that command.

For example, to request prompting for the CRTLIB command, first select option 5 and specify CRTLIB:



```
                              PROGRAMMER MENU
Select one of the following:
  1. Design/execute DFU app    (app), ,(options)
  2. Design/execute query app  (app), ,(options)
  3. Create object             object name, type, pgm for CMD, (text)
  4. Call program              program name
  5. Execute command           command
  6. Submit job                (job name), (command)
  7. Display submitted jobs
  8. Edit source               (srcmbr), (type), (text)
  9. Design display format     (srcmbr)
 90. Sign off                  (*NOLIST *LIST)

Types:  BSCF, CBL, CL, CLP, CMD, CMNF, DSPF, LF, PF, PRTF, RPG, TXT

Option: 5_ Parm: __________ Type: ____ Parm 2: ____________________________
Command: crtlib_________________________________________________________________
_____________________________________________________________________________
Text: _____________________________________________ Log requests: *YES
Src file: _________ Src lib: *LIBL_____ Obj lib: __________ Jobd: QBATCH____
    CF3-Command entry  CF4-Prompt (3 & 5 only)   CF6-DSPMSG
```

Then press CF4. The following display appears:



```
                    Create Library (CRTLIB) Prompt
Enter the following:
  Library name:                     LIB         R   __________
  Library type (*PROD *TEST):       TYPE        P   *PROD
  Public authority                  PUBAUT
    (*NORMAL *ALL *NONE):                           *NORMAL
  Text 'description':               TEXT            *BLANK
```

A Parameters that you are required to use are designated by R; you must key in a valid value. If R is not shown, the parameter is optional. P is shown if the optional parameter can be coded positionally (without a keyword).

B When there is a default for a parameter, the default value is automatically displayed; however, you can replace the default value with another valid value. To request a listing of valid values, enter a ? instead of a value.

Command prompting helps you enter correct information and minimize keystrokes. For detailed information on using prompting functions, refer to the *Programmer's/User's Work Station Guide.*

SOURCE ENTRY UTILITY

In addition to entering source from diskette, you can enter and edit source and create source members from a work station using the *source entry utility* (SEU), which is part of the Interactive Data Base Utilities (IDU). SEU operates on *members* of a source file.

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

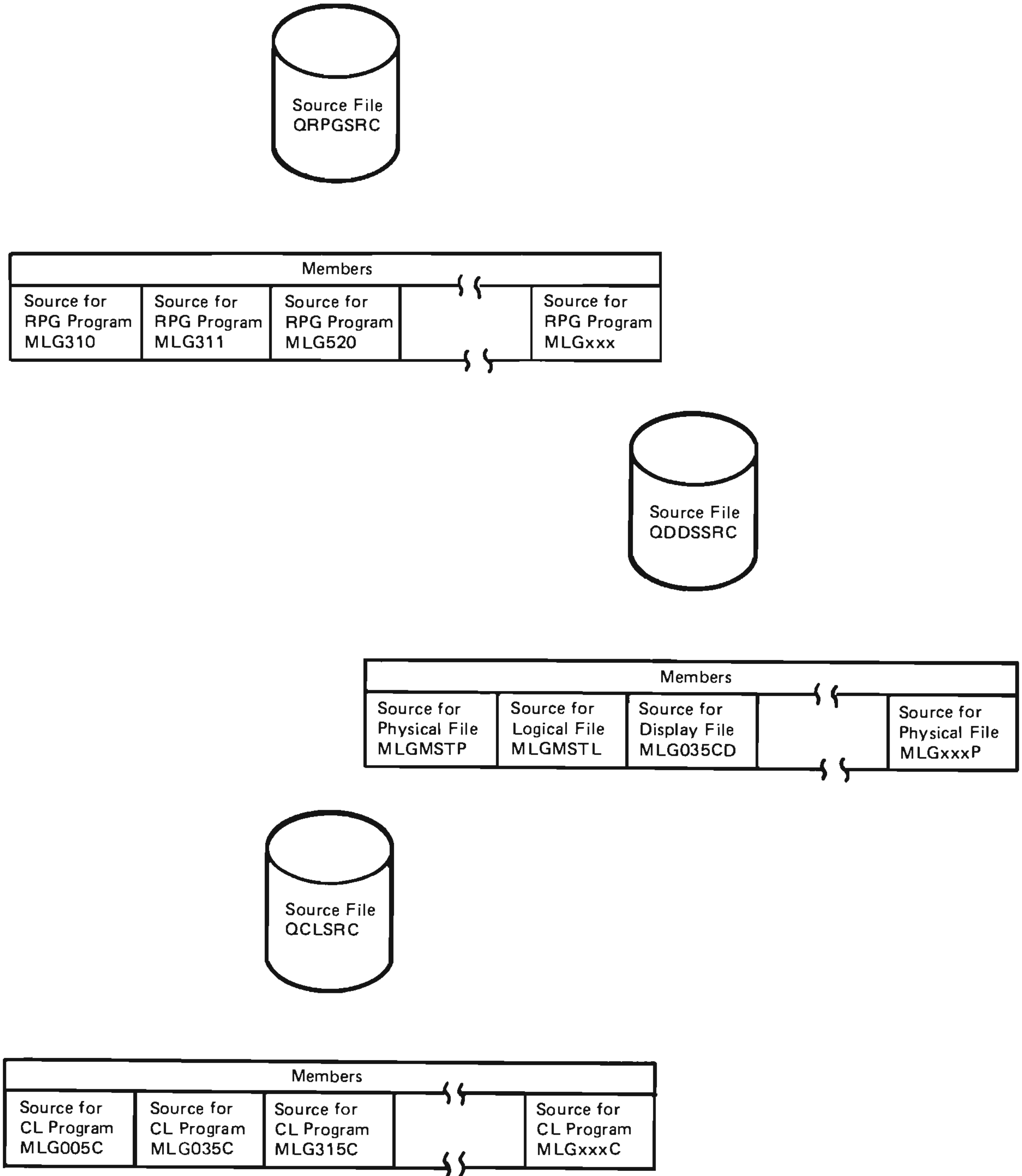
When you use SEU, source records are entered into a member of a data base file. IBM supplies several source files, each for a different type of source. The four source files used in this example are:

- QRPGSRC for RPG III source
- QDDSSRC for DDS source
- QCLSRC for CL source
- QUDSSRC for IDU source

A source file can have multiple members, with each member containing a separate set of source statements, as shown in the following illustration.

In this illustration, each member of file QRPGSRC contains the source statements for a different RPG program. Similarly, each member of the file QCLSRC contains the source statements for a different CL program, and each member of the file QDDSSRC contains the source statements for a different file.

The name of the member is the same as the name of the associated program or file, unless you specify a different name when you create the program or file from the source.


Source File
QRPGSRC
Members
Source for RPG Program MLG310
Source for RPG Program MLG311
Source for RPG Program MLG520
Source for RPG Program MLGxxx
Source File
QDDSSRC
Members
Source for Physical File MLGMSTP
Source for Logical File MLGMSTL
Source for Display File MLG035CD
Source for Physical File MLGxxxP
Source File
QCLSRC
Members
Source for CL Program MLG005C
Source for CL Program MLG035C
Source for CL Program MLG315C
Source for CL Program MLGxxxC

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

- 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.

- 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 TAG) are not detected.

- While a source statement is being entered, the formats supplied by IBM for source types, such as DDS and RPG, can be used to enter data only in fields that apply to a format. This method of entering data can prevent both positional and field-content keying errors.

- 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

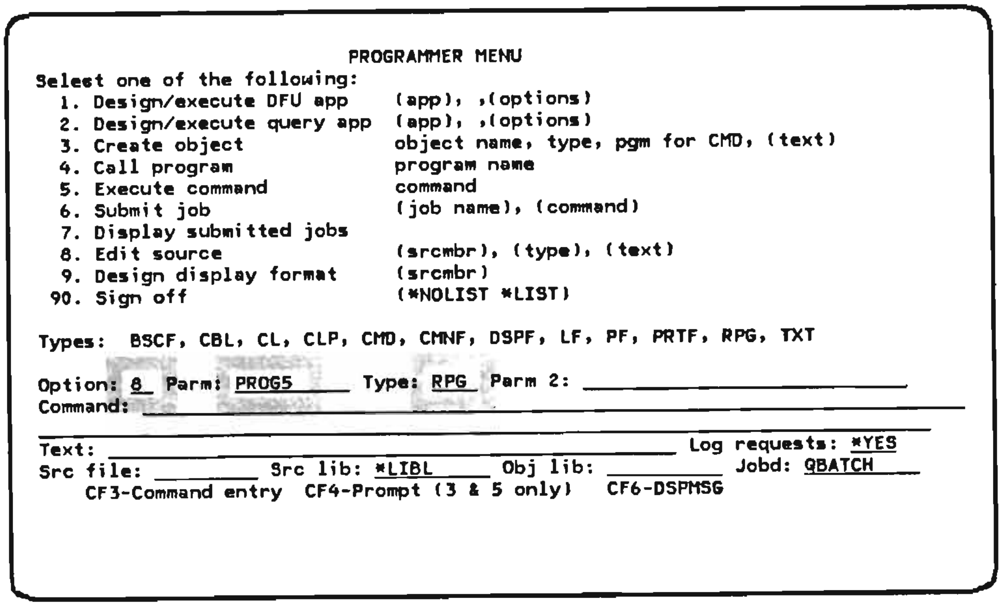
SEU can be invoked by using the programmer menu as follows:

- Select option 8 to create a new source member or change an existing source member.

- Specify the name of the source member (srcmbr) to be created or changed.

- Specify the type, such as RPG or CLP.

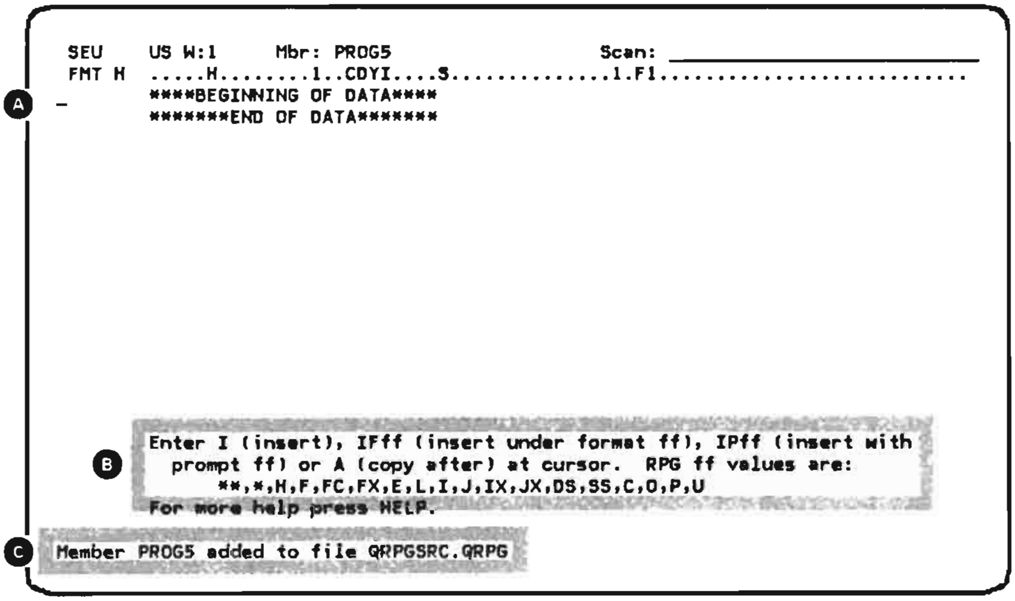
Assume that you want to add (create) a *new* source member for an RPG program PROG5. To request SEU, you select option 8 and enter the member name and type on the programmer menu:



```
                            PROGRAMMER MENU
Select one of the following:
  1. Design/execute DFU app    (app), ,(options)
  2. Design/execute query app  (app), ,(options)
  3. Create object             object name, type, pgm for CMD, (text)
  4. Call program              program name
  5. Execute command           command
  6. Submit job                (job name), (command)
  7. Display submitted jobs
  8. Edit source               (srcmbr), (type), (text)
  9. Design display format     (srcmbr)
 90. Sign off                  (*NOLIST *LIST)

Types:  BSCF, CBL, CL, CLP, CMD, CMNF, DSPF, LF, PF, PRTF, RPG, TXT

Option: 8_ Parm: PROG5____  Type: RPG_ Parm 2: ____________________
Command: ___________________________________________________________
____________________________________________________________________
Text: _____________________________________  Log requests: *YES
Src file: ________ Src lib: *LIBL____ Obj lib: _________ Jobd: QBATCH___
    CF3-Command entry  CF4-Prompt (3 & 5 only)   CF6-DSPMSG
```

Through SEU, a member PROG5 is added to the appropriate source file for the type of source you specified. Because you specified RPG for the source type in this example, the member is added to the QRPGSRC file. However, if you had specified a different source file (in the *Src file* field), the member would have been added to that file instead of QRPGSRC, even though you specified RPG for the type.

You next receive an SEU edit display on which you can enter the source statements for the new member:



```
SEU     US W:1     Mbr: PROG5                  Scan: ____________________
FMT H   .....H........1..CDYI....S..............1.F1...........................
 _       ****BEGINNING OF DATA****
         ********END OF DATA********



        Enter I (insert), IFff (insert under format ff), IPff (insert with
          prompt ff) or A (copy after) at cursor.  RPG ff values are:
              **,*,H,F,FC,FX,E,L,I,J,IX,JX,DS,SS,C,O,P,U
        For more help press HELP.
Member PROG5 added to file QRPGSRC.QRPG
```

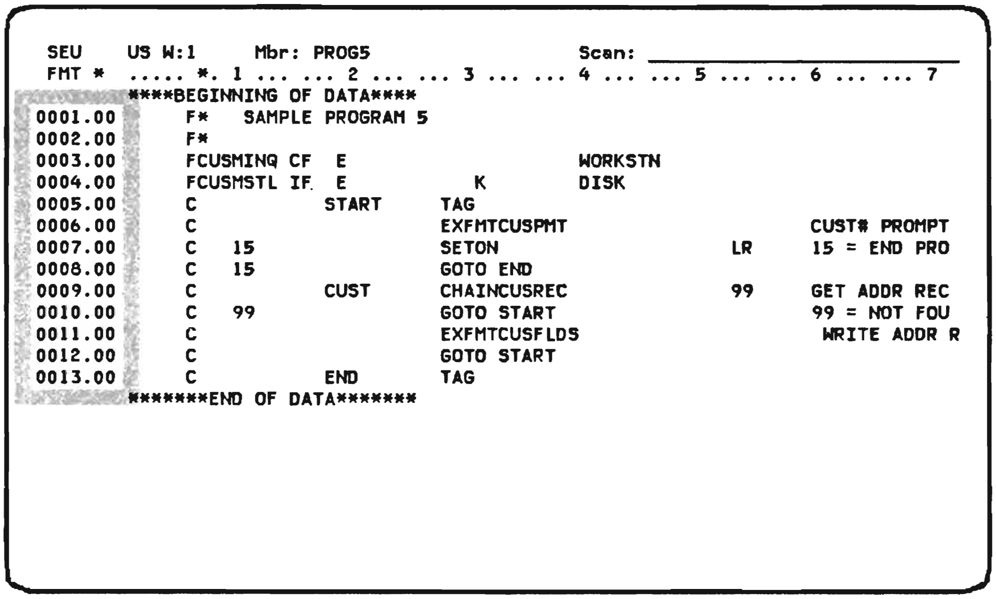
A The cursor is initially positioned here for entering a line command.

B Instructions for using line commands.

C Message indicates that new member was added.

You specify how you want to enter the source statements by entering one of the SEU line commands in the position indicated by the cursor. For example, you might enter the line command *IPH* to indicate that you want to insert a line (*I*) with prompting (*P*) for the RPG III control (*H*) specifications, and input fields would be shown at the bottom of the screen to prompt you for entries in the format of the control specifications. As indicated on the display, you can obtain additional details about the line commands by pressing the Help key.

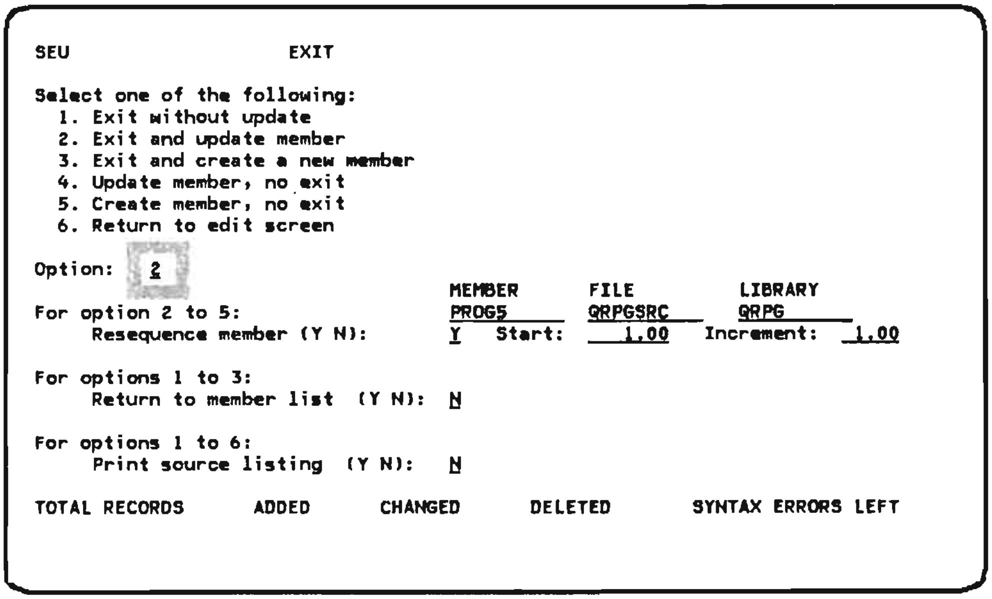
As you enter each source statement, it is given a sequence number:



```
 SEU    US W:1     Mbr: PROG5                  Scan: ___________________________
 FMT *  ..... *. 1 ... ... 2 ... ... 3 ... ... 4 ... ... 5 ... ... 6 ... ... 7
        ****BEGINNING OF DATA****
0001.00      F*   SAMPLE PROGRAM 5
0002.00      F*
0003.00      FCUSMINQ CF  E                    WORKSTN
0004.00      FCUSMSTL IF  E           K        DISK
0005.00      C           START     TAG
0006.00      C                     EXFMTCUSPMT                     CUST# PROMPT
0007.00      C   15                SETON                    LR     15 = END PRO
0008.00      C   15                GOTO END
0009.00      C           CUST      CHAINCUSREC              99     GET ADDR REC
0010.00      C   99                GOTO START                      99 = NOT FOU
0011.00      C                     EXFMTCUSFLDS                     WRITE ADDR R
0012.00      C                     GOTO START
0013.00      C           END       TAG
        *******END OF DATA*******
```

If you need to use an SEU line command again, you key it in over the sequence number.

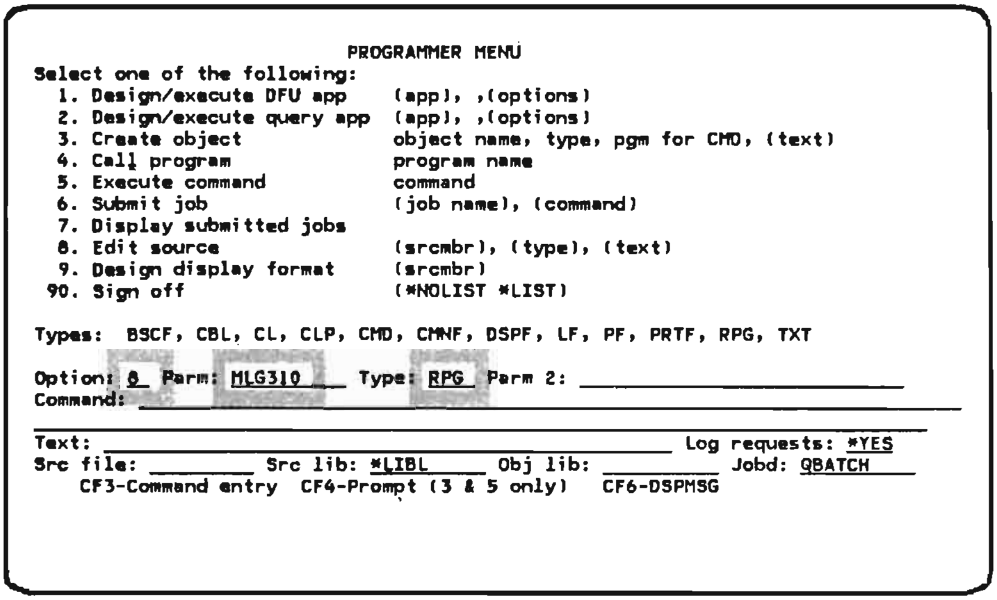
When you have completed entering the source statements, you press the CF1 key. An exit display appears next:



```
SEU                   EXIT

Select one of the following:
  1. Exit without update
  2. Exit and update member
  3. Exit and create a new member
  4. Update member, no exit
  5. Create member, no exit
  6. Return to edit screen

Option:  2
                                    MEMBER       FILE         LIBRARY
For option 2 to 5:                  PROG5        QRPGSRC      QRPG
     Resequence member (Y N):       Y   Start:     1.00   Increment:   1.00

For options 1 to 3:
     Return to member list  (Y N):  N

For options 1 to 6:
     Print source listing  (Y N):   N

TOTAL RECORDS      ADDED      CHANGED      DELETED      SYNTAX ERRORS LEFT
```

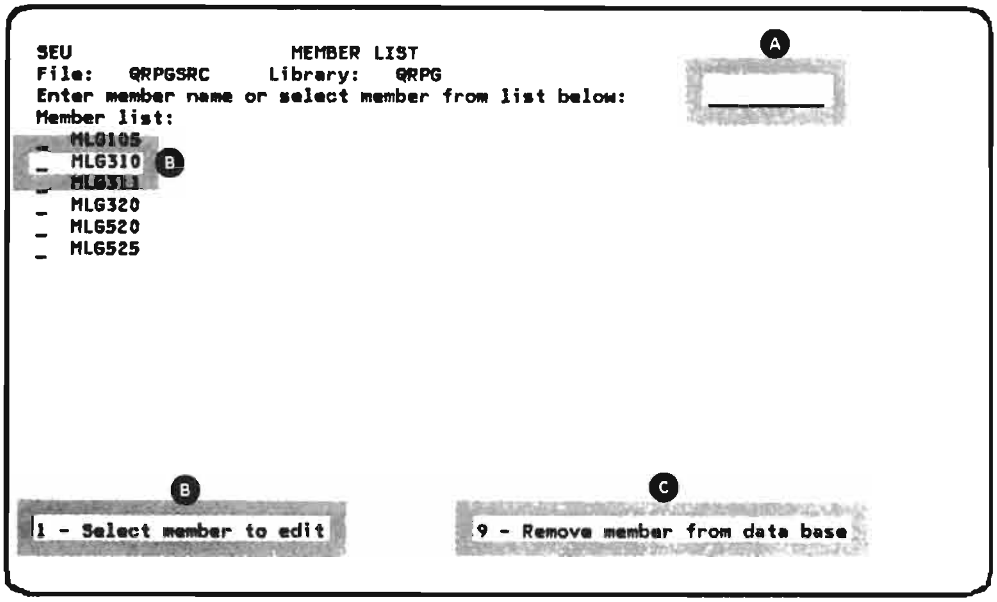
Up to this point, the source statements you have entered have been placed in an SEU work space. What you specify on the exit display determines whether the source is actually placed in the member. Because you want the source statements to be placed in the member PROG5 in this example, you select option 2 to update the member. If you select option 1, the source file will remain as it was when you entered SEU.

To *change* an existing source member, you follow a similar procedure. For example, to change the source for the RPG program MLG310, you select option 8 and specify the name and type on the programmer menu:



```
                              PROGRAMMER MENU
Select one of the following:
  1. Design/execute DFU app    (app), ,(options)
  2. Design/execute query app  (app), ,(options)
  3. Create object             object name, type, pgm for CMD, (text)
  4. Call program              program name
  5. Execute command           command
  6. Submit job                (job name), (command)
  7. Display submitted jobs
  8. Edit source               (srcmbr), (type), (text)
  9. Design display format     (srcmbr)
 90. Sign off                  (*NOLIST *LIST)

Types:  BSCF, CBL, CL, CLP, CMD, CMNF, DSPF, LF, PF, PRTF, RPG, TXT

Option: 8_ Parm: MLG310___  Type: RPG_ Parm 2: ____________________________
Command: ____________________________________________________________________
____________________________________________________________________________
Text: __________________________________________________ Log requests: *YES
Src file: _________ Src lib: *LIBL_____ Obj lib: __________ Jobd: QBATCH____
    CF3-Command entry  CF4-Prompt (3 & 5 only)   CF6-DSPMSG
```

If you selected option 8 and specified a source type *without* specifying a member name, you would first receive a list of all members in the source file:



```
SEU                    MEMBER LIST
File:   QRPGSRC      Library:   QRPG
Enter member name or select member from list below:                 ________
Member list:
_  MLG105
_  MLG310
_  MLG311
_  MLG320
_  MLG520
_  MLG525



1 - Select member to edit                  9 - Remove member from data base
```

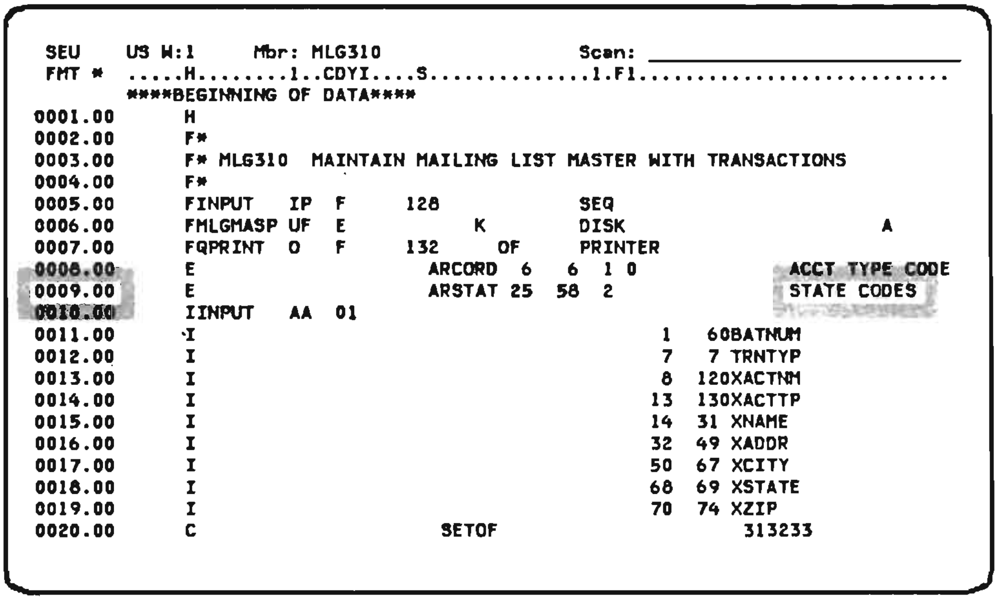
A Enter the member name here
or

B Enter a 1 beside the member name.

C You can remove a member by entering a 9 beside the member name.

You can select the member you want directly from this display. In this example, you want the member MLG310, so you enter a 1 in the input field beside MLG310.

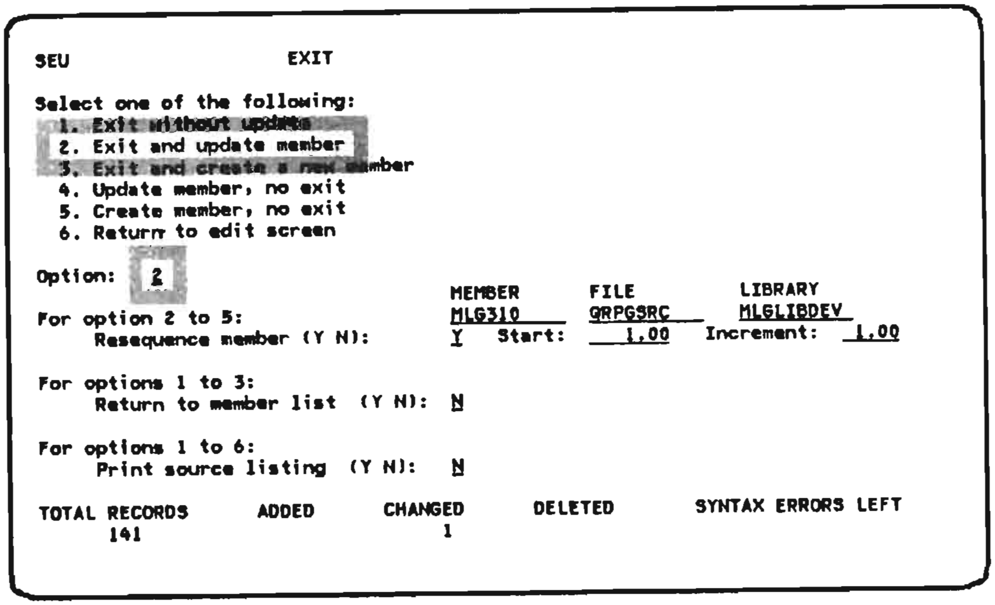
Whether you specify a member name on the programmer menu or select a member from the member list, you receive an SEU edit display that shows the source statements for the member:



```
 SEU    US W:1     Mbr: MLG310                 Scan: ______________________
 FMT *  .....H........1..CDYI....S..............1.F1...........................
        ****BEGINNING OF DATA****
0001.00      H
0002.00      F*
0003.00      F* MLG310  MAINTAIN MAILING LIST MASTER WITH TRANSACTIONS
0004.00      F*
0005.00      FINPUT   IP  F     128            SEQ
0006.00      FMLGMASP UF  E           K        DISK                      A
0007.00      FQPRINT  O   F     132     OF     PRINTER
0008.00      E                    ARCORD  6   6  1 0             ACCT TYPE CODE
0009.00      E                    ARSTAT 25  58  2               STATE CODES
0010.00      IINPUT   AA  01
0011.00      I                                        1   60BATNUM
0012.00      I                                        7    7 TRNTYP
0013.00      I                                        8  120XACTNM
0014.00      I                                       13  130XACTTP
0015.00      I                                       14  31 XNAME
0016.00      I                                       32  49 XADDR
0017.00      I                                       50  67 XCITY
0018.00      I                                       68  69 XSTATE
0019.00      I                                       70  74 XZIP
0020.00      C                     SETOF                     313233
```

To change a source statement, you key the correction in the appropriate position on the line containing the statement. For example, to change STATE CODES to STATE ABBREV on line 0009.00 of the display shown here, you enter ABBREV on top of CODES. By entering SEU line commands over particular sequence numbers, you can insert, delete, resequence, or request prompting for specific source statements.

When you have completed modifying the source, you press the CF1 key. You receive the exit display. The edited source statements do not replace the actual source in the source member unless you select an option that updates the member, as in this example:



```
SEU                      EXIT

Select one of the following:
  1. Exit without update
  2. Exit and update member
  3. Exit and create a new member
  4. Update member, no exit
  5. Create member, no exit
  6. Return to edit screen

Option:  2
                                        MEMBER       FILE         LIBRARY
For option 2 to 5:                      MLG310       QRPGSRC      MLGLIBDEV
     Resequence member (Y N):           Y   Start:      1.00   Increment:   1.00

For options 1 to 3:
     Return to member list  (Y N):  N

For options 1 to 6:
     Print source listing  (Y N):   N

TOTAL RECORDS        ADDED        CHANGED        DELETED        SYNTAX ERRORS LEFT
      141                               1
```

In the following chapters, various programs and files will be created from source that has been entered through SEU. For more details on SEU and its use, refer to the *SEU Reference Manual and User's Guide*.

Chapter 6. Libraries and Files

OVERVIEW

In Chapters 3 and 4, all of the objects used were placed in a library named QGPL (general purpose library), which is the IBM-supplied default library. Putting all user objects in a single library is probably not typical of library use on the System/38, because it does not separate multiple applications. One use of libraries is to group objects that have something in common so that, for example, the objects associated with each application are in separate libraries (which you define). Libraries can help you organize and control your system.

This chapter discusses libraries and files as follows:

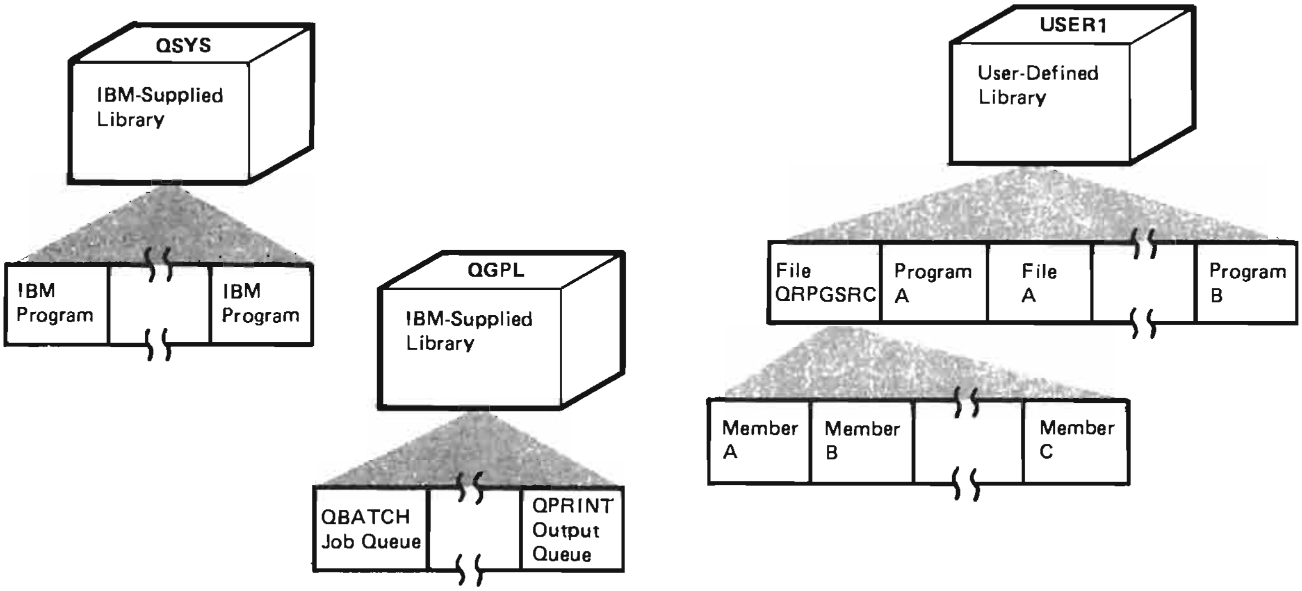
- The basic concepts of libraries and library lists.
- How to create a library and the basic objects needed for application development work.
- Considerations for using the programmer menu after the application is developed.
- The basic concept of a field reference file.
- How to create the data base files used in the remainder of this publication.

LIBRARIES

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

On System/38, a library can contain objects of different types, including programs, source files, and other kinds of files. 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.

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 CPF. 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. RPG, COBOL, the Interactive Data Base Utilities (IDU), and the Conversion Reformat Utility also have their own libraries (QRPG, QCBL, QIDU, and QS3E).

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.

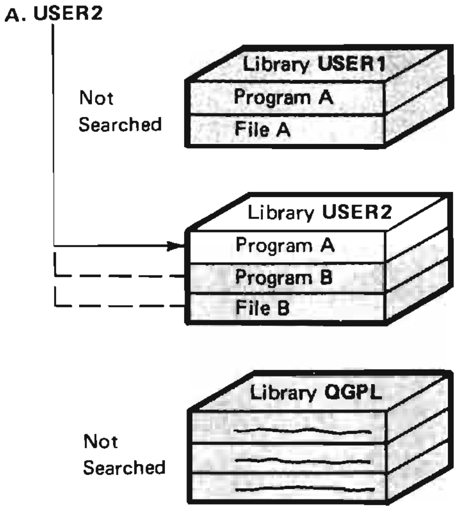
Qualified Names and the Library List

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 object name and the name of the library that contains the object are used, such as:

A.USER2

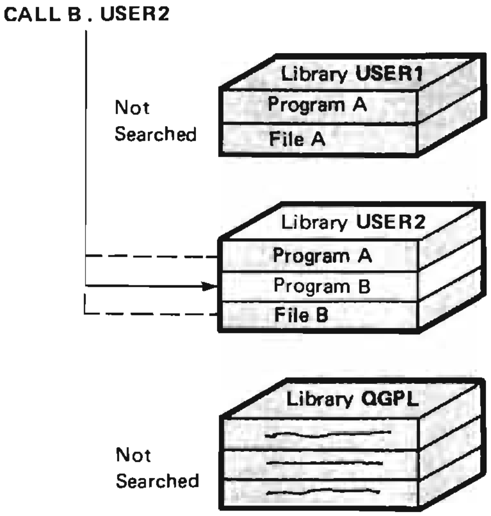
A period must separate the two names. If you use a qualified name when you request an object, only the library you specified is searched for the object. No other library is searched.



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 B.USER2

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.

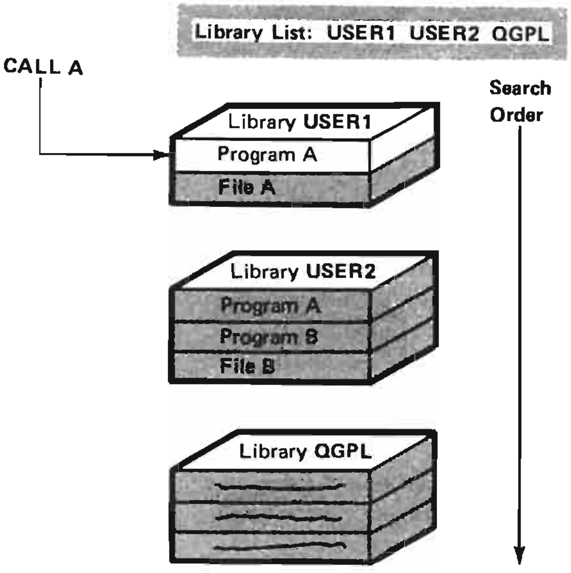


If a qualified name is not given when an object is requested, the system searches for the object by using the *library list* that is associated with the job in which the request was made. A library list is an ordered list of library names indicating which libraries are to be searched, and the order in 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
```

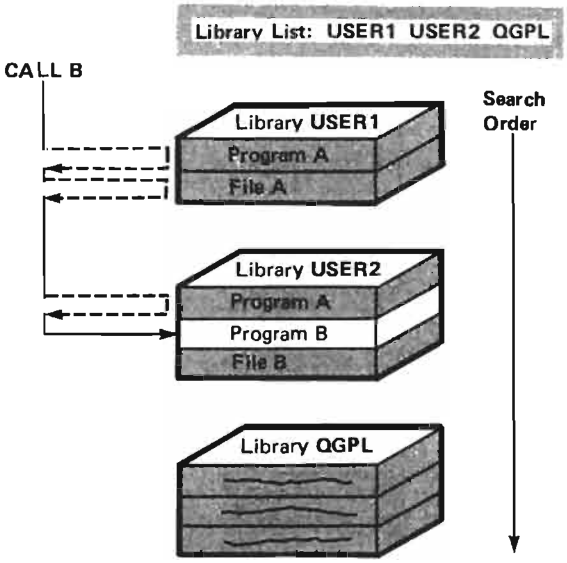
The first library, USER1, is searched for a program named A. Because program A is found in the first library, program A in USER2 is ignored.



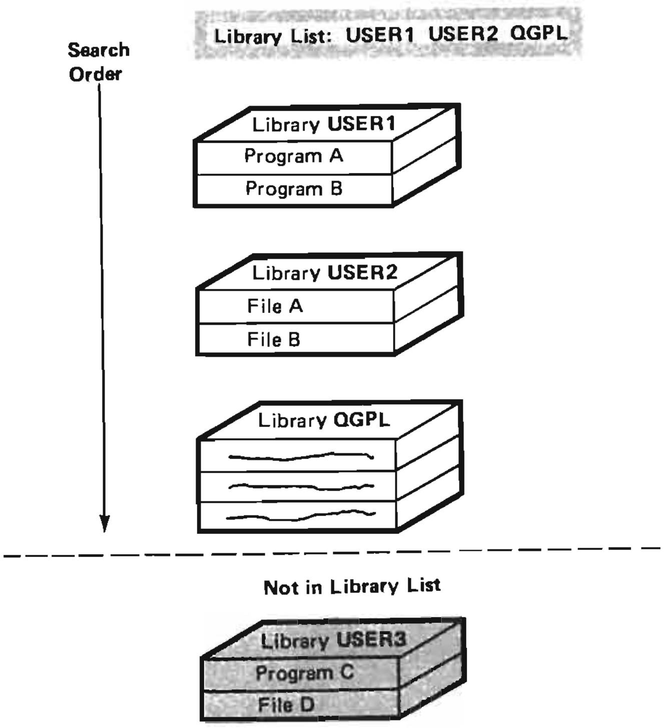
If you specify

```
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.



The libraries USER1, USER2, and QGPL are in the library list and the library USER3 is not. If you specify

```
CALL C.USER3
```

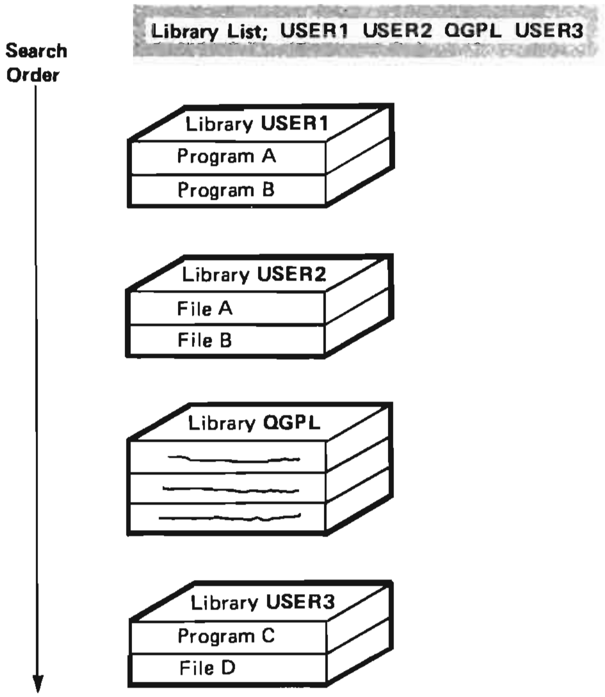
the program C is found in USER3 because C.USER3 is a qualified name.

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.

RPG III 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

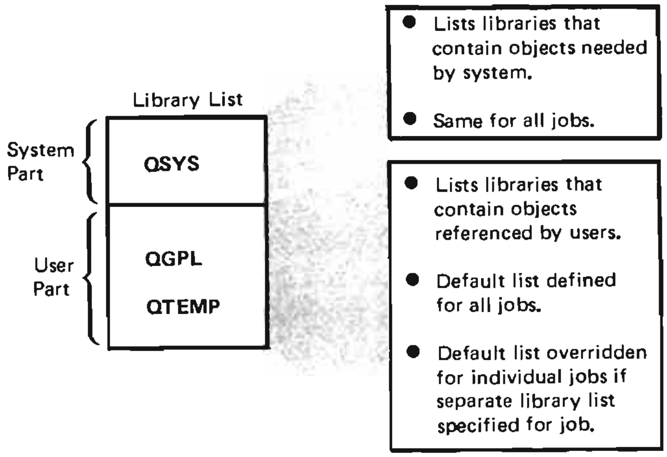
```
CALL C
```

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 6-1.



- Lists libraries that contain objects needed by system.
- Same for all jobs.

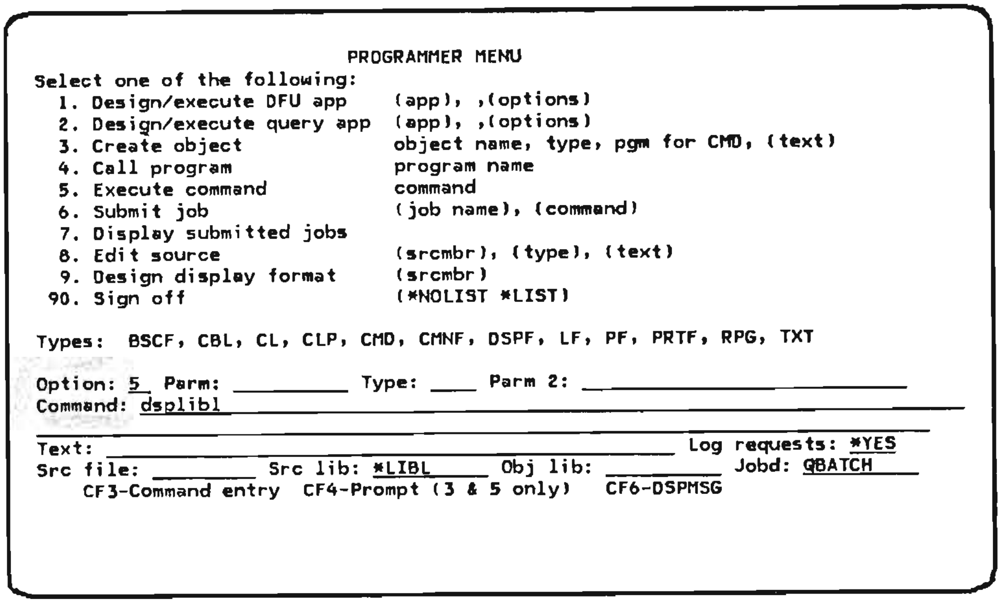
- Lists libraries that contain objects referenced by users.
- Default list defined for all jobs.
- Default list overridden for individual jobs if separate library list specified for job.

Figure 6-1. Job Library List

Figure 6-1 shows the library list as shipped by IBM. The system part contains only the QSYS library; the user part contains only the QGPL and QTEMP libraries (for more details, see the *CPF Programmer's Guide*).

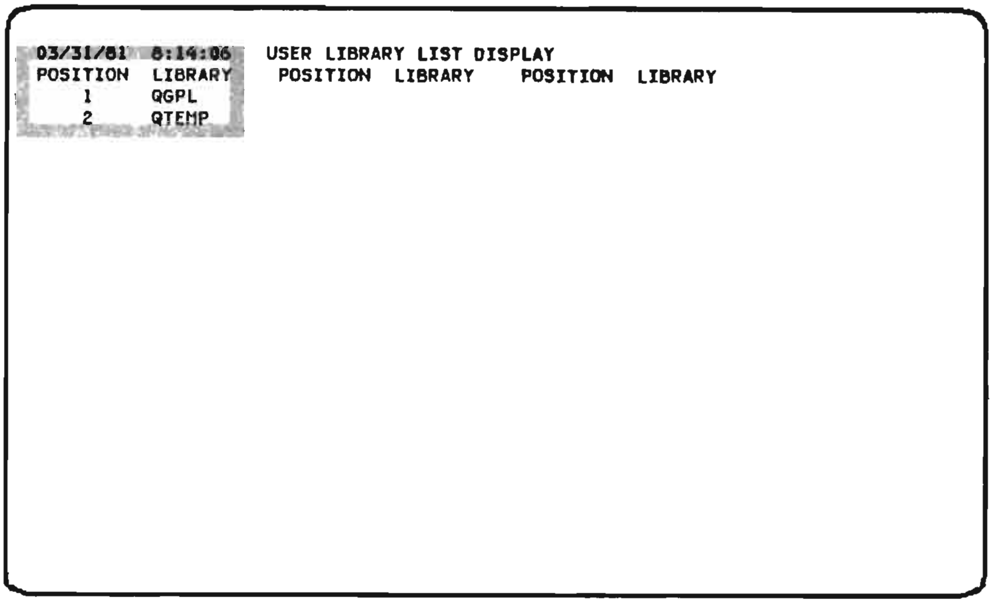
Generally, only the user part is apparent to the system users because it affects objects they request in individual jobs. Therefore, the discussion of library lists in this publication is restricted to the user part of the library list.

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. For example, you can use the programmer menu as follows to display the library list for your interactive job at a work station:



```
                           PROGRAMMER MENU
Select one of the following:
  1. Design/execute DFU app     (app), ,(options)
  2. Design/execute query app   (app), ,(options)
  3. Create object              object name, type, pgm for CMD, (text)
  4. Call program               program name
  5. Execute command             command
  6. Submit job                 (job name), (command)
  7. Display submitted jobs
  8. Edit source                (srcmbr), (type), (text)
  9. Design display format      (srcmbr)
 90. Sign off                   (*NOLIST *LIST)

Types:  BSCF, CBL, CL, CLP, CMD, CMNF, DSPF, LF, PF, PRTF, RPG, TXT

Option: 5_ Parm: __________ Type: ____ Parm 2: ____________________________
Command: dsplibl____________________________________________________________
____________________________________________________________________________
Text: __________________________________________________ Log requests: *YES
Src file: __________ Src lib: *LIBL_____ Obj lib: __________ Jobd: QBATCH____
    CF3-Command entry  CF4-Prompt (3 & 5 only)   CF6-DSPMSG
```

You receive a display of the following type:



```
03/31/81  8:14:06   USER LIBRARY LIST DISPLAY
POSITION  LIBRARY    POSITION  LIBRARY    POSITION  LIBRARY
    1     QGPL
    2     QTEMP
```

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. For example, if the application references objects in the QRPG and QIDU libraries, those libraries should be included in the library list for each job using the application. 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.

To avoid this problem, you need to replace the default library list by a library list that includes all libraries containing objects referenced in the application. The new library list must be defined for all jobs using the application.

Each application should have a method to *define its own library list*. For example, an interactive job that requests the mailing list application could first call a program that uses the Replace Library List (RPLLIBL) command to define the correct library list. For a batch job, the library list could be specified in one of the following ways:

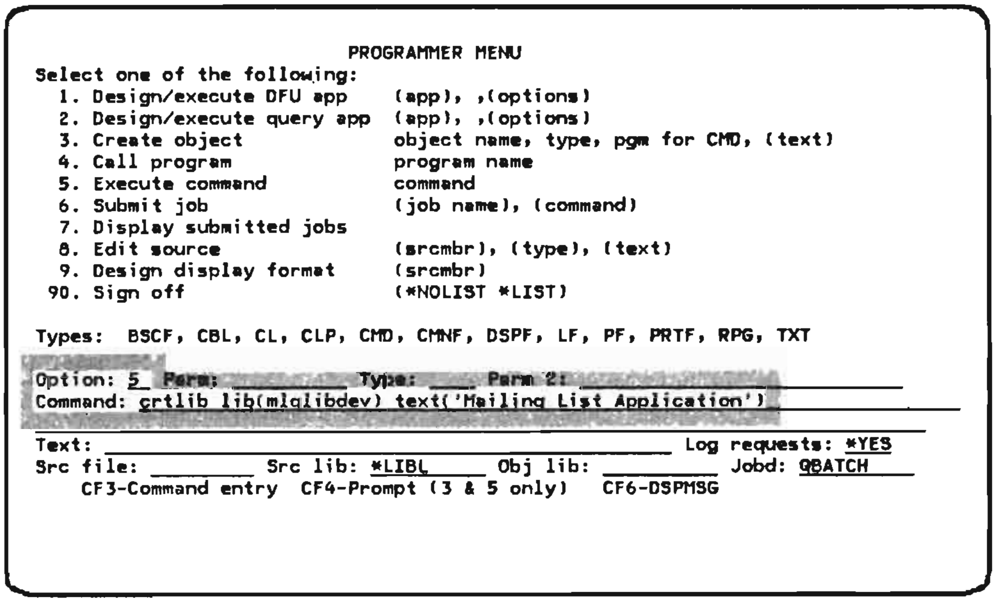
- The job description for the batch job contains an initial library list (INLLIBL parameter of Create Job Description command).
- The Job or Submit Job command used to submit the batch job specifies an initial library list (INLLIBL parameter).
- The batch job contains the RPLLIBL command, which specifies a new library list.

Because it is desirable to specify a library list only once, a job description that contains a specific library list should be created for use by all batch jobs in an application.

The new library list specified by the INLLIBL parameter or the RPLLIBL command overrides (completely replaces) the default user library list.

THE MAILING LIST LIBRARY AND STANDARD OBJECTS

In the remaining application approaches, a user-created library named MLGLIBDEV (mailing library development) is used. It will be used to contain all of the objects created for the application. It is created from the programmer menu as follows:



```
                               PROGRAMMER MENU
Select one of the following:
  1. Design/execute DFU app    (app), ,(options)
  2. Design/execute query app  (app), ,(options)
  3. Create object             object name, type, pgm for CMD, (text)
  4. Call program              program name
  5. Execute command           command
  6. Submit job                (job name), (command)
  7. Display submitted jobs
  8. Edit source               (srcmbr), (type), (text)
  9. Design display format     (srcmbr)
 90. Sign off                  (*NOLIST *LIST)

Types:  BSCF, CBL, CL, CLP, CMD, CMNF, DSPF, LF, PF, PRTF, RPG, TXT

Option: 5_ Parm: __________ Type: ____ Parm 2: ____________________
Command: crtlib lib(mlglibdev) text('Mailing List Application')_____
____________________________________________________________________
Text: ________________________________________ Log requests: *YES
Src file: ________ Src lib: *LIBL_____ Obj lib: _________ Jobd: QBATCH____
    CF3-Command entry  CF4-Prompt (3 & 5 only)   CF6-DSPMSG
```

This library will contain standard objects that can be used in any application and objects associated only with the mailing list application. The standard objects that can be used in any application (and are required for the remaining approaches) are the following:

- Source files
- Job description
- Set library list program

This section describes how these standard objects are created and used.

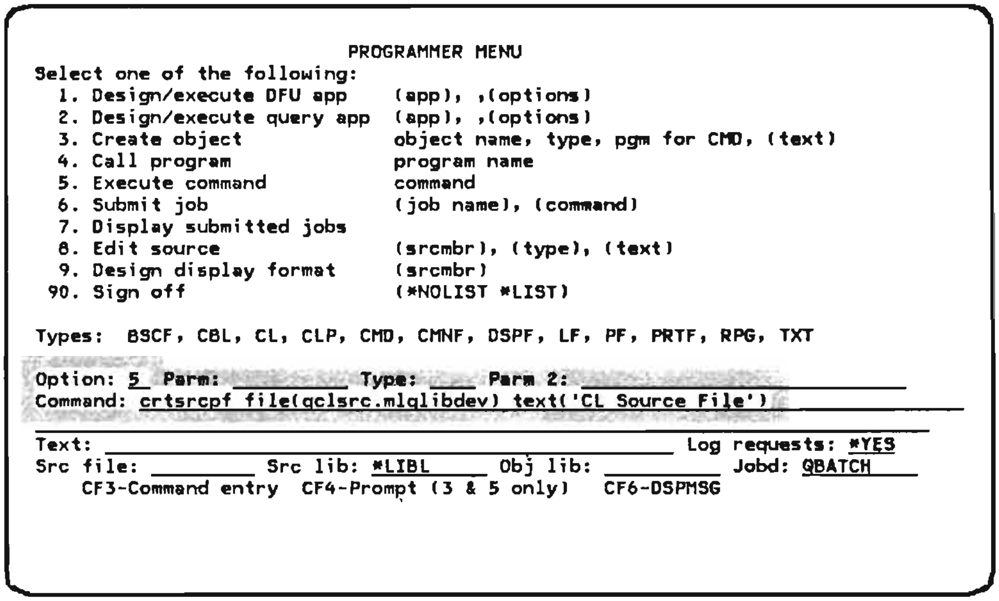
If this library will only be used for testing purposes, also specify TYPE(*TEST) in the CRTLIB command. Defining a library as a test library facilitates the use of debugging functions (not described in this publication; see the *CPF Programmer's Guide*).

Source Files

Four of the IBM-supplied source files, QCLSRC, QDDSSRC, QRPGSRC, QUDSSRC, are used in this example. They are designed to contain source statements for items such as high-level language programs and 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. The names of the IBM-supplied source files will be used as the names of the user-created source files because the IBM-supplied source file names are defaults on various commands. Because the user-created source files will be in MLGLIBDEV, they can have the same names as the IBM-supplied source files, which are in IBM-supplied libraries.

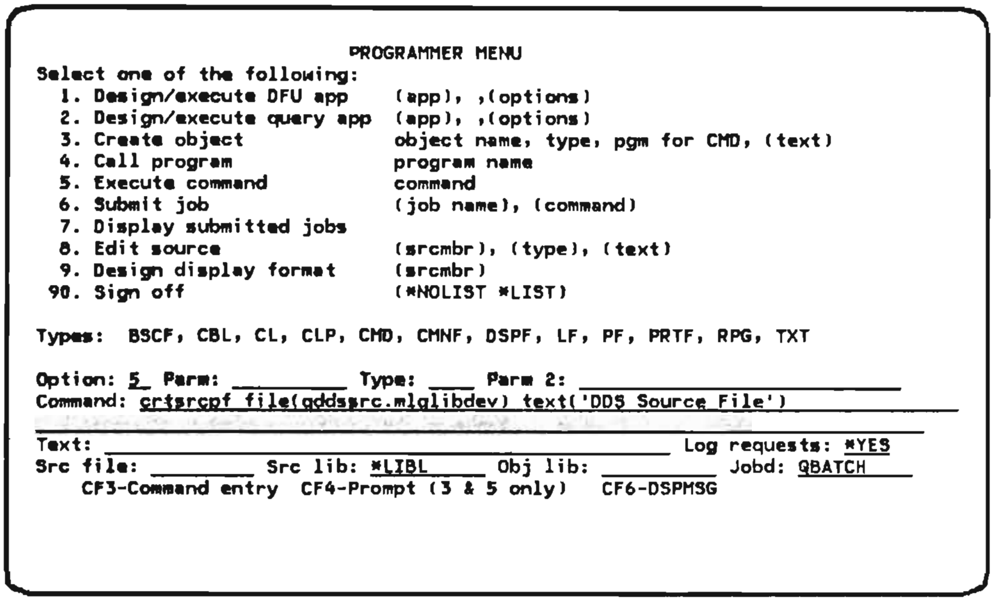
A QCLSRC file for CL program source is created as follows:



```
                              PROGRAMMER MENU
Select one of the following:
  1. Design/execute DFU app    (app), ,(options)
  2. Design/execute query app  (app), ,(options)
  3. Create object             object name, type, pgm for CMD, (text)
  4. Call program              program name
  5. Execute command           command
  6. Submit job                (job name), (command)
  7. Display submitted jobs
  8. Edit source               (srcmbr), (type), (text)
  9. Design display format     (srcmbr)
 90. Sign off                  (*NOLIST *LIST)

Types:  BSCF, CBL, CL, CLP, CMD, CMNF, DSPF, LF, PF, PRTF, RPG, TXT

Option: 5_ Parm: __________ Type: ____ Parm 2: ______________________
Command: crtsrcpf file(qclsrc.mlglibdev) text('CL Source File')_______
_____________________________________________________________________
Text: __________________________________________ Log requests: *YES
Src file: ________ Src lib: *LIBL_____ Obj lib: __________ Jobd: QBATCH____
    CF3-Command entry  CF4-Prompt (3 & 5 only)   CF6-DSPMSG
```

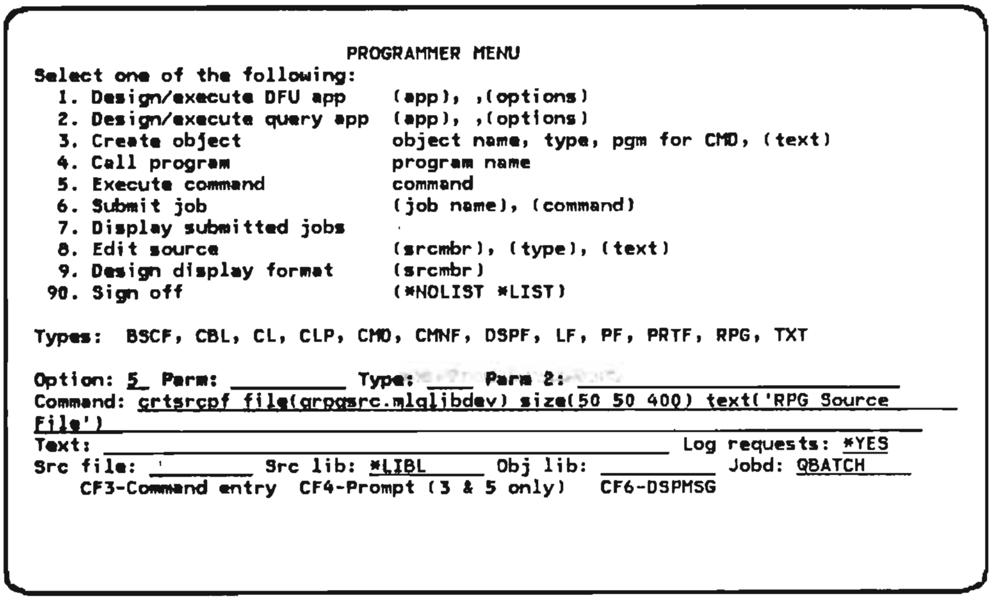
A qualified name (qclsrc.mlglibdev) is used for the file when it is created so that the file is placed in the correct library.

A QDDSSRC source file for physical file, logical file, and display file source is created as follows:



```
                             PROGRAMMER MENU
Select one of the following:
  1. Design/execute DFU app     (app), ,(options)
  2. Design/execute query app   (app), ,(options)
  3. Create object              object name, type, pgm for CMD, (text)
  4. Call program               program name
  5. Execute command            command
  6. Submit job                 (job name), (command)
  7. Display submitted jobs
  8. Edit source                (srcmbr), (type), (text)
  9. Design display format      (srcmbr)
 90. Sign off                   (*NOLIST *LIST)

Types:  BSCF, CBL, CL, CLP, CMD, CMNF, DSPF, LF, PF, PRTF, RPG, TXT

Option: 5_ Parm: __________ Type: ____ Parm 2: ____________________________
Command: crtsrcpf file(qddssrc.mlqlibdev) text('DDS Source File')__________
___________________________________________________________________________
Text: _______________________________________________ Log requests: *YES
Src file: __________ Src lib: *LIBL______ Obj lib: __________ Jobd: QBATCH____
    CF3-Command entry  CF4-Prompt (3 & 5 only)   CF6-DSPMSG
```

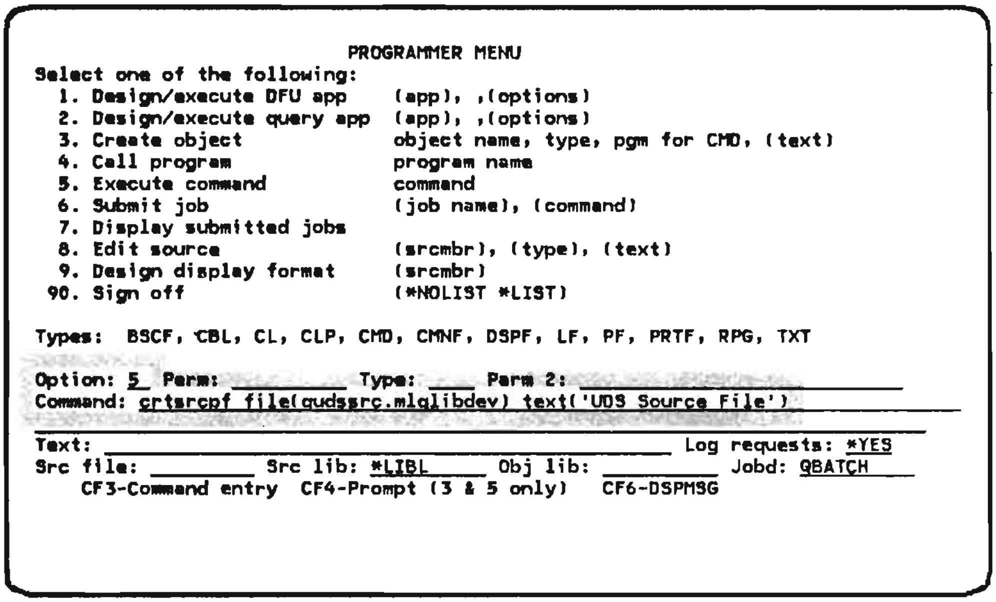
A QRPGSRC source file for RPG III program source is created as follows:



```
                             PROGRAMMER MENU
Select one of the following:
  1. Design/execute DFU app     (app), ,(options)
  2. Design/execute query app   (app), ,(options)
  3. Create object              object name, type, pgm for CMD, (text)
  4. Call program               program name
  5. Execute command            command
  6. Submit job                 (job name), (command)
  7. Display submitted jobs
  8. Edit source                (srcmbr), (type), (text)
  9. Design display format      (srcmbr)
 90. Sign off                   (*NOLIST *LIST)

Types:  BSCF, CBL, CL, CLP, CMD, CMNF, DSPF, LF, PF, PRTF, RPG, TXT

Option: 5_ Parm: __________ Type: ____ Parm 2: ____________________________
Command: crtsrcpf file(qrpgsrc.mlqlibdev) size(50 50 400) text('RPG Source_____
File')_____________________________________________________________________
Text: _______________________________________________ Log requests: *YES
Src file: __________ Src lib: *LIBL______ Obj lib: __________ Jobd: QBATCH____
    CF3-Command entry  CF4-Prompt (3 & 5 only)   CF6-DSPMSG
```

The values that are listed for the SIZE parameter will provide better storage allocation than the default values because RPG members normally contain more source statements than other types of members.

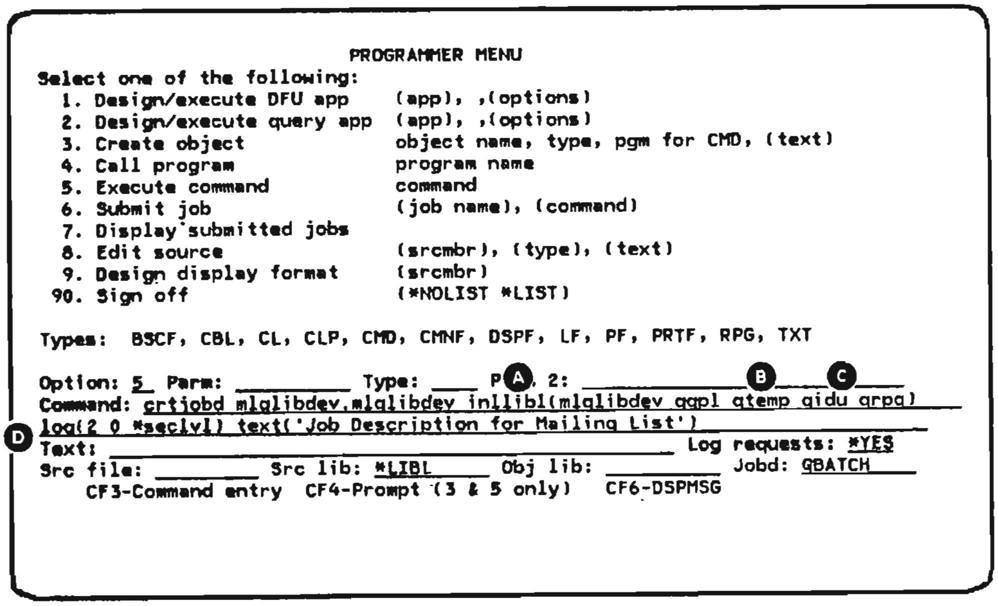
A QUDSSRC source file for the utility definition statements for IDU applications is created as follows:



```
                              PROGRAMMER MENU
Select one of the following:
  1. Design/execute DFU app    (app), ,(options)
  2. Design/execute query app  (app), ,(options)
  3. Create object             object name, type, pgm for CMD, (text)
  4. Call program              program name
  5. Execute command           command
  6. Submit job                (job name), (command)
  7. Display submitted jobs
  8. Edit source               (srcmbr), (type), (text)
  9. Design display format     (srcmbr)
 90. Sign off                  (*NOLIST *LIST)

Types:  BSCF, CBL, CL, CLP, CMD, CMNF, DSPF, LF, PF, PRTF, RPG, TXT

Option: 5_ Perm: __________ Type: ___ Perm 2: _____________________________
Command: crtsrcpf file(qudssrc.mlqlibdev) text('UDS Source File')___________
____________________________________________________________________________
Text: ___________________________________________________ Log requests: *YES
Src file: _________ Src lib: *LIBL_____ Obj lib: __________ Jobd: QBATCH____
    CF3-Command entry  CF4-Prompt (3 & 5 only)   CF6-DSPMSG
```

Job Description

Each job has a set of attributes that define how the system is to handle the job. Specifying all the attributes each time a job is submitted would be tedious and time-consuming. Because many jobs in an application have the same (or similar) attributes, CPF supports an object called a *job description* in which the attributes of a job can be predefined. When an application job is submitted, it need only reference the job description to ensure that the correct attributes are used. A job description named MLGLIBDEV will be created to control the execution of batch jobs in this application. Included in this job description is the library list to be used for the batch jobs. The job description is created as follows:



```
                          PROGRAMMER MENU
Select one of the following:
  1. Design/execute DFU app    (app), ,(options)
  2. Design/execute query app  (app), ,(options)
  3. Create object             object name, type, pgm for CMD, (text)
  4. Call program              program name
  5. Execute command           command
  6. Submit job                (job name), (command)
  7. Display submitted jobs
  8. Edit source               (srcmbr), (type), (text)
  9. Design display format     (srcmbr)
 90. Sign off                  (*NOLIST *LIST)

Types:  BSCF, CBL, CL, CLP, CMD, CMNF, DSPF, LF, PF, PRTF, RPG, TXT

Option: 5_ Parm: _________ Type: ____ P  , 2: ______________________
Command: crtjobd mlglibdev.mlglibdev inllibl(mlglibdev qgpl qtemp qidu qrpg)
log(2 0 *seclvl) text('Job Description for Mailing List')
Text: ________________________________________ Log requests: *YES
Src file: _________ Src lib: *LIBL_____ Obj lib: _________ Jobd: QBATCH____
    CF3-Command entry  CF4-Prompt (3 & 5 only)   CF6-DSPMSG
```

A The INLLIBL parameter specifies the initial library list so that MLGLIBDEV is the first library searched, followed by QGPL (general purpose library), QTEMP (temporary library), QIDU (IDU library), and QRPG (RPG library).

B A temporary library (QTEMP) is created at the beginning of each job and then deleted when the job is completed.

C The QIDU and QRPG libraries contain the IBM-supplied programs for the IDU and RPG III program products.

D The LOG parameter defines the amount and type of information to be logged in the job log.

Note that the user library, MLGLIBDEV, which contains the source files just defined, is specified before the IBM-supplied libraries, which contain the default source files. The libraries must be specified in this order for correct use of commands and the programmer menu.

For each batch job that uses this job description, the library list specified in the INLLIBL parameter replaces the default user library list (see Figure 6-1). The application library, MLGLIBDEV, would not normally be included in the default library list.

Because this application does not use COBOL or the Conversion Reformat Utility, the COBOL library (QCBL) and Conversion Reformat library (QS3E) are not included in the library list. You may need to include QCBL and QS3E in the library list for your applications.

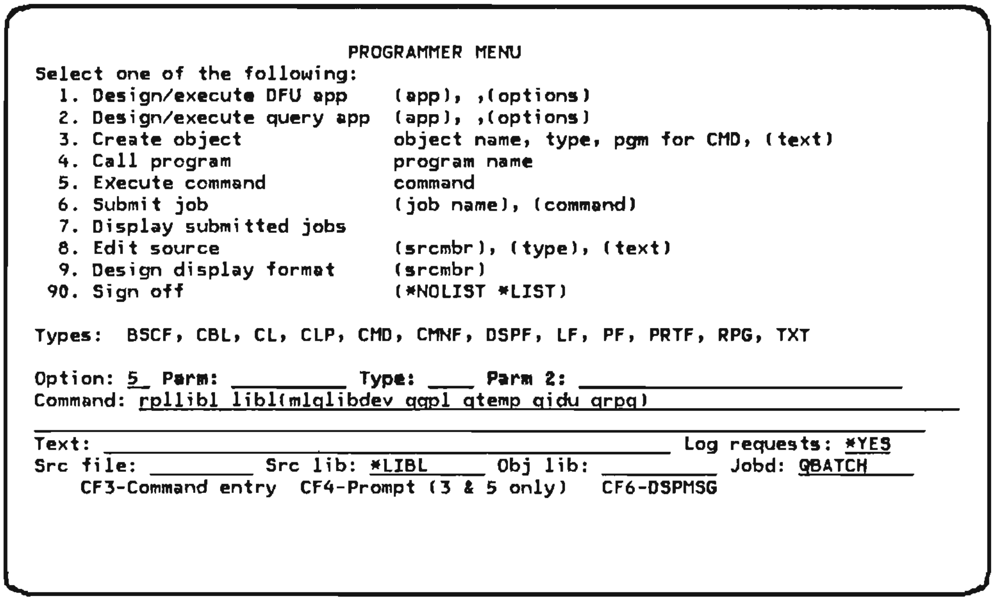
When the job description is created, default values are used for the job queue (QBATCH) and the output queue (QPRINT). These defaults are used for all examples in this publication.

Set Library List Program

The job description just created is used to replace the default library list for batch jobs executing in the application. Now, a CL program named SETLIBL will be created to replace the default library list in interactive jobs executing in the application. After you create this program, the Replace Library List (RPLLIBL) command does not need to be entered for each interactive job; however, the SETLIBL program must be executed each time you sign on to do an interactive job in this application.

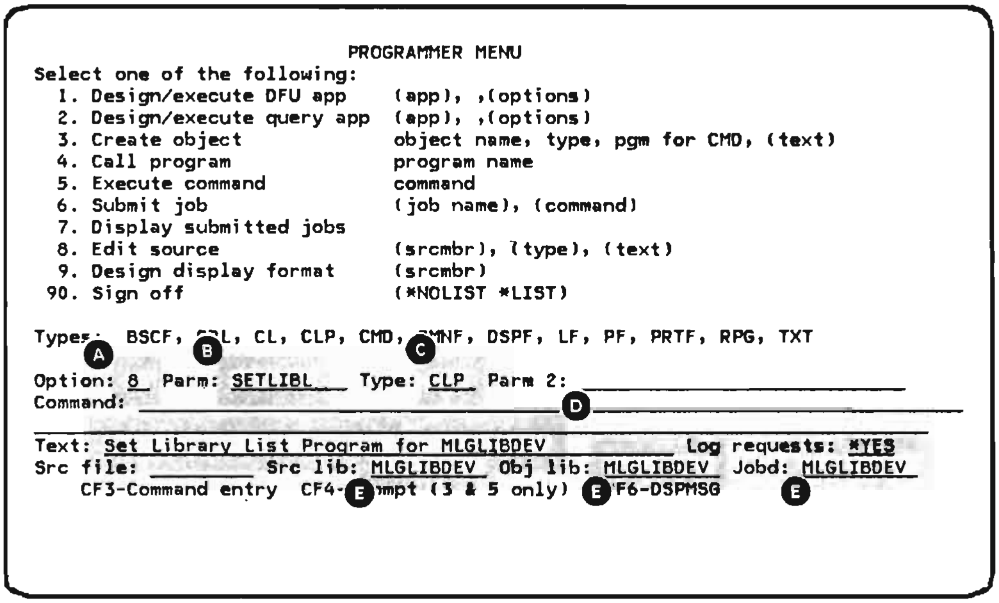
The following procedure can be used to create the SETLIBL program.

Enter the RPLLIBL command as follows so that MLGLIBDEV is part of the library list and the system can find the job description entered in the next step.



```
                           PROGRAMMER MENU
Select one of the following:
  1. Design/execute DFU app    (app), ,(options)
  2. Design/execute query app  (app), ,(options)
  3. Create object             object name, type, pgm for CMD, (text)
  4. Call program              program name
  5. Execute command           command
  6. Submit job                (job name), (command)
  7. Display submitted jobs
  8. Edit source               (srcmbr), (type), (text)
  9. Design display format     (srcmbr)
 90. Sign off                  (*NOLIST *LIST)

Types:  BSCF, CBL, CL, CLP, CMD, CMNF, DSPF, LF, PF, PRTF, RPG, TXT

Option: 5_ Parm: __________ Type: ____ Parm 2: ____________________________
Command: rpllibl libl(mlglibdev qgpl qtemp qidu qrpg)______________________
___________________________________________________________________________
Text: __________________________________________________ Log requests: *YES
Src file: _________ Src lib: *LIBL_____ Obj lib: __________ Jobd: QBATCH____
    CF3-Command entry  CF4-Prompt (3 & 5 only)   CF6-DSPMSG
```

Now request SEU as follows:



```
                              PROGRAMMER MENU
Select one of the following:
  1. Design/execute DFU app    (app), ,(options)
  2. Design/execute query app  (app), ,(options)
  3. Create object             object name, type, pgm for CMD, (text)
  4. Call program              program name
  5. Execute command           command
  6. Submit job                (job name), (command)
  7. Display submitted jobs
  8. Edit source               (srcmbr), (type), (text)
  9. Design display format     (srcmbr)
 90. Sign off                  (*NOLIST *LIST)

Types:  BSCF, CBL, CL, CLP, CMD, CMNF, DSPF, LF, PF, PRTF, RPG, TXT

Option: 8_ Parm: SETLIBL___ Type: CLP_ Parm 2: ______________________
Command: ________________________________________________________________
_________________________________________________________________________
Text: Set Library List Program for MLGLIBDEV__________ Log requests: *YES
Src file: _________ Src lib: MLGLIBDEV_ Obj lib: MLGLIBDEV_ Jobd: MLGLIBDEV_
    CF3-Command entry  CF4-Prompt (3 & 5 only)  CF6-DSPMSG
```

A Select option 8 to request SEU.

B Key in the name of the source member here.

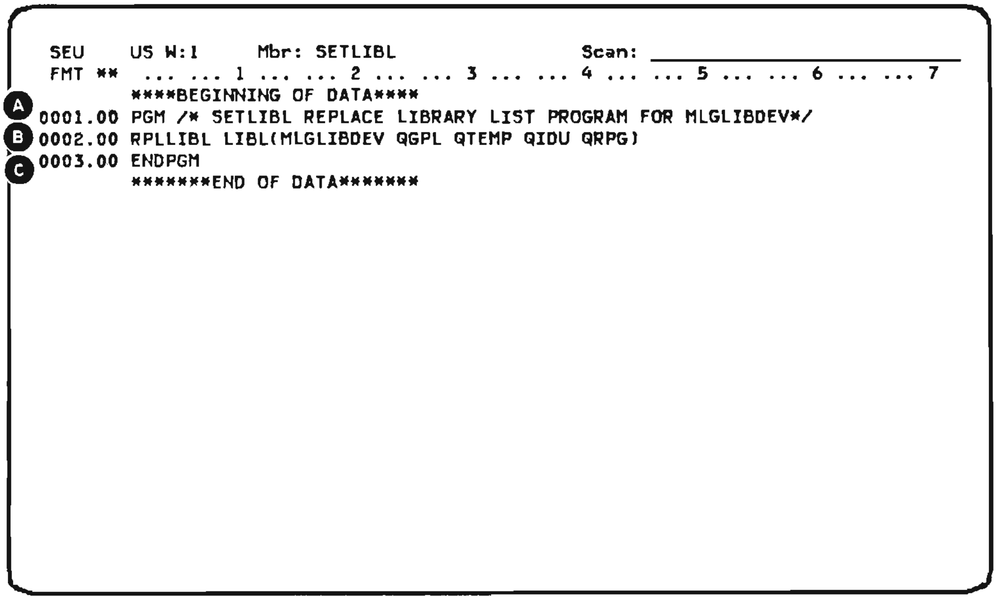
C Key in the type here.

D You can key in text here to describe the source member.

E Once you have made these entries for *Src lib*, *Obj lib*, and *Jobd*, they will remain on the menu, so that you do not have to enter them again for the remainder of your interactive job at the work station.

The source library (*Src lib*) must be specified when option 1, 2, 3, 8, or 9 is selected on the programmer menu. The object library (*Obj lib*) must be specified when option 3 or 4 is selected. However, if you create an object by selecting option 5, you must specify the library MLGLIBDEV as you did when you created the MLGLIBDEV job description. A job description (*Jobd*) must be specified when option 3 or 6 is selected. MLGLIBDEV is keyed in as the source library, the object library, and the job description (which you previously created). A source file (*Src file*) name is not selected because the default file names (such as QCLSRC) will be used.

SETLIBL is now a member of QCLSRC and space is allocated for the source statements. An SEU edit display appears that allows you to enter the source statements as follows:



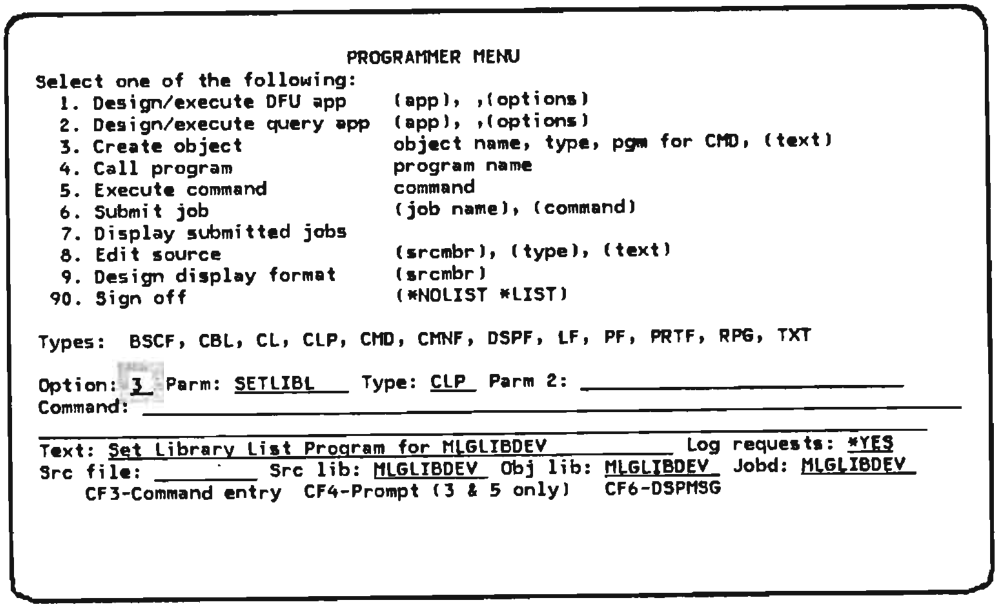
```
SEU    US W:1     Mbr: SETLIBL                 Scan: ______________________
FMT **  ... ... 1 ... ... 2 ... ... 3 ... ... 4 ... ... 5 ... ... 6 ... ... 7
        ****BEGINNING OF DATA****
0001.00 PGM /* SETLIBL REPLACE LIBRARY LIST PROGRAM FOR MLGLIBDEV*/
0002.00 RPLLIBL LIBL(MLGLIBDEV QGPL QTEMP QIDU QRPG)
0003.00 ENDPGM
        *******END OF DATA*******
```

A The PGM command begins a CL program.

B Any job that calls this program will have its library list replaced as defined by this RPLLIBL command.

C The ENDPGM command identifies the end of the program.

After you enter the source statements as shown and exit SEU, the programmer menu reappears. The program SETLIBL can now be created as an object within MLGLIBDEV by using the programmer menu as follows:



```
                              PROGRAMMER MENU
Select one of the following:
  1. Design/execute DFU app    (app), ,(options)
  2. Design/execute query app  (app), ,(options)
  3. Create object             object name, type, pgm for CMD, (text)
  4. Call program              program name
  5. Execute command           command
  6. Submit job                (job name), (command)
  7. Display submitted jobs
  8. Edit source               (srcmbr), (type), (text)
  9. Design display format     (srcmbr)
 90. Sign off                  (*NOLIST *LIST)

Types:  BSCF, CBL, CL, CLP, CMD, CMNF, DSPF, LF, PF, PRTF, RPG, TXT

Option: 3_ Parm: SETLIBL___ Type: CLP_ Parm 2: ____________________________
Command: _____________________________________________________________________
______________________________________________________________________________
Text: Set Library List Program for MLGLIBDEV__________ Log requests: *YES
Src file: _________ Src lib: MLGLIBDEV_ Obj lib: MLGLIBDEV_ Jobd: MLGLIBDEV_
    CF3-Command entry  CF4-Prompt (3 & 5 only)   CF6-DSPMSG
```

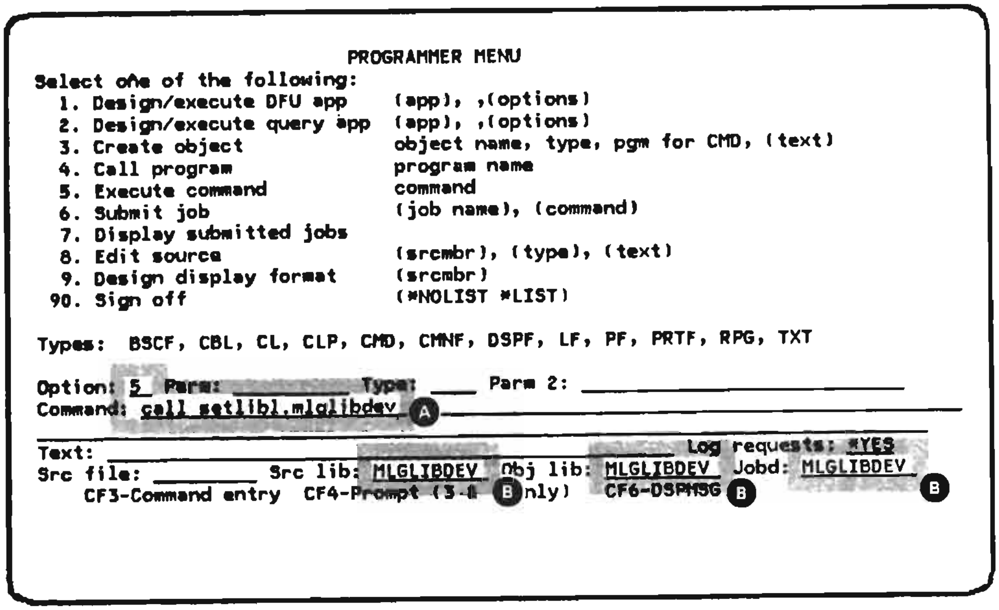
You only need to enter a 3 (for option) because all of your other previous entries still appear on the programmer menu. The entry in the option field disappears after each requested function is complete, but all other entries remain.

All create operations that result from selecting option 3 are submitted as batch jobs. Input at the work station is not inhibited while an object is being created.

Note the difference between selecting option 3, which was used to create the SETLIBL program, and selecting option 5, which was used to create the job description. Option 5 requires the entire command, including the name of the library that contains SETLIBL, to be keyed in. The command is executed immediately when the Enter key is pressed. Option 3 requires only a few parameters to be keyed in. When the Enter key is pressed, the command is submitted for execution as a batch job. The system uses the source and object library names keyed in at the bottom of the programmer menu to generate the correct create command and place it as a batch job on the job queue. SETLIBL is created by the batch job and is placed in MLGLIBDEV. The library MLGLIBDEV is now ready to contain specific application objects.

Programmer Menu Considerations

Each time you are ready to work on the mailing list application, you can define your own library list from the programmer menu as follows:



```
                              PROGRAMMER MENU
Select one of the following:
  1. Design/execute DFU app    (app), ,(options)
  2. Design/execute query app  (app), ,(options)
  3. Create object             object name, type, pgm for CMD, (text)
  4. Call program              program name
  5. Execute command           command
  6. Submit job                (job name), (command)
  7. Display submitted jobs
  8. Edit source               (srcmbr), (type), (text)
  9. Design display format     (srcmbr)
 90. Sign off                  (*NOLIST *LIST)

Types:  BSCF, CBL, CL, CLP, CMD, CMNF, DSPF, LF, PF, PRTF, RPG, TXT

Option: 5_ Parm: __________ Type: ___ Parm 2: ________________________
Command: call setlibl,mlglibdev_______________________________________
______________________________________________________________________
Text: ______________________________________________ Log requests: *YES
Src file: ________ Src lib: MLGLIBDEV_ Obj lib: MLGLIBDEV_ Jobd: MLGLIBDEV_
    CF3-Command entry  CF4-Prompt (3-4 nly)  CF6-DSPMSG
```

A These entries call the SETLIBL program, which replaces your library list.

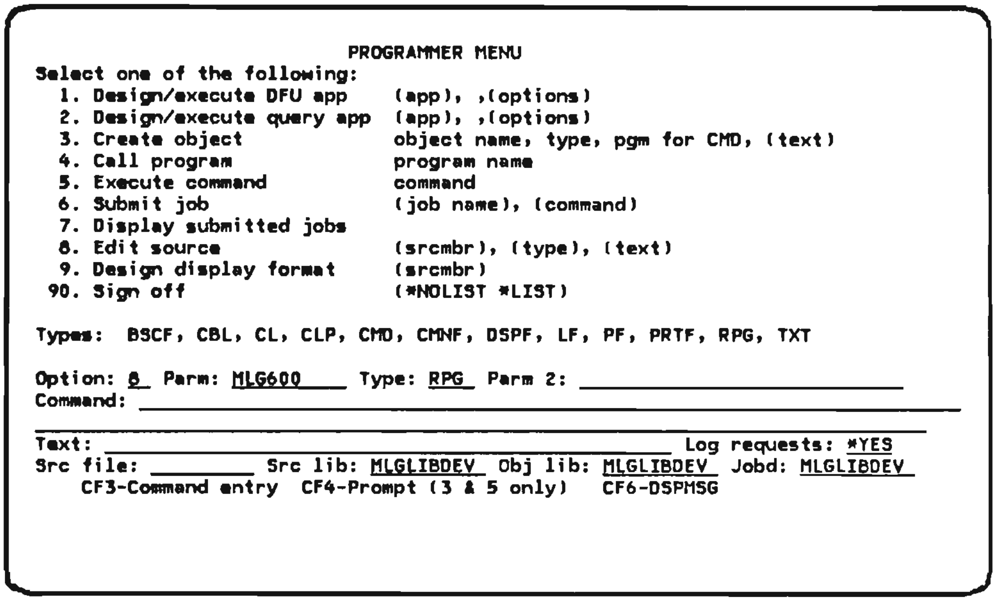
B These entries are made once per work station job.

The MLGLIBDEV entries that are made once establish the environment for application development and maintenance. It is assumed that you have established the environment for the remaining application approaches in this publication.

This section describes a typical programmer task and shows how the programmer menu can be used to perform this task. Assume that source for an RPG III program named MLG600 has been previously entered. The source must now be changed for one of the following reasons:

- The program did not compile because the RPG compiler found errors in it.
- The program did not execute as expected.
- Record formats have changed.
- The program is being enhanced.

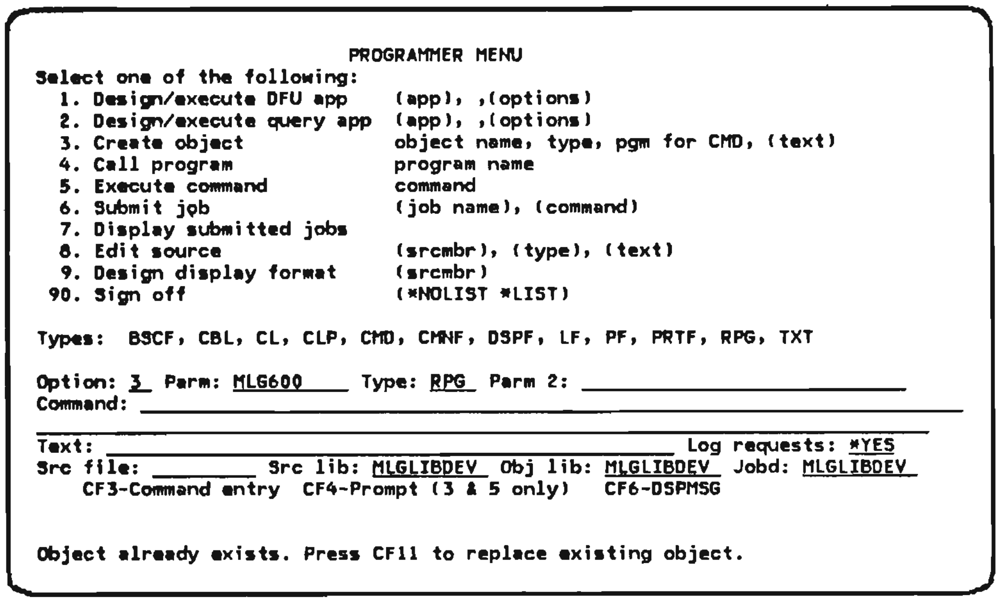
To change the source, you request SEU by making the following entries on the programmer menu:



```
                              PROGRAMMER MENU
Select one of the following:
  1. Design/execute DFU app     (app), ,(options)
  2. Design/execute query app  (app), ,(options)
  3. Create object              object name, type, pgm for CMD, (text)
  4. Call program               program name
  5. Execute command            command
  6. Submit job                 (job name), (command)
  7. Display submitted jobs
  8. Edit source                (srcmbr), (type), (text)
  9. Design display format      (srcmbr)
 90. Sign off                   (*NOLIST *LIST)

Types:  BSCF, CBL, CL, CLP, CMD, CMNF, DSPF, LF, PF, PRTF, RPG, TXT

Option: 8_ Parm: MLG600____ Type: RPG_ Parm 2: ___________________________
Command: ___________________________________________________________________
____________________________________________________________________________
Text: ___________________________________________________ Log requests: *YES
Src file: _________ Src lib: MLGLIBDEV_ Obj lib: MLGLIBDEV_ Jobd: MLGLIBDEV_
    CF3-Command entry  CF4-Prompt (3 & 5 only)   CF6-DSPMSG
```

You receive an SEU display that shows the source statements (see *Source Entry Utility* in Chapter 5). After you make the required changes and exit SEU, the programmer menu reappears. It looks like the previous display (with the entries on it) except the option field is blank.

If you try to use option 3 to create MLG600 and a program named MLG600 already exists in MLGLIBDEV, the following error message appears:



```
                              PROGRAMMER MENU
Select one of the following:
  1. Design/execute DFU app     (app), ,(options)
  2. Design/execute query app  (app), ,(options)
  3. Create object              object name, type, pgm for CMD, (text)
  4. Call program               program name
  5. Execute command            command
  6. Submit job                 (job name), (command)
  7. Display submitted jobs
  8. Edit source                (srcmbr), (type), (text)
  9. Design display format      (srcmbr)
 90. Sign off                   (*NOLIST *LIST)

Types:  BSCF, CBL, CL, CLP, CMD, CMNF, DSPF, LF, PF, PRTF, RPG, TXT

Option: 3_ Parm: MLG600____ Type: RPG_ Parm 2: ___________________________
Command: ___________________________________________________________________
____________________________________________________________________________
Text: ___________________________________________________ Log requests: *YES
Src file: _________ Src lib: MLGLIBDEV_ Obj lib: MLGLIBDEV_ Jobd: MLGLIBDEV_
    CF3-Command entry  CF4-Prompt (3 & 5 only)   CF6-DSPMSG


Object already exists. Press CF11 to replace existing object.
```

The existing program named MLG600 must be deleted before the new program named MLG600 can be created. This can be done by replacing the existing MLG600 with the new MLG600 as follows:

1. Press the Error Reset key to remove the error message and allow input.

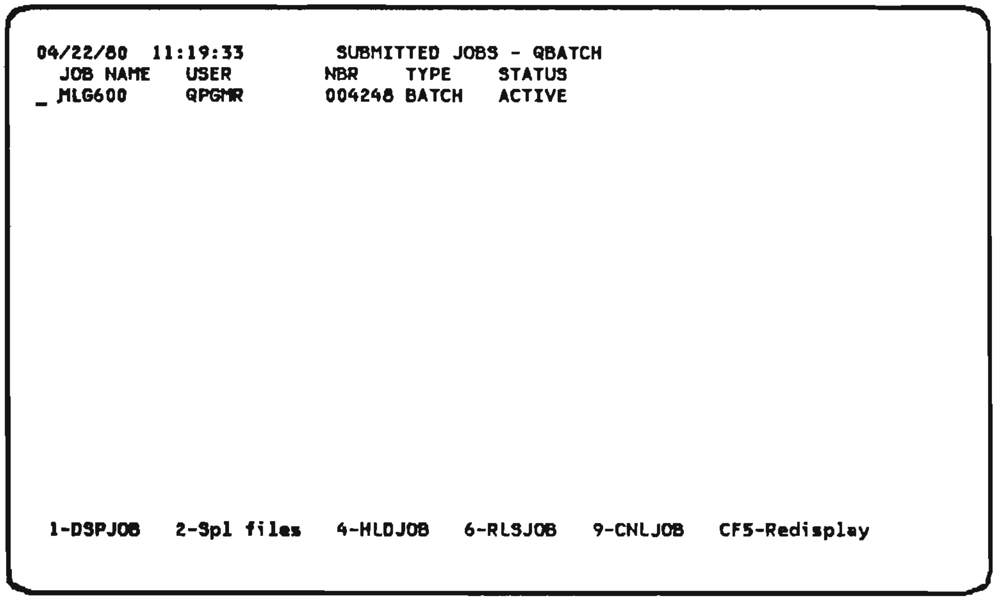
2. Press CF11 to delete the existing object and then create the new object.

If you did not intend to give the two objects the same name and do not want to replace the existing object, press the Error Reset key, change the request, and press the Enter key to enter the new request.

Another way to replace an object is to select option 3 and then press CF11 (instead of pressing Enter). This means that the object you specified should be replaced if it exists. No error message is displayed.

When some objects are created for the final production version, you may want to specify certain parameters on the Create commands instead of using defaults. You can use command prompting to help you do this from the programmer menu. When option 3 is selected, press CF4 to display the prompt. Specific options can then be selected and the modified Create command is submitted as a batch job.

Each time a request to create an object is submitted by selecting option 3, that request becomes a job. Submitted jobs can be reviewed by selecting option 7 on the programmer menu. You receive a display of the following type:



```
04/22/80  11:19:33        SUBMITTED JOBS - QBATCH
  JOB NAME   USER         NBR    TYPE    STATUS
_ MLG600     QPGMR        004248 BATCH   ACTIVE











 1-DSPJOB  2-Spl files  4-HLDJOB  6-RLSJOB  9-CNLJOB  CF5-Redisplay
```

While the submitted job is on the job queue or is being compiled, the programmer can be working in the same or a different application area. When the compilation is finished, a message saying whether or not the compilation was successful is sent to the work station. Press CF6 from the programmer menu to display messages.

The compilation listing produced by the compiler can be displayed at the work station and/or printed. System/38 allows all programmer functions to be performed without using paper output. A full discussion of this technique is beyond the scope of this publication.

FILES

The following three types of files are needed for the interactive approaches in Chapters 7 through 11:

- Field reference file
- Master files
- Transaction files

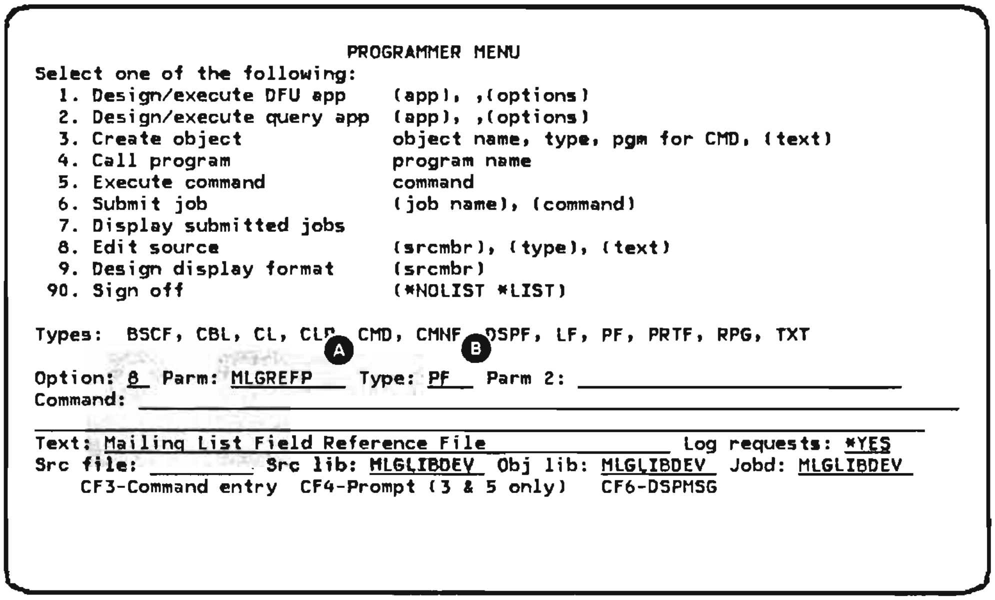
This section discusses these objects and shows how to create them.

Field Reference File

A *field reference file* is a physical file whose record format describes the fields used by a group of files. A field reference file contains no members (data records). The field descriptions in the field reference file can be referred to when the DDS for other files are written. Some benefits of using a field reference file are:

- It is a single source for all field information (such as attributes and text description).
- It allows using a reference to a previously described file.
- It improves documentation throughout the files and programs.
- It simplifies the coding needed to achieve consistency and documentation.

The field reference file named MLGREFP contains the record format for the mailing list example. To enter the source for MLGREFP, request SEU from the programmer menu as follows:

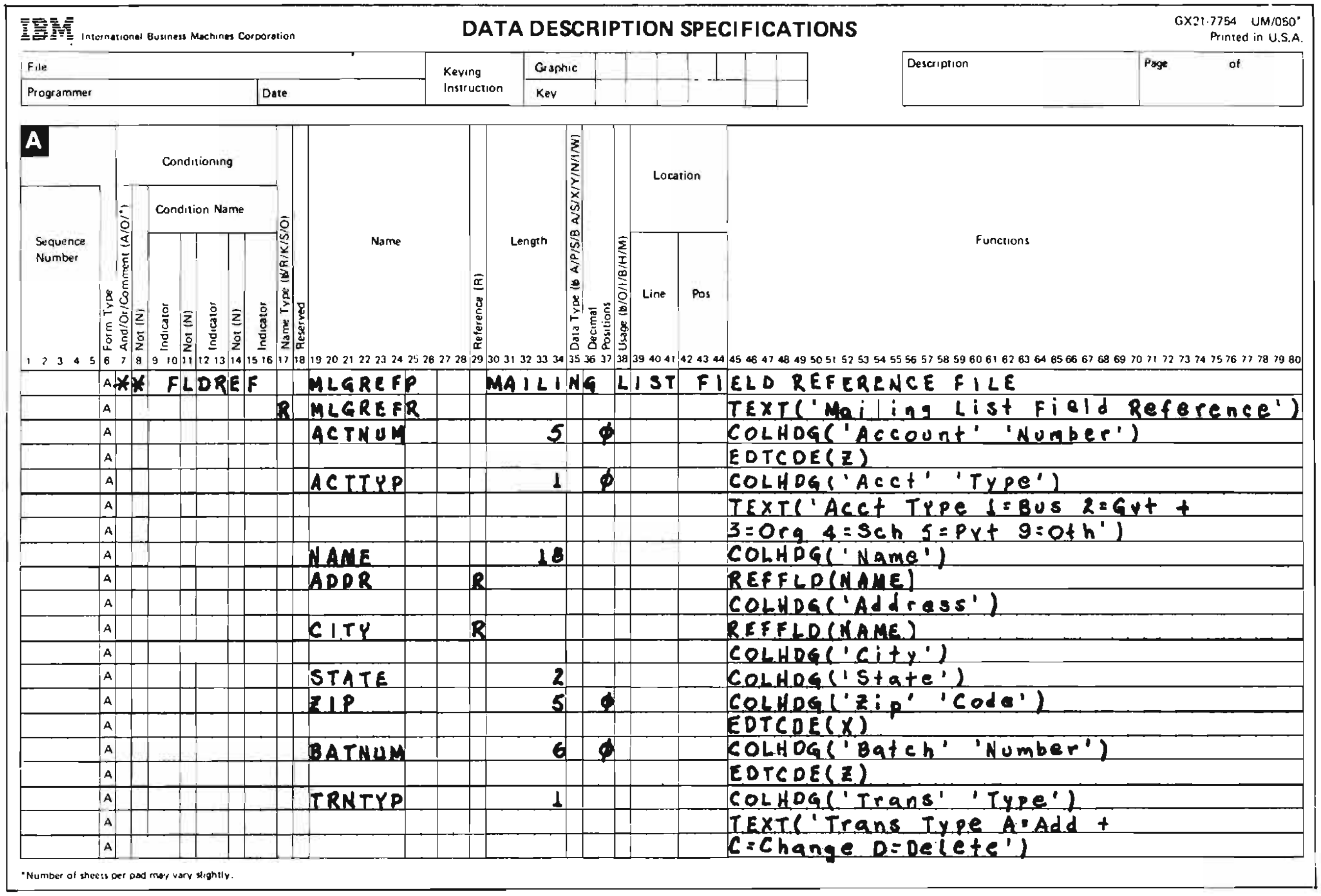


```
                              PROGRAMMER MENU
Select one of the following:
  1. Design/execute DFU app    (app), ,(options)
  2. Design/execute query app  (app), ,(options)
  3. Create object             object name, type, pgm for CMD, (text)
  4. Call program              program name
  5. Execute command           command
  6. Submit job                (job name), (command)
  7. Display submitted jobs
  8. Edit source               (srcmbr), (type), (text)
  9. Design display format     (srcmbr)
 90. Sign off                  (*NOLIST *LIST)

Types:  BSCF, CBL, CL, CLP, CMD, CMNF, DSPF, LF, PF, PRTF, RPG, TXT

Option: 8  Parm: MLGREFP    Type: PF   Parm 2: ____________________________
Command: ____________________________________________________________________
_______________________________________________________________________________
Text: Mailing List Field Reference File                  Log requests: *YES
Src file: _________ Src lib: MLGLIBDEV  Obj lib: MLGLIBDEV  Jobd: MLGLIBDEV
    CF3-Command entry  CF4-Prompt (3 & 5 only)   CF6-DSPMSG
```

A The name keyed in for srcmbr (MLGREFP) is normally the same as the name for the file that will be created (MLGREFP). Using the same name helps control source and object relationships.

B Because you indicated that the type is a file (a *physical file* in this case), the source will be placed in QDDSSRC.

You receive an SEU display that allows you to enter the DDS source for MLGREFP. The DDS to be entered are shown in Figure 6-2.



IBM International Business Machines Corporation

DATA DESCRIPTION SPECIFICATIONS

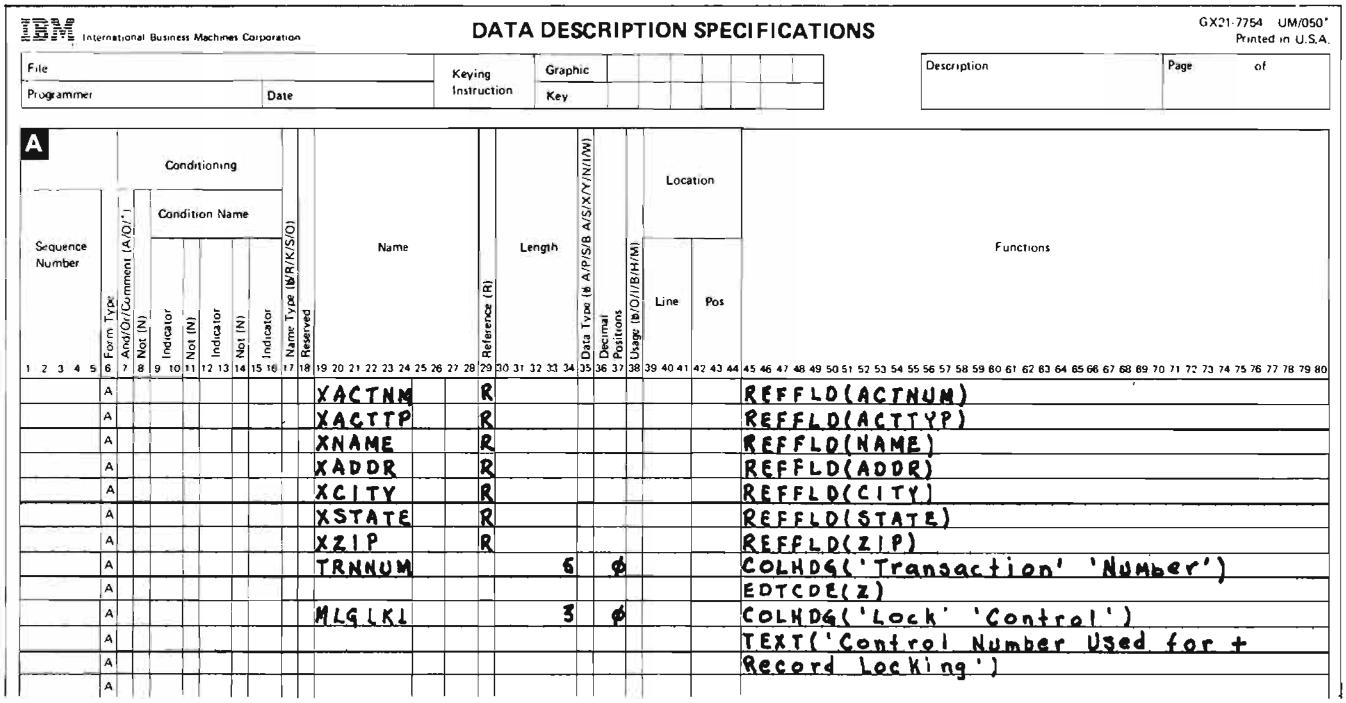
GX21-7754 UM/050*
Printed in U.S.A.

File | Keying Instruction | Graphic | Key | Description | Page of
Programmer | Date

A — Sequence Number (1 2 3 4 5) · Form Type (6) · And/Or/Comment (A/O/*) (7) · Conditioning: Condition Name — Not (N) (8), Indicator (9 10), Not (N) (11), Indicator (12 13), Not (N) (14), Indicator (15 16) · Name Type (b/R/K/S/O) (17) · Reserved (18) · Name (19–28) · Reference (R) (29) · Length (30–34) · Data Type (b A/P/S/B A/S/X/Y/N/I/W) (35) · Decimal Positions (36 37) · Usage (b/O/I/B/H/M) (38) · Location: Line (39 40 41), Pos (42 43 44) · Functions (45–80)

```
     A** FLDREF    MLGREFP     MAILING LIST FIELD REFERENCE FILE
     A          R MLGREFR                   TEXT('Mailing List Field Reference')
     A            ACTNUM         5  Ø       COLHDG('Account' 'Number')
     A                                      EDTCODE(Z)
     A            ACTTYP         1  Ø       COLHDG('Acct' 'Type')
     A                                      TEXT('Acct Type 1=Bus 2=Gvt +
     A                                      3=Org 4=Sch 5=Pvt 9=Oth')
     A            NAME          18          COLHDG('Name')
     A            ADDR      R               REFFLD(NAME)
     A                                      COLHDG('Address')
     A            CITY      R               REFFLD(NAME)
     A                                      COLHDG('City')
     A            STATE          2          COLHDG('State')
     A            ZIP            5  Ø       COLHDG('Zip' 'Code')
     A                                      EDTCODE(X)
     A            BATNUM         6  Ø       COLHDG('Batch' 'Number')
     A                                      EDTCODE(Z)
     A            TRNTYP         1          COLHDG('Trans' 'Type')
     A                                      TEXT('Trans Type A=Add +
     A                                      C=Change D=Delete')
```

*Number of sheets per pad may vary slightly.



IBM International Business Machines Corporation

DATA DESCRIPTION SPECIFICATIONS

GX21-7754 UM/050*
Printed in U.S.A.

File | Keying Instruction | Graphic | Key | Description | Page of
Programmer | Date

A — Sequence Number (1 2 3 4 5) · Form Type (6) · And/Or/Comment (A/O/*) (7) · Conditioning: Condition Name — Not (N) (8), Indicator (9 10), Not (N) (11), Indicator (12 13), Not (N) (14), Indicator (15 16) · Name Type (b/R/K/S/O) (17) · Reserved (18) · Name (19–28) · Reference (R) (29) · Length (30–34) · Data Type (b A/P/S/B A/S/X/Y/N/I/W) (35) · Decimal Positions (36 37) · Usage (b/O/I/B/H/M) (38) · Location: Line (39 40 41), Pos (42 43 44) · Functions (45–80)

```
     A            XACTNM    R               REFFLD(ACTNUM)
     A            XACTTP    R               REFFLD(ACTTYP)
     A            XNAME     R               REFFLD(NAME)
     A            XADDR     R               REFFLD(ADDR)
     A            XCITY     R               REFFLD(CITY)
     A            XSTATE    R               REFFLD(STATE)
     A            XZIP      R               REFFLD(ZIP)
     A            TRNNUM         6  Ø       COLHDG('Transaction' 'Number')
     A                                      EDTCODE(Z)
     A            MLGLK1         3  Ø       COLHDG('Lock' 'Control')
     A                                      TEXT('Control Number Used for +
     A                                      Record Locking')
     A
```

Figure 6-2. DDS for MLGREFP Field Reference File

Each field uses the COLHDG keyword to define a column heading that is used by IDU. If the TEXT keyword is not defined for a field, the column heading is used as the text description (which is also used by IDU). This description also appears on the RPG III compilation listing when the described field is copied as part of an externally described file.

The descriptive information (COLHDG and TEXT) is entered in both uppercase and lowercase letters. SEU usually changes lowercase entries to uppercase. However, if you press the CF12 key, SEU will not change lowercase to uppercase; your entries will remain either lowercase or uppercase exactly as you key them in (you press the Shift key for uppercase). When uppercase and lowercase information is printed on a system printer, an optional print belt is available for printing both uppercase and lowercase. If a print belt containing only uppercase letters is mounted, System/38 translates the lowercase letters so that they are printed as uppercase letters. This is the default for IBM-supplied files.

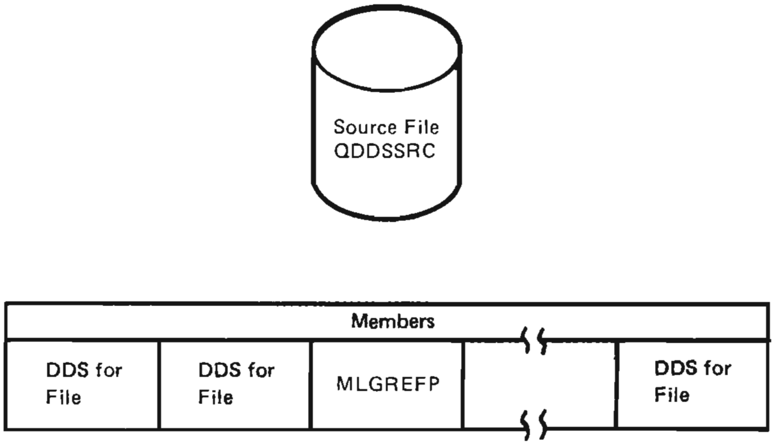
The ADDR and CITY fields are defined by a reference function: R in position 29 and REFFLD beginning in position 45. The reference function allows a field to be defined the same as a previously defined field. In this example, the ADDR and CITY fields are defined the same as the NAME field. If the size of the NAME field changes, the size of the ADDR and CITY fields automatically changes within the field reference file. The transaction fields (those fields whose names begin with X), also shown in Figure 6-2, are defined by the reference function. The transaction fields each have unique names because RPG requires unique storage areas to have unique names.

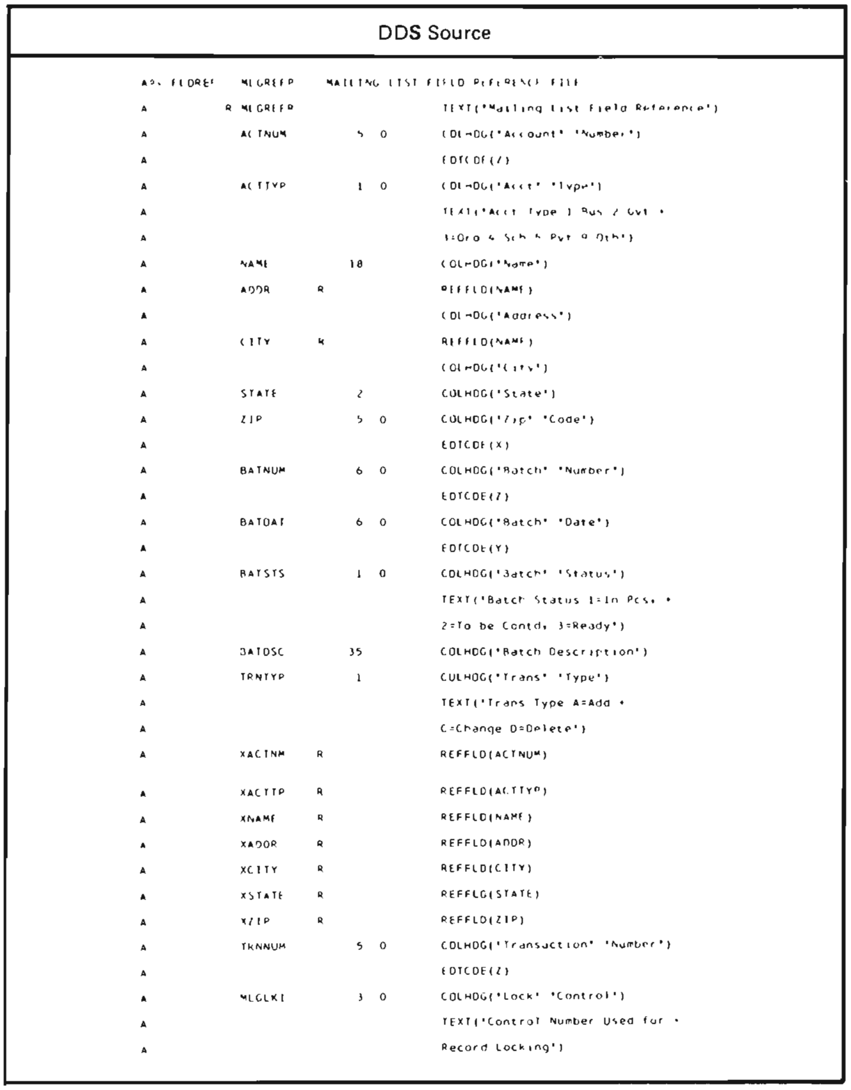
The TRNNUM field contains the transaction number. This is a consecutive number which will be assigned by the data file utility (DFU).

The MLGLK1 field is used in a later approach and will be explained then (see Chapter 11).

The EDTCDE keyword is used on most of the numeric fields to define the proper editing of the field. This is the default when the field is displayed or used by DFU or by Query. The edit code X for the ZIP field means that no editing will be performed and is used to prevent any default editing that may be applied by DFU or Query. The edit code Z is used to suppress the leading zeros in the ACTNUM, BATNUM, and TRNNUM fields. Edit codes are described in the *CPF Reference Manual—DDS*.

When you have entered all the source shown in Figure 6-2, and exit SEU, the programmer menu is displayed again. MLGREFP is now a member of the source file as shown in Figure 6-3.

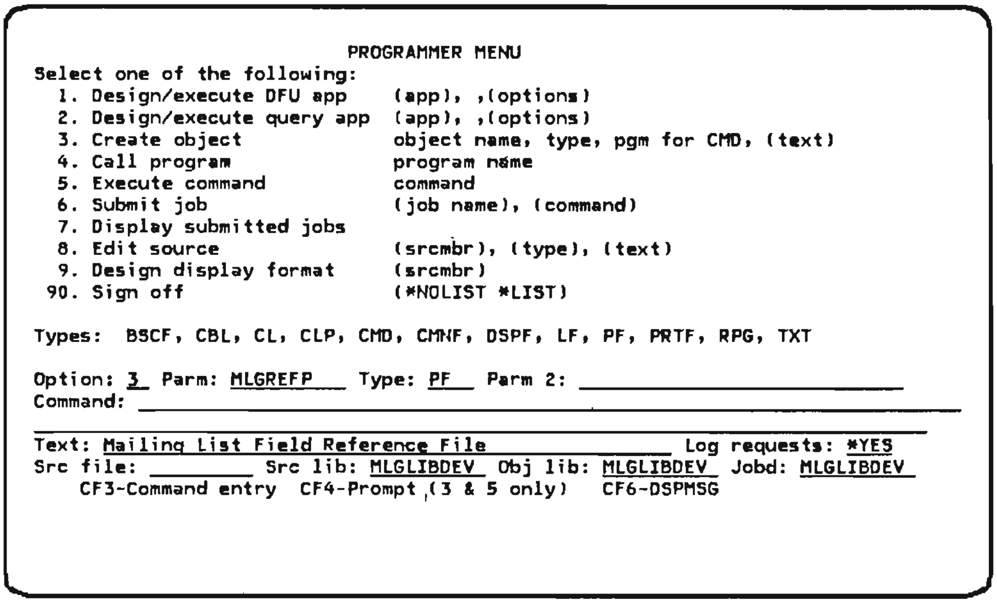



DDS Source

```
A*  FLDREF     MLGREFP    MAILING LIST FIELD REFERENCE FILE
A          R MLGREFP                     TEXT('Mailing List Field Reference')
A            ACTNUM          5  0        COLHDG('Account' 'Number')
A                                        EDTCDE(Z)
A            ACTTYP          1  0        COLHDG('Acct' 'Type')
A                                        TEXT('Acct Type 1 Bus 2 Gvt +
A                                        3=Org 4 Sch 5 Pvt 9 Oth')
A            NAME           18           COLHDG('Name')
A            ADDR      R                 REFFLD(NAME)
A                                        COLHDG('Address')
A            CITY      R                 REFFLD(NAME)
A                                        COLHDG('City')
A            STATE           2           COLHDG('State')
A            ZIP             5  0        COLHDG('Zip' 'Code')
A                                        EDTCDE(X)
A            BATNUM          6  0        COLHDG('Batch' 'Number')
A                                        EDTCDE(Z)
A            BATDAT          6  0        COLHDG('Batch' 'Date')
A                                        EDTCDE(Y)
A            BATSTS          1  0        COLHDG('Batch' 'Status')
A                                        TEXT('Batch Status 1=In Pcs, +
A                                        2=To be Contd, 3=Ready')
A            BATDSC         35           COLHDG('Batch Description')
A            TRNTYP          1           COLHDG('Trans' 'Type')
A                                        TEXT('Trans Type A=Add +
A                                        C=Change D=Delete')
A            XACTNM    R                 REFFLD(ACTNUM)

A            XACTTP    R                 REFFLD(ACTTYP)
A            XNAME     R                 REFFLD(NAME)
A            XADDR     R                 REFFLD(ADDR)
A            XCITY     R                 REFFLD(CITY)
A            XSTATE    R                 REFFLG(STATE)
A            XZIP      R                 REFFLD(ZIP)
A            TRNNUM          5  0        COLHDG('Transaction' 'Number')
A                                        EDTCDE(Z)
A            MLGLKI          3  0        COLHDG('Lock' 'Control')
A                                        TEXT('Control Number Used for +
A                                        Record Locking')
```

Figure 6-3. QDDSSRC Contains Member MLGREFP, which Contains DDS Source Statements

The physical file MLGREFP can now be created using the source in member MLGREFP of file QDDSSRC (Figure 6-3). Because you previously entered the name (MLGREFP) and the type (PF) on the programmer menu when you entered the source, you need only enter a 3 for the option number to create MLGREFP:



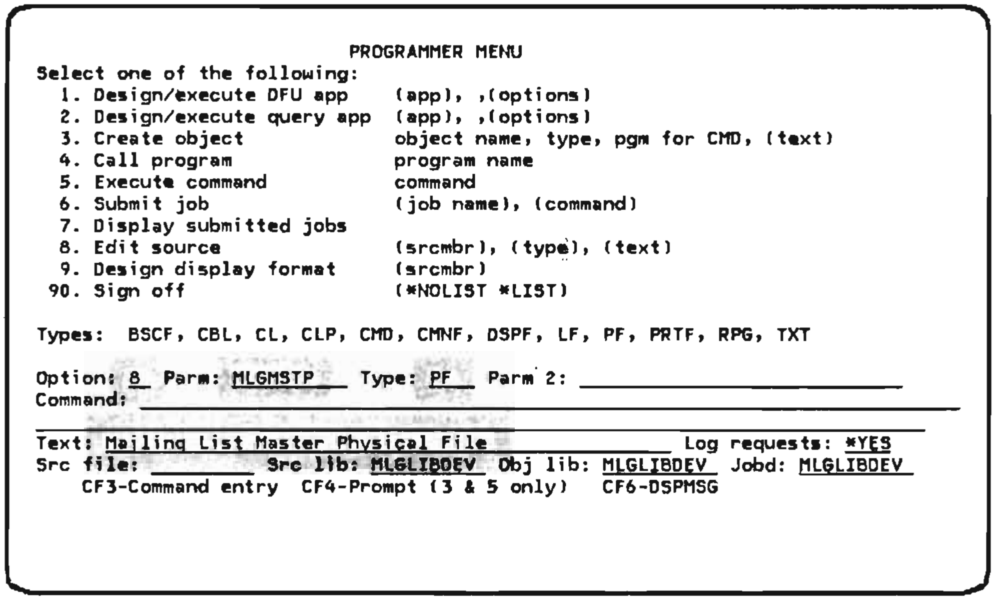
```
                              PROGRAMMER MENU
Select one of the following:
  1. Design/execute DFU app    (app), ,(options)
  2. Design/execute query app  (app), ,(options)
  3. Create object             object name, type, pgm for CMD, (text)
  4. Call program              program name
  5. Execute command           command
  6. Submit job                (job name), (command)
  7. Display submitted jobs
  8. Edit source               (srcmbr), (type), (text)
  9. Design display format     (srcmbr)
 90. Sign off                  (*NOLIST *LIST)

Types:  BSCF, CBL, CL, CLP, CMD, CMNF, DSPF, LF, PF, PRTF, RPG, TXT

Option: 3_ Parm: MLGREFP___ Type: PF__ Parm 2: ____________________________
Command: ___________________________________________________________________
____________________________________________________________________________
Text: Mailing List Field Reference File________________ Log requests: *YES
Src file: _________ Src lib: MLGLIBDEV_ Obj lib: MLGLIBDEV_ Jobd: MLGLIBDEV_
    CF3-Command entry  CF4-Prompt (3 & 5 only)   CF6-DSPMSG
```

Master Files

The master files used in the mailing list application are a physical file, MLGMSTP, and a logical file, MLGMSTL. MLGMSTP contains the master data for the mailing list application and MLGMSTL provides the format and access path to access the master data. Accessing the master data through a logical file can help you achieve data independence.

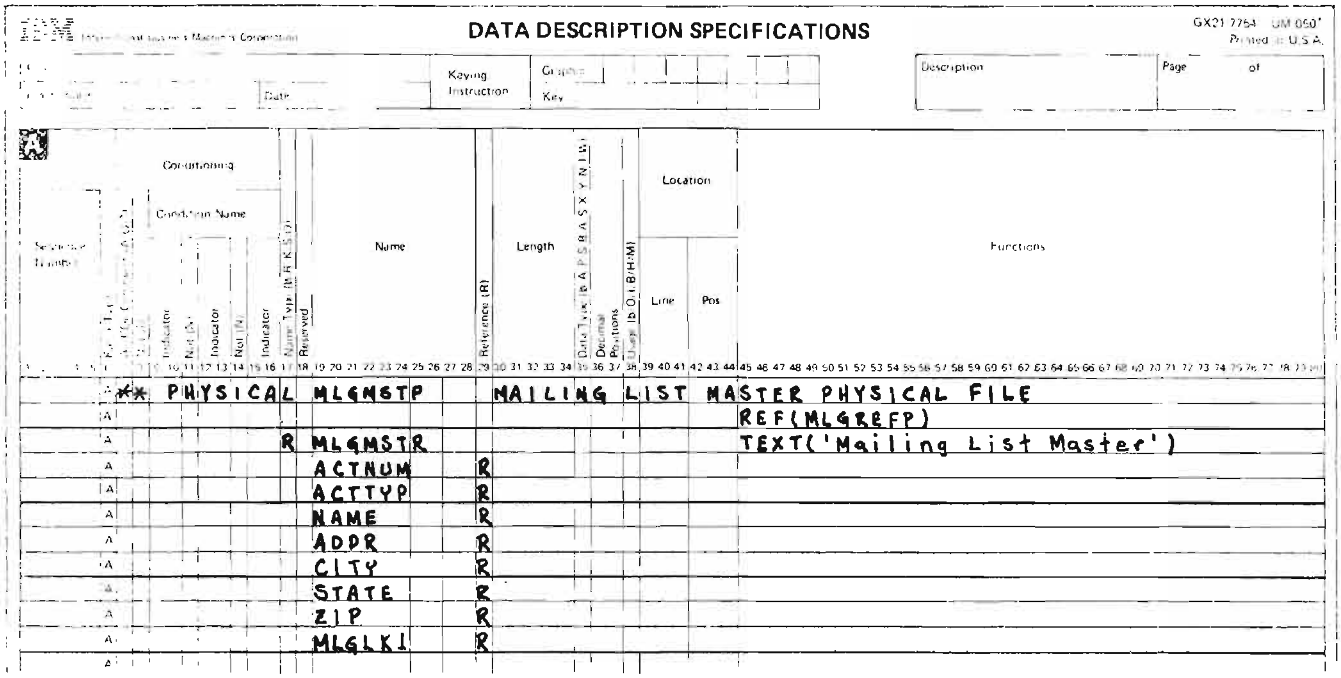
One of the advantages of System/38 data base support is the ability to achieve data independence. This means that any program using a specific record format does not need to be recompiled if changes occur to the physical data format but not to the logical record format used by the program. For example, if a field is added to the file such that the locations of some of the original fields are changed, the existing programs can be used without being recompiled. Refer to the *CPF Programmer's Guide* for information on how to do this.

Because MLGMSTL is a logical file used to access the data in MLGMSTP, the definition for MLGMSTP must be created first. To enter the source for MLGMSTP, you request SEU from the programmer menu as follows:



```
                              PROGRAMMER MENU
Select one of the following:
  1. Design/execute DFU app    (app), ,(options)
  2. Design/execute query app  (app), ,(options)
  3. Create object             object name, type, pgm for CMD, (text)
  4. Call program              program name
  5. Execute command           command
  6. Submit job                (job name), (command)
  7. Display submitted jobs
  8. Edit source               (srcmbr), (type), (text)
  9. Design display format     (srcmbr)
 90. Sign off                  (*NOLIST *LIST)

Types:  BSCF, CBL, CL, CLP, CMD, CMNF, DSPF, LF, PF, PRTF, RPG, TXT

Option: 8_ Parm: MLGMSTP___ Type: PF__ Parm 2: ___________________________
Command: __________________________________________________________________
___________________________________________________________________________
Text: Mailing List Master Physical File_______________ Log requests: *YES
Src file: _________ Src lib: MLGLIBDEV_ Obj lib: MLGLIBDEV_ Jobd: MLGLIBDEV_
    CF3-Command entry  CF4-Prompt (3 & 5 only)   CF6-DSPMSG
```

You receive an SEU display that allows you to enter the DDS source for MLGMSTP. The DDS to be entered are shown in Figure 6-4. The REF keyword specifies the field reference file (MLGREFP) previously created. An R in position 29 specifies that the field reference function is being used. This tells the system to copy the complete description of the field (including COLHDG and TEXT) from the field reference file. No key field is specified for this file.

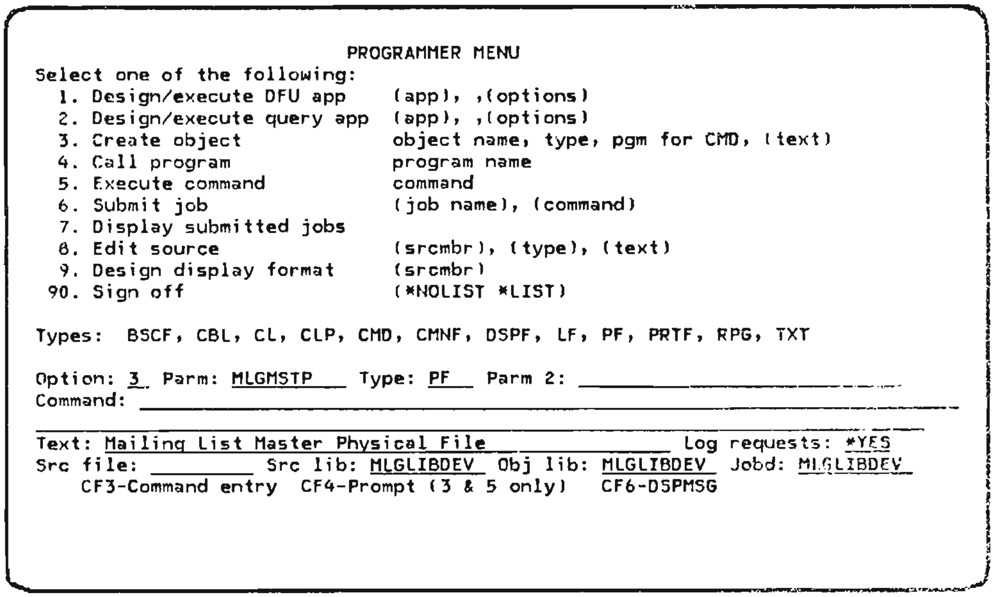


DATA DESCRIPTION SPECIFICATIONS

```
     A** PHYSICAL MLGMSTP  MAILING LIST MASTER PHYSICAL FILE
     A                                        REF(MLGREFP)
     A          R MLGMSTR                     TEXT('Mailing List Master')
     A            ACTNUM    R
     A            ACTTYP    R
     A            NAME      R
     A            ADDR      R
     A            CITY      R
     A            STATE     R
     A            ZIP       R
     A            MLGLKI    R
     A
```

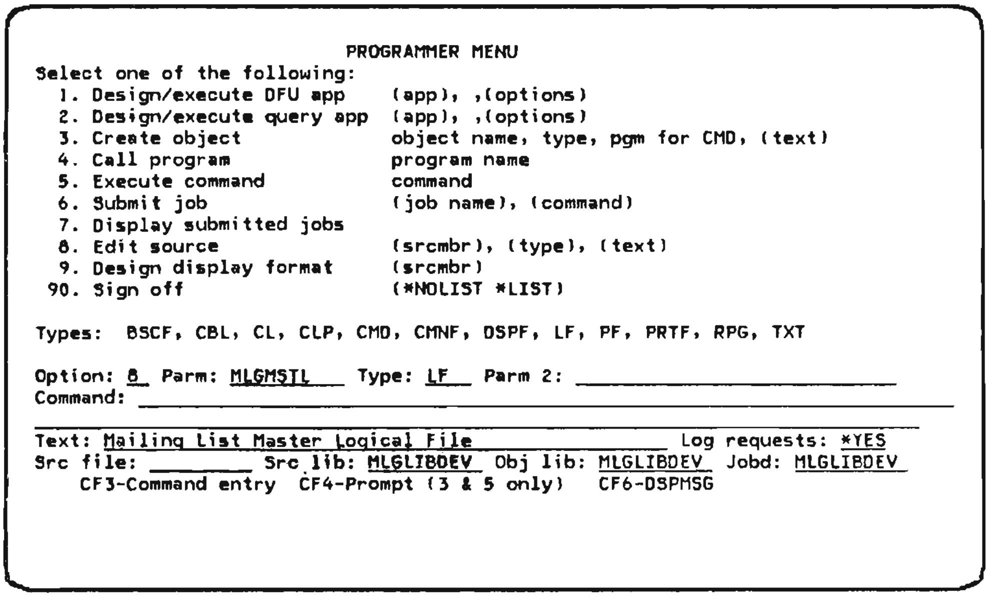
Figure 6-4. DDS for MLGMSTP File

When you exit SEU after entering the DDS for MLGMSTP, the programmer menu is displayed again. To create the physical file MLGMSTP, you need only select option 3:



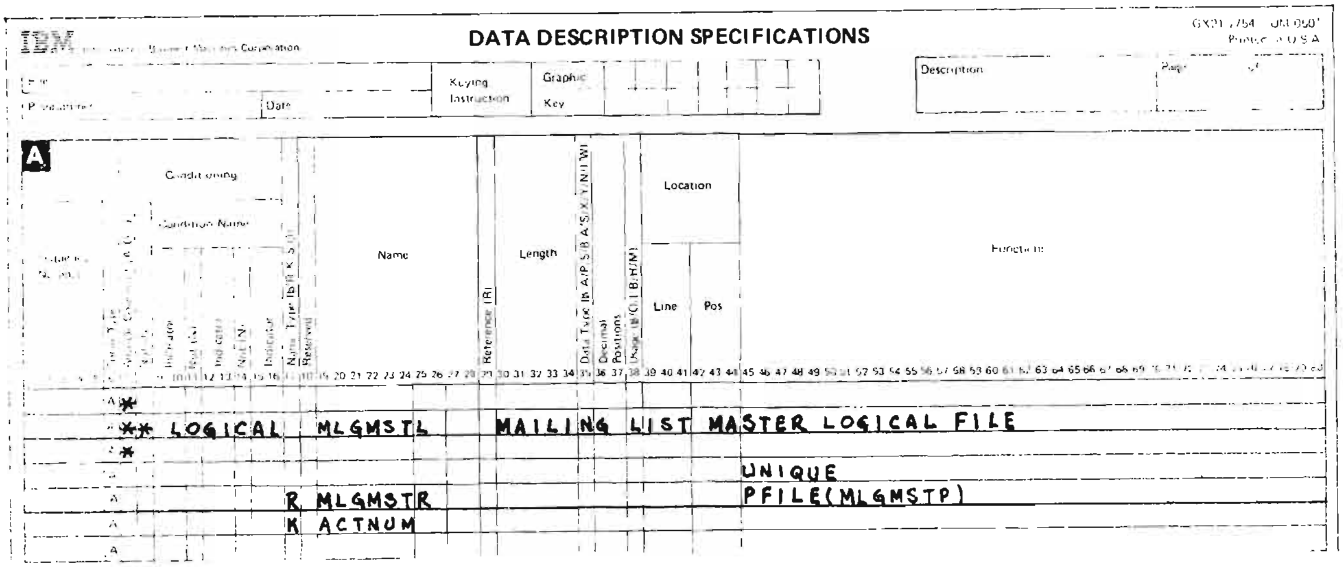
```
                              PROGRAMMER MENU
Select one of the following:
  1. Design/execute DFU app     (app), ,(options)
  2. Design/execute query app   (app), ,(options)
  3. Create object              object name, type, pgm for CMD, (text)
  4. Call program               program name
  5. Execute command            command
  6. Submit job                 (job name), (command)
  7. Display submitted jobs
  8. Edit source                (srcmbr), (type), (text)
  9. Design display format      (srcmbr)
 90. Sign off                   (*NOLIST *LIST)

Types:  BSCF, CBL, CL, CLP, CMD, CMNF, DSPF, LF, PF, PRTF, RPG, TXT

Option: 3  Parm: MLGMSTP    Type: PF   Parm 2: ____________________________
Command: ____________________________________________________________________

Text: Mailing List Master Physical File               Log requests: *YES
Src file: _________  Src lib: MLGLIBDEV  Obj lib: MLGLIBDEV  Jobd: MLGLIBDEV
    CF3-Command entry  CF4-Prompt (3 & 5 only)   CF6-DSPMSG
```

Now, the definition for MLGMSTL is created. You request SEU from the programmer menu as follows:



```
                          PROGRAMMER MENU
Select one of the following:
  1. Design/execute DFU app    (app), ,(options)
  2. Design/execute query app  (app), ,(options)
  3. Create object             object name, type, pgm for CMD, (text)
  4. Call program              program name
  5. Execute command           command
  6. Submit job                (job name), (command)
  7. Display submitted jobs
  8. Edit source               (srcmbr), (type), (text)
  9. Design display format     (srcmbr)
 90. Sign off                  (*NOLIST *LIST)

Types:  BSCF, CBL, CL, CLP, CMD, CMNF, DSPF, LF, PF, PRTF, RPG, TXT

Option: 8  Parm: MLGMSTL    Type: LF   Parm 2: ______________________
Command: ____________________________________________________________

Text: Mailing List Master Logical File                 Log requests: *YES
Src file: ________ Src lib: MLGLIBDEV  Obj lib: MLGLIBDEV  Jobd: MLGLIBDEV
    CF3-Command entry  CF4-Prompt (3 & 5 only)   CF6-DSPMSG
```

You receive an SEU display that allows you to enter the DDS for MLGMSTL. The DDS to be entered are shown in Figure 6-5.



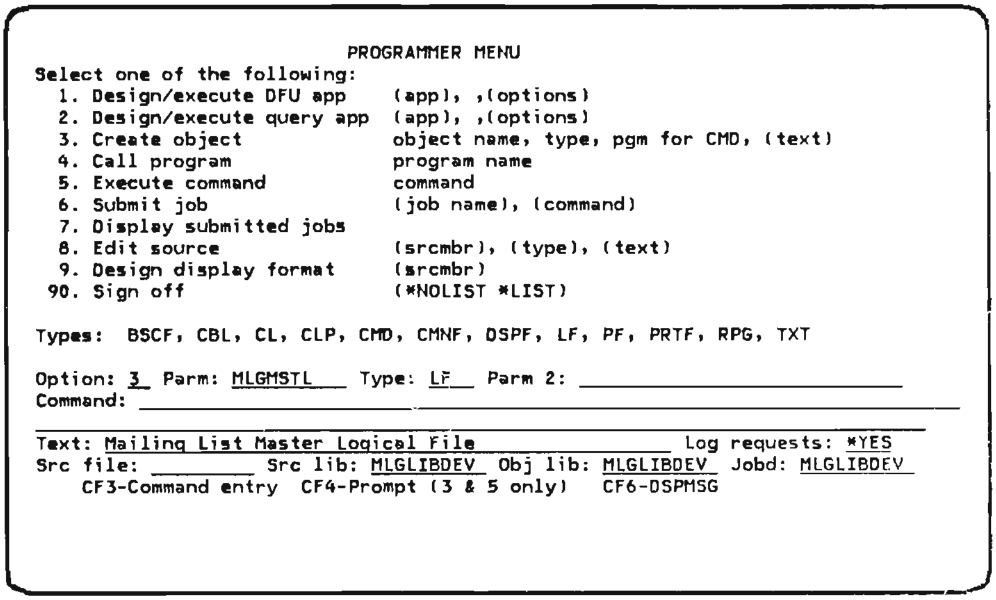
IBM DATA DESCRIPTION SPECIFICATIONS

```
     A*
     A** LOGICAL  MLGMSTL      MAILING LIST MASTER LOGICAL FILE
     A*
     A                                      UNIQUE
     A          R MLGMSTR                   PFILE(MLGMSTP)
     A          K ACTNUM
     A
```

Figure 6-5. DDS for MLGMSTL File

The keyword UNIQUE specifies that duplicate key values are not allowed within a member of this logical file. The keyword PFILE identifies the physical file containing the data to be accessed through this logical file. Because individual fields are not specified for the logical file record format, the fields will be the same as the physical file MLGMSTP (see Figure 6-4). The K in position 17 identifies ACTNUM as a key field.

After the source for MLGMSTL is entered, the logical file MLGMSTL can be created by selecting option 3 on the programmer menu:

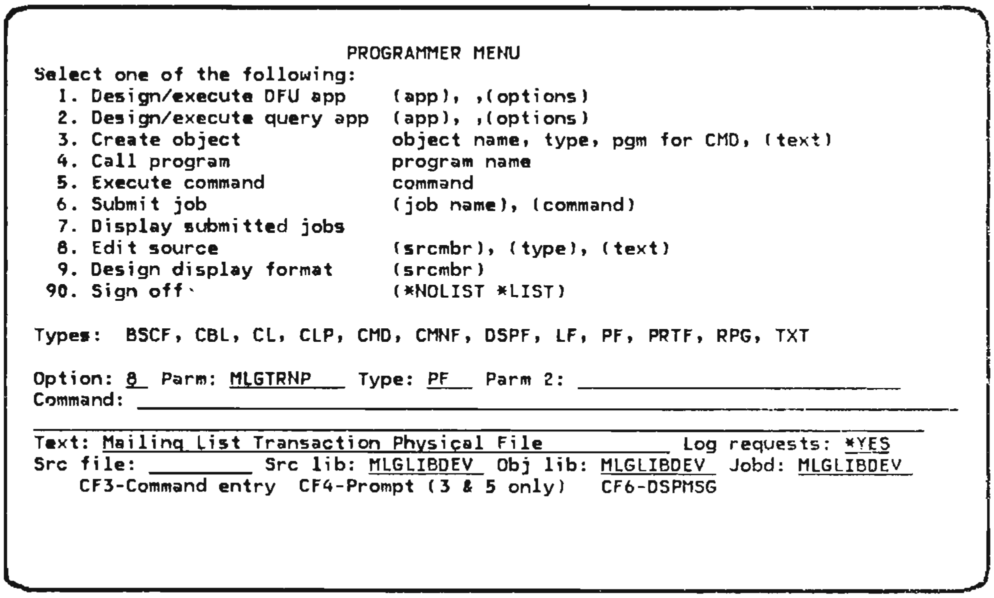


```
                            PROGRAMMER MENU
Select one of the following:
  1. Design/execute DFU app    (app), ,(options)
  2. Design/execute query app  (app), ,(options)
  3. Create object             object name, type, pgm for CMD, (text)
  4. Call program              program name
  5. Execute command           command
  6. Submit job                (job name), (command)
  7. Display submitted jobs
  8. Edit source               (srcmbr), (type), (text)
  9. Design display format     (srcmbr)
 90. Sign off                  (*NOLIST *LIST)

Types:  BSCF, CBL, CL, CLP, CMD, CMNF, DSPF, LF, PF, PRTF, RPG, TXT

Option: 3_ Parm: MLGMSTL___ Type: LF__ Parm 2: ____________________________
Command: ____________________________________________________________________
_________________________________________________________________________
Text: Mailing List Master Logical File________________ Log requests: *YES
Src file: _________ Src lib: MLGLIBDEV_ Obj lib: MLGLIBDEV_ Jobd: MLGLIBDEV_
    CF3-Command entry  CF4-Prompt (3 & 5 only)   CF6-DSPMSG
```

Transaction Files

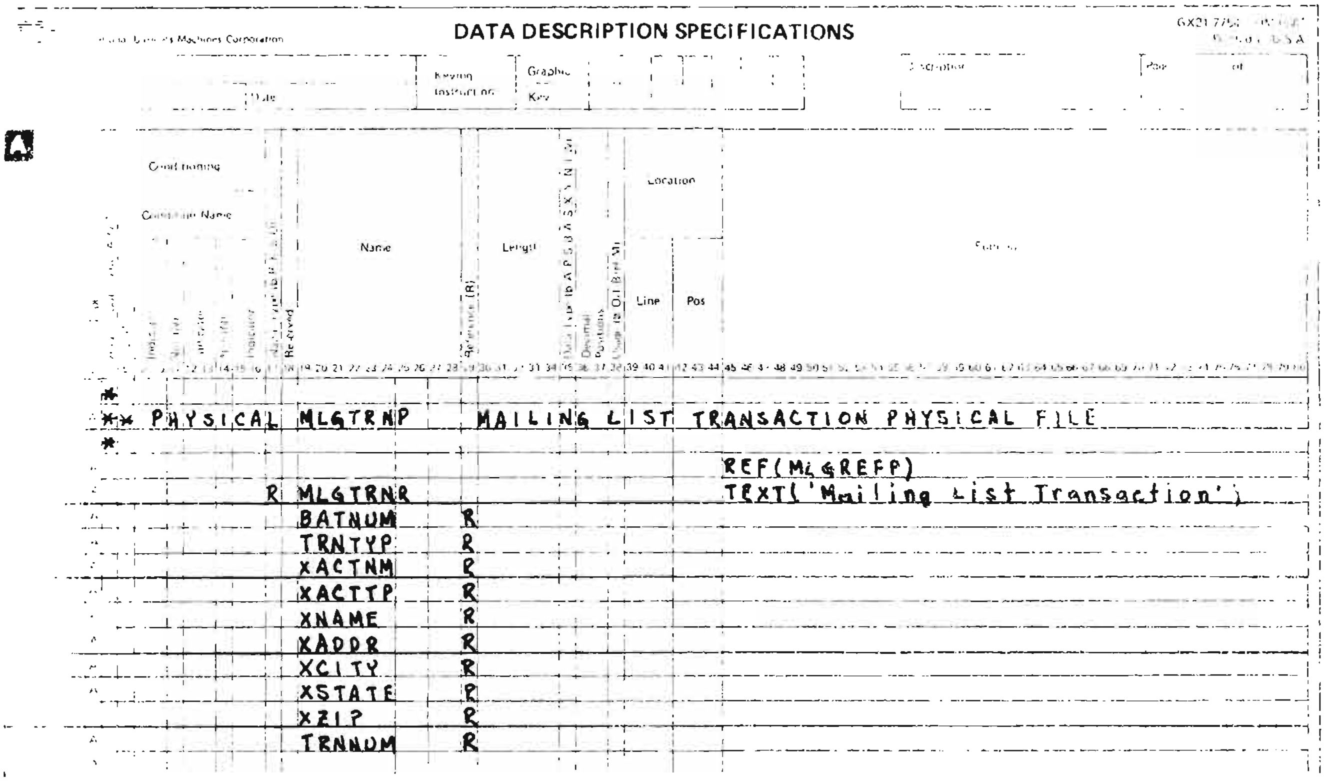
The transaction files are a physical file, MLGTRNP, and a logical file, MLGTRNL. MLGTRNP contains transactions to be applied to the mailing list master file. MLGTRNL provides the access path to the transaction data.

MLGTRNP is created first. Request SEU from the programmer menu as follows:



```
                            PROGRAMMER MENU
Select one of the following:
  1. Design/execute DFU app    (app), ,(options)
  2. Design/execute query app  (app), ,(options)
  3. Create object             object name, type, pgm for CMD, (text)
  4. Call program              program name
  5. Execute command           command
  6. Submit job                (job name), (command)
  7. Display submitted jobs
  8. Edit source               (srcmbr), (type), (text)
  9. Design display format     (srcmbr)
 90. Sign off                  (*NOLIST *LIST)

Types:  BSCF, CBL, CL, CLP, CMD, CMNF, DSPF, LF, PF, PRTF, RPG, TXT

Option: 8_ Parm: MLGTRNP___ Type: PF__ Parm 2: ____________________________
Command: ____________________________________________________________________
_________________________________________________________________________
Text: Mailing List Transaction Physical File__________ Log requests: *YES
Src file: _________ Src lib: MLGLIBDEV_ Obj lib: MLGLIBDEV_ Jobd: MLGLIBDEV_
    CF3-Command entry  CF4-Prompt (3 & 5 only)   CF6-DSPMSG
```

You receive an SEU display that allows you to enter the DDS for MLGTRNP. The DDS to be entered are shown in Figure 6-6.



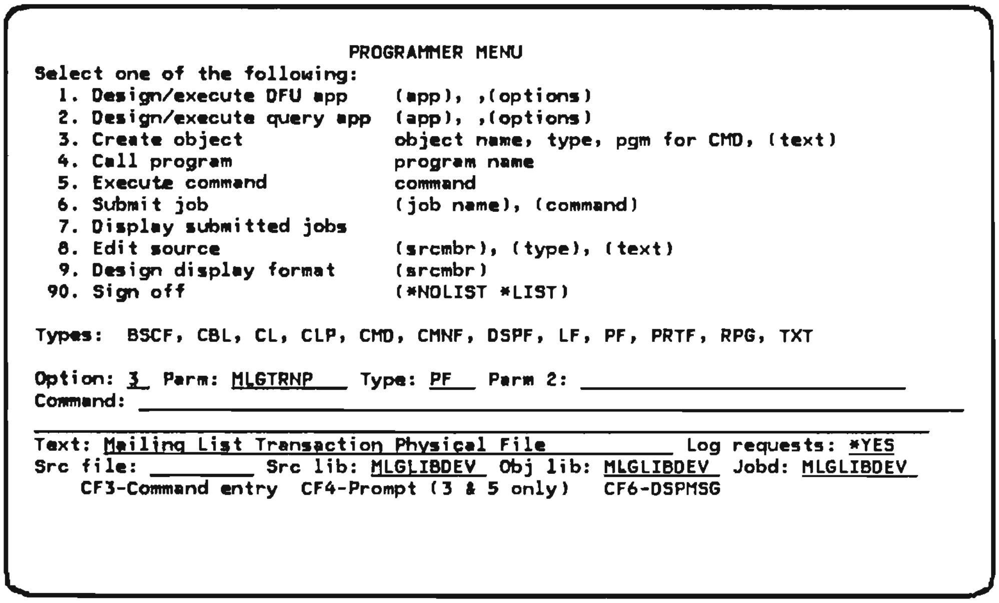
DATA DESCRIPTION SPECIFICATIONS

```
     *
     ** PHYSICAL MLGTRNP      MAILING LIST TRANSACTION PHYSICAL FILE
     *
                                            REF(MLGREFP)
                R MLGTRNR                   TEXT('Mailing List Transaction')
                  BATNUM    R
                  TRNTYP    R
                  XACTNM    R
                  XACTTP    R
                  XNAME     R
                  XADDR     R
                  XCITY     R
                  XSTATE    R
                  XZIP      R
                  TRNNUM    R
```

Figure 6-6. DDS for MLGTRNP File

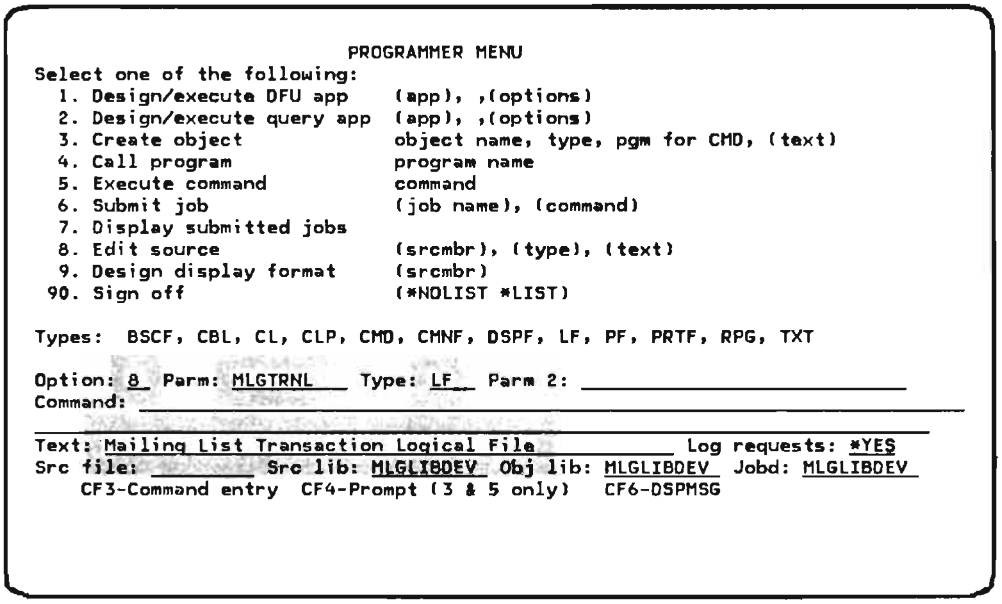
A record format (MLGTRNR) is defined with the same field names that were used in the transaction records discussed in Chapter 4. A transaction number (TRNNUM) field is added; this field will contain a consecutive number assigned by DFU. The keyword REF and the R in position 29 specify that the field reference file is used.

After the DDS for MLGTRNP are entered, the physical file MLGTRNP can be created by selecting option 3 on the programmer menu:



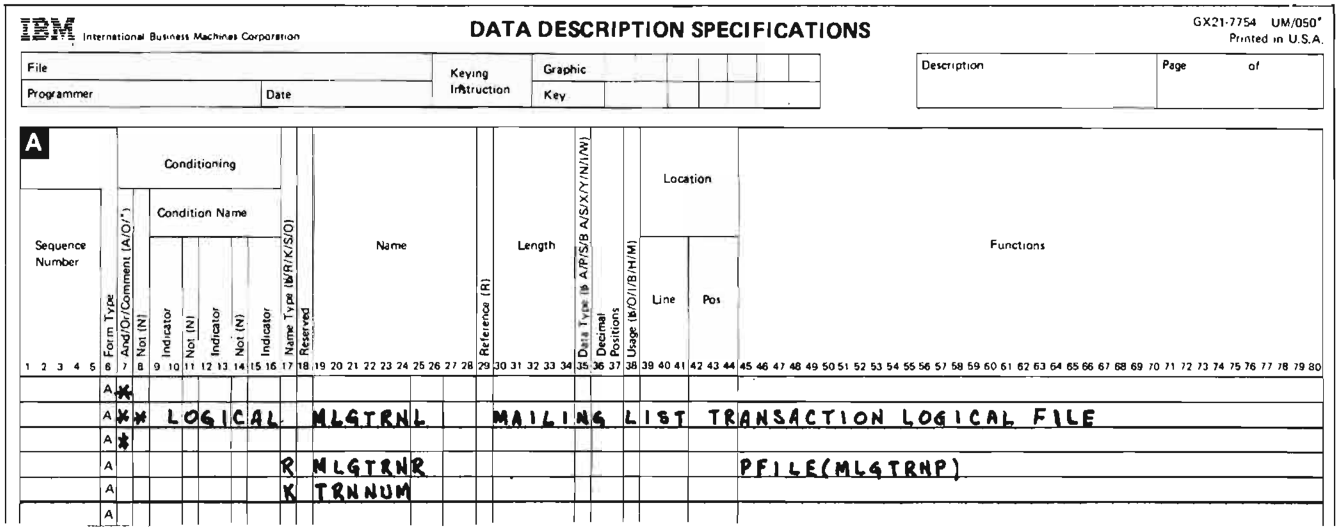
```
                              PROGRAMMER MENU
Select one of the following:
  1. Design/execute DFU app     (app), ,(options)
  2. Design/execute query app   (app), ,(options)
  3. Create object              object name, type, pgm for CMD, (text)
  4. Call program               program name
  5. Execute command            command
  6. Submit job                 (job name), (command)
  7. Display submitted jobs
  8. Edit source                (srcmbr), (type), (text)
  9. Design display format      (srcmbr)
 90. Sign off                   (*NOLIST *LIST)

Types:  BSCF, CBL, CL, CLP, CMD, CMNF, DSPF, LF, PF, PRTF, RPG, TXT

Option: 3_ Parm: MLGTRNP___ Type: PF__ Parm 2: ____________________________
Command: ____________________________________________________________________
____________________________________________________________________________
Text: Mailing List Transaction Physical File__________ Log requests: *YES
Src file: _________ Src lib: MLGLIBDEV_ Obj lib: MLGLIBDEV_ Jobd: MLGLIBDEV_
    CF3-Command entry  CF4-Prompt (3 & 5 only)   CF6-DSPMSG
```

Now MLGTRNL can be created. Request SEU as follows:



```
                              PROGRAMMER MENU
Select one of the following:
  1. Design/execute DFU app     (app), ,(options)
  2. Design/execute query app   (app), ,(options)
  3. Create object              object name, type, pgm for CMD, (text)
  4. Call program               program name
  5. Execute command            command
  6. Submit job                 (job name), (command)
  7. Display submitted jobs
  8. Edit source                (srcmbr), (type), (text)
  9. Design display format      (srcmbr)
 90. Sign off                   (*NOLIST *LIST)

Types:  BSCF, CBL, CL, CLP, CMD, CMNF, DSPF, LF, PF, PRTF, RPG, TXT

Option: 8_ Parm: MLGTRNL___ Type: LF__ Parm 2: ____________________________
Command: ____________________________________________________________________
____________________________________________________________________________
Text: Mailing List Transaction Logical File___________ Log requests: *YES
Src file: _________ Src lib: MLGLIBDEV_ Obj lib: MLGLIBDEV_ Jobd: MLGLIBDEV_
    CF3-Command entry  CF4-Prompt (3 & 5 only)   CF6-DSPMSG
```

You receive an SEU display that allows you to enter the DDS for MLGTRNL. The DDS to be entered are shown in Figure 6-7.



IBM International Business Machines Corporation

DATA DESCRIPTION SPECIFICATIONS

GX21-7754 UM/050*
Printed in U.S.A.

File | Keying Instruction | Graphic | Key
Programmer | Date
Description | Page of

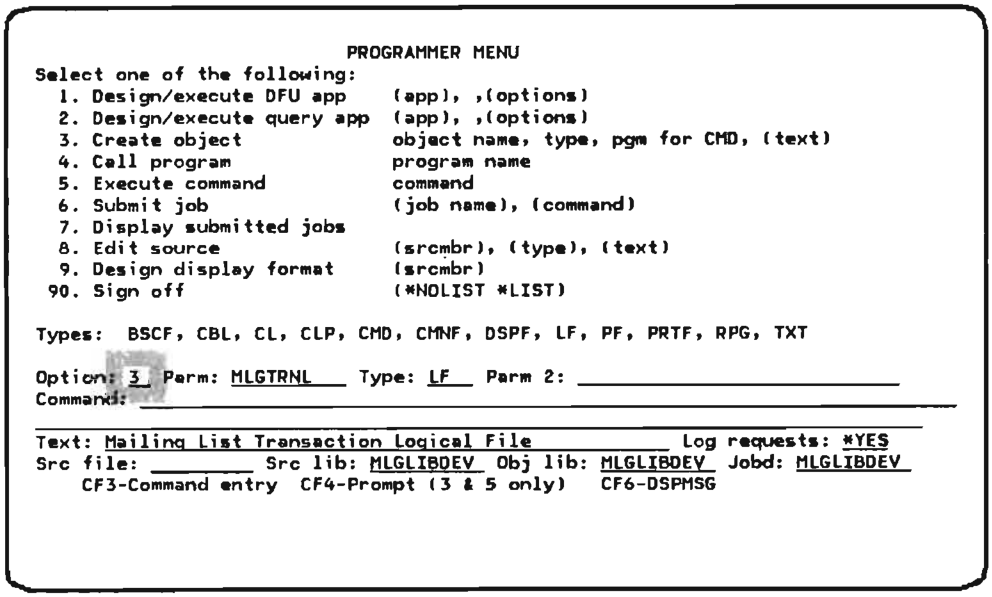
```
A*
A** LOGICAL  MLGTRNL     MAILING LIST TRANSACTION LOGICAL FILE
A*
A           R MLGTRNR                  PFILE(MLGTRNP)
A           K TRNNUM
A
```

A
Conditioning
Condition Name
Sequence Number
Form Type
And/Or/Comment (A/O/*)
Not (N)
Indicator
Not (N)
Indicator
Not (N)
Indicator
Name Type (b/R/K/S/O)
Reserved
Name
Reference (R)
Length
Data Type (b A/P/S/B A/S/X/Y/N/I/W)
Decimal Positions
Usage (b/O/I/B/H/M)
Location
Line
Pos
Functions
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80

Figure 6-7. DDS for MLGTRNL File

The PFILE keyword specifies the format (MLGTRNP) over the physical data. The K in position 17 identifies TRNNUM as a key field. Because the DDS do not specify individual fields for the logical file record format, the fields will be the same as in the record format of the physical file MLGTRNP (see Figure 6-6).

After the DDS for MLGTRNL are entered, the logical file MLGTRNL can be created by selecting option 3 on the programmer menu:



```
                            PROGRAMMER MENU
Select one of the following:
  1. Design/execute DFU app     (app), ,(options)
  2. Design/execute query app   (app), ,(options)
  3. Create object              object name, type, pgm for CMD, (text)
  4. Call program               program name
  5. Execute command             command
  6. Submit job                 (job name), (command)
  7. Display submitted jobs
  8. Edit source                (srcmbr), (type), (text)
  9. Design display format      (srcmbr)
 90. Sign off                   (*NOLIST *LIST)

Types:  BSCF, CBL, CL, CLP, CMD, CMNF, DSPF, LF, PF, PRTF, RPG, TXT

Option: 3  Parm: MLGTRNL     Type: LF   Parm 2:
Command:

Text: Mailing List Transaction Logical File            Log requests: *YES
Src file:           Src lib: MLGLIBDEV  Obj lib: MLGLIBDEV  Jobd: MLGLIBDEV
    CF3-Command entry  CF4-Prompt (3 & 5 only)   CF6-DSPMSG
```

A Review of Files and Their Usage

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

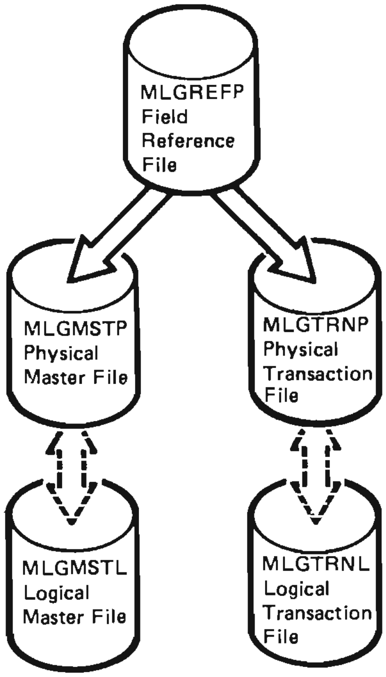
- If a field reference file is to be used, it must be created before any physical files are created.
- Physical files must be created before any logical files that use them are created.
- Externally described data files must be created before the program that uses them is created.

A field reference file will normally grow as an application is developed. Changes to the field reference file do not require that physical and logical files be re-created unless the change impacts the physical and logical files.

A summary of the files used by this application at this point follows.

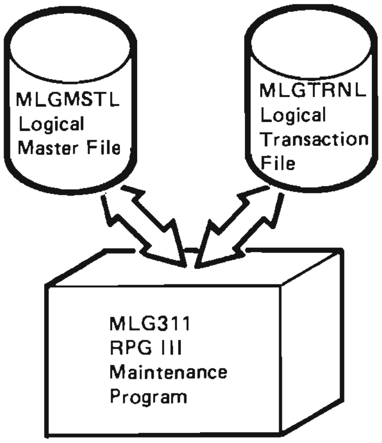
When you define files, all of the files shown below are used and they must be created in the correct sequence. When the logical files are created, the field reference file is not used.

When Defining Files



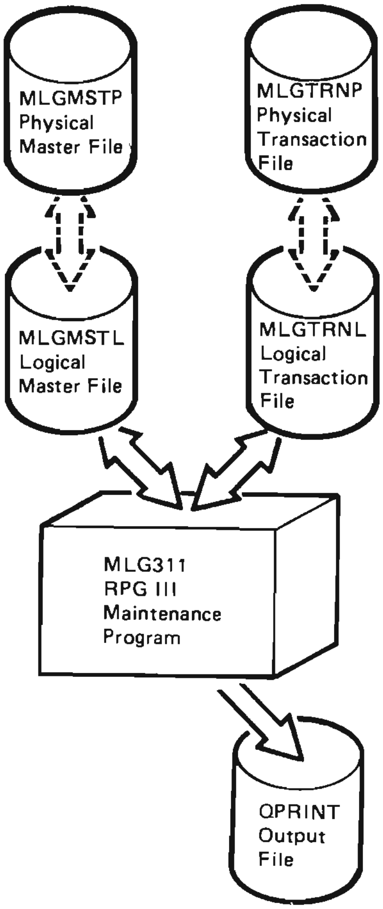
When you create programs, only the externally described files are used, as shown below (MLG311 will be discussed and used in Chapter 7). Note that the field reference file, physical files, or program described output file (QPRINT) are not used.

When Creating Programs



When you execute programs, all of the files except the field reference file are used, as shown below.

When Executing Programs



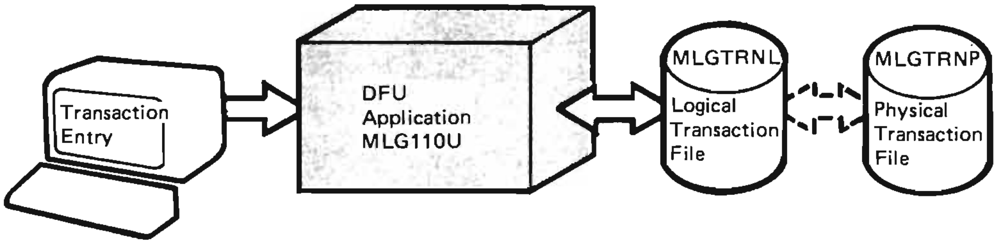
Chapter 7. Interactive Input and Batch Maintenance

OVERVIEW

In the pervious approach of Chapter 4, master records in a data base file were updated from transaction records that were read from diskette. The approach in this chapter allows transactions to be entered from a work station into a data base transaction file. The group of records entered into the transaction file is then applied to the data base master file by a maintenance program.

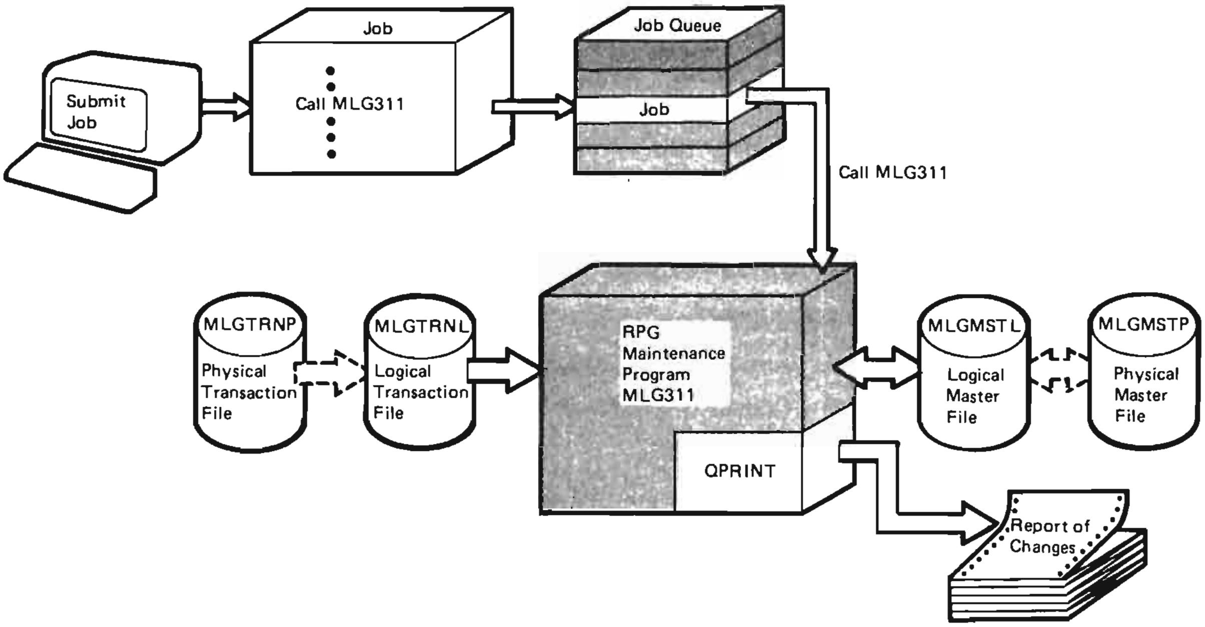
The transaction entry part of this approach consists of the following:

- A work station user enters transactions using a DFU application, MLG110U.
- The transaction records entered at the work station are stored in the physical transaction file, MLGTRNP. A logical transaction file, MLGTRNL, provides the access path between the MLG110U application and the physical transaction file.



The maintenance part of this approach consists of the following:

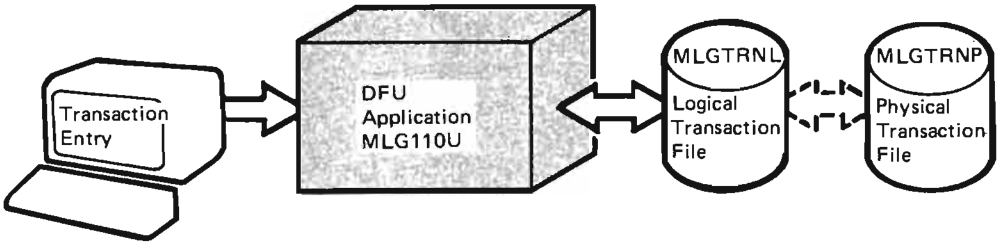
- A work station user invokes the RPG III maintenance program, MLG311, by submitting a batch job that calls MLG311.
- The MLG311 program takes the records from the transaction file and applies them to the master file. A logical transaction file, MLGTRNL, provides the access path between the MLG311 program and the physical transaction file, MLGTRNP. Similarly, a logical master file, MLGMSTL, provides the access path between the program and the physical master file, MLGMSTP.



ENTERING TRANSACTIONS

In Chapter 4, the transaction records were on diskette, and the fields of a record were described on input specifications for the RPG III maintenance program. In this approach, transactions are entered at the work station into a data base file, MLGTRNP, which also contains a description of the record format.

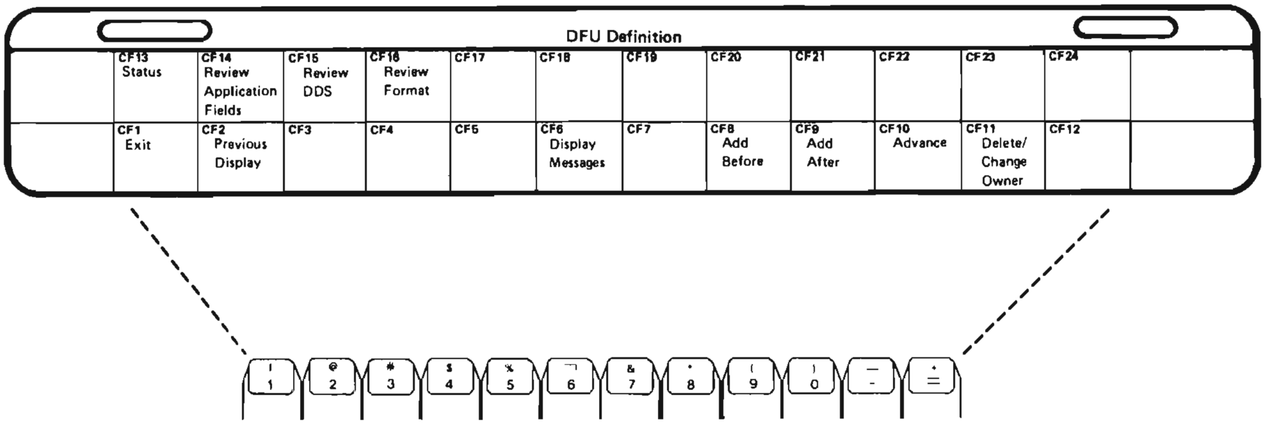
The transactions are placed in MLGTRNP by a *data file utility* (DFU) application, MLG110U. DFU is the part of the IDU that is used to create, maintain, and display records in a data base file. A DFU application for a particular file (in this case, a transaction file) is created by responding to a series of prompts. That file must be an externally described data file and must be processed in *keyed sequence*, that is, based on the contents of key fields contained in the records.



Creating the DFU Application

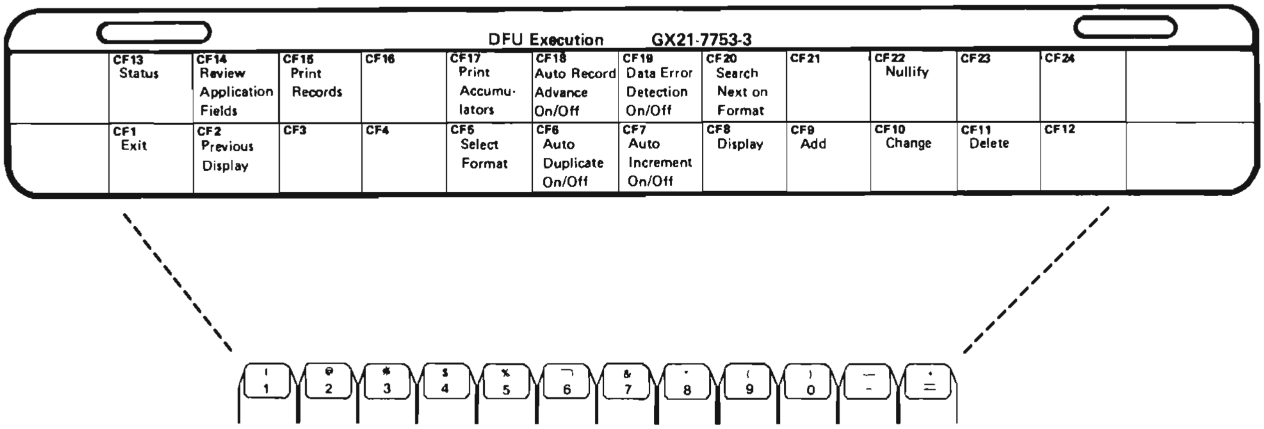
The first step in implementing this approach is to create the DFU application, MLG110U, so that a work station user can enter transactions into the transaction file.

When creating or executing DFU applications, you may find the *Keyboard Template* (GX21-7756) useful. This set of templates identifies command function keys available for DFU, as well as for CPF, query, SDA, and SEU.



DFU Definition

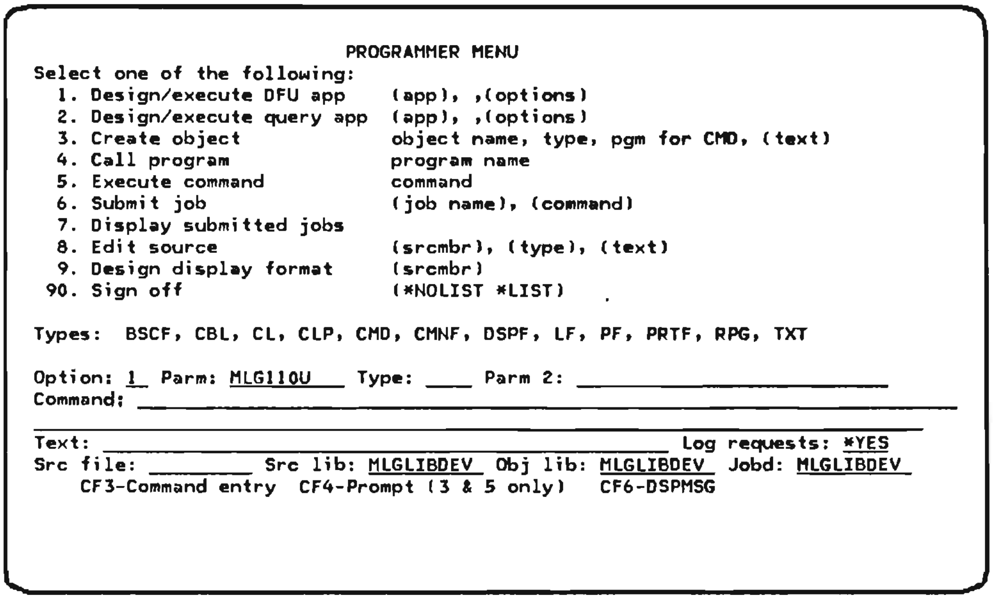
CF13 Status	CF14 Review Application Fields	CF15 Review DDS	CF16 Review Format	CF17	CF18	CF19	CF20	CF21	CF22	CF23	CF24
CF1 Exit	CF2 Previous Display	CF3	CF4	CF5	CF6 Display Messages	CF7	CF8 Add Before	CF9 Add After	CF10 Advance	CF11 Delete/ Change Owner	CF12



DFU Execution GX21-7753-3

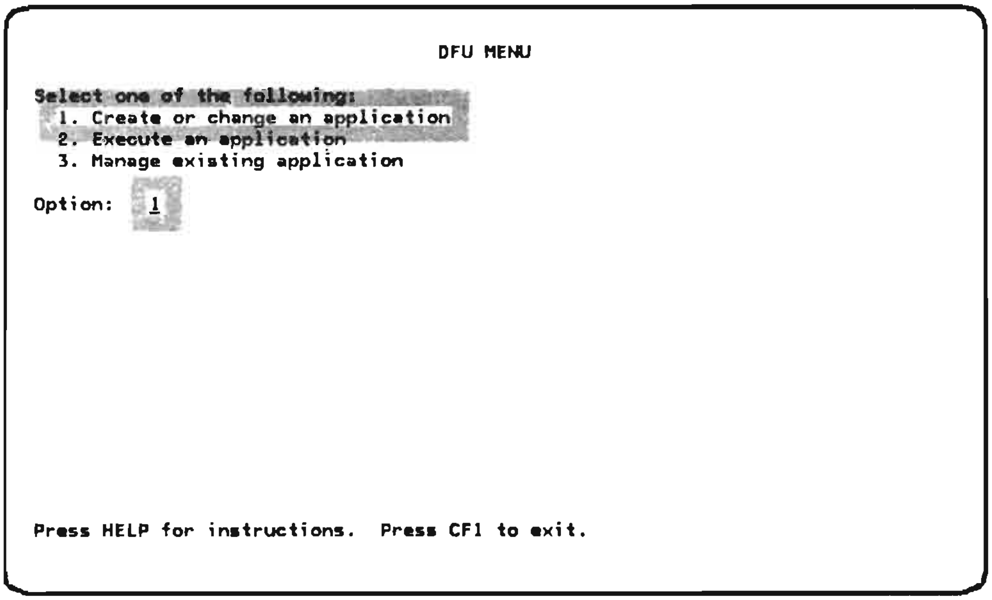
CF13 Status	CF14 Review Application Fields	CF15 Print Records	CF16	CF17 Print Accumu-lators	CF18 Auto Record Advance On/Off	CF19 Data Error Detection On/Off	CF20 Search Next on Format	CF21	CF22 Nullify	CF23	CF24
CF1 Exit	CF2 Previous Display	CF3	CF4	CF5 Select Format	CF6 Auto Duplicate On/Off	CF7 Auto Increment On/Off	CF8 Display	CF9 Add	CF10 Change	CF11 Delete	CF12

To create the DFU application MLG110U, you request DFU from the programmer menu as follows:



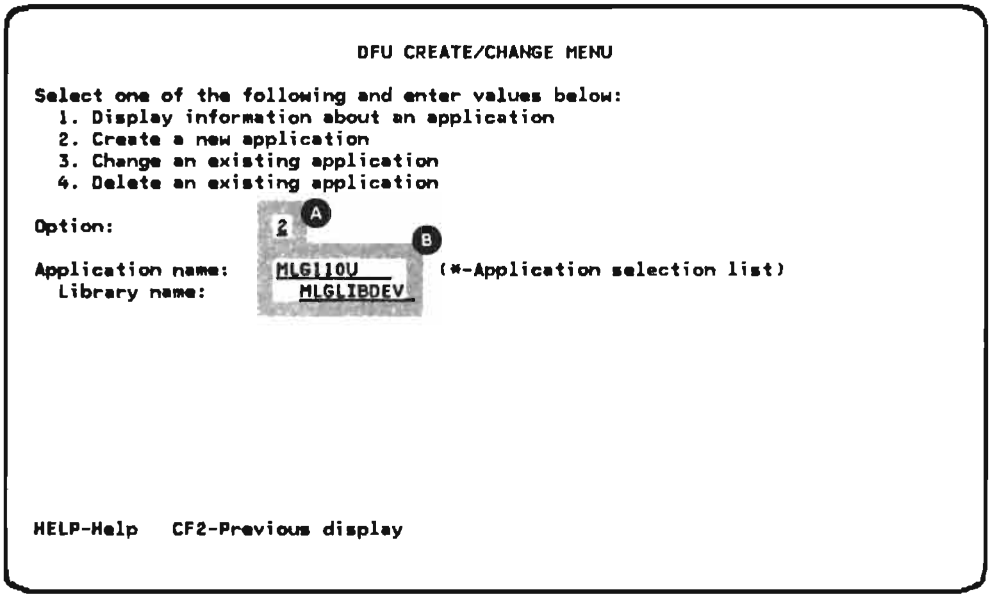
```
                         PROGRAMMER MENU
Select one of the following:
  1. Design/execute DFU app     (app), ,(options)
  2. Design/execute query app   (app), ,(options)
  3. Create object              object name, type, pgm for CMD, (text)
  4. Call program               program name
  5. Execute command            command
  6. Submit job                 (job name), (command)
  7. Display submitted jobs
  8. Edit source                (srcmbr), (type), (text)
  9. Design display format      (srcmbr)
 90. Sign off                   (*NOLIST *LIST)     .

Types:  BSCF, CBL, CL, CLP, CMD, CMNF, DSPF, LF, PF, PRTF, RPG, TXT

Option: 1_ Parm: MLG110U___ Type: ____ Parm 2: ___________________________
Command; _____________________________________________________________________
_____________________________________________________________________________
Text: ________________________________________________ Log requests: *YES
Src file: _________ Src lib: MLGLIBDEV_ Obj lib: MLGLIBDEV_ Jobd: MLGLIBDEV_
    CF3-Command entry  CF4-Prompt (3 & 5 only)   CF6-DSPMSG
```

You receive the DFU menu. Because you want to create an application, you select option 1 on this menu:



```
                              DFU MENU

Select one of the following:
  1. Create or change an application
  2. Execute an application
  3. Manage existing application

Option:   1





Press HELP for instructions.  Press CF1 to exit.
```

When you press the Enter key, you receive the DFU create/change menu:

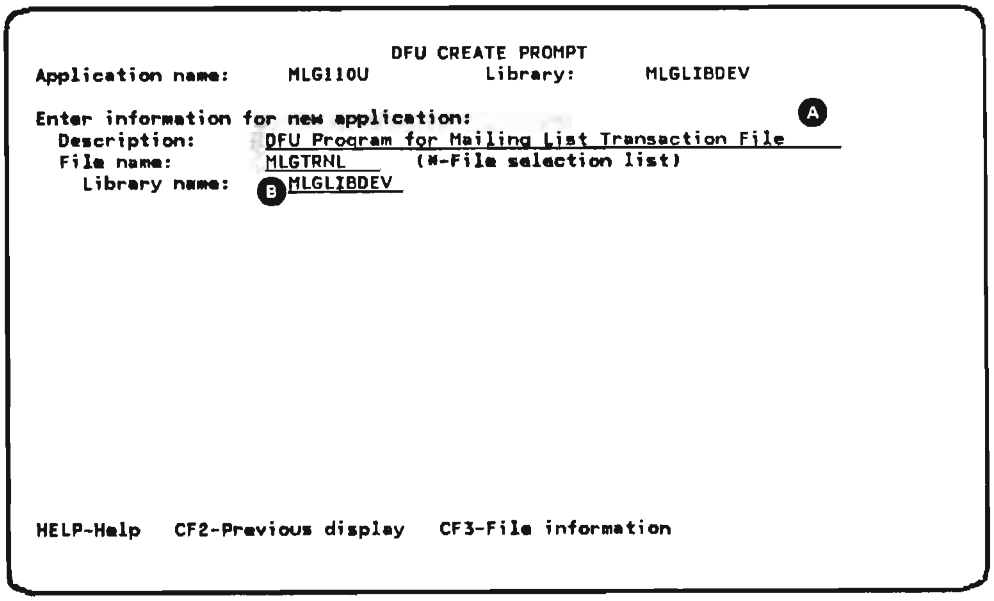


```
                       DFU CREATE/CHANGE MENU

Select one of the following and enter values below:
  1. Display information about an application
  2. Create a new application
  3. Change an existing application
  4. Delete an existing application

Option:             2

Application name:   MLG110U      (*-Application selection list)
  Library name:       MLGLIBDEV




HELP-Help   CF2-Previous display
```

A Select option 2.

B The application name and library name are already filled in because you specified an application name and object library on the programmer menu.

You then receive the DFU create prompt. You use this prompt to specify the text description of the application you are creating and the file you want the application to access:



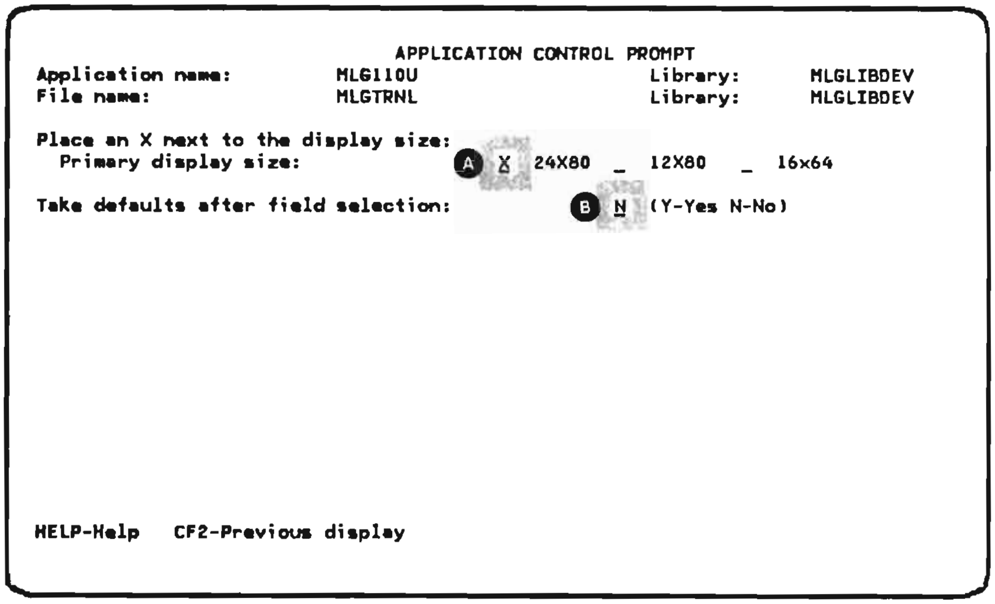
```
                              DFU CREATE PROMPT
Application name:     MLG110U          Library:      MLGLIBDEV

Enter information for new application:
  Description:      DFU Program for Mailing List Transaction File
  File name:        MLGTRNL       (*-File selection list)
    Library name:   MLGLIBDEV








HELP-Help   CF2-Previous display   CF3-File information
```

A Key in the text description of the application.

B Key in the name of the file to be accessed and the library where the file is located.

For the file name, you specify the name of the logical transaction file, MLGTRNL. The DFU application will interact with the records in the physical transaction file, MLGTRNP, through the access path defined by MLGTRNL (the procedures for creating MLGTRNP and MLGTRNL were discussed in Chapter 6).

When you press the Enter key after completing the create prompt, you receive the application control prompt. You only need to specify the size of the display screen you want the application to execute on:

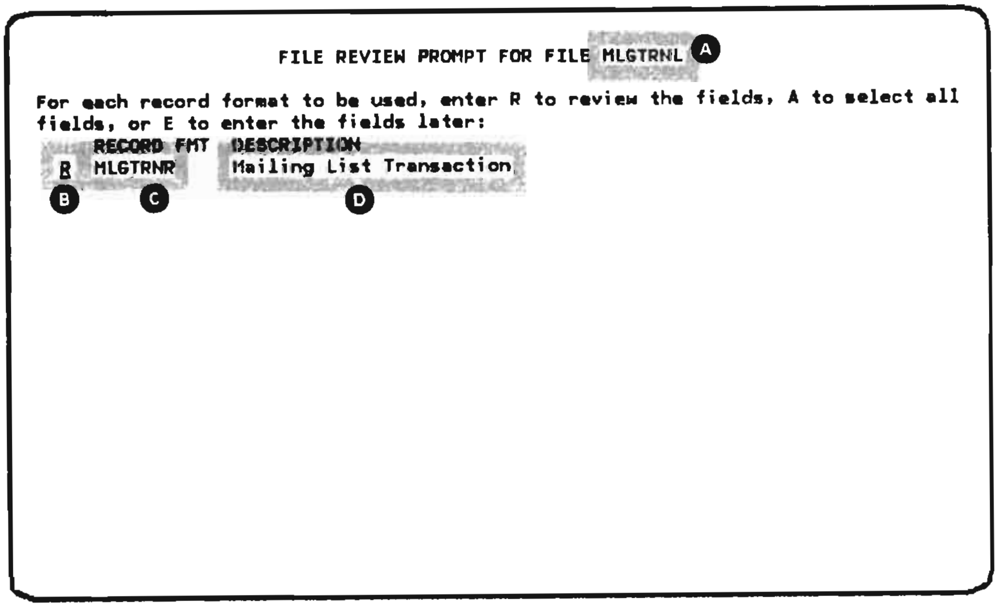


```
                            APPLICATION CONTROL PROMPT
Application name:          MLG110U                        Library:     MLGLIBDEV
File name:                 MLGTRNL                        Library:     MLGLIBDEV

Place an X next to the display size:
  Primary display size:              X  24X80  _  12X80  _  16x64

Take defaults after field selection:          N  (Y-Yes N-No)









HELP-Help   CF2-Previous display
```

A Key in X if not shown here.

B Do not change this default value because you will need to modify the definitions.

A file review prompt is displayed next:



```
                      FILE REVIEW PROMPT FOR FILE MLGTRNL

For each record format to be used, enter R to review the fields, A to select all
fields, or E to enter the fields later:
    RECORD FMT   DESCRIPTION
  R MLGTRNR      Mailing List Transaction
```

A You specified this file name on the DFU create prompt.

B Key in R to review all fields.

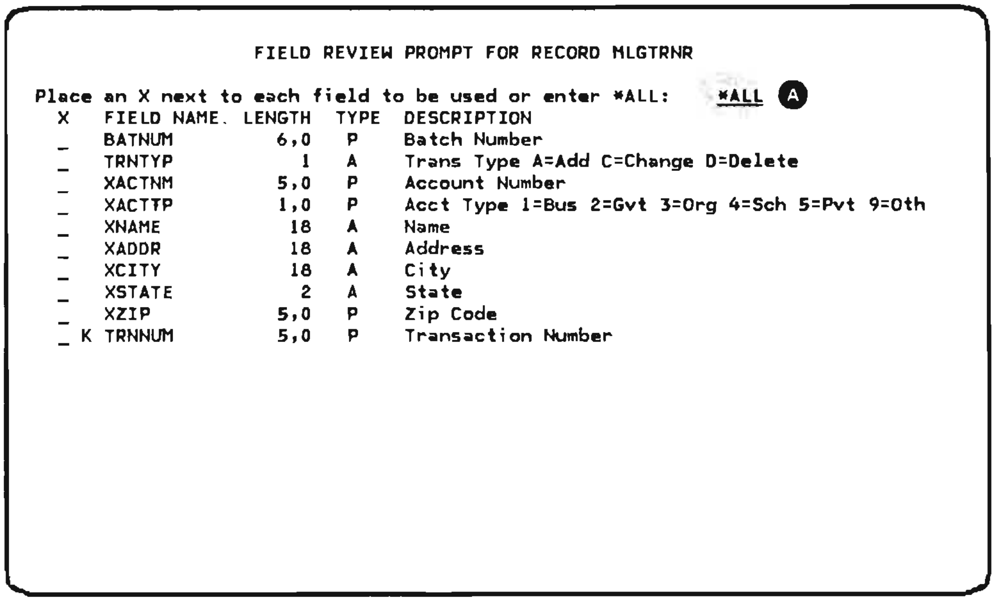
C This record format was entered as part of the DDS for MLGTRNL.

D This information was entered as part of the DDS TEXT keyword for the record format MLGTRNR.

To respond to this display, you can enter R, A, or E on the blank line preceding MLGTRNR. By entering A, you specify that you will be entering data into all fields and the field review prompt is skipped.

By entering E, you will bypass the field review prompt. These fields must be added later on the basic field definition prompt.

By entering R, which is the option used for this example, you can review the fields for records in MLGTRNL. (This information was entered in the DDS for MLGTRNL, as discussed in Chapter 6.) The following field review prompt is displayed:

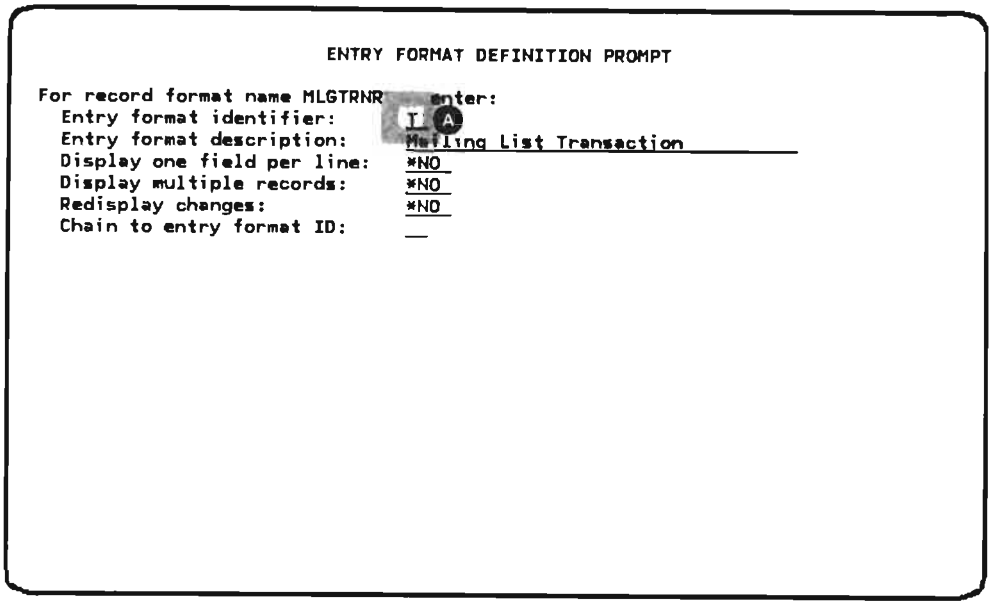


```
                        FIELD REVIEW PROMPT FOR RECORD MLGTRNR

Place an X next to each field to be used or enter *ALL:     *ALL  A
  X    FIELD NAME. LENGTH  TYPE  DESCRIPTION
  _    BATNUM        6,0    P    Batch Number
  _    TRNTYP          1    A    Trans Type A=Add C=Change D=Delete
  _    XACTNM        5,0    P    Account Number
  _    XACTTP        1,0    P    Acct Type 1=Bus 2=Gvt 3=Org 4=Sch 5=Pvt 9=Oth
  _    XNAME          18    A    Name
  _    XADDR          18    A    Address
  _    XCITY          18    A    City
  _    XSTATE          2    A    State
  _    XZIP          5,0    P    Zip Code
  _ K  TRNNUM        5,0    P    Transaction Number
```

A Key in *ALL to indicate data will be entered into all fields (which is done in this example).

You can choose the fields into which you will be entering data by placing an X next to those fields.

The next prompt is for entry format definition:

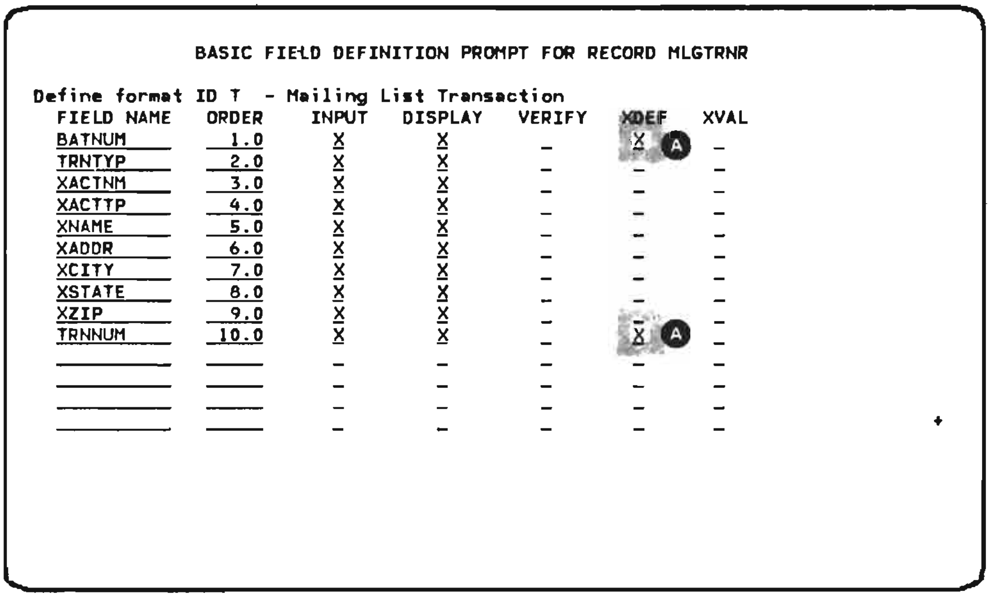


```
                    ENTRY FORMAT DEFINITION PROMPT

For record format name MLGTRNR   enter:
  Entry format identifier:       T
  Entry format description:      Mailing List Transaction
  Display one field per line:    *NO
  Display multiple records:      *NO
  Redisplay changes:             *NO
  Chain to entry format ID:      __
```

A Key in T (for transaction).

The default entry format description shown on the prompt was taken from the DDS for record format MLGTRNR.

The following basic field definition prompt lists the fields selected on the field review prompt and allows you to enter specifications for each field:



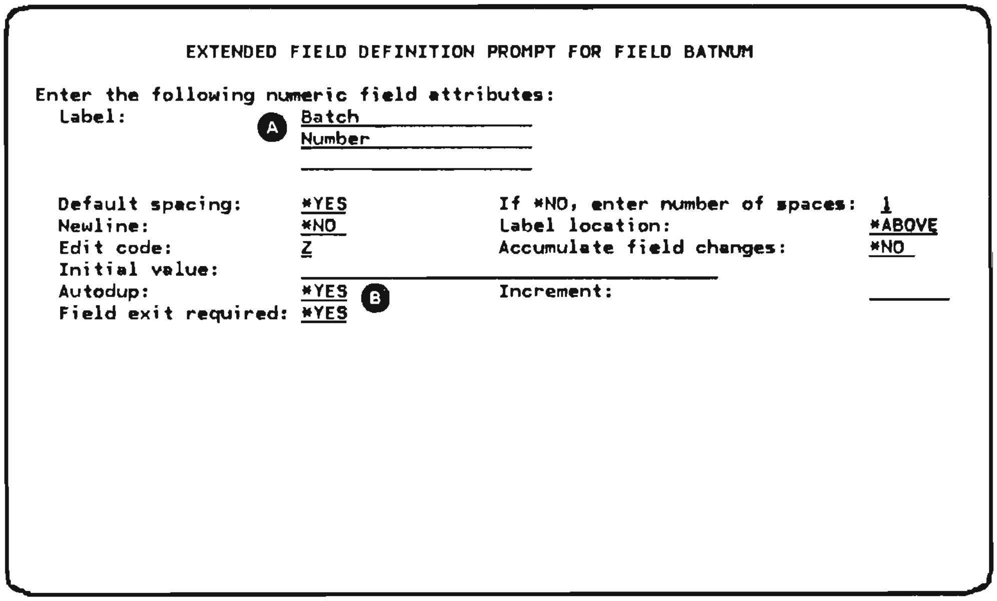
```
          BASIC FIELD DEFINITION PROMPT FOR RECORD MLGTRNR

Define format ID T  - Mailing List Transaction
  FIELD NAME   ORDER   INPUT   DISPLAY   VERIFY   XDEF   XVAL
  BATNUM         1.0     X        X        _        X      _
  TRNTYP         2.0     X        X        _        _      _
  XACTNM         3.0     X        X        _        _      _
  XACTTP         4.0     X        X        _        _      _
  XNAME          5.0     X        X        _        _      _
  XADDR          6.0     X        X        _        _      _
  XCITY          7.0     X        X        _        _      _
  XSTATE         8.0     X        X        _        _      _
  XZIP           9.0     X        X        _        _      _
  TRNNUM        10.0     X        X        _        X      _
  __________   _____     _        _        _        _      _
  __________   _____     _        _        _        _      _
  __________   _____     _        _        _        _      _
  __________   _____     _        _        _        _      _        +
```

A Key in X so that extended definitions can be specified for these fields.

For each field, Xs automatically appear in the INPUT and DISPLAY columns. In this example, the Xs should remain; however, if you need to remove an X in another application, enter a blank over that X.

The ORDER column defaults to the sequence of the record format. This is the order in which the fields will be displayed. The order can be changed, but the default order is used in this approach.

You next receive the extended field definition prompt for each field that had an X under the XDEF column. The extended field definition prompt for BATNUM is displayed first with the default entries:

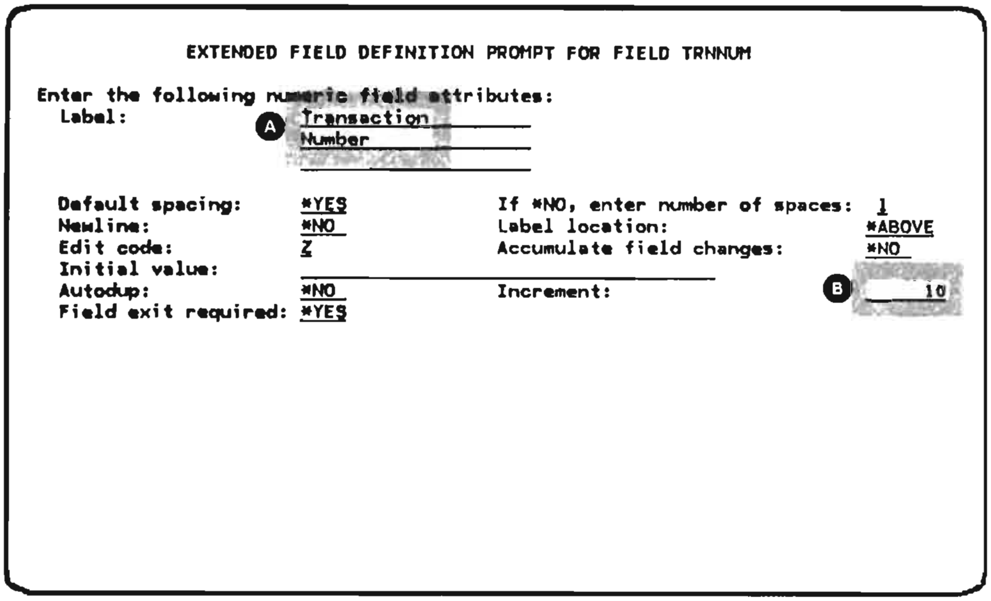


```
              EXTENDED FIELD DEFINITION PROMPT FOR FIELD BATNUM

Enter the following numeric field attributes:
  Label:                Batch
                        Number


  Default spacing:      *YES          If *NO, enter number of spaces:  1
  Newline:              *NO           Label location:                  *ABOVE
  Edit code:            Z             Accumulate field changes:        *NO
  Initial value:
  Autodup:              *YES          Increment:
  Field exit required:  *YES
```

A This information was entered as part of the DDS COLHDG keyword for this field in record format MLGTRNR.

B Key in *YES (*NO is the default) because the batch number for each record needs to be automatically duplicated in this example.

The extended field definition prompt for TRNNUM is displayed next:



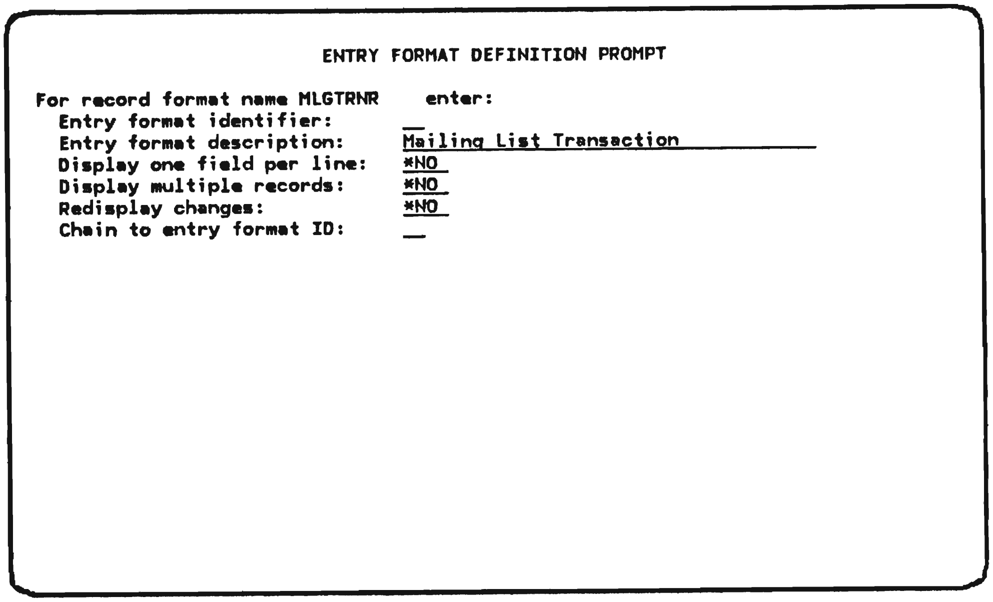
```
                    EXTENDED FIELD DEFINITION PROMPT FOR FIELD TRNNUM

 Enter the following numeric field attributes:
   Label:                   Transaction
                            Number

   Default spacing:      *YES           If *NO, enter number of spaces:  1
   Newline:              *NO            Label location:                  *ABOVE
   Edit code:            Z              Accumulate field changes:        *NO
   Initial value:
   Autodup:              *NO            Increment:                           10
   Field exit required:  *YES
```

A This information was entered as part of the DDS COLHDG keyword for this field in record format MLGTRNR.

B Key in 10 because the transaction number (TRNNUM) for each record needs to be automatically incremented in this example. This means that TRNNUM will equal 10 for the first record, 20 for the second record, and so on.

TRNNUM can be used as a key field to access existing transactions for modification or display; however, this use of a DFU program is not discussed in this publication. DFU requires any file it uses to have keyed access paths.

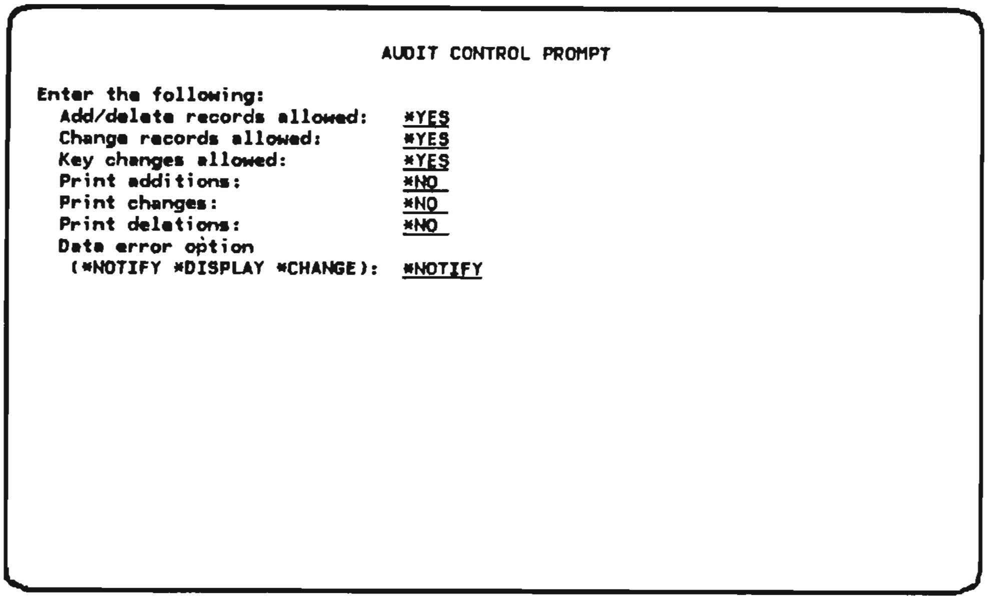
The entry format definition prompt is displayed again so that you can define another format:



```
                        ENTRY FORMAT DEFINITION PROMPT

For record format name MLGTRNR    enter:
  Entry format identifier:       _
  Entry format description:      Mailing List Transaction
  Display one field per line:    *NO
  Display multiple records:      *NO
  Redisplay changes:             *NO
  Chain to entry format ID:      _
```

Because no additional entry format needs to be defined, press the Enter key (without making any entries) to continue the prompting sequence.

The last prompt used to create the DFU definition is the audit control prompt:



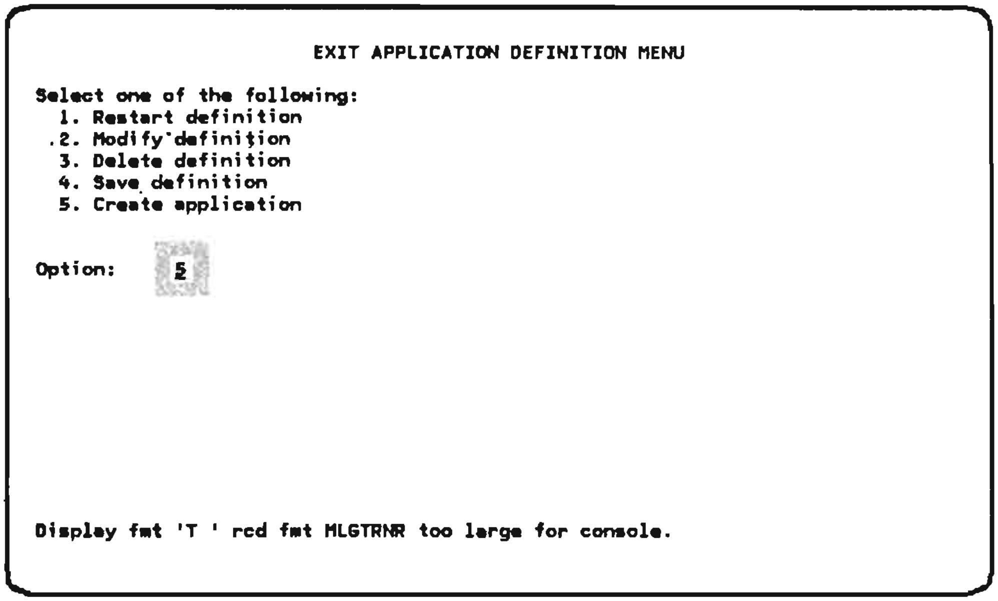
```
                                AUDIT CONTROL PROMPT

Enter the following:
  Add/delete records allowed:   *YES
  Change records allowed:       *YES
  Key changes allowed:          *YES
  Print additions:              *NO
  Print changes:                *NO
  Print deletions:              *NO
  Data error option
   (*NOTIFY *DISPLAY *CHANGE):  *NOTIFY
```

This display allows you to specify the following:

- Whether records in the file can be added/deleted, be changed, or have their key fields changed.
- Whether a printed report will contain additions, changes, and/or deletions.
- Whether the display that notifies you of errors during the creation process (if any errors occur) allows you to:
  - Only bypass or delete the record in error (*NOTIFY).
  - Also display or print information about the record in error (*DISPLAY).
  - Display and bypass or delete the record in error, or correct the error before the creation process continues (*CHANGE).

The defaults that automatically appear are used, and no additional information needs to be entered.

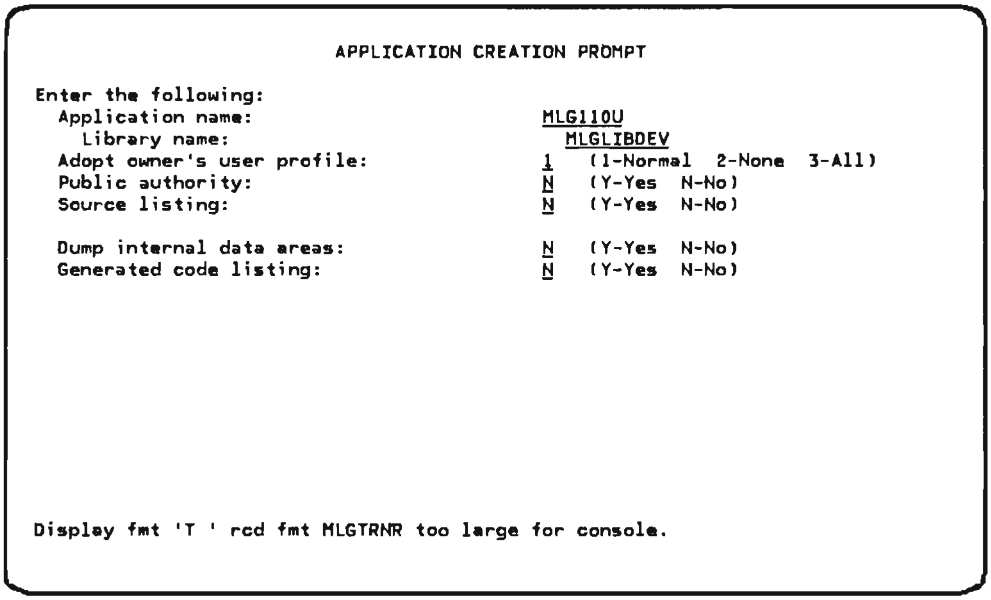
The definition of the MLG110U application is now complete, and you receive the exit application definition menu. You select option 5 to indicate that you want to create the application:



```
                          EXIT APPLICATION DEFINITION MENU

Select one of the following:
  1. Restart definition
  2. Modify definition
  3. Delete definition
  4. Save definition
  5. Create application

Option:     5







Display fmt 'T ' rcd fmt MLGTRNR too large for console.
```

The definition that was entered becomes a source member in the source file QUDSSRC. The intermediate output of DFU is a *utility definition specification* (UDS), which is a group of source statements from which IDU applications are created. The statements are stored in the source file QUDSSRC and allow you to modify an existing DFU application. The name of the source member is MLG110U, which becomes the name of the DFU application when the application is created.

Because you selected option 5 on the exit application definition menu, you receive an application creation prompt:



```
                          APPLICATION CREATION PROMPT

 Enter the following:
   Application name:                      MLG110U
     Library name:                          MLGLIBDEV
   Adopt owner's user profile:            1   (1-Normal  2-None  3-All)
   Public authority:                      N   (Y-Yes  N-No)
   Source listing:                        N   (Y-Yes  N-No)

   Dump internal data areas:              N   (Y-Yes  N-No)
   Generated code listing:                N   (Y-Yes  N-No)









 Display fmt 'T ' rcd fmt MLGTRNR too large for console.
```

To create the MLG110U application, you need only accept the defaults shown and press the Enter key. When the create function is complete, the user library MLGLIBDEV contains a DFU application named MLG110U as well as a DFU display file named MLG110U.

As soon as the application has been created, you receive a menu for executing it. In this approach, you are creating the application for later use by work station users and do not want to enter transactions; therefore, you press the CF1 key to exit DFU. The DFU menu is shown again. When you press the CF1 key, the programmer menu reappears.

Executing the DFU Application

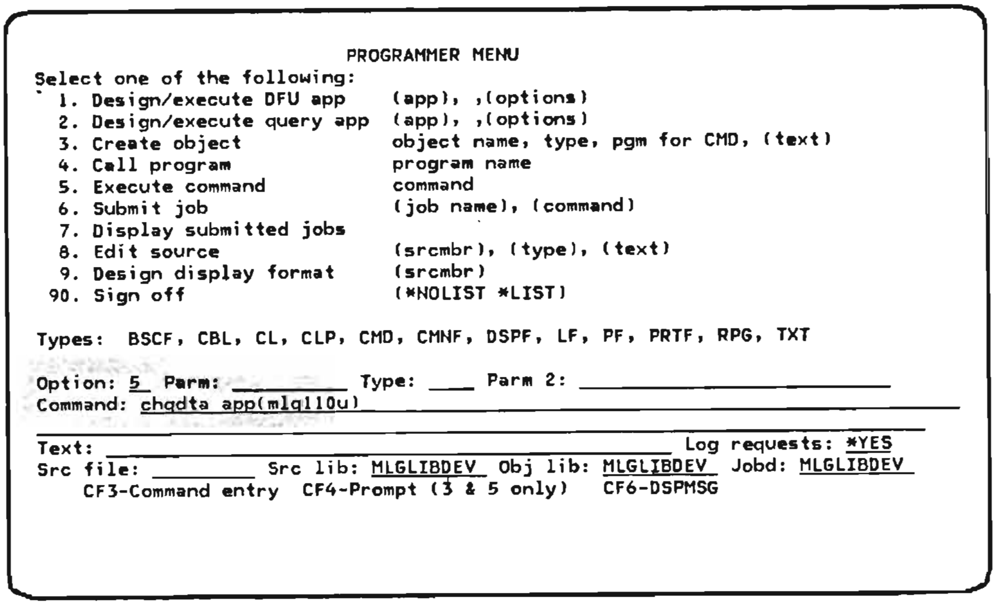
Once the DFU application MLG110U has been created, work station users can enter transactions into the transaction file by executing the application. You can execute a DFU application by:

- Selecting option 1 on the programmer menu, as you did to create the MLG110U application, and then selecting option 2 on the DFU menu when it is displayed.

- Enter the Change Data (CHGDTA) command.

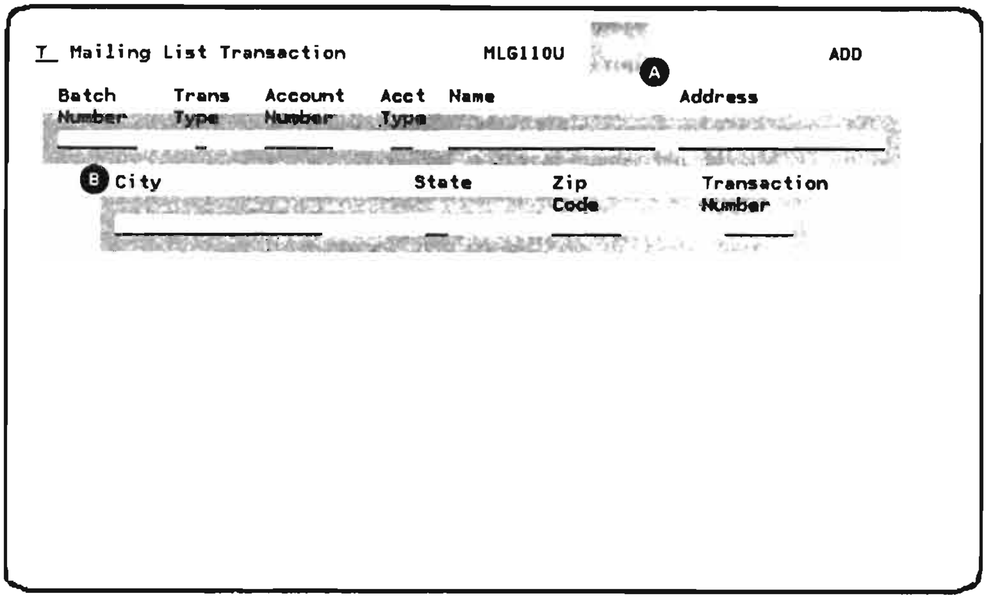
There are several ways that the CHGDTA command might be entered by the work station user. A method that uses user-created menus is described in Appendix B.

Assume at this point that you enter the CHGDTA command through the programmer menu to test the DFU application MLG110U. The command is entered on the programmer menu as follows:



```
                              PROGRAMMER MENU
Select one of the following:
  1. Design/execute DFU app     (app), ,(options)
  2. Design/execute query app   (app), ,(options)
  3. Create object              object name, type, pgm for CMD, (text)
  4. Call program               program name
  5. Execute command            command
  6. Submit job                 (job name), (command)
  7. Display submitted jobs
  8. Edit source                (srcmbr), (type), (text)
  9. Design display format      (srcmbr)
 90. Sign off                   (*NOLIST *LIST)

Types:  BSCF, CBL, CL, CLP, CMD, CMNF, DSPF, LF, PF, PRTF, RPG, TXT

Option: 5_ Parm: __________ Type: ____ Parm 2: ___________________________
Command: chgdta app(mlg110u)_______________________________________________
___________________________________________________________________________
Text: ______________________________________________ Log requests: *YES
Src file: _________ Src lib: MLGLIBDEV_ Obj lib: MLGLIBDEV_ Jobd: MLGLIBDEV_
    CF3-Command entry  CF4-Prompt (3 & 5 only)   CF6-DSPMSG
```

When MLG110U is executed, the following display appears if no records exist in the file:



```
T_ Mailing List Transaction              MLG110U                    ADD

 Batch    Trans   Account   Acct  Name                      Address
 Number   Type    Number    Type
 _______    _     _______    __   ______________________    ______________________

     City                           State      Zip          Transaction
                                               Code         Number
     ______________________           __       _______        _______
```

A The Auto Dup indicator is blank initially. An A is displayed when the Auto Dup key is pressed.

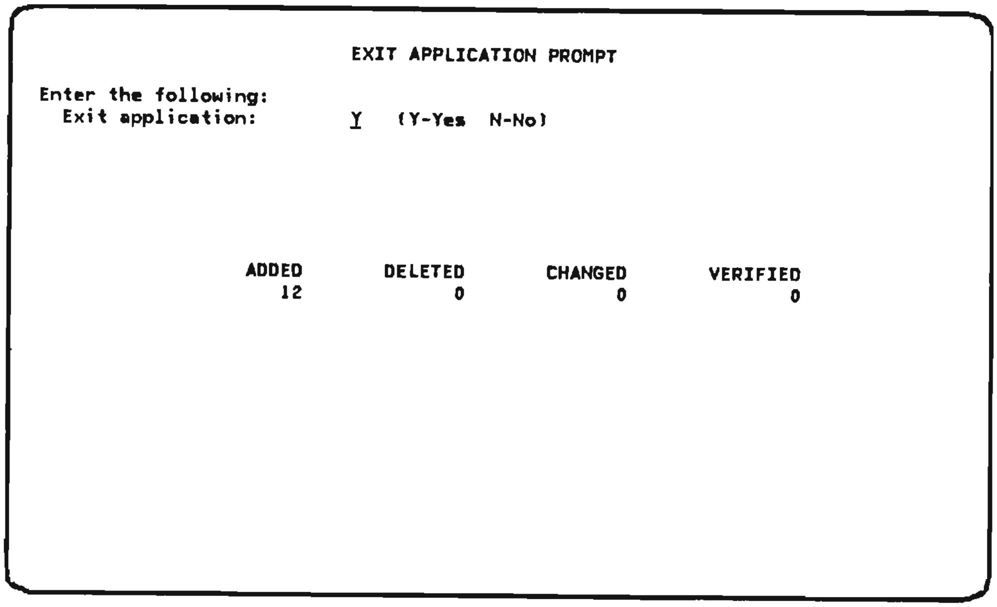
B You can key in transaction records on the blank lines.

The auto increment indicator must be on before a record is entered. The indicator is set on by pressing the Auto Increment key (CF7). The first record is assigned the transaction number of 10. Assign a unique batch number to the first record and, after this record is entered, press the Auto Dup key (CF6). Because Autodup was specified in the extended field definition for BATNUM, the batch number is automatically duplicated for each subsequent record until the Auto Dup key is pressed again.

A unique transaction number is automatically assigned to each transaction record. Because an increment of 10 was specified in the extended field definition for TRNNUM, transaction numbers for consecutive records are consecutive and in increments of 10.

If records exist in the file when MLG110U is executed, a display that allows you to enter the transaction number of the transaction record you want to change appears. If you want to add a transaction record, press CF9 and the display (previously shown) that allows you to enter new transaction records appears.

When all of your transactions are entered, press the Exit Application key (CF1). The exit application prompt is displayed as follows with a summary of the types of records you entered:

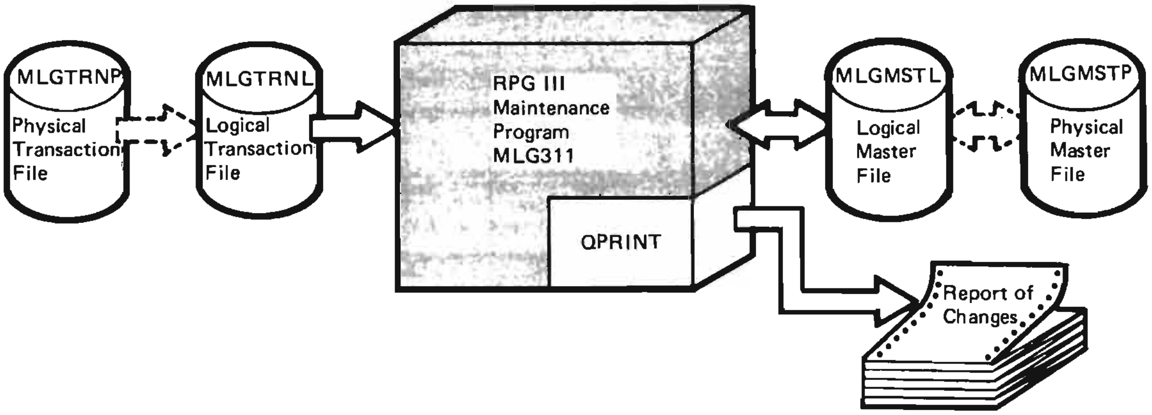


```
                         EXIT APPLICATION PROMPT

 Enter the following:
   Exit application:       Y   (Y-Yes  N-No)



                 ADDED       DELETED       CHANGED       VERIFIED
                    12             0             0              0
```

To end the job, press the Enter key. The programmer menu is then displayed.

APPLYING TRANSACTIONS TO THE MASTER FILE

Transactions are applied in a batch manner to the master file by using an RPG III maintenance program. In Chapter 4, the maintenance program, MLG310, used program-described data. In this approach, the RPG III maintenance program, MLG311, will be using externally described data in the data base file, MLGTRNL. MLG311 applies the transactions from MLGTRNL to the master mailing list file, MLGMSTL.



Creating the Maintenance Program

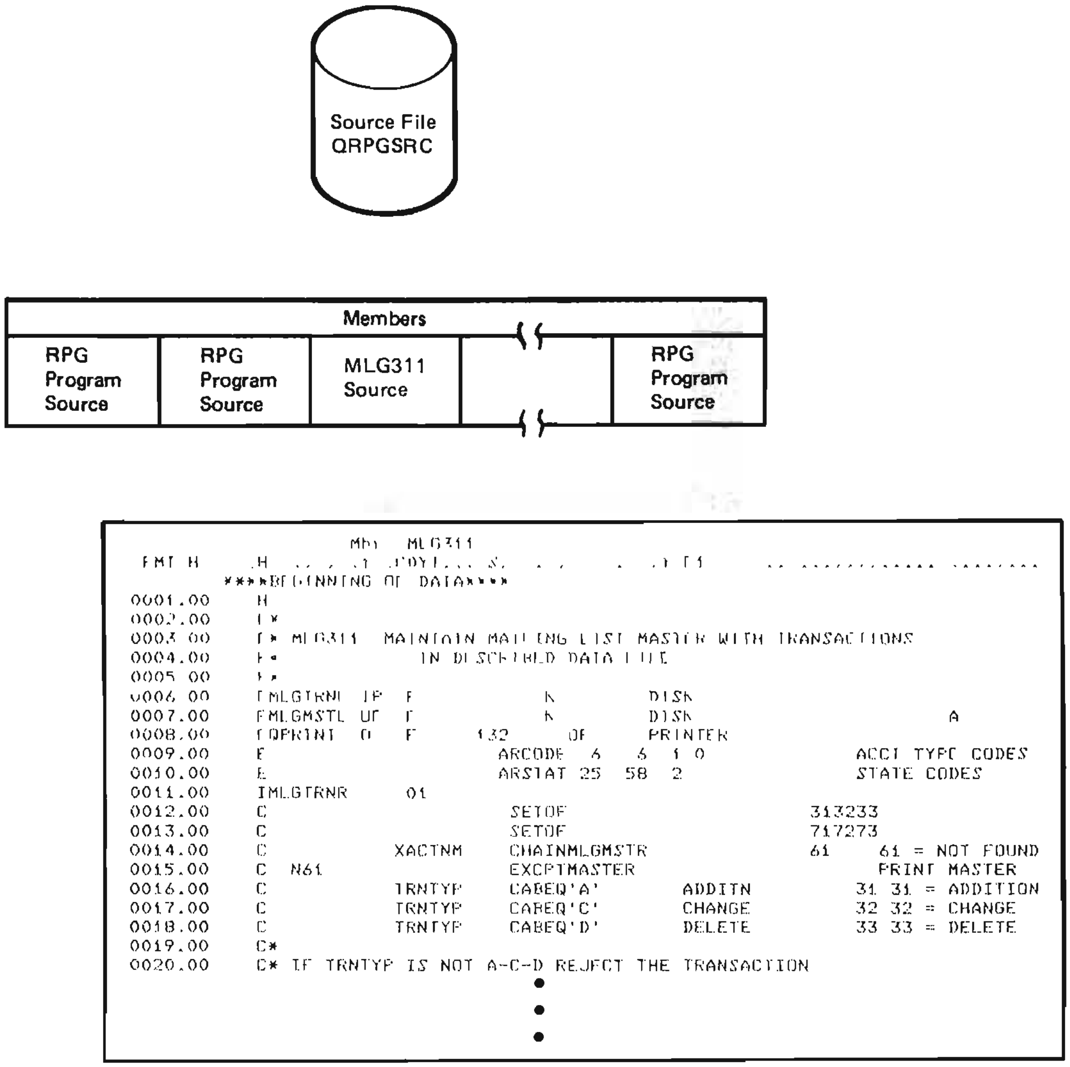
The steps to create MLG311 are outlined here:

- Invoke SEU from the programmer menu by selecting option 8 and specifying MLG311 for srcmbr, RPG for type, and MLGLIBDEV for the source library (see the discussion of the *Source Entry Utility* in Chapter 5).

- Enter the source statements.

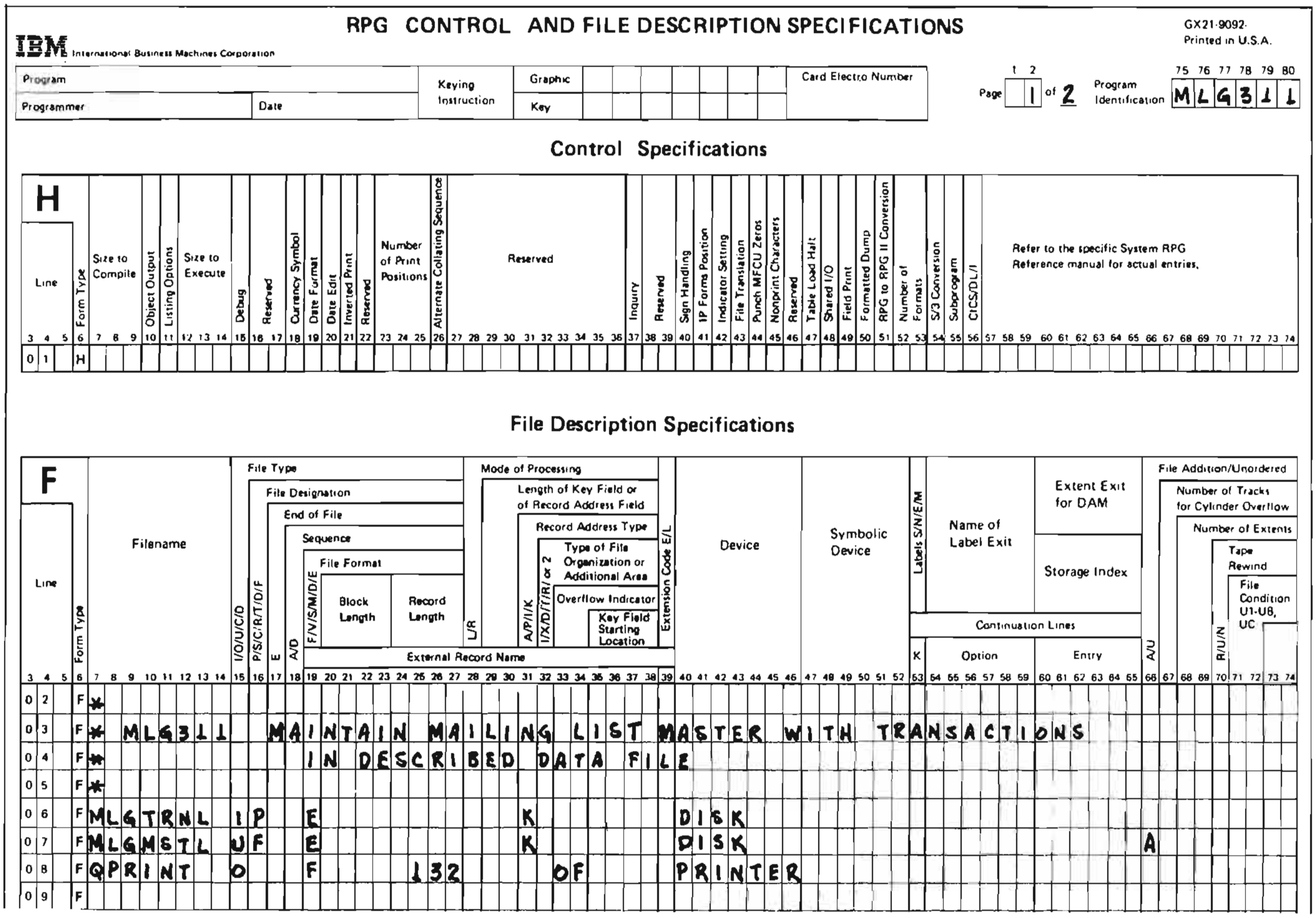
  The source for MLG311 is placed in QRPGSRC using SEU, as shown in Figure 7-1. The source is the same as that for MLG310 except for the following:
  - Substitute the RPG Control and File Description Specifications in Figure 7-2 for those in Figure 4-3.
  - Substitute the RPG Input Specifications in Figure 7-3 for those in Figure 4-5.

- Select option 3 on the programmer menu to create the object (RPG III maintence program) named MLG311.



```
                      Mbr   MLG311
 FMT H     .H  .. . .1 .PDYT... S.  . .      .  .1 F1      .. ............ ........
         ****BEGINNING OF DATA****
0001.00      H
0002.00      F*
0003.00      F* MLG311   MAINTAIN MAILING LIST MASTER WITH TRANSACTIONS
0004.00      F*                 IN DESCRIBED DATA FILE
0005.00      F*
0006.00      FMLGTRNL  IP  F              K          DISK
0007.00      FMLGMSTL  UF  F              K          DISK                   A
0008.00      FQPRINT   O   F      132      OF       PRINTER
0009.00      E                      ARCODE  6   6  1 0              ACCT TYPE CODES
0010.00      E                      ARSTAT 25  58  2                STATE CODES
0011.00      IMLGTRNR      01
0012.00      C                       SETOF                      313233
0013.00      C                       SETOF                      717273
0014.00      C             XACTNM    CHAINMLGMSTR               61      61 = NOT FOUND
0015.00      C  N61                  EXCPTMASTER                          PRINT MASTER
0016.00      C             TRNTYP    CABEQ'A'        ADDITN          31 31 = ADDITION
0017.00      C             TRNTYP    CABEQ'C'        CHANGE          32 32 = CHANGE
0018.00      C             TRNTYP    CABEQ'D'        DELETE          33 33 = DELETE
0019.00      C*
0020.00      C* IF TRNTYP IS NOT A-C-D REJECT THE TRANSACTION
                                ●
                                ●
                                ●
```

Figure 7-1. The MLG311 Maintenance Program is a Member of QRPGSRC



IBM International Business Machines Corporation

RPG CONTROL AND FILE DESCRIPTION SPECIFICATIONS

GX21-9092-
Printed in U.S.A.

Program | Programmer | Date | Keying Instruction | Graphic | Key | Card Electro Number

Page 1 of 2 Program Identification MLG311

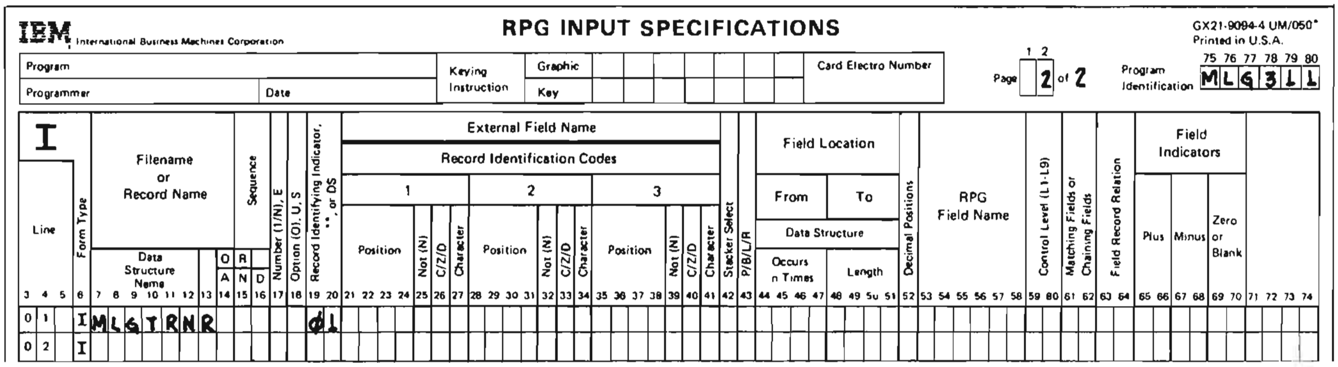
Control Specifications

```
01 H
```

File Description Specifications

```
02 F*
03 F* MLG311   MAINTAIN MAILING LIST MASTER WITH TRANSACTIONS
04 F*           IN DESCRIBED DATA FILE
05 F*
06 FMLGTRNL IP  E           K        DISK
07 FMLGMSTL UF  E           K        DISK                      A
08 FQPRINT  O   F     132      OF    PRINTER
09 F
```

Figure 7-2. Control and File Description Specifications for MLG311 Maintenance Program



IBM International Business Machines Corporation

RPG INPUT SPECIFICATIONS

GX21-9094-4 UM/050*
Printed in U.S.A.

Program | Programmer | Date | Keying Instruction | Graphic | Key | Card Electro Number

Page 2 of 2 Program Identification MLG311

```
01 IMLGTRNR        Ø1
02 I
```

Figure 7-3. Input Specifications for MLG311 Maintenance Program

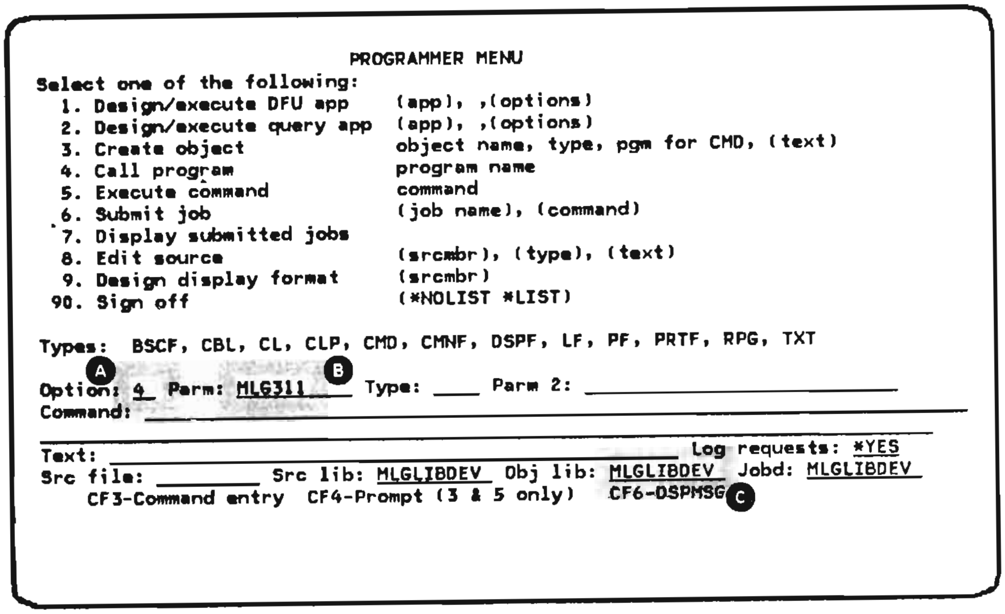
Executing the Maintenance Program

When MLG311 is executed:

- The transactions from MLGTRNL are applied to the master file MLGMSTL.

- A report that lists the changes made to the master file MLGMSTL is printed.

A work station user executes the MLG311 maintenance program by calling it. For example, you could call the program by using option 4 on the programmer menu:



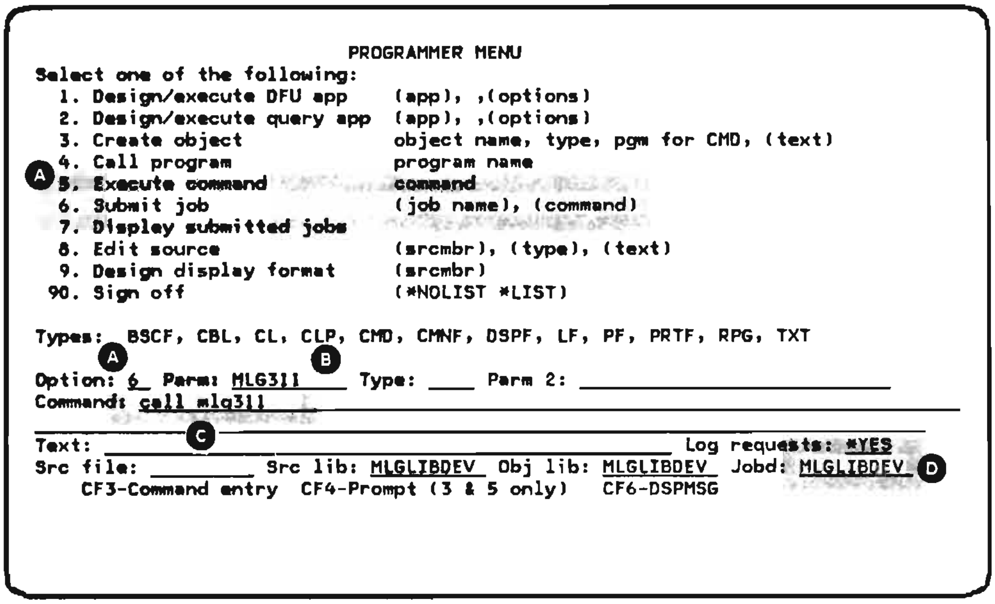
```
                              PROGRAMMER MENU
Select one of the following:
  1. Design/execute DFU app      (app), ,(options)
  2. Design/execute query app    (app), ,(options)
  3. Create object               object name, type, pgm for CMD, (text)
  4. Call program                program name
  5. Execute command             command
  6. Submit job                  (job name), (command)
  7. Display submitted jobs
  8. Edit source                 (srcmbr), (type), (text)
  9. Design display format       (srcmbr)
 90. Sign off                    (*NOLIST *LIST)

Types:  BSCF, CBL, CL, CLP, CMD, CMNF, DSPF, LF, PF, PRTF, RPG, TXT

Option: 4  Parm: MLG311       Type: ____  Parm 2: ________________________
Command: _______________________________________________________________
________________________________________________________________________
Text: ______________________________________________ Log requests: *YES
Src file: _________  Src lib: MLGLIBDEV  Obj lib: MLGLIBDEV  Jobd: MLGLIBDEV
    CF3-Command entry  CF4-Prompt (3 & 5 only)  CF6-DSPMSG
```

A Option 4 calls a program.

B The name of the program to be called is specified here.

C If a library name is specified here, the system searches only the specified library for the program instead of searching the libraries in the library list.

The disadvantage of using this method is that you cannot use the work station again until the program is completed. However, you can go on to other tasks at your work station if you include the CALL command in a batch job that you submit from your work station. You can submit a batch job from the programmer menu to call the MLG311 program:



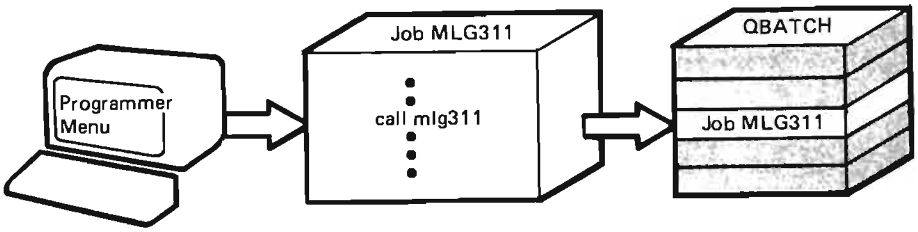
```
                                 PROGRAMMER MENU
Select one of the following:
  1. Design/execute DFU app    (app), ,(options)
  2. Design/execute query app  (app), ,(options)
  3. Create object             object name, type, pgm for CMD, (text)
  4. Call program              program name
  5. Execute command           command
  6. Submit job                (job name), (command)
  7. Display submitted jobs
  8. Edit source               (srcmbr), (type), (text)
  9. Design display format     (srcmbr)
 90. Sign off                  (*NOLIST *LIST)

Types:  BSCF, CBL, CL, CLP, CMD, CMNF, DSPF, LF, PF, PRTF, RPG, TXT

Option: 6_ Parm: MLG311____ Type: ____ Parm 2: __________________________
Command: call mlg311_______________________________________________________
___________________________________________________________________________
Text: ___________________________________________ Log requests: *YES
Src file: ________ Src lib: MLGLIBDEV  Obj lib: MLGLIBDEV  Jobd: MLGLIBDEV
    CF3-Command entry  CF4-Prompt (3 & 5 only)  CF6-DSPMSG
```

A Option 6 submits the job for batch processing. The job is placed on the QBATCH job queue.

B You specify the job name here. If you leave this field blank, QBATCH is used for the job name.

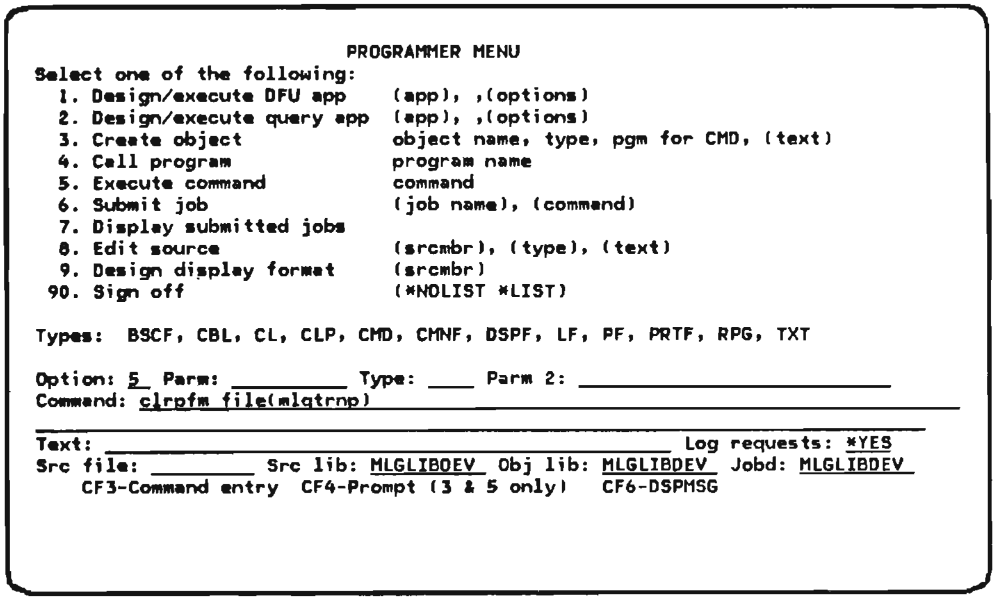
C This CALL command is submitted as part of the job and is executed when the job is selected from the job queue.

D The submitted job uses the job description MLGLIBDEV as specified here. Because you did not specify a job queue when you created the MLGLIBDEV job description (Chapter 6), the default job queue, QBATCH, is used.



When the *job* MLG311 is selected from the QBATCH job queue, the CALL command is executed and the MLG311 program begins applying the transactions to the master file. After MLG311 has updated the master file, a listing of the updates is printed by the spooling writer.

The data in the transaction file can be removed when it is no longer needed. MLGTRNP is cleared by entering the Clear Physical File Member (CLRPFM) command on the programmer menu as follows:



```
                              PROGRAMMER MENU
Select one of the following:
  1. Design/execute DFU app    (app), ,(options)
  2. Design/execute query app  (app), ,(options)
  3. Create object             object name, type, pgm for CMD, (text)
  4. Call program              program name
  5. Execute command           command
  6. Submit job                (job name), (command)
  7. Display submitted jobs
  8. Edit source               (srcmbr), (type), (text)
  9. Design display format     (srcmbr)
 90. Sign off                  (*NOLIST *LIST)

Types:  BSCF, CBL, CL, CLP, CMD, CMNF, DSPF, LF, PF, PRTF, RPG, TXT

Option: 5_ Parm: __________ Type: ____ Parm 2: ___________________________
Command: clrpfm file(mlgtrnp)_____________________________________________
__________________________________________________________________________
Text: ________________________________________________ Log requests: *YES
Src file: _________ Src lib: MLGLIBOEV  Obj lib: MLGLIBDEV  Jobd: MLGLIBDEV
    CF3-Command entry  CF4-Prompt (3 & 5 only)   CF6-DSPMSG
```

After MLGTRNP is cleared, it is ready to have a new group of transactions entered into it.

RECOVERY CONSIDERATIONS

If incorrect data has been entered for a transaction, the entire transaction should be reentered with the correct data.

If there is a system failure while the maintenance program MLG311 is executing, the current batch of transactions should be reprocessed in most situations. The maintenance program rejects invalid transactions, such as a request to add a record with an account number that already exist in the master file. If a system failure occurs when you are not executing the maintenance program, no recovery is normally needed.

If there is a system failure while the transactions are being entered through the DFU application MLG110U, the transaction file (MLGTRNP) will normally be readable after another initial microprogram load (IMPL). DFU allows the work station user to determine the last transaction in the file and then continue from that point.

If neither the DFU application nor the maintenance program is executing when a system failure occurs, normally there are no recovery considerations for the application.

To allow for recovery from damaged objects (assuming you are maintaining the master file with one or more batches per day), the master file could be saved every 2 to 3 days. If the damaged master file is not usable, the backup can be restored and the saved transactions can be reprocessed. You should keep both the current backup and at least one earlier level of backup. The data in the transaction file MLGTRNP must be considered for backup prior to entering another batch of transactions. Two approaches could be:

- Save each batch of transactions after it is processed.

- Copy the processed transactions to a common file for all of the transaction batches and save that file at the end of the day. This approach is probably more desirable because it minimizes what the system operator must do during the day; however, there is an exposure that the transactions must be rekeyed if the entire system becomes unusable.

Neither of these approaches is described in this publication, but it is assumed that one of them has been implemented.

PROGRAMMING CONSIDERATIONS

Part of the approach in this chapter described submitting the batch maintenance job, backing up the transactions, and clearing the transaction file. The work station user could have initiated these tasks by submitting a CL program through a user-written menu that would execute the program as a batch job. User-written menus are discussed in Chapter 8, and submitting jobs from a menu is described in Appendix B.

Providing a way for the work station user to submit a CL program that executes a series of standard commands is usually a good technique because:

- The work station user knows when the batch is complete.
- The system operator does not have to be involved (assuming that the backup is done by copying the transactions to a common file during the day).

Chapter 8. Interactive Input and Interactive Maintenance

OVERVIEW

In Chapter 7, transactions were entered interactively into a transaction file from a work station, and this batch of transactions was then applied to the master file by an RPG III maintenance program. The approach in this chapter allows work station users to interactively update the master file itself.

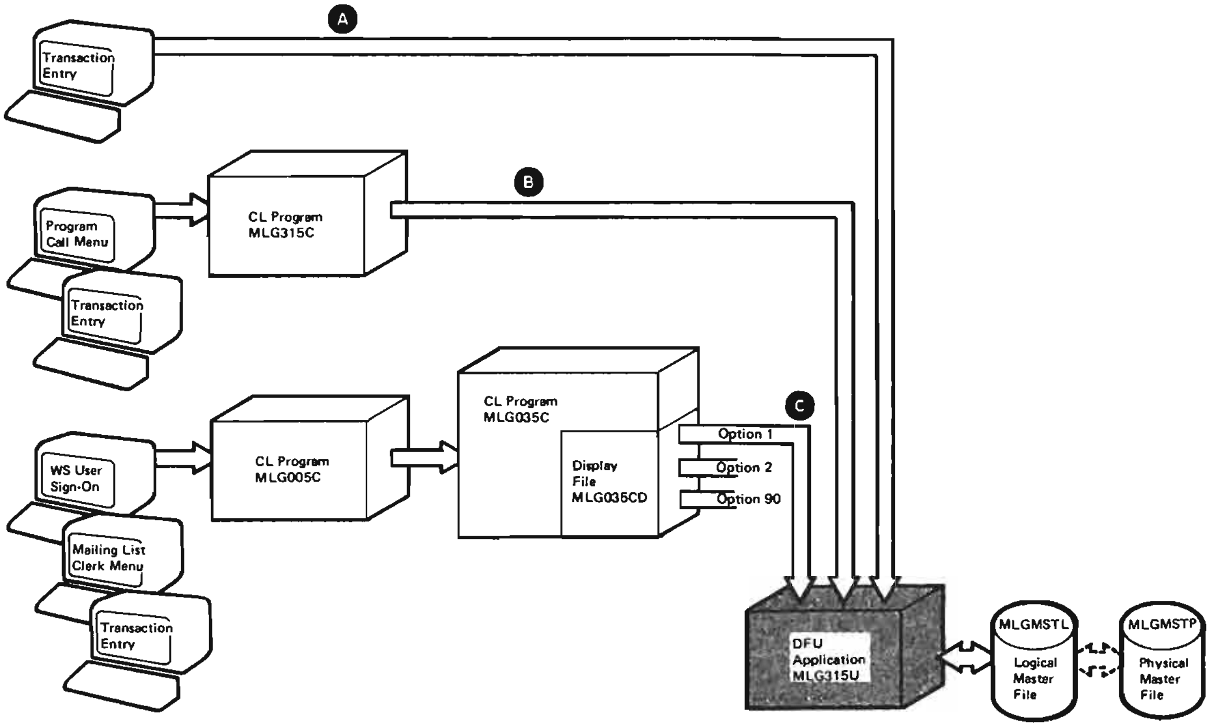
A DFU application, MLG315U, provides the displays on which the work station users enter transaction records. The application applies the entered records to the physical master file, MLGMSTP, through the access path provided by the logical master file, MLGMSTL.

Each work station user who invokes the MLG315U application sees the same set of transaction-entry displays. However, different work station users could invoke the application in different ways. Three ways of invoking the application are described in this chapter (see illustration on the next page):

Ⓐ The work station user invokes MLG315U directly from his basic working display, such as by entering the Change Data (CHGDTA) command on the programmer menu.

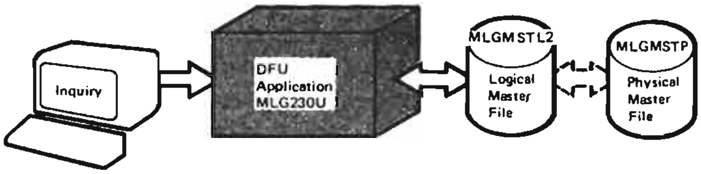
Ⓑ The work station user calls a CL program, MLG315C, from the IBM-supplied program call menu, and the MLG315C program invokes MLG315U.

Ⓒ The work station user signs on with a special user profile that defines the CL program, MLG005C, as the initial program. The MLG005C program establishes the proper application environment by replacing the library list and then calls a CL program, MLG035C, which provides an application-oriented menu. When the work station user selects an option on the specialized menu, the MLG315U application is invoked.

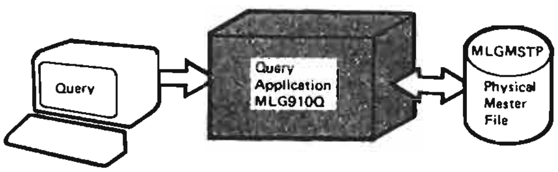


This chapter also discusses two IDU functions for making inquiries into the master file:

- Using the DFU application, MLG230U, to make inquiries by zip code into the master file, MLGMSTP, through a second access path defined by the logical file, MLGMSTL2.

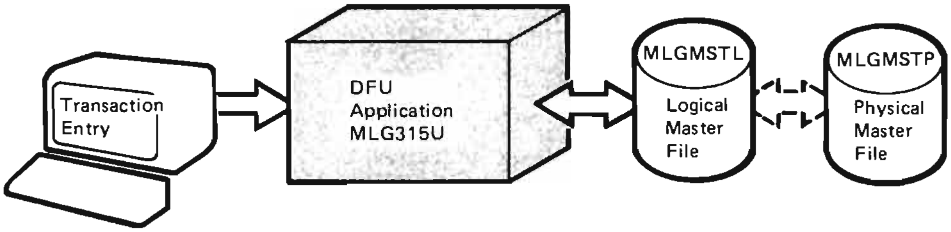


- Using the Query application, MLG910Q, to select and print records from the master file, MLGMSTP.



DFU APPLICATION FOR INTERACTIVE MAINTENANCE

The approach in Chapter 7 allowed work station users to interactively enter transaction records, but the actual maintenance of the master file (applying the transaction records to the master records) was done by an RPG III maintenance program. In the approach presented in this chapter, changes are entered directly into the master file by a DFU application named MLG315U.



This online maintenance has the following advantages:

- Most businesses that use online maintenance experience fewer clerical errors. The individuals making the changes tend to operate in a more thorough manner because they can readily see the impact of their work.

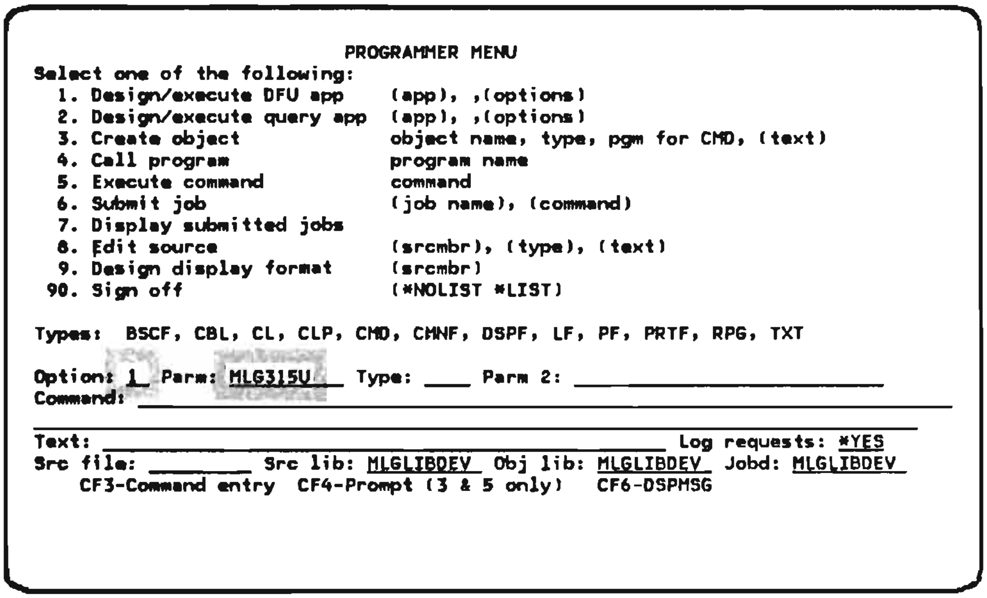
- A change to the master file is effective when it is entered and the changed record is available to other programs.

- The master file tends to be more accurate because an existing record is displayed before it is changed or deleted. Consequently, there is better validity checking, which can reduce the time required for correcting errors such as invalid codes and duplicate account numbers.

- The individual who knows a change that should be made can make the change himself. This approach would bypass the steps of transcribing the changes to a source document and then into a machine processable form.

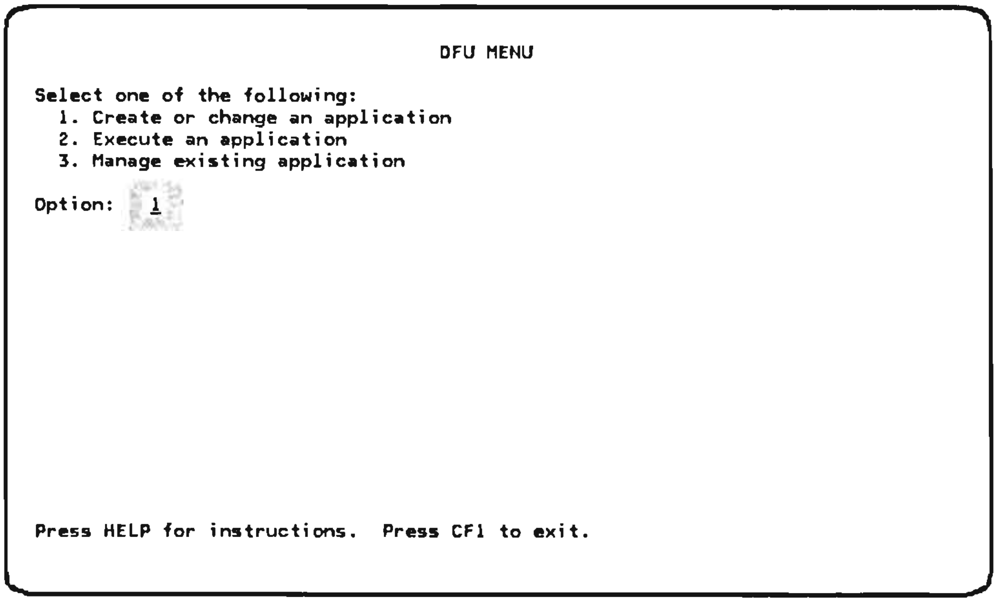
Creating the DFU Application

To create the DFU application MLG315U, select option 1 on the programmer menu as follows:



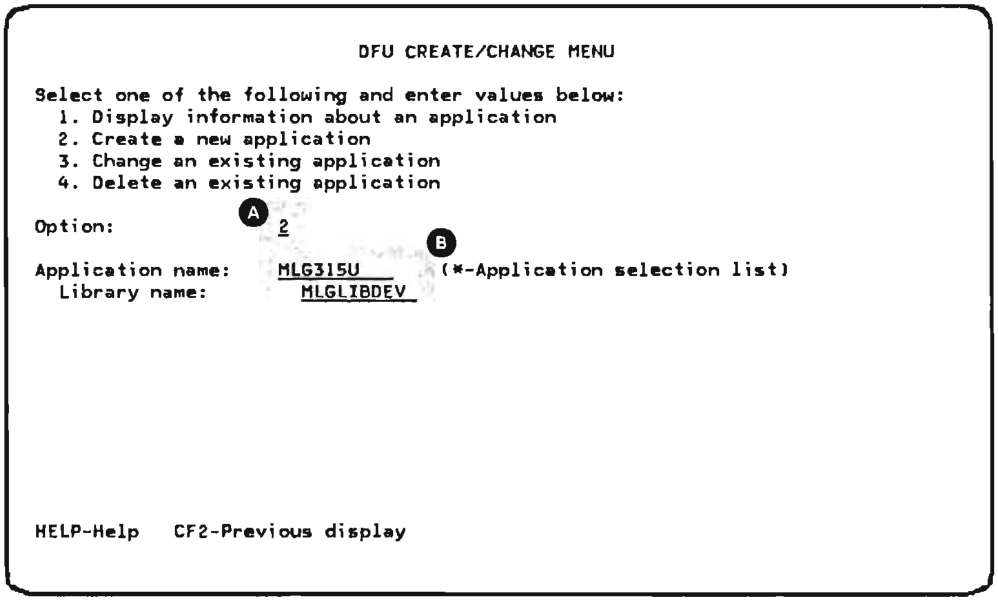
```
                            PROGRAMMER MENU
Select one of the following:
   1. Design/execute DFU app    (app), ,(options)
   2. Design/execute query app  (app), ,(options)
   3. Create object             object name, type, pgm for CMD, (text)
   4. Call program              program name
   5. Execute command           command
   6. Submit job                (job name), (command)
   7. Display submitted jobs
   8. Edit source               (srcmbr), (type), (text)
   9. Design display format     (srcmbr)
  90. Sign off                  (*NOLIST *LIST)

Types:  BSCF, CBL, CL, CLP, CMD, CMNF, DSPF, LF, PF, PRTF, RPG, TXT

Option: 1_ Parm: MLG315U___ Type: ____ Parm 2: __________________________
Command: _______________________________________________________________
________________________________________________________________________
Text: ________________________________________________ Log requests: *YES
Src file: _________ Src lib: MLGLIBDEV_ Obj lib: MLGLIBDEV_ Jobd: MLGLIBDEV_
    CF3-Command entry  CF4-Prompt (3 & 5 only)   CF6-DSPMSG
```

You receive a series of DFU displays on which you define the MLG315U application. The first display is the DFU menu. Select option 1 to create an application:



```
                              DFU MENU

Select one of the following:
  1. Create or change an application
  2. Execute an application
  3. Manage existing application

Option:   1










Press HELP for instructions.  Press CF1 to exit.
```

The DFU create/change menu appears next:

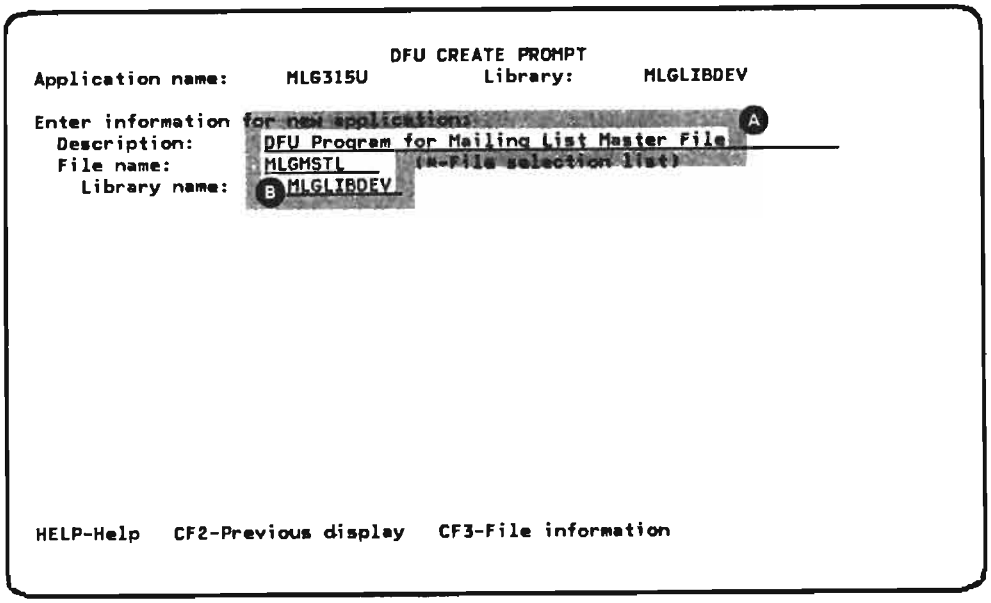


```
                        DFU CREATE/CHANGE MENU

Select one of the following and enter values below:
  1. Display information about an application
  2. Create a new application
  3. Change an existing application
  4. Delete an existing application

Option:                2

Application name:      MLG315U___     (*-Application selection list)
  Library name:          MLGLIBDEV_








HELP-Help   CF2-Previous display
```

A Select option 2.

B The application name and object library that you specified on the programmer menu are shown here.

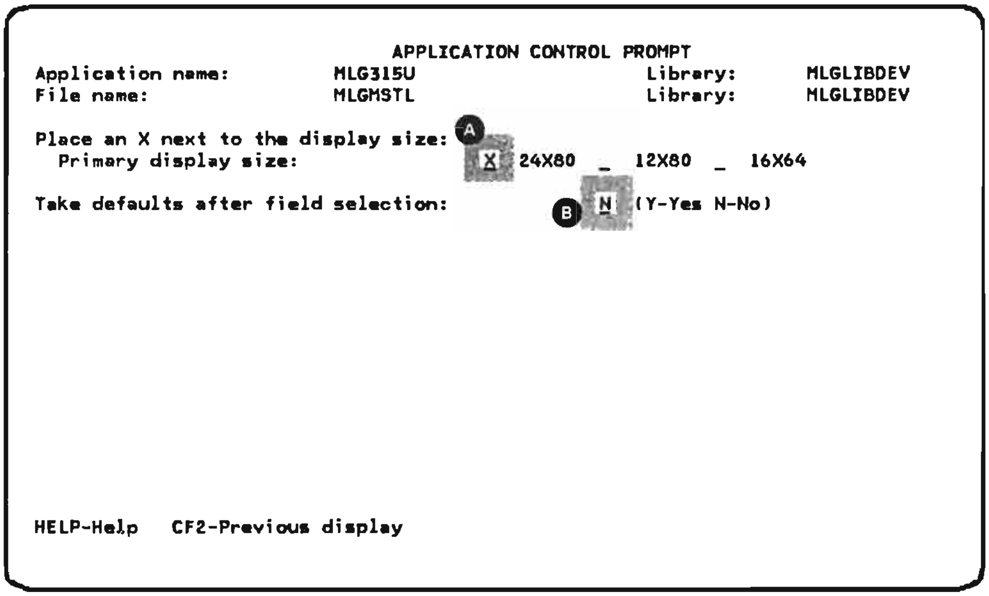
You then receive the DFU create prompt:



```
                              DFU CREATE PROMPT
Application name:     MLG315U          Library:      MLGLIBDEV

Enter information for new application:
  Description:       DFU Program for Mailing List Master File
  File name:         MLGMSTL     (*-File selection list)
    Library name:    MLGLIBDEV




HELP-Help    CF2-Previous display    CF3-File information
```

- A Key in the text description of the application.
- B Key in the name of the file to be accessed and the library where the file is located.

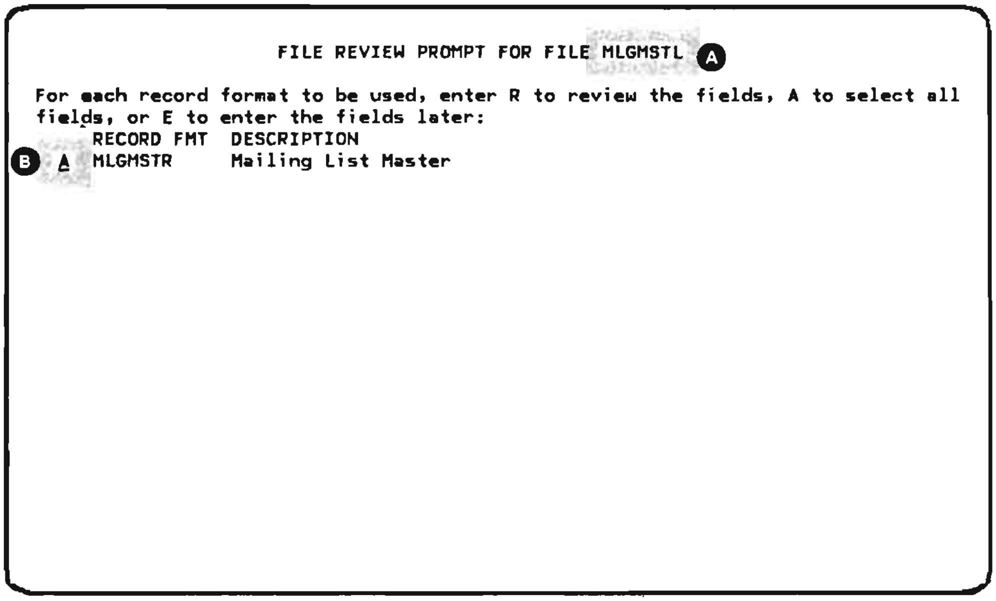
After completing the create prompt, you receive an application control prompt that shows the application name (MLG315U), file name (MLGMSTL), and their respective library names as you specified previously:



```
                              APPLICATION CONTROL PROMPT
Application name:         MLG315U                    Library:     MLGLIBDEV
File name:                MLGMSTL                    Library:     MLGLIBDEV

Place an X next to the display size:
  Primary display size:              X  24X80   _  12X80   _  16X64

Take defaults after field selection:     N  (Y-Yes N-No)




HELP-Help    CF2-Previous display
```

- A Key in X if not shown here.
- B Do not change this default value; you will be modifying the definitions.

A file review prompt is displayed next:

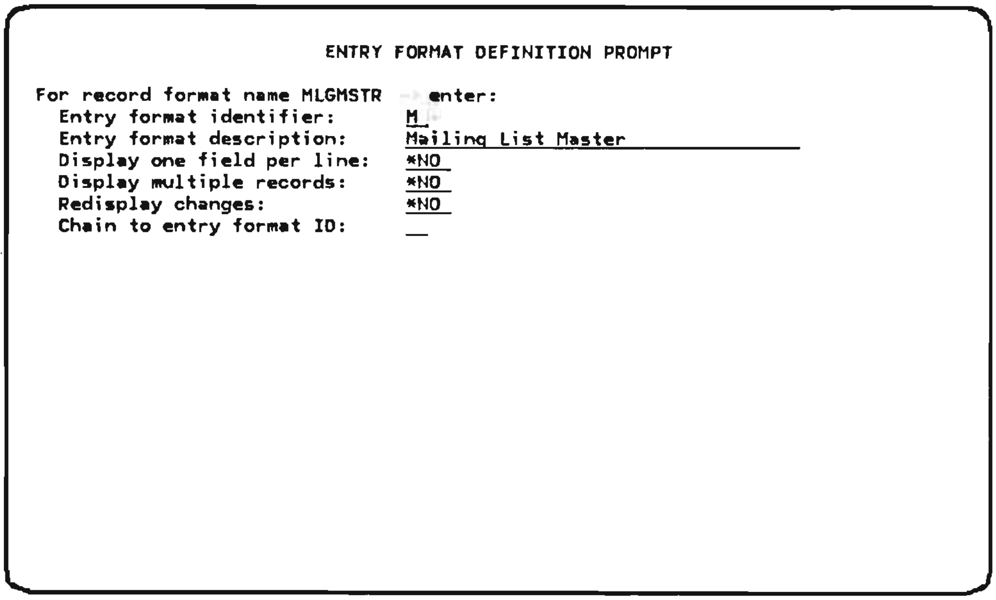


```
                   FILE REVIEW PROMPT FOR FILE MLGMSTL

For each record format to be used, enter R to review the fields, A to select all
fields, or E to enter the fields later:
    RECORD FMT  DESCRIPTION
  A MLGMSTR     Mailing List Master
```

A This is the file name you specified on the DFU create prompt.

B Key in A to specify that you will be entering data into all fields.

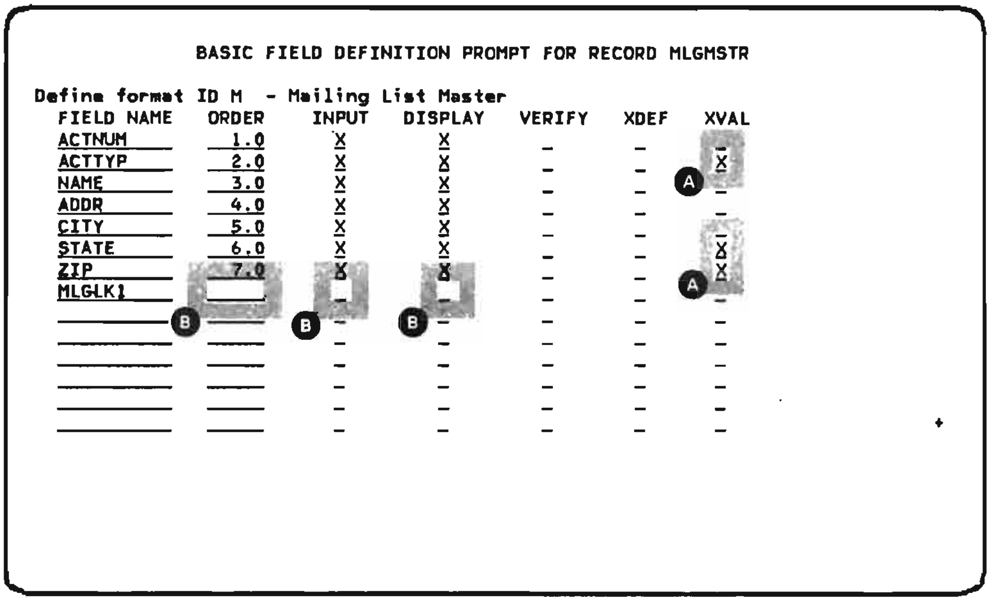
The next prompt is for entry format definition:



```
                     ENTRY FORMAT DEFINITION PROMPT

For record format name MLGMSTR    enter:
  Entry format identifier:        M
  Entry format description:       Mailing List Master
  Display one field per line:     *NO
  Display multiple records:       *NO
  Redisplay changes:              *NO
  Chain to entry format ID:       __
```

A Key in M (for master file).

You then receive the basic field definition prompt, which lists all fields and allows you to select specifications for each field:



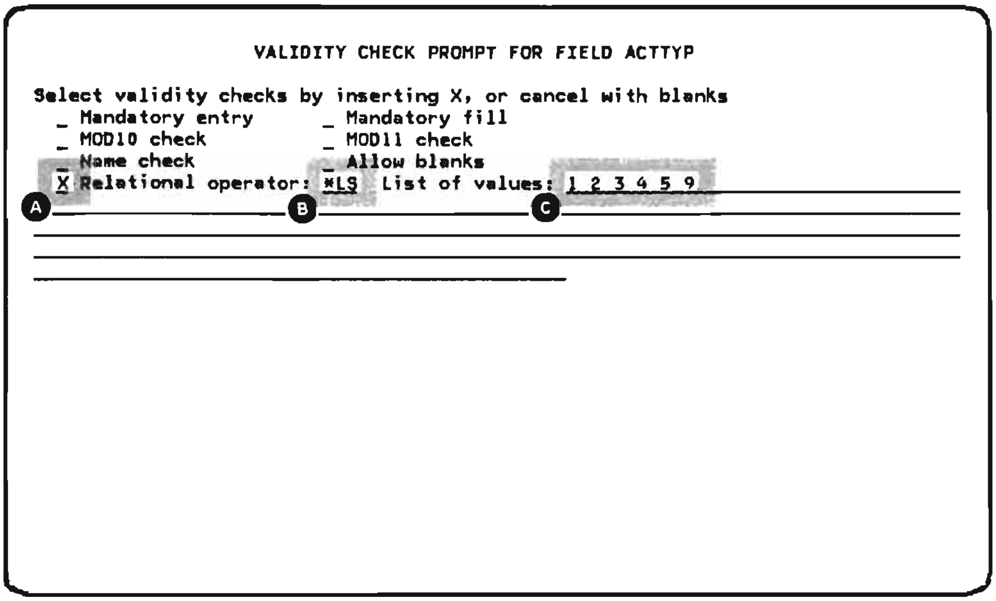
```
                 BASIC FIELD DEFINITION PROMPT FOR RECORD MLGMSTR

 Define format ID M  - Mailing List Master
   FIELD NAME   ORDER    INPUT   DISPLAY   VERIFY   XDEF   XVAL
   ACTNUM        1.0       X        X        _        _      _
   ACTTYP        2.0       X        X        _        _      X
   NAME          3.0       X        X        _        _      _
   ADDR          4.0       X        X        _        _      _
   CITY          5.0       X        X        _        _      _
   STATE         6.0       X        X        _        _      X
   ZIP           7.0       X        X        _        _      X
   MLGLK1        ____      _        _        _        _      _
   __________    ____      _        _        _        _      _
   __________    ____      _        _        _        _      _
   __________    ____      _        _        _        _      _
   __________    ____      _        _        _        _      _
   __________    ____      _        _        _        _      _
   __________    ____      _        _        _        _      _                +
```

A Key in X so that validity checking can be specified for these fields.

B Key a blank into each position (the MLGLK1 field is not used in this approach).

The Xs that automatically appear in the INPUT and DISPLAY columns should remain.

The validity check prompt for ACTTYP is displayed first:



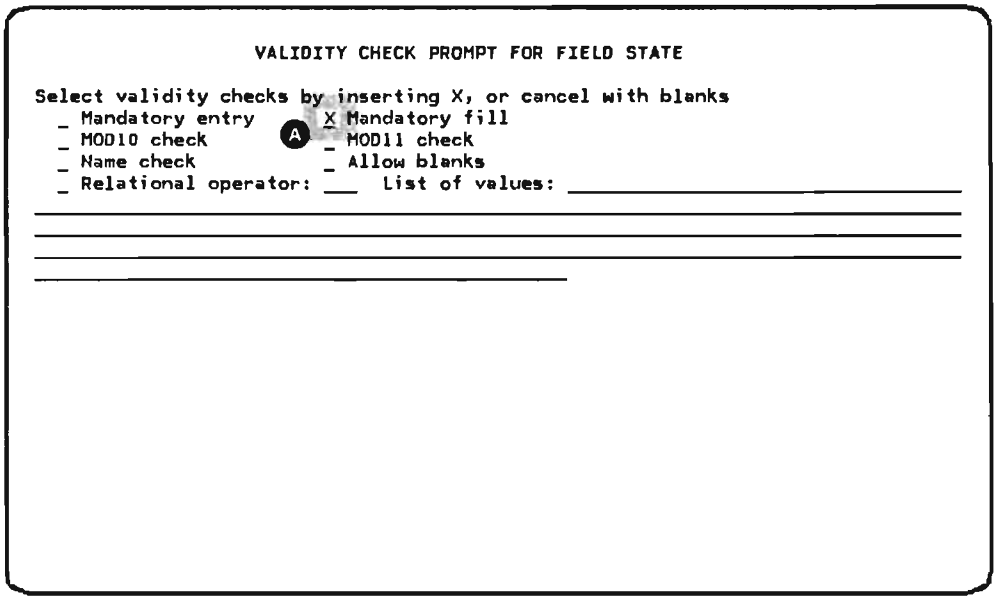
```
                   VALIDITY CHECK PROMPT FOR FIELD ACTTYP

Select validity checks by inserting X, or cancel with blanks
  _ Mandatory entry        _ Mandatory fill
  _ MOD10 check            _ MOD11 check
  _ Name check             _ Allow blanks
  X Relational operator: *LS  List of values: 1 2 3 4 5 9
```

A Key in X.

B Key in *LS to specify there will be a list of values.

C Key in the valid values for which this field will be checked.

The validity check prompt for STATE is displayed next:

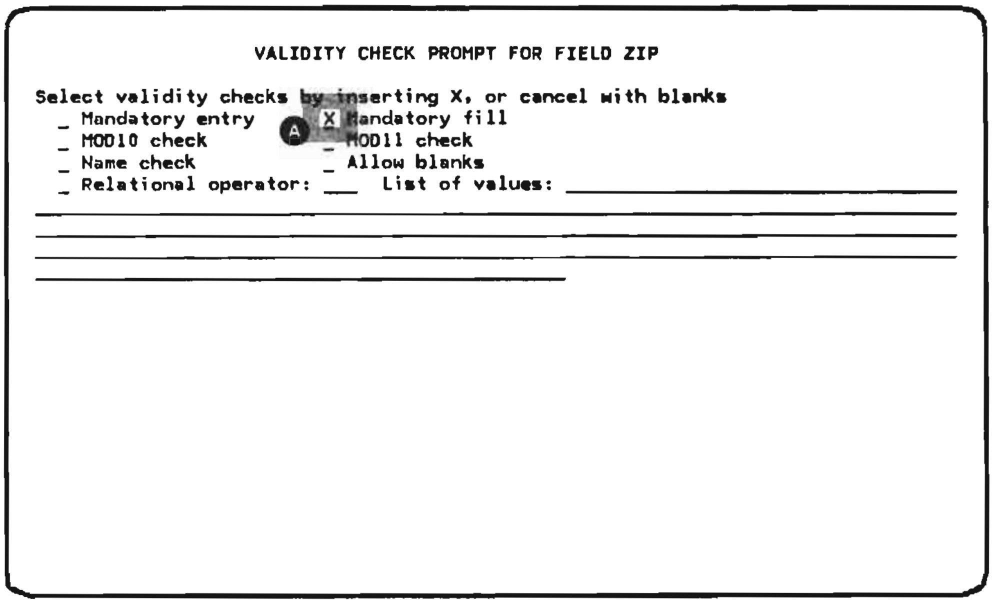


```
                   VALIDITY CHECK PROMPT FOR FIELD STATE

Select validity checks by inserting X, or cancel with blanks
  _ Mandatory entry        X Mandatory fill
  _ MOD10 check            _ MOD11 check
  _ Name check             _ Allow blanks
  _ Relational operator: ___  List of values:
```

A Key in X to specify that this field must contain an entry in each position.

DFU can check a list of up to 20 values. Consequently, all of the valid state codes cannot be checked. The validity checking for the STATE field differs from the previous example for the ACTTYP field where an RPG array was used to validate the field.

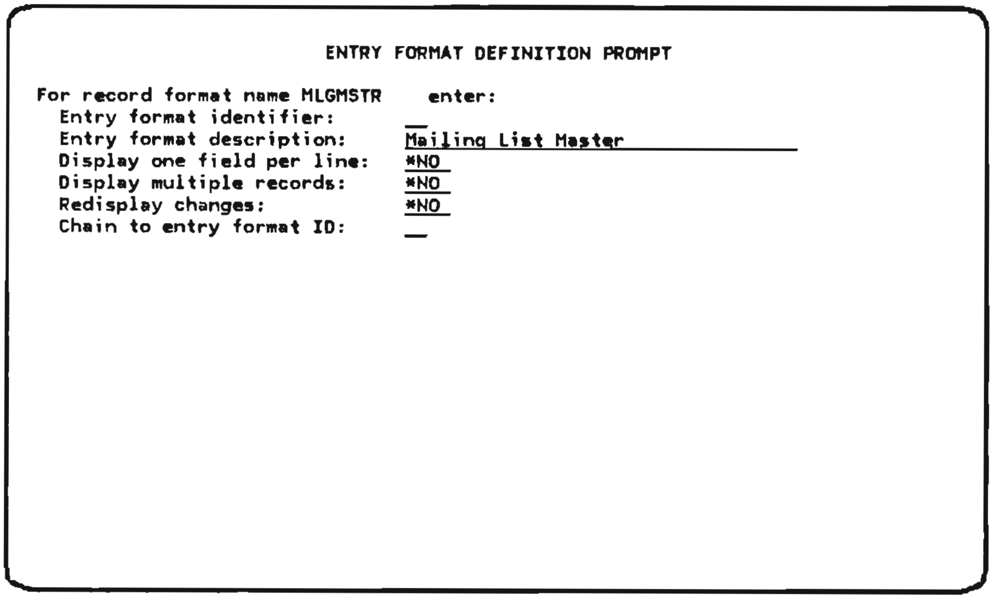
The validity check prompt for ZIP is displayed next:



```
                    VALIDITY CHECK PROMPT FOR FIELD ZIP

Select validity checks by inserting X, or cancel with blanks
  _ Mandatory entry          X Mandatory fill
  _ MOD10 check              _ MOD11 check
  _ Name check               _ Allow blanks
  _ Relational operator: ___   List of values: __________________________
____________________________________________________________________________
____________________________________________________________________________
____________________________________________________________________________
____________________________________________
```

A Key in X to specify that this field must contain an entry in each position.

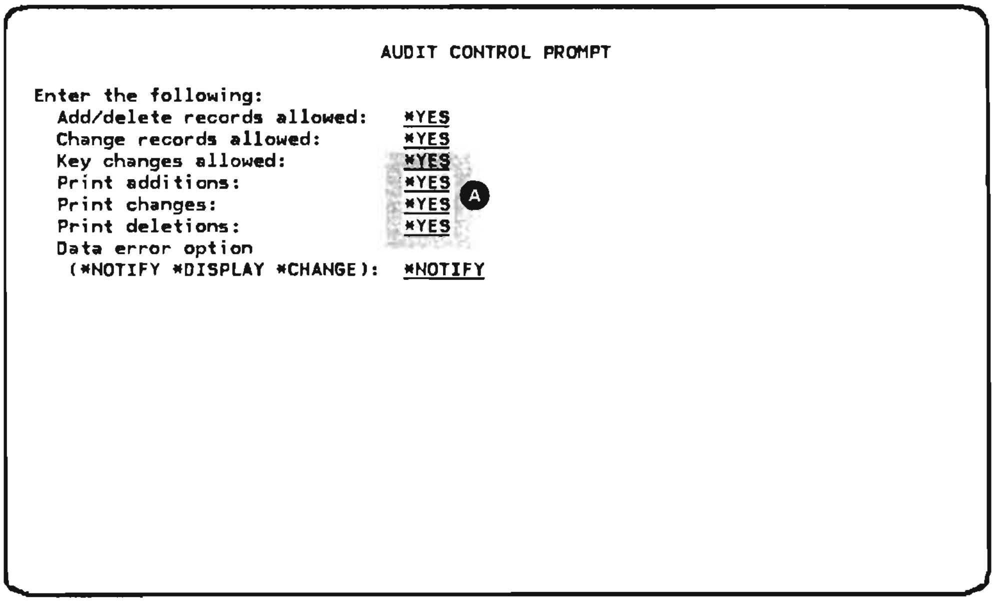
The entry format definition prompt is displayed again:



```
                     ENTRY FORMAT DEFINITION PROMPT

For record format name MLGMSTR    enter:
  Entry format identifier:          __
  Entry format description:       Mailing List Master
  Display one field per line:     *NO
  Display multiple records:       *NO
  Redisplay changes:              *NO
  Chain to entry format ID:       __
```

Because no additional formatting needs to be defined, press the Enter key (without making any entries) to continue the prompting sequence.

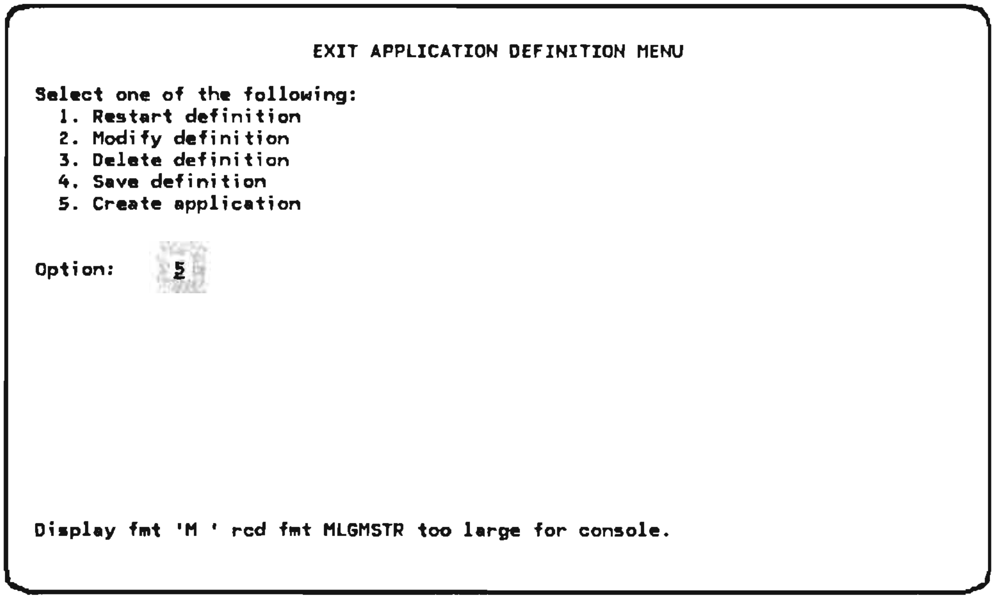
The last prompt used to create the DFU definition is the audit control prompt:



```
                              AUDIT CONTROL PROMPT

 Enter the following:
   Add/delete records allowed:   *YES
   Change records allowed:       *YES
   Key changes allowed:          *YES
   Print additions:              *YES
   Print changes:                *YES
   Print deletions:              *YES
   Data error option
    (*NOTIFY *DISPLAY *CHANGE):  *NOTIFY
```

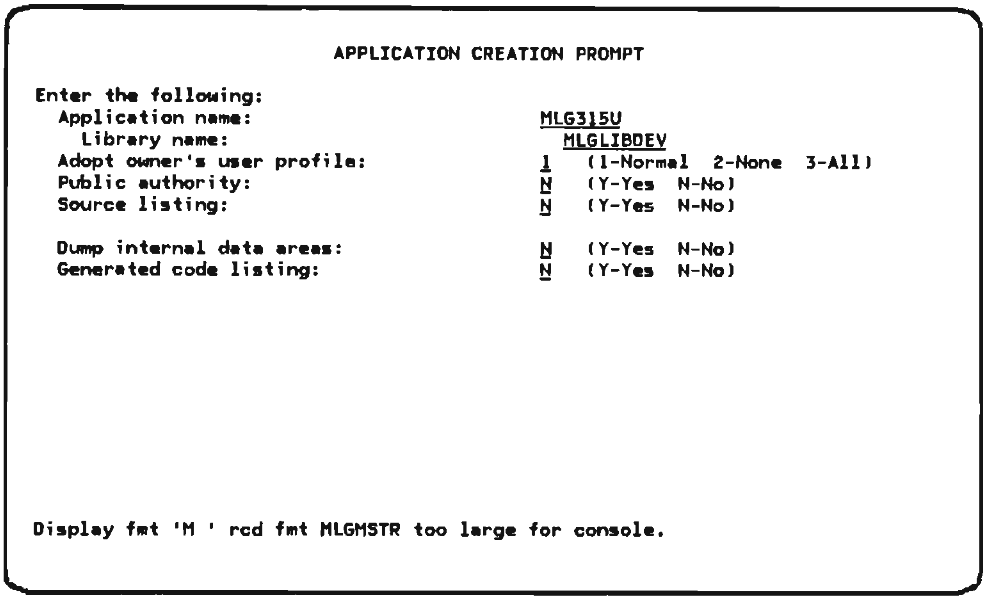
A Key in *YES (*NO is the default) to receive a printed listing.

The exit application definition menu is displayed next. Select option 5 to indicate that you want to create the application.



```
                        EXIT APPLICATION DEFINITION MENU

 Select one of the following:
   1. Restart definition
   2. Modify definition
   3. Delete definition
   4. Save definition
   5. Create application

 Option:     5






 Display fmt 'M ' rcd fmt MLGMSTR too large for console.
```

Because you selected option 5 on the exit application definition menu, you receive the application creation prompt:



```
                          APPLICATION CREATION PROMPT

Enter the following:
  Application name:                       MLG315U
    Library name:                           MLGLIBDEV
  Adopt owner's user profile:             1  (1-Normal  2-None  3-All)
  Public authority:                       N  (Y-Yes  N-No)
  Source listing:                         N  (Y-Yes  N-No)

  Dump internal data areas:               N  (Y-Yes  N-No)
  Generated code listing:                 N  (Y-Yes  N-No)









Display fmt 'M ' rcd fmt MLGMSTR too large for console.
```

To create the MLG315U application that you have just defined, you need only accept the defaults as shown and press the Enter key. When the application creation is complete, the MLGLIBDEV library contains a DFU application named MLG315U as well as a DFU display file named MLG315U.

You next receive a menu for executing the application. In this approach, you do not want to use the application at this point, so you press the CF1 key to exit DFU. You receive the DFU menu. When you press the CF1 key again, the programmer menu reappears.

Executing the DFU Application

By executing the DFU application MLG315U, work station users can enter transactions to update MLGMSTP using the access path defined by MLGMSTL. The MLG315U application can be executed by:

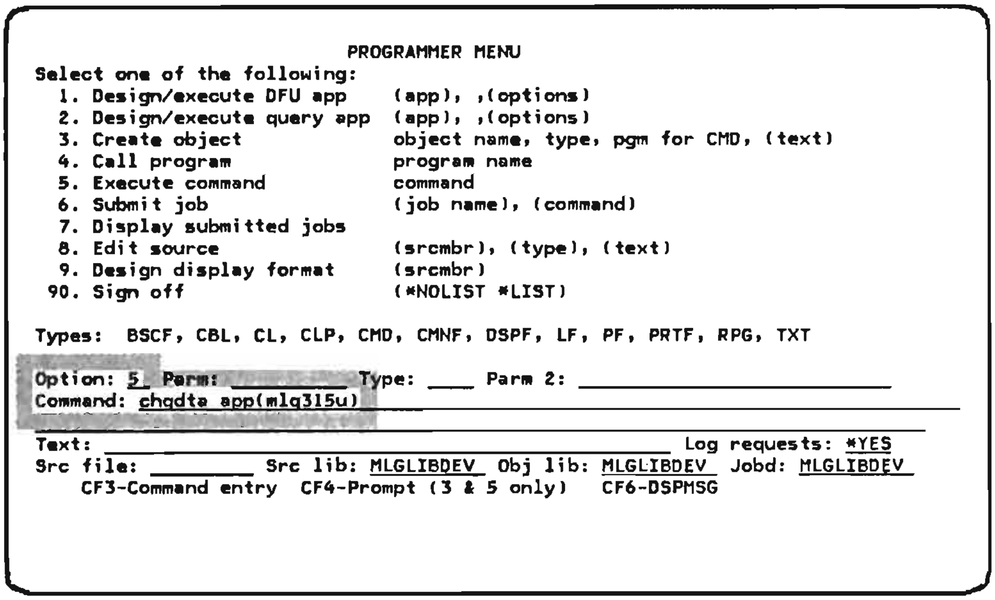
- Selecting option 1 on the programmer menu, as you did to create the application, and then selecting option 2 on the DFU menu when it is displayed.

- Entering the Change Data (CHGDTA) command.

In this approach, the CHGDTA command is used, because its direct result is the first display of the application without any intermediate displays (the DFU menu).

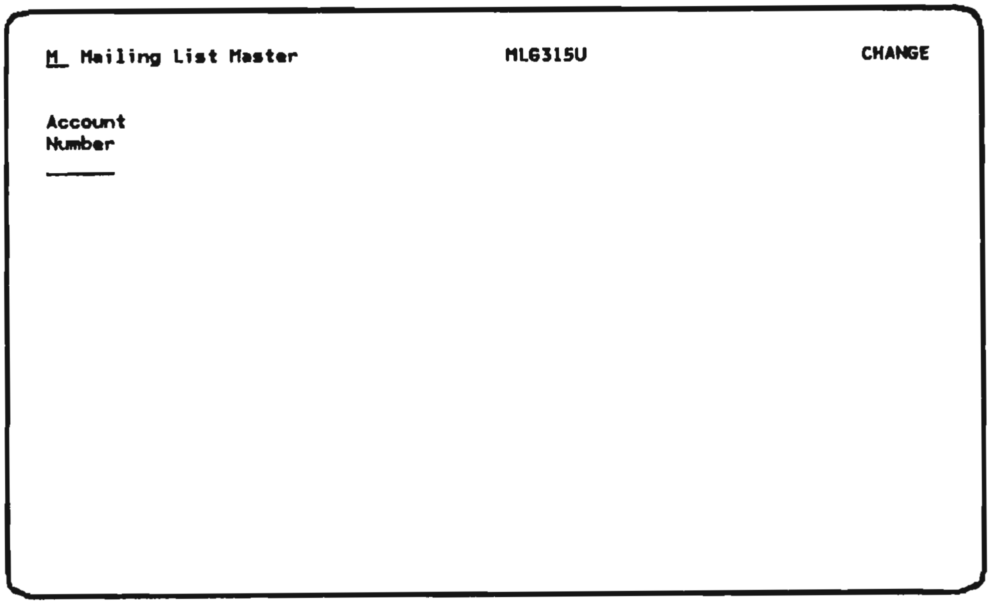
There are several ways that this command could be entered by the work station user. A menu approach is discussed later in this chapter and in Appendix B.

Assume at this point that the command is entered by the programmer through the programmer menu to test the DFU application MLG315U. The command is entered on the programmer menu as follows:



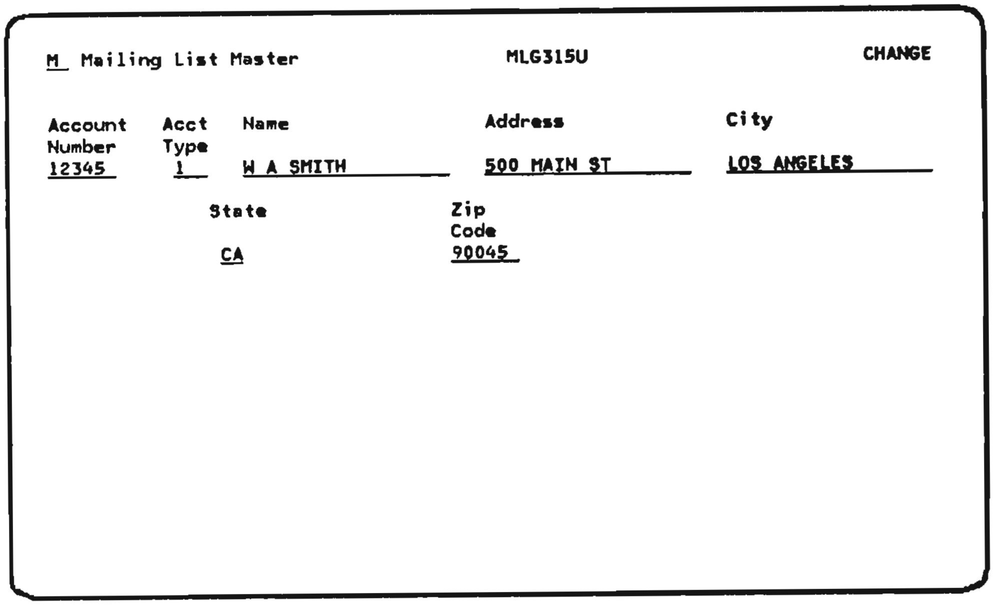
```
                               PROGRAMMER MENU
Select one of the following:
  1. Design/execute DFU app    (app), ,(options)
  2. Design/execute query app  (app), ,(options)
  3. Create object             object name, type, pgm for CMD, (text)
  4. Call program              program name
  5. Execute command           command
  6. Submit job                (job name), (command)
  7. Display submitted jobs
  8. Edit source               (srcmbr), (type), (text)
  9. Design display format     (srcmbr)
 90. Sign off                  (*NOLIST *LIST)

Types:  BSCF, CBL, CL, CLP, CMD, CMNF, DSPF, LF, PF, PRTF, RPG, TXT

Option: 5_ Parm: __________ Type: ____ Parm 2: ____________________________
Command: chgdta app(mlg315u)________________________________________________
____________________________________________________________________________
Text: _______________________________________________________ Log requests: *YES
Src file: ________ Src lib: MLGLIBDEV  Obj lib: MLGLIBDEV  Jobd: MLGLIBDEV
    CF3-Command entry  CF4-Prompt (3 & 5 only)   CF6-DSPMSG
```

When MLG315U is executed, the following display will be in add mode if no records exist in the file. If records do exist, the display will be in change mode as follows:



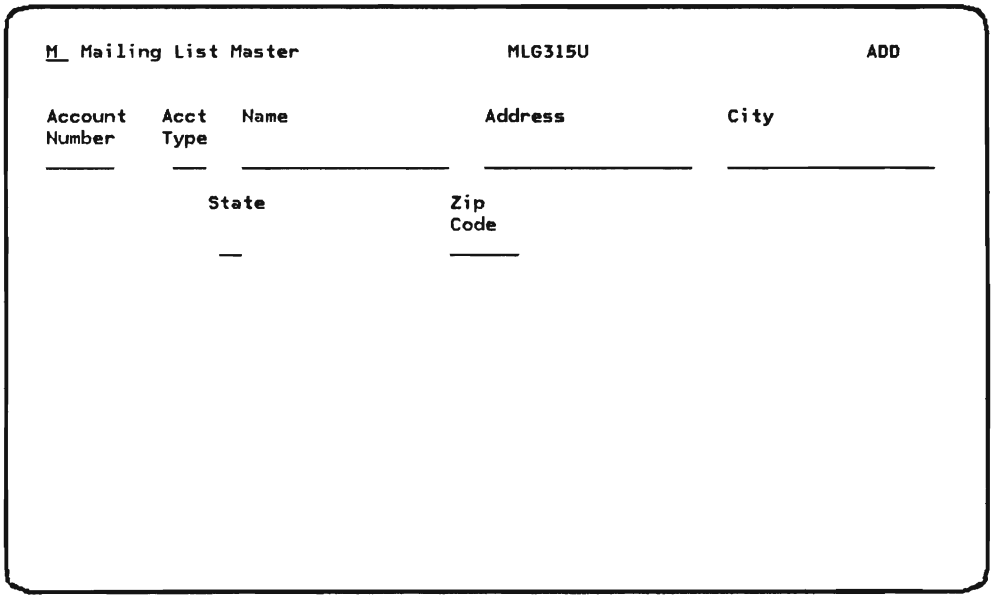
```
M_ Mailing List Master                MLG315U                CHANGE

Account
Number
______
```

This display allows you to enter the account number of the record you wish to change. After you have entered an account number, the record with that account number is displayed as follows:



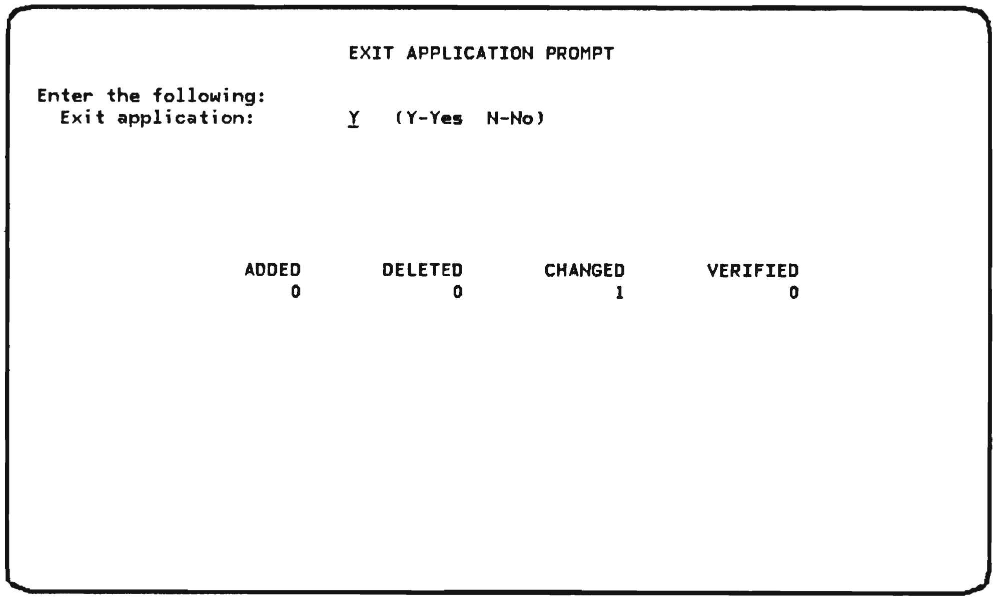
```
M_ Mailing List Master                MLG315U                CHANGE

Account   Acct   Name                 Address            City
Number    Type
12345_     1_    W A SMITH_________   500 MAIN ST_____   LOS ANGELES_______

               State               Zip
                                   Code
                CA                 90045_
```

You can change any of the fields on this display by keying in the changes and pressing the Enter key. The CHANGE prompt will be displayed again. Also, the command keys can be used to perform various functions at this time. For example, if you press the Add key (CF9), the following is displayed and you can add records to the master file:



```
M_ Mailing List Master                   MLG315U                        ADD

Account   Acct   Name                    Address                  City
Number    Type
______    __     __________________      __________________       __________________

               State                  Zip
                                      Code
                 _                    ______
```

When all of your maintenance has been performed, press the Exit Application key (CF1). The following exit application prompt is displayed:



```
                           EXIT APPLICATION PROMPT

Enter the following:
  Exit application:          Y   (Y-Yes  N-No)



                  ADDED        DELETED        CHANGED        VERIFIED
                      0              0              1               0
```

To end the application, use the default (Y) as shown.

A listing of the changes, additions, and deletions made to the master file is written to a spooled output file and placed on the QPRINT output queue for printing (this was specified in the audit control prompt). A sample printed listing is shown below.

```
5714UT1  R02M00 810302                          AUDIT LOG                                   09/21/81   14:41:57   PAGF   1

Application:   MLG315U.MLGLIBDEV
File:          MLGMSTL.MLGLIBDEV      Member:        MLGMSTL

ACTION       RECORD        Account     Acct     Name                  Address                 City                 State   Zip
                           Number      Type                                                                                Code
PEFORE CHG   MLGMSTR        12345       1       W A SMITH          * 500 MAIN ST            * LOS ANGELES           CA    * 90045
AFTER  CHG   MLGMSTR        12345       1       W A SMITH            320 PACIFIC AVE          SAN JOSF              CA      95112
```

USING THE PROGRAM CALL MENU

Any work station user can be given authorization to any command on the system, be limited to specific commands, or be given only application-oriented functions. Therefore, you must consider:

- What the work station user should be allowed to do.

- How the work station user's task can be made easier.

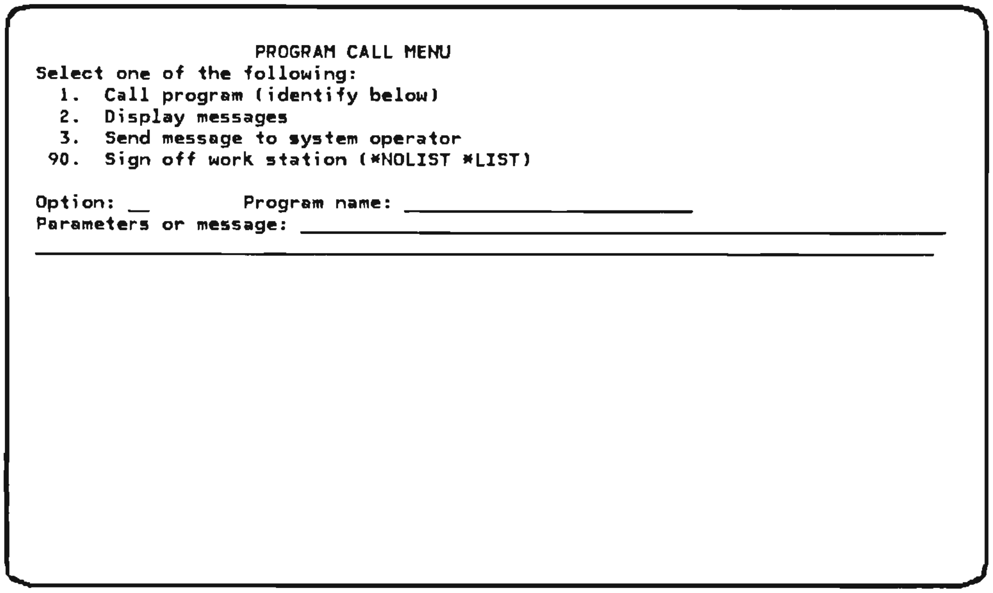
There are many alternatives to signing on and presenting the first display for the work station users. These alternatives may require changes such as:

- Modifying the IBM-supplied password

- Creating unique user profiles

- Creating application-oriented menus (discussed later in this chapter and in Appendix B)

In the previous approach, transactions were entered as a result of the programmer using the CHGDTA command to test the DFU application MLG315U. To limit the direct exposure of the work station user to the control language commands, this approach assumes the IBM-supplied user profile QUSER and the IBM-supplied program call menu (QCALLMENU) are used. This menu is displayed when the work station user signs on with the password USER:

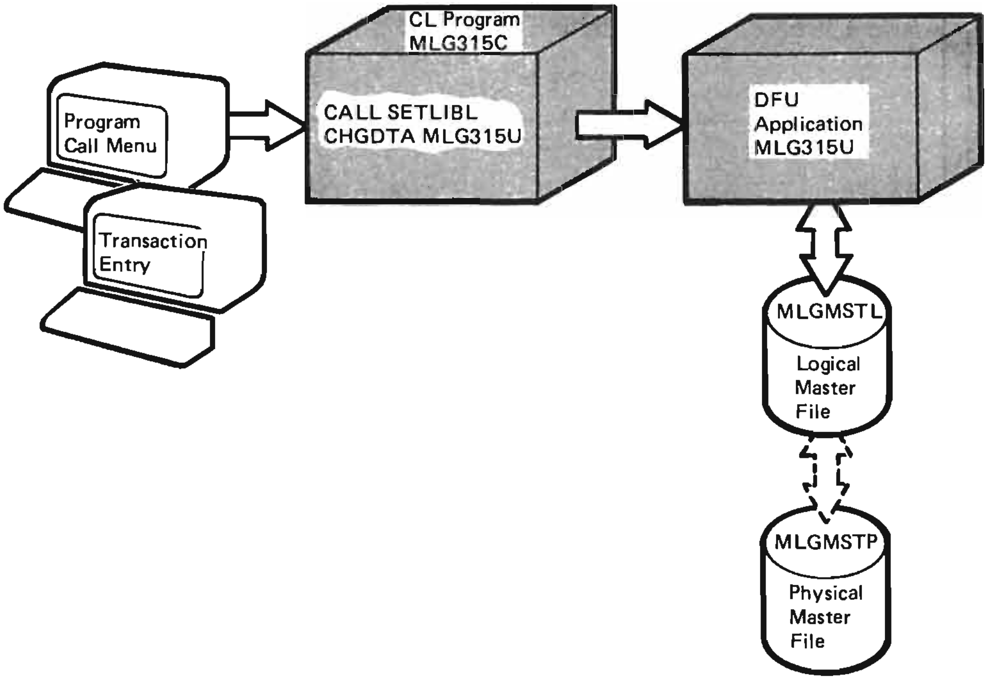


```
                    PROGRAM CALL MENU
Select one of the following:
  1.  Call program (identify below)
  2.  Display messages
  3.  Send message to system operator
 90.  Sign off work station (*NOLIST *LIST)

Option: __        Program name: ______________________
Parameters or message: ________________________________________________
_______________________________________________________________________
```

The program call menu allows the work station user to call a program that has been created by the programmer. The other options let the user:

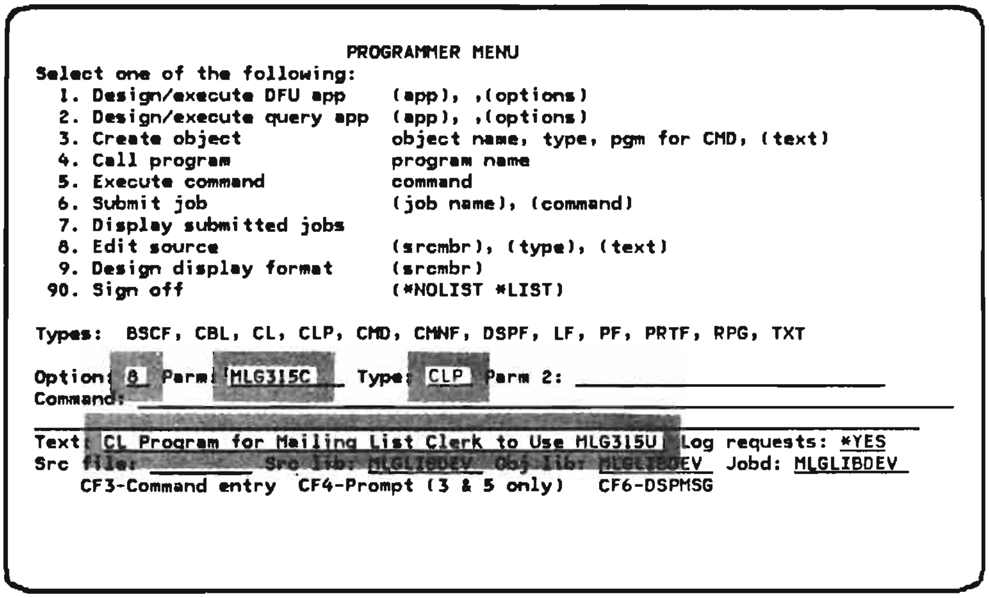
- Display any messages sent to his work station
- Send a message to the system operator
- Sign off

Although the program call menu allows the work station user to call a program, it does not allow the execution of most commands. To directly execute the DFU application MLG315U described earlier in this chapter, the CHGDTA command must be used. You can create a CL program to execute one or more commands for the work station user. For this approach to signing on and calling the application from the program call menu, the programmer develops a CL program named MLG315C. This CL program allows the work station user to execute the commands necessary for the DFU application without knowing the specific commands.



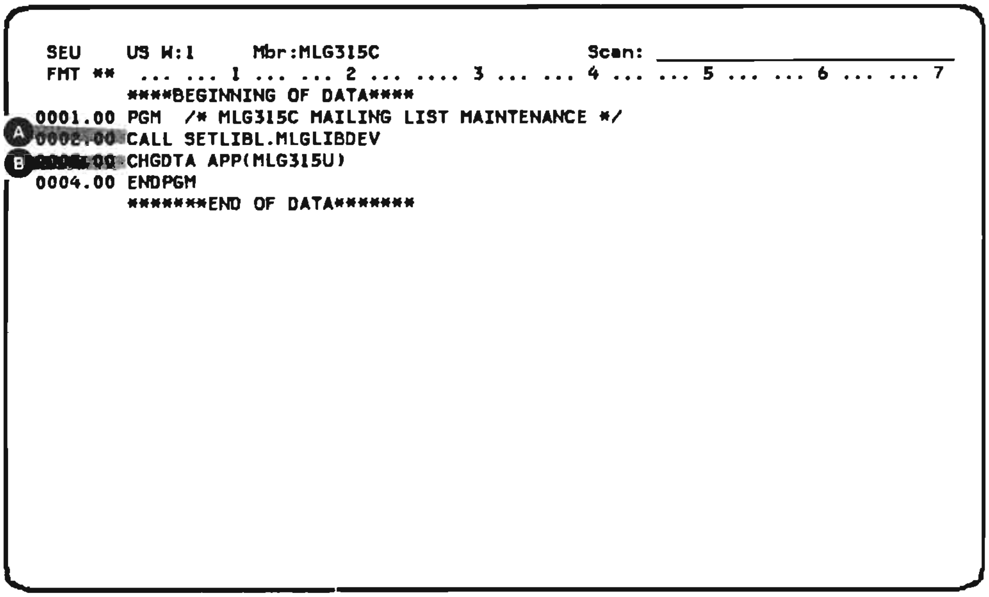
Creating the Control Language Program

Because SEU will be used to enter the source for the CL program, called MLG315C, select option 8 on the programmer menu as follows:



```
                              PROGRAMMER MENU
Select one of the following:
  1. Design/execute DFU app    (app), ,(options)
  2. Design/execute query app  (app), ,(options)
  3. Create object             object name, type, pgm for CMD, (text)
  4. Call program              program name
  5. Execute command           command
  6. Submit job                (job name), (command)
  7. Display submitted jobs
  8. Edit source               (srcmbr), (type), (text)
  9. Design display format     (srcmbr)
 90. Sign off                  (*NOLIST *LIST)

Types:  BSCF, CBL, CL, CLP, CMD, CMNF, DSPF, LF, PF, PRTF, RPG, TXT

Option: 8  Parm: MLG315C    Type: CLP  Parm 2: ___________________________
Command: _____________________________________________________________________
_______________________________________________________________________________
Text: CL Program for Mailing List Clerk to Use MLG315U  Log requests: *YES
Src file: ________  Src lib: MLGLIBDEV  Obj lib: MLGLIBDEV  Jobd: MLGLIBDEV
    CF3-Command entry  CF4-Prompt (3 & 5 only)   CF6-DSPMSG
```

When you receive the SEU display, enter the following source on the display:

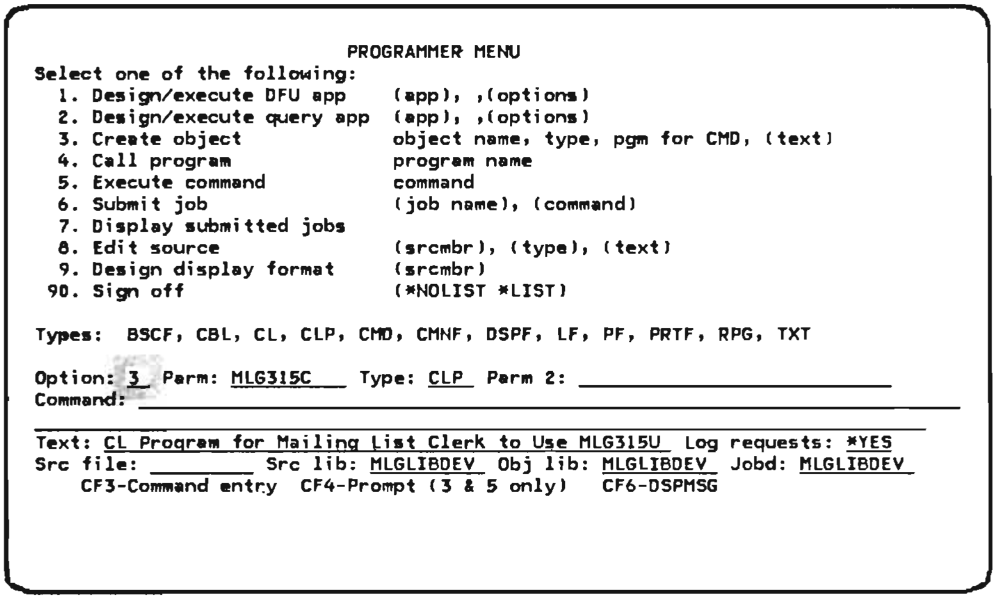


```
 SEU    US W:1     Mbr:MLG315C                   Scan: ________________________
 FMT **  ... ... 1 ... ... 2 ... .... 3 ... ... 4 ... ... 5 ... ... 6 ... ... 7
        ****BEGINNING OF DATA****
0001.00 PGM  /* MLG315C MAILING LIST MAINTENANCE */
0002.00 CALL SETLIBL.MLGLIBDEV
0003.00 CHGDTA APP(MLG315U)
0004.00 ENDPGM
        *******END OF DATA*******
```

A The CALL command is needed to replace the library list for the mailing list application.

B This command is the same as the CHGDTA command previously used to test this application (see *Executing the DFU Application* in this chapter).

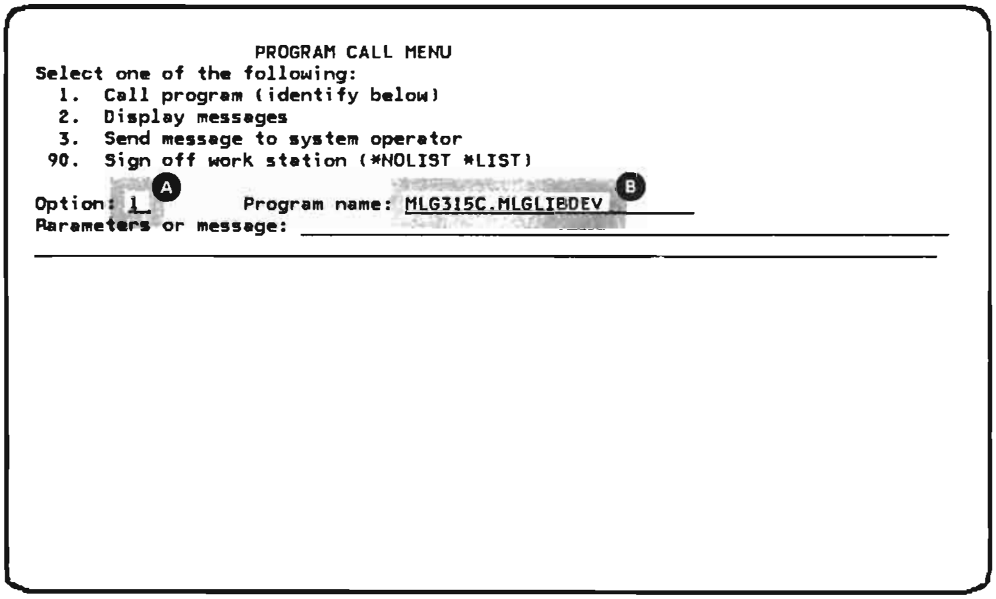
After exiting SEU, you create the CL program from the programmer menu as follows:



```
                              PROGRAMMER MENU
Select one of the following:
  1. Design/execute DFU app    (app), ,(options)
  2. Design/execute query app  (app), ,(options)
  3. Create object             object name, type, pgm for CMD, (text)
  4. Call program              program name
  5. Execute command           command
  6. Submit job                (job name), (command)
  7. Display submitted jobs
  8. Edit source               (srcmbr), (type), (text)
  9. Design display format     (srcmbr)
 90. Sign off                  (*NOLIST *LIST)

Types:  BSCF, CBL, CL, CLP, CMD, CMNF, DSPF, LF, PF, PRTF, RPG, TXT

Option: 3_ Parm: MLG315C___ Type: CLP_ Parm 2: ______________________________
Command: ______________________________________________________________________
_____________________________________________________________________________
Text: CL Program for Mailing List Clerk to Use MLG315U_ Log requests: *YES
Src file: __________ Src lib: MLGLIBDEV_ Obj lib: MLGLIBDEV_ Jobd: MLGLIBDEV_
    CF3-Command entry  CF4-Prompt (3 & 5 only)   CF6-DSPMSG
```

Using the Control Language Program

After the CL program is created, the work station user can call the program from the program call menu as follows:



```
                        PROGRAM CALL MENU
Select one of the following:
  1.  Call program (identify below)
  2.  Display messages
  3.  Send message to system operator
 90.  Sign off work station (*NOLIST *LIST)

Option: 1_        Program name: MLG315C.MLGLIBDEV_______
Parameters or message: ________________________________________________
_____________________________________________________________________________
```

A Key in 1 to call a program.

B Key in the qualified name (program name and library name).

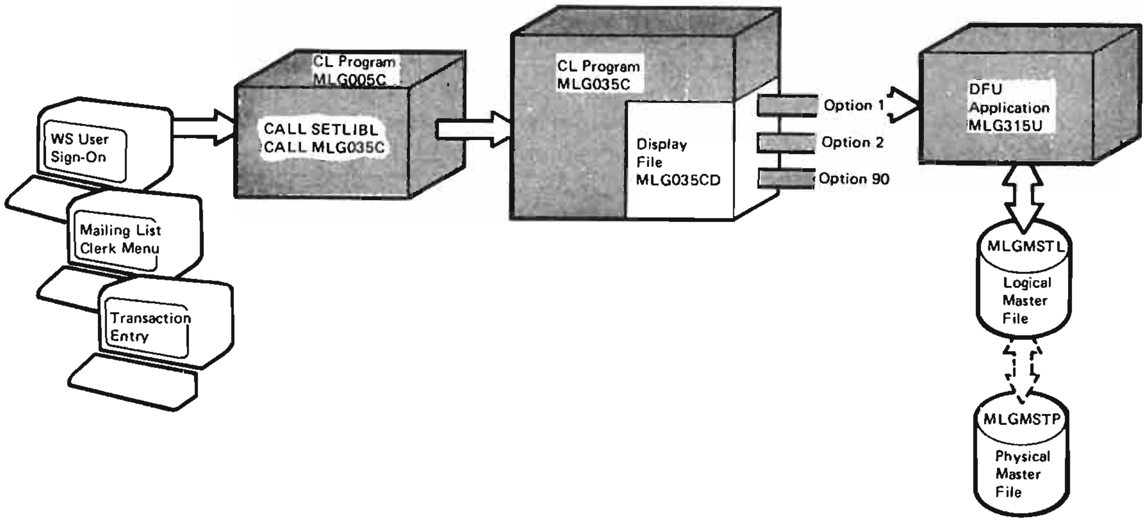
After the work station user has entered the indicated information on this display, the DFU application MLG315U begins. The displays that follow would be the same as those previously described in this chapter under *Executing the DFU Application.*

Programming Considerations

In this approach, the work station user had to enter a qualified name (both the program name and the library name) for the program to be called. When the work station user signed on, a system wide default library list was used. This list would normally not include the library MLGLIBDEV. Calling the program would have been easier for the work station user if the CL program MLG315C had been placed in the general purpose library (QGPL). The work station user would have only been required to enter the program name. Another alternative for calling this program is discussed next in this chapter.

USING A USER PROFILE AND AN APPLICATION-ORIENTED MENU

In the previous section, work station users used the program call menu to simplify their data entry work. In this section, a different approach to simplifying the work station user's job is discussed: a user profile is created such that an application-oriented menu appears after the work station user signs on. This type of menu not only simplifies but helps control what the work station user can do. For example, you may want to restrict which work station users can maintain the master file or what applications can be used by a specific user.

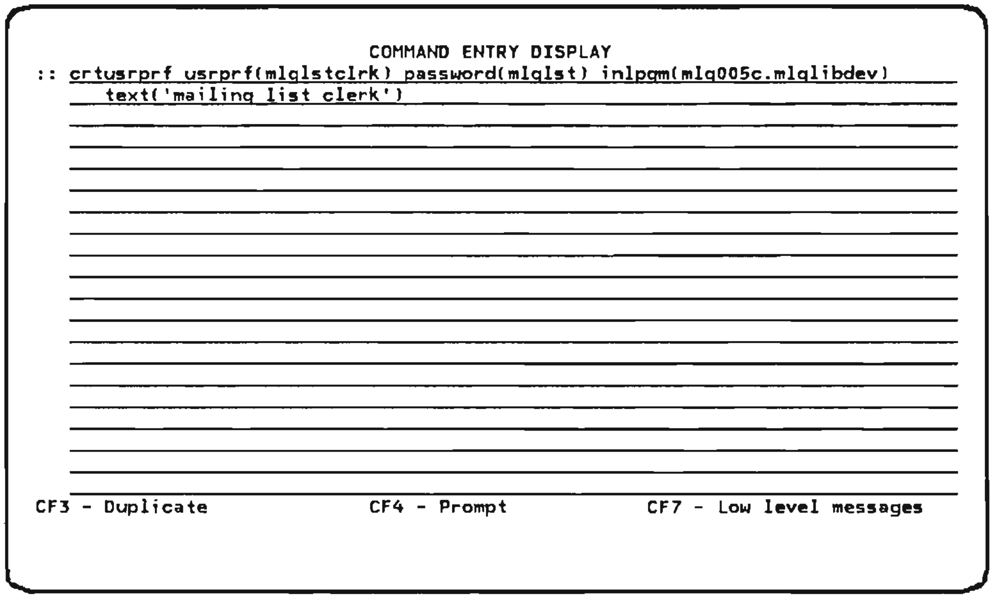


Creating the User Profile

There are several ways to allow the work station user to receive a specific menu after he signs on. This section discusses how to create a user profile to do this.

Only the security officer can create or change a user profile. The security officer may be the DP manager, the programmer, or another person signing on with the security officer password. When you sign on with the security officer profile, you receive the command entry display.

To create a user profile for all work station users that will be working with mailing lists, enter the CRTUSRPRF command on the command entry display as follows:



```
                          COMMAND ENTRY DISPLAY
:: crtusrprf usrprf(mlglstclrk) password(mlglst) inlpgm(mlg005c.mlglibdev)
      text('mailing list clerk')

CF3 - Duplicate             CF4 - Prompt             CF7 - Low level messages
```

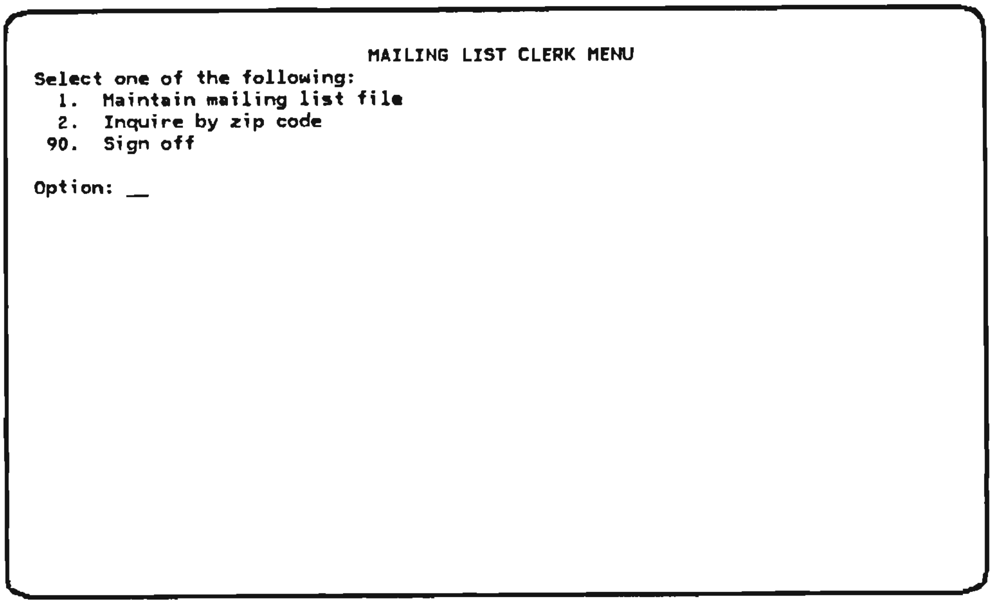
The work station user can now sign on with the password MLGLST. The name of his user profile is MLGLSTCLRK, and the user profile calls the CL program MLG005C in the library MLGLIBDEV every time the work station user signs on. The program displays the mailing list clerk menu as the first display after the password MLGLST is entered.

Programming Considerations

The user profile that was created could be used by one person or by more than one person (all could be active at the same time). In many work environments it would be desirable to create a unique user profile for each user.

Application-Oriented Menu

The application-oriented menu for this approach is shown below.



```
                              MAILING LIST CLERK MENU
Select one of the following:
  1.  Maintain mailing list file
  2.  Inquire by zip code
 90.  Sign off

Option: __
```

Three options are shown:

- If option 1 is selected, the CL program MLG005C executes the DFU application MLG315U, which allows the work station user to update the mailing list master file.

- If option 2 is selected, the CL program executes the DFU application MLG230U, which allows the work station user to display the master file by name and zip code.

- If option 90 is selected, the SIGNOFF command is executed and the sign-on prompt appears.

Before this menu can be displayed, the application environment must be established. This is done by the CL program MLG005C.

The CL program MLG005C replaces the library list for the application, then calls the program MLG035C, which displays the application-oriented menu. The source for the CL program MLG005C would be entered by using SEU (select option 8 and specify type CLP on the programmer menu). The CL program source follows:

```
PGM /*MLG005C MAILING LIST INVOCATION FOR MAILING LIST CLERK */
CALL SETLIBL.MLGLIBDEV
CALL MLG035C
ENDPGM
```

The CL program MLG005C is created by selecting option 3 and specifying MLG005C for source member and CLP for type on the programmer menu. The set library list program (SETLIBL) must be called before the CL program (MLG035C) for the mailing list clerk menu is called because the program MLG035C is coded to find a file (MLG035CD) by using the library list. When the program MLG035C is executed, it displays the mailing list clerk menu.

Creating the Menu

To create the menu for the mailing list clerk, two items must be created in the following order:

1. Display file for menu

2. CL program for displaying menu

There are two ways you can create the menu:

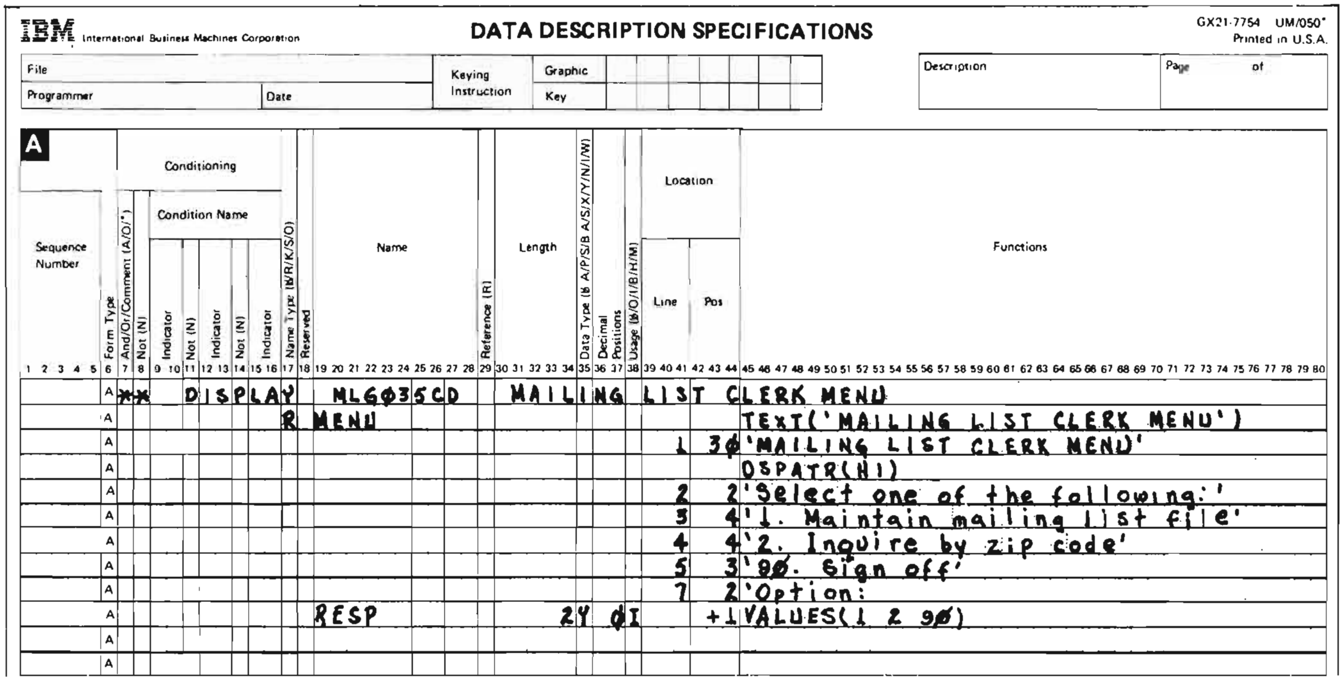
- Code and create the display file and CL program separately.

- Use the *screen design aid* (SDA), which creates the display file and CL program for you.

The procedures for separately creating the display file and the CL program will be described first, followed by the procedure for using SDA.

Creating the Display File MLG035CD

The display file must be created first because the CL program MLG035C uses the externally described data. The DDS for the display file MLG035CD are shown in Figure 8-1.



IBM International Business Machines Corporation

DATA DESCRIPTION SPECIFICATIONS

GX21-7754 UM/050*
Printed in U.S.A.

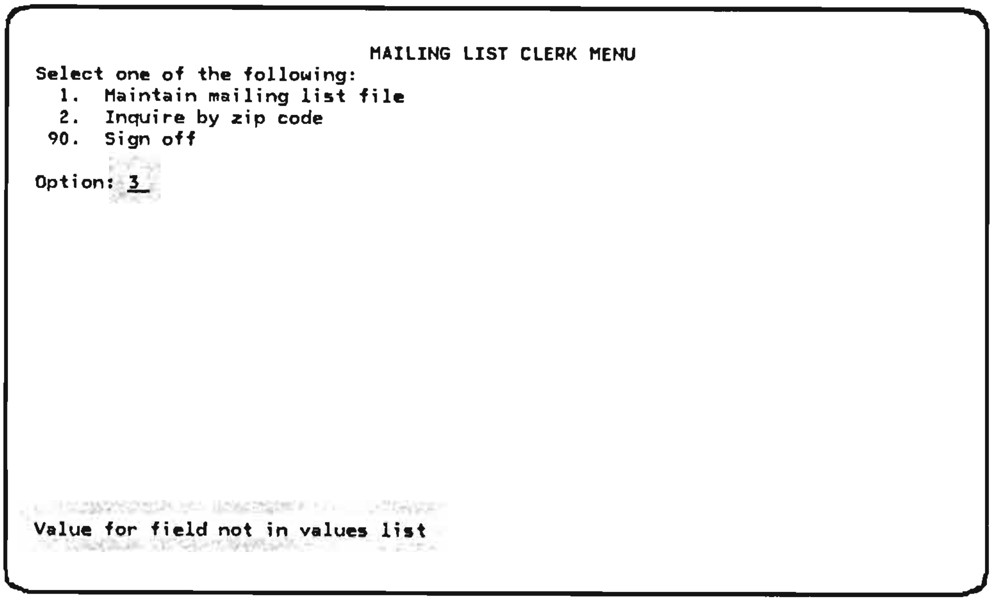
File | Keying Instruction | Graphic | Key | Programmer | Date | Description | Page of

A | Sequence Number | Form Type | And/Or/Comment (A/O/*) | Conditioning: Condition Name: Not (N), Indicator, Not (N), Indicator, Not (N), Indicator | Name Type (b/R/K/S/O) | Reserved | Name | Reference (R) | Length | Data Type (b A/P/S/B A/S/X/Y/N/I/W) | Decimal Positions | Usage (b/O/I/B/H/M) | Location: Line, Pos | Functions

```
     A** DISPLAY  MLGØ35CD   MAILING LIST CLERK MENU
     A          R MENU                      TEXT('MAILING LIST CLERK MENU')
     A                                  1 3Ø'MAILING LIST CLERK MENU'
     A                                      DSPATR(HI)
     A                                  2  2'Select one of the following:'
     A                                  3  4'1. Maintain mailing list file'
     A                                  4  4'2. Inquire by zip code'
     A                                  5  3'9Ø. Sign off'
     A                                  7  2'Option:
     A            RESP           2Y Ø    +1VALUES(1 2 9Ø)
     A
     A
```

Figure 8-1. DDS for MLG035CD Display File

The DDS define the format of the menu and primarily consist of constants and an input field (RESP). The RESP field is checked for valid entries. If an invalid value (a value other than one of those listed for VALUES) is entered, the device support detects the error and a message is displayed:



```
                              MAILING LIST CLERK MENU
Select one of the following:
  1.  Maintain mailing list file
  2.  Inquire by zip code
 90.  Sign off

Option: 3_














Value for field not in values list
```

Consequently, you do not need to check for invalid codes in the CL program.

The first line in the DDS is a comment (designated by * in position 7). The file contains one record format, named MENU (designated by R in position 17).

A text description, TEXT('MAILING LIST CLERK MENU'), is given for the format name. Constants that will be displayed are quoted (enclosed in apostrophes). The line and position entries (under location) specify where the fields will be displayed on the screen. The position entry specifies the first (leftmost) position of the field.

The device displays the title in high intensity because the keyword DSPATR(HI) is specified.

The DDS for the display file MLG035CD would be entered using SEU by selecting option 8 and specifying MLG035C for source member and DSPF for type on the programmer menu. To create the display file, select option 3 and specify DSPF for type on the programmer menu.

Creating the Control Language Program MLG035C

The CL program (MLG035C) for displaying the mailing list menu defined by the MLG035CD display file is the following:

```
     PGM /* MLG035C  MAILING LIST MENU */
     DCLF MLG035CD
BEGIN:  SNDRCVF RCDFMT(MENU)
     IF (&RESP *EQ 1) CHGDTA APP(MLG315U)
     IF (&RESP *EQ 2) DSPDTA APP(MLG230U)
     IF (&RESP *EQ 90) SIGNOFF
     GOTO BEGIN
     ENDPGM
```

The Declare File (DCLF) command specifies the name of the display file which, in this case, is MLG035CD. The display file fields that are to be passed between the display file and the CL program are automatically declared in the CL program when it is created.

The first command executed is the Send Receive File (SNDRCVF) command. This sends the record format MENU to the work station display and waits for a response (one of the options on the menu).

When a response is returned, it is compared with the three valid responses in the IF statements. The &RESP field designates a variable that is being used in the CL program. This variable is defined in the display file as RESP and is automatically declared within the program as &RESP. Variables within CL programs always begin with an & symbol. Different commands are executed depending on the response. If a DFU application is executed, the Exit Application key (CF1) must be pressed before you return to the CL program.

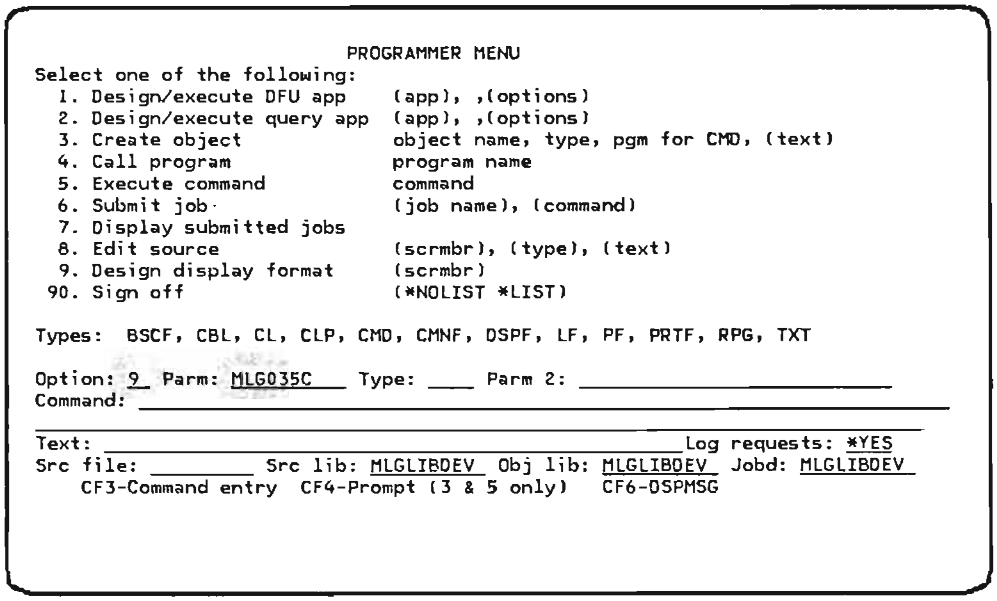
When you return to the CL program, the value of &RESP is not changed and the remaining IF statement(s) test the value of &RESP. These tests are not successful and the program branches to the BEGIN label, which displays the menu again.

If the SIGNOFF command is executed, the CL program is terminated.

The CL program MLG035C would be entered using SEU by selecting option 8 and specifying MLG035C for source member and CLP for type on the programmer menu. To create the program, select option 3 and specify CLP for type on the programmer menu.

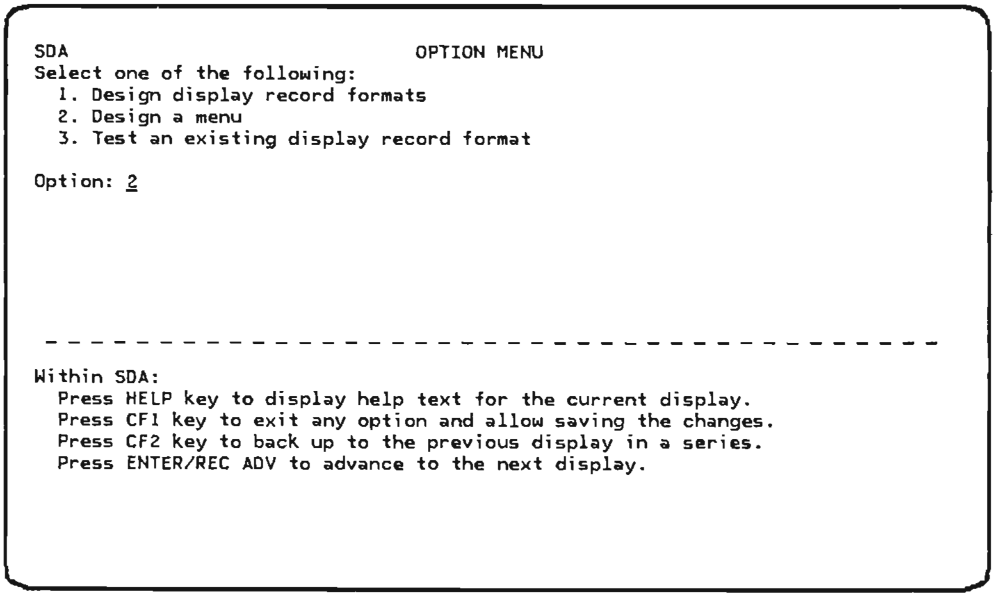
The previous two sections showed how you can create a specialized menu by separately coding and then creating the display file (MLG035CD) and the CL program (MLG035C) for the menu. By using the *screen design aid* (SDA) utility, you can avoid the task of coding the display file and CL program. You define the menu on the SDA displays, and SDA then creates the display file and CL program for you.

To request SDA, use option 9 on the programmer menu as follows:



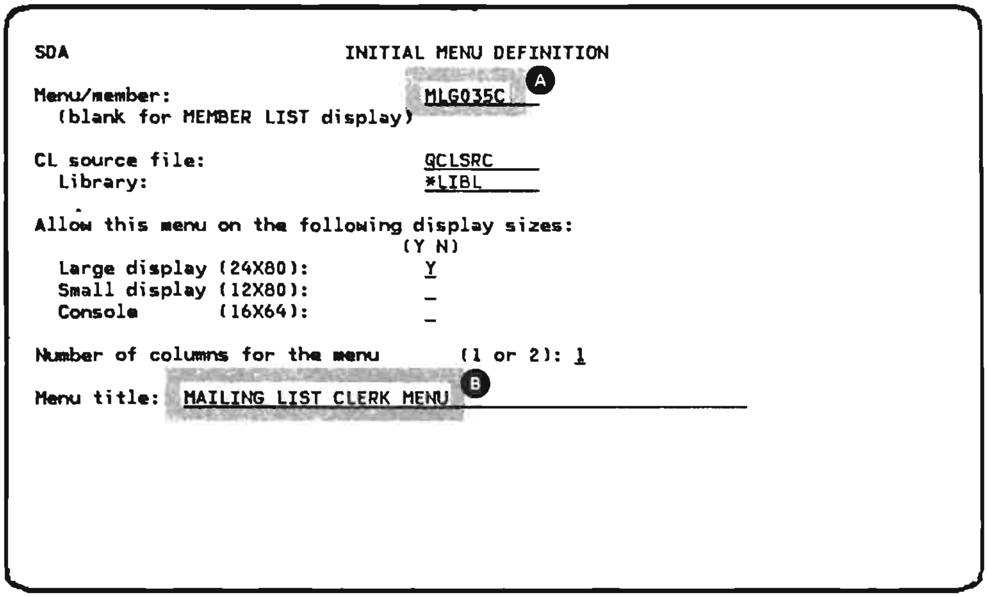
```
                           PROGRAMMER MENU
Select one of the following:
  1. Design/execute DFU app    (app), ,(options)
  2. Design/execute query app  (app), ,(options)
  3. Create object             object name, type, pgm for CMD, (text)
  4. Call program              program name
  5. Execute command           command
  6. Submit job·               (job name), (command)
  7. Display submitted jobs
  8. Edit source               (scrmbr), (type), (text)
  9. Design display format     (scrmbr)
 90. Sign off                  (*NOLIST *LIST)

Types:  BSCF, CBL, CL, CLP, CMD, CMNF, DSPF, LF, PF, PRTF, RPG, TXT

Option: 9_ Parm: MLG035C___ Type: ____ Parm 2: ___________________________
Command: _____________________________________________________________________
______________________________________________________________________________
Text: ___________________________________________________Log requests: *YES
Src file: _________ Src lib: MLGLIBDEV  Obj lib: MLGLIBDEV  Jobd: MLGLIBDEV
    CF3-Command entry  CF4-Prompt (3 & 5 only)   CF6-DSPMSG
```

You receive the first SDA display. To create a menu, select option 2:



```
SDA                               OPTION MENU
Select one of the following:
  1. Design display record formats
  2. Design a menu
  3. Test an existing display record format

Option: 2






   - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

Within SDA:
  Press HELP key to display help text for the current display.
  Press CF1 key to exit any option and allow saving the changes.
  Press CF2 key to back up to the previous display in a series.
  Press ENTER/REC ADV to advance to the next display.
```

Because you indicated you want to design a menu, you next receive the initial menu definition display. You can accept the values already shown on this display, or you can change them. Use this display to specify the title of the menu.

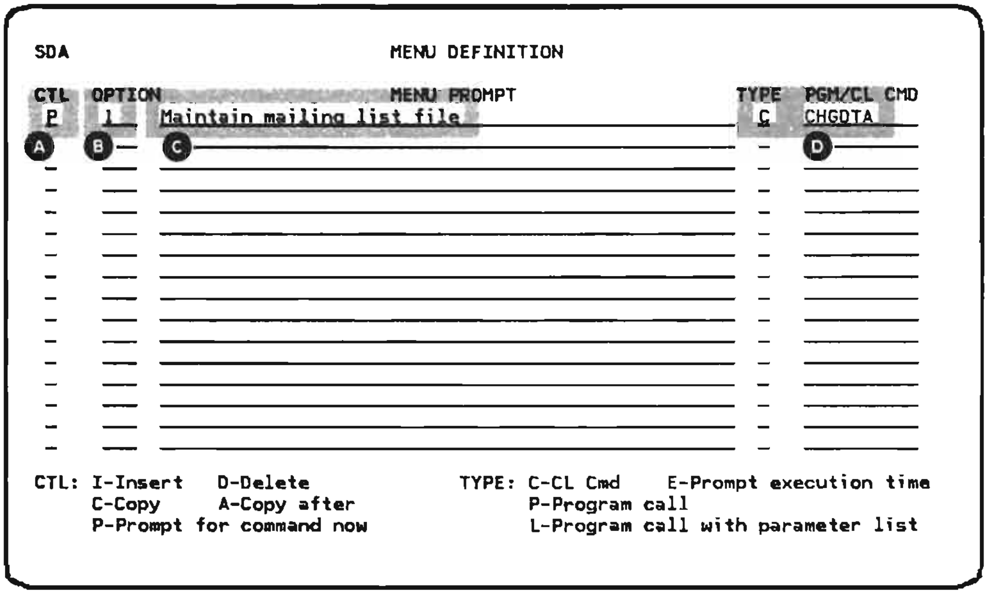


```
SDA                          INITIAL MENU DEFINITION

Menu/member:                             MLG035C___
  (blank for MEMBER LIST display)

CL source file:                          QCLSRC____
  Library:                               *LIBL_____

Allow this menu on the following display sizes:
                                        (Y N)
  Large display (24X80):                 Y
  Small display (12X80):                 _
  Console       (16X64):                 _

Number of columns for the menu       (1 or 2): 1

Menu title:  MAILING LIST CLERK MENU_____________________
```

A This is the name you entered on the programmer menu.

B Specify the title that is to appear at the top of the menu.

You next receive a display on which you define each option of the menu:



```
SDA                               MENU DEFINITION

CTL  OPTION                       MENU PROMPT                  TYPE  PGM/CL CMD
 P    1_   Maintain mailing list file__________________________  C   CHGDTA____
 _    __   ____________________________________________________  _   __________
 _    __   ____________________________________________________  _   __________
 _    __   ____________________________________________________  _   __________
 _    __   ____________________________________________________  _   __________
 _    __   ____________________________________________________  _   __________
 _    __   ____________________________________________________  _   __________
 _    __   ____________________________________________________  _   __________
 _    __   ____________________________________________________  _   __________
 _    __   ____________________________________________________  _   __________
 _    __   ____________________________________________________  _   __________
 _    __   ____________________________________________________  _   __________
 _    __   ____________________________________________________  _   __________
 _    __   ____________________________________________________  _   __________
 _    __   ____________________________________________________  _   __________
 _    __   ____________________________________________________  _   __________

CTL: I-Insert    D-Delete                TYPE: C-CL Cmd    E-Prompt execution time
     C-Copy      A-Copy after                  P-Program call
     P-Prompt for command now                  L-Program call with parameter list
```

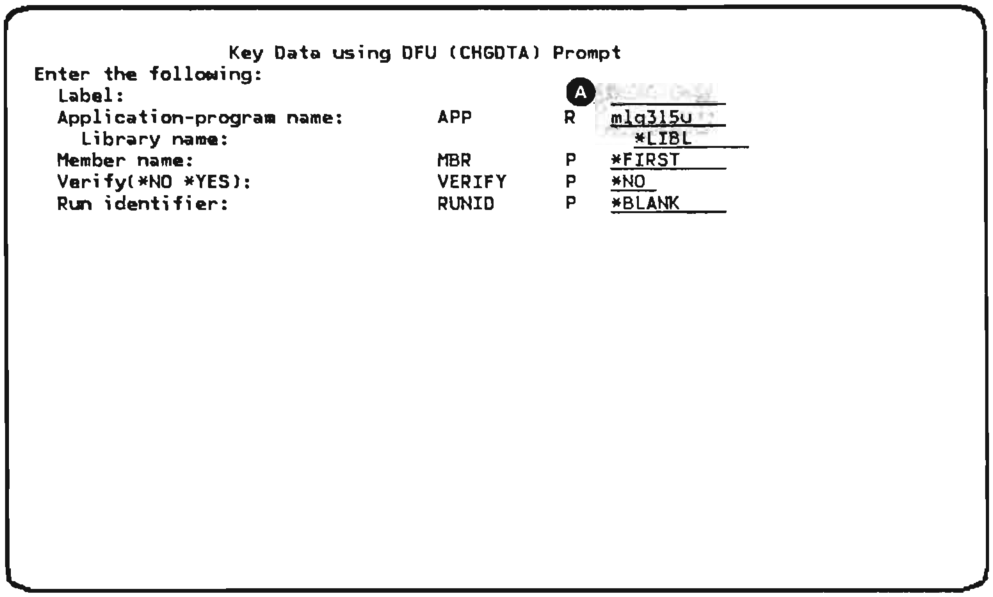
A Requests the prompt for the CHGDTA command specified in field D, so that you can enter parameters.

B Indicates the first option.

C Defines the text description to appear on the menu.

D Specifies that the CHGDTA command is to be executed when option 1 is selected.

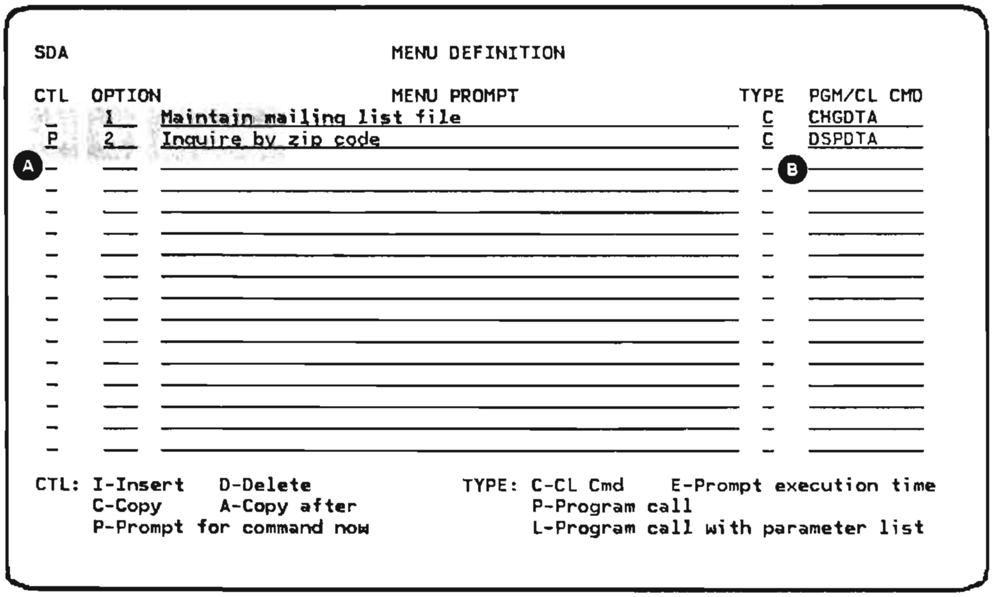
Because you requested prompting, you receive the CHGDTA prompt when you press the Enter key:



```
                 Key Data using DFU (CHGDTA) Prompt
Enter the following:
  Label:                                          ________
  Application-program name:       APP        R   mlg315u_
    Library name:                                  *LIBL___
  Member name:                    MBR        P   *FIRST___
  Verify(*NO *YES):               VERIFY     P   *NO_
  Run identifier:                 RUNID      P   *BLANK___
```

A Key in MLG315U here to specify that the MLG315U application is to start execution when a user selects option 1 on the menu.

When you press the Enter key after completing the prompt, the menu definition display appears again for you to define the second option of the menu:

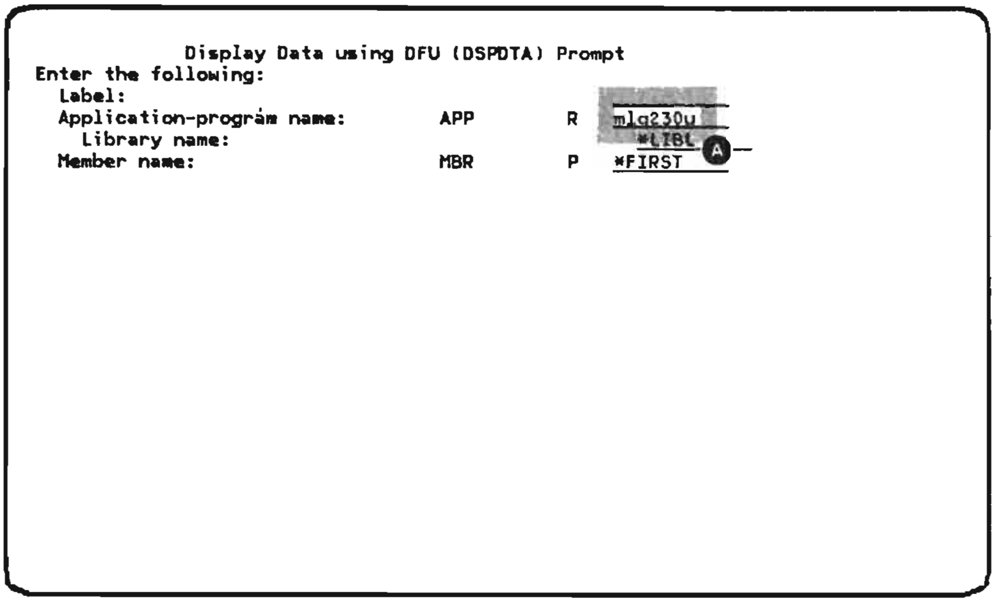


```
SDA                               MENU DEFINITION

CTL  OPTION                       MENU PROMPT                    TYPE  PGM/CL CMD
 _     1    Maintain mailing list file                              C    CHGDTA
 P     2    Inquire by zip code                                     C    DSPDTA
 _    ___   ______________________________________________          _    ________
 _    ___   ______________________________________________          _    ________
 _    ___   ______________________________________________          _    ________
 _    ___   ______________________________________________          _    ________
 _    ___   ______________________________________________          _    ________
 _    ___   ______________________________________________          _    ________
 _    ___   ______________________________________________          _    ________
 _    ___   ______________________________________________          _    ________
 _    ___   ______________________________________________          _    ________
 _    ___   ______________________________________________          _    ________
 _    ___   ______________________________________________          _    ________
 _    ___   ______________________________________________          _    ________
 _    ___   ______________________________________________          _    ________
 _    ___   ______________________________________________          _    ________

CTL: I-Insert   D-Delete                TYPE: C-CL Cmd    E-Prompt execution time
     C-Copy     A-Copy after                  P-Program call
     P-Prompt for command now                 L-Program call with parameter list
```

A Requests prompting for the DSPDTA command specified in field **B**.

B Specifies that the DSPDTA command is to be executed when option 2 is selected.

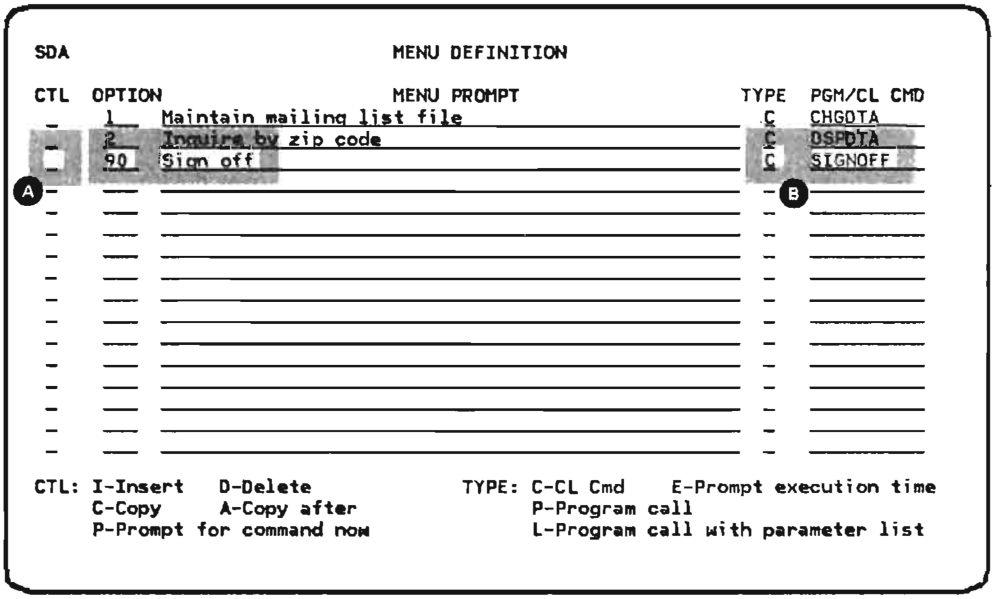
When you press the Enter key, you receive the DSPDTA prompt:



```
              Display Data using DFU (DSPDTA) Prompt
Enter the following:
  Label:
  Application-program name:        APP        R  mlg230u
    Library name:                                  *LIBL
  Member name:                     MBR        P  *FIRST
```

A Key in MLG230U here to specify that the MLG230U application is to start execution when a user selects option 2 on the menu.

After completing the DSPDTA prompt, you again receive the menu definition display to define the next option:

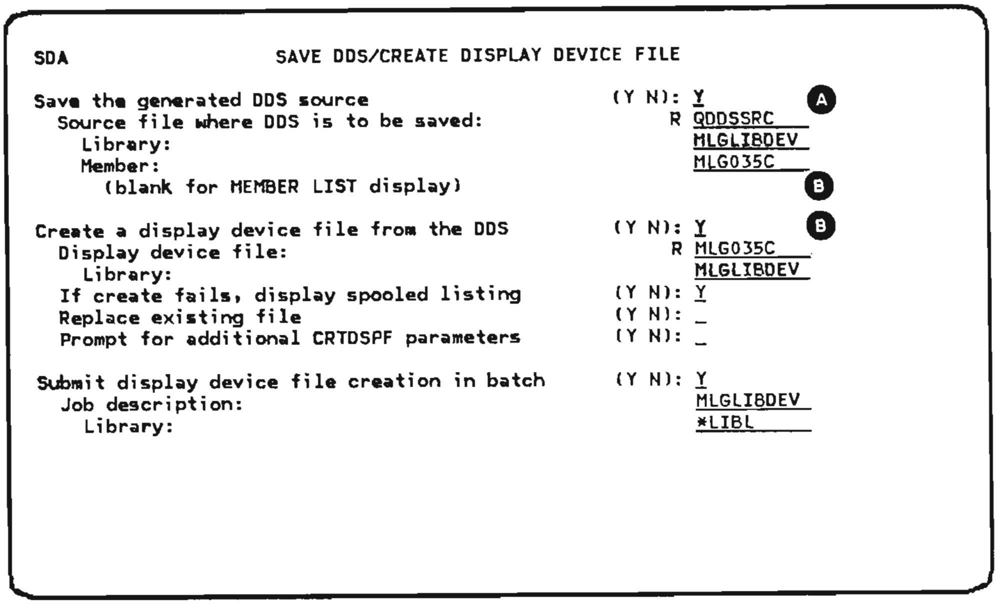


```
SDA                              MENU DEFINITION

CTL  OPTION                      MENU PROMPT                         TYPE  PGM/CL CMD
 _    1    Maintain mailing list file                                  C   CHGDTA
 _    2    Inquire by zip code                                         C   DSPDTA
      90   Sign off                                                    C   SIGNOFF

CTL: I-Insert    D-Delete                    TYPE: C-CL Cmd     E-Prompt execution time
     C-Copy      A-Copy after                      P-Program call
     P-Prompt for command now                      L-Program call with parameter list
```

A Prompting is *not* requested because default values are to be used for the SIGNOFF command specified in field B.

B Specifies that the SIGNOFF command is to be executed when option 90 is selected.

When you have defined the third option, you press the CF1 key to indicate that the menu definition is complete. You then receive a display on which you verify the names to be used in creating the display device file for the menu. The names shown are based on what you specified on the programmer menu. In this example, you want the name of the display file and its associated source member in QDDSSRC to be MLG035CD, so you change the names shown:

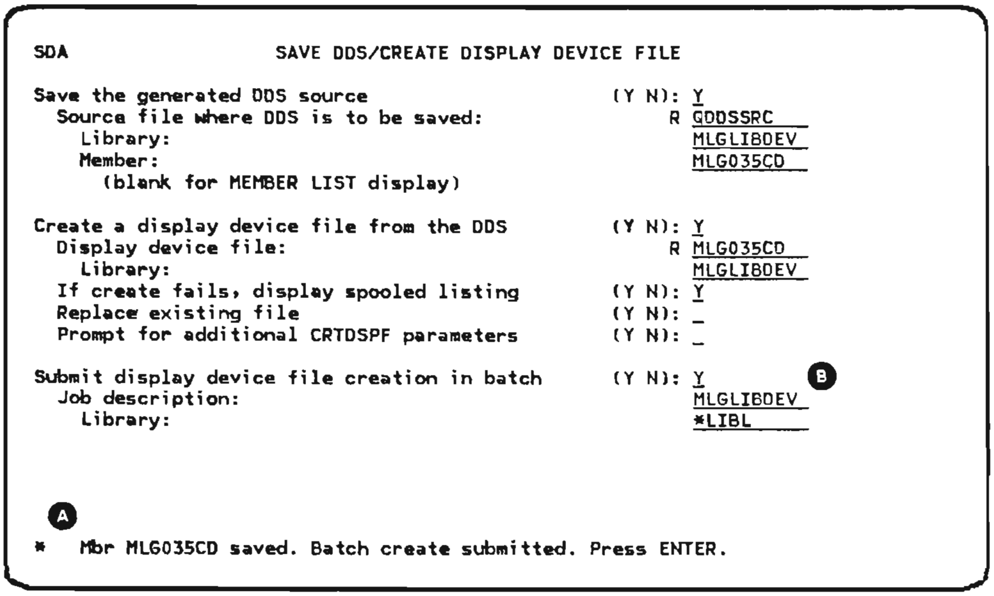


```
SDA                    SAVE DDS/CREATE DISPLAY DEVICE FILE

Save the generated DDS source                        (Y N): Y
  Source file where DDS is to be saved:                   R QDDSSRC
    Library:                                                MLGLIBDEV
    Member:                                                 MLG035C
      (blank for MEMBER LIST display)

Create a display device file from the DDS            (Y N): Y
  Display device file:                                    R MLG035C
    Library:                                                MLGLIBDEV
  If create fails, display spooled listing           (Y N): Y
  Replace existing file                              (Y N): _
  Prompt for additional CRTDSPF parameters           (Y N): _

Submit display device file creation in batch         (Y N): Y
  Job description:                                          MLGLIBDEV
    Library:                                                *LIBL
```

A The source for the display file will be placed in QDDSSRC.

B Change the source member name and the display file name to MLG035CD.

When you press the Enter key, a batch job is submitted to create the MLG035CD display file:

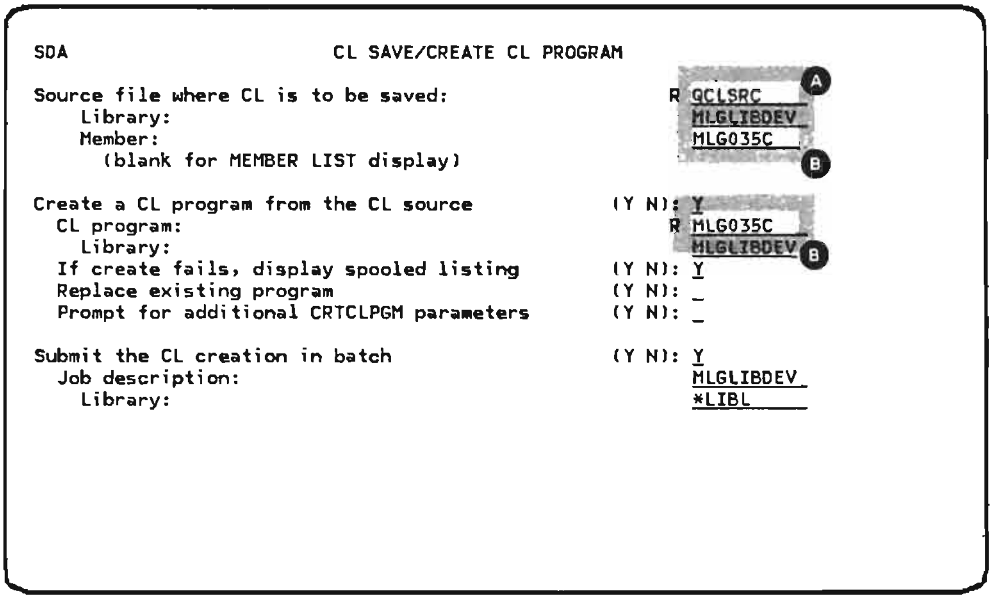


```
SDA                    SAVE DDS/CREATE DISPLAY DEVICE FILE

Save the generated DDS source                        (Y N): Y
  Source file where DDS is to be saved:                   R QDDSSRC
    Library:                                                MLGLIBDEV
    Member:                                                 MLG035CD
      (blank for MEMBER LIST display)

Create a display device file from the DDS            (Y N): Y
  Display device file:                                    R MLG035CD
    Library:                                                MLGLIBDEV
  If create fails, display spooled listing           (Y N): Y
  Replace existing file                              (Y N): _
  Prompt for additional CRTDSPF parameters           (Y N): _

Submit display device file creation in batch         (Y N): Y
  Job description:                                          MLGLIBDEV
    Library:                                                *LIBL


*  Mbr MLG035CD saved. Batch create submitted. Press ENTER.
```

A Message indicates a batch job submitted to create the display file.

B The submitted job will use the job description that you specified on the programmer menu.

When you press the Enter key again, you receive a display to verify the names to be used in creating the CL program for the menu. The names shown are based on what you specified on the programmer menu:

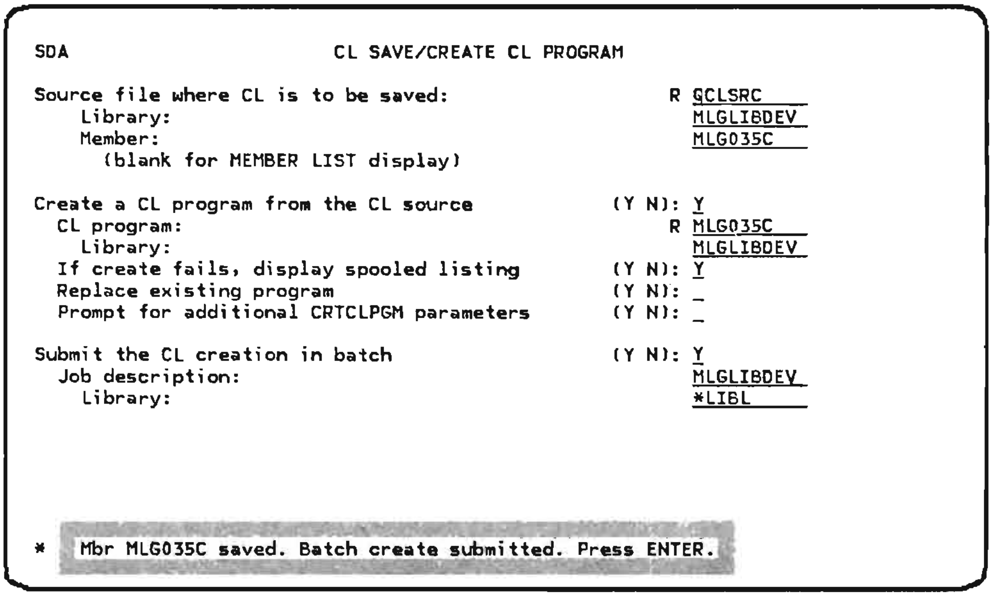


```
SDA                         CL SAVE/CREATE CL PROGRAM

Source file where CL is to be saved:                      R QCLSRC
    Library:                                                MLGLIBDEV
    Member:                                                 MLG035C
      (blank for MEMBER LIST display)

Create a CL program from the CL source              (Y N): Y
  CL program:                                             R MLG035C
    Library:                                                MLGLIBDEV
  If create fails, display spooled listing          (Y N): Y
  Replace existing program                          (Y N): _
  Prompt for additional CRTCLPGM parameters         (Y N): _

Submit the CL creation in batch                     (Y N): Y
  Job description:                                          MLGLIBDEV
    Library:                                                *LIBL
```

A The source for the CL program will be placed in QCLSRC.

B The source member in QCLSRC and the CL program will be given this name.

Because you want the CL program name to be as shown, you press the Enter key without changing the display. The creation of the MLG035C program is submitted as a batch job:



```
SDA                         CL SAVE/CREATE CL PROGRAM

Source file where CL is to be saved:                      R QCLSRC
    Library:                                                MLGLIBDEV
    Member:                                                 MLG035C
      (blank for MEMBER LIST display)

Create a CL program from the CL source              (Y N): Y
  CL program:                                             R MLG035C
    Library:                                                MLGLIBDEV
  If create fails, display spooled listing          (Y N): Y
  Replace existing program                          (Y N): _
  Prompt for additional CRTCLPGM parameters         (Y N): _

Submit the CL creation in batch                     (Y N): Y
  Job description:                                          MLGLIBDEV
    Library:                                                *LIBL



*  Mbr MLG035C saved. Batch create submitted. Press ENTER.
```

The menu definition is now complete. When you press the Enter key again, you receive the initial menu definition display to begin the definition of another menu. To exit the menu definition process, you press the CF1 key. You then receive the SDA option menu. To exit SDA, you press the CF1 key again.

When the creation of the display file is completed, you receive a message that the batch job used to create the display file has ended normally. You receive a similar message when the creation of the CL program is completed. The MLGLIBDEV library then contains:

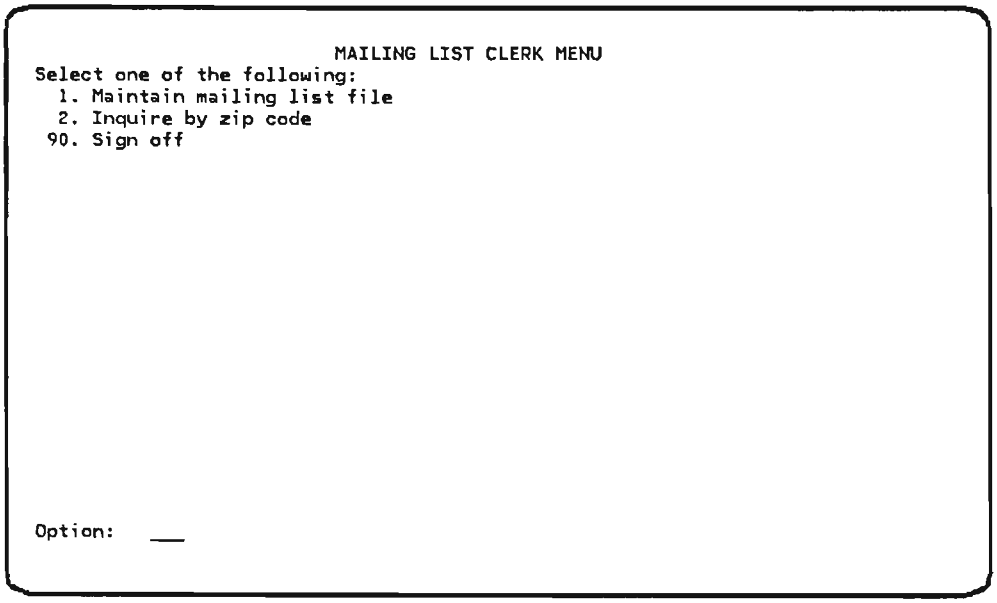
- A member named MLG035CD in the QDDSSRC source file.

- A display file named MLG035CD.

- A member named MLG035C in the QCLSRC source file.

- A CL program named MLG035C.

The resulting menu is similar to the menu that was defined earlier in this chapter by separately coding the DDS for the display file and the CL program.



```
                           MAILING LIST CLERK MENU
Select one of the following:
  1. Maintain mailing list file
  2. Inquire by zip code
 90. Sign off






















Option:  ___
```

Note that the option field is at the bottom of the screen. The SDA process for creating menus places the option field on the second to bottom line of the screen. If you want the option field to be higher, you can change its position by one of the following methods:

- Use SEU to change the DDS source member MLG035CD in the file QDDSSRC and then use option 3 on the programmer menu to re-create the display file.

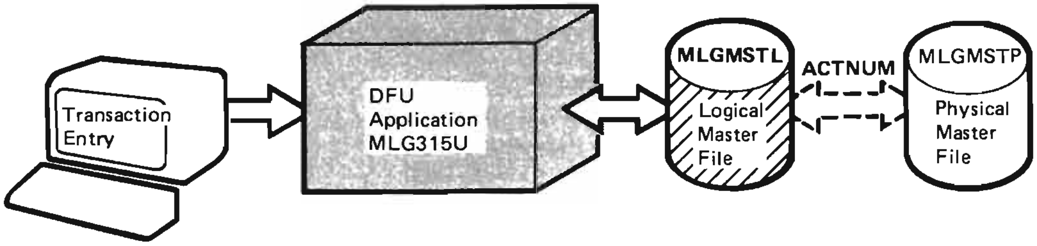
- Use option 1 on the SDA option menu to change the display record format for the menu and then re-create the display file (for details, see the *SDA Reference Manual and User's Guide*).

When you re-create the display file, you must specify LVLCHK(*NO) to avoid an error condition in the creation process.

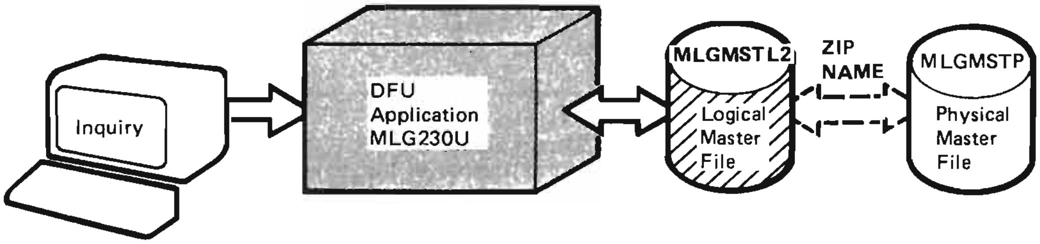
USING MULTIPLE ACCESS PATHS

A common requirement in a mailing list application is to determine an account number when only the name and address associated with the account are known. An inquiry into the mailing list by customer name to find the account number may not be practical if the list is large; however, an inquiry into the mailing list by another field (such as zip code) may be more practical.

By using logical files, data can be accessed in various sequences as described in Chapter 4. This method of retrieving data is similar to sorting on other systems where the data is processed sequentially in the desired sequence. In Chapter 7 and earlier in this chapter, the mailing list file (MLGMSTP) was updated by accessing records by account number. This method is similar to using a keyed index to a file on other systems. The logical file MLGMSTL defined the access path on account number.



Now a second access path will be created for accessing the same mailing list file on name and zip code instead of account number. This access path will be defined by the logical file MLGMSTL2.



Both access paths to one file are immediately maintained. An access path has *immediate maintenance* if it is updated regardless of whether the file is open. For example, a record can be added by one work station user and all other users can immediately access that record on the same or a different access path.

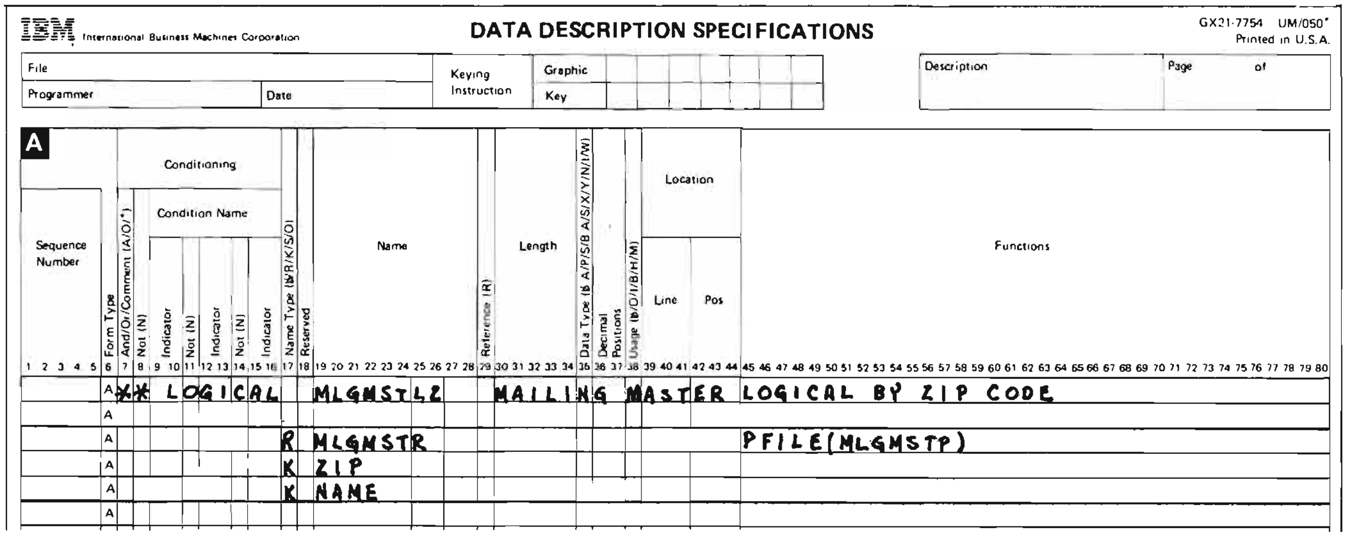
To create and use an access path on both *zip code and name*, the following must be created:

- A logical file, MLGMSTL2, that has key fields for zip code and name
- A DFU application, MLG230U, that inquires into the file by zip code and name

Because the methods for creating source members, files, and programs, and for entering source are shown in detail in previous chapters, these methods are not repeated in the remainder of this publication. Any data that you need to enter is discussed; however, the displays on which you enter this data are shown only where needed for the discussion.

Creating the Logical File

A source member named MLGMSTL2 is created by selecting option 8 and specifying MLGMSTL2 for source member and LF for type on the programmer menu. You can also enter a text description for this source member. The DDS for MLGMSTL2 is shown in Figure 8-2.



IBM International Business Machines Corporation

DATA DESCRIPTION SPECIFICATIONS

GX21-7754 UM/050*
Printed in U.S.A.

File
Programmer
Date
Keying Instruction
Graphic
Key
Description
Page of

```
     A** LOGICAL  MLGMSTL2   MAILING MASTER LOGICAL BY ZIP CODE
     A
     A          R MLGMSTR                   PFILE(MLGMSTP)
     A          K ZIP
     A          K NAME
     A
```

A
Conditioning
Condition Name
Sequence Number
Form Type
And/Or/Comment (A/O/*)
Not (N)
Indicator
Not (N)
Indicator
Not (N)
Indicator
Name Type (b/R/K/S/O)
Reserved
Name
Reference (R)
Length
Data Type (b A/P/S/B A/S/X/Y/N/I/W)
Decimal Positions
Usage (b/O/I/B/H/M)
Location
Line
Pos
Functions
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80

Figure 8-2. DDS ⁴or MLGMSTL2 File

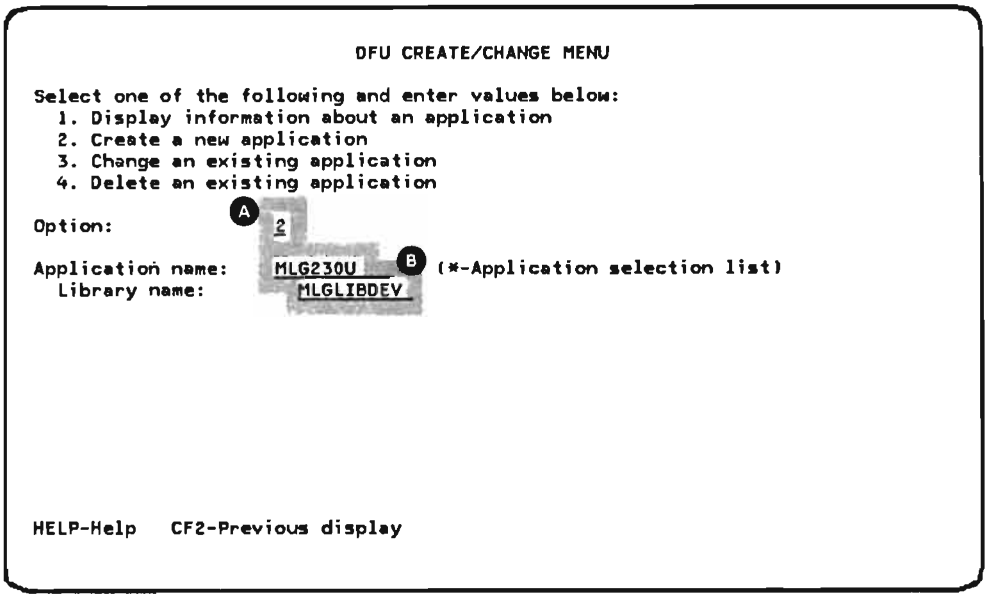
The DDS for logical file MLGMSTL2 are similar to those for logical files that were created previously. The physical file is named by the PFILE keyword. The existing format is used (MLGMSTR). Two key fields are specified (ZIP and NAME). The sequence in which the fields are written determines the order of the access path. In this case, ZIP is the high-order key field so that the access path is arranged by zip code and by name within each zip code.

The logical file MLGMSTL2 is then created by selecting option 3 and specifying LF for type on the programmer menu. The access path is built and immediately maintained. This means that any additions, deletions, or changes to names or zip codes are immediately reflected on the name and zip code access path.

Creating the DFU Application

The programmer menu is used to invoke DFU to enter the definition for the DFU application MLG230U. On the programmer menu, select option 1 and specify MLG230U for the source member. Also, select option 1 on the DFU menu when it is displayed.

After the DFU menu, you receive the create/change menu:

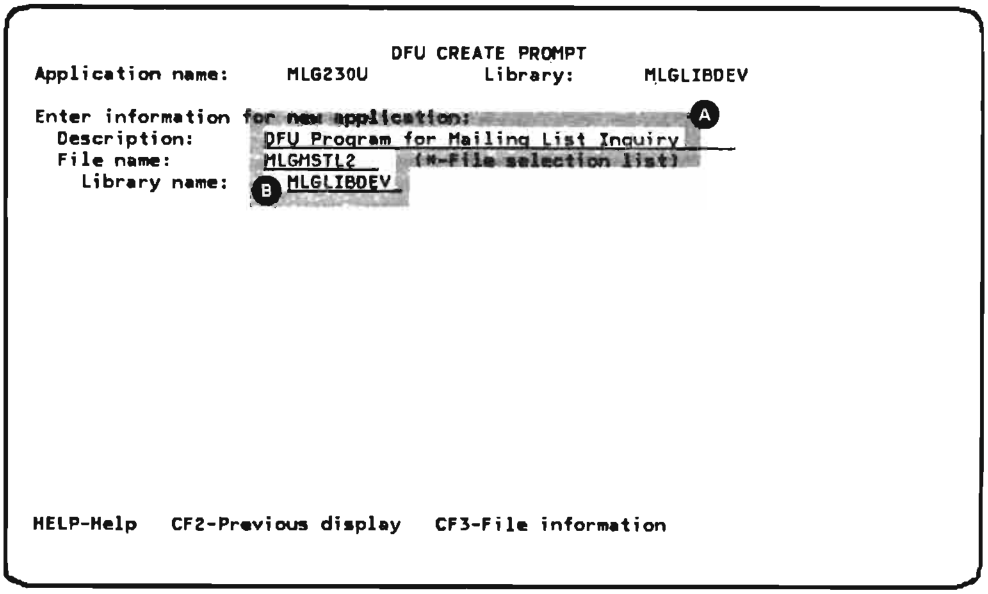


```
                              DFU CREATE/CHANGE MENU

Select one of the following and enter values below:
  1. Display information about an application
  2. Create a new application
  3. Change an existing application
  4. Delete an existing application

Option:               2

Application name:     MLG230U     (*-Application selection list)
  Library name:         MLGLIBDEV





HELP-Help   CF2-Previous display
```

A Select option 2 to create an application.

B The application name and object library name that you specified on the programmer menu are shown here.

You then receive the DFU create prompt:

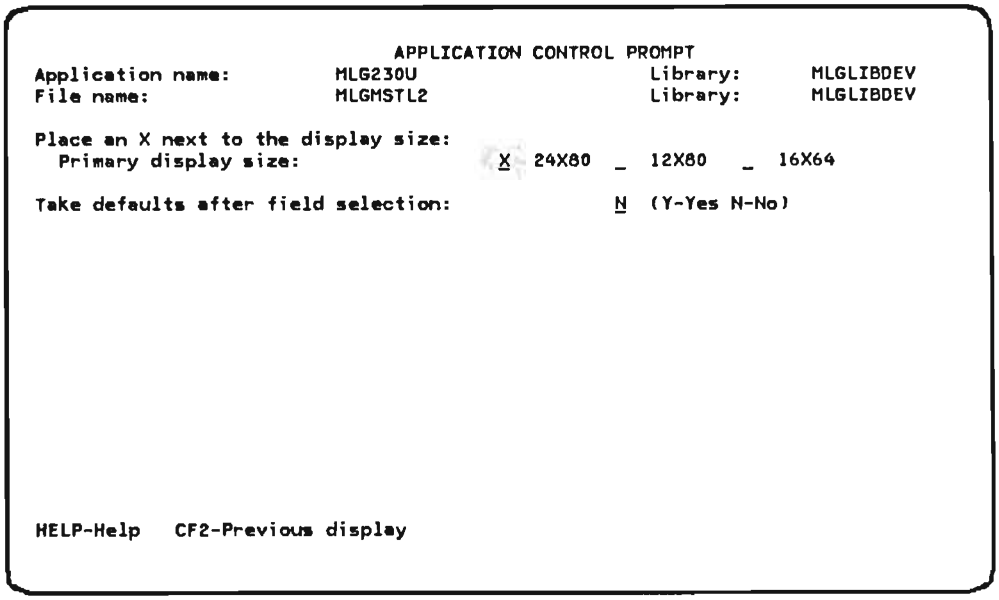


```
                                DFU CREATE PROMPT
Application name:     MLG230U          Library:      MLGLIBDEV

Enter information for new application:
  Description:        DFU Program for Mailing List Inquiry
  File name:          MLGMSTL2     (*-File selection list)
    Library name:      MLGLIBDEV





HELP-Help   CF2-Previous display   CF3-File information
```

A Key in the text that describes the DFU program.

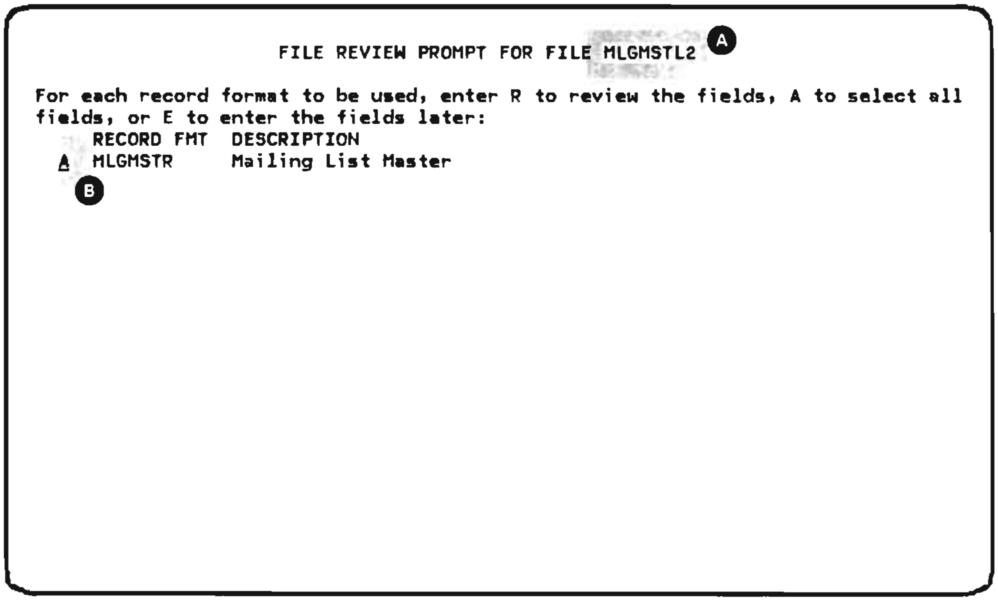
B Key in the name of the file that defines the access path and the library where the file is located.

The application control prompt appears next. You need only verify that the display size is 24 x 80:



```
                         APPLICATION CONTROL PROMPT
Application name:        MLG230U                   Library:     MLGLIBDEV
File name:               MLGMSTL2                  Library:     MLGLIBDEV

Place an X next to the display size:
  Primary display size:            X  24X80  _  12X80  _  16X64

Take defaults after field selection:          N  (Y-Yes N-No)










HELP-Help   CF2-Previous display
```

You then receive the file review prompt:

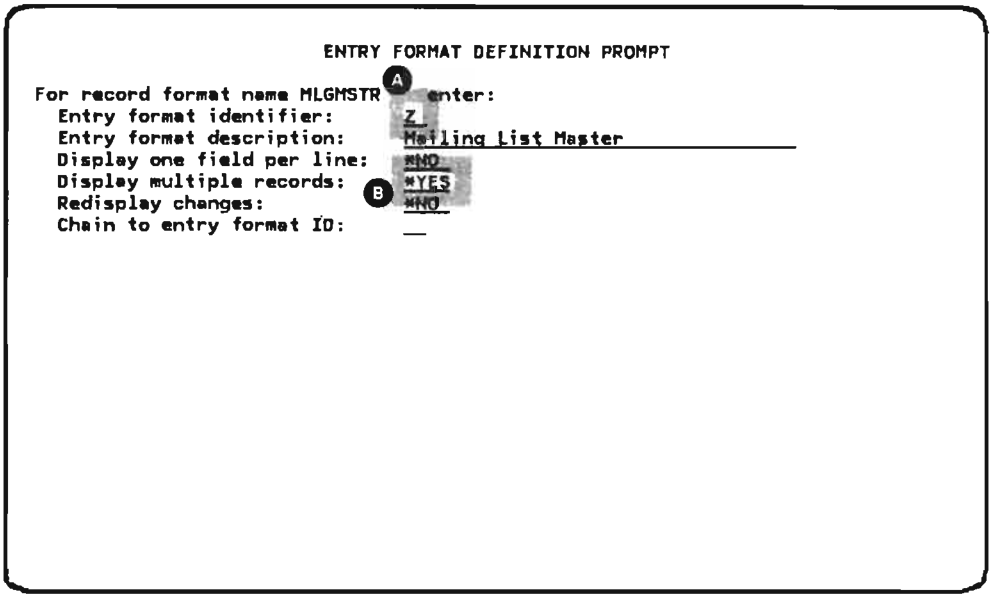


```
                     FILE REVIEW PROMPT FOR FILE MLGMSTL2

For each record format to be used, enter R to review the fields, A to select all
fields, or E to enter the fields later:
    RECORD FMT  DESCRIPTION
  A MLGMSTR     Mailing List Master
```

A This is the file name you specified on the create prompt.

B Key in A to select all fields.

The next prompt is for entry format definition:

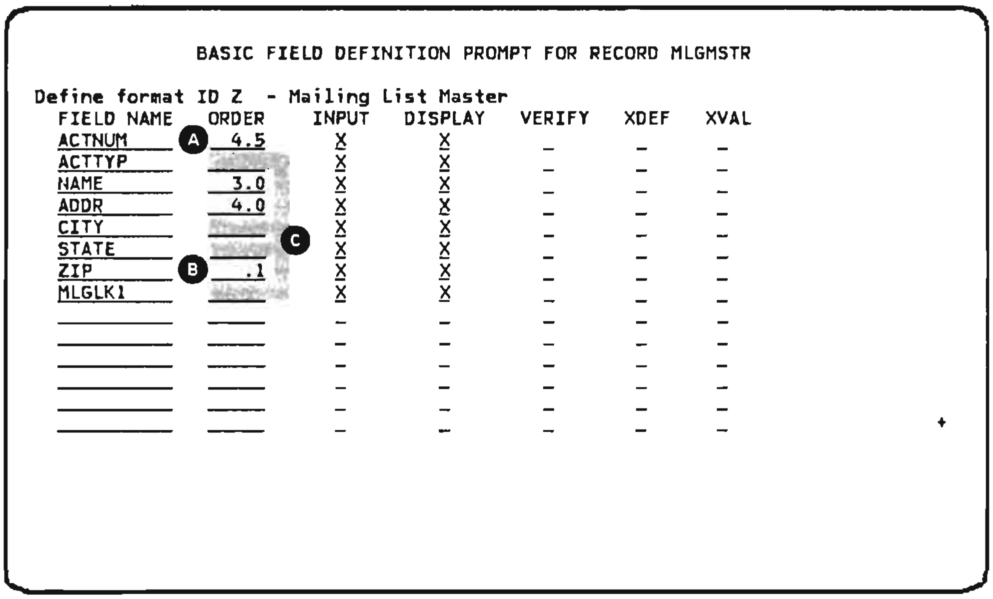


```
                    ENTRY FORMAT DEFINITION PROMPT

For record format name MLGMSTR   enter:
  Entry format identifier:       Z_
  Entry format description:      Mailing List Master________________
  Display one field per line:    *NO_
  Display multiple records:      *YES
  Redisplay changes:             *NO_
  Chain to entry format ID:      __
```

A Key in Z (for zip code).

B Key in *YES (*NO is the default) so DFU will place as many records as possible on the execution display when the Roll Up (↑) and Roll Down (↓) keys are pressed.

You then receive the basic field definition prompt:



```
              BASIC FIELD DEFINITION PROMPT FOR RECORD MLGMSTR

Define format ID Z  - Mailing List Master
  FIELD NAME  ORDER    INPUT   DISPLAY   VERIFY   XDEF   XVAL
  ACTNUM____   4.5       X        X        _        _      _
  ACTTYP____   _____     X        X        _        _      _
  NAME______   3.0       X        X        _        _      _
  ADDR______   4.0       X        X        _        _      _
  CITY______   _____     X        X        _        _      _
  STATE_____   _____     X        X        _        _      _
  ZIP_______     .1      X        X        _        _      _
  MLGLK1____   _____     X        X        _        _      _
  __________   _____     _        _        _        _      _
  __________   _____     _        _        _        _      _
  __________   _____     _        _        _        _      _
  __________   _____     _        _        _        _      _
  __________   _____     _        _        _        _      _
  __________   _____     _        _        _        _      _        +
```

A Change the order value for ACTNUM from 1.0 to 4.5.

B Change the order value for ZIP from 7.0 to .1.

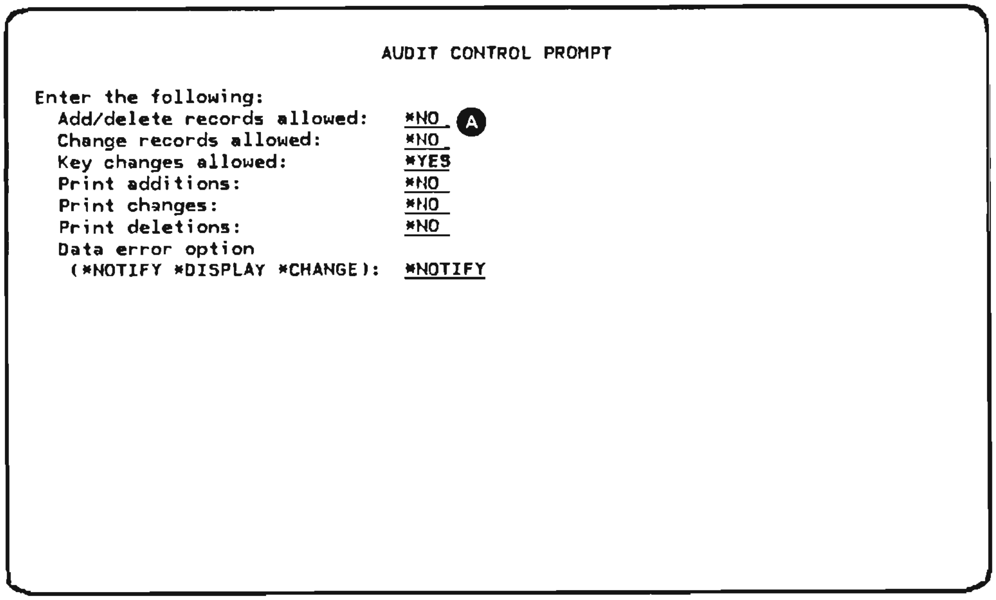
C Key blanks into these positions.

This prompt allows you to design the display layout such that each record has only one line on the display. The width of each column on the display is implicitly determined by the width of the field, editing requested, width of the column head as specified in the DDS (COLHDG keyword), and spaces between the fields defined by DFU.

Determine which fields the work station user needs to identify an account. In this case, the ACTNUM, NAME, ADDR, and ZIP fields need to be displayed. To specify which fields will appear and the order in which they will appear, change the ORDER column on the display as indicated.

The INPUT column already has an X in it for each field. These Xs will be ignored when the audit control prompt is entered later so that the data is displayed only (the data cannot be changed).

After the basic field definition prompt is entered, the entry format definition prompt appears again. Because no additional formatting needs to be done, the defaults that automatically appear are used and no information needs to be keyed in. The audit control prompt is displayed next:



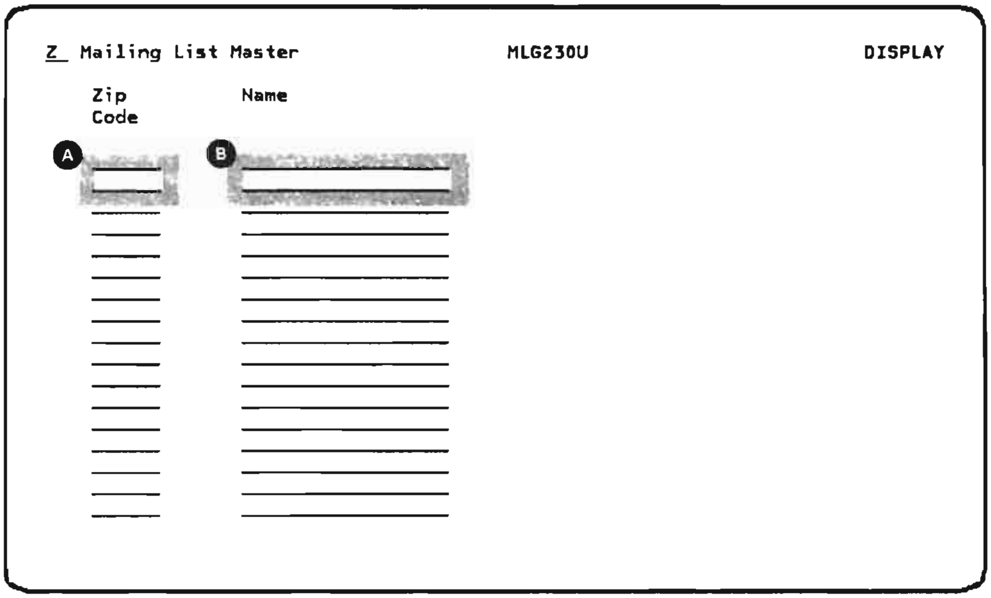
```
                               AUDIT CONTROL PROMPT

 Enter the following:
   Add/delete records allowed:   *NO
   Change records allowed:       *NO
   Key changes allowed:          *YES
   Print additions:              *NO
   Print changes:                *NO
   Print deletions:              *NO
   Data error option
    (*NOTIFY *DISPLAY *CHANGE):  *NOTIFY
```

A Key in *NO because no changes should be made to the file (*YES is the default).

After the audit control prompt is entered, the exit application definition menu appears. Enter option 5 to create the DFU application. You then receive the application creation prompt. Accept the defaults as shown and press the Enter key to complete the creation of MLG230U.

Executing the DFU Application

The DFU application can be tested from the programmer menu by selecting option 5 and entering the command DSPDTA APP(MLG230U). The following prompt is displayed:



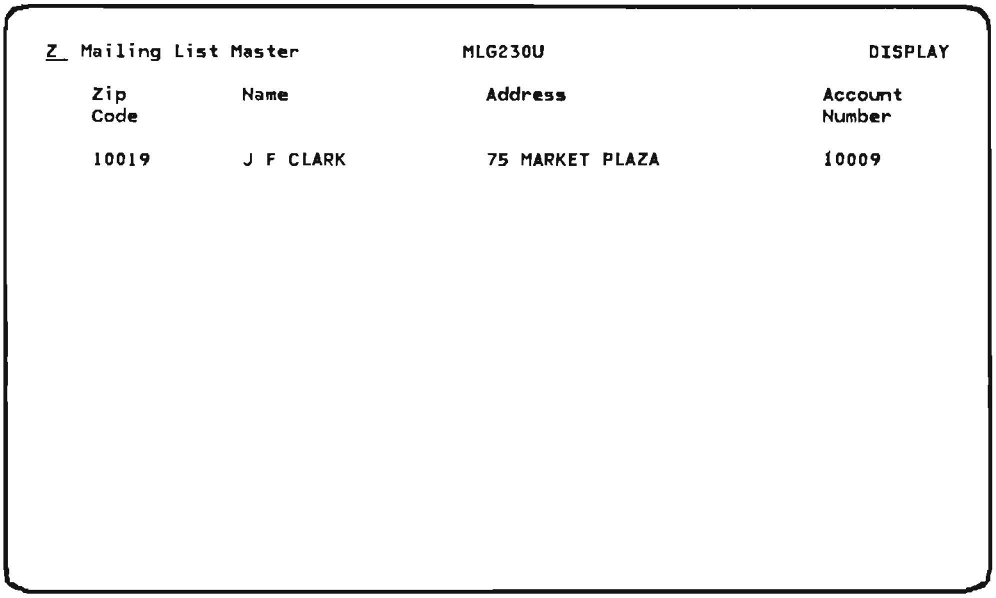
```
Z_ Mailing List Master                    MLG230U                    DISPLAY

   Zip          Name
   Code
```

Key in the zip code A and the name B of the record to be displayed; for example:

A 10019

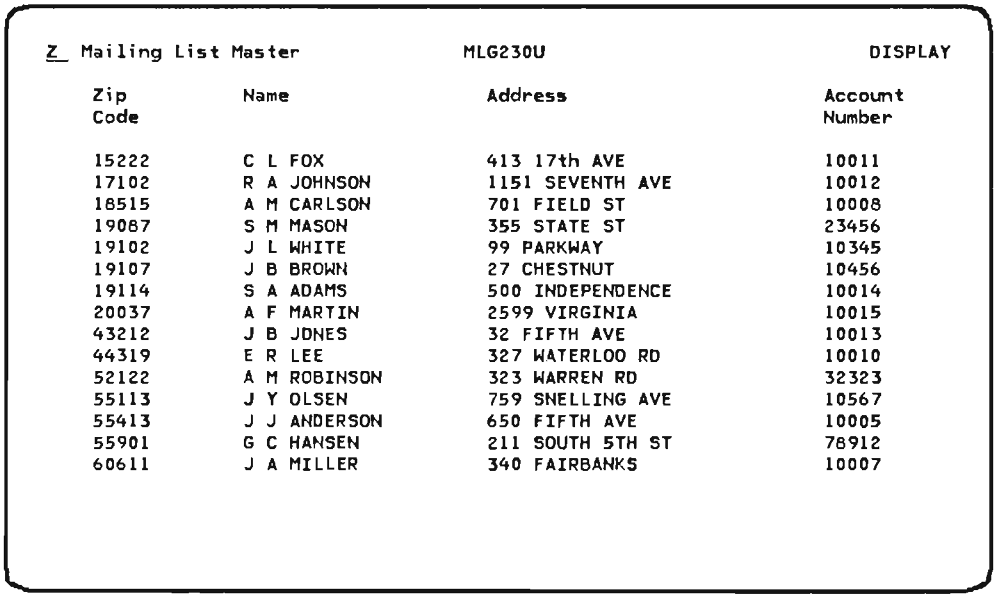
B J F Clark

Next, the master record for the name and zip code specified in the previous prompt is displayed. The zip code and name fields were used to access the first record that was equal to or greater than the record just entered. The format of the display for this record was defined on the basic field definition prompt; the display looks like this:



```
Z_ Mailing List Master              MLG230U                            DISPLAY

     Zip          Name                 Address                        Account
     Code                                                             Number

     10019        J F CLARK            75 MARKET PLAZA                10009
```

The work station user can use the Roll Up (↑) and Roll Down (↓) keys to look at the data using the name and zip code access path. Records are read sequentially using the access path to fill the display. If you press the Roll Up key, the following display appears:



```
Z_ Mailing List Master              MLG230U                            DISPLAY

     Zip          Name                 Address                        Account
     Code                                                             Number

     15222        C L FOX              413 17th AVE                   10011
     17102        R A JOHNSON          1151 SEVENTH AVE               10012
     18515        A M CARLSON          701 FIELD ST                   10008
     19087        S M MASON            355 STATE ST                   23456
     19102        J L WHITE            99 PARKWAY                     10345
     19107        J B BROWN            27 CHESTNUT                    10456
     19114        S A ADAMS            500 INDEPENDENCE               10014
     20037        A F MARTIN           2599 VIRGINIA                  10015
     43212        J B JONES            32 FIFTH AVE                   10013
     44319        E R LEE              327 WATERLOO RD                10010
     52122        A M ROBINSON         323 WARREN RD                  32323
     55113        J Y OLSEN            759 SNELLING AVE               10567
     55413        J J ANDERSON         650 FIFTH AVE                  10005
     55901        G C HANSEN           211 SOUTH 5TH ST               78912
     60611        J A MILLER           340 FAIRBANKS                  10007
```

By pressing the Enter key, the work station user can return to the initial application display and enter a new zip code and name to begin inquiring at a different point on the access path. When the inquiry is completed, the work station user can press the Exit Application key (CF1) to end the DFU application.

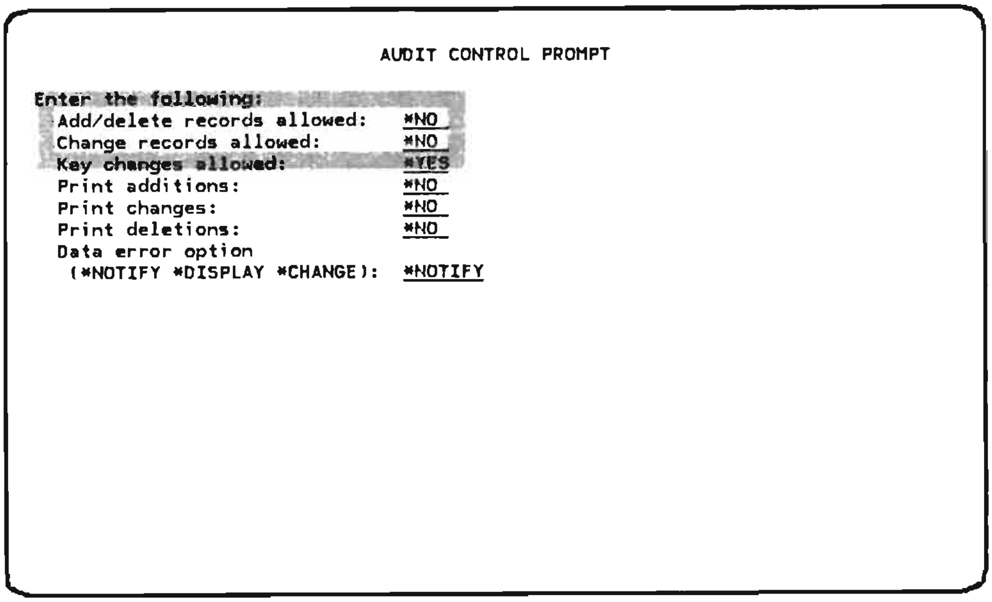
Programming Considerations

- The work station user could request the DFU application MLG230U from an application-oriented menu like the one shown earlier in this chapter in association with CL program MLG035C.

- The work station user could also request the application MLG230U by calling it from the program call menu. A CL program similar to MLG315C, discussed earlier in this chapter, is required.

- The Display Data (DSPDTA) command can be executed for any DFU application. For example, if DSPDTA is used to execute MLG315U, records are displayed and no changes can be made to them.

- The Change Data (CHGDTA) command can be specified for the DFU application MLG230U; however, changes would not be applied to any master record because the following entries were made on the audit control prompt:



```
                              AUDIT CONTROL PROMPT

Enter the following:
  Add/delete records allowed:    *NO
  Change records allowed:        *NO
  Key changes allowed:           *YES
  Print additions:               *NO
  Print changes:                 *NO
  Print deletions:               *NO
  Data error option
   (*NOTIFY *DISPLAY *CHANGE):   *NOTIFY
```

USING THE QUERY UTILITY

This section discusses how a query can be made against the mailing list file, as shown below. Suppose that you need to know what business or school accounts (the value of ACTTYP is 1 or 4) there are in either Pennsylvania or New Jersey (the value of STATE is PA or NJ). You need a printed report of the zip codes used in Pennsylvania and New Jersey with a line for each record. The report should be sequenced on account type, state, and zip code.

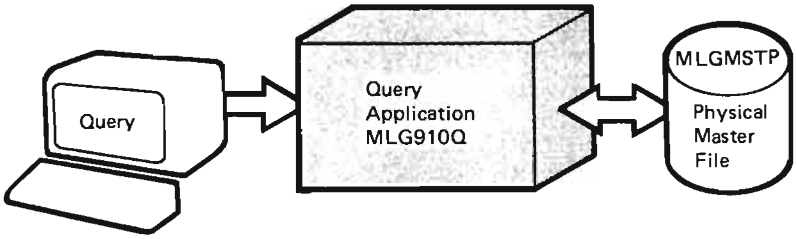
Because this sequence is different from any of the access paths that already exist for this application, the physical file will be queried (no key fields are defined). The records are selected in arrival sequence and then sorted for the desired sequence.

The query utility is the part of IDU that is used to extract, from a file, one or more records based upon a criterion. A query application is created by responding to a series of prompts (similar to those to which you respond to create a DFU application). The query utility predominantly performs a subset of functions that can be done using RPG III and logical files.

A query is different from an inquiry using DFU in the following ways:

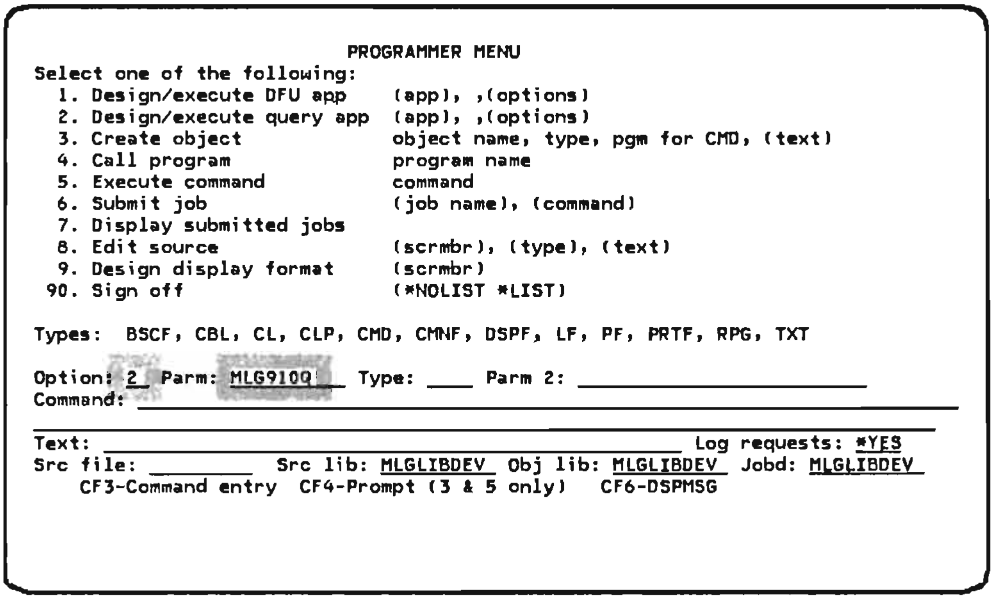
- Query can analyze records based on a specified criterion; whereas, a DFU inquiry cannot.
- Query can prepare a printed report; whereas, a DFU inquiry normally does not.

The query utility can be used by those who are unfamiliar with DP terminology and coding. Those who are familiar with DP terminology and coding can use query as an alternative to certain application coding.



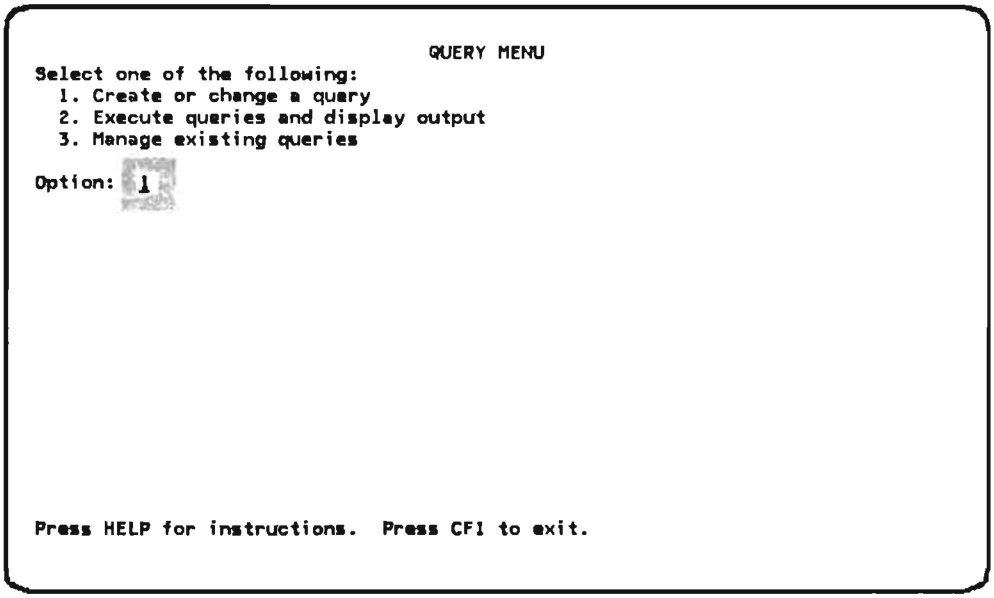
Creating the Query Application

To create a query application, select option 2 on the programmer menu as follows:



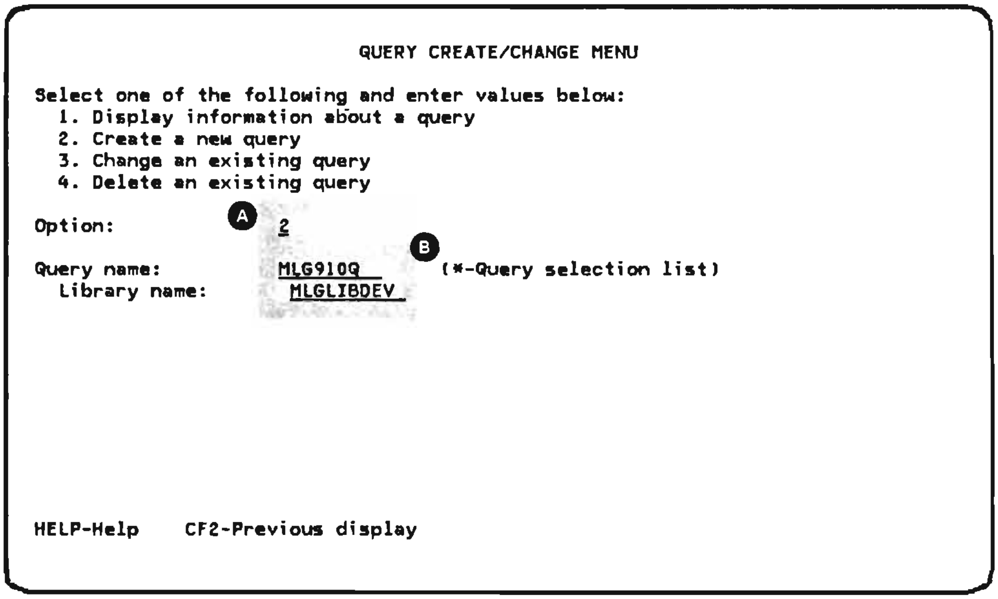
```
                              PROGRAMMER MENU
Select one of the following:
  1. Design/execute DFU app    (app), ,(options)
  2. Design/execute query app  (app), ,(options)
  3. Create object             object name, type, pgm for CMD, (text)
  4. Call program              program name
  5. Execute command           command
  6. Submit job                (job name), (command)
  7. Display submitted jobs
  8. Edit source               (scrmbr), (type), (text)
  9. Design display format     (scrmbr)
 90. Sign off                  (*NOLIST *LIST)

Types:  BSCF, CBL, CL, CLP, CMD, CMNF, DSPF, LF, PF, PRTF, RPG, TXT

Option: 2  Parm: MLG910Q    Type: ____  Parm 2: _________________________
Command: __________________________________________________________________
___________________________________________________________________________
Text: __________________________________________________ Log requests: *YES
Src file: _________  Src lib: MLGLIBDEV  Obj lib: MLGLIBDEV  Jobd: MLGLIBDEV
    CF3-Command entry  CF4-Prompt (3 & 5 only)   CF6-DSPMSG
```

You receive a series of displays on which you define the query. The first display is the query menu. Select option 1 on this menu:



```
                                    QUERY MENU
Select one of the following:
  1. Create or change a query
  2. Execute queries and display output
  3. Manage existing queries

Option: 1











Press HELP for instructions.  Press CF1 to exit.
```

You then receive the query create/change menu:

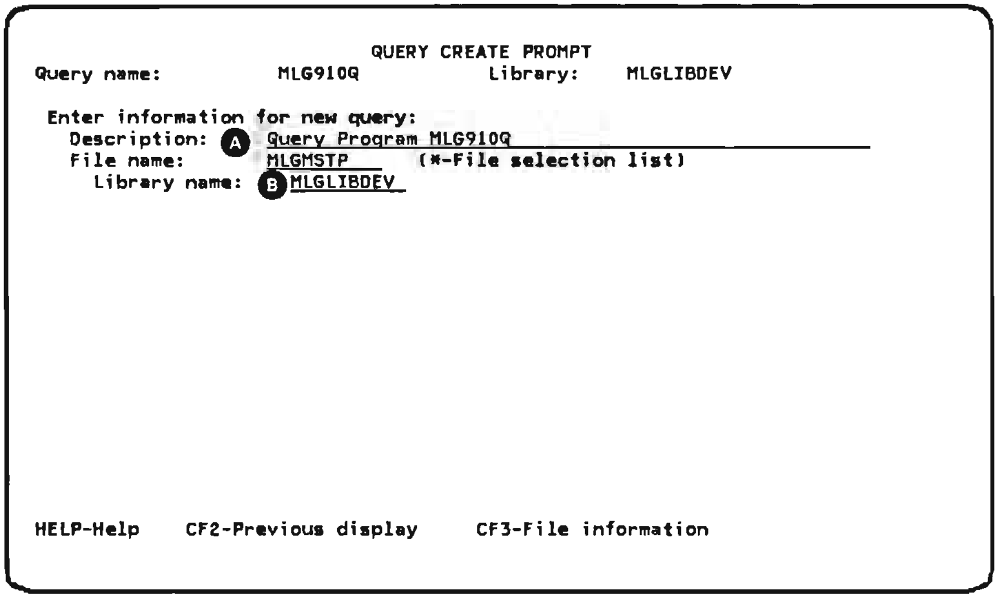


```
                         QUERY CREATE/CHANGE MENU

Select one of the following and enter values below:
  1. Display information about a query
  2. Create a new query
  3. Change an existing query
  4. Delete an existing query

Option:              2

Query name:          MLG910Q      (*-Query selection list)
  Library name:       MLGLIBDEV




HELP-Help    CF2-Previous display
```

A Select option 2.

B The query name and object library that you specified on the programmer menu are shown here.

The query create prompt appears next:

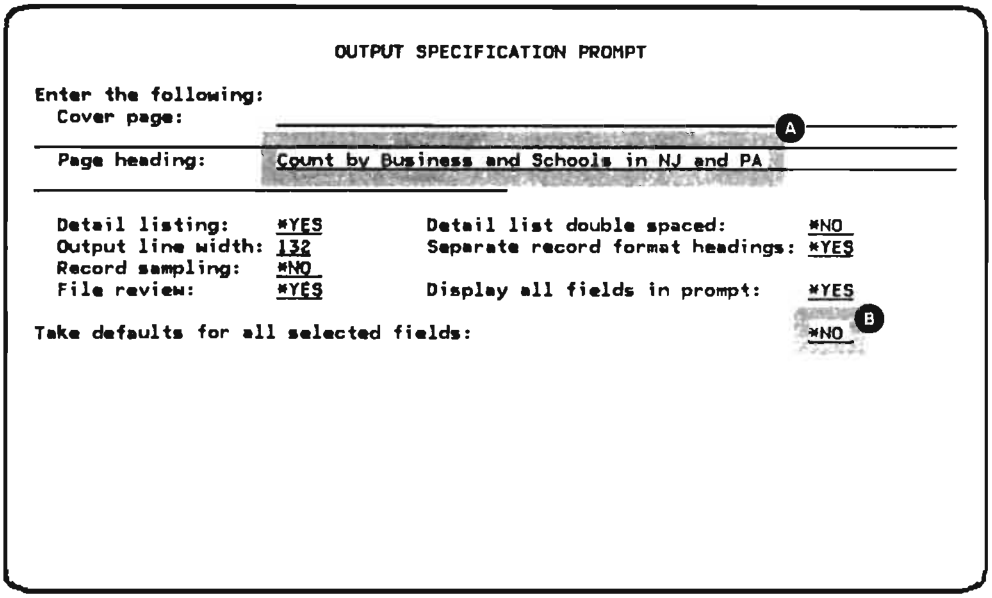


```
                          QUERY CREATE PROMPT
Query name:          MLG910Q          Library:    MLGLIBDEV

 Enter information for new query:
   Description:      Query Program MLG910Q
   File name:        MLGMSTP     (*-File selection list)
     Library name:    MLGLIBDEV




HELP-Help    CF2-Previous display    CF3-File information
```

A Key in the text description of the query.

B Key in the name of the file to be queried and the library where the file is located.

You then receive the output specification prompt, so that you can specify the headings for your report:

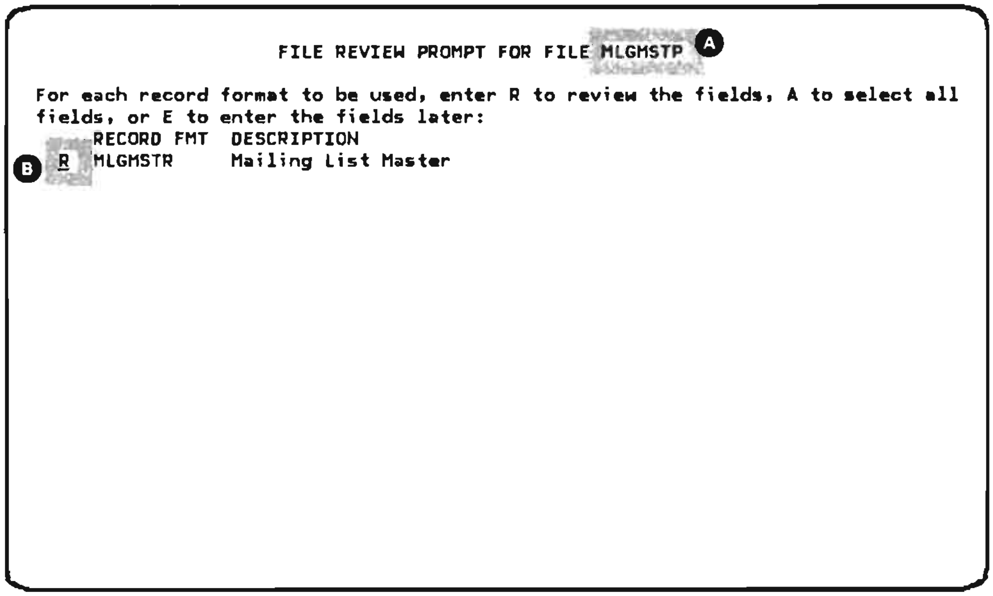


```
                         OUTPUT SPECIFICATION PROMPT

Enter the following:
  Cover page:

  Page heading:      Count by Business and Schools in NJ and PA


  Detail listing:    *YES          Detail list double spaced:       *NO
  Output line width: 132           Separate record format headings: *YES
  Record sampling:   *NO
  File review:       *YES          Display all fields in prompt:    *YES

Take defaults for all selected fields:                              *NO
```

A Key in the page heading.

B Do not change the *NO shown here because you need to modify field definitions for your query.

The file review prompt is displayed next:

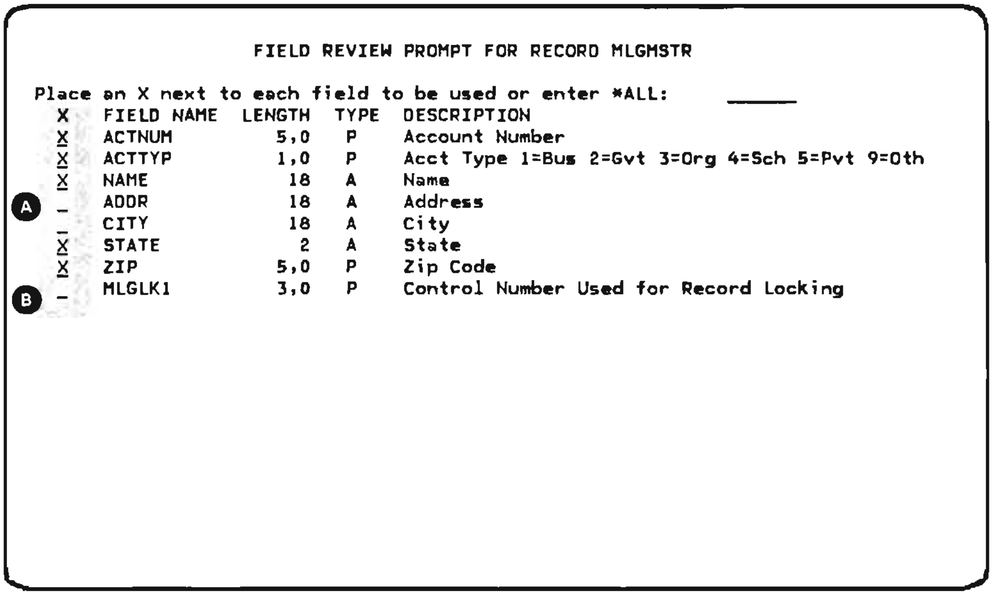


```
                     FILE REVIEW PROMPT FOR FILE MLGMSTP

For each record format to be used, enter R to review the fields, A to select all
fields, or E to enter the fields later:
      RECORD FMT  DESCRIPTION
    R MLGMSTR     Mailing List Master
```

A This is the file name you specified on the query create prompt.

B Key in R to review all fields.

The field review prompt is displayed after the file review prompt:

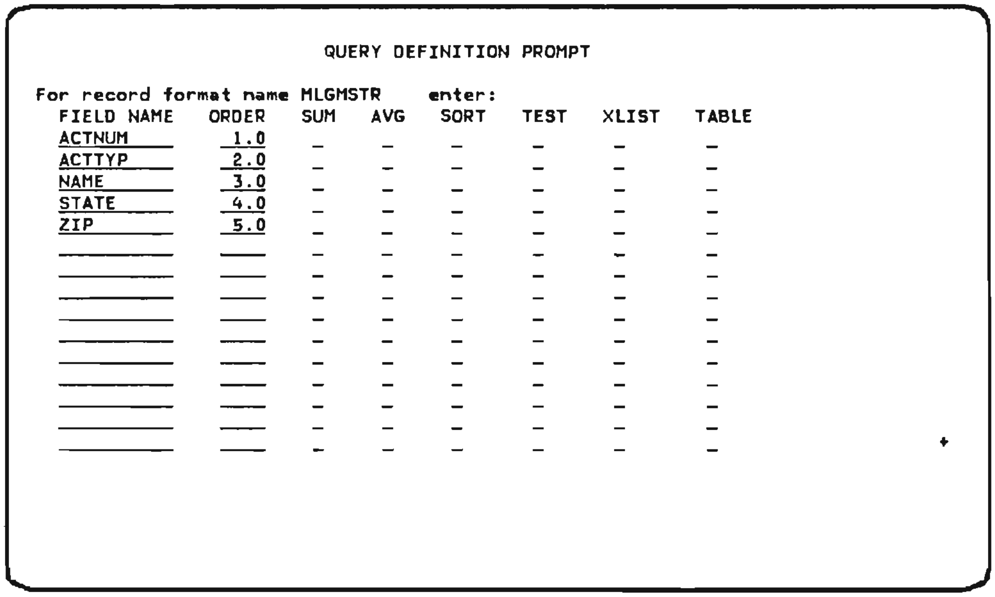


```
                     FIELD REVIEW PROMPT FOR RECORD MLGMSTR

Place an X next to each field to be used or enter *ALL:     ______
   X   FIELD NAME  LENGTH  TYPE  DESCRIPTION
   X   ACTNUM        5,0    P    Account Number
   X   ACTTYP        1,0    P    Acct Type 1=Bus 2=Gvt 3=Org 4=Sch 5=Pvt 9=Oth
   X   NAME           18    A    Name
   _   ADDR           18    A    Address
   _   CITY           18    A    City
   X   STATE           2    A    State
   X   ZIP           5,0    P    Zip Code
   _   MLGLK1        3,0    P    Control Number Used for Record Locking
```

A Key in X next to the field(s) you want to be in the query output.

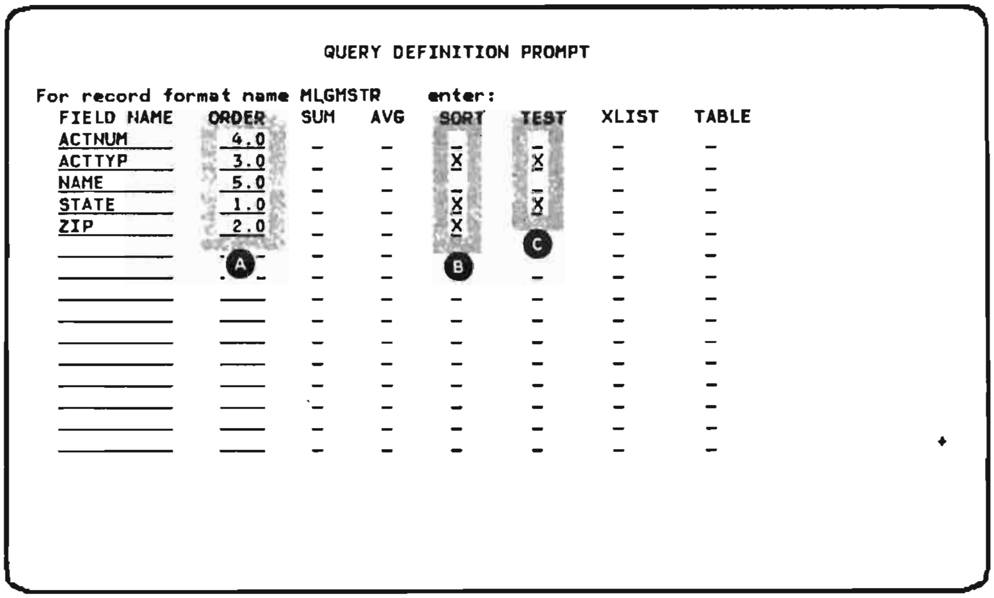
B Do not key an X next to MLGLK1 (this field will be used in a later approach).

You then receive the query definition prompt, which shows the fields you selected on the previous prompt:



```
                          QUERY DEFINITION PROMPT

For record format name MLGMSTR     enter:
   FIELD NAME   ORDER   SUM   AVG   SORT   TEST   XLIST   TABLE
   ACTNUM        1.0     _     _     _      _      _       _
   ACTTYP        2.0     _     _     _      _      _       _
   NAME          3.0     _     _     _      _      _       _
   STATE         4.0     _     _     _      _      _       _
   ZIP           5.0     _     _     _      _      _       _
   __________   _____    _     _     _      _      _       _
   __________   _____    _     _     _      _      _       _
   __________   _____    _     _     _      _      _       _
   __________   _____    _     _     _      _      _       _
   __________   _____    _     _     _      _      _       _
   __________   _____    _     _     _      _      _       _
   __________   _____    _     _     _      _      _       _
   __________   _____    _     _     _      _      _       _
   __________   _____    _     _     _      _      _       _
   __________   _____    _     _     _      _      _       _          +
```

For this mailing list application, the query definition prompt should look like this:



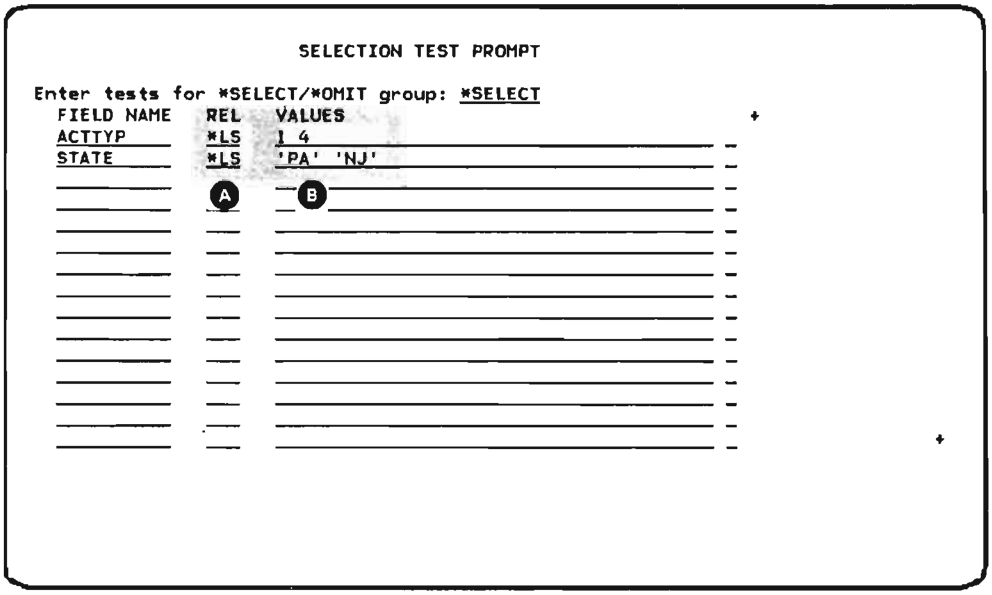
```
                          QUERY DEFINITION PROMPT

For record format name MLGMSTR    enter:
  FIELD NAME   ORDER  SUM   AVG   SORT   TEST   XLIST   TABLE
  ACTNUM____   _4.0    _     _     _      _      _       _
  ACTTYP____   _3.0    _     _     X      X      _       _
  NAME______   _5.0    _     _     _      _      _       _
  STATE_____   _1.0    _     _     X      X      _       _
  ZIP_______   _2.0    _     _     X      _      _       _
  __________   ____    _     _     _      _      _       _
  __________   ____    _     _     _      _      _       _
  __________   ____    _     _     _      _      _       _
  __________   ____    _     _     _      _      _       _
  __________   ____    _     _     _      _      _       _
  __________   ____    _     _     _      _      _       _
  __________   ____    _     _     _      _      _       _
  __________   ____    _     _     _      _      _       _
  __________   ____    _     _     _      _      _       _
  __________   ____    _     _     _      _      _       _
```

A The ORDER column specifies the sequence (from left to right) that the fields will have on the query output. Change the default values so that the fields will have the following sequence: STATE ZIP ACTTYP ACTNUM NAME.

B Key in X to specify which fields are to be sorted.

C Key in X to specify which fields have test criteria.

After the query definition prompt is completed, the selection test prompt is displayed for ACTTYP and STATE, which were selected on the previous prompt:



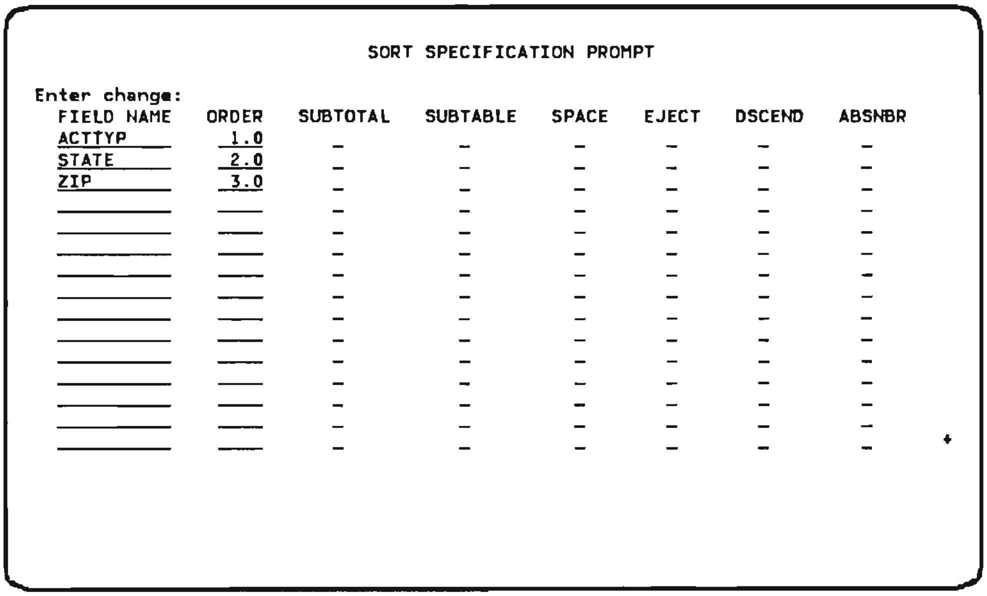
```
                       SELECTION TEST PROMPT

 Enter tests for *SELECT/*OMIT group: *SELECT
   FIELD NAME   REL   VALUES                                  +
   ACTTYP       *LS   1 4
   STATE        *LS   'PA' 'NJ'
```

A The REL column specifies the type of relational test you want performed on the field. Key in *LS (for list type).

B The VALUES column specifies the values to be tested. Key in 1 4 (for ACTTYP) and 'PA' 'NJ' (for STATE). PA NJ must be keyed in uppercase; otherwise, the test will be for the lowercase values, pa nj. All character values such as PA and NJ must be quoted (enclosed in apostrophes).

The selection test prompt is displayed again to allow you to specify multiple select/omit groups. Because no additional testing needs to be done, no information needs to be entered.

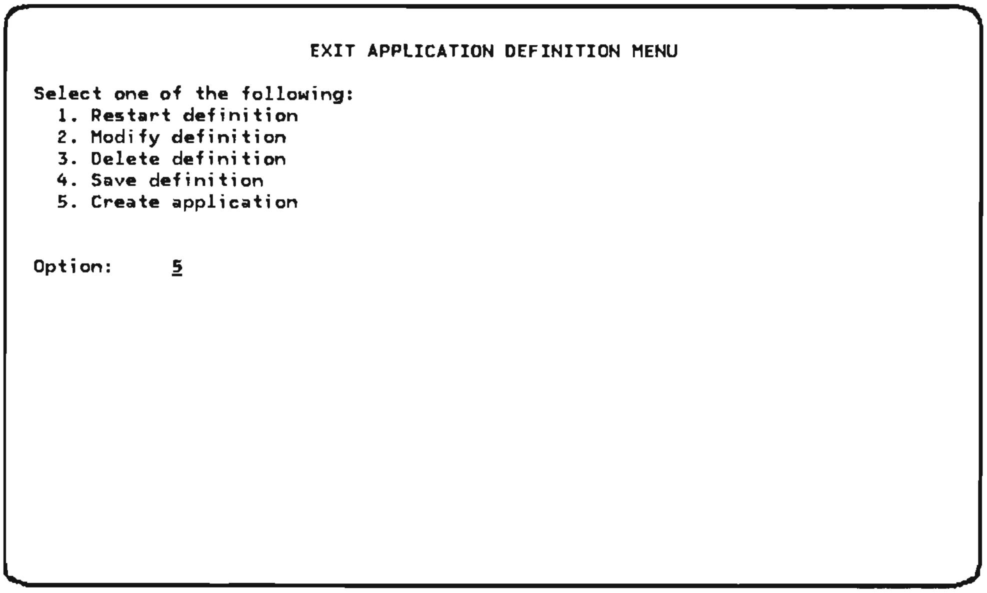
The sort specification prompt is displayed next:



```
                         SORT SPECIFICATION PROMPT

Enter change:
  FIELD NAME  ORDER   SUBTOTAL   SUBTABLE   SPACE   EJECT   DSCEND   ABSNBR
  ACTTYP____  __1.0      _          _         _       _       _        _
  STATE_____  __2.0      _          _         _       _       _        _
  ZIP_______  __3.0      _          _         _       _       _        _
  __________  _____      _          _         _       _       _        _
  __________  _____      _          _         _       _       _        _
  __________  _____      _          _         _       _       _        _
  __________  _____      _          _         _       _       _        _
  __________  _____      _          _         _       _       _        _
  __________  _____      _          _         _       _       _        _
  __________  _____      _          _         _       _       _        _
  __________  _____      _          _         _       _       _        _
  __________  _____      _          _         _       _       _        _
  __________  _____      _          _         _       _       _        _
  __________  _____      _          _         _       _       _        _
  __________  _____      _          _         _       _       _        _   +
```

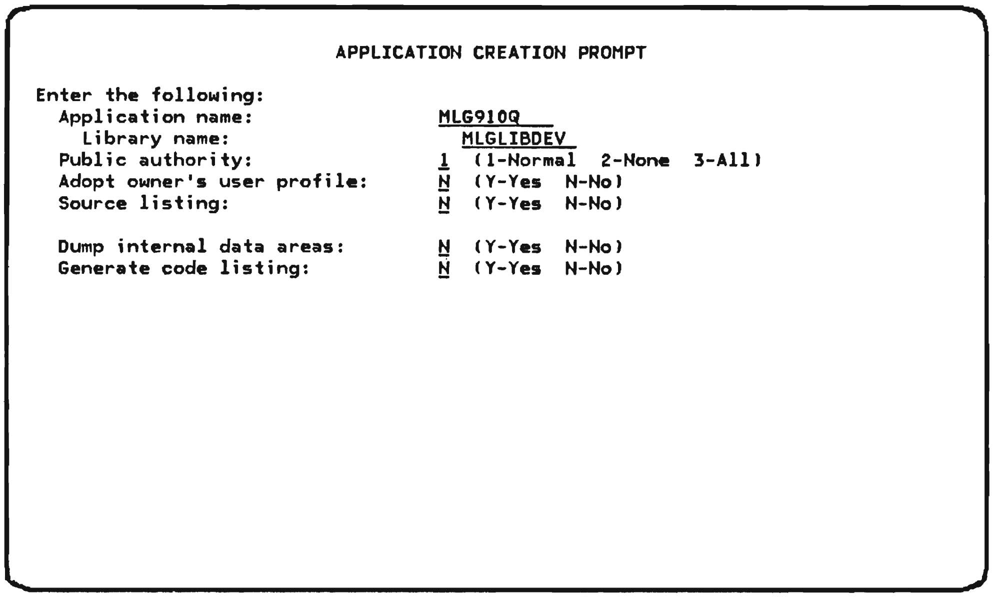
The default values in the ORDER column are correct for this query.

The exit application definition menu for a permanent application is displayed next. Select option 5 to create the query application:



```
                    EXIT APPLICATION DEFINITION MENU

Select one of the following:
  1. Restart definition
  2. Modify definition
  3. Delete definition
  4. Save definition
  5. Create application


Option:     5
```

Because you indicated that you want to create an application, you next receive the application creation prompt:



```
                          APPLICATION CREATION PROMPT

Enter the following:
  Application name:               MLG910Q
    Library name:                   MLGLIBDEV
  Public authority:               1  (1-Normal  2-None  3-All)
  Adopt owner's user profile:     N  (Y-Yes  N-No)
  Source listing:                 N  (Y-Yes  N-No)

  Dump internal data areas:       N  (Y-Yes  N-No)
  Generate code listing:          N  (Y-Yes  N-No)
```

In this example, the defaults shown on the prompt are all acceptable, so you press the Enter key without making any changes.

When the creation of the MLG910Q application is completed, a query execution prompt is displayed for you to specify the details of how the query is to be executed. If you do not want to execute the query at that point, press CF1 to exit the query utility.

Executing the Query Application

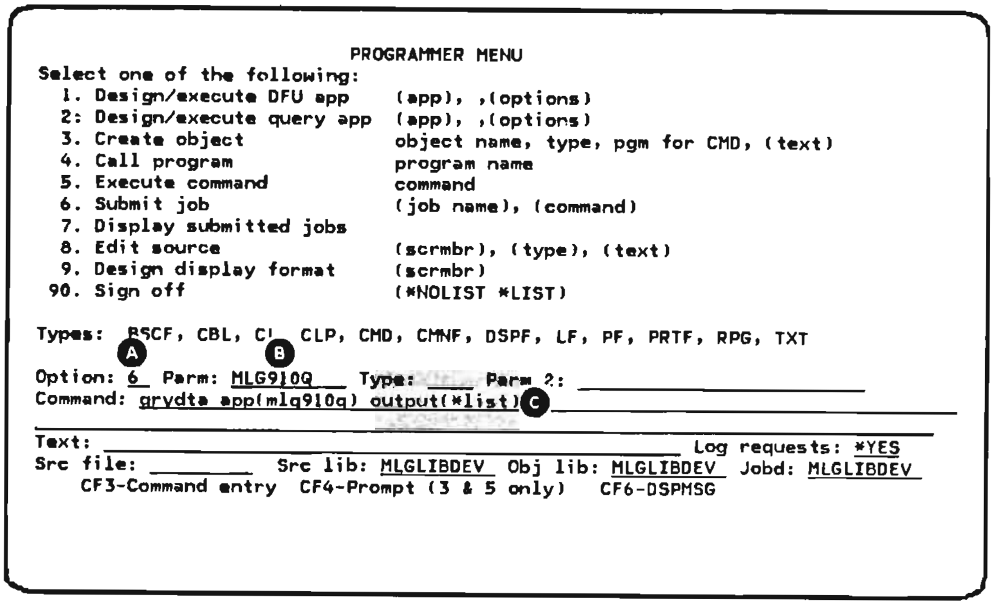
To execute a query application:

- Select option 2 on the programmer menu, as you did to create the application, and then select option 2 on the query menu when it is displayed.

- Enter the Query Data (QRYDTA) command.

In this example, you request a printed query report by submitting the QRYDTA command in a batch job from the programmer menu. Submitting the query as a batch job allows you to go on to other tasks at your work station while the query processing is being done.

The batch job to execute the query application MLG910Q is submitted from the programmer menu as follows:



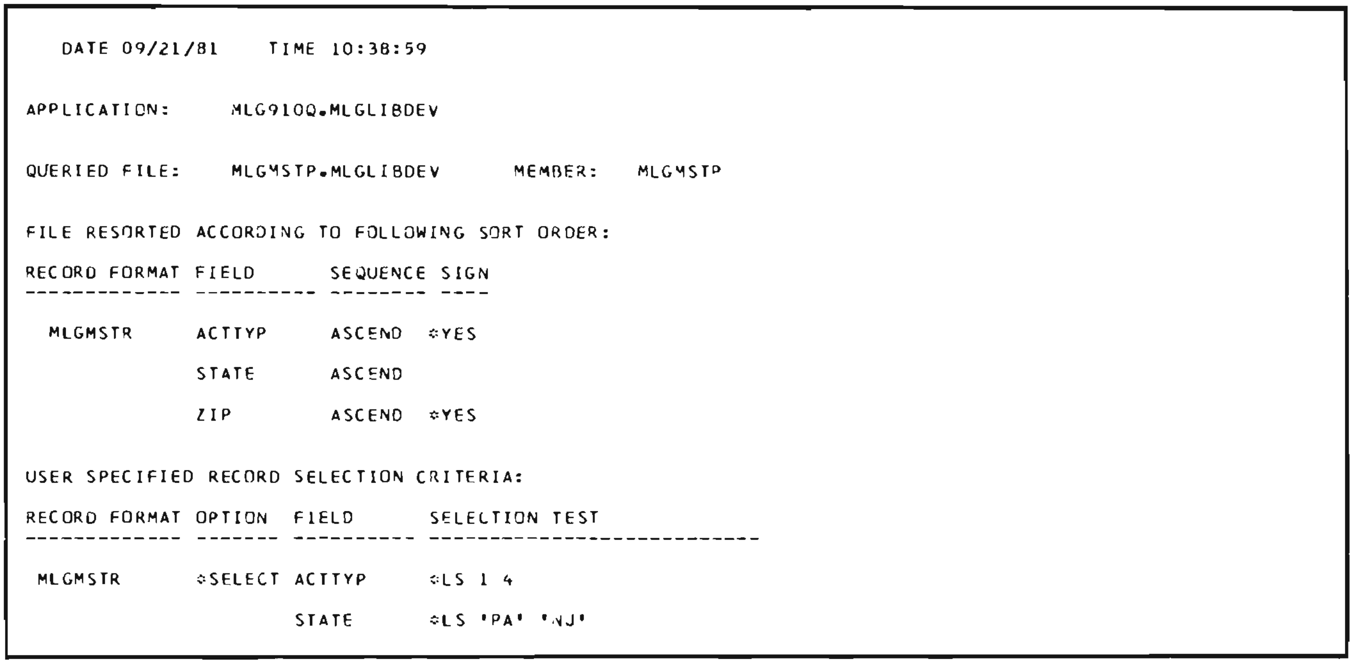
```
                              PROGRAMMER MENU
Select one of the following:
  1. Design/execute DFU app     (app), ,(options)
  2: Design/execute query app   (app), ,(options)
  3. Create object              object name, type, pgm for CMD, (text)
  4. Call program               program name
  5. Execute command            command
  6. Submit job                 (job name), (command)
  7. Display submitted jobs
  8. Edit source                (scrmbr), (type), (text)
  9. Design display format      (scrmbr)
 90. Sign off                   (*NOLIST *LIST)

Types:  BSCF, CBL, CL, CLP, CMD, CMNF, DSPF, LF, PF, PRTF, RPG, TXT

Option: 6  Parm: MLG910Q    Type: ____ Parm 2: ____________________
Command: qrydta app(mlg910q) output(*list)_____________________________
______________________________________________________________________
Text: ____________________________________________ Log requests: *YES
Src file: ________  Src lib: MLGLIBDEV  Obj lib: MLGLIBDEV  Jobd: MLGLIBDEV
    CF3-Command entry  CF4-Prompt (3 & 5 only)   CF6-DSPMSG
```

A Option 6 submits a job for batch processing.

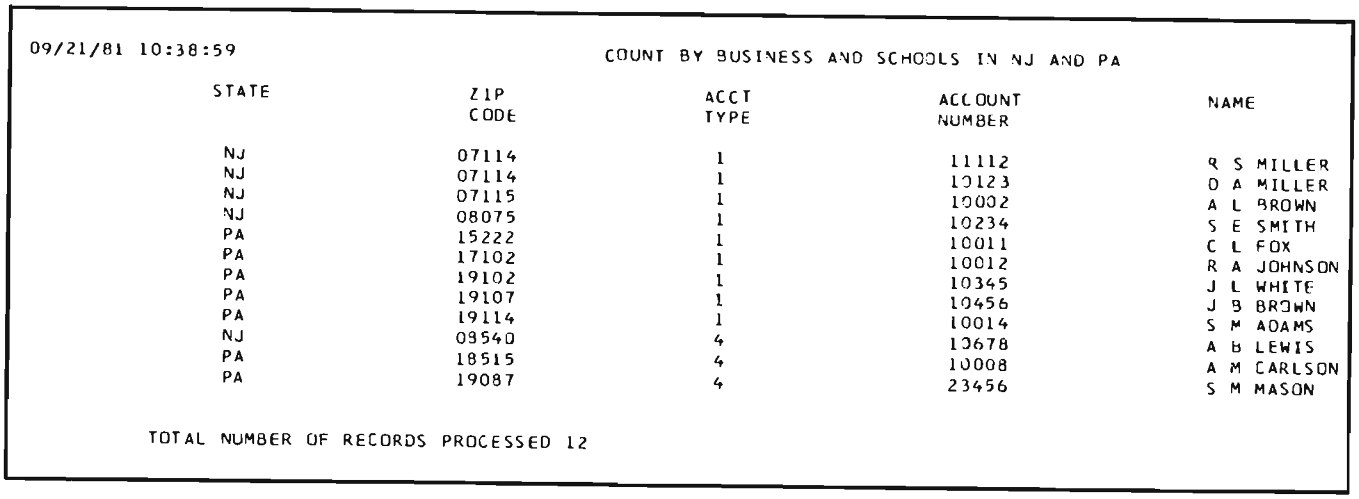
B The submitted batch job is given the name MLG910Q, as specified here.

C The QRYDTA command requests execution of the query program MLG910Q. The output is to be printed.

A sample of the query printed output is shown below.



```
   DATE 09/21/81    TIME 10:38:59

APPLICATION:      MLG910Q.MLGLIBDEV

QUERIED FILE:     MLGMSTP.MLGLIBDEV      MEMBER:    MLGMSTP

FILE RESORTED ACCORDING TO FOLLOWING SORT ORDER:

RECORD FORMAT FIELD      SEQUENCE SIGN
------------- ---------- -------- ----

  MLGMSTR     ACTTYP     ASCEND   *YES

              STATE      ASCEND

              ZIP        ASCEND   *YES

USER SPECIFIED RECORD SELECTION CRITERIA:

RECORD FORMAT OPTION  FIELD      SELECTION TEST
------------- ------- ---------- -------------------------

 MLGMSTR      *SELECT ACTTYP     *LS 1 4

                      STATE      *LS 'PA' 'NJ'
```



```
09/21/81 10:38:59
                                                 COUNT BY BUSINESS AND SCHOOLS IN NJ AND PA
                   STATE              ZIP             ACCT            ACCOUNT           NAME
                                      CODE            TYPE            NUMBER

                    NJ               07114              1              11112            R S MILLER
                    NJ               07114              1              10123            D A MILLER
                    NJ               07115              1              10002            A L BROWN
                    NJ               08075              1              10234            S E SMITH
                    PA               15222              1              10011            C L FOX
                    PA               17102              1              10012            R A JOHNSON
                    PA               19102              1              10345            J L WHITE
                    PA               19107              1              10456            J B BROWN
                    PA               19114              1              10014            S M ADAMS
                    NJ               08540              4              10678            A B LEWIS
                    PA               18515              4              10008            A M CARLSON
                    PA               19087              4              23456            S M MASON

          TOTAL NUMBER OF RECORDS PROCESSED 12
```

Query supports several other functions that allow you to make a more complex analysis. For information on these functions, refer to the *Query Utility Reference Manual and User's Guide.*

RECOVERY CONSIDERATIONS

If incorrect data has been entered for a master record, the application can be executed again to change the incorrect data. In this application approach, the changes are applied immediately to the master file (rather than to a transaction file and then to the master file through a batch maintenance program, as in Chapter 7).

If there is a system failure while the work station user is entering transactions, one of the following approaches could be used:

- When the master file MLGMSTP is created by the Create Physical File (CRTPF) command, there is a parameter named force write ratio (FRCRATIO) to control the frequency in which records are written to auxiliary storage. Using the default for this parameter provides the best performance. If the default (*NONE) is used, the work station users should check the last several transactions they entered to ensure they were recorded in auxiliary storage.

  The system may not write the records to auxiliary storage in the same sequence in which the updates occurred. All of the transactions entered may not be reflected because some records may have been in main storage at the time the system failure occurred and cannot be recovered.

  The listing that is printed upon the exit from the application may be used to determine what transactions were entered. The transactions will be listed in the correct sequence, but the listing may not be completely up-to-date. Printed records are spooled and are kept in buffers in main storage; therefore, some records may be lost if a system failure occurs.

- If the force write ratio parameter value is specified as 1, the work station users should look at their last entry to determine if it exists before continuing. In this situation, the printed listing may not be as up-to-date as the data base.

- A file can be created with a force write ratio parameter value of 1 and used as a data base log. Following a system failure, the last several records in the log can be reapplied to the data base. See the *CPF Programmer's Guide* for more information on data base logging.

To recover from damaged objects, such as when the master file (MLGMSTP) is unusable, restore the backup version of the file. The work station users will have to reenter the transactions entered since the last backup was made.

DFU does not provide a data base transaction file of the changes that were made to the file being updated; however, data base logging could be used to provide this type of file. A user-written recovery program could reapply the log to a backup copy of the master file. The log could also serve as an activity trace of changes to the data base and can help you correct records that were updated by faulty programs. Refer to the *CPF Programmer's Guide* for more information on data base logging.

Chapter 9. Batch Maintenance of Multiple Diskette Batches

OVERVIEW

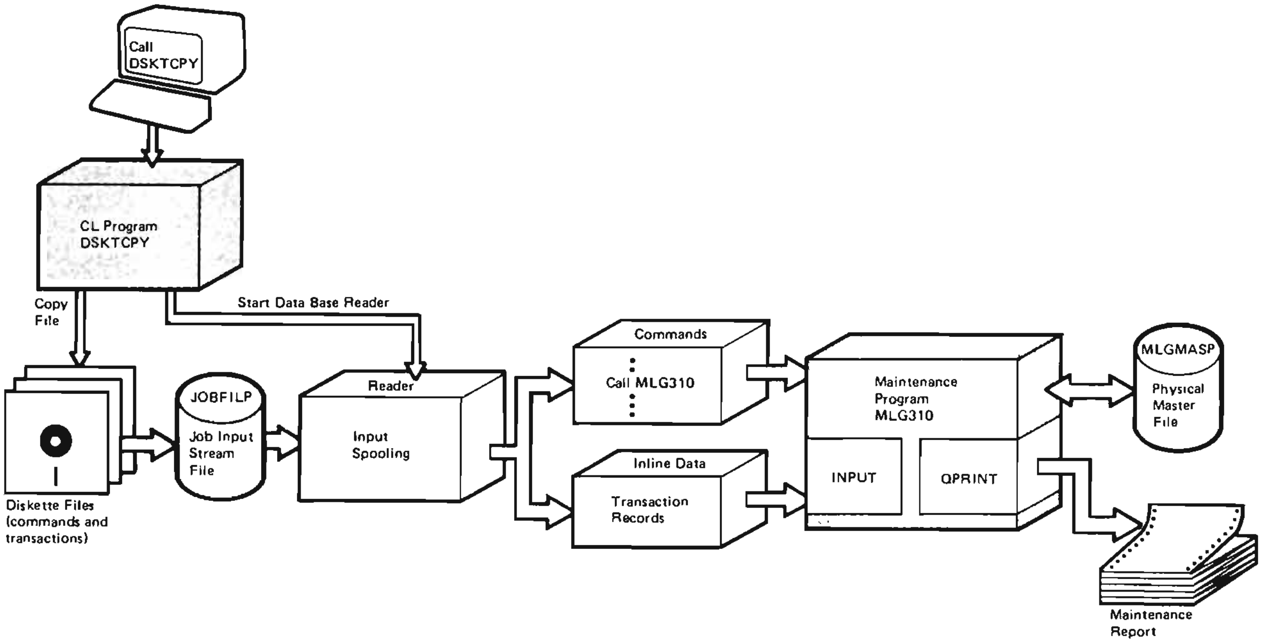
The application approach in Chapter 4 allowed multiple key-entry operators to key transactions onto diskettes, but the system operator had to enter a command to read *each* diskette into storage. The approach in this chapter allows the system operator to read *multiple* diskettes by entering a single command.

This approach to the mailing list application has the following characteristics:

- Batches of transactions are keyed onto diskettes that have been formatted to define the transactions as data for a batch job.

- The CL program; DSKTCPY, when called from a work station, copies the diskette data into a data base file, JOBFILP, creating an input stream of batch jobs. When all of the diskettes have been copied, the program starts a data base reader, which transfers the batch jobs to a job queue.

- When each batch job is executed, a CALL command within the job instructions calls an RPG III maintenance program (MLG310). The maintenance program updates a master file (MLGMASP) with the transactions that were copied from diskette as data for the job.



DISKETTE FILE ARRANGEMENT

Each diskette contains only two files. The *first file* contains the following CL for the job:

```
// JOB JOBD(MLGLIBDEV.MLGLIBDEV) JOB(MLG310)
CALL MLG310 /* MAINTAIN MAILING LIST FILE */
//  DATA  FILE(INPUT)
```

The *second file* contains transactions and is reset (the end-of-file pointer is changed at the offline device) each time the key entry operator enters a new batch of transactions.

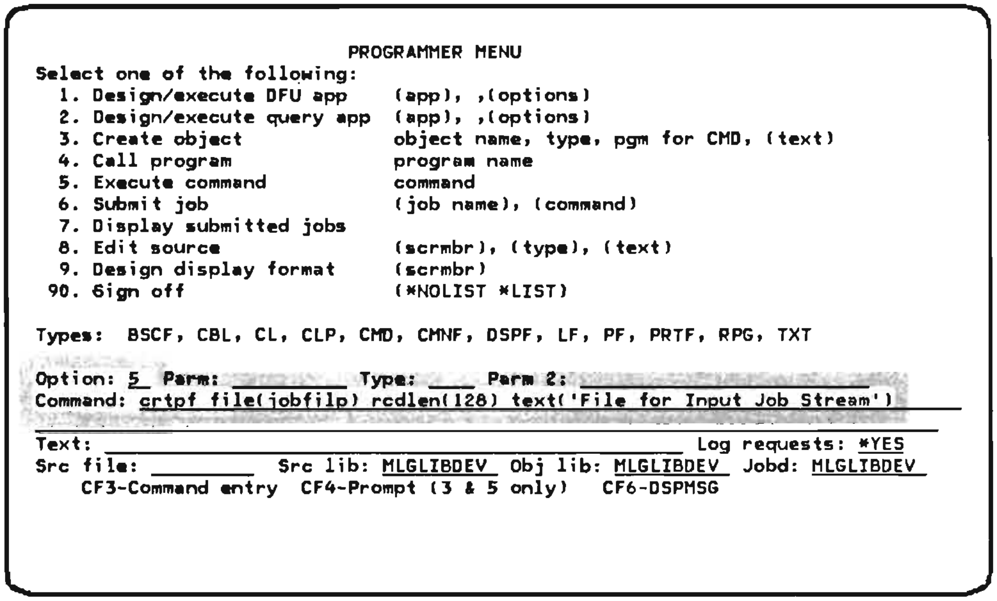
When the two files have been read into storage, the data base file that is created contains the following:

```
// JOB JOBD(MLGLIBDEV.MLGLIBDEV) JOB(MLG310)
CALL MLG310 /* MAINTAIN MAILING LIST FILE */
//  DATA  FILE(INPUT)
      •
      • (data from second file)
      •
```

On the next diskette that is read, the // JOB command designates both the end of the previous job and the beginning of a new job. Multiple diskettes can be read consecutively to create a data base file that contains an input stream with multiple jobs. This input stream is transferred to a job queue when the Start Data Base Reader (STRDBRDR) command is executed.

JOBFILP FILE

To use this approach, first create a data base file to contain the input stream that will be read from the diskettes:



```
                            PROGRAMMER MENU
Select one of the following:
  1. Design/execute DFU app     (app), ,(options)
  2. Design/execute query app   (app), ,(options)
  3. Create object              object name, type, pgm for CMD, (text)
  4. Call program               program name
  5. Execute command            command
  6. Submit job                 (job name), (command)
  7. Display submitted jobs
  8. Edit source                (scrmbr), (type), (text)
  9. Design display format      (scrmbr)
 90. Sign off                   (*NOLIST *LIST)

Types:  BSCF, CBL, CL, CLP, CMD, CMNF, DSPF, LF, PF, PRTF, RPG, TXT

Option: 5_ Parm: _________ Type: ___ Parm 2: _________________________
Command: crtpf file(jobfilp) rcdlen(128) text('File for Input Job Stream')
_____________________________________________________________________
Text: __________________________________________________ Log requests: *YES
Src file: ________ Src lib: MLGLIBDEV  Obj lib: MLGLIBDEV  Jobd: MLGLIBDEV
    CF3-Command entry  CF4-Prompt (3 & 5 only)   CF6-DSPMSG
```

This file is not read by the RPG III maintenance program (MLG310); it is only a means of holding the input stream. Field level DDS are not needed because the record format of the file can be defined as one field and the file is processed only sequentially. When the RCDLEN parameter is used, no DDS source is required. This command creates a definition for a 128-byte record in the file JOBFILP. Because no library name is used in the CRTPF command, the data base file is created in the general purpose library (QGPL).

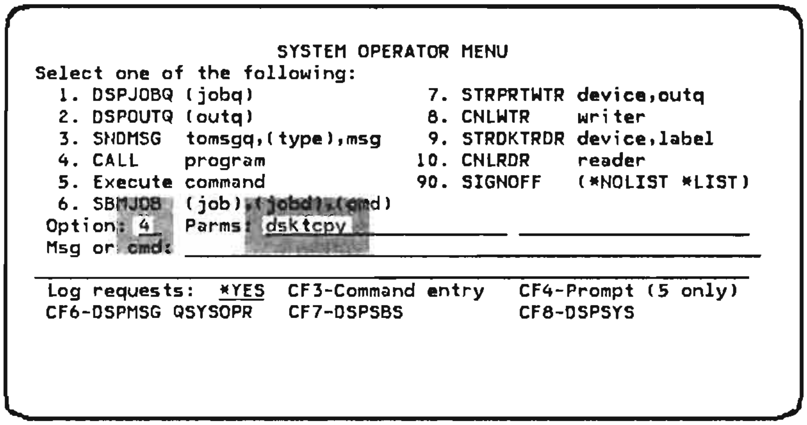
By default when a file is created, only the owner can clear the entire file. To allow the system operator to clear the file JOBFILP, the Grant Object Authority (GRTOBJAUT) command is issued:

```
GRTOBJAUT OBJ(JOBFILP) OBJTYPE(*FILE) USER(QSYSOPR)
   AUT(*OBJMGT)
```

The object management (*OBJMGT) authorization allows the system operator to clear the file.

THE DSKTCPY PROGRAM

After the diskettes to be read are mounted in the diskette magazine drive, the system operator selects option 4 on the system operator menu to call a program named DSKTCPY:

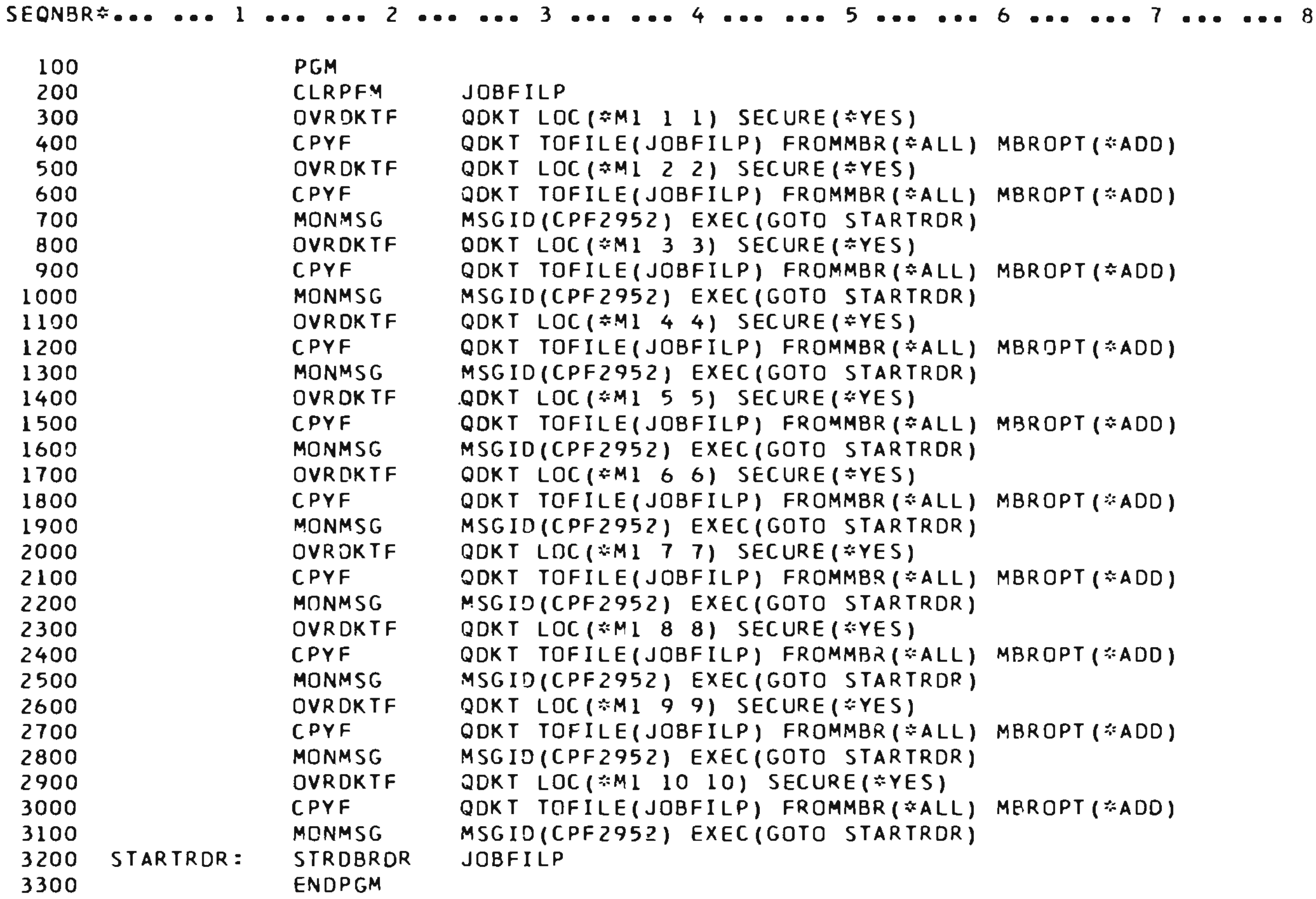


```
                      SYSTEM OPERATOR MENU
Select one of the following:
  1. DSPJOBQ (jobq)               7. STRPRTWTR device,outq
  2. DSPOUTQ (outq)               8. CNLWTR    writer
  3. SNDMSG  tomsgq,(type),msg    9. STRDKTRDR device,label
  4. CALL    program             10. CNLRDR    reader
  5. Execute command             90. SIGNOFF   (*NOLIST *LIST)
  6. SBMJOB  (job),(jobd),(cmd)
 Option: 4   Parms: dsktcpy___________  _________________
 Msg or cmd: ____________________________________________
________________________________________________________
 Log requests: *YES  CF3-Command entry   CF4-Prompt (5 only)
 CF6-DSPMSG QSYSOPR  CF7-DSPSBS          CF8-DSPSYS
```

The DSKTCPY program reads each diskette and adds the file data to the file JOBFILP. If the magazine is not full of diskettes, the system operator is prompted with an inquiry message when the first empty slot is encountered. The system operator should respond by entering a C (for cancel) so that no diskettes will be read after the first empty slot. When the reading of the diskettes is completed, the program starts a data base reader that places all the jobs on the QBATCH job queue.

The CL program DSKTCPY is shown in Figure 9-1. It begins with a Clear Physical File Member (CLRPFM) command to reset JOBFILP. Each diskette slot in the magazine is positioned by the Override with Diskette File (OVRDKTF) command. It describes the location of the diskette within the magazine and also specifies that the override is secure. This means that any override commands previously entered by the system operator will not be used. The Copy File (CPYF) command copies all the files into the file QDKT and adds them to JOBFILP.

For the second through the tenth CPYF commands, a Monitor Message (MONMSG) command is used to monitor for the message CPF2952. CPF2952 is an escape message that will be sent to the program if the system operator responds with a C (cancel) to the message that appears when the first empty slot is encountered. When a C is entered (or when all slots are filled and the diskette in the last slot has been read), the program branches to the Start Data Base Reader (STRDBRDR) command.



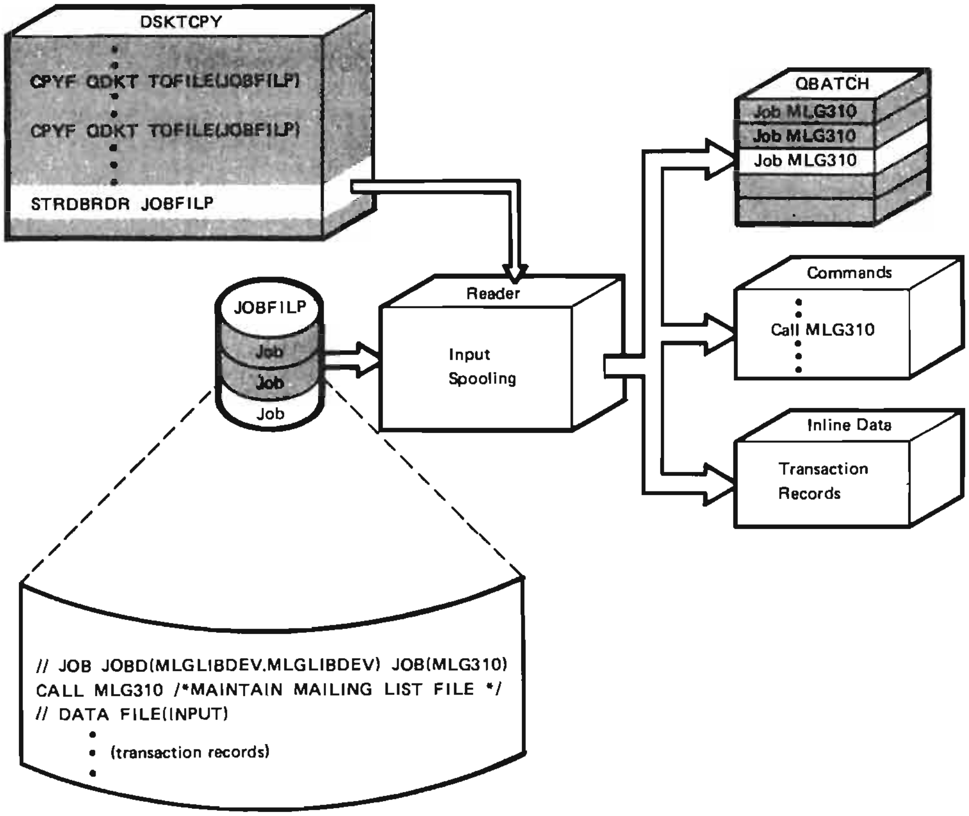
```
SEQNBR*... ... 1 ... ... 2 ... ... 3 ... ... 4 ... ... 5 ... ... 6 ... ... 7 ... ... 8
                    PGM
                    CLRPFM     JOBFILP
                    OVRDKTF    QDKT LOC(*M1 1 1) SECURE(*YES)
                    CPYF       QDKT TOFILE(JOBFILP) FROMMBR(*ALL) MBROPT(*ADD)
                    OVRDKTF    QDKT LOC(*M1 2 2) SECURE(*YES)
                    CPYF       QDKT TOFILE(JOBFILP) FROMMBR(*ALL) MBROPT(*ADD)
                    MONMSG     MSGID(CPF2952) EXEC(GOTO STARTRDR)
                    OVRDKTF    QDKT LOC(*M1 3 3) SECURE(*YES)
                    CPYF       QDKT TOFILE(JOBFILP) FROMMBR(*ALL) MBROPT(*ADD)
                    MONMSG     MSGID(CPF2952) EXEC(GOTO STARTRDR)
                    OVRDKTF    QDKT LOC(*M1 4 4) SECURE(*YES)
                    CPYF       QDKT TOFILE(JOBFILP) FROMMBR(*ALL) MBROPT(*ADD)
                    MONMSG     MSGID(CPF2952) EXEC(GOTO STARTRDR)
                    OVRDKTF    QDKT LOC(*M1 5 5) SECURE(*YES)
                    CPYF       QDKT TOFILE(JOBFILP) FROMMBR(*ALL) MBROPT(*ADD)
                    MONMSG     MSGID(CPF2952) EXEC(GOTO STARTRDR)
                    OVRDKTF    QDKT LOC(*M1 6 6) SECURE(*YES)
                    CPYF       QDKT TOFILE(JOBFILP) FROMMBR(*ALL) MBROPT(*ADD)
                    MONMSG     MSGID(CPF2952) EXEC(GOTO STARTRDR)
                    OVRDKTF    QDKT LOC(*M1 7 7) SECURE(*YES)
                    CPYF       QDKT TOFILE(JOBFILP) FROMMBR(*ALL) MBROPT(*ADD)
                    MONMSG     MSGID(CPF2952) EXEC(GOTO STARTRDR)
                    OVRDKTF    QDKT LOC(*M1 8 8) SECURE(*YES)
                    CPYF       QDKT TOFILE(JOBFILP) FROMMBR(*ALL) MBROPT(*ADD)
                    MONMSG     MSGID(CPF2952) EXEC(GOTO STARTRDR)
                    OVRDKTF    QDKT LOC(*M1 9 9) SECURE(*YES)
                    CPYF       QDKT TOFILE(JOBFILP) FROMMBR(*ALL) MBROPT(*ADD)
                    MONMSG     MSGID(CPF2952) EXEC(GOTO STARTRDR)
                    OVRDKTF    QDKT LOC(*M1 10 10) SECURE(*YES)
                    CPYF       QDKT TOFILE(JOBFILP) FROMMBR(*ALL) MBROPT(*ADD)
                    MONMSG     MSGID(CPF2952) EXEC(GOTO STARTRDR)
         STARTRDR:  STRDBRDR   JOBFILP
                    ENDPGM
```

Figure 9-1. CL Program DSKTCPY

The DSKTCPY program is created by using the programmer menu and SEU. The procedure was described in previous chapters.

UPDATING THE MASTER FILE

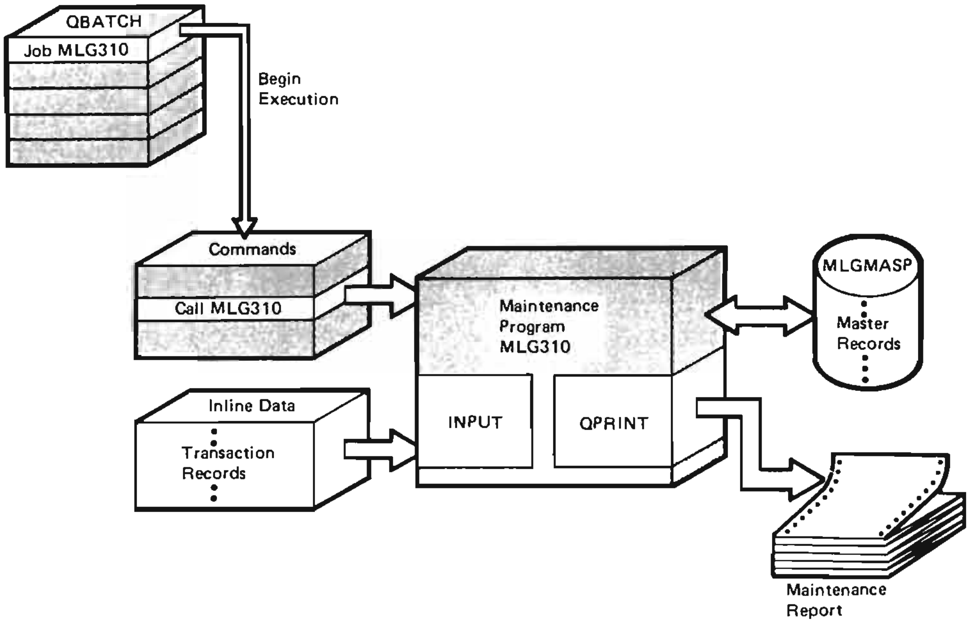
The data base reader started by the DSKTCPY program transfers the jobs from the JOBFILP data base file to the QBATCH job queue. The transactions that are defined as data for each job are stored in an inline data file.



```
CPYF QDKT TOFILE(JOBFILP)
CPYF QDKT TOFILE(JOBFILP)
STRDBRDR JOBFILP
```

```
// JOB JOBD(MLGLIBDEV.MLGLIBDEV) JOB(MLG310)
CALL MLG310 /*MAINTAIN MAILING LIST FILE */
// DATA FILE(INPUT)
  •
  •  (transaction records)
  •
```

When one of the batch jobs becomes the highest priority job on the QBATCH job queue, the job is executed and the MLG310 maintenance program is called. As described in Chapter 4, the MLG310 program updates the data base master file, MLGMASP, with the transactions in the inline data file associated with the job (the RPG III specifications for MLG310 are shown in Figures 4-3 through 4-7).



The procedure just described for using diskette input can be used to read in transactions for multiple applications intermixed within the diskette magazines. Each type of transaction file would be processed by a unique program.

RECOVERY CONSIDERATIONS

The recovery considerations for the approach in Chapter 4 also apply to this approach.

Chapter 10. Batch Maintenance of Multiple Work Station Entries

OVERVIEW

The application approach in Chapter 7 allowed only one work station user at a time to be entering transactions for subsequent batch processing. The approach in this chapter allows multiple work station users to enter batches of transactions at the same time. This approach provides the following capabilities:

- Allows new batch input from multiple work station users to be entered at the same time a maintenance program is processing batches of transactions that are completely entered.

- Allows transactions to be added to a batch of transactions that have been entered but not yet processed by the maintenance program.

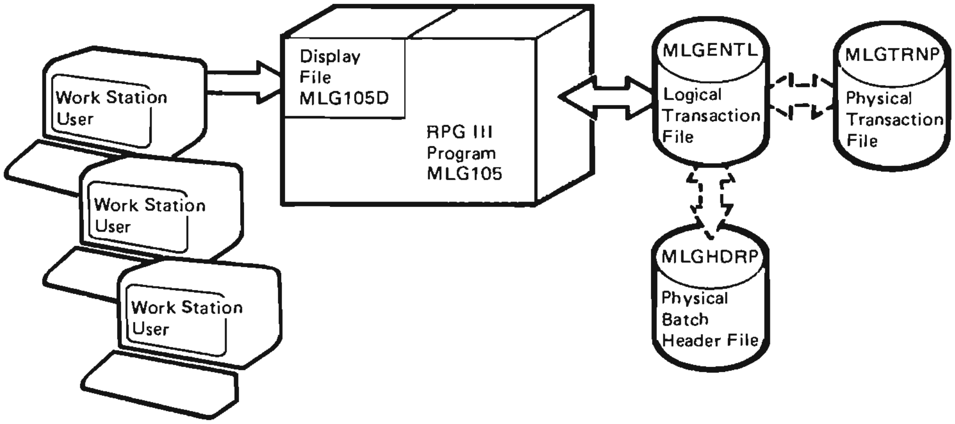
- Prevents duplicate batch numbers from being entered.

- Prevents the maintenance program from processing a batch of transactions that is not completely entered.

- Allows the work station user to review the last transaction he entered in case he loses his place or recovery is needed.

Individual batches are distinguished by having work station users provide a batch identifier when they begin entering transactions. The status of a batch (that is, whether it is available for processing) is defined by having the work station user indicate when the batch is complete. The batch identifier and status are stored in a physical header file, MLGHDRP, and the transactions in a physical transaction file, MLGTRNP.

An RPG III program, MLG105, and an associated display file, MLG105D, provide the display and processing support for entering the transactions from a work station. The MLG105 program accesses the header file and physical transaction file through a logical transaction file, MLGENTL.



This approach assumes that the completed batches of transactions in the transaction file are applied to the master file by a batch maintenance program similar to MLG310, as described in Chapter 7. The maintenance program will apply a batch of transactions to the master file only if the associated batch header records indicate that the batch is ready to be processed.

BATCH HEADER RECORD

This approach begins with creating a unique header record for each batch of transactions that is to be entered. This record contains headings for a batch of transaction records and control codes that specify the current status of that batch. The header record has the following format:

Field Description	Field Name	Length (Characters)	Decimal Positions	Field Type
Batch Number	BATNUM	6	0	Numeric
Batch Date	BATDAT	6	0	Numeric
Batch Status	BATSTS	1	0	Numeric
Batch Description	BATDSC	35		Character

The *batch status* is specified by one of the three following codes:

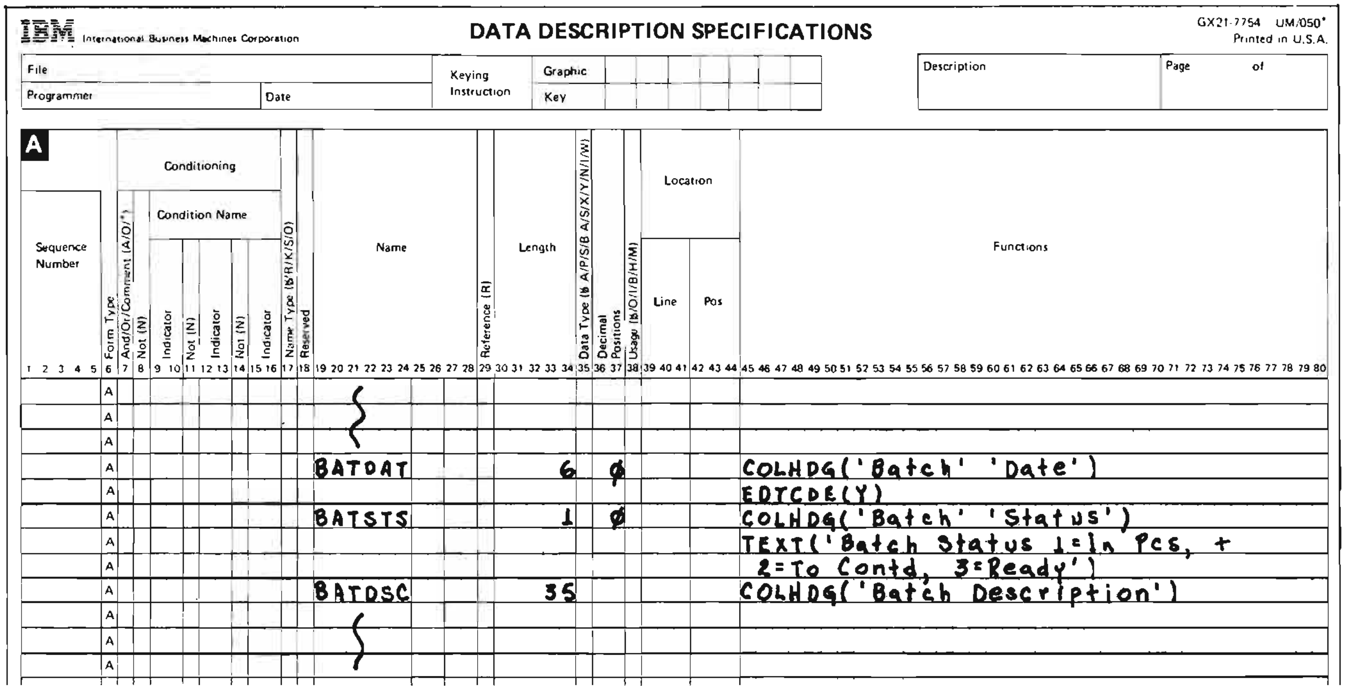
Code	Meaning
1	Data entry is in process
2	Data entry is to be continued
3	This batch is ready to be processed by the maintenance program

Other valid codes could be added to the batch processing program MLG105; however, additional codes are not shown in this publication.

The batch number, batch date, batch status, and batch description fields should be added to the field reference file MLGREFP (which was created in Chapter 6).

You can add these fields to MLGREFP by the following procedure.:

- Use option 8 on the programmer menu to obtain the SEU display of the member MLGREFP in the source file QDDSSRC. Add the DDS in Figure 10-1 to the existing DDS in the source member (Figure 6-2).
- Use option 3 on the programmer menu to recreate the file (press the CF11 key instead of the Enter key to replace the existing MLGREFP file by the new MLGREFP file).



IBM International Business Machines Corporation

DATA DESCRIPTION SPECIFICATIONS

GX21-7754 UM/050*
Printed in U.S.A.

File
Programmer
Date
Keying Instruction
Graphic
Key
Description
Page of

A

Sequence Number
Form Type
And/Or/Comment (A/O/*)
Not (N)
Indicator
Not (N)
Indicator
Not (N)
Indicator
Conditioning
Condition Name
Name Type (b/R/K/S/O)
Reserved
Name
Reference (R)
Length
Data Type (b A/P/S/B A/S/X/Y/N/I/W)
Decimal Positions
Usage (b/O/I/B/H/M)
Location
Line
Pos
Functions
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80

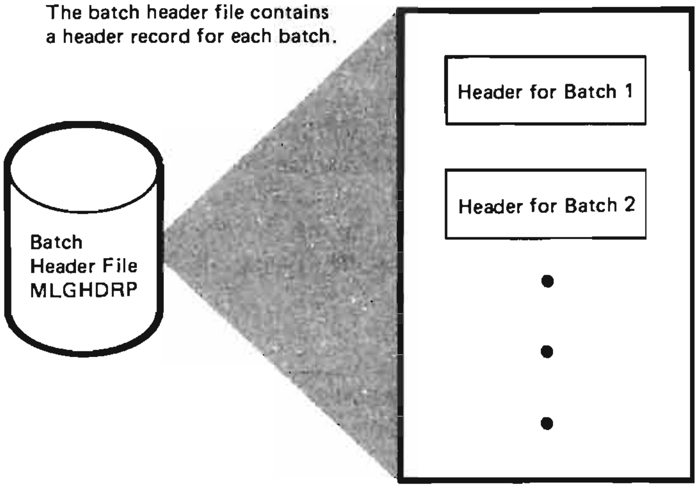
```
     A
     A
     A
     A                  BATDAT         6  Ø       COLHDG('Batch' 'Date')
     A                                            EDTCDE(Y)
     A                  BATSTS         1  Ø       COLHDG('Batch' 'Status')
     A                                            TEXT('Batch Status 1=In Pcs, +
     A                                             2=To Contd, 3=Ready')
     A                  BATDSC        35          COLHDG('Batch Description')
     A
     A
     A
```

Figure 10-1. Additional DDS for MLGREFP Field Reference File

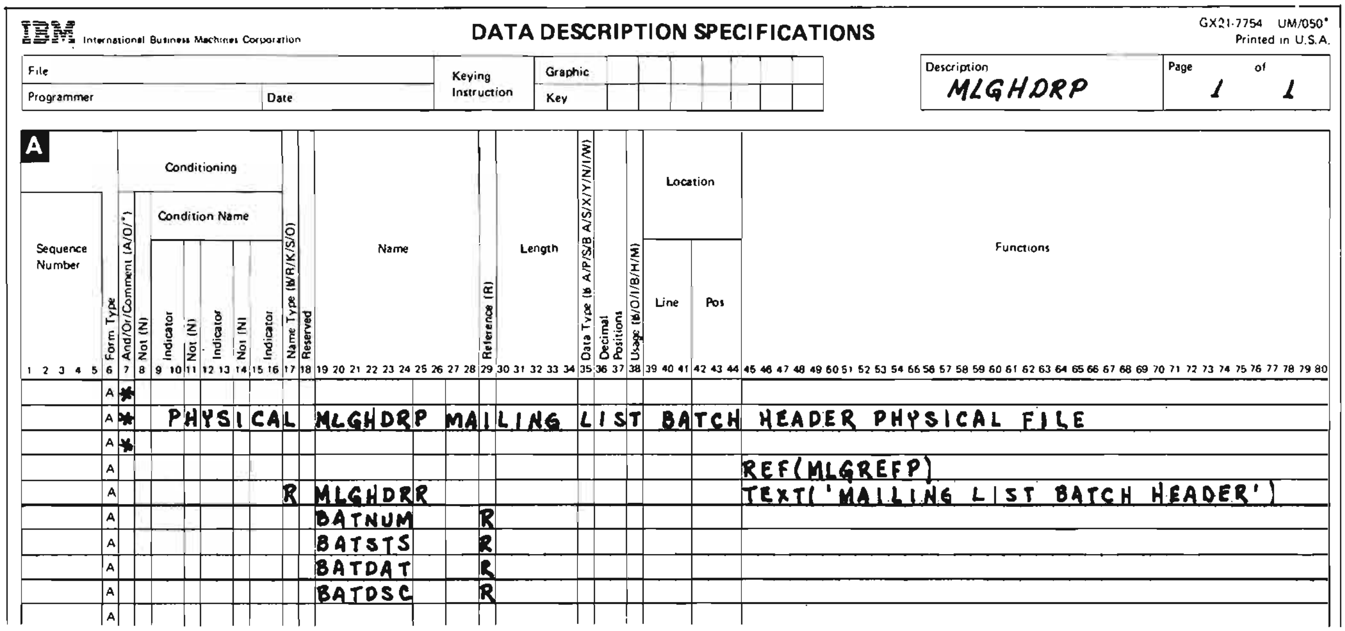
The EDTCDE keyword specifies that BATDAT is edited with the edit code Y when it is used by DFU. For a description of edit codes, refer to the *CPF Reference Manual—DDS*.

Because fields were *added* to MLGREFP, no changes need to be made to the existing physical and logical files.

A physical file named MLGHDRP is created to contain the batch header data, as shown in the following illustration.


The batch header file contains a header record for each batch.
Batch Header File MLGHDRP
Header for Batch 1
Header for Batch 2

The DDS for MLGHDRP are shown in Figure 10-2.


IBM International Business Machines Corporation

DATA DESCRIPTION SPECIFICATIONS

GX21-7754 UM/050*
Printed in U.S.A.

File
Programmer
Date
Keying Instruction
Graphic
Key

Description MLGHDRP
Page 1 of 1

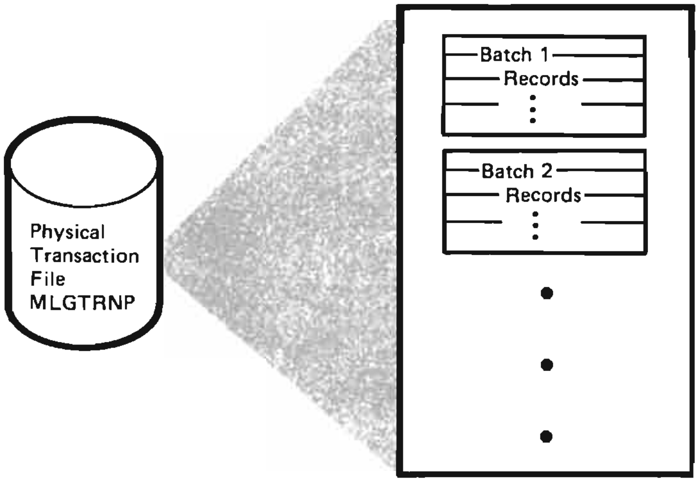
A

```
     A*
     A* PHYSICAL MLGHDRP MAILING LIST BATCH HEADER PHYSICAL FILE
     A*
     A                                      REF(MLGREFP)
     A          R MLGHDRR                   TEXT('MAILING LIST BATCH HEADER')
     A            BATNUM    R
     A            BATSTS    R
     A            BATDAT    R
     A            BATDSC    R
     A
```

Figure 10-2. DDS for MLGHDRP Batch Header File

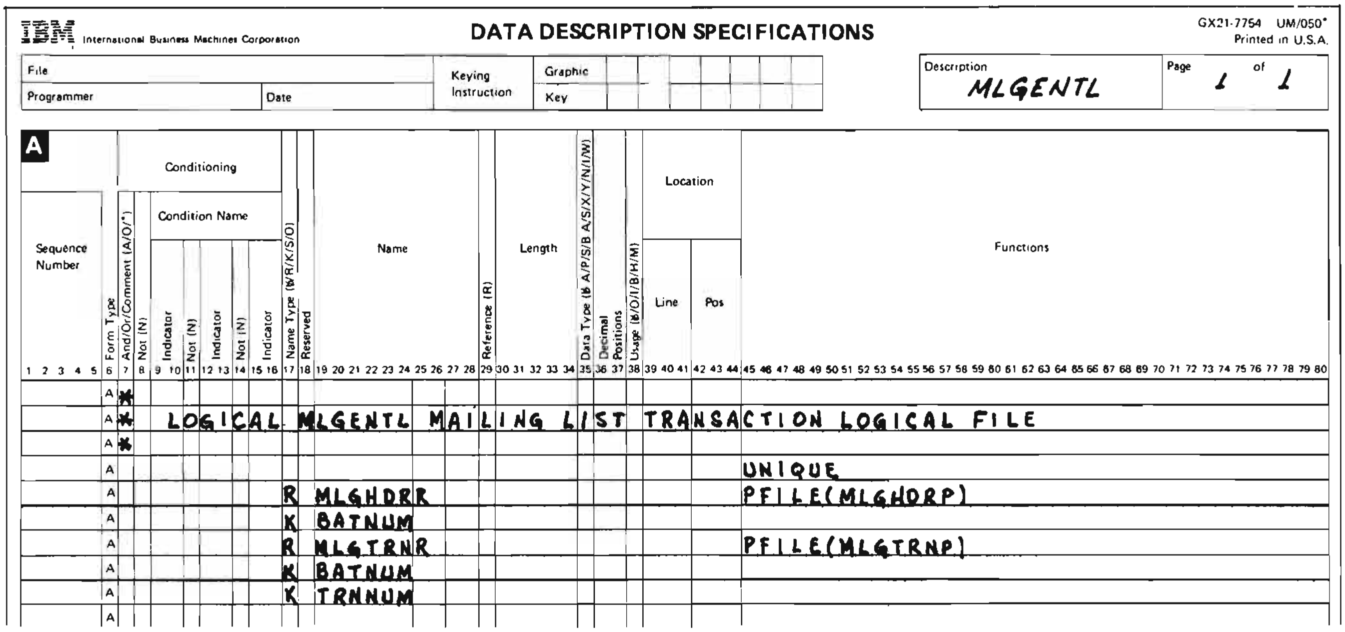
TRANSACTION FILES

A physical file named MLGTRNP is used to contain the transaction records by batch, as shown in the following illustration:



MLGTRNP was created and used in Chapter 6 (the DDS for MLGTRNP are shown in Figure 6-6).

A logical file named MLGENTL is created for the batches of transactions as they are being entered. Both header information and transaction records are in this file. The DDS for MLGENTL are shown in Figure 10-3.



IBM International Business Machines Corporation

DATA DESCRIPTION SPECIFICATIONS

GX21-7754 UM/050*
Printed in U.S.A.

File | Programmer | Date | Keying Instruction | Graphic | Key

Description: MLGENTL — Page 1 of 1

A

Sequence Number | Form Type | And/Or/Comment (A/O/*) | Conditioning: Condition Name — Not (N), Indicator, Not (N), Indicator, Not (N), Indicator | Name Type (b/R/K/S/O) | Reserved | Name | Reference (R) | Length | Data Type (b A/P/S/B A/S/X/Y/N/I/W) | Decimal Positions | Usage (b/O/I/B/H/M) | Location: Line, Pos | Functions

```
     A*
     A*  LOGICAL MLGENTL MAILING LIST TRANSACTION LOGICAL FILE
     A*
     A                                      UNIQUE
     A          R MLGHDRR                   PFILE(MLGHDRP)
     A          K BATNUM
     A          R MLGTRNR                   PFILE(MLGTRNP)
     A          K BATNUM
     A          K TRNNUM
     A
```

Figure 10-3. DDS for MLGENTL Transaction File

The two physical files MLGHDRP and MLGTRNP are used. A key field in each of these files is BATNUM. The physical transaction (MLGTRNP) file also uses TRNNUM as a key field. The keyword UNIQUE specifies that the entries in the key fields must be unique within each physical file. This means that the batch header records must have unique batch numbers and the transaction records must have unique batch numbers and unique transaction numbers.

Because MLGHDRP is specified first in the DDS, batch header data is on the access path of MLGENTL before any transaction records of that same batch. MLGHDRP does not contain the field TRNNUM (every record type in a file is not required to contain the same number of key fields).

TRANSACTION DATA ENTRY PROGRAM AND DISPLAY FILE

An RPG III program named MLG105 and a display file named MLG105D are created to allow a work station user to enter batches of transactions at a work station. The work station user begins by calling MLG105 (the method used to call the program is not discussed here, although MLG105 can be called as the result of selecting an option on a menu).

Program Flow

A flowchart of the program is shown in Figure 10-4. This flow of the program centers around the entries made by the work station user.

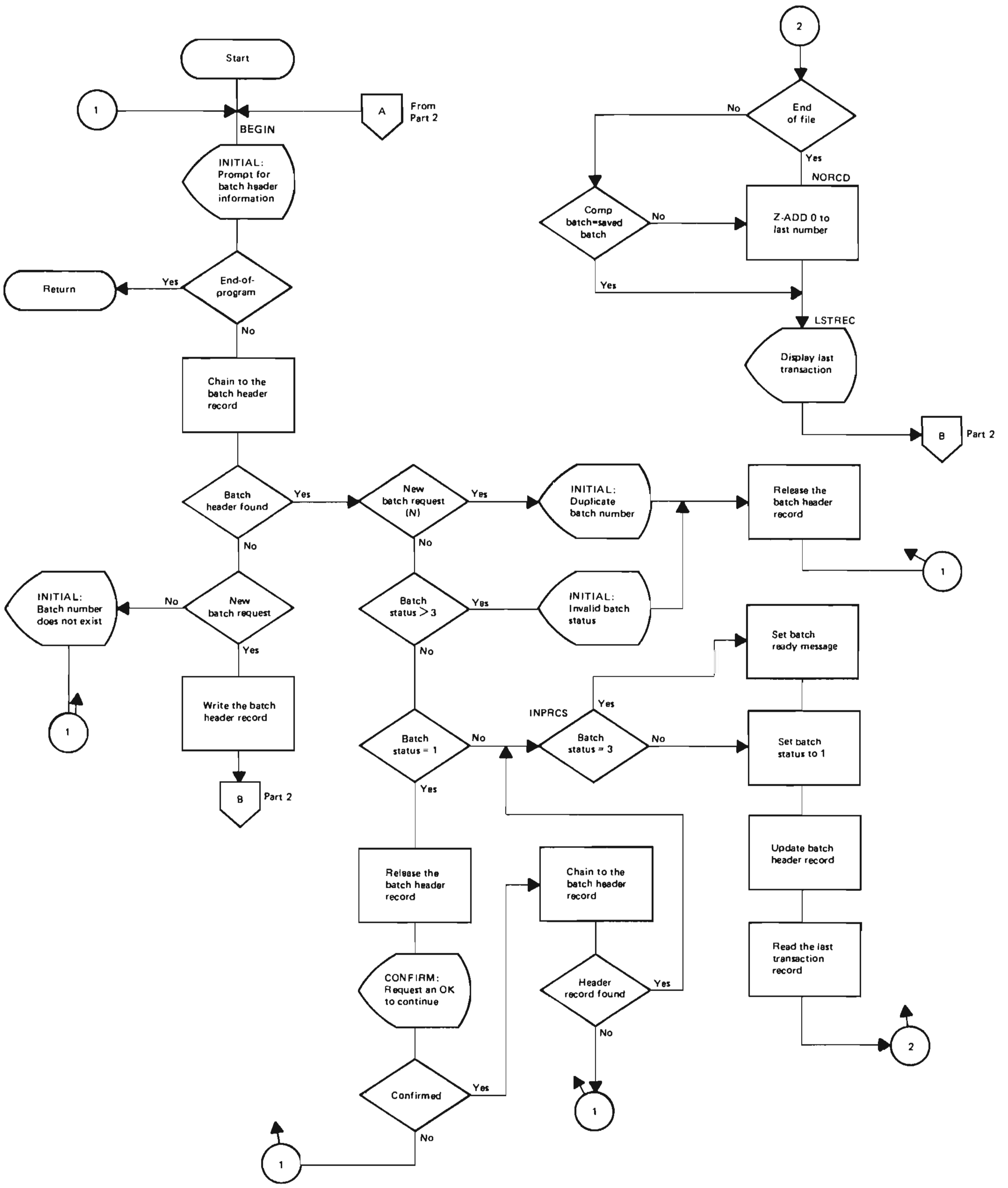


Figure 10-4 (Part 1 of 2). Flowchart of MLG105 Transaction Data Entry Program

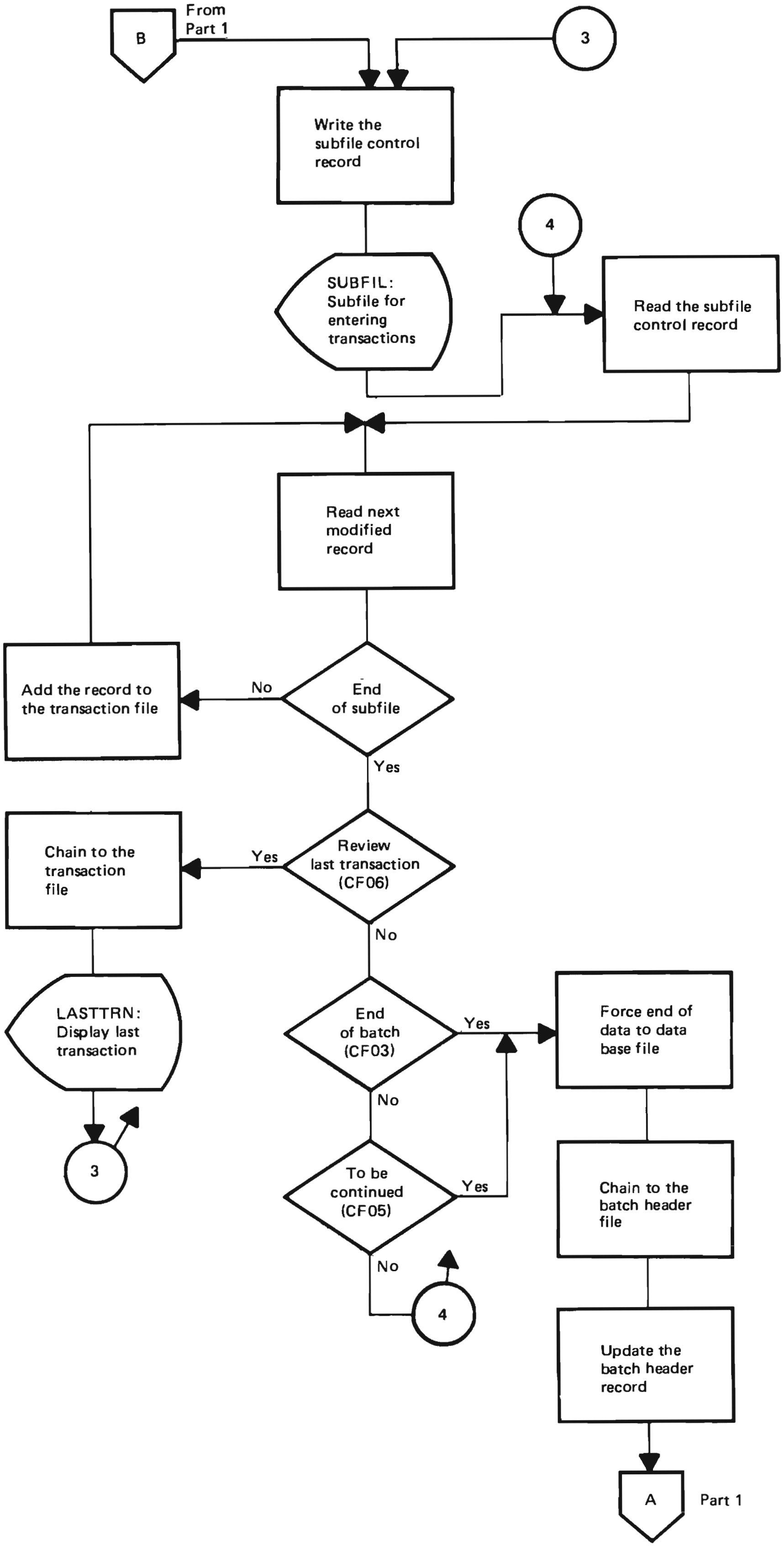
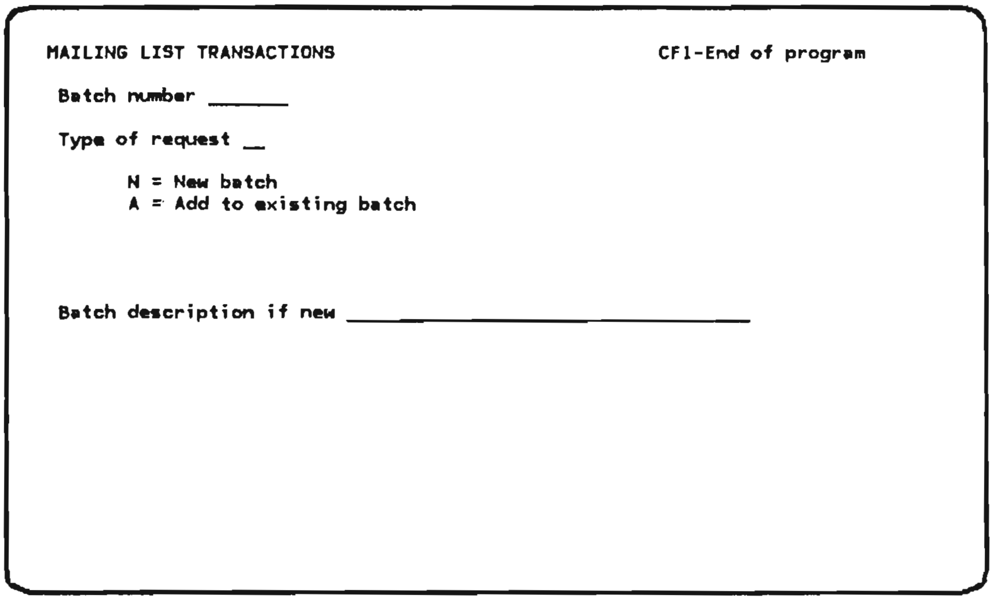

B
From
Part 1
3
Write the subfile control record
4
SUBFIL: Subfile for entering transactions
Read the subfile control record
Read next modified record
No
End of subfile
Add the record to the transaction file
Yes
Yes
Review last transaction (CF06)
Chain to the transaction file
No
LASTTRN: Display last transaction
3
End of batch (CF03)
Yes
Force end of data to data base file
No
To be continued (CF05)
Yes
Chain to the batch header file
No
4
Update the batch header record
A
Part 1

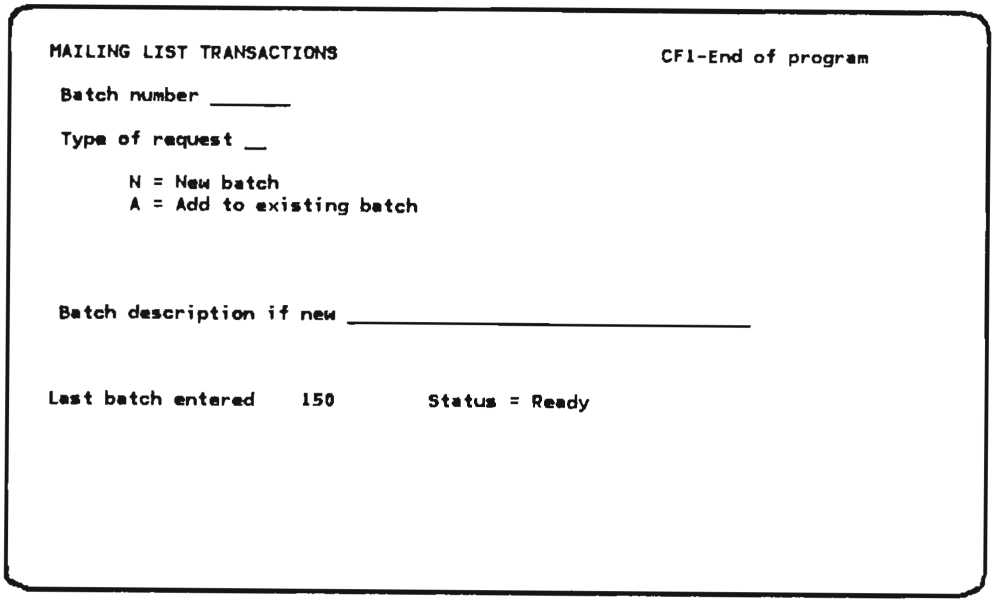
Figure 10-4 (Part 2 of 2). Flowchart of MLG105 Transaction Data Entry Program

The work station user begins by keying in batch header data on the following display, which is part of MLG105D and named INITIAL:



```
MAILING LIST TRANSACTIONS                                CF1-End of program
 Batch number _______

 Type of request __

       N = New batch
       A = Add to existing batch


 Batch description if new ___________________________________
```

The work station user keys in a batch number and requests to work on either a new batch or an existing batch. If the user requests to work on a new batch, a batch description may also be keyed in. After at least one batch is entered, the initial prompt looks like this:



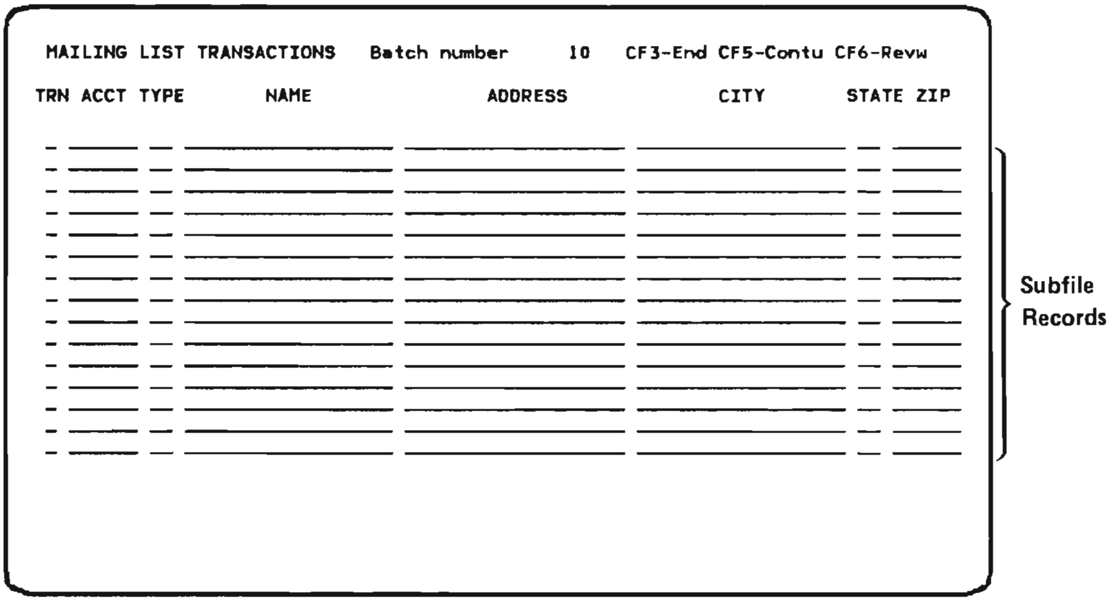
```
MAILING LIST TRANSACTIONS                                CF1-End of program
 Batch number _______

 Type of request __

       N = New batch
       A = Add to existing batch


 Batch description if new ___________________________________


Last batch entered    150        Status = Ready
```

The display shows the batch number and the status of the last batch of transactions entered.

The entries that are made on this prompt must be checked for validity because the primary purpose of this prompt is to do one of the following:

- Write a new batch header
- Find an existing batch header if the work station user requests to add to an existing batch
- Confirm that a work station user can add to an existing batch

Transactions are keyed in on the following display, which uses subfile records of record name SUBFIL in display file MLG105D:



```
 MAILING LIST TRANSACTIONS   Batch number    10   CF3-End CF5-Contu CF6-Revw
TRN ACCT TYPE       NAME                ADDRESS                 CITY       STATE ZIP
```

Subfile Records

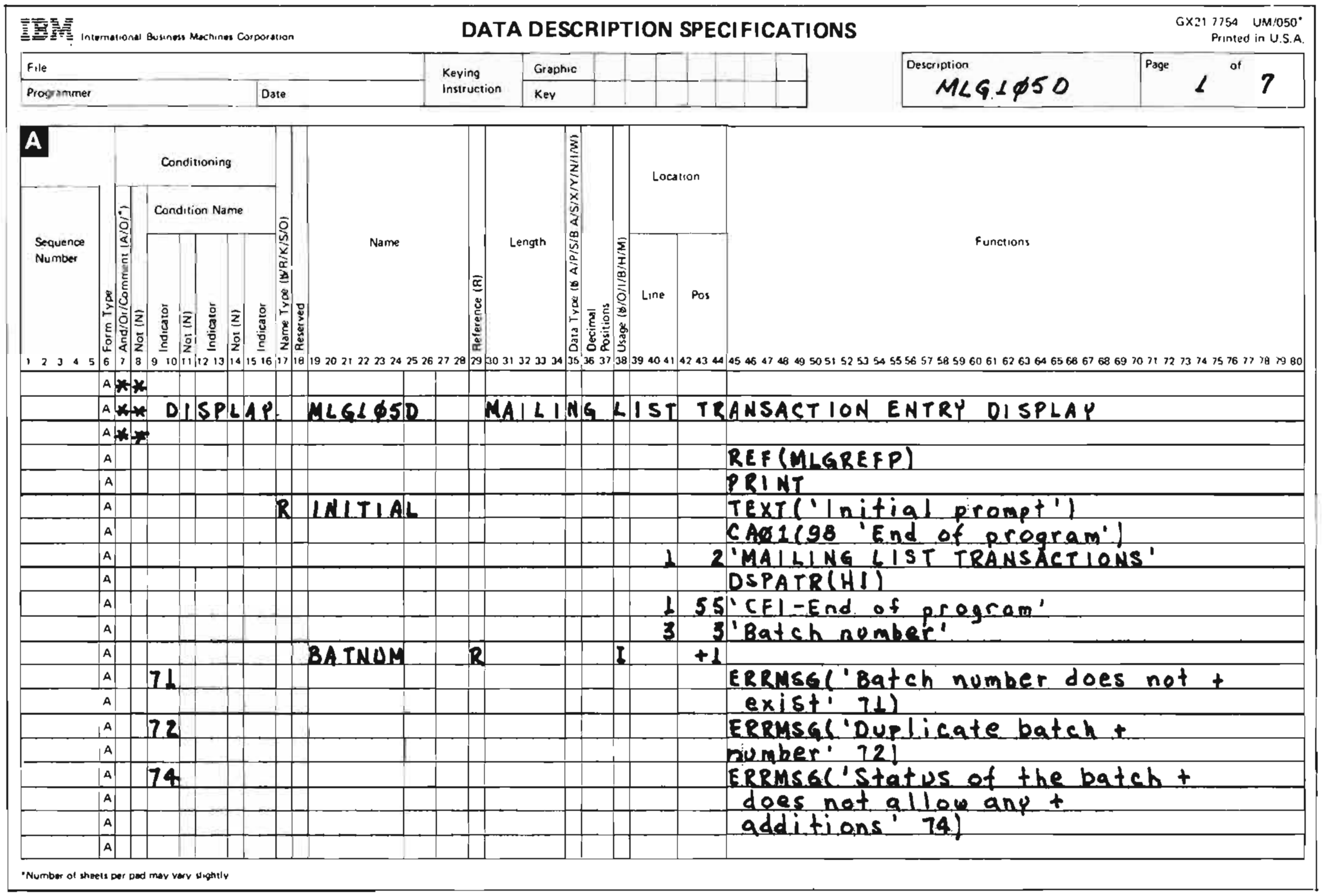
Each transaction entered is a record in a *subfile*. A subfile is a group of records of the same record format that can be displayed concurrently at a work station. The system sends the entire group of records to the work station in a single operation and receives the group in another operation. The program can process records one at a time even though a group of records was entered.

The work station user keys one or more transactions into the subfile and presses the Enter key. As soon as the data is accepted by the system, the fields that were entered are erased and the keyboard is unlocked to allow more entries. This programming technique helps optimize the time used to enter transactions, but does not allow the program to edit the transactions and provide feedback for errors. This type of key entry is often called *heads down,* meaning that the work station user is keying from a source document and does not look at the display. The work station user can fill the entire display with transactions before pressing the Enter key. This reduces the number of times that the system must interact with the user, thereby providing for a fast rate of key entry. Editing of the data occurs later, while the batch is being processed by another RPG III program as in some of the previous approaches.

Data Description Specifications for MLG105D

The DDS for the display file MLG105D are shown in Figure 10-5.

The DDS could be generated using the screen design aid, which was introduced in Chapter 8 and is described in detail in the *SDA Reference Manual and User's Guide*.



IBM International Business Machines Corporation

DATA DESCRIPTION SPECIFICATIONS

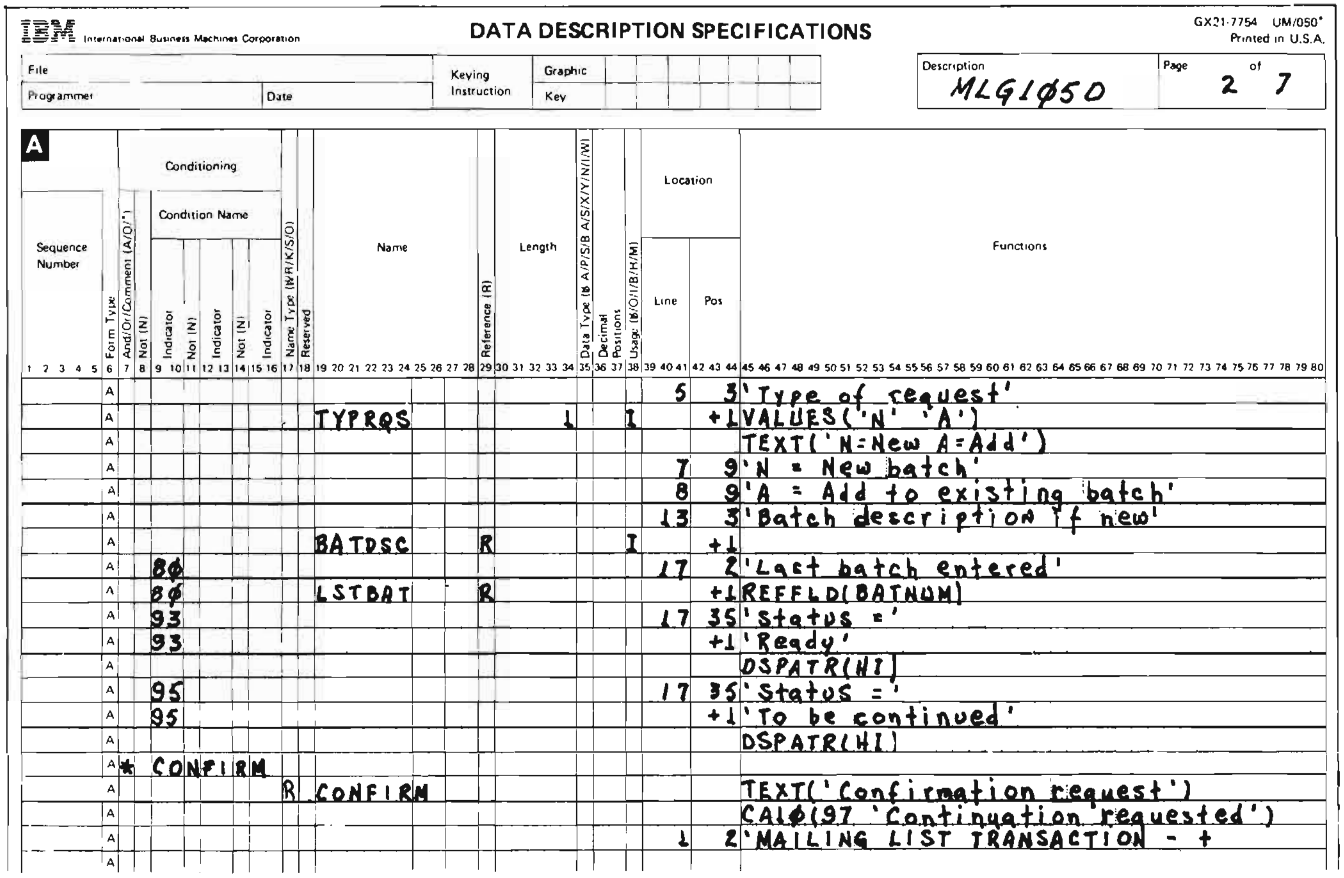
GX21 7754 UM/050*
Printed in U.S.A.

File | Programmer | Date | Keying Instruction | Graphic | Key | Description MLG1Ø5D | Page 1 of 7

```
     A**
     A** DISPLAY   MLG1Ø5D    MAILING LIST TRANSACTION ENTRY DISPLAY
     A**
     A                                      REF(MLGREFP)
     A                                      PRINT
     A          R INITIAL                   TEXT('Initial prompt')
     A                                      CAØ1(98 'End of program')
     A                                  1  2'MAILING LIST TRANSACTIONS'
     A                                      DSPATR(HI)
     A                                  1 55'CF1-End of program'
     A                                  3  5'Batch number'
     A            BATNUM    R        I     +1
     A  71                                  ERRMSG('Batch number does not +
     A                                       exist' 71)
     A  72                                  ERRMSG('Duplicate batch +
     A                                      number' 72)
     A  74                                  ERRMSG('Status of the batch +
     A                                       does not allow any +
     A                                       additions' 74)
     A
```

*Number of sheets per pad may vary slightly

Figure 10-5 (Part 1 of 4). DDS for MLG105D Display File



IBM International Business Machines Corporation

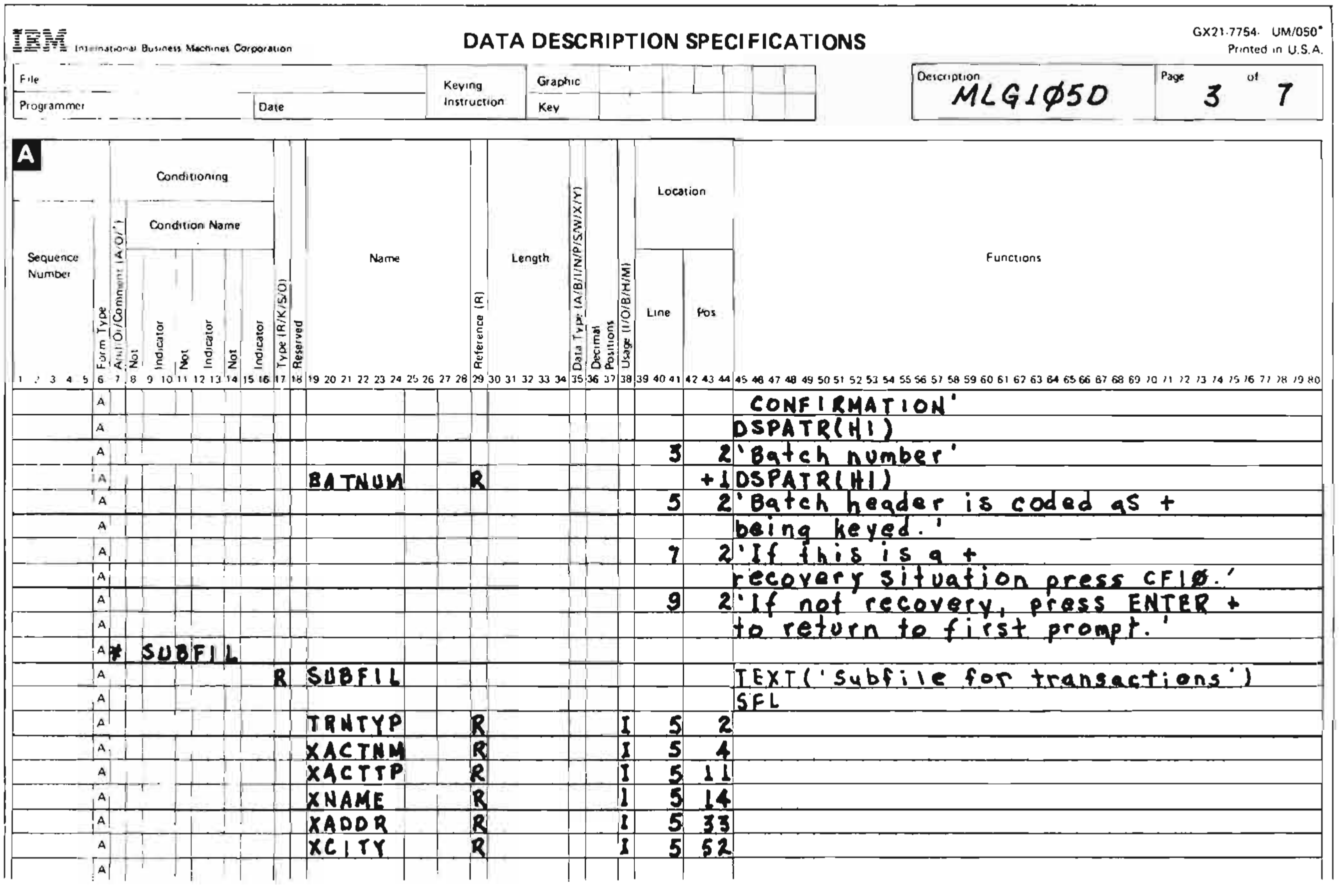
DATA DESCRIPTION SPECIFICATIONS

GX21-7754 UM/050*
Printed in U.S.A.

File | Keying Instruction | Graphic | Key | Programmer | Date

Description MLG1Ø5D   Page 2 of 7

```
     A                                  5  3'Type of request'
     A            TYPRQS         1   I    +1VALUES('N' 'A')
     A                                      TEXT('N=New A=Add')
     A                                  7  9'N = New batch'
     A                                  8  9'A = Add to existing batch'
     A                                 13  3'Batch description if new'
     A            BATDSC    R        I    +1
     A  8Ø                             17  2'Last batch entered'
     A  8Ø        LSTBAT    R             +1REFFLD(BATNUM)
     A  93                             17 35'Status ='
     A  93                                +1'Ready'
     A                                      DSPATR(HI)
     A  95                             17 35'Status ='
     A  95                                +1'To be continued'
     A                                      DSPATR(HI)
     A* CONFIRM
     A          R CONFIRM                   TEXT('Confirmation request')
     A                                      CA1Ø(97 'Continuation requested')
     A                                  1  2'MAILING LIST TRANSACTION - +
     A
```



IBM International Business Machines Corporation

DATA DESCRIPTION SPECIFICATIONS

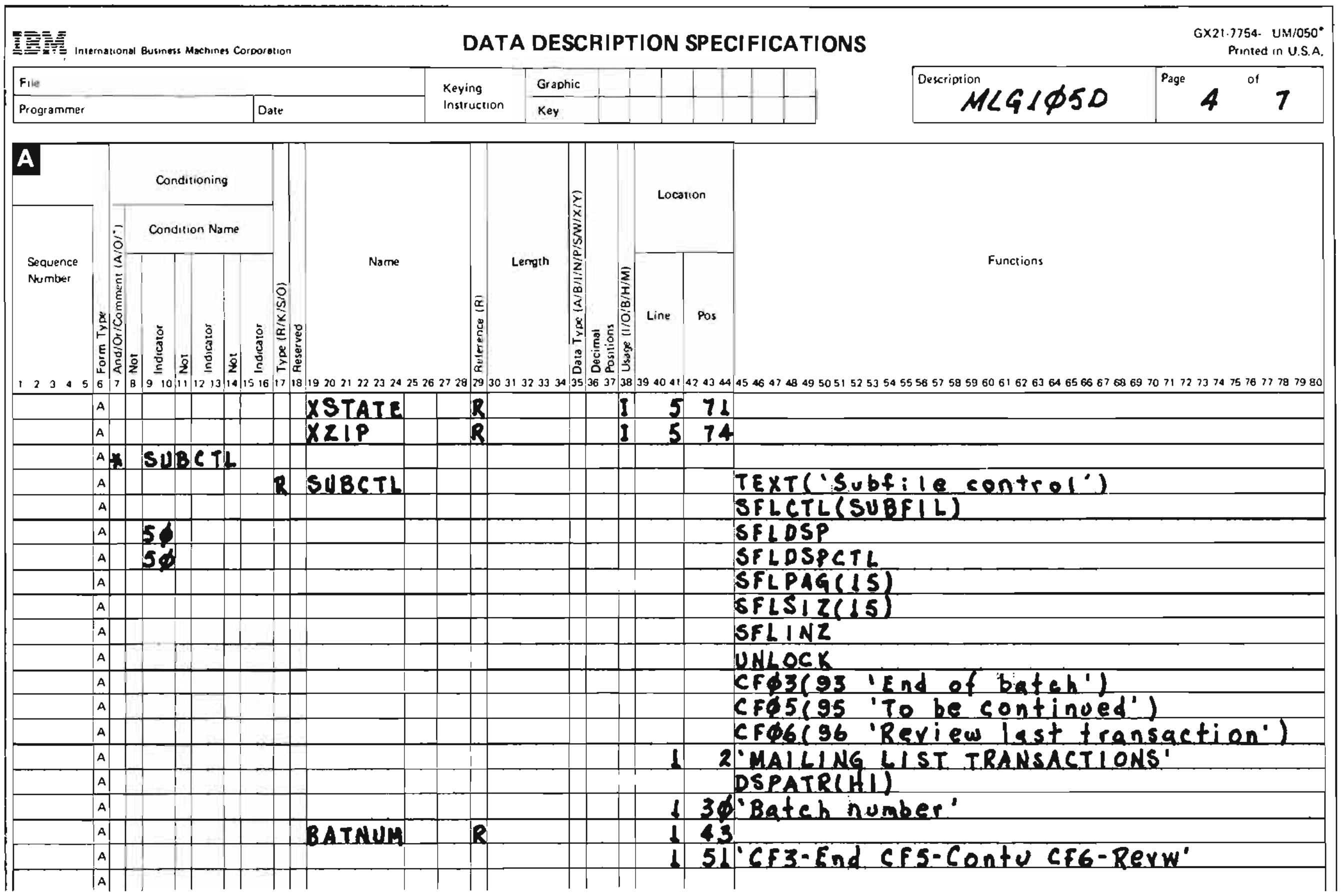
GX21-7754 UM/050*
Printed in U.S.A.

File | Keying Instruction | Graphic | Key | Programmer | Date

Description MLG1Ø5D   Page 3 of 7

```
     A                                       CONFIRMATION'
     A                                      DSPATR(HI)
     A                                  3  2'Batch number'
     A            BATNUM    R             +1DSPATR(HI)
     A                                  5  2'Batch header is coded as +
     A                                      being keyed.'
     A                                  7  2'If this is a +
     A                                      recovery situation press CF1Ø.'
     A                                  9  2'If not recovery, press ENTER +
     A                                      to return to first prompt.'
     A* SUBFIL
     A          R SUBFIL                    TEXT('Subfile for transactions')
     A                                      SFL
     A            TRNTYP    R        I  5  2
     A            XACTNM    R        I  5  4
     A            XACTTP    R        I  5 11
     A            XNAME     R        I  5 14
     A            XADDR     R        I  5 33
     A            XCITY     R        I  5 52
     A
```

Figure 10-5 (Part 2 of 4). DDS for MLG105D Display File



IBM International Business Machines Corporation

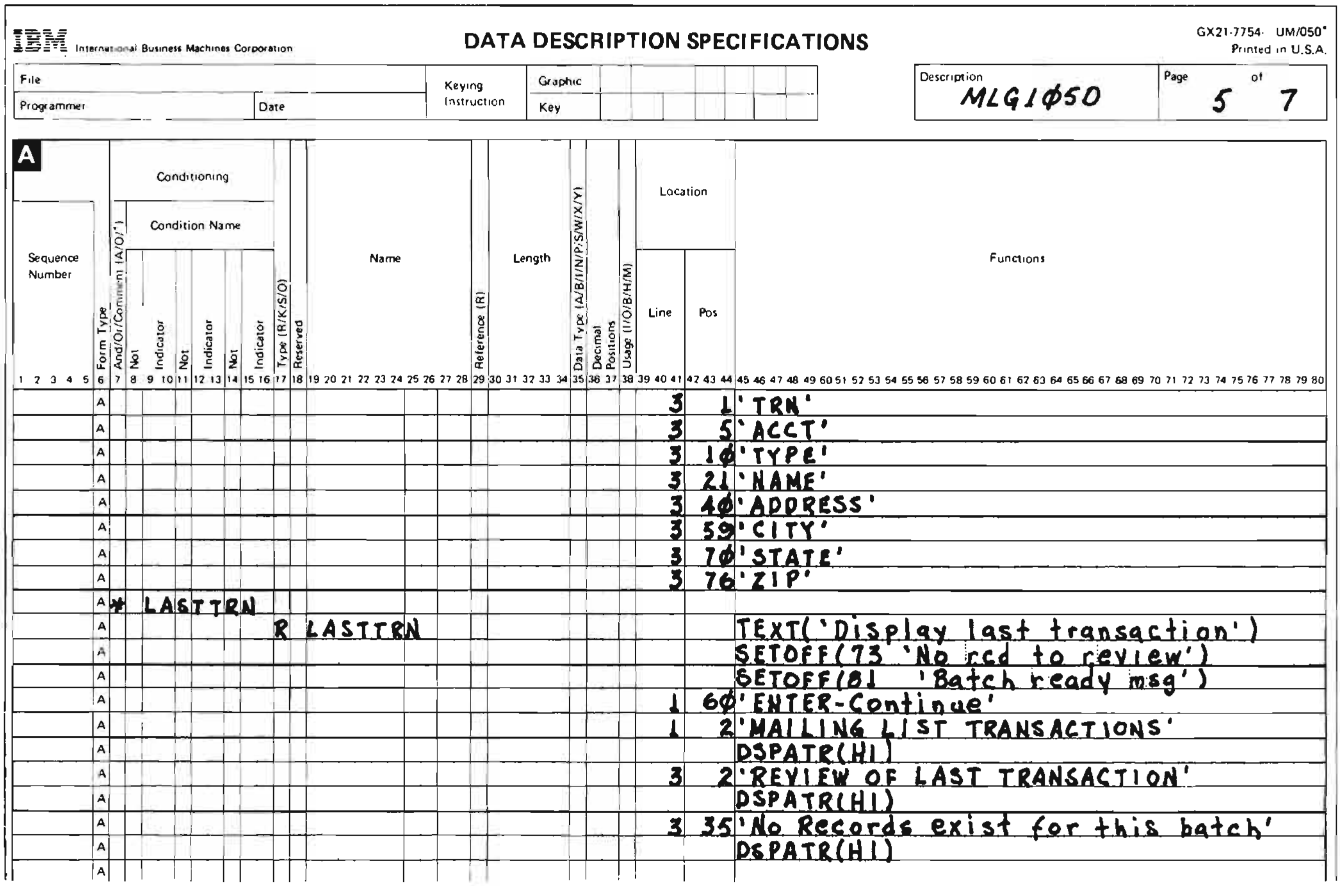
DATA DESCRIPTION SPECIFICATIONS

GX21-7754- UM/050*
Printed in U.S.A.

File
Programmer
Date
Keying Instruction
Graphic
Key

Description: MLG1Ø5D
Page 4 of 7

Form Type	And/Or/Comment	Condition Name / Indicator	Type (R/K/S/O)	Name	Reference (R)	Usage	Line	Pos	Functions
A				XSTATE	R	I	5	71	
A				XZIP	R	I	5	74	
A	*	SUBCTL							
A			R	SUBCTL					TEXT('Subfile control')
A									SFLCTL(SUBFIL)
A		5Ø							SFLDSP
A		5Ø							SFLDSPCTL
A									SFLPAG(15)
A									SFLSIZ(15)
A									SFLINZ
A									UNLOCK
A									CFØ3(93 'End of batch')
A									CFØ5(95 'To be continued')
A									CFØ6(96 'Review last transaction')
A							1	2	'MAILING LIST TRANSACTIONS'
A									DSPATR(HI)
A							1	3Ø	'Batch number'
A				BATNUM	R		1	43	
A							1	51	'CF3-End CF5-Contu CF6-Revw'
A									



IBM International Business Machines Corporation

DATA DESCRIPTION SPECIFICATIONS

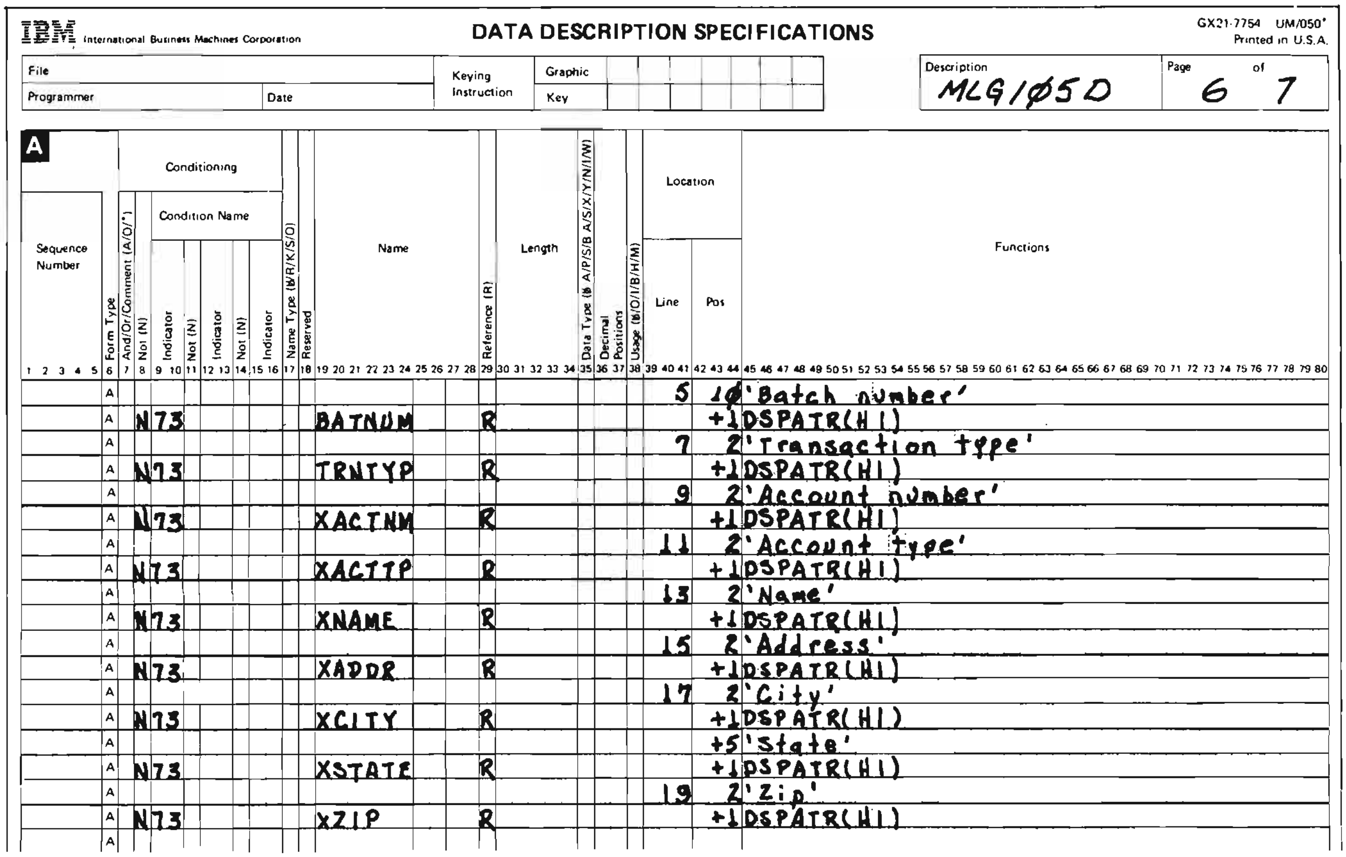
GX21-7754- UM/050*
Printed in U.S.A.

File
Programmer
Date
Keying Instruction
Graphic
Key

Description: MLG1Ø5O
Page 5 of 7

Form Type	And/Or/Comment	Condition Name / Indicator	Type (R/K/S/O)	Name	Reference (R)	Usage	Line	Pos	Functions
A							3	1	'TRN'
A							3	5	'ACCT'
A							3	1Ø	'TYPE'
A							3	21	'NAME'
A							3	4Ø	'ADDRESS'
A							3	59	'CITY'
A							3	7Ø	'STATE'
A							3	76	'ZIP'
A	*	LASTTRN							
A			R	LASTTRN					TEXT('Display last transaction')
A									SETOFF(73 'No rcd to review')
A									SETOFF(81 'Batch ready msg')
A							1	6Ø	'ENTER-Continue'
A							1	2	'MAILING LIST TRANSACTIONS'
A									DSPATR(HI)
A							3	2	'REVIEW OF LAST TRANSACTION'
A									DSPATR(HI)
A							3	35	'No Records exist for this batch'
A									DSPATR(HI)
A									

Figure 10-5 (Part 3 of 4). DDS for MLG105D Display File



IBM International Business Machines Corporation

DATA DESCRIPTION SPECIFICATIONS

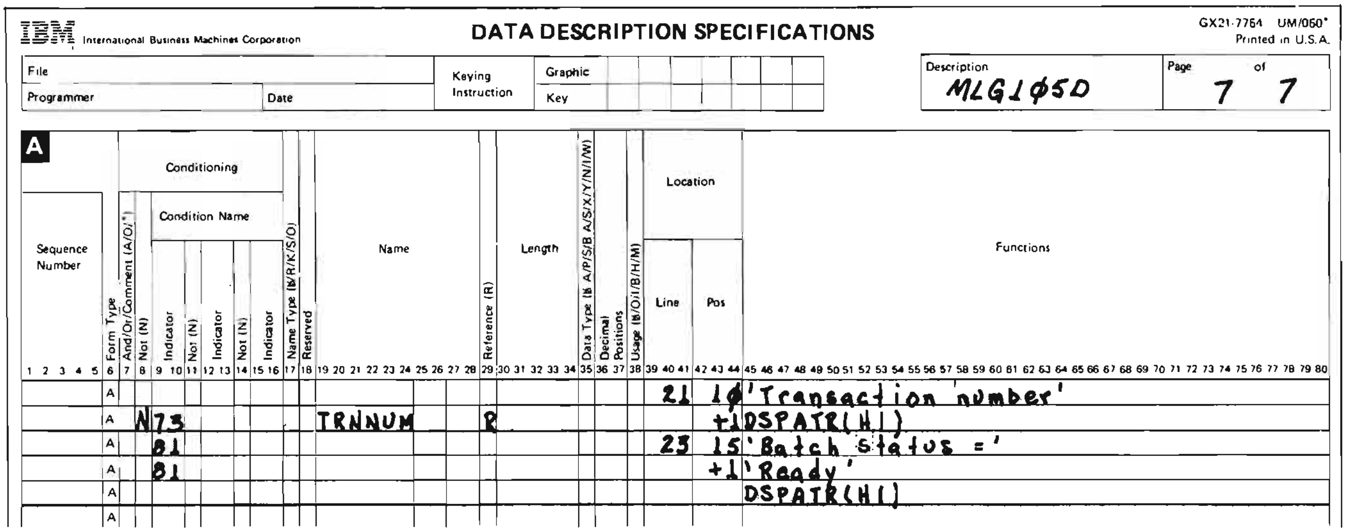
GX21-7754 UM/050* Printed in U.S.A.

File
Programmer
Date
Keying Instruction
Graphic
Key

Description MLG1Ø5D

Page 6 of 7

Form Type	Not (N)	Indicator	Name	Reference (R)	Line	Pos	Functions
A					5	1Ø	'Batch number'
A	N	73	BATNUM	R		+1	DSPATR(HI)
A					7	2	'Transaction type'
A	N	73	TRNTYP	R		+1	DSPATR(HI)
A					9	2	'Account number'
A	N	73	XACTNM	R		+1	DSPATR(HI)
A					11	2	'Account type'
A	N	73	XACTTP	R		+1	DSPATR(HI)
A					13	2	'Name'
A	N	73	XNAME	R		+1	DSPATR(HI)
A					15	2	'Address'
A	N	73	XADDR	R		+1	DSPATR(HI)
A					17	2	'City'
A	N	73	XCITY	R		+1	DSPATR(HI)
A						+5	'State'
A	N	73	XSTATE	R		+1	DSPATR(HI)
A					19	2	'Zip'
A	N	73	XZIP	R		+1	DSPATR(HI)
A							



IBM International Business Machines Corporation

DATA DESCRIPTION SPECIFICATIONS

GX21-7754 UM/050* Printed in U.S.A.

File
Programmer
Date
Keying Instruction
Graphic
Key

Description MLG1Ø5D

Page 7 of 7

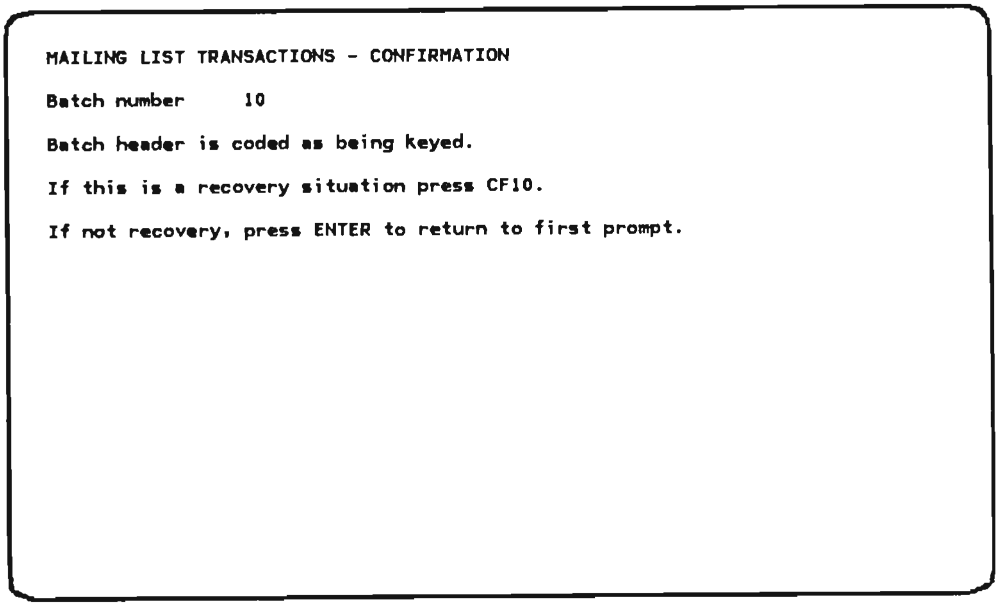
Form Type	Not (N)	Indicator	Name	Reference (R)	Line	Pos	Functions
A					21	1Ø	'Transaction number'
A	N	73	TRNNUM	R		+1	DSPATR(HI)
A		81			23	15	'Batch status ='
A		81				+1	'Ready'
A							DSPATR(HI)
A							

Figure 10-5 (Part 4 of 4). DDS for MLG105D Display File

The *REF keyword* specifies use of the field reference file MLGREFP to minimize coding.

The *name INITIAL* specifies the format for the prompt that requests batch header data. The name TYPRQS specifies a validity check for the request entry made by the work station user. This eliminates checking for a valid request in the program.

The *name CONFIRM* specifies the format for the prompt that confirms the work station user has requested to work on a batch with a status code of 1 (data entry is in process). It looks like this:



```
MAILING LIST TRANSACTIONS - CONFIRMATION

Batch number     10

Batch header is coded as being keyed.

If this is a recovery situation press CF10.

If not recovery, press ENTER to return to first prompt.
```

This confirmation is necessary because the program does not distinguish between a batch that has data being entered into it and a recovery situation. A recovery situation may occur if either the system or the program fails and the batch header record is not updated.

The *record format names SUBFIL (subfile) and SUBCTL (subfile control)* specify the format for the prompt that allows the work station user to enter transactions. SUBFIL is designated by the keyword SFL and specifies the position for the first subfile transaction record on the display. Data is entered into fields that are also in the transaction file MLGENTL. The Rs entered in position 29 specify using the field reference file for the definitions of each of these fields. (The field reference file, MLGREFP, was specified at the beginning of the DDS for this file.)

The SUBCTL format is associated with the subfile (SUBFIL) by the keyword SFLCTL. SUBCTL specifies the format of the subfile with constants and fields. Keywords that define and control the subfile are also listed here. The subfile and the subfile control are displayed each time the SUBCTL format is written because indicator 50 is set on at the beginning of the RPG III program (MLG105).

The keywords SFLPAG and SFLSIZ specify that 15 records will be on a page and there is only one page. The SFLINZ keyword initializes all fields in the subfile so that the work station user can enter data into the fields.

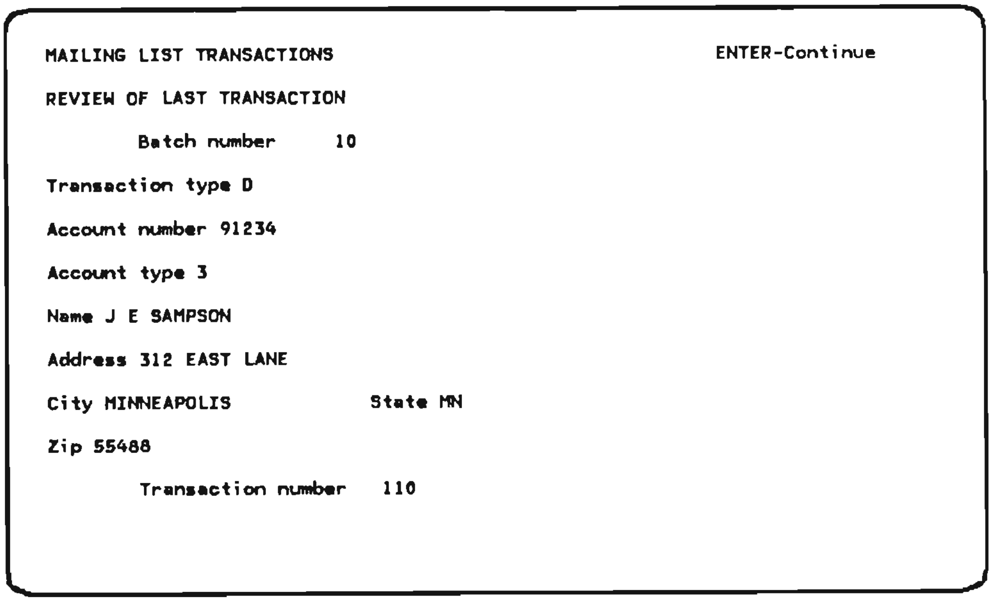
The keyword UNLOCK specifies that the keyboard is unlocked when the system accepts data. If UNLOCK is not specified, the keyboard is not unlocked until the program sends a format to the display. Key entry throughput is improved by specifying UNLOCK; however, the program cannot check the data for validity and send an error response because the work station user may already be keying in the next group of transactions.

Functions for the following three command keys are also specified:

- CF3 indicates the end of a completed batch.
- CF5 indicates that the work station user is stopping work on this batch (the user may sign off) and will continue work on this batch at a later time. The batch should not be processed by the maintenance program because it is not complete.
- CF6 indicates that the work station user needs to see the last transaction entered. This can be used during transaction entry and is the same function that is automatically invoked upon returning to a batch to continue entering transactions.

Note that the three command keys are coded CFxx and not CAxx. CFxx coding sends the data that is on the display *and* the key number that was pressed to the system. The program first processes the data from the display and then tests for whether a command key was or was not pressed. The CAxx command keys send only the key number (not any data); any transactions entered by the work station user would be lost.

The *name LASTTRN* specifies the format that is used to review the last transaction that was keyed in; the format looks like this:



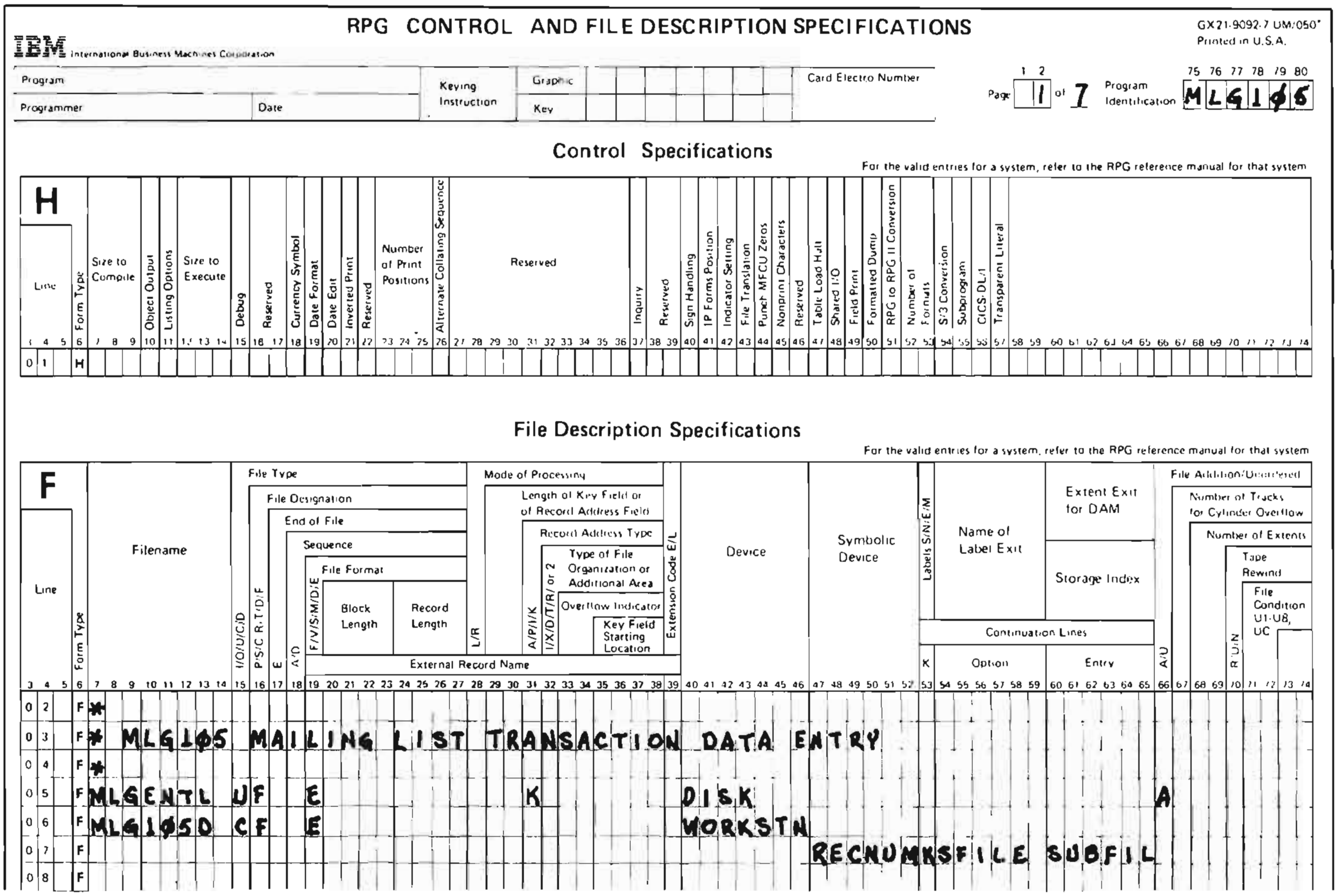
```
MAILING LIST TRANSACTIONS                                    ENTER-Continue

REVIEW OF LAST TRANSACTION

        Batch number     10

Transaction type D

Account number 91234

Account type 3

Name J E SAMPSON

Address 312 EAST LANE

City MINNEAPOLIS            State MN

Zip 55488

        Transaction number   110
```

The work station user sees this review display in the following situations:

- When specifically requesting to review the last transaction.
- When requesting to add transactions to an existing batch.
  - The work station user previously terminated the batch in a *to be continued* status, or
  - A system or program failure occurred and the operator is requesting recovery.

RPG III Specifications for MLG105

The RPG III specifications for the transaction data entry program are shown in Figures 10-6 through 10-8.



IBM International Business Machines Corporation

RPG CONTROL AND FILE DESCRIPTION SPECIFICATIONS

GX21-9092-7 UM/050* Printed in U.S.A.

Page 1 of 7 Program Identification MLG1Ø5

Control Specifications

For the valid entries for a system, refer to the RPG reference manual for that system

```
01 H
```

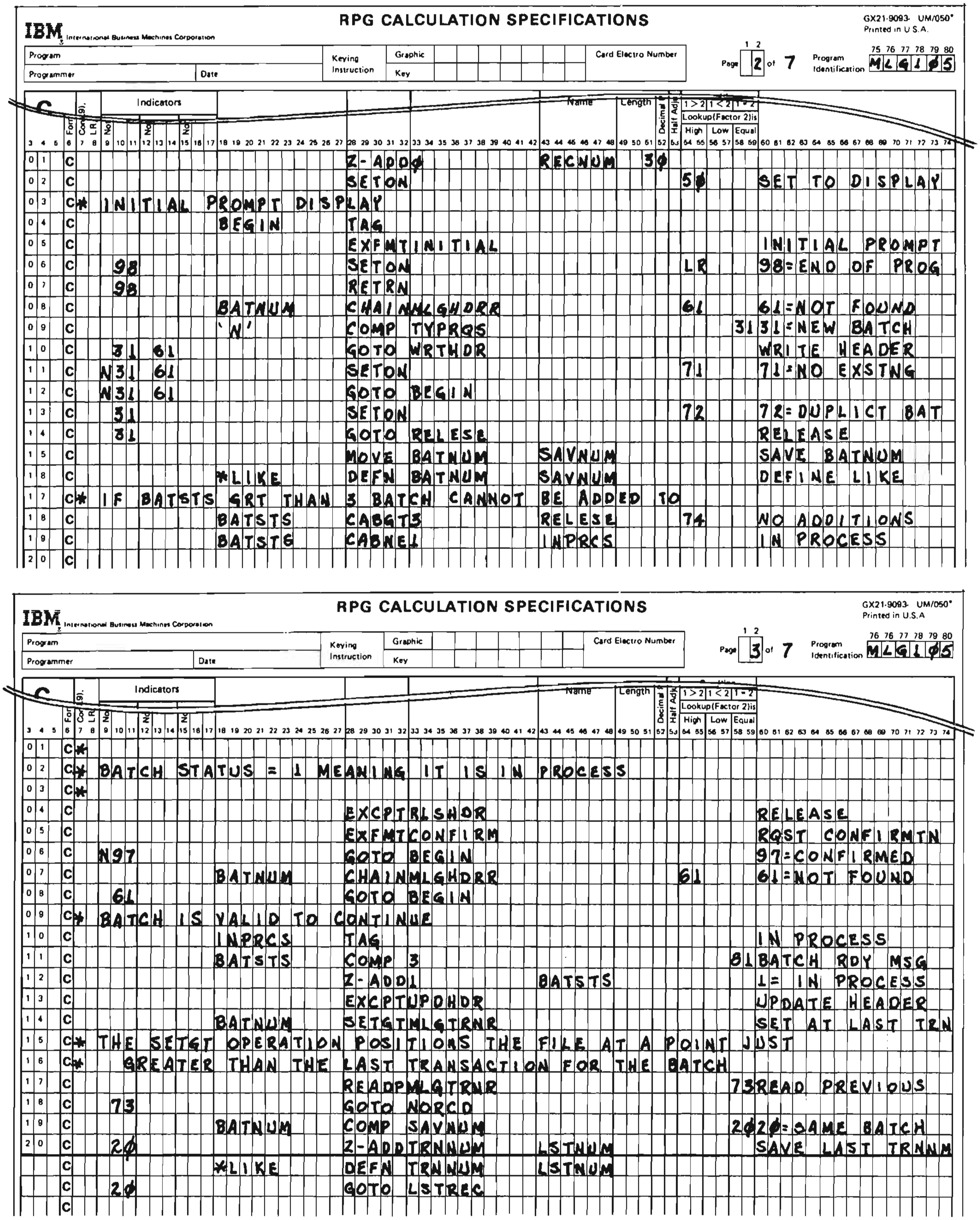
File Description Specifications

For the valid entries for a system, refer to the RPG reference manual for that system

```
02 F*
03 F* MLG1Ø5 MAILING LIST TRANSACTION DATA ENTRY
04 F*
05 FMLGENTL UF  E           K        DISK                      A
06 FMLG1Ø5D CF  E                    WORKSTN
07 F                                        RECNUMKSFILE SUBFIL
08 F
```

Figure 10-6. Control and File Description Specifications for MLG105 Transaction Data Entry Program

RPG Control and File Description Specifications, shown in Figure 10-6, describe two files: MLGENTL and MLG105D. The U in position 15 specifies that MLGENTL is updated by the program and the A in position 66 specifies that records are added to MLGENTL during execution of the program. The K in position 53 (of the last line) and the SFILE keyword specify that MLG105D uses a subfile. RECNUM is the name of the field that contains the relative record number when records are written to the subfile named SUBFIL.



IBM International Business Machines Corporation

RPG CALCULATION SPECIFICATIONS

GX21-9093- UM/050* Printed in U.S.A.

Page 2 of 7 Program Identification MLG1∅5

```
01 C                     Z-ADD∅         RECNUM  3∅
02 C                     SETON                     5∅    SET TO DISPLAY
03 C*  INITIAL PROMPT DISPLAY
04 C           BEGIN     TAG
05 C                     EXFMTINITIAL                    INITIAL PROMPT
06 C   98                SETON                     LR    98=END OF PROG
07 C   98                RETRN
08 C           BATNUM    CHAINMLGHDRR              61    61=NOT FOUND
09 C           'N'       COMP  TYPRQS                  3131=NEW BATCH
10 C   31 61             GOTO  WRTHDR                    WRITE HEADER
11 C  N31 61             SETON                     71    71=NO EXSTNG
12 C  N31 61             GOTO  BEGIN
13 C   31                SETON                     72    72=DUPLICT BAT
14 C   31                GOTO  RELESE                    RELEASE
15 C                     MOVE  BATNUM    SAVNUM          SAVE BATNUM
16 C           *LIKE     DEFN  BATNUM    SAVNUM          DEFINE LIKE
17 C* IF BATSTS GRT THAN 3 BATCH CANNOT BE ADDED TO
18 C           BATSTS    CABGT3          RELESE     74    NO ADDITIONS
19 C           BATSTS    CABNE1          INPRCS          IN PROCESS
20 C
```

IBM International Business Machines Corporation

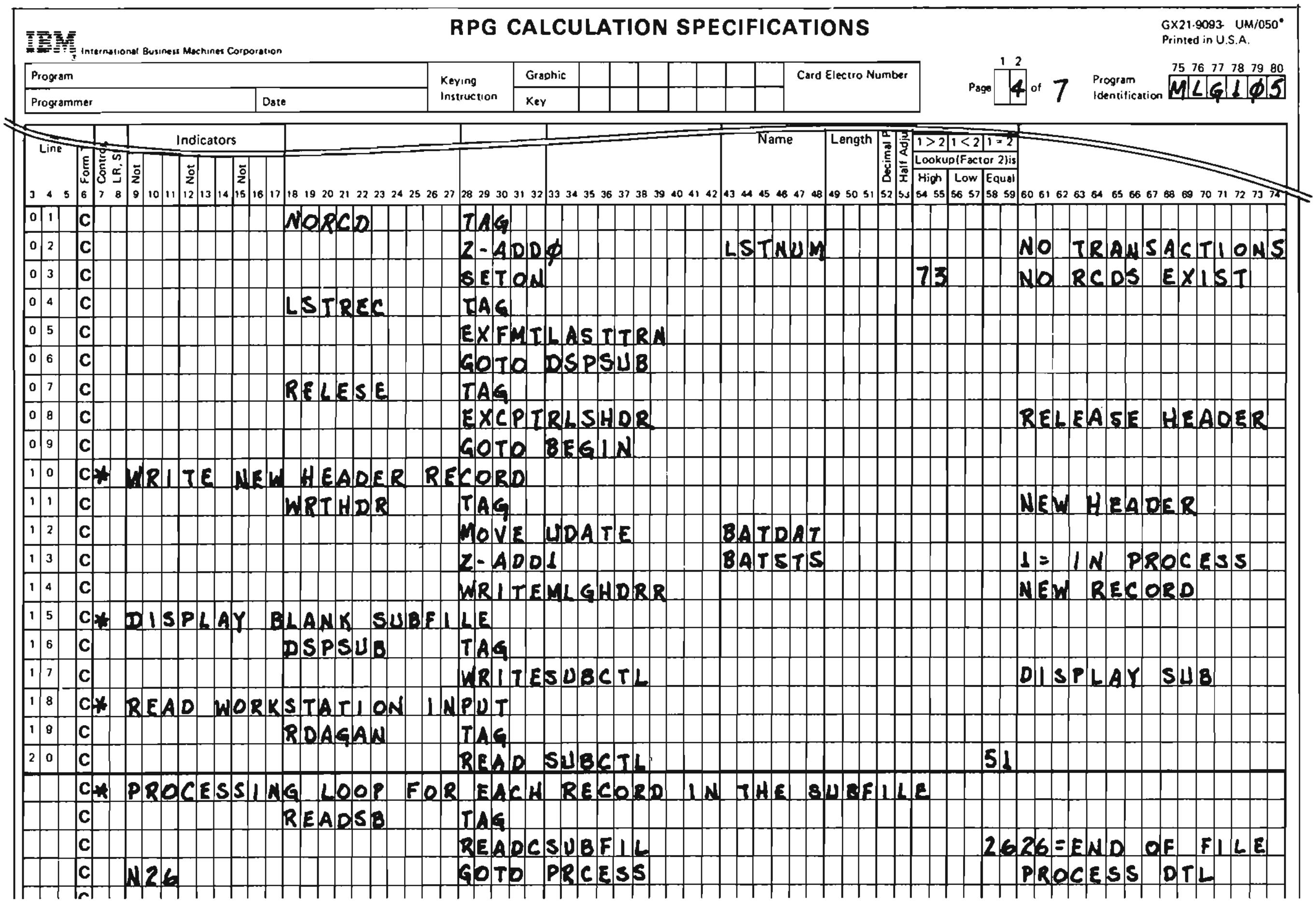
RPG CALCULATION SPECIFICATIONS

GX21-9093- UM/050* Printed in U.S.A.

Page 3 of 7 Program Identification MLG1∅5

```
01 C*
02 C* BATCH STATUS = 1 MEANING IT IS IN PROCESS
03 C*
04 C                     EXCPTRLSHDR                     RELEASE
05 C                     EXFMTCONFIRM                    RQST CONFIRMTN
06 C  N97                GOTO  BEGIN                     97=CONFIRMED
07 C           BATNUM    CHAINMLGHDRR              61    61=NOT FOUND
08 C   61                GOTO  BEGIN
09 C* BATCH IS VALID TO CONTINUE
10 C           INPRCS    TAG                             IN PROCESS
11 C           BATSTS    COMP  3                       81BATCH RDY MSG
12 C                     Z-ADD1          BATSTS          1= IN PROCESS
13 C                     EXCPTUPDHDR                     UPDATE HEADER
14 C           BATNUM    SETGTMLGTRNR                    SET AT LAST TRN
15 C* THE SETGT OPERATION POSITIONS THE FILE AT A POINT JUST
16 C*    GREATER THAN THE LAST TRANSACTION FOR THE BATCH
17 C                     READPMLGTRNR                  73READ PREVIOUS
18 C   73                GOTO  NORCD
19 C           BATNUM    COMP  SAVNUM                  2∅2∅=SAME BATCH
20 C   2∅                Z-ADDTRNNUM     LSTNUM          SAVE LAST TRNNM
   C           *LIKE     DEFN  TRNNUM    LSTNUM
   C   2∅                GOTO  LSTREC
   C
```

Figure 10-7 (Part 1 of 3). Calculation Specifications for MLG105 Transaction Data Entry Program



IBM International Business Machines Corporation

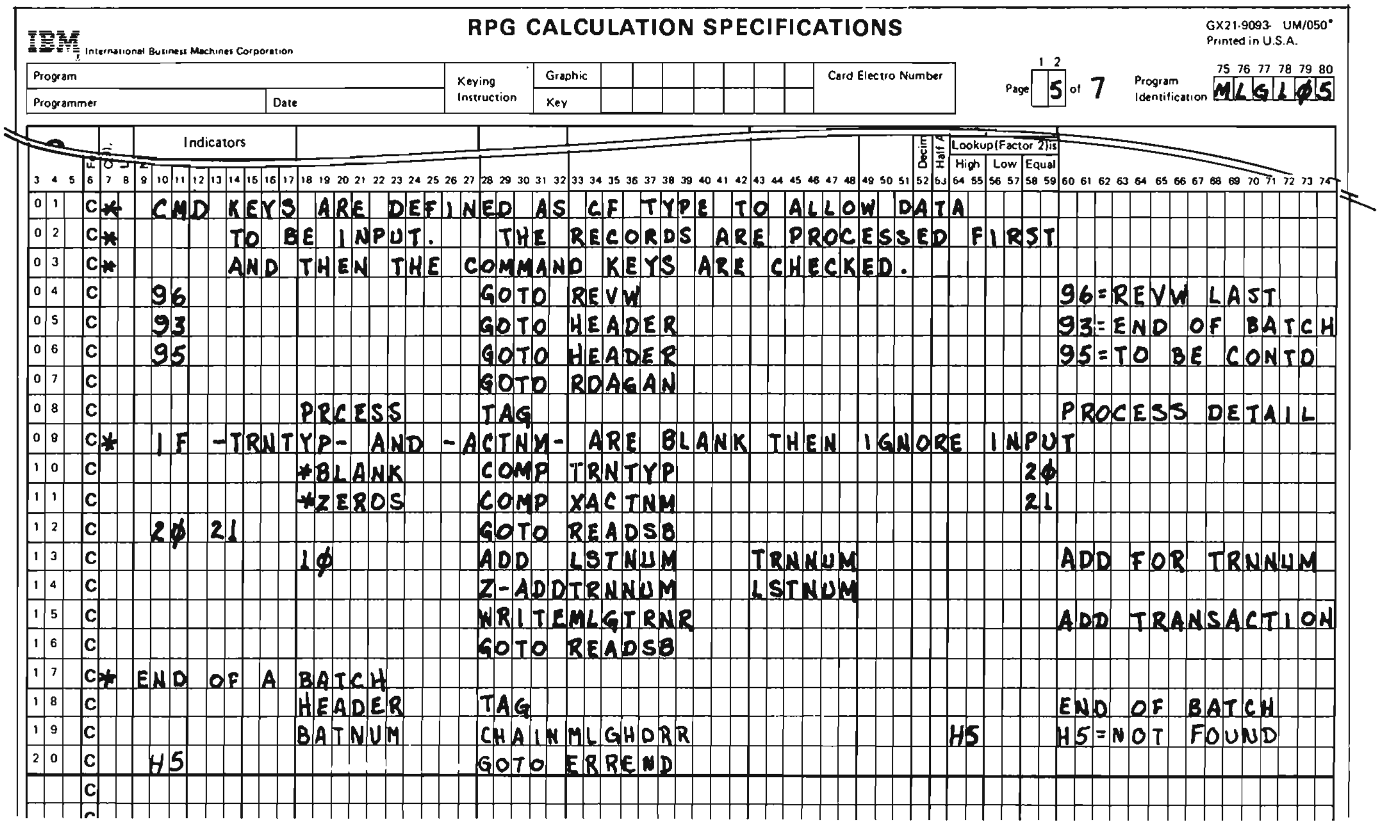
RPG CALCULATION SPECIFICATIONS

GX21-9093- UM/050*
Printed in U.S.A.

Program | Programmer | Date | Keying Instruction | Graphic | Key | Card Electro Number

Page 4 of 7 | Program Identification MLG1Ø5

```
01 C           NORCD     TAG
02 C                     Z-ADDØ         LSTNUM           NO TRANSACTIONS
03 C                     SETON                     73    NO RCDS EXIST
04 C           LSTREC    TAG
05 C                     EXFMTLASTTRN
06 C                     GOTO DSPSUB
07 C           RELESE    TAG
08 C                     EXCPTRLSHDR                     RELEASE HEADER
09 C                     GOTO BEGIN
10 C* WRITE NEW HEADER RECORD
11 C           WRTHDR    TAG                             NEW HEADER
12 C                     MOVE UDATE     BATDAT
13 C                     Z-ADD1         BATSTS           1= IN PROCESS
14 C                     WRITEMLGHDRR                    NEW RECORD
15 C* DISPLAY BLANK SUBFILE
16 C           DSPSUB    TAG
17 C                     WRITESUBCTL                     DISPLAY SUB
18 C* READ WORKSTATION INPUT
19 C           RDAGAN    TAG
20 C                     READ SUBCTL                 51
   C* PROCESSING LOOP FOR EACH RECORD IN THE SUBFILE
   C           READSB    TAG
   C                     READCSUBFIL                 2626=END OF FILE
   C   N26               GOTO PRCESS                     PROCESS DTL
```



IBM International Business Machines Corporation

RPG CALCULATION SPECIFICATIONS

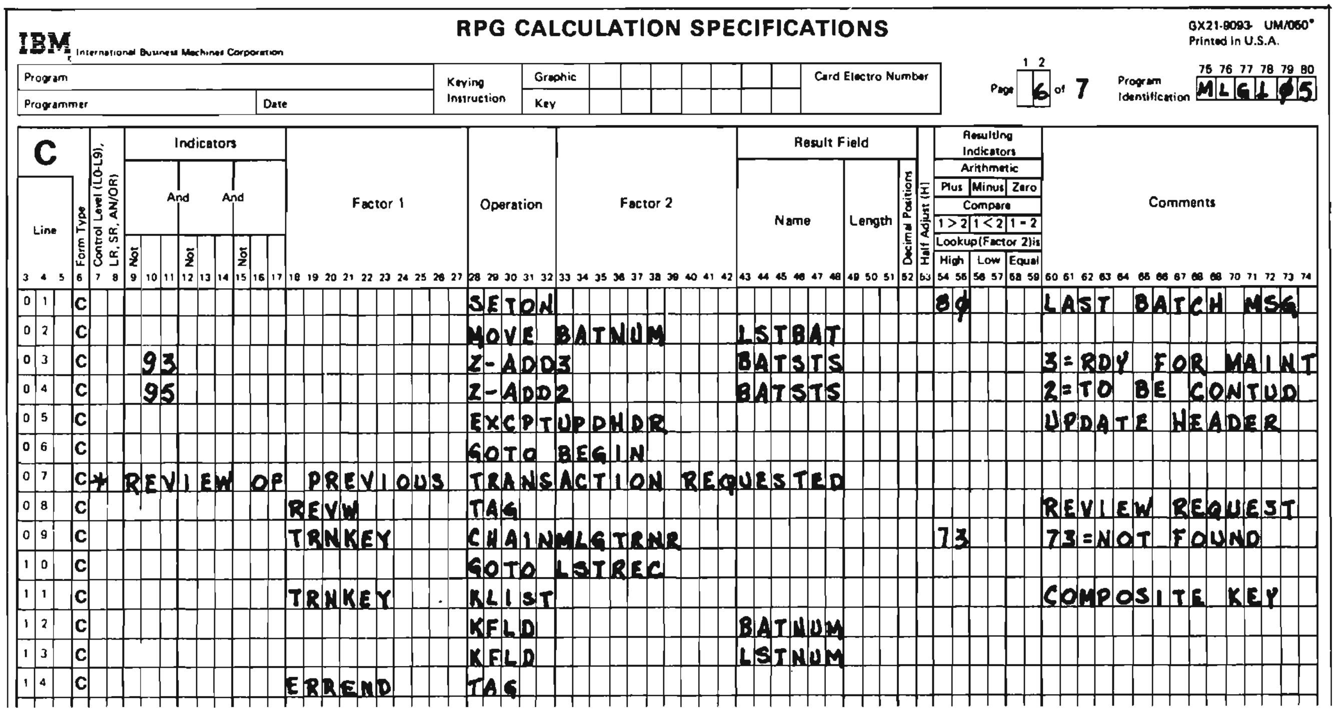
GX21-9093- UM/050*
Printed in U.S.A.

Program | Programmer | Date | Keying Instruction | Graphic | Key | Card Electro Number

Page 5 of 7 | Program Identification MLG1Ø5

```
01 C*  CMD KEYS ARE DEFINED AS CF TYPE TO ALLOW DATA
02 C*      TO BE INPUT.   THE RECORDS ARE PROCESSED FIRST
03 C*      AND THEN THE COMMAND KEYS ARE CHECKED.
04 C   96                GOTO REVW                       96=REVW LAST
05 C   93                GOTO HEADER                     93=END OF BATCH
06 C   95                GOTO HEADER                     95=TO BE CONTD
07 C                     GOTO RDAGAN
08 C           PRCESS    TAG                             PROCESS DETAIL
09 C*  IF -TRNTYP- AND -ACTNM- ARE BLANK THEN IGNORE INPUT
10 C           *BLANK    COMP TRNTYP                 2Ø
11 C           *ZEROS    COMP XACTNM                 21
12 C   2Ø 21             GOTO READSB
13 C           1Ø        ADD  LSTNUM    TRNNUM           ADD FOR TRNNUM
14 C                     Z-ADDTRNNUM    LSTNUM
15 C                     WRITEMLGTRNR                    ADD TRANSACTION
16 C                     GOTO READSB
17 C* END OF A BATCH
18 C           HEADER    TAG                             END OF BATCH
19 C           BATNUM    CHAINMLGHDRR              H5    H5=NOT FOUND
20 C   H5                GOTO ERREND
```

Figure 10-7 (Part 2 of 3). Calculation Specifications for MLG105 Transaction Data Entry Program



IBM International Business Machines Corporation

RPG CALCULATION SPECIFICATIONS

GX21-9093 UM/050*
Printed in U.S.A.

Program | Programmer | Date | Keying Instruction | Graphic | Key | Card Electro Number

Page 6 of 7 — Program Identification MLG1Ø5

```
                    Factor 1  Operation  Factor 2  Result  Resulting  Comments
                                                   Name    Indicators
01 C                     SETON                     8Ø    LAST BATCH MSG
02 C                     MOVE BATNUM    LSTBAT
03 C   93                Z-ADD3         BATSTS           3=RDY FOR MAINT
04 C   95                Z-ADD2         BATSTS           2=TO BE CONTUD
05 C                     EXCPTUPDHDR                     UPDATE HEADER
06 C                     GOTO BEGIN
07 C* REVIEW OF PREVIOUS TRANSACTION REQUESTED
08 C           REVW      TAG                             REVIEW REQUEST
09 C           TRNKEY    CHAINMLGTRNR              73    73=NOT FOUND
10 C                     GOTO LSTREC
11 C           TRNKEY    KLIST                           COMPOSITE KEY
12 C                     KFLD           BATNUM
13 C                     KFLD           LSTNUM
14 C           ERREND    TAG
```

Figure 10-7 (Part 3 of 3). Calculation Specifications for MLG105 Transaction Data Entry Program

In the *Calculation Specifications*, shown in Figure 10-7, the processing of a batch begins with indicator 50 being set on. This indicator is used to specify that the subfile and subfile control formats should be displayed each time the subfile control format is written.

The EXFMT INITIAL statement displays the INITIAL prompt (which is described by the DDS in Figure 10-5) and the program waits for a response. If the response is a request to end the program, the program sets on the LR indicator and returns. It is important for RPG III programs to set the LR indicator on when the program is complete. This allows the working storage required by the program to be eliminated by the system. Use of the explicit RETRN operation code causes the program to return to where it was called. The RPG III program cycle still exists even with interactive programs. It is more straightforward to end the program explicitly using RETRN than to rely upon the RPG III program cycle to perform the return operation.

If the response is not a request to end the program, the batch number is used to attempt to access an existing batch header with the same batch number. Depending on the type of request (add or new), this attempt can result in one of the following error messages on the INITIAL prompt:

- Batch number does not exist (for add requests)
- Duplicate batch number (for new requests)

The CHAIN statement uses the batch number (BATNUM) in Factor 1. Factor 2 specifies that the batch header record (MLGHDRR) is accessed. The Factor 1 field used in the CHAIN statement must have the same attributes as the key field of the record format name being chained. Note that the record format name is being specified and not the file name. When the CHAIN statement is used, there must be an exact match between the key field attributes of Factor 1 and the record format.

If continuation is requested and the batch header exists, the batch status field is checked for a valid entry. If the batch status is greater than 3, the following error message is displayed on the INITIAL prompt:

Status of the batch does not allow any additions

If the batch status is 2 or 3, the batch may be continued; the batch header record updated, and processing continued. If the batch status is 1, the CONFIRM format (Figure 10-5) is displayed. The work station user needs to confirm his request because the program does not distinguish between the following:

- An invalid request by another work station user (to continue work on a batch that is currently being worked on)
- A recovery situation in which the batch header was not updated to have a status code of 2 or 3

A simpler design would not distinguish between a status code of 1 or 2; a more complex design would recognize a recovery situation and change all status codes of 1 to 2.

A batch header can exist without having transactions associated with it. When the work station user requests to review the last transaction entered in this batch, the LASTTRN format (Figure 10-5) is displayed with the following message:

No records exist for this batch

The LSTREC routine writes the LASTTRN format and the WRTHDR routine adds the new batch header to the file.

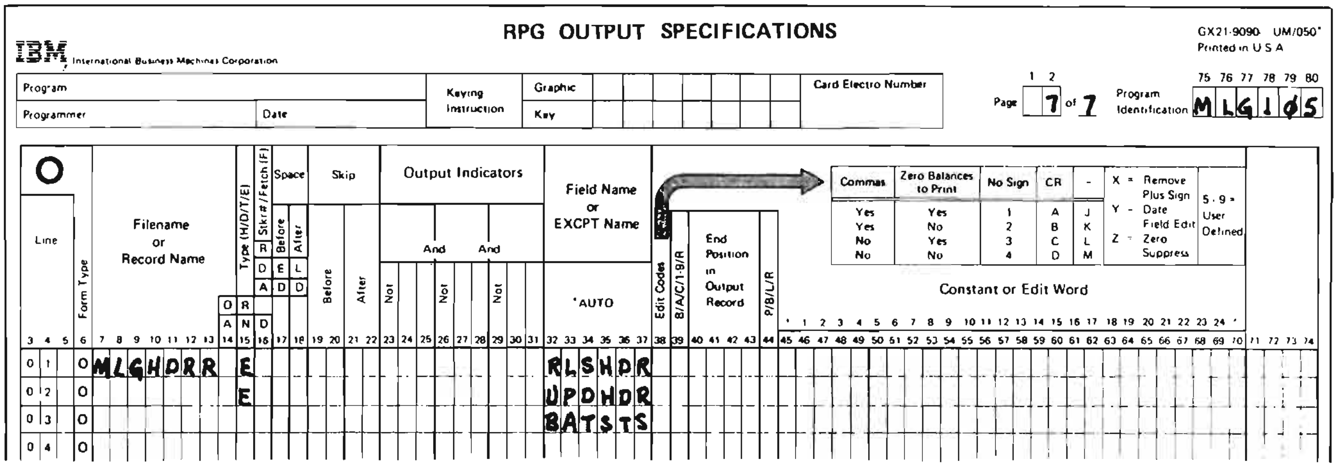
The DSPSUB routine writes to the display that has the format subfile control and subfile. A separate statement reads the format. EXFMT is not used because of the heads down environment. The loop begins with the operation READC, which requests to read the next changed record from the subfile. Only those records entered since the last write to the display are provided. When a record is entered by the work station user, it is considered active for the remainder of the program; consequently, the program tests the first two fields of the input record. If they are blank or have values of 0, the program assumes that no record was entered. A valid record from the subfile is written to the transaction file.

When there are no records remaining to be processed in the subfile, the program checks which command key was pressed. It is valid to press a command key whether records were or were not entered. The program loops back to read the subfile control if no command key was pressed. The work station user may have already made an entry on a display while the previous display is being processed because UNLOCK was specified for SUBCTL. If the work station user is still keying in data, the program waits for input. If the work station user has already pressed a command key or the Enter key, the work station device locks the keyboard and waits for the read request from the program.

When CF3 (end of batch) is pressed, the program accesses the batch header. If the batch header has been deleted, the program terminates and the halt indicator H5 is on. If the batch header is accessed successfully, its status code is updated to reflect the batch header's current status.

When CF6 (review) is pressed, the transaction key (which is the key fields BATNUM and .LSTNUM) is used to retrieve the last transaction record. If there is no transaction record, the following message is displayed:

No records exist for this batch



IBM International Business Machines Corporation

RPG OUTPUT SPECIFICATIONS

GX21-9090 UM/050*
Printed in U S A

Program
Programmer
Date
Keying Instruction
Graphic
Key
Card Electro Number

Page 7 of 7

Program Identification MLG1Ø5

O

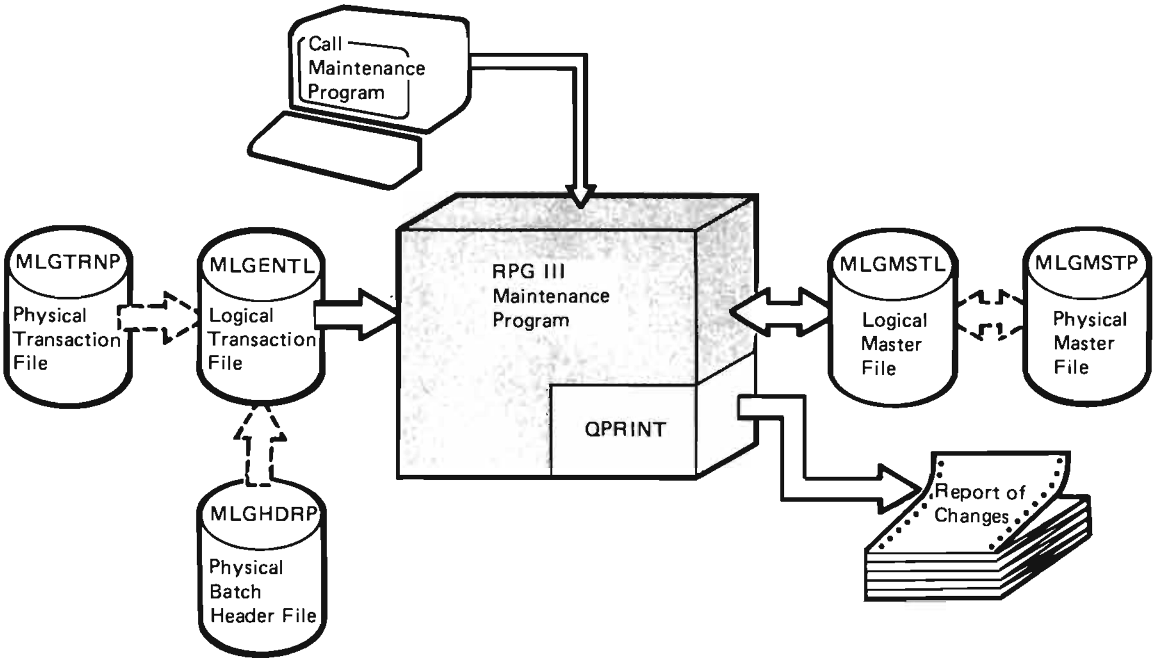
Commas	Zero Balances to Print	No Sign	CR	-	X = Remove Plus Sign Y = Date Field Edit Z = Zero Suppress	5 - 9 = User Defined
Yes	Yes	1	A	J		
Yes	No	2	B	K		
No	Yes	3	C	L		
No	No	4	D	M		

Line	Form Type	Filename or Record Name	Type (H/D/T/E)	Field Name or EXCPT Name
01	O	MLGHDRR	E	RLSHDR
02	O		E	UPDHDR
03	O			BATSTS
04	O			

Figure 10-8. Output Specifications for MLG105 Transaction Data Entry Program

APPLYING TRANSACTION BATCHES TO MASTER FILE

The batches of transactions in the transaction file are applied to the master file when a maintenance program is called. The maintenance program could be similar to the MLG311 program described in Chapter 7. In this case, however, a batch of transactions will be processed by the maintenance program only if the header record in the MLGHDRP batch header file indicates a batch status (BATSTS) of 3 (see *Batch Header Record* in this chapter).



PROGRAMMING CONSIDERATIONS

- No validity checking occurs during MLG105. This program is used for batch input and allows transactions to be entered efficiently. If there are any errors in the input, they are detected by the maintenance program that applies transactions to a master file.

- In addition to an application that has data entry, a batch header can be used in applications that do the following:
  - Edit
  - Allow the work station user to correct invalid transactions
  - Automatically update after a successful edit

  Various status codes can specify the status of each batch.

- In the approach discussed in this section, multiple work station users add records to the same file. The batch number separates the different batches in the file. Using one file with key fields to separate the logical groups is preferable to having a file or member for each batch, because it reduces overhead in the system.

Also in this approach, the batch header record was released at various points to remove the lock held by the program. When a file is coded as an update type, the system automatically locks the record until the update is performed or the lock is released. The lock prevents other users from retrieving the same record for update. Because of this, it is normally a good programming practice to retain locks for only a brief period of time in a data base system. Therefore, the batch header record was either updated immediately or the lock was released by doing an update by use of exception output with no fields being changed. Releasing a lock in the manner shown in the approach is more efficient than updating the record. If a request for a new batch is in error because the batch number already exists, a release of the existing header record occurs.

RECOVERY CONSIDERATIONS

If incorrect data has been entered for a transaction, the entire transaction should be reentered with correct data.

If there is a system failure while a work station user is doing key entry work, two ways to recover are:

- To specify a value of 1 for the force write ratio (FRCRATIO) parameter when creating the physical file for the batch header records; and to use the FEOD (force end of data) operation on the transaction file, MLGENTL, just before the batch header's status is updated (for completion). This will force the records to nonvolatile (auxiliary) storage.

- To use data base logging on both the batch header file and the transaction file. Specify a value of 1 for the force write ratio parameter when creating the log file and execute a recovery program at the next IMPL (following a system failure) to synchronize the log file and the data base files.

In either approach, the system operator does not need to determine who was using the system or help the work station users to restart. The work station users perform the recovery themselves in conjunction with the program design. Because the program provides for a specific work station user request to review the last transaction, the actual design and program code devoted to recovery is minimal.

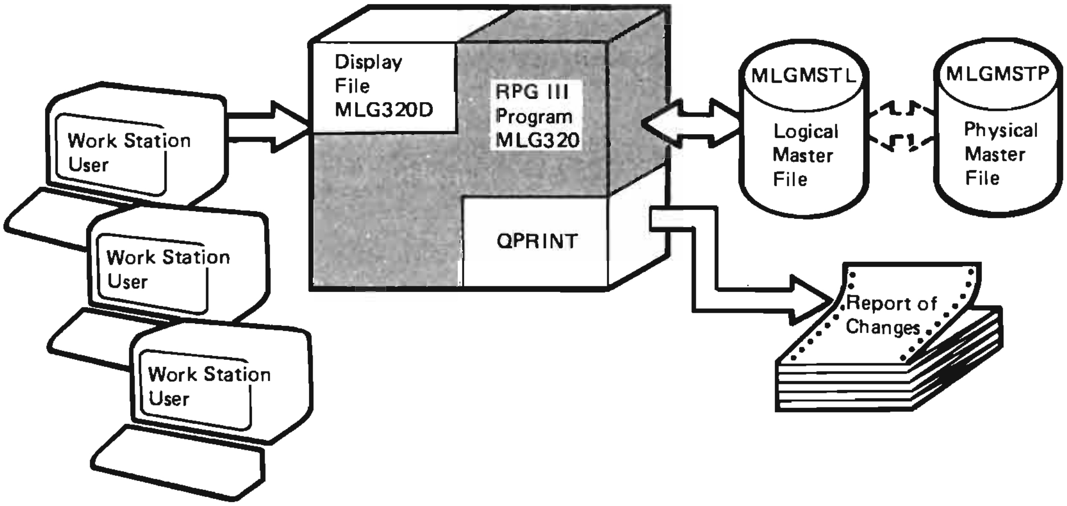
To allow for recovery from damaged objects (assuming you are keying in one or more batches of transactions per day), the transaction file could be saved every two to three days. If the damaged transaction file is not usable and the first recovery method described above is used, the backup can be restored. If the second recovery method described above is used, the transaction file can be rebuilt using the entries in the data base log.

Chapter 11. Interactive Maintenance from Multiple Work Stations

OVERVIEW

In Chapter 8, work station users interactively applied transactions to a master file through a DFU application. The more advanced approach in this chapter uses an RPG III maintenance program in combination with a display file to provide a series of displays that allow work station users to interactively maintain a master file. This approach has the following characteristics:

- A display file, MLG320D, formats four displays: PROMPT, CHANGE, ADDTN, DSPLY.
- Work station users enter data on these displays to:
  - Add, change, delete, or review a record in the master file
  - Specify a title for the printed output report
- An RPG III program, MLG320, applies work station input to a master file, MLGMSTP, through a logical file, MLGMSTL.
- Printed output occurs for any changes to the master file.



INTERACTIVE MAINTENANCE PROGRAM AND DISPLAY FILE

An RPG program named MLG320 and a display file named MLG320D are created to allow work station users to display and modify the records in a master file. The work station user begins by calling MLG320, such as by selecting an option on a menu that calls the program (see the discussion of the program call menu and specialized user menus in Chapter 8).

Program Flow

A flowchart of the RPG III program is shown in Figure 11-1. This flow of the program centers around the entries made by the work station user.

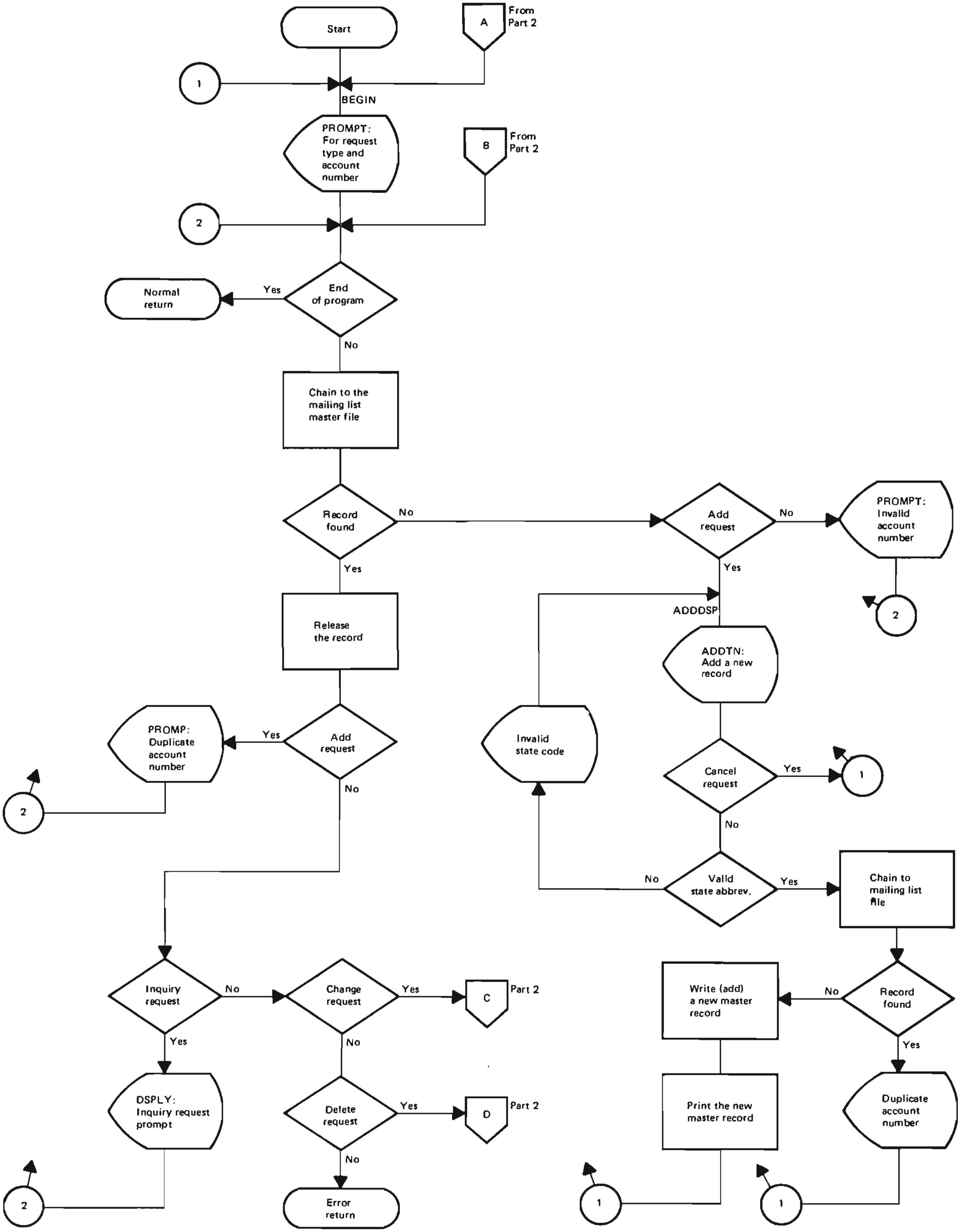


Figure 11-1 (Part 1 of 2). Flowchart of MLG320 Maintenance Program

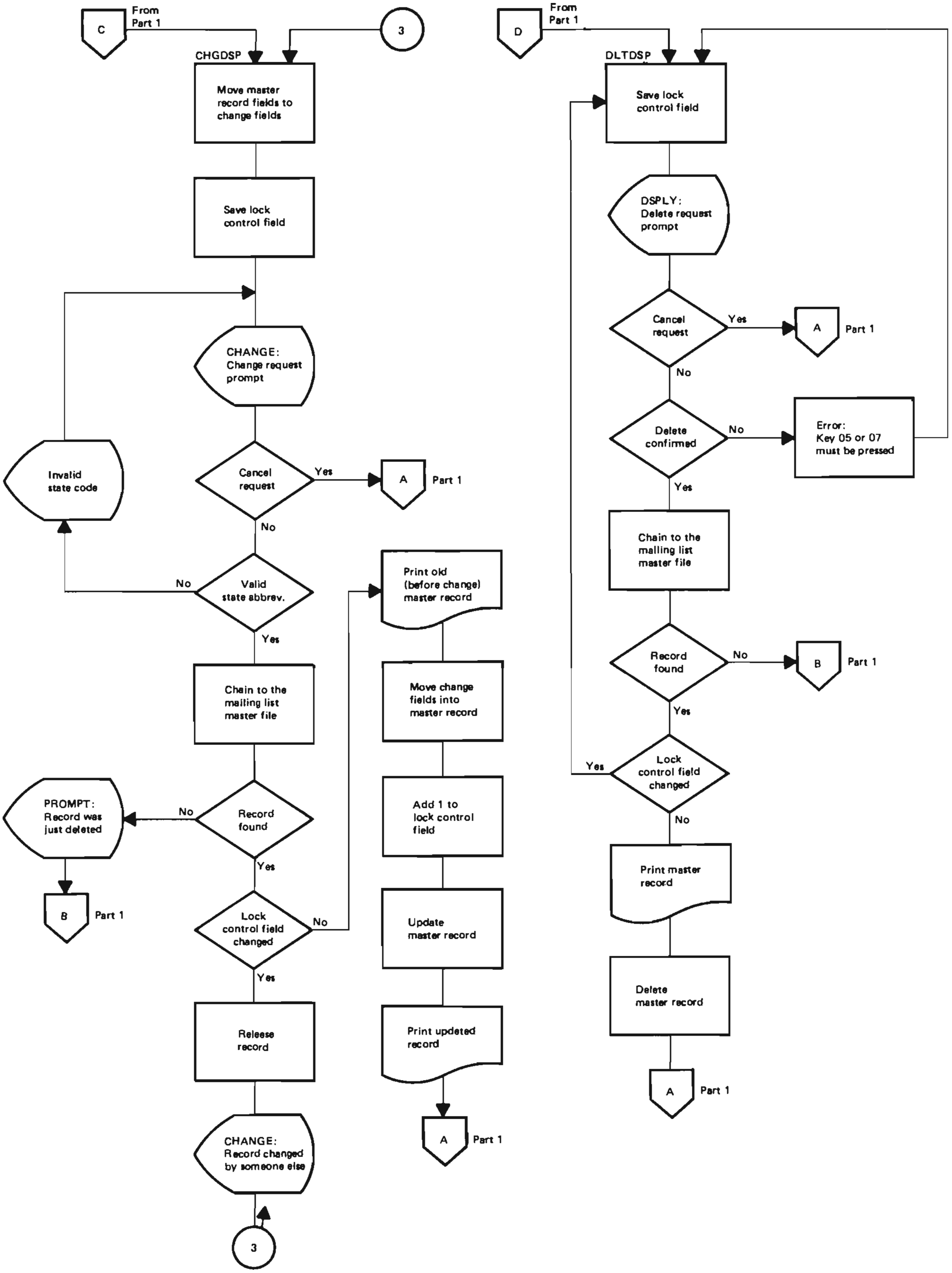
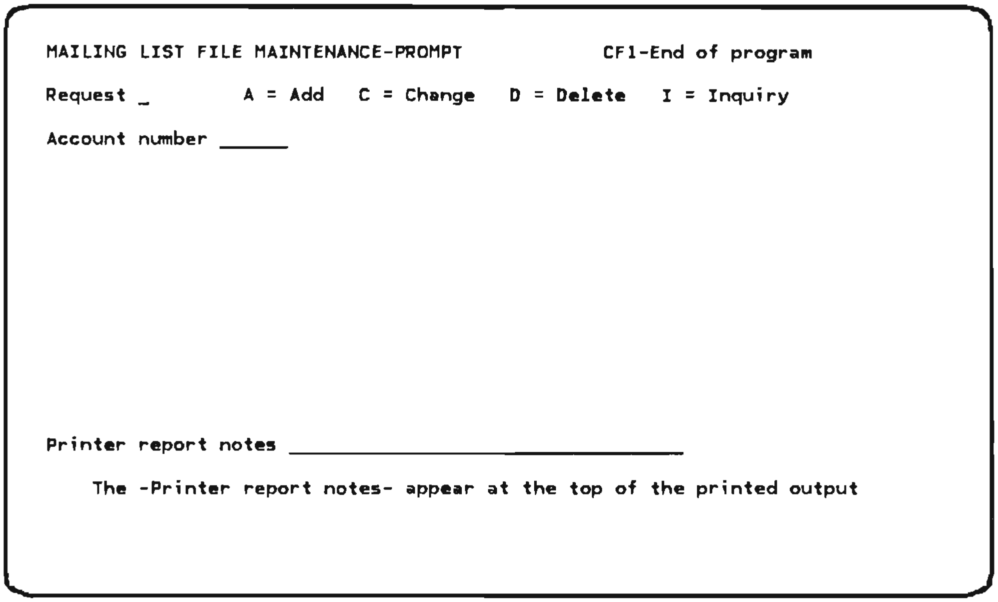

C
From Part 1
3
CHGDSP
Move master record fields to change fields
Save lock control field
CHANGE: Change request prompt
Invalid state code
Cancel request
Yes
A
Part 1
No
Valid state abbrev.
No
Yes
Chain to the mailing list master file
Record found
No
PROMPT: Record was just deleted
B
Part 1
Yes
Lock control field changed
No
Yes
Release record
CHANGE: Record changed by someone else
3
Print old (before change) master record
Move change fields into master record
Add 1 to lock control field
Update master record
Print updated record
A
Part 1
D
From Part 1
DLTDSP
Save lock control field
DSPLY: Delete request prompt
Cancel request
Yes
A
Part 1
No
Delete confirmed
No
Error: Key 05 or 07 must be pressed
Yes
Chain to the mailing list master file
Record found
No
B
Part 1
Yes
Lock control field changed
Yes
No
Print master record
Delete master record
A
Part 1

Figure 11-1 (Part 2 of 2). Flowchart of MLG320 Maintenance Program

The work station user begins by keying in a request type, account number, and report title on the following display, which is named PROMPT:



```
MAILING LIST FILE MAINTENANCE-PROMPT          CF1-End of program

Request _        A = Add    C = Change    D = Delete    I = Inquiry

Account number ______









Printer report notes ______________________________

    The -Printer report notes- appear at the top of the printed output
```

A record can be added to the master file, changed, reviewed (inquiry), or deleted from the master file. An existing or new account number can be entered. Text (such as date or batch number) can be entered for the *Printer report notes*. If CF1 (end of program) is pressed, the program returns.

The program tests for a valid combination of entries. If an addition is requested, the account number cannot exist on the file. If change, delete, or inquiry is requested, the account number must exist.

If the entries are not a valid combination, one of the following error messages appears on the PROMPT display:

Duplicate account number
Invalid account number

The program branches to unique routines depending on the type of request.

When a request is made to retrieve a record, that record becomes locked because the file is specified as *update*. Other work station users (or a batch program) who request that same record for a change or deletion cannot retrieve the record until the lock is released. (The programs can be coded with a time-out exception handling routine; however, this is not discussed in this publication.) To prevent a record from being locked to other users, the program releases the record.

The record is retrieved again if the work station user requests to change or delete it. When the record is retrieved, it may not be the same (another work station user may have changed it). If the record is not the same, one of the following error messages appears:

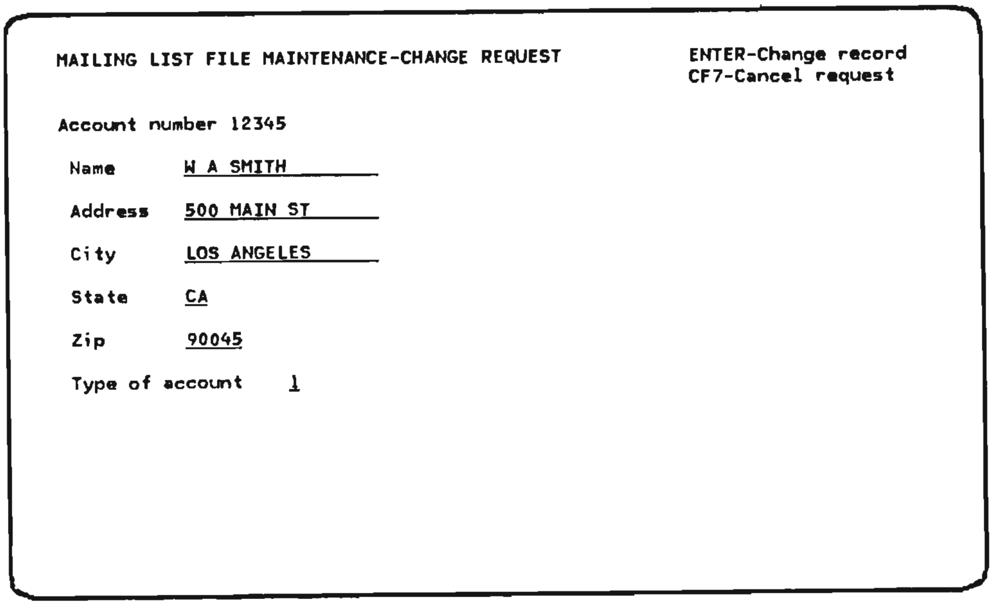
Record was just deleted (on PROMPT display)
Record has just been changed by someone else. Review request (on CHANGE display)

The program displays these messages based on the value of the lock control field (MLGLK1) in the retrieved record. (MLGLK1 is a three-digit field and is shown in the field reference file MLGREFP in Chapter 6.) When a request is made from the PROMPT display to retrieve a record, the value in MLGLK1 is saved. When the record is retrieved again for the actual update, the value in MLGLK1 is compared with the saved MLGLK1 value. If the values are the same, the record has not been changed; however, if the values are not the same, the record has been changed because the value in MLGLK1 is incremented by 1 each time an update is made. This ensures that multiple work station users are notified when they are trying to update the same record at the same time. Because MLGLK1 is defined as a numeric field, its value is initialized at 0. At the 1000th update, the field is automatically reset to 000.

If a request to add a record is made, the PROMPT format is displayed and the work station user can enter a record or cancel the request. If a record is entered, the program checks whether or not the account number is still unique (another work station user may have just entered the same account number). If the account number is unique, the program adds the new record to the file and prints the record on the printer output report, as shown below:

```
  9/21/81                     MAILING LIST MAINTENANCE                 PAGE     1


ACCT   TYP  NAME                 ADDRESS             CITY               STATE   ZIP     COMMENT

12300   4   P M DAVIS            1320 SOUTH RD       PITTSBURGH          PA     15238   ADDED
```

If a request to change a record is made, the data from the master record fields is moved to the change fields (XNAME, XADDR, XCITY, XSTATE, XZIP, XACTTYP) and the value of MLGLK1 is saved. The CHANGE format is displayed:



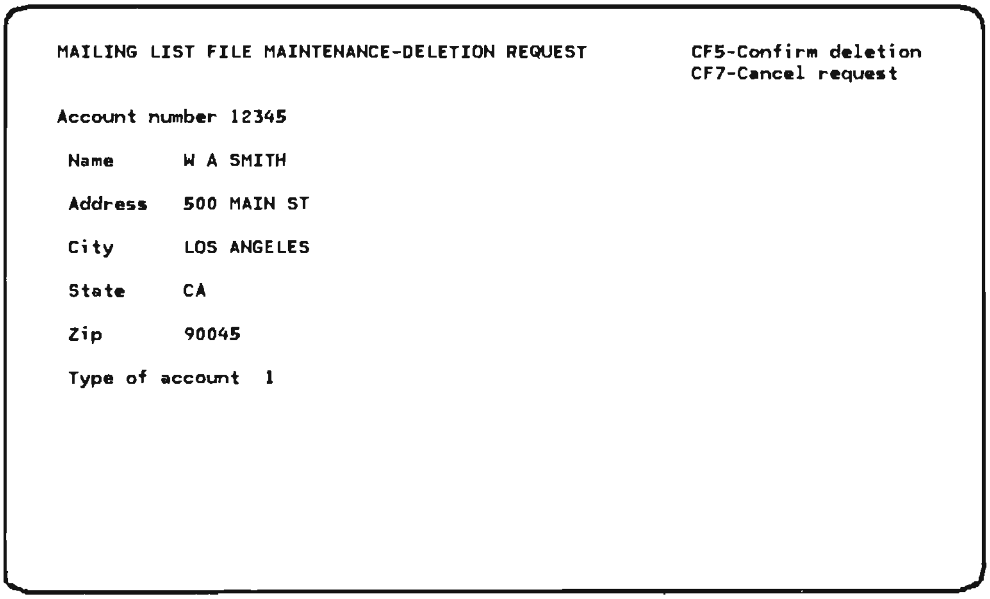
```
MAILING LIST FILE MAINTENANCE-CHANGE REQUEST          ENTER-Change record
                                                      CF7-Cancel request

Account number 12345

 Name      W A SMITH

 Address   500 MAIN ST

 City      LOS ANGELES

 State     CA

 Zip       90045

 Type of account   1
```

The request can be canceled (press CF7) or the record can be changed (press Enter). If Enter is pressed, the master record is retrieved again and the value of MLGLK1 is compared with the saved MLGLK1 value. If the values are the same:

- The existing master record is printed on the printer output report (as shown below).
- The change fields are moved to replace the existing master fields.
- MLGLK1 is incremented by 1.
- The master record is updated.
- The new master record is printed on the printer output report (as shown below).

```
  9/21/81                      MAILING LIST MAINTENANCE                   PAGE    1


ACCT   TYP   NAME                ADDRESS              CITY              STATE   ZIP     COMMENT


12345   1    W A SMITH           500 MAIN ST          LOS ANGELES        CA     90045   BEFORE
12345   1    W A SMITH           320 PACIFIC AVE      SAN JOSE           CA     95112   AFTER
```

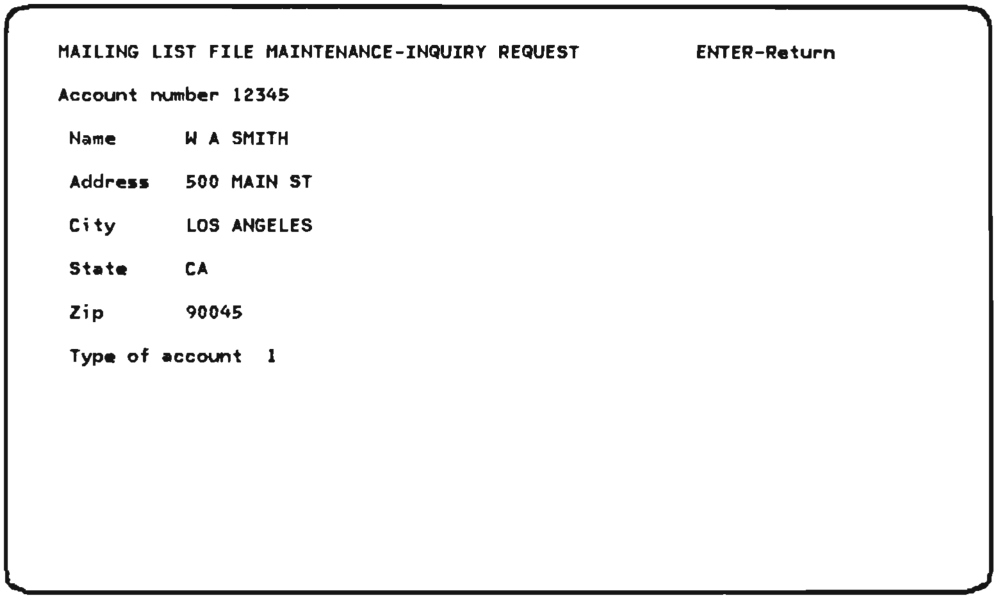
If a request to delete a record is made, the value in MLGLK1 is saved and a copy of the master record is displayed in the DSPLY format:



```
MAILING LIST FILE MAINTENANCE-DELETION REQUEST        CF5-Confirm deletion
                                                      CF7-Cancel request

Account number 12345

 Name       W A SMITH

 Address    500 MAIN ST

 City       LOS ANGELES

 State      CA

 Zip        90045

 Type of account  1
```

The request can be confirmed (press CF5) or canceled (press CF7). If it is confirmed, the master record is retrieved and the value of MLGLK1 is checked. If the value of MLGLK1 has not changed, the master record is printed and then deleted. If the work station user presses a command key other than CF5 or CF7, an error message is displayed.

If a request to inquire into a record is made, the DSPLY format is displayed:

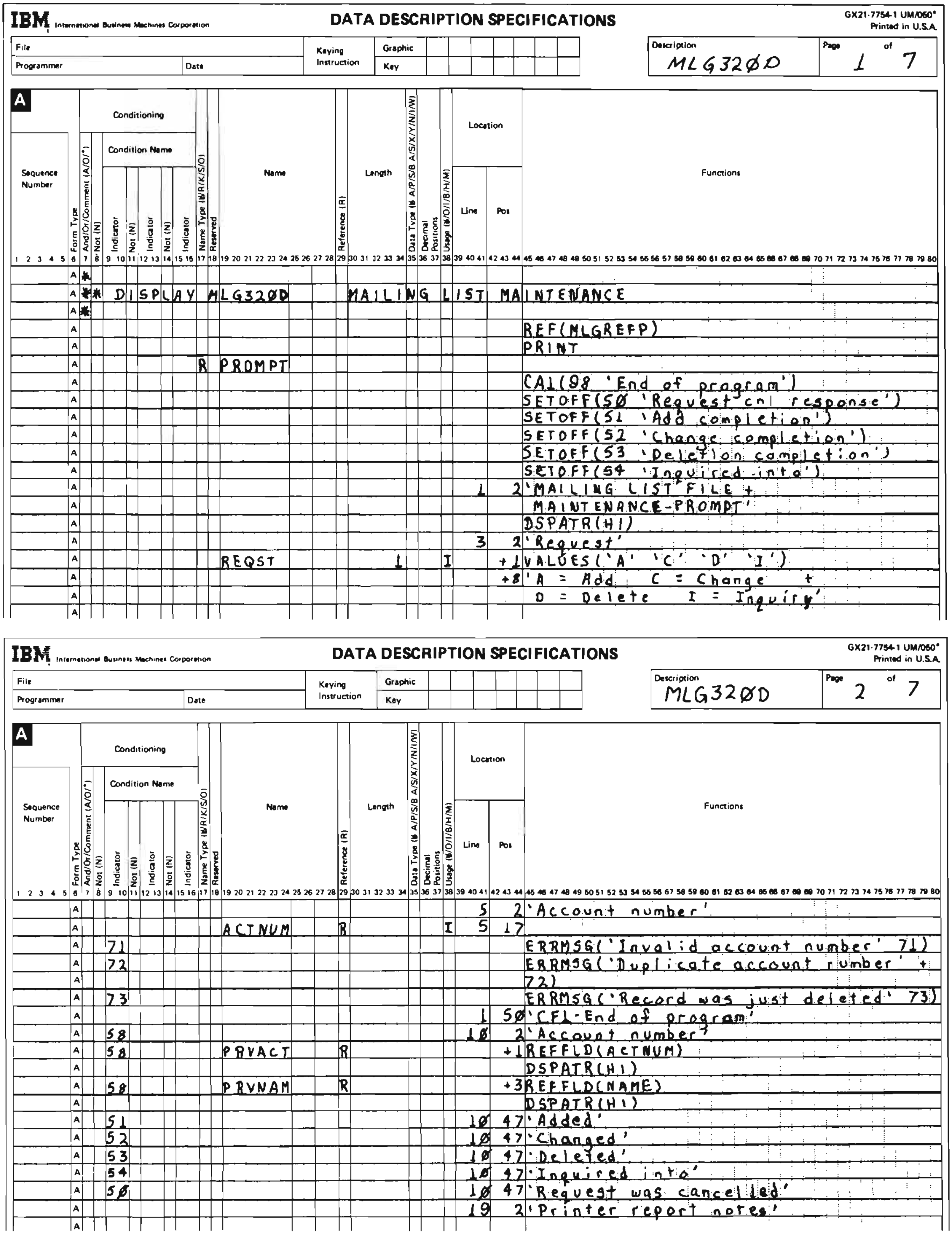


```
MAILING LIST FILE MAINTENANCE-INQUIRY REQUEST          ENTER-Return

Account number 12345

 Name      W A SMITH

 Address   500 MAIN ST

 City      LOS ANGELES

 State     CA

 Zip       90045

 Type of account  1
```

The master record is displayed. The work station user presses Enter (which is the only valid key) to return to the PROMPT display.

Data Description Specifications for MLG320D

The DDS for MLG320D are shown in Figure 11-2. These DDS could be generated using the screen design aid, which was introduced in Chapter 8 and is described in detail in the *SDA Reference Manual and User's Guide.*



IBM International Business Machines Corporation — DATA DESCRIPTION SPECIFICATIONS — GX21-7754-1 UM/050* Printed in U.S.A.

Description: MLG32ØD Page 1 of 7

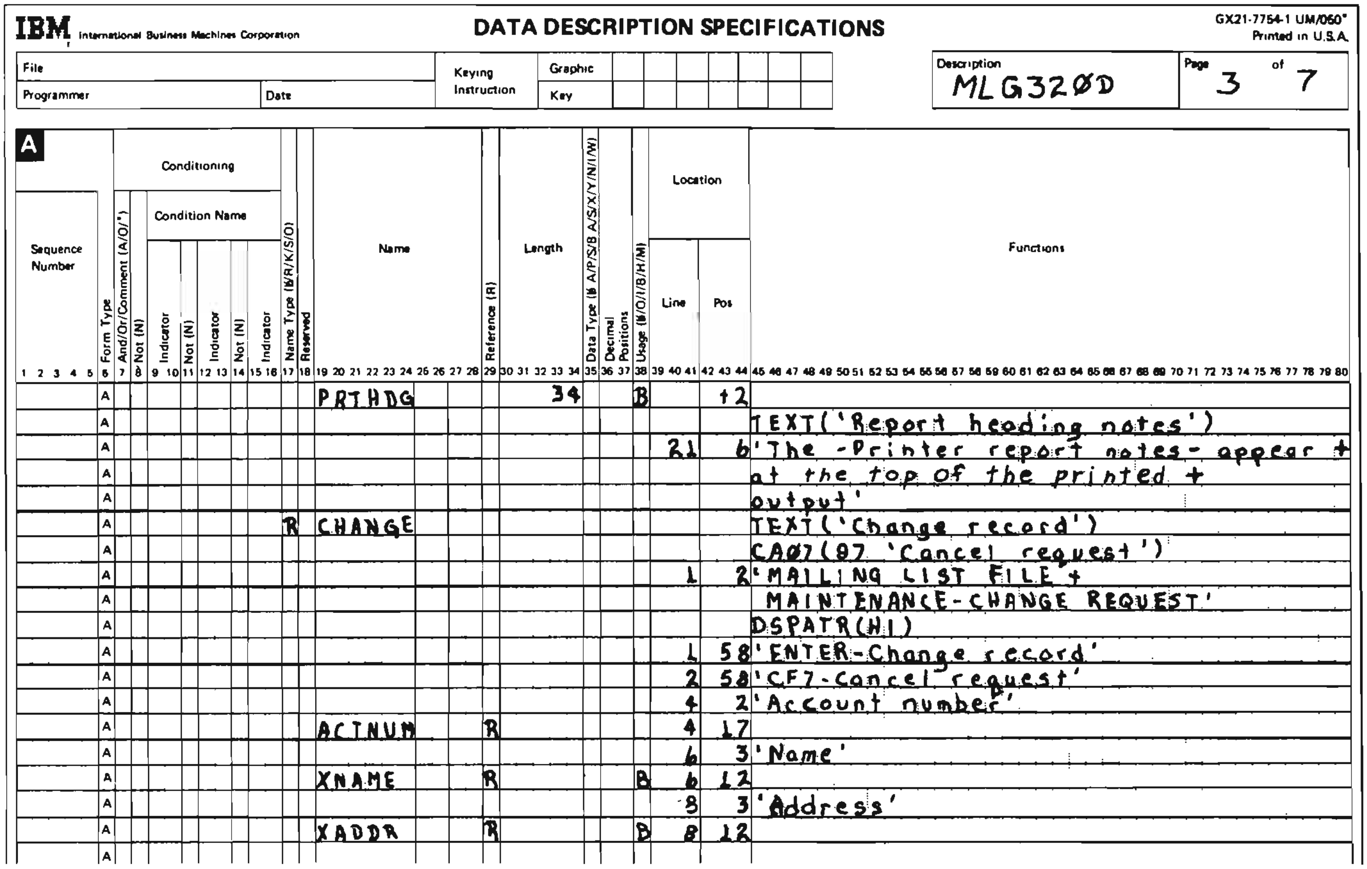
```
     A*
     A** DISPLAY MLG32ØD     MAILING LIST MAINTENANCE
     A*
     A                                      REF(MLGREFP)
     A                                      PRINT
     A          R PROMPT
     A                                      CA1(98 'End of program')
     A                                      SETOFF(5Ø 'Request cnl response')
     A                                      SETOFF(51 'Add completion')
     A                                      SETOFF(52 'Change completion')
     A                                      SETOFF(53 'Deletion completion')
     A                                      SETOFF(54 'Inquired into')
     A                                  1  2'MAILING LIST FILE +
     A                                       MAINTENANCE-PROMPT'
     A                                      DSPATR(HI)
     A                                  3  2'Request'
     A            REQST         1   I     +1VALUES('A' 'C' 'D' 'I')
     A                                    +8'A = Add   C = Change    +
     A                                       D = Delete   I = Inquiry'
     A
```

IBM International Business Machines Corporation — DATA DESCRIPTION SPECIFICATIONS — GX21-7754-1 UM/050* Printed in U.S.A.

Description: MLG32ØD Page 2 of 7

```
     A                                  5  2'Account number'
     A            ACTNUM    R           I  5 17
     A  71                                  ERRMSG('Invalid account number' 71)
     A  72                                  ERRMSG('Duplicate account number' +
     A                                      72)
     A  73                                  ERRMSG('Record was just deleted' 73)
     A                                  1 5Ø'CF1-End of program'
     A  58                             1Ø  2'Account number'
     A  58        PRVACT    R             +1REFFLD(ACTNUM)
     A                                      DSPATR(HI)
     A  58        PRVNAM    R             +3REFFLD(NAME)
     A                                      DSPATR(HI)
     A  51                             1Ø 47'Added'
     A  52                             1Ø 47'Changed'
     A  53                             1Ø 47'Deleted'
     A  54                             1Ø 47'Inquired into'
     A  5Ø                             1Ø 47'Request was cancelled'
     A                                 19  2'Printer report notes'
     A
```

Figure 11-2 (Part 1 of 4). DDS for MLG320D Display File



IBM International Business Machines Corporation

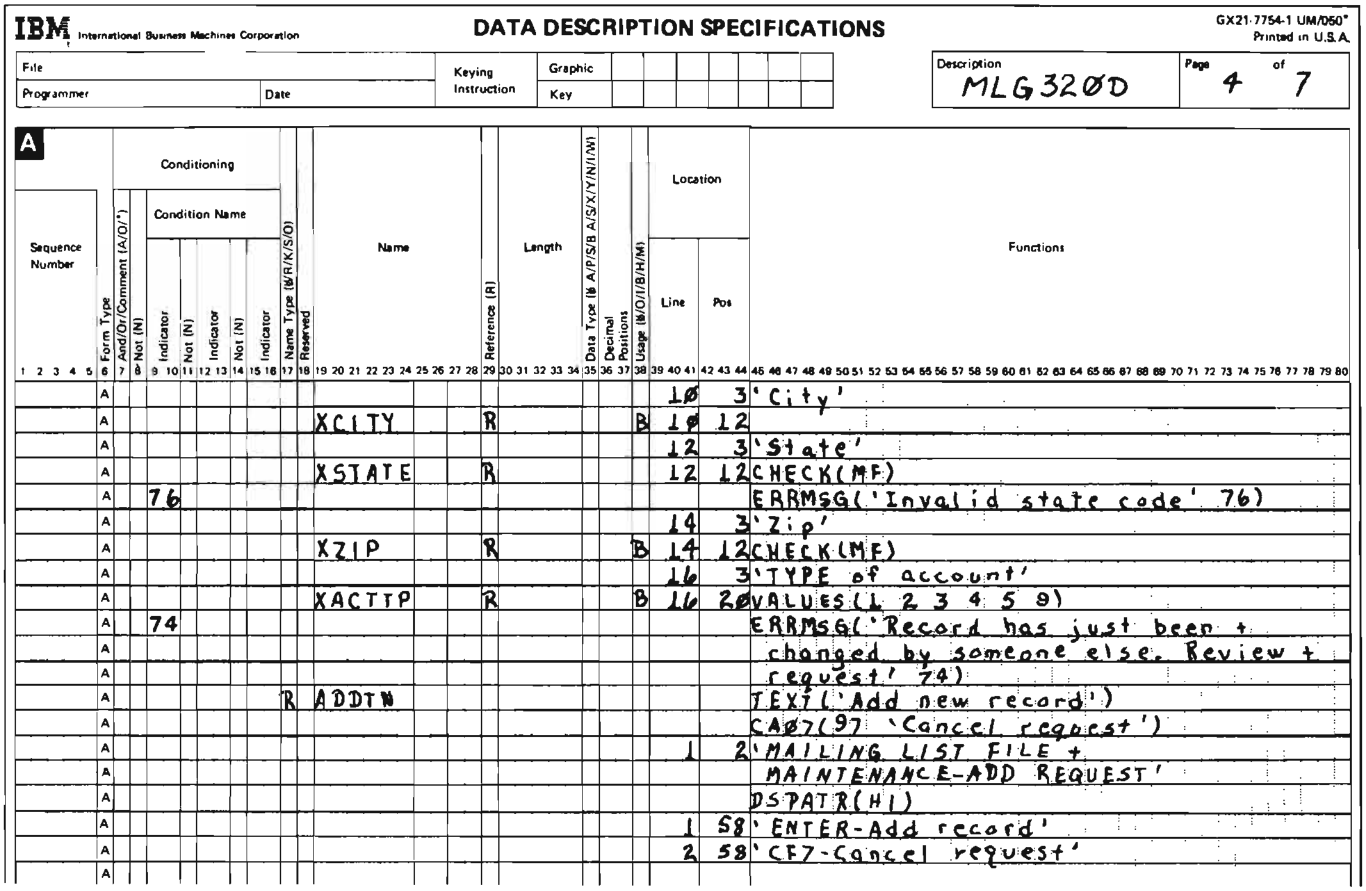
DATA DESCRIPTION SPECIFICATIONS

GX21-7754-1 UM/050*
Printed in U.S.A.

File | Programmer | Date | Keying Instruction | Graphic | Key

Description MLG32ØD Page 3 of 7

```
     A            PRTHDG        34   B    +2
     A                                      TEXT('Report heading notes')
     A                                 21  6'The -Printer report notes- appear +
     A                                      at the top of the printed +
     A                                      output'
     A          R CHANGE                    TEXT('Change record')
     A                                      CAØ7(97 'Cancel request')
     A                                  1  2'MAILING LIST FILE +
     A                                       MAINTENANCE-CHANGE REQUEST'
     A                                      DSPATR(HI)
     A                                  1 58'ENTER-Change record'
     A                                  2 58'CF7-Cancel request'
     A                                  4  2'Account number'
     A            ACTNUM    R           4 17
     A                                  6  3'Name'
     A            XNAME     R        B  6 12
     A                                  8  3'Address'
     A            XADDR     R        B  8 12
     A
```



IBM International Business Machines Corporation

DATA DESCRIPTION SPECIFICATIONS

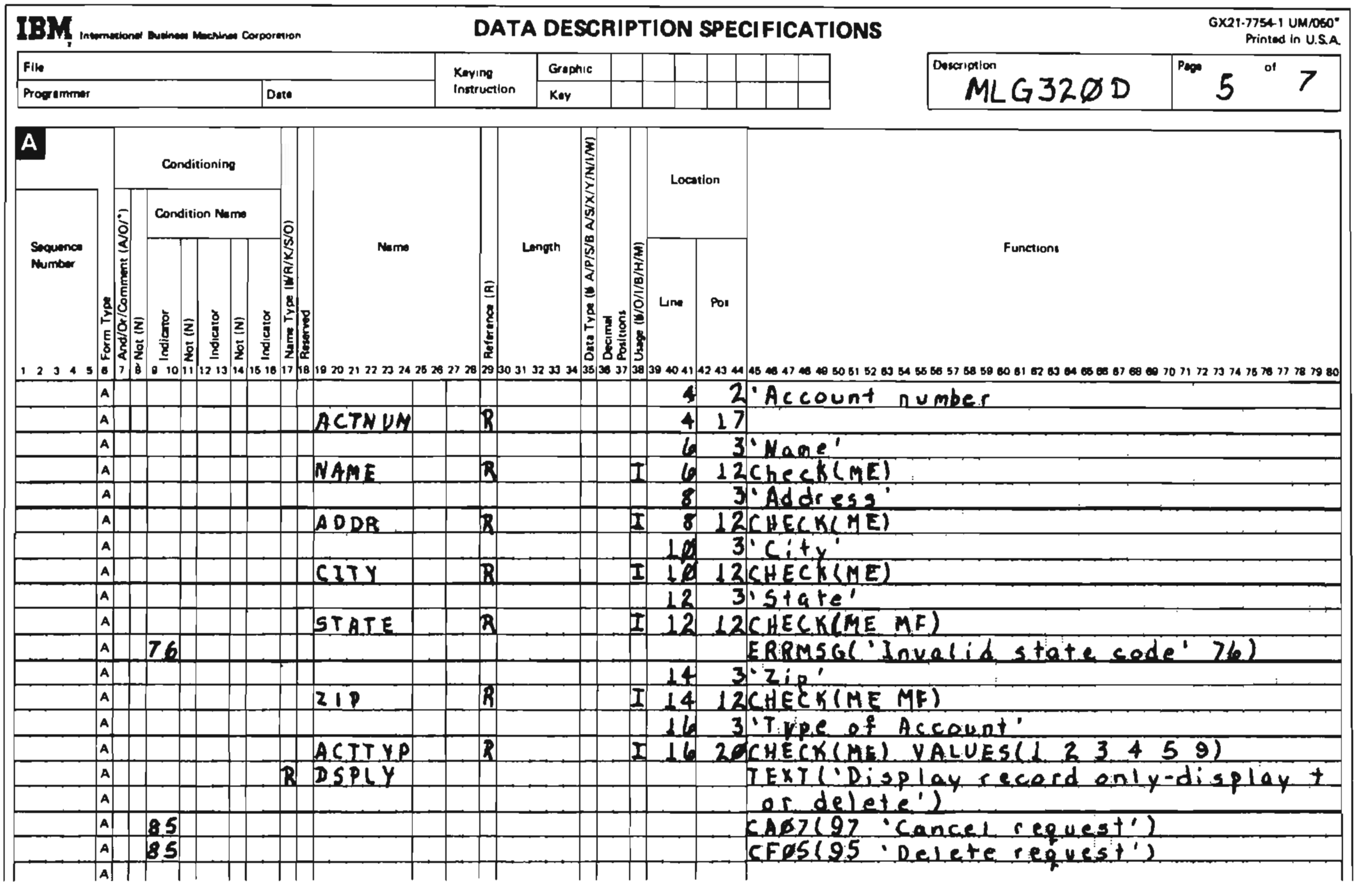
GX21-7754-1 UM/050*
Printed in U.S.A.

File | Programmer | Date | Keying Instruction | Graphic | Key

Description MLG32ØD Page 4 of 7

```
     A                                 1Ø  3'City'
     A            XCITY     R        B 1Ø 12
     A                                 12  3'State'
     A            XSTATE    R          12 12CHECK(MF)
     A  76                                  ERRMSG('Invalid state code' 76)
     A                                 14  3'Zip'
     A            XZIP      R        B 14 12CHECK(MF)
     A                                 16  3'TYPE of account'
     A            XACTTP    R        B 16 2ØVALUES(1 2 3 4 5 9)
     A  74                                  ERRMSG('Record has just been +
     A                                       changed by someone else. Review +
     A                                       request' 74)
     A          R ADDTN                     TEXT('Add new record')
     A                                      CAØ7(97 'Cancel request')
     A                                  1  2'MAILING LIST FILE +
     A                                       MAINTENANCE-ADD REQUEST'
     A                                      DSPATR(HI)
     A                                  1 58'ENTER-Add record'
     A                                  2 58'CF7-Cancel request'
     A
```

Figure 11-2 (Part 2 of 4). DDS for MLG320D Display File



IBM International Business Machines Corporation

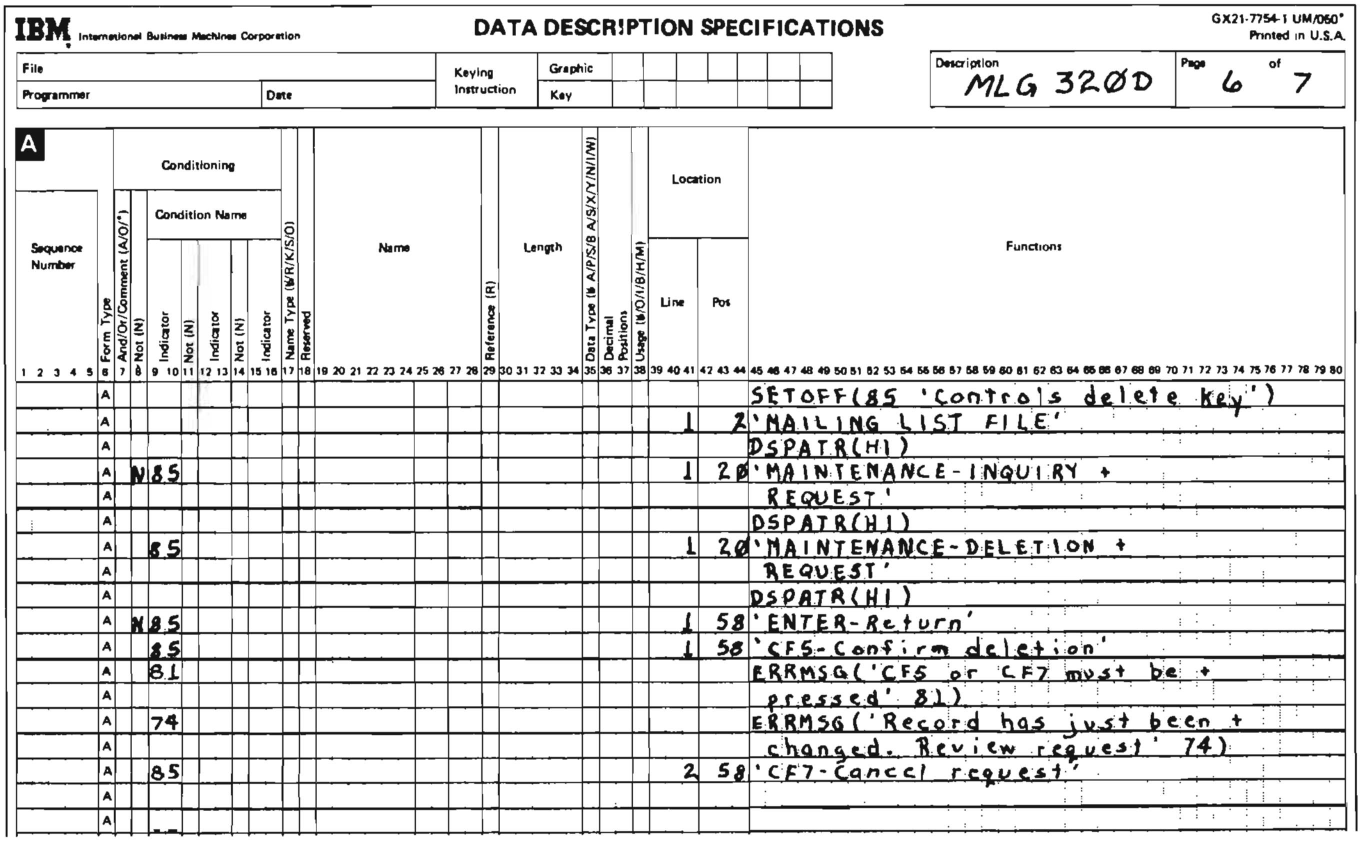
DATA DESCRIPTION SPECIFICATIONS

GX21-7754-1 UM/050*
Printed in U.S.A.

File | Keying Instruction | Graphic | Key | Programmer | Date

Description: MLG32ØD Page 5 of 7

```
     A                                  4  2'Account number
     A            ACTNUM    R           4 17
     A                                  6  3'Name'
     A            NAME      R        I  6 12Check(ME)
     A                                  8  3'Address'
     A            ADDR      R        I  8 12CHECK(ME)
     A                                 1Ø  3'City'
     A            CITY      R        I 1Ø 12CHECK(ME)
     A                                 12  3'State'
     A            STATE     R        I 12 12CHECK(ME MF)
     A  76                                  ERRMSG('Invalid state code' 76)
     A                                 14  3'Zip'
     A            ZIP       R        I 14 12CHECK(ME MF)
     A                                 16  3'Type of Account'
     A                                 
     A            ACTTYP    R        I 16 2ØCHECK(ME) VALUES(1 2 3 4 5 9)
     A          R DSPLY                     TEXT('Display record only-display +
     A                                       or delete')
     A  85                                  CAØ7(97 'Cancel request')
     A  85                                  CFØ5(95 'Delete request')
     A
```



IBM International Business Machines Corporation

DATA DESCRIPTION SPECIFICATIONS

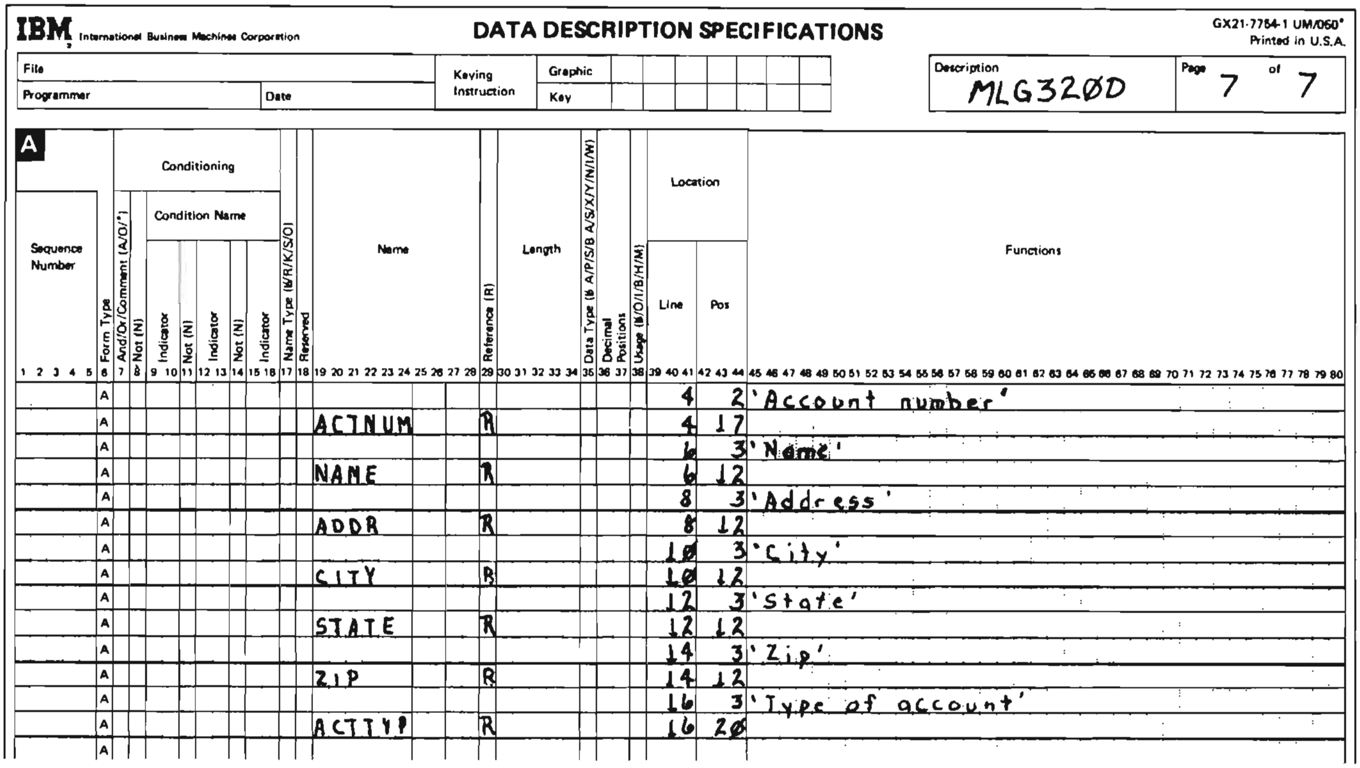
GX21-7754-1 UM/050*
Printed in U.S.A.

File | Keying Instruction | Graphic | Key | Programmer | Date

Description: MLG 32ØD Page 6 of 7

```
     A                                      SETOFF(85 'Controls delete key')
     A                                  1  2'MAILING LIST FILE'
     A                                      DSPATR(HI)
     A N85                              1 2Ø'MAINTENANCE-INQUIRY +
     A                                       REQUEST'
     A                                      DSPATR(HI)
     A  85                              1 2Ø'MAINTENANCE-DELETION +
     A                                       REQUEST'
     A                                      DSPATR(HI)
     A N85                              1 58'ENTER-Return'
     A  85                              1 58'CF5-Confirm deletion'
     A  81                                  ERRMSG('CF5 or CF7 must be +
     A                                       pressed' 81)
     A  74                                  ERRMSG('Record has just been +
     A                                       changed. Review request' 74)
     A  85                              2 58'CF7-Cancel request'
     A
     A
```

Figure 11-2 (Part 3 of 4). DDS for MLG320D Display File



IBM International Business Machines Corporation

DATA DESCRIPTION SPECIFICATIONS

GX21-7754-1 UM/050*
Printed in U.S.A.

File
Programmer
Date
Keying Instruction
Graphic
Key

Description MLG32ØD
Page 7 of 7

A

Sequence Number	Form Type	And/Or/Comment (A/O/*)	Not (N)	Indicator	Not (N)	Indicator	Not (N)	Indicator	Name Type (b/R/K/S/O)	Reserved	Name	Reference (R)	Length	Data Type (b A/P/S/B A/S/X/Y/N/I/W)	Decimal Positions	Usage (b/O/I/B/H/M)	Line	Pos	Functions
1 2 3 4 5	6	7	8	9 10	11	12 13	14	15 16	17	18	19 20 21 22 23 24 25 26 27 28	29	30 31 32 33 34	35	36 37	38	39 40 41	42 43 44	45 … 80
	A																4	2	'Account number'
	A										ACTNUM	R					4	17	
	A																6	3	'Name'
	A										NAME	R					6	12	
	A																8	3	'Address'
	A										ADDR	R					8	12	
	A																1Ø	3	'City'
	A										CITY	R					1Ø	12	
	A																12	3	'State'
	A										STATE	R					12	12	
	A																14	3	'Zip'
	A										ZIP	R					14	12	
	A																16	3	'Type of account'
	A										ACTTYP	R					16	2Ø	
	A																		

Figure 11-2 (Part 4 of 4). DDS for MLG320D Display File

The *REF keyword* specifies that the field reference file MLGREFP is to be used.

The *name PROMPT* specifies the format for the first prompt, which requests the type of work to be done. The fields REQST and ACTNUM are input fields. The keyword ERRMSG specifies the error conditions that are determined by the program. The fields PRVACT and PRVNAM are output fields. These two fields are conditioned to properly describe the last function specified. This feedback is to assist the work station user. The SETOFF keywords are used to ensure that the values are not displayed again. The field PRTHDG is an I/O field.

The *name CHANGE* specifies the format for the prompt which allows a record to be changed. CF7 is the cancel command key, which cancels the request. The fields into which changes are entered have names different from those in the master record. This is because the master record is released and retrieved again after changes have been entered. The B in position 38 specifies that those fields contain data that can be both displayed and changed.

The *name ADDTN* specifies the format of the prompt that allows you to add a record. The request can be canceled by pressing CF7. The I in position 38 specifies all fields as input fields.

The *name DSPLY* specifies the format for the prompts that allow delete and inquiry. CF7 (cancel request) and CF5 (delete request) are valid only if a delete request is made. Because position 38 is blank for each field, these fields are output fields and the data in them cannot be changed by the work station user. The constant *CF5-Confirm Deletion* is always on the delete display, but it will appear in high intensity when an error occurs. The following are possible error conditions:

- A command key, which is required on a delete request, is not pressed. The work station user must press a command key and not the Enter key.

- A delete occurs and the value in MLGLK1 is changed. It is probably an operational error to delete that record.

RPG III Specifications for MLG320

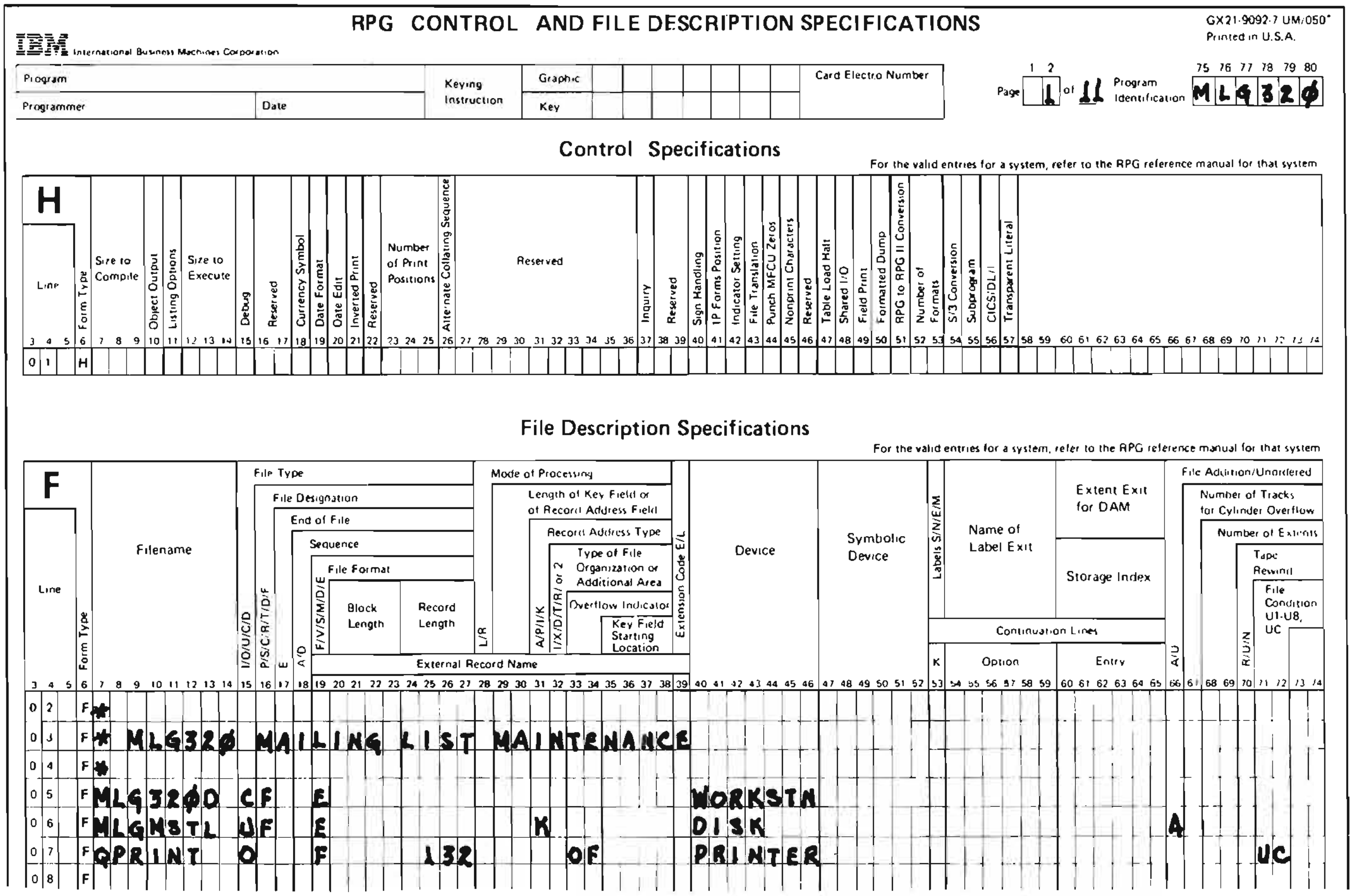
The RPG III specifications for the mailing list maintenance program MLG320 are shown in Figures 11-3 through 11-5. A description of the new RPG III operation codes–DO, IF, ELSE, END–follows.

The DO statement is a convenient form to group statements that must be performed under the same condition. The DO statement may be conditioned as it is in statement 🅐 (Figure 11-4, part 1). The END statement ends the DO group.

The IF statement may be used to test a condition and perform a series of statements. In statement 🅑 (Figure 11-4, part 3), the IFEQ operation is used to test indicator 97 for an *on* condition (the value 1 is *on* and 0 is *off*). *IN97 is the field name for indicator 97.

In statement 🅒 (Figure 11-4, part 3), the IFEQ operation is used to test indicator 95 for an *off* condition. The ELSE operation in statement 🅓 (Figure 11-4, part 3), is used to begin a series of statements that are executed if indicator 95 is *on*.

In statement 🅔 (Figure 11-4, part 3), another DO operation is nested within the previous IF statement. The term nested means that the statements are a subgroup of the previous group. The nested DO group ends on statement 🅕 (Figure 11-4, part 3). The ELSE statement ends on statement 🅖 (Figure 11-4, part 3). The RPG III compiler will print a nest level number on each statement to assist you in determining where a level begins and ends.



IBM International Business Machines Corporation

RPG CONTROL AND FILE DESCRIPTION SPECIFICATIONS

GX21-9092-7 UM/050*
Printed in U.S.A.

Program | Keying Instruction | Graphic | Key | Card Electro Number
Programmer | Date

Page 1 of 11 Program Identification MLG320

Control Specifications

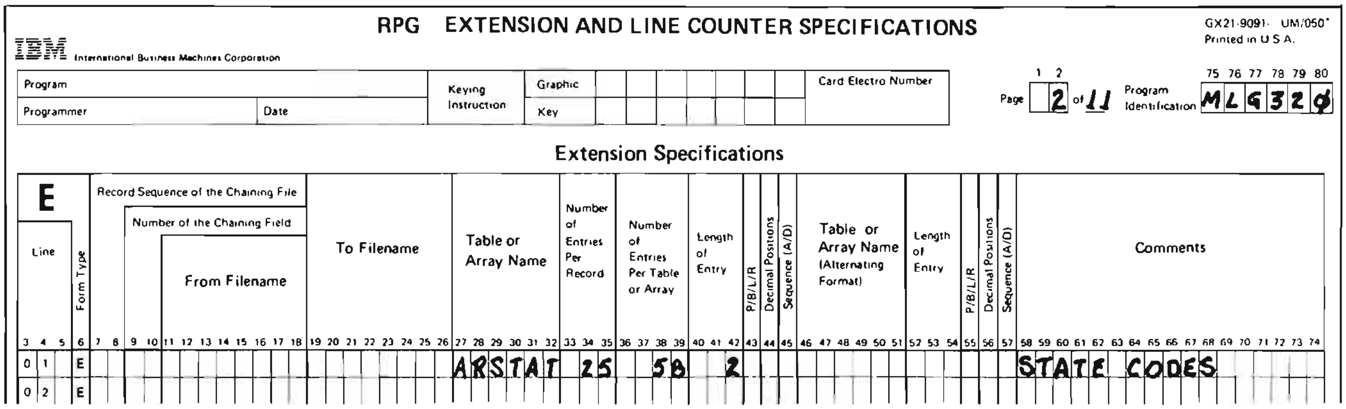
For the valid entries for a system, refer to the RPG reference manual for that system

```
01 H
```

File Description Specifications

For the valid entries for a system, refer to the RPG reference manual for that system

```
02 F*
03 F* MLG320 MAILING LIST MAINTENANCE
04 F*
05 FMLG320D CF  E                    WORKSTN
06 FMLGMSTL UF  E           K        DISK                   A
07 FQPRINT  O   F     132     OF     PRINTER                     UC
08 F
```



IBM International Business Machines Corporation

RPG EXTENSION AND LINE COUNTER SPECIFICATIONS

GX21-9091- UM/050*
Printed in U S A.

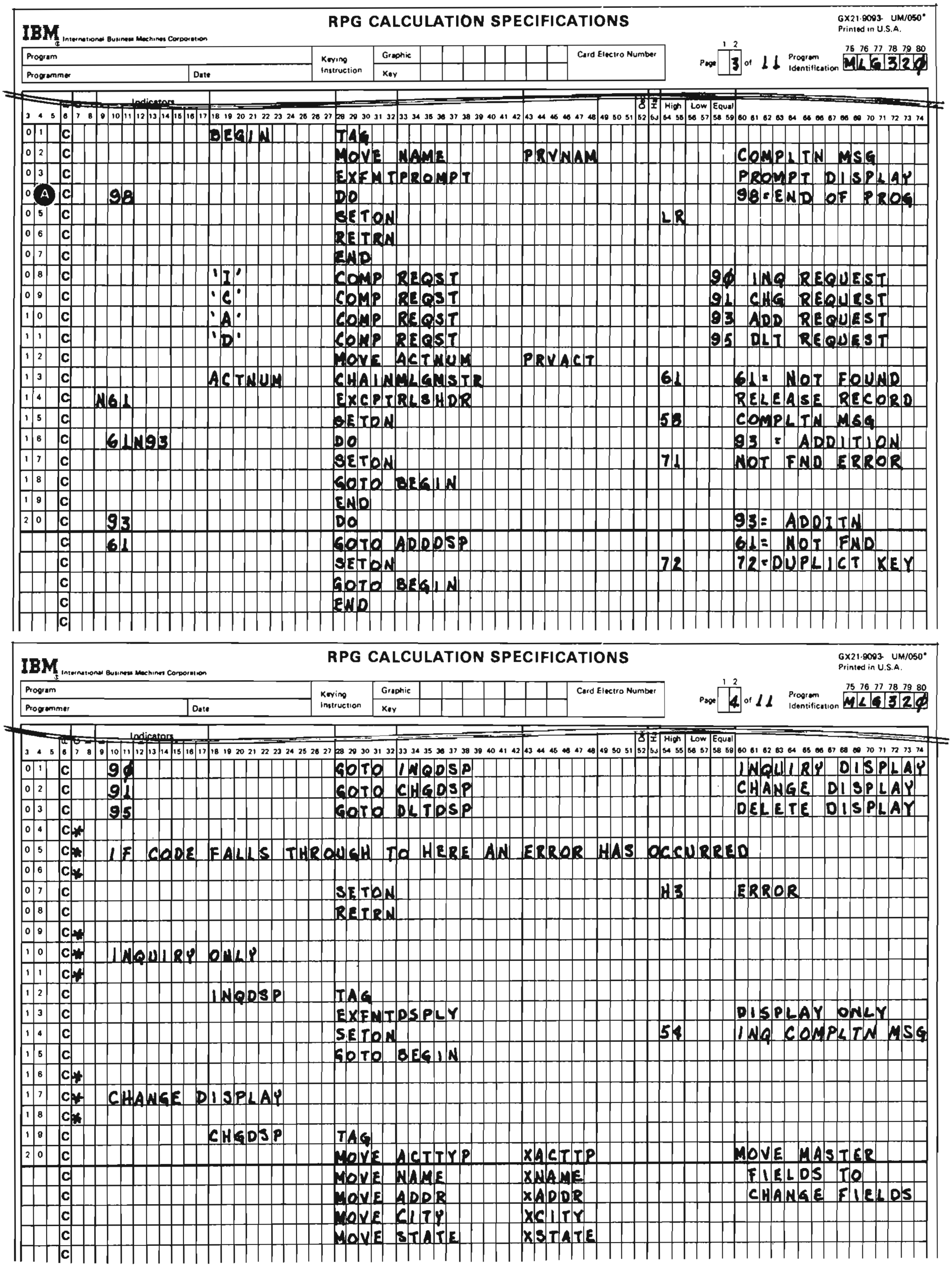
Program | Keying Instruction | Graphic | Key | Card Electro Number
Programmer | Date

Page 2 of 11 Program Identification MLG320

Extension Specifications

```
01 E                    ARSTAT 25  58  2               STATE CODES
02 E
```

Figure 11-3. Control and File Description Specifications for MLG320 Maintenance Program



IBM International Business Machines Corporation

RPG CALCULATION SPECIFICATIONS

GX21-9093- UM/050* Printed in U.S.A.

Program | Programmer | Date | Keying Instruction: Graphic / Key | Card Electro Number | Page 3 of 11 | Program Identification MLG32Ø

```
01 C           BEGIN     TAG
02 C                     MOVE NAME      PRVNAM           COMPLTN MSG
03 C                     EXFMTPROMPT                     PROMPT DISPLAY
0Ⓐ C   98                DO                              98=END OF PROG
05 C                     SETON                     LR
06 C                     RETRN
07 C                     END
08 C           'I'       COMP REQST                    9Ø INQ REQUEST
09 C           'C'       COMP REQST                    91 CHG REQUEST
10 C           'A'       COMP REQST                    93 ADD REQUEST
11 C           'D'       COMP REQST                    95 DLT REQUEST
12 C                     MOVE ACTNUM    PRVACT
13 C           ACTNUM    CHAINMLGMSTR              61    61= NOT FOUND
14 C  N61                EXCPTRLSHDR                     RELEASE RECORD
15 C                     SETON                     58    COMPLTN MSG
16 C   61N93             DO                              93 = ADDITION
17 C                     SETON                     71    NOT FND ERROR
18 C                     GOTO BEGIN
19 C                     END
20 C   93                DO                              93= ADDITN
   C   61                GOTO ADDDSP                     61= NOT FND
   C                     SETON                     72    72=DUPLICT KEY
   C                     GOTO BEGIN
   C                     END
   C
```

IBM International Business Machines Corporation

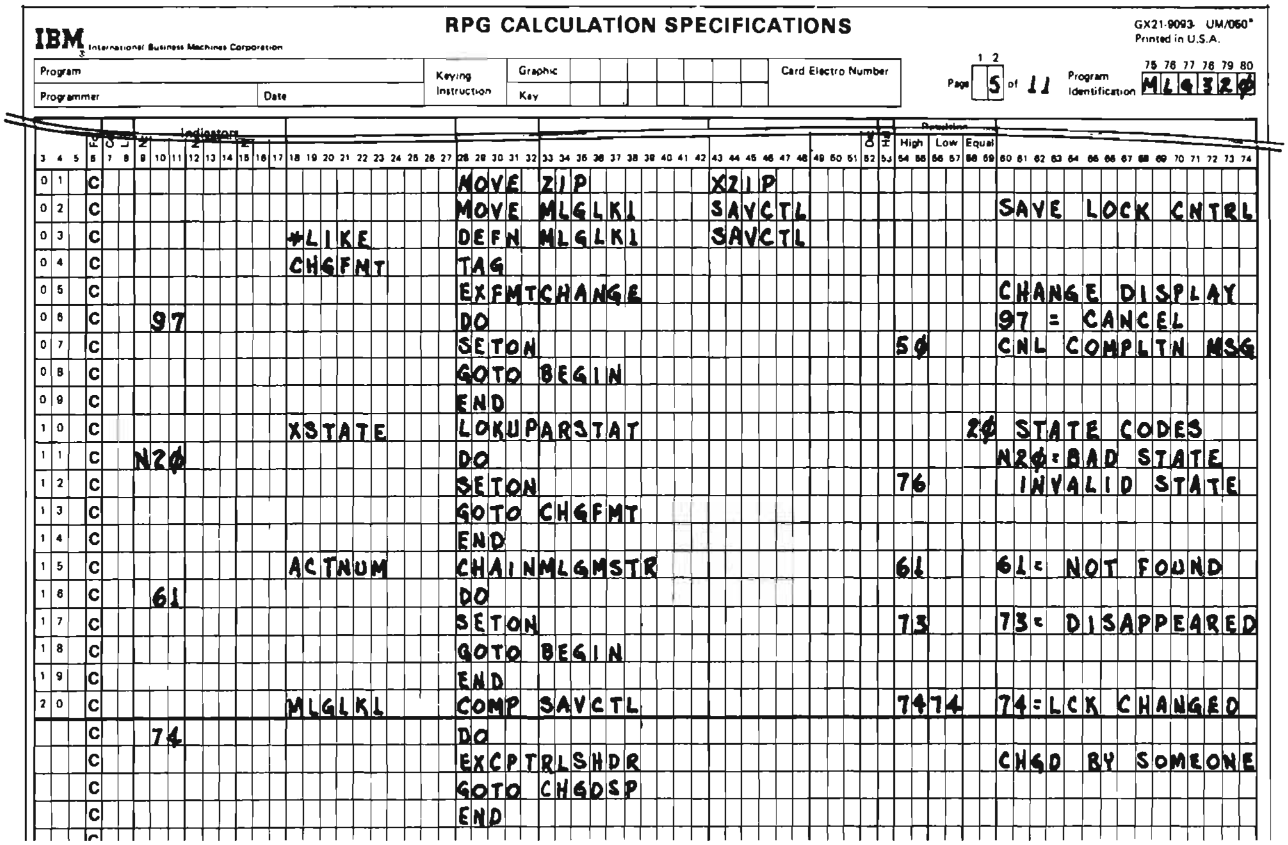
RPG CALCULATION SPECIFICATIONS

GX21-9093- UM/050* Printed in U.S.A.

Program | Programmer | Date | Keying Instruction: Graphic / Key | Card Electro Number | Page 4 of 11 | Program Identification MLG32Ø

```
01 C   9Ø                GOTO INQDSP                     INQUIRY DISPLAY
02 C   91                GOTO CHGDSP                     CHANGE DISPLAY
03 C   95                GOTO DLTDSP                     DELETE DISPLAY
04 C*
05 C*  IF CODE FALLS THROUGH TO HERE AN ERROR HAS OCCURRED
06 C*
07 C                     SETON                     H3    ERROR
08 C                     RETRN
09 C*
10 C*  INQUIRY ONLY
11 C*
12 C           INQDSP    TAG
13 C                     EXFMTDSPLY                      DISPLAY ONLY
14 C                     SETON                     54    INQ COMPLTN MSG
15 C                     GOTO BEGIN
16 C*
17 C*  CHANGE DISPLAY
18 C*
19 C           CHGDSP    TAG
20 C                     MOVE ACTTYP    XACTTP           MOVE MASTER
   C                     MOVE NAME      XNAME             FIELDS TO
   C                     MOVE ADDR      XADDR             CHANGE FIELDS
   C                     MOVE CITY      XCITY
   C                     MOVE STATE     XSTATE
   C
```

Figure 11-4 (Part 1 of 4). Calculation Specifications for MLG320 Maintenance Program



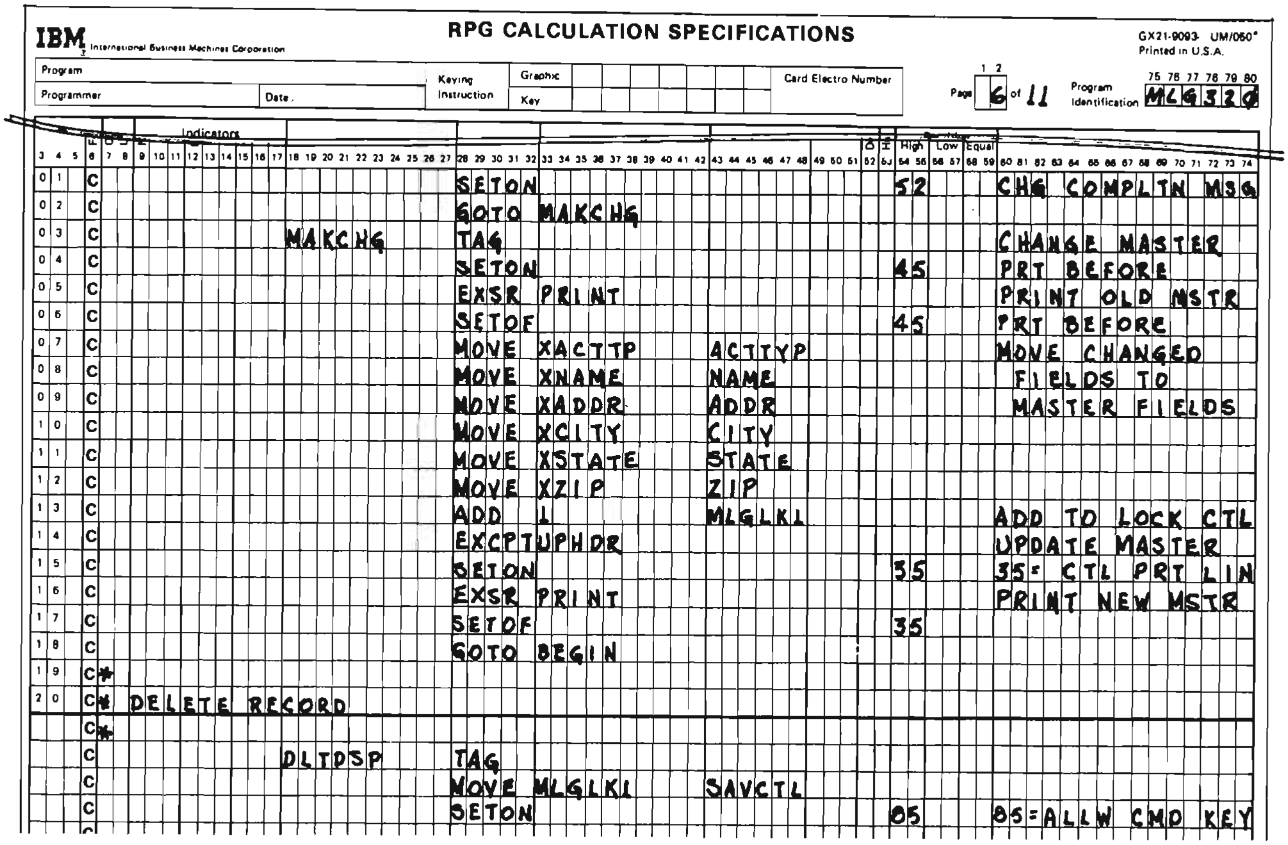
IBM International Business Machines Corporation

RPG CALCULATION SPECIFICATIONS

GX21-9093 UM/050* Printed in U.S.A.

Page 5 of 11 — Program Identification MLG32Ø

```
01 C                MOVE  ZIP        XZIP
02 C                MOVE  MLGLK1     SAVCTL        SAVE LOCK CNTRL
03 C     *LIKE      DEFN  MLGLK1     SAVCTL
04 C     CHGFMT     TAG
05 C                EXFMT CHANGE                   CHANGE DISPLAY
06 C  97            DO                             97 = CANCEL
07 C                SETON                   5Ø     CNL COMPLTN MSG
08 C                GOTO  BEGIN
09 C                END
10 C     XSTATE     LOKUP ARSTAT                2Ø STATE CODES
11 C N2Ø            DO                             N2Ø=BAD STATE
12 C                SETON                   76       INVALID STATE
13 C                GOTO  CHGFMT
14 C                END
15 C     ACTNUM     CHAIN MLGMSTR           61     61= NOT FOUND
16 C  61            DO
17 C                SETON                   73     73= DISAPPEARED
18 C                GOTO  BEGIN
19 C                END
20 C     MLGLKL     COMP  SAVCTL            7474   74=LCK CHANGED
   C  74            DO
   C                EXCPT RLSHDR                   CHGD BY SOMEONE
   C                GOTO  CHGDSP
   C                END
```



IBM International Business Machines Corporation

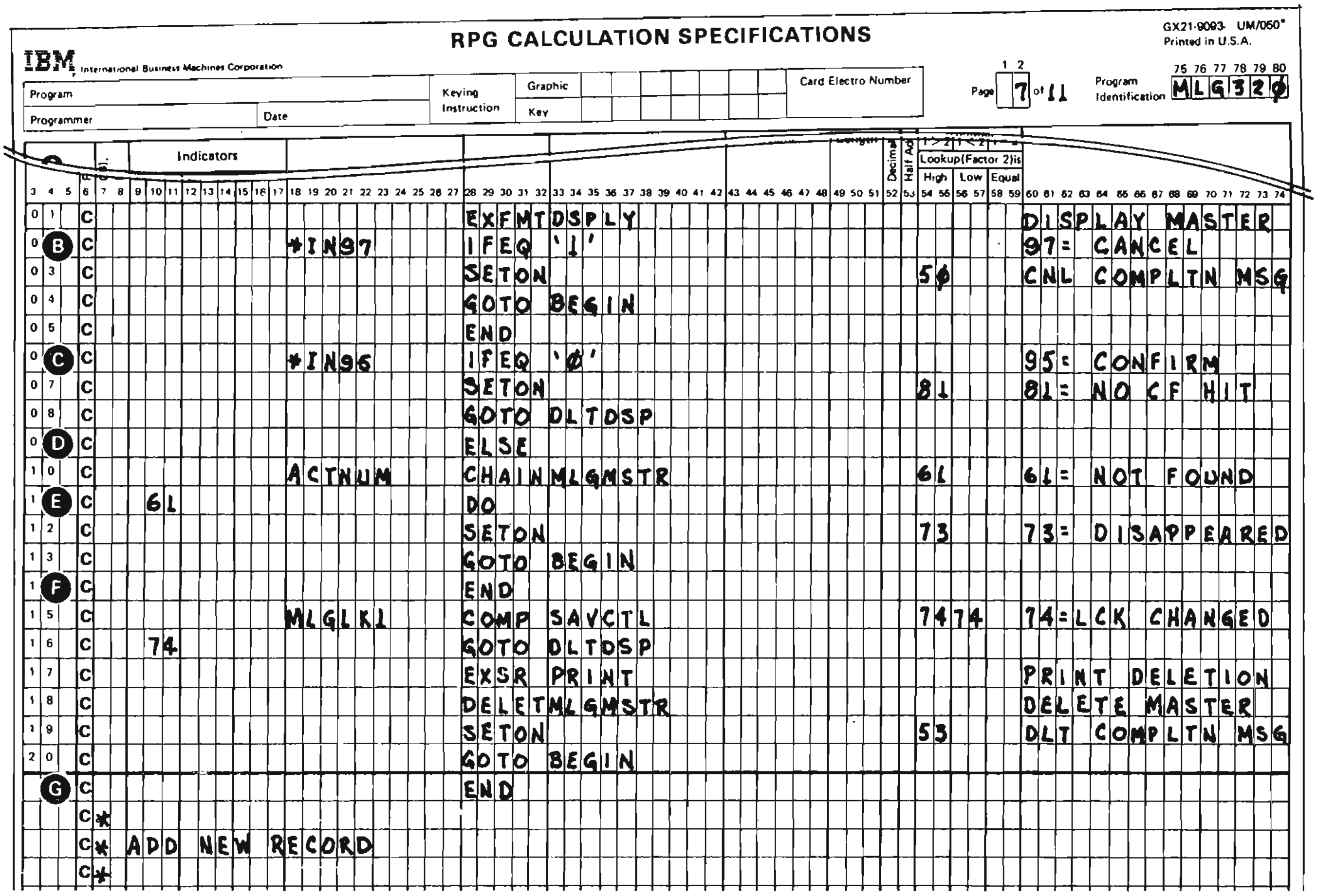
RPG CALCULATION SPECIFICATIONS

GX21-9093 UM/050* Printed in U.S.A.

Page 6 of 11 — Program Identification MLG32Ø

```
01 C                SETON                   52     CHG COMPLTN MSG
02 C                GOTO  MAKCHG
03 C     MAKCHG     TAG                            CHANGE MASTER
04 C                SETON                   45     PRT BEFORE
05 C                EXSR  PRINT                    PRINT OLD MSTR
06 C                SETOF                   45     PRT BEFORE
07 C                MOVE  XACTTP     ACTTYP        MOVE CHANGED
08 C                MOVE  XNAME      NAME           FIELDS TO
09 C                MOVE  XADDR      ADDR           MASTER FIELDS
10 C                MOVE  XCITY      CITY
11 C                MOVE  XSTATE     STATE
12 C                MOVE  XZIP       ZIP
13 C                ADD   1          MLGLK1        ADD TO LOCK CTL
14 C                EXCPT UPHDR                    UPDATE MASTER
15 C                SETON                   35     35= CTL PRT LIN
16 C                EXSR  PRINT                    PRINT NEW MSTR
17 C                SETOF                   35
18 C                GOTO  BEGIN
19 C*
20 C* DELETE RECORD
   C*
   C     DLTDSP     TAG
   C                MOVE  MLGLK1     SAVCTL
   C                SETON                   85     85=ALLW CMD KEY
```

Figure 11-4 (Part 2 of 4). Calculation Specifications for MLG320 Maintenance Program



IBM International Business Machines Corporation

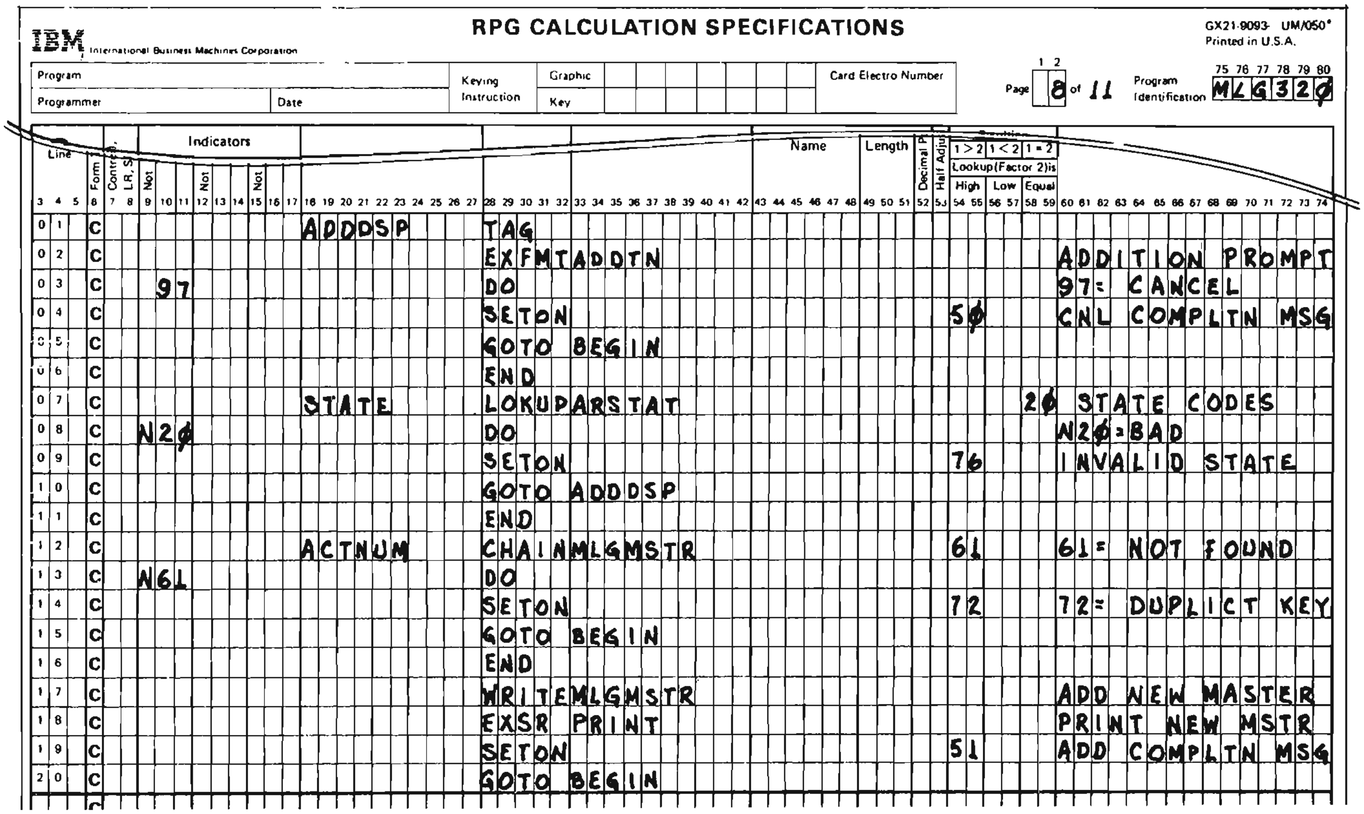
RPG CALCULATION SPECIFICATIONS

GX21-9093- UM/050*
Printed in U.S.A.

Program | Programmer | Date | Keying Instruction | Graphic | Key | Card Electro Number

Page 7 of 11 Program Identification MLG32Ø

```
01 C                     EXFMTDSPLY                      DISPLAY MASTER
0B C           *IN97     IFEQ '1'                        97= CANCEL
03 C                     SETON                     5Ø    CNL COMPLTN MSG
04 C                     GOTO BEGIN
05 C                     END
0C C           *IN96     IFEQ 'Ø'                        95= CONFIRM
07 C                     SETON                     81    81= NO CF HIT
08 C                     GOTO DLTDSP
0D C                     ELSE
10 C           ACTNUM    CHAINMLGMSTR              61    61= NOT FOUND
1E C   61                DO
12 C                     SETON                     73    73= DISAPPEARED
13 C                     GOTO BEGIN
1F C                     END
15 C           MLGLK1    COMP SAVCTL               7474  74=LCK CHANGED
16 C   74                GOTO DLTDSP
17 C                     EXSR PRINT                      PRINT DELETION
18 C                     DELETMLGMSTR                    DELETE MASTER
19 C                     SETON                     53    DLT COMPLTN MSG
20 C                     GOTO BEGIN
 G C                     END
   C*
   C* ADD NEW RECORD
   C*
```



IBM International Business Machines Corporation

RPG CALCULATION SPECIFICATIONS

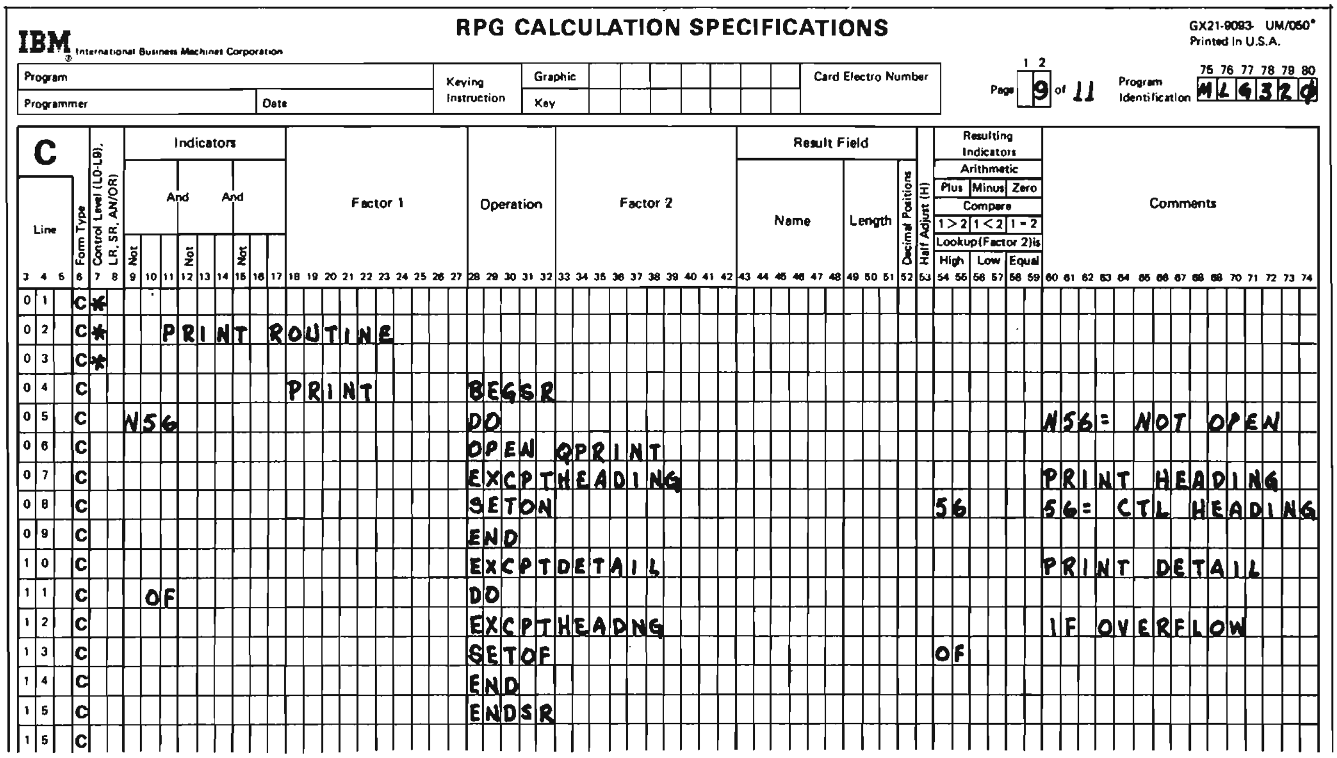
GX21-9093- UM/050*
Printed in U.S.A.

Program | Programmer | Date | Keying Instruction | Graphic | Key | Card Electro Number

Page 8 of 11 Program Identification MLG32Ø

```
01 C           ADDDSP    TAG
02 C                     EXFMTADDTN                      ADDITION PROMPT
03 C   97                DO                              97= CANCEL
04 C                     SETON                     5Ø    CNL COMPLTN MSG
05 C                     GOTO BEGIN
06 C                     END
07 C           STATE     LOKUPARSTAT                   2Ø STATE CODES
08 C  N2Ø                DO                              N2Ø=BAD
09 C                     SETON                     76    INVALID STATE
10 C                     GOTO ADDDSP
11 C                     END
12 C           ACTNUM    CHAINMLGMSTR              61    61= NOT FOUND
13 C  N61                DO
14 C                     SETON                     72    72= DUPLICT KEY
15 C                     GOTO BEGIN
16 C                     END
17 C                     WRITEMLGMSTR                    ADD NEW MASTER
18 C                     EXSR PRINT                      PRINT NEW MSTR
19 C                     SETON                     51    ADD COMPLTN MSG
20 C                     GOTO BEGIN
   C
```

Figure 11-4 (Part 3 of 4). Calculation Specifications for MLG320 Maintenance Program



IBM International Business Machines Corporation

RPG CALCULATION SPECIFICATIONS

GX21-9093- UM/050*
Printed in U.S.A.

Program | Programmer | Date | Keying Instruction | Graphic | Key | Card Electro Number

Page 9 of 11 Program Identification MLG32Ø

```
01 C*
02 C*    PRINT ROUTINE
03 C*
04 C           PRINT     BEGSR
05 C  N56                DO                                N56= NOT OPEN
06 C                     OPEN QPRINT
07 C                     EXCPTHEADING                      PRINT HEADING
08 C                     SETON                     56      56= CTL HEADING
09 C                     END
10 C                     EXCPTDETAIL                       PRINT DETAIL
11 C   OF                DO
12 C                     EXCPTHEADNG                        IF OVERFLOW
13 C                     SETOF                     OF
14 C                     END
15 C                     ENDSR
15 C
```

Figure 11-4 (Part 4 of 4). Calculation Specifications for MLG320 Maintenance Program

This program specifies three files:

- MLG320D - a work station file
- MLGMSTL - a data base file
- QPRINT - a print file

The program causes data to be processed as described under *Program Flow* in this chapter.

The *statement EXFMT* (execute format) is followed by a test for an end-of-program request. If requested, LR is set on and the program returns (RETRN).

The *ACTNUM field* and the request entered by the work station user are validated. If an error occurs, the PROMPT display is shown again. The master record is released immediately.

The program branches to one of the following four routines, depending on the work station user's request.

The *INQUIRY ONLY routine* is an inquiry that branches to the PROMPT display.

The *CHANGE DISPLAY routine* moves a copy of the master record fields to the change fields so the data in these fields can be changed. The CHANGE format is displayed and the program recognizes the following situations:

- The work station user wants to cancel the request.
- The record was deleted since the request to change it was made.
- The value of MLGLK1 has changed.

When a change is to be made, the existing master record is printed, the changed fields replace the existing fields, the value of MLGLK1 is incremented, the master record is updated (EXCPT UPDHDR), and the changed master record is printed (EXSR PRINT).

The *DELETE RECORD routine* saves the value of MLGLK1 and displays the record to be deleted (EXFMT DSPLY). The program recognizes the following situations:

- The work station user wants to cancel the request.
- The record has just been deleted by someone else.
- CF5 or CF7 was not pressed.

DELET MLGMST deletes the record. No program can access the record.

The *ADD NEW RECORD routine* displays the ADDTN format (EXFMT ADDTN) and recognizes the following situations:

- The work station user wants to cancel the request.
- The account number has just been used by someone else.

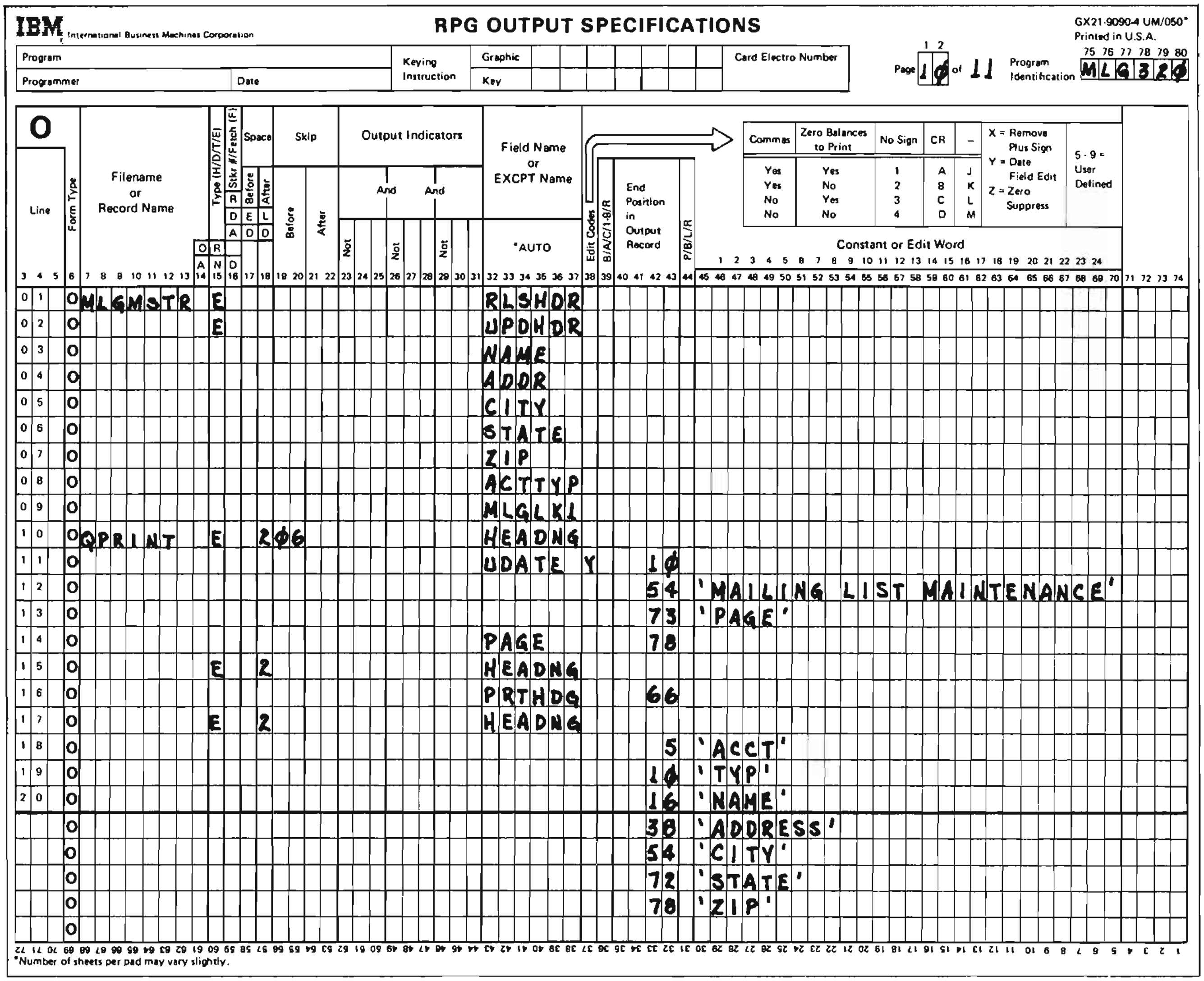
When an addition is to be made, the new record is added to the file (WRITE MLGMSTR) and printed (EXSR PRINT). Because the WRITE statement writes all fields for the file, output specifications are not needed.

The *PRINT routine* does printing by exception time output (E in position 15). The print file is opened the first time printing is requested. Printed output does not occur if the program is used for inquiry only. The report title is printed the first time the routine is used. Indicator 56 is set on to prevent the print file from being opened again and the report title from being printed. The detail line is printed (EXCPT DETAIL), and overflow is checked. (Because all printing is done through exception output, the normal RPG III cycle control cannot be used for either heading or overflow.) If there is overflow, the heading is printed.

The output specifications have two records for the MLGMSTR record. The first (RLSHDR) is one with no fields specified. This releases the lock on the record so that other jobs on the system can request an update to the same record. This method of releasing the lock is more efficient than updating the record with the same information. The other record (UPDHDR) does the update for a changed condition.

The output to print the heading lines and the record is in the QPRINT file. Conditioned constants on the record specify the type of record being printed.

The output specifications in Figure 11-5 specify the layout of the printer output.



IBM International Business Machines Corporation

RPG OUTPUT SPECIFICATIONS

GX21-9090-4 UM/050*
Printed in U.S.A.

Program
Programmer
Date
Keying Instruction: Graphic / Key
Card Electro Number
Page 1Ø of 11
Program Identification MLG32Ø

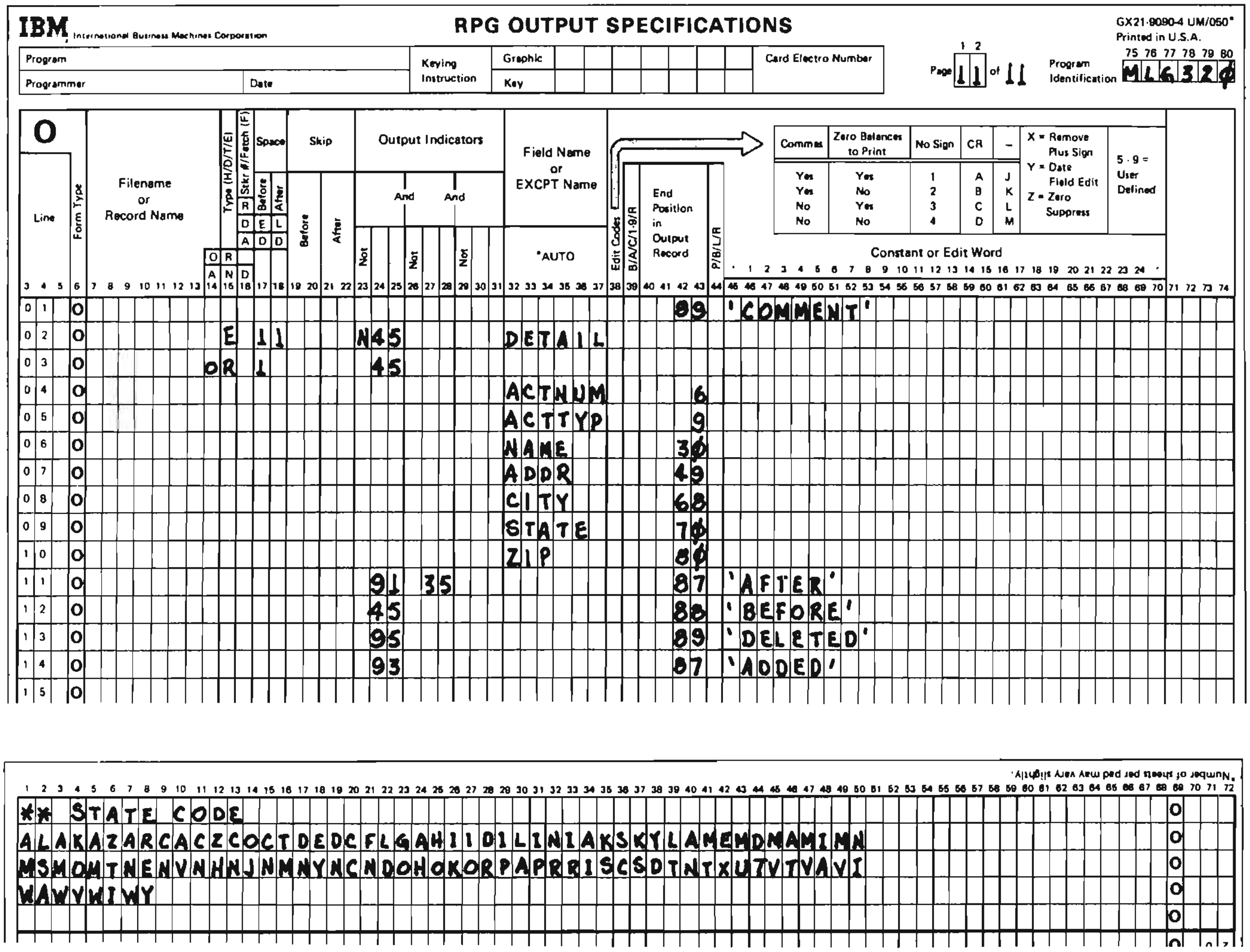
O

Line	Form Type	Filename or Record Name	Type (H/D/T/E)	Space Before	Space After	Skip Before	Skip After	Output Indicators	Field Name or EXCPT Name	Edit Codes	B/A/C/1-9/R	End Position in Output Record	P/B/L/R	Constant or Edit Word
01	O	MLGMSTR	E						RLSHDR					
02	O		E						UPDHDR					
03	O								NAME					
04	O								ADDR					
05	O								CITY					
06	O								STATE					
07	O								ZIP					
08	O								ACTTYP					
09	O								MLGLK1					
10	O	QPRINT	E		2	Ø6			HEADNG					
11	O								UDATE	Y		1Ø		
12	O											54		'MAILING LIST MAINTENANCE'
13	O											73		'PAGE'
14	O								PAGE			78		
15	O		E		2				HEADNG					
16	O								PRTHDG			66		
17	O		E		2				HEADNG					
18	O											5		'ACCT'
19	O											1Ø		'TYP'
20	O											16		'NAME'
	O											38		'ADDRESS'
	O											54		'CITY'
	O											72		'STATE'
	O											78		'ZIP'
	O													

Commas / Zero Balances to Print / No Sign / CR / –: Yes Yes 1 A J; Yes No 2 B K; No Yes 3 C L; No No 4 D M
X = Remove Plus Sign; Y = Date Field Edit; Z = Zero Suppress; 5-9 = User Defined

*Number of sheets per pad may vary slightly.

Figure 11-5 (Part 1 of 2). Output Specifications for MLG320 Maintenance Program



IBM International Business Machines Corporation

RPG OUTPUT SPECIFICATIONS

GX21-9090-4 UM/050*
Printed in U.S.A.

Program		Keying Instruction	Graphic	Card Electro Number
Programmer	Date		Key	

Page 11 of 11 — Program Identification MLG32Ø

Line	Form Type	Filename or Record Name	Type (H/D/T/E)	OR / AND	Space Before	Space After	Output Indicators	Field Name or EXCPT Name	End Position in Output Record	Constant or Edit Word
01	O								89	'COMMENT'
02	O		E		1	1	N45	DETAIL		
03	O			OR	1		45			
04	O							ACTNUM	6	
05	O							ACTTYP	9	
06	O							NAME	3Ø	
07	O							ADDR	49	
08	O							CITY	68	
09	O							STATE	7Ø	
10	O							ZIP	8Ø	
11	O						91 35		87	'AFTER'
12	O						45		88	'BEFORE'
13	O						95		89	'DELETED'
14	O						93		87	'ADDED'
15	O									

*Number of sheets per pad may vary slightly.

```
** STATE CODE
ALAKAZARCACZCOCTDEDCFLGAHIIDILINIAKSKYLAMEMDMAMIMN
MSMOMTNENVNHNJNMNYNCNDOHOKORPAPRRISCSDTNTXUTVTVAVI
WAWVWIWY
```

Figure 11-5 (Part 2 of 2). Output Specifications for MLG320 Maintenance Program

ALTERNATIVE SOLUTION FOR THIS APPLICATION APPROACH

In this application approach, the technique of using change fields (for example, XNAME) was used to hold the new values while the master record containing the disk record fields (for example, NAME) was reaccessed. The purpose of reaccessing the master record was to avoid a locked record condition while the work station user made a decision. If the same field names are used, reaccessing the disk record after reading the changes from the display will cause the field values from the disk record to overlay the changed values from the work station user so that the changes would be lost. This situation is similar to other maintenance programs in this publication where different names are used for the change fields.

It is possible to use the same field names and not lock the master record during the user decision time by using a DDS option and minor additional coding. The major advantage of using the same names is to reduce the maintenance considerations of adding new fields to the master record. For example, in the program as shown, if a new field is added to the master record that must be maintained in this program, the new field must be described in multiple places within the program.

An alternative solution is to use the DDS keyword RTNDTA. This allows the data from the work station to be reread from the main storage buffer after accessing the master record. Without the DDS keyword, the data would be read from the work station again. This can cause confusion to the operator and also causes a loss of performance. The DDS keyword RTNDTA allows a simple form of rereading the same data.

The basic steps in the sequence of events for a CHANGE request were coded as:

1. EXFMT the PROMPT format
2. For a CHANGE request, CHAIN to the master record
3. Release the lock on the record
4. Save the lock control field
5. Move the master fields to change fields
6. EXFMT the CHANGE format
7. CHAIN to the master record
8. Move the change fields to the master field
9. Update the master

With the RTNDTA keyword approach, Step 5 could be removed. Instead of the current Step 8, a READ to the CHANGE format would be used. This would overlay the master fields that were reset by the CHAIN statement in Step 7.

RECOVERY CONSIDERATIONS

If incorrect data has been entered for a transaction, the RPG maintenance program MLG320 can be executed again to change the incorrect data.

If there is a system failure while the work station user is entering transactions, one of the following approaches could be used:

- When the master file MLGMSTP is created by the Create Physical File (CRTPF) command, there is a parameter named force write ratio (FRCRATIO) to control the frequency in which records are written to auxiliary storage. Using the default for this parameter provides the best performance. If the default (*NONE) is used, the work station users should check the last several transactions they entered to ensure they were recorded in auxiliary storage.

  The system may not write the records to auxiliary storage in the same sequence in which the updates occurred. All of the transactions entered may not be reflected because some records may have been in main storage at the time the system failure occurred and cannot be recovered.

  The listing that is printed upon the exit from the application may be used to determine what transactions were entered. The transactions will be listed in the correct sequence, but the listing may not be completely up-to-date. Printed records are spooled and are kept in buffers in main storage; therefore, some records may be lost if a system failure occurs.

- If the force write ratio parameter value is specified as 1, the work station users should look at their last entry to determine if it exists before continuing. In this situation, the printed listing may not be as up-to-date as the data base.

- Use data base logging where the log file has a force ratio of 1. Synchronize the data base log and the data base file after a system failure.

To recover from damaged objects, two approaches are:

- Reenter the transactions entered since the last backup was made.

- Use data base logging to log the changes to the file. The master file can be backed up periodically. The data base log can be backed up daily (more frequently if desired). To recover the file, restore the backup and execute a special recovery program to reapply the log entries.

Appendix A. An Approach to Using Libraries

System/38 supports multiple libraries, which allow you to organize your work so that your application is easy to use.

When designing an application, you need to consider many requirements, such as production operations, program development, backup, security, and operator training. The approach to using libraries presented in this appendix considers these requirements in a mailing list application. This appendix will:

- Describe multiple mailing list libraries and their purpose

- Show how to create these libraries

- Show how to use these libraries

MAILING LIST LIBRARIES AND THEIR PURPOSE

There are two groups of mailing list libraries, each with a different owner (a user with a unique user profile). These two owners are:

- Development programmer (DEVPGMR)

- Administrative programmer (ADMPGMR)

The development programmer does the initial program development and program maintenance. To help protect against unintentionally modifying production data, the development programmer is not authorized to create or delete objects in the production library.

The administrative programmer controls the production system. He does very little actual programming. He copies the source code created or changed by the development programmer and creates or re-creates that code for the production system.

While this approach may appear cumbersome and requires additional auxiliary storage and program compilations, it does provide a basis for better control than a single library and a single programmer. From a system management point of view, it may be highly desirable to control changes to the production system.

The mailing list libraries, their owners, and their purposes are summarized below:

Library	Owner	Purpose
MLGLIBDEV	DEVPGMR	Program development and maintenance objects
MLGLIBBLD	ADMPGMR	Build library for production programs
MLGLIBPGM	ADMPGMR	Production programs and display files
MLGLIBOP	ADMPGMR	Production data that is infrequently saved
MLGLIBOP1	ADMPGMR	Production data that is frequently saved

The relationship of these libraries is indicated in Figure A-1.

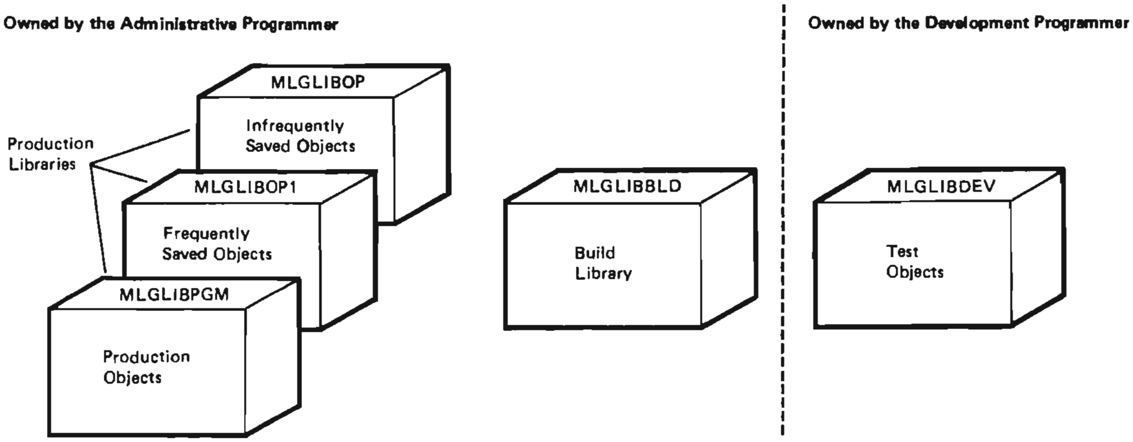


Figure A-1. The Mailing List Libraries and Their Owners

The discussion that follows describes these libraries and their objects in more detail.

MLGLIBDEV is used for program development, testing, and maintenance. It is created as a test library to allow the default use of the Enter Debug Mode (ENTDBG) command to permit the updating of files within this library. This library contains the following objects:

- Source files–QCLSRC, QDDSSRC, QRPGSRC, QUDSSRC; these source files are created for testing such that only the development programmer can update them. The source files would be created in a manner similar to that described in Chapter 6. The authority to use the files would then be changed so that only the development programmer can update them. This would be done by using commands such as:

  ```
  RVKOBJAUT QCLSRC.MLGLIBDEV OBJTYPE(*FILE) USER(*PUBLIC)
        AUT(*ADD *DLT *UPD)
  ```

- Physical files–contain test data.

- Logical files–used to access the test data in physical files.

- Set library list program–contains the RPLLIBL command for the following libraries: MLGLIBDEV, MLGLIBPGM, QGPL, QTEMP, QIDU, QRPG. If the reformat utility is needed in the application, QS3E would be added at the end of the list. The program would be created in a manner similar to that described in Chapter 6.

- Job description–contains the same library list as the replace library list program. The job description would be created in a manner similar to that described in Chapter 6.

- Job description data area–contains the name of the job description; allows CL programs that require a job description to be written independently of the library that contains the CL program (for more information on this concept, see Appendix B).

  The data area would be created by the following command:

  ```
  CRTDTAARA JOBDTAARA.MLGLIBDEV TYPE(*CHAR)
        LEN(10) VALUE(MLGLIBDEV)
  ```

- Programs–test versions of the production programs.

- Display files–test copies of the production files.

MLGLIBBLD contains source files and serves as a staging area for creating objects. This library is authorized such that only the administrative programmer can add objects. It contains the following:

- Source files–QCLSRC, QDDSSRC, QRPGSRC, QUDSSRC; these source files are created such that only the adminstrative programmer can update them.

- Set library list program–contains the RPLLIBL command for the following libraries: MLGLIBBLD, MLGLIBPGM, MLGLIBOP, MLGLIBOP1, QGPL, QTEMP, QIDU, QRPG. If the reformat utility is needed in the application, QSE3 would be added at the end of the list.

- Job description–contains the same library list as the set library list program.

- Job description data area–contains the name of the job description; allows CL programs requiring a job description to be written independently of the library that contains the CL program.

- Objects–all objects are created in this library and then moved to one of the three production libraries (MLGLIBPGM, MLGLIBOP1, MLGLIBOP).

MLGLIBPGM contains the production programs and files needed for this approach to the mailing list application. This library is created such that only the administrative programmer can add objects. It contains the following:

- Source files–QCLSRC, QDDSSRC, QRPGSRC, QUDSSRC; these source files are created such that only the administrative programmer can update them.

- Programs–used to process the production data.

- Display files–for the program.

MLGLIBOP contains production objects infrequently saved. It is created such that only the administrative programmer can add objects and contains the following:

- Physical files–infrequently saved.
- Logical files–used to access the production data.
- Set library list program–contains the RPLLIBL command for the following libraries: MLGLIBPGM, MLGLIBOP, MLGLIBOP1, QGPL, QTEMP, QIDU, QRPG. If the reformat utility is needed in the application, QS3E would be added at the end of the list.
- Job description–contains the same library list as the set library list program and is used for batch jobs.
- Job description data area.
- Data areas–infrequently saved; communicates data, such as CL variable values, between the programs.

MLGLIBOP1 contains production objects frequently saved. It is created such that only the administrative programmer can add objects and contains the following:

- Physical files.
- Data areas.

USING THE MAILING LIST LIBRARIES

The intent of the following discussion is to show how the development programmer and administrative programmer interact as owners of various libraries within the same application.

Sign-On for Development Programmer

The development programmer signs on using a unique password. The library list is created by calling the set library list program in the library MLGLIBDEV. Using the programmer menu, the compilations are submitted to batch with a unique job description. Defaults are generally used on all create commands. Consequently, when creating a logical file, the maintenance default of *immediate* is used during the test phase, even though that logical file may be designed for *rebuild* maintenance. This simplifies testing by eliminating repetitive specifications each time an object is re-created.

The development programmer could be prevented from adding objects to QGPL (assuming that all of his objects are in one library, MLGLIBDEV) by removing the public authority to add to QGPL. Removing the public authority to the QBATCH job description will assist the development programmer in specifying the correct entries on the programmer menu.

Program Development and Maintenance

When the development programmer needs to develop programs or update certain objects, the production programs and production display files can be used, along with a test version of any object that is being changed. The production data in MLGLIBOP and MLGLIBOP1 should not be used.

Sign-On for the Administrative Programmer

The administrative programmer signs on using a unique password. The library list is created by calling the set library list program in the library MLGLIBBLD. This library is owned by the administrative programmer and is not used by the other production library users.

Placing Objects into Production Libraries

After the development programmer has created the source for the application, a transmittal form is prepared for the administrative programmer. This transmittal form could be either informal or formal. The administrative programmer then copies the source into his library, MLGLIBBLD, and recreates the objects in this library. Options on create commands, such as MAINT(*REBLD) or TEXT description, are specified at this time.

After the administrative programmer has satisfied himself that the object is correct, he moves it to the appropriate production library depending upon object type and backup requirements. If an object already exists in a production library, it must be deleted prior to moving the new version of the object. The production source is then copied to the source file in MLGLIBPGM to serve as the official source for the production system.

If a data file is being replaced, the new version is created in MLGLIBBLD. The Copy File (CPYF) command can then be used to copy any data from the old version to the new version. The old version is then deleted and the new version moved to the appropriate library.

The library MLGLIBBLD allows the administrative programmer to assemble all the objects necessary so they may be placed into a production library at the same time. When the objects are actually moved to the production library, the users of the production library must *not* be using the objects that are being moved. By successfully compiling all of the objects into MLGLIBBLD, the administrative programmer can minimize the time when the users of the production library cannot use the production objects and the number of times an incorrect change makes the production system inoperable.

Critical Applications

When critical applications are being updated, the changes could be made first to a training library. This library would contain duplicates of the important objects. In this way, the work station users can be trained and the programs can be tested at the same time without unintentionally changing the production data.

Sign-On for the Users of the Production System

The set library list program is called when the work station user signs on and requests a function from the production system. This program is placed in MLGLIBOP, instead of in MLGLIBPGM, to protect against unintentional changes which may be made by the development programmer if he can execute programs in MLGLIBPGM.

Online Backup

Online backup is internally copying a file using the CPYF or CPYFI command. To allow normal recovery, the file should not be changed while it is being copied.

Online backup allows:

- Faster backup and recovery operations.
- The copy of the data base to be saved offline at any time without impacting the production system.
- Less frequent offline saving of objects (depending upon the recovery requirements).

To provide online backup in an unattended environment, a batch program is run using the job description in library MLGLIBOP. The specified files and data areas in MLGLIBOP1 are copied to a user-created system-wide backup library or an application-oriented library. To do this, the Copy File (CPYF) command is used with the COMPRESS(*NO) parameter to provide a high-speed copy function. A data area object cannot be copied, but the contents of a data area object may be read from one library and written to another. The backup library should be saved frequently.

PROGRAMMING CONSIDERATIONS

In some applications it would be easier and provide better control if the development programmer has a unique set of objects and does not have library MLGLIBPGM on his library list. This approach would cause additional auxiliary storage requirements for the system, but a simpler concept is the result.

Appendix B. Using Menus

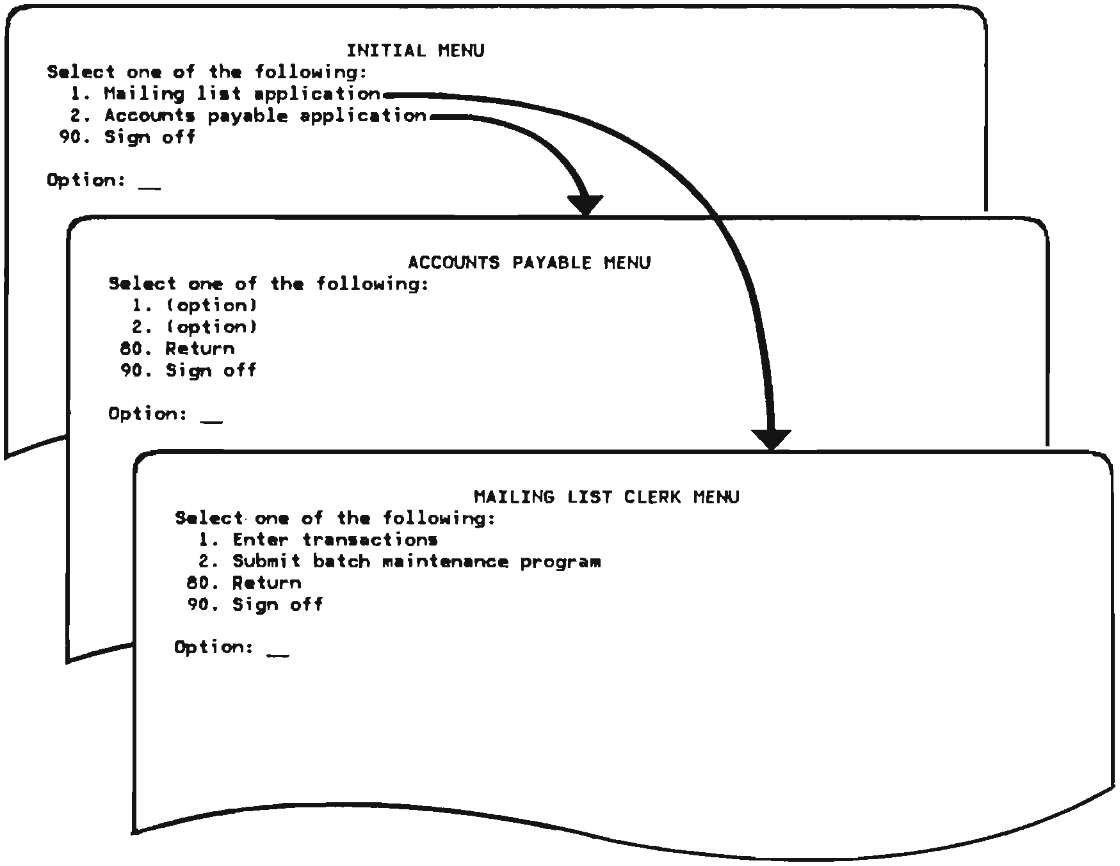
In many applications, you may want to create menus that are appropriate and time-saving for your application. This appendix discusses the following two uses for menus:

- A series of menus
- Batch jobs submitted from a menu

A SERIES OF MENUS

In some applications, you may want to provide various task options on a menu for the work station user. You can do this by creating a series of menus. For example, the work station user who updates the mailing list may also need to update accounts payable. All task options for the mailing list application and the accounts payable application could be on one menu; however, dividing the options by application makes the tasks easier to understand and use, and makes the applications more flexible.

A sample series of menus is shown here:



```
                    INITIAL MENU
Select one of the following:
  1. Mailing list application
  2. Accounts payable application
 90. Sign off

Option: __
```

```
                    ACCOUNTS PAYABLE MENU
Select one of the following:
  1. (option)
  2. (option)
 80. Return
 90. Sign off

Option: __
```

```
                    MAILING LIST CLERK MENU
Select one of the following:
  1. Enter transactions
  2. Submit batch maintenance program
 80. Return
 90. Sign off

Option: __
```

The initial menu is similar to the menu created in Chapter 8. Each option causes a CALL command in a CL program, such as the following, to be executed:

```
     •
     •
     •
  IF (&RESP *EQ 1) CALL MLG005C.MLGLIBOP
  IF (&RESP *EQ 2) CALL APY008C.APYLIBOP
     •
     •
     •
```

Note that a qualified name is used to call the program. The program, MLG005C, replaces the library list for the mailing list application and invokes the mailing list clerk menu program.

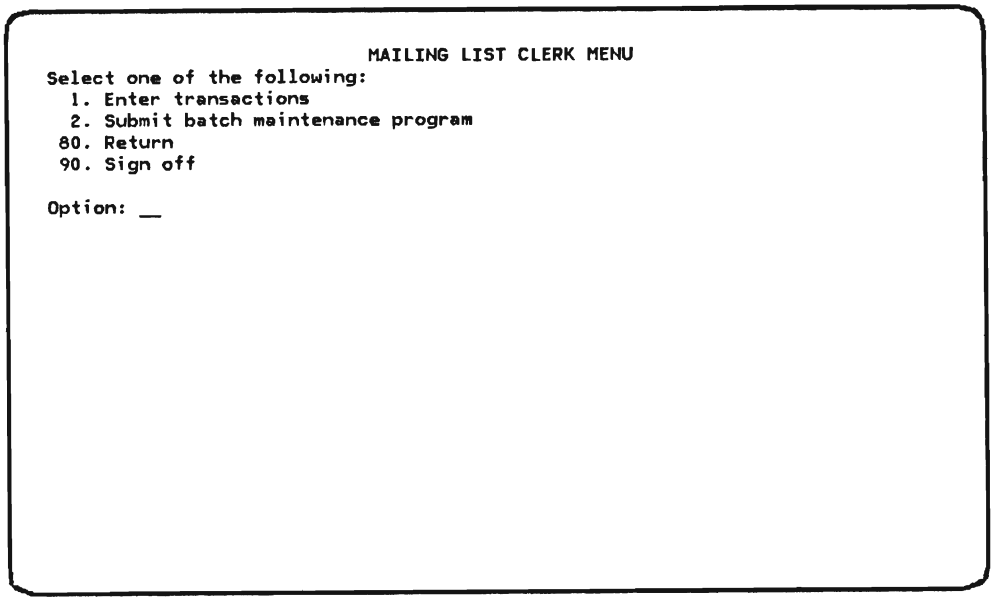
The menus that follow the initial menu each have several options relating to that specific application, a sign-off option, and a return option. The return option causes the previous menu to appear again. (There is no return option on the initial menu because this is the first menu to appear.) The return option is coded as follows:

```
     •
     •
     •
  IF (&RESP *EQ 80) RETURN
     •
     •
     •
```

The Return (RETURN) command in a CL program ends the program and returns to the previous program.

BATCH JOBS SUBMITTED FROM A MENU

The work station user often can determine when a batch job should be submitted. For example, a work station user enters a batch of transactions and wants the batch maintenance program executed. He can do this without the assistance of the system operator by selecting an option on a menu like the one shown here:

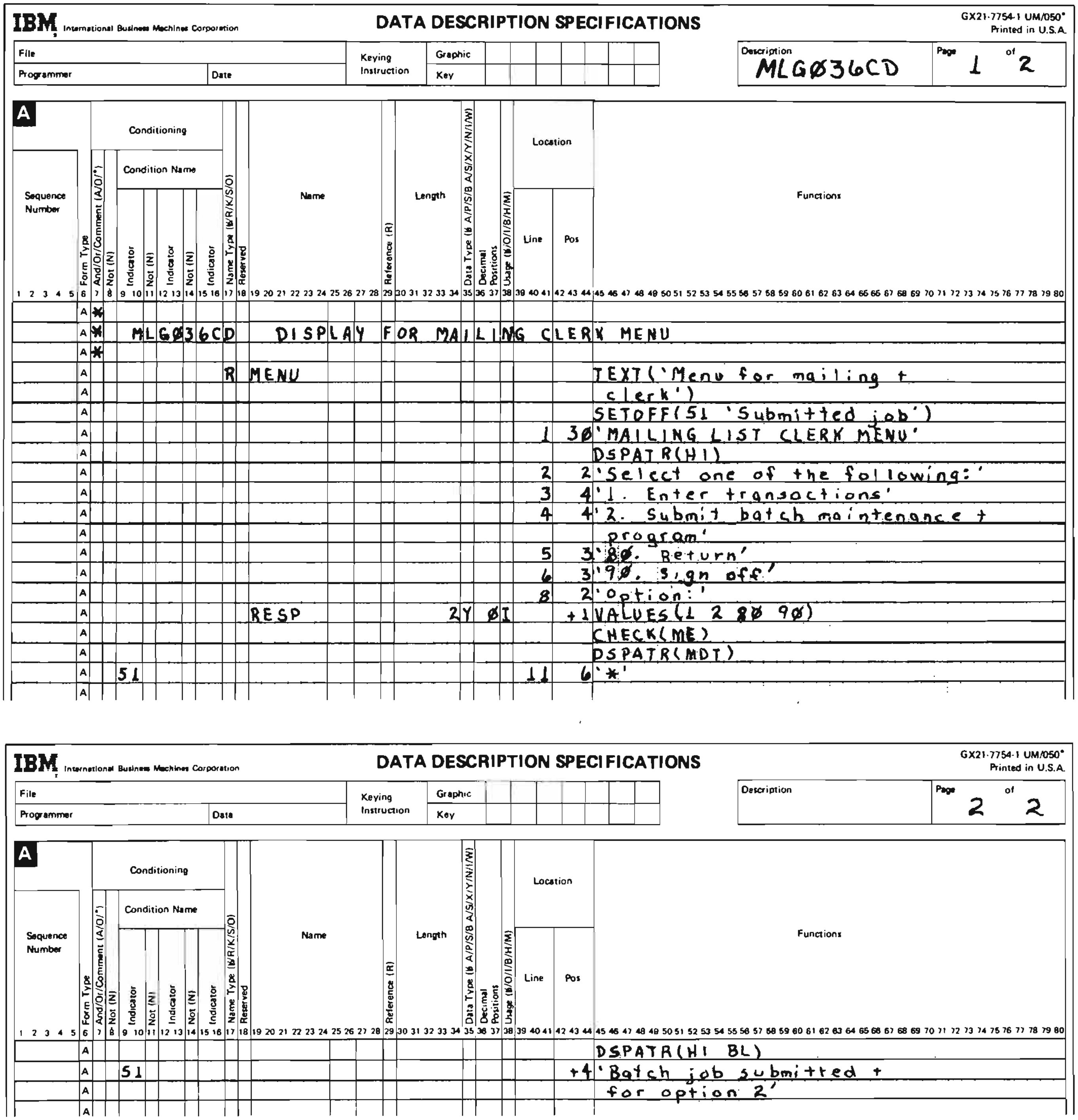


```
                                   MAILING LIST CLERK MENU
Select one of the following:
  1. Enter transactions
  2. Submit batch maintenance program
 80. Return
 90. Sign off

Option: __
```

If the work station user selects option 1, he can enter data into a transaction file using the DFU application MLG110U (which was discussed in Chapter 7).

If the work station user selects option 2, he submits a job to update the master file using the RPG III program MLG311 (which was discussed in Chapter 7).

The DDS for the display file that creates this menu are shown in Figure B-1. The display file is named MLG036CD.



IBM International Business Machines Corporation — DATA DESCRIPTION SPECIFICATIONS — GX21-7754-1 UM/050* Printed in U.S.A.

File | Programmer | Date | Keying Instruction: Graphic, Key | Description: MLGØ36CD | Page 1 of 2

```
     A*
     A*  MLGØ36CD     DISPLAY FOR MAILING CLERK MENU
     A*
     A          R MENU                      TEXT('Menu for mailing +
     A                                      clerk')
     A                                      SETOFF(51 'Submitted job')
     A                                  1 3Ø'MAILING LIST CLERK MENU'
     A                                      DSPATR(HI)
     A                                  2  2'Select one of the following:'
     A                                  3  4'1. Enter transactions'
     A                                  4  4'2. Submit batch maintenance +
     A                                      program'
     A                                  5  3'8Ø. Return'
     A                                  6  3'9Ø. sign off'
     A                                  8  2'Option:'
     A            RESP           2Y ØI  +1VALUES(1 2 8Ø 9Ø)
     A                                      CHECK(ME)
     A                                      DSPATR(MDT)
     A  51                             11  6'*'
     A
```

IBM International Business Machines Corporation — DATA DESCRIPTION SPECIFICATIONS — GX21-7754-1 UM/050* Printed in U.S.A.

File | Programmer | Date | Keying Instruction: Graphic, Key | Description | Page 2 of 2

```
     A                                      DSPATR(HI BL)
     A  51                                +4'Batch job submitted +
     A                                      for option 2'
     A
```

Figure B-1. DDS for MLG036CD Display File

This display file is similar to MLG035CD. If indicator 51 is set on, an * on the display is highlighted and blinks. This tells the work station user that his input has been accepted. Your application should have some type of feedback (message or intervening display) so that the work station user does not try to submit his input again and cause multiple batch jobs to be submitted for the same function.

The SETOFF keyword specifies that the indicator is to be set off each time a response is sent to the program. The display device data management does not set off the indicator until the option indicators to define the display format are used. Because SETOFF is a record level keyword, it must be specified before the options in the DDS.

If option 80 is selected, the previous menu returns as described under *A Series of Menus* earlier in this appendix.

CL Program with Commands Using Fixed Values

The CL program to execute the options on the mailing list clerk menu is named MLG036C and is similar to MLG035C (which was discussed in Chapter 8). The code for MLG036C follows:

```
     PGM   /* MLG036C MAILING CLERK MENU */
     DCLF   MLG036CD
BEGIN:     SNDRCVF RCDFMT(MENU)
     IF    (&RESP *EQ 1) CHGDTA APP(MLG110U)/* ENTER +
           TRANSACTIONS */
     IF    (&RESP *EQ 2)  +
           DO /* SUBMIT MAINTENANCE */
           SBMJOB JOB(MLGMAINT) JOBD(MLGLIBDEV) RQSDTA('CALL MLG311')
           CHGVAR &IN51 '1' /* SET FOR RESPONSE CONSTANT */
           ENDO
     IF    (&RESP *EQ 80) RETURN
     IF    (&RESP *EQ 90) SIGNOFF
     GOTO BEGIN
     ENDPGM
```

If the work station user selects option 2, a DO group starts. All the commands between DO and ENDDO are executed at this time. Within the DO group is a Submit Job (SBMJOB) and a Change Variable (CHGVAR) command. SBMJOB places a job containing *request data* (which is normally a CL command) on a job queue. The JOB parameter specifies a name for the batch job, which is MLGMAINT. The JOBD parameter specifies the name of the job description, which is MLGLIBDEV (this is the same job description you used to submit batch compilations in Chapters 6, 7, and 8). The RQSDTA parameter specifies the command that is to be executed, which is CALL MLG311. That command and its parameter are enclosed in apostrophes. The RQSDTA parameter accepts only a single value (as opposed to a list of values) that can be either a variable or a character string. In this case, the apostrophes define the character string. The request to call MLG311 is the same command that was submitted from the programmer menu in Chapter 7. In that example there is a discussion of the work station user submitting a CL program that includes the call to the maintenance program, a Copy File (CPYF) command to permit the backup of all transaction batches, and a Clear Physical File Member (CLRPFM) command. This approach could be used instead of invoking the RPG III program MLG311.

The Change Variable (CHGVAR) command sets on indicator 51 by assigning the variable &IN51 a value of 1. &IN51 is the name used for indicator 51 in the display file. The indicators listed in the display file are automatically declared to the program by the Declare File (DCLF) command the same way the &RESP field is declared (see *Creating the Control Language Program MLG035C* in Chapter 8). When indicator 51 is on, the following message appears:

- Batch job submitted for option 2

Using a Data Area for a Job Description

In the previous example, the JOBD parameter value of the SBMJOB command is MLGLIBDEV. This means the same program could not be used for both development and production. To allow joint use, a data area can be established in each of the environments that contains the name of the job description to be used. This approach was described in Appendix A.

Assume that a data area exists in each environment (as described in Appendix A) by the name of JOBDDTAARA. The CL program executing the SBMJOB command would include:

```
     •
     •
 DCLDTAARA JOBDDTAARA
     •
 RCVDTAARA JOBDDTAARA
     •
 SBMJOB JOB(MLGPRINT) JOBD(&JOBDDTAARA) RQSDTA('CALL MLG311')
     •
     •
```

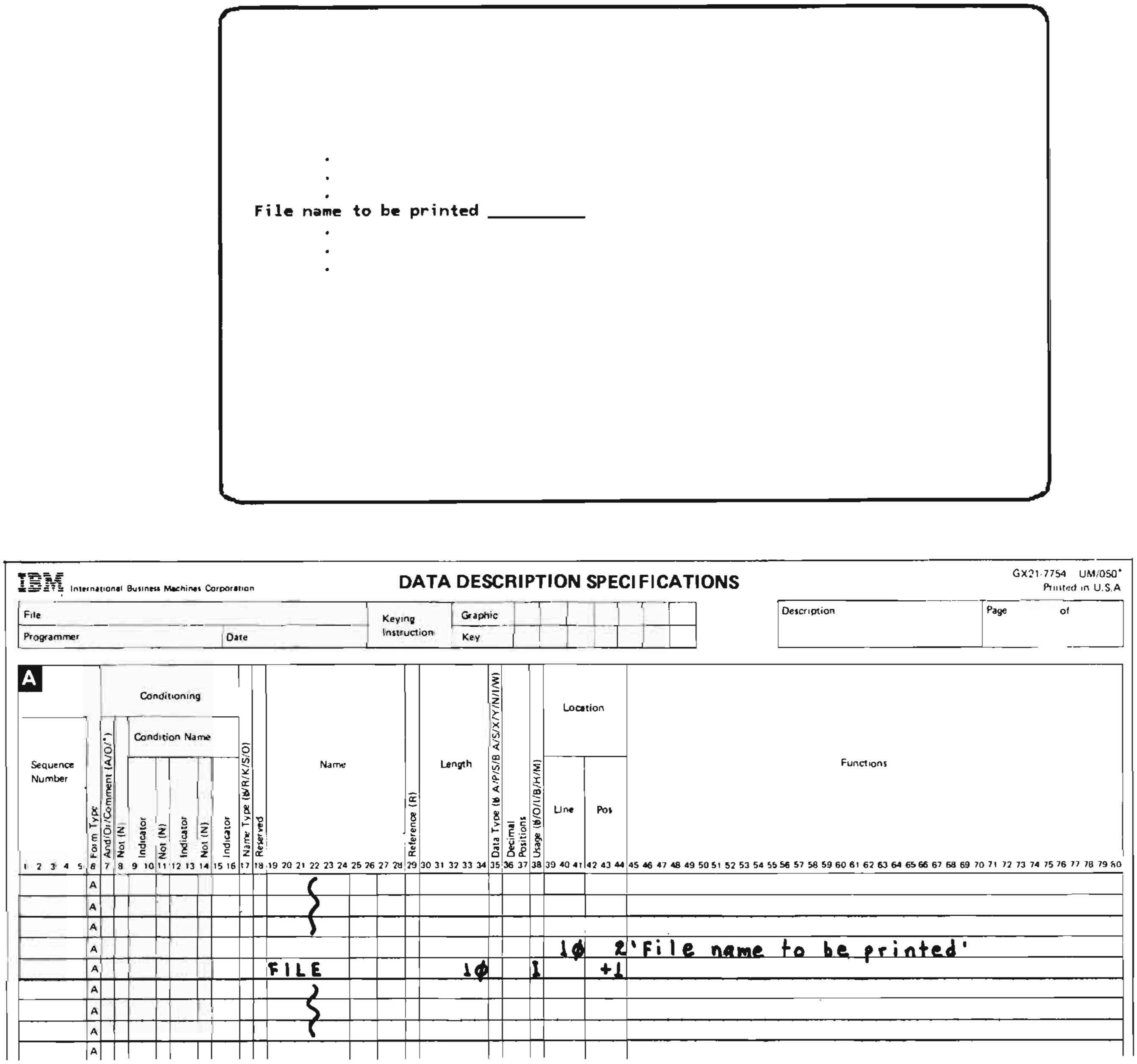
Note that the name of the data area is specified as an object name (no &) in the Declare Data Area (DCLDTAARA) and Receive Data Area (RCVDTAARA) commands and as a variable (preceded by &) when the name is specified for the JOBD parameter. The object name (no &) is used when the object itself is specified.

At the time the program is created, the data area is accessed to determine its attributes similarly to the way an externally described data file is accessed to determine its attributes. Because a qualified name was not used, the program will use a data area named JOBDDTAARA through the library list. Depending upon how the library list is established, the program will access a different value. Assume the library structure described in Appendix A is used. In this situation, the value accessed will be MLGLIBDEV for the development library or MLGLIBOP from the production libraries. The RCVDTAARA command retrieves the actual value from the data area and places it in the CL program. The submitted job will now use the same library list that was used by the interactive program.

Another alternative is to use the same name for the job descriptions (such as MLGLIB) in both library lists. This would allow the program to use a constant name instead of extracting the name from a data area.

Using a Single Variable Character Value

In the previous section (*CL Program with Commands Using Fixed Values*), the Submit Job (SBMJOB) command submitted a job with the same command; that is SBMJOB always submitted a job with the command CALL MLG311. This type of command is appropriate when the work station user enters a fixed value for a response (in this case, option 2). If the menu prompts the work station user to key in a variable entry, such as a file name, an override command must be used. An example of this type of menu and the DDS for this part of the menu are shown in Figure B-2.



```
        .
        .
        .
File name to be printed __________
        .
        .
        .
```

IBM International Business Machines Corporation

DATA DESCRIPTION SPECIFICATIONS

GX21-7754 UM/050*
Printed in U.S.A

File | Keying Instruction | Graphic | Key | Description | Page of
Programmer | Date

```
     A                                       1Ø  2'File name to be printed'
     A            FILE          1Ø   I        +1
```

Figure B-2. Menu and DDS for a Variable Command Program

The display file allows input to the variable named FILE. Because the display file is an externally described file, the variable named FILE will be automatically declared within the CL program. The variable name is changed to &FILE. The & is used within CL programs to distinguish variable names from constants.

Key parts of the CL program that creates the menu follow:

```
     .
     .
     .
  DCL &CMD TYPE(*CHAR) LEN(100)
     .
     .
     .
  CHGVAR &CMD VAR('CALL MLGxxxC PARM(' *CAT &FILE *CAT ')')
  SBMJOB JOB(MLGPRINT) JOBD(MLGLIBDEV) RQSDTA(&CMD)
     .
     .
     .
```

The CL program that executes as a result of a response to the menu follows:

```
  PGM PARM(&FILE) /* MLGxxxC */
  DCL &FILE TYPE(*CHAR) LEN(10)
  OVRDBF MLGMSTP TOFILE(&FILE)
  CALL MLG615
  ENDPGM
```

The CL program that displays the menu declares a variable named &CMD, which is a 100-character field. This variable becomes the command that is within the job that is submitted to the batch job queue. Note that a variable cannot be within the job that is submitted directly to the batch job queue; however, a character string can be within a job that is submitted to the batch job queue.

The Change Variable (CHGVAR) command creates the command (that was within the job that is submitted to the batch job queue) by concatenating (*CAT) constants and a variable into a character string. For example, if FILEA is entered on the menu by the work station user, the CHGVAR command creates the following character string in the variable &CMD.

```
  CALL MLG615C PARM(FILEA)
```

Note that quotes are used to define the constants, including the parentheses that surround the variable. The VAR keyword uses the outermost parentheses to define the limits of the character string.

The CL program to be executed as a result of a response to the menu is submitted for processing within a batch job. It accepts a parameter variable, &FILE, which must also be declared to the program. &FILE is used in the TOFILE keyword of the Override with Data Base File (OVRDBF) command (OVRDBF is discussed under *Mailing List Program* in Chapter 4).

CL Program with Commands Using Numeric and Character Values

Multiple variable values can be passed to a CL program. These may include either or both character variables and numeric variables.

Assume the work station user is prompted for both file name (for example, a character variable named FILE) and date (for example, a 6-digit numeric variable named DATEN). These values must be submitted as parameters to a batch program. The key parts of the CL program that submits the batch job are as follows:

```
.
.
.
DCL &CMD TYPE(*CHAR) LEN(100)
DCL &DATEA TYPE(*CHAR) LEN(6)
.
.
.
CHGVAR &DATEA &DATEN
CHGVAR &CMD VAR('CALL MLGxxxC PARM(' +
   *CAT &FILE *CAT ' ' *CAT &DATEN *CAT ')')
SBMJOB JOB(MLGPRINT) JOBD(MLGLIBDEV) RQSDTA(&CMD)
.
.
.
```

The DATEN and FILE variables would be declared in the display file. The &DATEA variable is declared in the program.

The work station user would key the date into a 6-position numeric field (DATEN). Because the *CAT function only operates on character data, it is necessary to declare a character variable (&DATEA) and then move the numeric date to the alphabetic date variable. This is done by the CHGVAR command.

The command to be submitted is then built up by use of the second CHGVAR command. Note that a blank position must be concatenated between the two variables so that they appear as separate parameters. The actual command that is submitted might look like:

```
CALL MLGxxxC PARM(FILEA 100181)
```

The CL program that executes in batch would be coded as follows:

```
PGM PARM(&FILE &DATEN)
DCL &FILE TYPE(*CHAR) LEN(10)
DCL &DATEN TYPE(*DEC) LEN(15 5)
OVRDBF MLGMSTP TOFILE(&FILE)
CALL MLGxxx PARM(&DATEN)
ENDPGM
```

Note that a numeric variable (any value beginning with a digit is considered a numeric variable) is passed by the system with LEN(15 5) meaning 10 whole numbers and 5 decimal positions. Thus, the date would appear as 0000100181.00000. The CL program must define this value as LEN(15 5) in order to receive the value correctly.

In this example, the date is passed to an RPG program for further processing. The RPG program must define the field with the same definition as was passed. See the *RPG III Reference Manual and Programmer's Guide* for details on how to receive parameters. The system requirement of LEN(15 5) for numeric values is required only for the case where:

- The program is called interactively.

- The program is submitted to batch.

If the CL program is called by another program, the actual definition such as LEN(6 0) could be used. See the *CPF Programmer's Guide* for details on how to pass parameters.

Appendix C. Alphabetic Listing of the Field Reference File

Field	Length	Decimal	Column Heading	Text Description
ACTNUM	5	0	Account Number	Account number
ACTTYP	1	0	Acct Type	Account type 1 = business 2 = government 3 = organization 4 = school 5 = private 9 = other
ADDR	18		Address	Address
BATDAT	6	0	Batch Date	Batch date
BATDSC	35		Batch Description	Batch description
BATNUM	6	0	Batch Number	Batch number
BATSTS	1	0	Batch Status	Batch status 1 = in process 2 = to be continued 3 = ready
CITY	18		City	City
NAME	18		Name	Name
STATE	2		State	State
MLGLK1	3	0	Lock Control	Control number used for record locking
TRNNUM	5	0	Transaction Number	Transaction number

TRNTYP	1		Trans Type	Transaction type
				A = add
				C = change
				D = delete

XACTNM	5	0	Account Number	Account number
XACTTP	1	0	Acct Type	Account type 1 = business 2 = government 3 = organization 4 = school 5 = private 9 = other
XADDR	18		Address	Address
XCITY	18		City	City
XNAME	18		Name	Name
XSTATE	2		State	State
XZIP	5	0	Zip Code	Zip code
ZIP	5	0	Zip Code	Zip code

abnormal termination: System termination by a means other than the successful execution of the Power Down System (PWRDWNSYS) command. See also *system termination*.

access path: The means by which CPF provides a logical organization to the data in a data base file so that the data can be processed by a program. See also *arrival sequence access path* and *keyed sequence access path*.

activity level: An attribute of a storage pool or the system that specifies the maximum number of jobs that can execute concurrently in the storage pool or in the system.

allocate: To assign a resource for use in performing a specific task. Contrast with *deallocate*.

application: (1) A particular data processing task, such as an inventory control application or a payroll application. (2) In IDU, specialized program created by IDU from user input. An application is later called by DFU or the query utility.

application program: A program used to perform a particular data processing task such as inventory control or payroll.

arrival sequence access path: An access path that is based on the order in which records are stored in a physical file. Contrast with *keyed sequence access path*.

attribute: A characteristic; for example, attributes of a field include its length and data type, and attributes of a job include its user name and job date.

attribute character: A character associated with a field in a display file that defines how the field is displayed (such as underlined, blinking, or intensified).

authority: The right to access objects, resources, or functions.

authorization: The process of giving a user either complete or restricted access to an object, resource, or function.

auto dup feature: In DFU, a function that duplicates certain types of information from predetermined fields in a previous record into the current record.

auto report: A function of the RPG III licensed program that uses simplified specifications and standard RPG specifications to generate a complete RPG source program.

auto report option specifications: An RPG coding form the programmer uses to specify options for an auto report program.

autostart job: A job that is automatically initiated when a subsystem is started.

auxiliary storage: All addressable storage other than main storage. Auxiliary storage is located in the system's nonremovable disk enclosures.

basic working display: The display that serves as the base from which you make requests of the system at a work station. When the request is completed, you return to the display. It is usually the display you receive when you sign on.

batch job: A group of processing actions submitted as a predefined series of actions to be performed with little or no interaction between the user and the system.

batch processing: A method of executing a program or a series of programs in which one or more records (a batch) is processed with little or no interaction with the user or operator. Contrast with *interactive processing*.

batch subsystem: A subsystem in which batch jobs are to be processed. IBM supplies one batch subsystem: QBATCH.

branching: The technique of bypassing specific instructions or operations to alter the sequential execution of instructions in a program.

branching instruction: An instruction that changes the sequence of program execution.

breakpoint: A place in a program (specified by a command or a condition) where the system halts execution and gives control to the work station user or to a specified program.

breakpoint program: For a batch job, a user program that can be invoked when a breakpoint is reached.

buffer: A portion of main storage into which data is read or from which it is written.

byte: A group of eight adjacent binary digits that represents one EBCDIC character.

calculation specifications: An RPG coding form on which the programmer describes the processing to be done by the program.

call: (1) To instruct that a program is to begin execution. (2) An instruction to a program to begin execution. (3) In data communications, the action necessary to make a connection between two stations on a switched line.

central processing unit: Abbreviated CPU. See *processor*.

CF key: See *command function key*.

character: Any letter, digit, 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 a particular unit of information and that can contain any of the characters in the data character set. Contrast with *numeric field*.

CL: See *control language*.

class: An object that contains the execution parameters for a routing step. The system-recognized identifier for the object type is *CLS.

close: A data manipulation function that ends the connection between a file and a program. Contrast with *open*.

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 function key: At a work station, a keyboard key that is used with the command (CMD) function control key to request preassigned functions. At the system console, a keyboard key, called a CF key, that is used to request preassigned functions.

comment: A word or statement in a program, command, or file that serves as documentation instead of as instructions. A comment is ignored by a compiler.

compile: To translate a source program into an executable program (an object).

compile time: The time during which a source program is translated by a compiler into an executable program.

compile-time array or table: An array or table in which the data is compiled with the source program and becomes a permanent part of the program. Contrast with *execution-time array* ,and *preexecution-time array or table*.

compiler: A program that translates a source program into an executable program.

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 source listing, a cross-reference list, diagnostic information, and for programs, the description of externally described files.

completion message: A message that conveys completion status of work.

conditioning: (1) In a file, the use of indicators to control when certain functions or operations are to be performed. For example, in a display file, indicators can select fields to be displayed. (2) In an RPG program, the use of indicators to control when certain functions or operations are to be done. For example, in an RPG program indicators can control calculation or output operations.

constant: Data that has an unchanging, predefined value to be used in processing. A constant does not change during the execution of a program, but the contents of a field or variable can. See also *literal*.

constant field: In an externally described display or printer file, an unnamed field that contains actual data that is passed to the display or printer but is unknown to the program passing it.

continuation lines: (1) Additional lines required to continue the coding of a CL command or a DDS keyword and its value. (2) In RPG, additional lines specified on the file description specifications to provide more information about the file being defined.

control language: The set of all commands with which a user requests functions. Abbreviated CL.

control language program: An executable object that is created from source consisting entirely of control language commands.

control language variable: A program variable that is declared in a control language program and is available only to the program.

Control Program Facility: The system support licensed program for System/38. It provides many functions that are fully integrated in the system such as work management, data base data management, job control, message handling, security, programming aids, and service. Abbreviated CPF.

control specification: An RPG coding form on which the programmer provides information that affects program generation and execution.

control statement: In RPG, entries on a control specification.

controlling subsystem: An interactive subsystem that is started automatically when the system is started and through which the system operator controls the system. IBM supplies one controlling subsystem: QCTL.

copy: The SEU operation in which records can be copied to a new location in a source member while the records are retained in their original location.

CPF: See *Control Program Facility.*

CPU: Central processing unit. See *processor.*

create: (1) The function used to bring an object into existence in the system. (2) To bring an object into existence in the system.

cursor: A movable spot of light, resembling a bright underscore, that shows where the next character will appear on the work station screen when a key on the keyboard is pressed.

data area: An object that is used to communicate data such as CL variable values between the programs within a job and between jobs. The system-recognized identifier for the object type is *DTAARA.

data base: The collection of all data base files stored in the system.

data base 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.*

data description specifications: A description of the user's data base or device files that is entered into the system using a fixed-form syntax. The description is then used to create files. Abbreviated DDS.

data file: Any nonsource file. A data file is created by the specification of FILETYPE(*DATA) on a create file command.

data file utility: The utility of the Interactive Data Base Utilities licensed program that is used to create, maintain, and display records in a data base file. Abbreviated DFU.

data type: An attribute used for defining data as numeric or character.

DDS: See *data description specifications.*

deallocate: To release a resource that is assigned to a specific task. Contrast with *allocate.*

debug mode: An environment in which programs can be tested.

default value: A value assumed when no value has been specified.

delay maintenance: A method of maintaining keyed access paths for data base files. This method does not update an access path when the file is closed, but it retains updates in a *delayed* form so that they can be quickly applied at the next open, avoiding a complete rebuild. Contrast with *rebuild maintenance* and *immediate maintenance.*

delete: (1) To remove an object or a unit of data (such as character, a field, or a record). (2) The SEU operation in which existing records can be removed from a source member.

delimiter: A character or a sequence of contiguous characters that identifies the end of a string of characters. A delimiter separates a string of characters from the following string of characters. A delimiter is not part of the string of characters that it delimits.

device file: An object that contains a description of how input data is to be presented to a program from an external device and/or how output data is to be presented to the external device from the program. External devices can be work stations, card devices, printers, diskette magazine drives, magazine tape drives, or a communications link.

DFU: See *data file utility*.

DFU application: See *application*.

digit: Any of the numerals from 0 through 9.

diskette file: A device file created by the user to support a diskette device.

diskette magazine drive: A diskette drive that can hold two magazines, each containing 10 diskettes, plus individual diskettes in three separate slots. It is used to transfer information between system internal storage and removable diskettes.

display: A visual presentation of information on a work station screen, usually in a specific format. Display is often used as a shortened version of information display.

display file: A device file created by the user to support a display work station or console.

display format: The name of the device file and the name of the record format to be used when the subsystem obtains routing data from the user.

display screen: An electronic display tube, similar to a TV picture tube, used to display information entered or received at the system console or a work station.

display station: An input/output device containing a display screen and an attached keyboard that lets a user send information to or receive information from the system.

do group: (1) A set of commands in a control language program delimited by a DO command and an ENDDO command that is conditionally executed as a group. (2) In RPG, a group of calculations that are executed 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 in a readable format from main or auxiliary storage onto an external medium such as tape, diskette, or printer.

edit: (1) To modify a numeric field to an external format by suppressing zeros and inserting commas, periods, currency symbols, the sign status, or other constant information. (2) The process of using SEU to key in new source records and update existing source records in a source member.

edit code: A letter or number indicating what kind of editing should be done before a field is displayed or printed.

edit display: The SEU display from which frequently performed operations, such as delete, copy, and insert, are requested.

edit word: A user-defined word with a specific format that indicates how editing should be done.

embedded blank: A blank that appears between characters.

end position: In RPG, an entry in the output specifications that indicates where the end position of a field or constant is to be placed in the output record.

enter: To press the Enter/Rec Adv key (on a work station keyboard) or the Enter key (on the system console) or a command function key to transfer keyed-in information to the system for processing. See also *key in*.

escape message: A message that can be monitored for and that describes a condition for which a program terminates without completing the requested function.

executable 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 executable program.

execute: To cause a program, command, utility, or other machine function to be performed.

execution: The carrying out of the instructions of a computer program by a processing unit.

execution time: The time during which the instructions of a computer program are executed by a processing unit.

execution-time array: In RPG, an array that is loaded or created by input or calculation specifications after actual execution begins. Contrast with *compile-time array or table* and *preexecution-time array or table.*

extension and line counter specifications: An RPG coding form on which the programmer provides information about record address files, arrays, and tables and their associated files used by a program and about the number of lines to be printed on the printer forms that are used.

external message queue: A message queue that is part of the job message queue and is used to send messages between an interactive job and the work station user. For batch jobs, messages sent to the external message queue only appear in the job log.

external storage: Data storage other than main or auxiliary storage.

externally described data: Data contained in a file for which the fields in the records are described to CPF, by using data description specifications, when the file is created. The field descriptions can be used by the program when the file is processed. Contrast with *program-described data.*

externally described file: A file for which the fields in the records are described to CPF, through data description specifications, when the file is created. The field descriptions can be used by the program when the file is processed. Contrast with *program-described file.*

factor: In RPG, an entry (for example, a field name, file name, literal, or data structure) that identifies the data to be used in a calculation operation.

field: An area that is reserved and used for a particular item of information.

field indicator: In RPG, an indicator used to indicate whether a given field in an input record is plus, minus, zero, or blank.

field record relation indicator: In RPG, an indicator used to associate 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 members and whose record format describes the fields used by a group of files.

file: A generic term for the object type that refers to a data base 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 description: The information contained in the file that describes the file and its contents.

file description specifications: An RPG coding form on which the programmer identifies and describes all files used in a program.

file key: In RPG, all the key fields defined for a file.

file operation code: In RPG, an operation code (for example, CHAIN) that lets the user control the input/output operations to a file.

file overrides: The file attributes specified at execution time that will override the attributes specified in the file description or in the program.

file reference function: A CPF function that lets the user track file usage on the system.

fold: To continue data for a line on the following printed or displayed line. Contrast with *truncate.*

form: The area between perforations on continuous printer paper.

format line: In SEU, the abbreviated names of the fields in the source line that are displayed directly above the source line. The format line is displayed when the F (format) line command is executed.

full procedural file: In RPG, a file for which the input operations are controlled by programmer-specified operation codes instead of by the program cycle. Contrast with *primary file.*

function check: A notification (by a message) that an unexpected condition has stopped the execution of a program.

function key: A keyboard key that is used to request a specific system function. See also *command function key.*

general-purpose library: The library provided by CPF to contain user-oriented, IBM-provided objects and user-created objects that are not explicitly placed in a different library when they are created. Named QGPL.

generic name: The initial characters common to object names that can be used to identify a group of objects. A generic name ends with an * (asterisk). For example, ORD* identifies all objects whose names begin with the characters ORD.

heading: A constant, or field, usually at the top of a page or display, that identifies the information on the page or display.

heading record: In RPG, output records that are generally printed at the top of a report and include report titles, column headings, or any other data needed to identify the information in the report.

help text: Information that is associated with an information display, a menu, or a prompt that explains options or values displayed. Help text is requested by pressing the Help key.

hexadecimal: Pertaining to a numbering system with a base of 16. Valid numbers are the digits 0 through 9 and the characters A through F, where A represents 10 and F represents 15.

high-level language: A programming language that relieves the programmer from the rigors of machine level or assembler level programming; for example, RPG III and COBOL. Abbreviated HLL.

history log: A log of information about system status and events. Named QHST.

HLL: See *high-level language.*

IDU: See *Interactive Data Base Utilities.*

immediate maintenance: A method of maintaining keyed access paths for data base files. This method updates the access path whenever changes are made to the data in the access path. Contrast with *rebuild maintenance* and *delay maintenance.*

indexed file: A data base file whose access path is built on key values. Each record in the file is identified by a key field.

indicator: (1) A 2-character entry on a specification form that is used to test a field or record or to tell when certain operations are to be performed. (2) An internal switch used by a program to remember when a certain event occurs and what to do when the event occurs.

information display: A display that presents information such as the status of the system to a user, but that rarely requests a response.

informational message: A message that conveys information about the normal condition of a function.

initial microprogram load: The process that loads the system microprogram code from the system auxiliary storage, then checks system hardware and prepares system programming for user operations. Abbreviated IMPL.

initial program: A program, specified in a user profile, that is to be executed when the user signs on and the command processor program QCL is invoked. QCL invokes the initial program.

initialize: To set to a starting position or value.

inline data file: A file described by a //DATA command that is included as part of a job when the job is read from an input device by a reader program.

input: Information (or data) to be processed.

input-capable field: Any field that can receive input from a user.

input field: A field in a display file into which data can be entered. An input field is passed from the device to the program when the program reads the record containing that field.

input file: A data base or device file that has been opened with the option to allow records to be read.

input specifications: An RPG coding form on which the programmer describes the records and their fields in a program-described input file, adds RPG functions to an externally described input file, or defines a data structure and its subfields.

input stream: A group of records submitted to the system as batch input that contains CL commands for one or more jobs and/or the data records for one or more inline data files.

inquiry: A request for information from a data file usually made against one record.

inquiry message: A message that conveys information and that requests a reply.

insert: The SEU operation during which source statements are keyed in and added as new records in a source member.

instruction: A statement that specifies an operation to be performed by the system and that identifies the data, if any, involved in that operation.

integrity: The protection of data and programs from inadvertent destruction or alteration.

interactive: Pertaining to a program or system that alternately accepts input and then responds. An interactive system is conversational; that is, a continuous dialog exists between the user and the system.

Interactive Data Base Utilities: A System/38 licensed program that consists of DFU, SEU, query, and SDA. Abbreviated IDU.

interactive job: A job in which the processing actions are performed in response to input provided by a work station user. During a job, a dialog exists between the user and the system.

interactive processing: Pertaining to a program or procedure that alternately accepts input and then responds to the input. Contrast with *batch processing.*

interactive subsystem: A subsystem in which interactive jobs are to be processed. IBM supplies three interactive subsystems: QCTL, QINTER, and QPGMR.

internal storage: All main and auxiliary storage in the system.

invocation: An instance of the execution of a program.

invocation level: Identifies the occurrence of the same program in the job's invocation stack. An invocation level is used in debug mode only. The first occurrence of a program in a job has an invocation level of 1.

invocation stack: A series of invocations linked together as a result of programs invoking other programs.

invoke: To instruct a specific program to start executing. Same as *call.*

job: A single identifiable sequence of processing actions that represents a single use of the system. A job is the basic unit by which work is identified on the system.

job description: An object that contains information defining the attributes of a job. The system-recognized identifier for the object type 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 CPF.

job message queue: A message queue that is created for each job. A job message queue is used for receiving requests to be processed (such as commands) and for sending 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 a job as identified to the system. For an interactive job, the job name is the name of the work station at which the job was initiated; for a batch job, the job name is specified in the command used to submit the job. Contrast with *qualified job name.*

job number: A number assigned to a job as it enters the system to distinguish the job from other jobs.

job priority: The order in which batch jobs on a job queue are selected for execution by CPF. More than one job can have the same priority.

job queue: An object that contains a list of batch jobs submitted to the system for execution and from which the batch jobs are selected for execution by CPF. The system-recognized identifier for the object type is *JOBQ.

key field: A field in a record whose contents are used to sequence the records of a particular type within a file member.

key in: The action of pressing keys on a keyboard to specify information that is to be processed. See also *enter*.

keyed sequence: The order in which records appear in an access path. The access path is based on the contents of one or more key fields contained in the records.

keyed sequence access path: An access path to a data base file that is ordered on the contents of key fields contained in the individual records. Contrast with *arrival sequence access path*.

keyword: (1) A name that identifies a parameter. Keywords are used in CL commands and in DDS. (2) In RPG, a word whose use is essential to the meaning and structure of a statement in a programming language.

label: (1) The name of a file on a diskette or tape. (2) An identifier of a command generally used for branching. (3) In RPG, 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.

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 file record format has changed since the program was compiled.

library: An object that serves as a directory to other objects. A library is used to group related objects and to find objects by name when they are used. The system-recognized identifier for the object type is *LIB.

library list: An ordered list of library names used to find an object. The library list indicates which libraries are to be searched and the order in which they are to be searched. The system-recognized identifier is *LIBL. *LIBL specifies to the system that a job's current library list is to be used to find the object.

licensed program: An IBM-written program that performs functions related to processing user data.

line commands: In SEU, commands (such as D for delete, I for insert, C for copy) that are keyed in the sequence number field of displayed records to request operations on source records.

line counter specifications: An RPG coding form on which the programmer indicates or overrides the system defaults for the length of the printer form and the number of lines to print on a page. Line counter specifications can be used for each printer file in a program.

listing: A printout usually containing the input and output of the compilation of a program, the creation (compilation) of an object, or the execution of a program. See also *compiler listing*.

lock state: The definition of how an object is allocated, how it is used (read or update), and whether the object can be shared (used by more than one job).

logical file: A description of how data is to be presented to or received from a program. This type of data base file contains no data, but it provides an ordering and format for one or more physical files. Contrast with *physical file*.

logical file member: A logical grouping of data records in a logical file. See also *member*.

magazine: A container that holds up to 10 diskettes and is inserted into a diskette magazine drive.

main storage: All storage in a computer from which instructions can be executed directly.

member: A description of a named subset of records in a physical or logical file. Each member conforms to the characteristics of the file and has its own access path. All I/O requests are directed to a specific member of a data base file.

menu: A display in which a list of options is shown.

message: A communication sent from one person or program to another person or program.

message queue: An object on which messages are placed when they are sent to a person or program. The system-recognized identifier for the object type is *MSGQ.

modulus 10 checking/modulus 11 checking: A technique for validity checking that involves the association of digits with data. It is used in entering or updating fields in a data record.

normal termination: System termination that results from the successful execution of the Power Down System (PWRDWNSYS) command. Contrast with *abnormal termination.*

numeric field: An area that is reserved for a particular unit of information and that can contain only the numeric digits 0 through 9. Contrast with *character field.*

object: A named unit that consists of a set of attributes (that describe the object) 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, and libraries.

object authority: The right to use or control an object. See *object rights* and *data rights.*

object name: The name of an object. Contrast with *qualified object name.*

object owner: A user who creates an object or to whom the ownership of an object has been transferred. The object owner has complete control over the object.

object rights: The authority that controls what a system user can do to an entire object. For example, object rights include deleting, moving, or renaming an object. There are three types of object rights: object existence, object management, and operational.

object type: The attributes that define the purpose of an object within the system. Each object type has associated with it a set of commands with which to process that type of object.

object user: A user who has been authorized by the object owner, the security officer, or a user with object existence rights to perform certain functions on an object.

omit function: A CPF function that determines which records from a physical file are to be omitted from a logical file's access path. Contrast with *select function.*

open: The function that connects a file to a program for processing. Contrast with *close.*

operation: A defined action performed on one or more data items, such as adding, multiplying, comparing, or moving information.

operation code: In RPG, a word or abbreviation, specified in the calculation specifications, that identifies an operation.

operator/service panel: A panel located adjacent to the system console on the system unit. This panel contains lights and switches that are used primarily when the system is started or serviced.

output: Data transferred from storage to an output device.

output file: A data base or device file that has been opened with the option to allow records to be written.

output indicator: In RPG, an indicator used to define the conditions under which an output record or an output field in the output specifications is written. An output indicator must be previously defined before it is used in the output specifications.

output priority: The priority used to determine the order in which spooled output files produced by the job are to be written. More than one file can have the same priority.

output queue: An object that contains a list of output files to be written to an output device by a writer. The system-recognized identifier for the object type is *OUTQ.

output specifications: An RPG coding form on which the programmer describes the records and their fields in a program-described output file or adds RPG functions to an externally described output file.

page: (1) A 512-byte block of information that can be transferred between auxiliary storage and main storage. (2) Each group of records in a subfile that are displayed concurrently. (3) One printer form.

page-in: The process of transferring a page from auxiliary storage to main storage.

page-out: The process of transferring a page from main storage to auxiliary storage.

parameter: (1) Data passed to or received from another program. (2) In CPF, an argument that identifies an individual value or group of values to be used by a command to tailor a function requested through the command.

password: A unique string of characters that a system user enters to identify himself to the system.

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 data base. A physical file contains one record format and one or more members. Contrast with *logical file*.

physical file member: A subset of the data records in a physical file. See also *member*.

preexecution-time array or table: In RPG, an array or table that is loaded at the same time as the source program, before actual execution of the program begins. See also *compile-time array or table* and *execution-time array*.

primary file: In RPG, if specified, the main file from which RPG 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*.

printer file: A device file created by the user to support a printer device.

printer/display layout: A coding form on which the programmer can design the format for a printed report or a display.

priority: The relative significance of one job to other jobs in competing for allocation of resources.

problem determination: The process of determining the source of a problem as a component problem, a machine failure, a common carrier link, a user-supplied element, or a user error.

procedural programming: In RPG, a programming technique in which the input and output operations are controlled by programmer-specified operation codes instead of by the program cycle.

processing: The action of performing operations on input data.

processor: The functional unit that interprets and executes instructions. Same as processing unit and *CPU*.

production library: A library containing objects needed for normal processing. Contrast with *test library*.

program: An object that contains a set of instructions that tell a computer where to get input, how to process it, and where to put the results. A program is created as a result of a compilation. The system-recognized identifier for the object type is *PGM.

program cycle: In RPG, a series of steps performed by a compiled RPG program in a specific order for each primary or secondary record that is read.

program data: The data associated with a program.

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 CPF, the record is viewed as a character string. Contrast with *externally described file*.

program message queue: A message queue used to hold messages that are sent between program invocations of a routing step. The program message queue is part of the job message queue.

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 invoked.

prompt: A displayed request for information or user action. The user must respond to allow the program to proceed.

protected field: A field in a display file in which data cannot be keyed, changed, or erased.

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 object name: An object name and the name of the library containing the object. Contrast with *object name.*

query: (1) A utility that is part of the Interactive Data Base Utilities licensed program. (2) A request to extract, from a file, one or more records based upon some combination of data.

query application: See *application.*

queue: A line or list formed by items in the system waiting for service; for example, work to be performed or messages to be displayed.

reader: A program that reads jobs from an input device or a data base file and places them on a job queue.

rebuild maintenance: A method of maintaining keyed access paths for data base files. This method updates the access path only while the file is open, not when the file is closed; the access path is rebuilt when the file is opened. Contrast with *immediate maintenance* and *delay maintenance.*

record: An ordered set of fields that make up a single occurrence of the basic unit of data transferred between a file and a program.

record class: In the query utility, one of the distinct groups into which the query utility classifies records during the preparation of a table.

record format: The definition of how data is structured in the records contained in a file. The definition includes the record name, field names, and field descriptions (such as length and data type). The record formats used in a file are contained in the file's description.

record identification code: In RPG, characters placed in a record to identify that record type.

record identifying indicator: In RPG, an indicator that identifies the record just read.

record type: In RPG, the classification of records in a file. Records of the same type have the same fields in the same order. For program-described files, these records have record identification codes; for externally described files, the records have the same record format name.

recovery: The act of resetting the system, or data stored in the system, to an operable state following damage.

relational operator: In CL, an operator that can be used in an arithmetic, character, or logical relation to indicate the comparison to be performed between the terms in the relation. The relational operators are *EQ or = (equal to), *GT or > (greater than), *LT or < (less than), *GE or >= (greater than or equal to), *LE or <= (less than or equal to), *NE or ¬= (not equal to), *NG or ¬> (not greater than), *NL or ¬< (not less than).

relative end position: In RPG, an entry on the output specifications that indicates the number of blank positions that are to appear between a field or constant and the field or constant defined on the preceding specification line. Contrast with *exact end position.*

request: A CL command, the selection of an option on a menu, or the entering of data that instructs the system to perform a function. A CL command can be entered interactively or in a batch job. A request is identified as RQS on the job log.

request data: Data to be put in a job message queue that is used by a job. For example, a single command or group of commands.

response indicator: A 1-character field passed with an input record from CPF to a program to provide information about the data record or actions taken by the work station user.

restore: To transfer specific objects or libraries from magnetic media such as diskettes or tape to internal storage by reconstructing them in internal storage. Contrast with *save.*

return indicator: In RPG, an indicator used to indicate to the internal RPG logic that control should be returned to the calling program. Abbreviated RT.

right-adjust: To place an entry in a field or to move the contents of a field so that the rightmost character of the data is in the rightmost position of the field.

routing data: A character string that CPF compares with character strings in the subsystem description routing entries to select the routing entry that is to be used to initiate a routing step. Routing data can be provided by a work station user, specified in a command, or provided through the job description for the job.

routing entry: An entry in a subsystem description that specifies the program to be invoked to control a routing step that executes in the subsystem.

routing step: The processing performed as a result of invoking a program specified in a routing entry.

RQS: See *request.*

save: To duplicate specific objects or libraries by transferring them from internal storage to magnetic media such as diskettes or tape. Contrast with *restore.*

scan: The SEU operation in which records are searched for a specified character string or syntax error.

screen design aid: The utility of the Interactive Data Base Utilities licensed program that is used to interactively design, create, and maintain display record formats and menus. Abbreviated SDA.

SDA: See *screen design aid.*

security: The control of access to, or use of, data or functions.

security officer: The individual at an installation who is designated to control the authorization of functions and data in System/38.

select function: A CPF function that determines which records from a physical file are to be selected for a logical file's access path. Contrast with *omit function.*

select/omit field: A field in a logical file record format whose value is compared with a constant, the contents of another field, a range of values, or a list of values to determine if a record is to be omitted from the access path of the logical file or selected for use by the logical file. See also *omit function* and *select function.*

sequence checking: An RPG function that checks the sequence of records in input, update, or combined files used as primary and secondary files.

sequence number: The number of a record that identifies the record within the source member.

sequential-by-key processing: A method of file processing that reads records from a keyed sequence file in the order in which the keys are arranged in the access path.

sequential file: A file in which records are processed in the order that they are stored in the file. of contention and error recovery, and the characteristics of the data stream. Sessions compete for network resources such as the class of service within the path control network. See also *half-session.* **Note**: Each session is uniquely identified in a TH by a pair of network addresses, identifying the origin and destination NAUs of any transmissions exchanged during the session.

SEU: See *source entry utility.*

shared access path: An access path used by more than one file to provide access to data common to the files.

sign off: To enter a command or to select an option from a menu at a work station that instructs the system to end an interactive job.

sign on: To enter a password that identifies the user to the system and instructs the system to establish an interactive job at a work station.

single-level storage: The technique of addressing multiple levels of storage through a single addressing structure.

source entry utility: The utility of the Interactive Data Base Utilities licensed program that is used to create and change source members. Abbreviated SEU.

source file: A file created by the specification of FILETYPE(*SRC). 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 diagnostics. See also *compiler listing.*

source member: A member of a data base source file that contains source statements such as RPG, COBOL, or DDS specifications. See also *member.*

source program: A set of instructions, written in a programming language such as RPG or COBOL, that represents a particular job as defined by a programmer. A source program is used as input to the compiler to create an executable program.

source statement: A statement written in symbols of a programming language. For example, RPG, COBOL, or DDS specifications are source statements.

spooled file: A generic term for three types of files: a device file that provides access to an inline data file or that creates a spooled output file, an inline data file, or a spooled output file.

spooled input file: See *inline data file.*

spooled output file: A device file that causes output data to be saved for later processing by a writer.

spooling: The CPF-provided execution-time support that reads and writes input and output streams on an intermediate device in a format convenient for later processing or output.

spooling subsystem: A subsystem that provides the operating environment needed by the CPF programs that read jobs onto job queues and write files from the output queues. IBM supplies one spooling subsystem: QSPL.

storage pool: A logical segment of main storage reserved for executing a group of jobs.

subfile: A group of records of the same record format that can be displayed concurrently at a work station. The system sends the entire group of records to the work station in a single operation and receives the group in another operation.

subfile record format: One of two record formats required to define a subfile in DDS. The subfile record format defines the fields in a subfile record and is used by the program to perform input, output, and update operations to the subfile.

subroutine: (1) In data communications, a group of statements in a program that can be executed several times in that program. (2) In RPG, a group of calculation specification statements in a program that can be executed several times in that program.

subsystem: An operating environment, defined by a subsystem description, through which CPF coordinates work flow and resource usage.

subsystem description: An object that contains information defining a subsystem and that CPF uses to control the subsystem. The system-recognized identifier for the object type is *SBSD.

syntax checking: A function of the command analyzer, a compiler, or SEU that checks single statements for violations of the rules governing the structure of the statement.

system console: The keyboard and display screen on the system unit that serve as a work station for communicating with and controlling the system. See also *operator/service panel* and *work station.*

system date: The date established for the system when it is started.

system library: The library provided by CPF to contain system-oriented objects provided as part of CPF. Named QSYS.

system operator: The person who operates the system and looks after the peripheral equipment necessary to initiate computer runs or finalize the computer output in the form of completed reports and documents.

system operator message queue: The message queue used by the system operator to receive and reply to messages from the system, work station users, and application programs. Named QSYSOPR.

system termination: The state in which all processing on the system is stopped. Depending on the cause of the termination, system power could be shut off (such as by a power interruption or by entering the Power Down System (PWRDWNSYS) command) or could remain on (such as caused by a machine error condition). See also *abnormal termination* and *normal termination.*

system time: The elapsed time from the point where the system was started to the current time. If the system time is changed to the local time when the system is started, the current system time is the local time of day.

system unit: The main unit of the system, which contains the processing unit, the system console keyboard/display, the operator/service panel, the diskette magazine drive, main storage, auxiliary storage, the work station controller, and the communications subsystem.

system value: A value that contains control information for the operation of certain parts of the system. A user can change the system default value to tailor the system to his working environment. System date and library list are examples of system values.

temporary library: A library that is automatically created for each job to contain temporary objects that are created by that job. The objects in the temporary library are deleted when the job ends. Named QTEMP.

terminal: In data communications, same as *work station.*

termination: The act of putting the system or an element of the system (such as CPF or a subsystem) in the state where it no longer performs its normal function. See also *system termination.*

test library: A library to be used in debug mode and that does not contain objects needed for normal processing. Contrast with *production library.*

timestamp: (1) To apply the current system time. (2) The value on an object that is an indication of the system time at some critical point in the object's history. (3) In query, the identification of the day and time a query report was created that query automatically provides on each report.

transaction: (1) In a batch or remote batch entry, a job or job step. (2) An exchange between a terminal and another device that accomplishes a particular action or result; for example, the entry of a customer's deposit and the updating of the customer's balance. (3) A specific set of input data that triggers the execution of a specific processor job; a message destined for an application program. (4) A unit of processing (consisting of one or more application programs) initiated by a single request. In many cases, the request will originate at a terminal (work station).

user name: The name by which a particular user is known to the system.

user password: A unique string of characters that a system user enters to identify himself to the system.

user profile: An object that contains a description of a particular user or group of users. A user profile contains a list of authorizations to objects and functions. The system-recognized identifier for the object type is *USRPRF.

utility definition specification: A group of source statements, which have the same syntax as CL commands, from which a DFU or query application is created. Abbreviated UDS.

variable: A named modifiable value. The value can be accessed or modified by referring to the name of the variable.

verify: In DFU, a method of checking the accuracy of entered data by entering it twice and comparing the second entry with the first.

virtual storage: The combination of main storage and auxiliary storage, treated as a single addressable unit. Abbreviated VS.

work station: A device that lets a person transmit information to or receive information from a computer as needed to perform his job.

work station message queue: A message queue that is associated with a particular work station and that is used for sending and receiving messages sent to the work station. The name of the message queue is the same as the name of the work station.

work station user: A person who uses a work station to communicate with System/38.

work station user profile: The CPF-supplied user profile that has the authority necessary for work station users. Named QUSER.

working display: See *basic working display.*

writer: A CPF program that writes spooled output files from an output queue to an external device, such as a printer.

Index

IBM System/38
Application Example I

READER'S COMMENT FORM

SC21-7881-1

Please use this form only to identify publication errors or to request changes in publications. Direct any requests for additional publications, technical questions about IBM systems, changes in IBM programming support, and so on, to your IBM representative or to your nearest IBM branch office.

□ If your comment does not need a reply (for example, pointing out a typing error) check this box and do not include your name and address below. If your comment is applicable, we will include it in the next revision of the manual.

□ If you would like a reply, check this box. Be sure to print your name and address below.

Page number(s): Comment(s):

Please contact your nearest IBM branch office to request additional publications.

Name ______________________________

Company or
Organization ______________________________

Address ______________________________

____

City State Zip Code

IBM may use and distribute any of the information you supply in any way it believes appropriate without incurring any obligation whatever. You may, of course, continue to use the information you supply.

No postage necessary if mailed in the U.S.A.

Cut Along Line

Fold and tape　　Please do not staple　　Fold and tape

NO POSTAGE
NECESSARY IF
MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS　　PERMIT NO. 40　　ARMONK, N. Y.

POSTAGE WILL BE PAID BY ADDRESSEE:

IBM CORPORATION
General Systems Division
Development Laboratory
Publications, Dept. 245
Rochester, Minnesota 55901

Fold and tape　　Please do not staple　　Fold and tape

IBM

IBM

IBM System/38 Application Example I (File No. S38-20) Printed in U.S.A. SC21-7881-1